POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es

Jorge Señor Sánchez

05 TRABAJO FIN DE GRADO

INDUSTRIALES

TRABAJO FIN DE GRADO

IMPLEMENTACION DE BLOCKCHAIN EN REDES INTERNET OF THINGS

TRABAJO FIN DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE GRADUADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

SEPTIEMBRE **2023**

Jorge Señor Sánchez

DIRECTOR DEL TRABAJO FIN DE GRADO:
Jorge Portilla Berrueco
Jaime Señor Sánchez

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

"La red es robusta en su simplicidad no estructurada."

– S. Nakamoto

AGRADECIMIENTOS

Con este trabajo finaliza una etapa que ha marcado un antes y un después en mi vida. Todos estos años en la Escuela han sido duros pero he aprendido muchas cosas. Por ejemplo, he aprendido que una caída no significa un fracaso, solo es un paso más hacia el éxito. También he averiguado que no conozco mis límites ya que hay momentos que tuve que superar que pensaba que sería incapaz, y aquí estoy finalizando mi grado después de todo. Considero de igual manera que he adquirido la capacidad de reinventarme, dada la variedad y complejidad de las asignaturas de este grado, y teniendo en cuenta la situación sociosanitaria que hemos vivido entre medias que es la pandemia del Covid-19 y su consecuente confinamiento. Además, me llevo de esta etapa personas que he conocido como son los amigos que me han acompañado día a día en clase y estudiando, y también el recuerdo de profesores que hacían que ciertas asignaturas que pudieran atraerme menos se convirtieran en momentos de interés y también risas.

Quiero hacer una mención especial a mi tutor, Jorge Portilla. Gracias a él me he acercado más al fascinante mundo que es la ingeniería electrónica. Primero le tuve como profesor en la asignatura de microprocesadores de la especialidad de electrónica y ahí ya consiguió confirmar que mi elección de especialidad había sido la correcta. Después, como mi tutor, durante este último año me ha ayudado siempre que he tenido alguna duda, cada día que estaba en el departamento conseguía que me riera con alguna broma y así el día era más ameno, y he sentido su plena confianza en mí para desarrollar este trabajo, detalle que le agradezco enormemente.

También quiero agradecer a todos mis amigos, sean de la Escuela o de fuera de ella. Quiero agradecer a cada uno de ellos que ha estado ahí apoyándome y entendiéndome. Todos aquellos que han sabido comprenderme, como por ejemplo si no les veía durante unas semanas por estar de exámenes y después esperarme con unas cervezas frías al salir del último para celebrar todo el trabajo realizado. Personas que en momentos muy difíciles para mí me han escuchado y dado un abrazo siempre que he necesitado, lo cual me ha permitido seguir luchando día a día y me ha llevado hasta aquí.

Por último, y más importante, quiero expresar mi enorme agradecimiento a mi familia, en especial a mi madre, mi abuelo y mi hermano.

A mi madre agradecerle haber estado siempre ahí y seguir estándolo a día de hoy. No hay palabras para describir lo orgulloso que estoy de ella, ver como ha luchado todos los días por mi hermano y por mí, y como nos ha cuidado y criado para que saliéramos adelante con una educación y unos valores que me considero afortunado de haber recibido.

Por todo esto y mucho más, gracias. Te quiero mamá, sin ti no estaría donde estoy hoy.

A mi abuelo José Luis agradecerle haber estado ahí desde que era niño. Sin duda el mejor profesor que he tenido enseñándome en casa antes de tiempo las tablas de multiplicar, a dividir o incluso la ecuación de segundo grado. No tengo ninguna duda de que eres uno de los grandes culpables de mi facilidad con parte de las matemáticas. Pero además de profesor de matemáticas, para mí has sido un profesor de la vida y siempre te admiraré. Me has enseñado a siempre estar ahí para tus seres queridos, cuidar de ellos e intentar que todo vaya lo mejor posible. Hoy soy quien soy en gran parte por ti. Espero que estés orgulloso de mí, te quiero.

Y también darle las gracias a mi hermano Jaime. Yo de pequeño nunca tuve un gusto por algo en concreto, me gustaban un poco algunas cosas y otro poco otras. Por eso estoy convencido de que si no te hubiera tenido a ti para descubrirme poco a poco el mundo de la ingeniería, hubiera tardado mucho más en descubrir qué me gusta realmente, así que gracias. Gracias también por ayudarme siempre que lo he necesitado con mis dudas en los estudios, tanto cuando estaba en el colegio como durante esta etapa. Siempre has sido un referente para mí. Te quiero hermano.

RESUMEN

En los últimos años, las redes *Internet of Things* se han extendido a muchos entornos de la vida de las personas, como puede ser en su trabajo, en casa o incluso en espacios públicos. Algunos de los posibles usos de este tipo de redes son aplicaciones del ámbito sanitario, de la fabricación o del transporte.

En estos años, también ha crecido mucho el interés por la tecnología *blockchain*. Este tipo de tecnología se caracteriza principalmente por la descentralización que conlleva, ya que se trata de una base de datos distribuida que no controla una única identidad, sino que es compartida y accedida por todos los dispositivos de la red.

En este proyecto se pretende cambiar la estructuración típica por capas de una red *Internet of Things*, formada por *edge*, *cloud* y *fog*, por una *blockchain* con la finalidad de aumentar la seguridad de la red, ya que la aplicación objetivo es del ámbito sanitario donde la seguridad y la privacidad de los datos es de suma importancia.

El trabajo comienza con una introducción sobre qué son las redes *Internet of Things*, qué posibles aplicaciones tienen ya sea en un contexto particular o empresarial y cómo se suelen estructurar. Además, se hace una breve explicación sobre qué es una *blockchain* para entender la razón que hay detrás de los objetivos de este proyecto.

A continuación, se expone una explicación en profundidad sobre *blockchain*, incluyendo los distintos tipos que existen y los principales algoritmos de consenso más utilizados, estudiados para decidir cuál podría ser buena opción para este trabajo. Tras esto, se explican más detalladamente los algoritmos de consenso *Proof of Work* y *Practical Byzantine Fault Tolerance*, que son los algoritmos elegidos para desarrollar los modelos de *blockchain* de este trabajo.

Se decide desarrollar e implementar dos modelos distintos de *blockchain* para analizar el comportamiento de ambos en los dispositivos de *Internet of Things* con los que se va a trabajar en la aplicación objetivo; dichos dispositivos son unos nodos desarrollados en el Centro de Electrónica Industrial denominados Cookies. Por un lado, un modelo trata sobre una *blockchain* pública basada en el algoritmo de consenso *Proof of Work*, y por otro lado, el otro modelo trata sobre una *blockchain* privada basada en el algoritmo de consenso *Practical Byzantine Fault Tolerance*.

Una vez que ya se ha detallado toda la base teórica, se procede a explicar las implementaciones propias de este trabajo. En primer lugar, se desarrolla un protocolo

de sincronización entre los nodos de la red necesario para el funcionamiento de ambos modelos. Tras esto, se comienza explicando el algoritmo específico de la *blockchain* con *Proof of Work* acompañándolo de flujogramas y detallando cómo se programa en el sistema operativo Contiki-NG, que es un sistema operativo orientado específicamente para aplicaciones de *Internet of Things*. A continuación se hace lo propio con la *blockchain* con *Practical Byzantine Fault Tolerance*.

Antes de extraer las conclusiones finales, se detalla cómo se evaluaron las implementaciones propuestas. Primero se explica de manera teórica la manera de calcular el consumo partiendo de las medidas experimentales en el laboratorio. Después, tanto en *Proof of Work* como en *Practical Byzantine Fault Tolerance*, se expone qué medidas se han tomado exactamente en cada caso acompañándolo de gráficas y tablas con datos numéricos. Tras esto, se realiza una comparación entre ambos modelos y se aprecia que el gasto energético es mayor en *Proof of Work* que en *Practical Byzantine Fault Tolerance*.

Después de todo el desarrollo del trabajo, se concluye que la elección de implementar una *blockchain* en una red *Internet of Things* para aumentar la seguridad parece adecuada siempre y cuando se escoja un modelo en el que el consumo energético sea lo menor posible, razón por la que entre los dos modelos de este trabajo se escoge aquel con *Practical Byzantine Fault Tolerance*. Además, se mencionan algunas nuevas líneas de desarrollo a seguir o algunos cambios posibles a realizar en el trabajo en el futuro.

Por último, se muestra la planificación y el presupuesto del proyecto y un pequeño análisis de impactos, incluyendo el aspecto social, económico y medioambiental entre otros.

Palabras clave

Internet of Things, *blockchain*, bloque, algoritmo de consenso, descentralización, *hash*, *Proof of Work*, *Practical Byzantine Fault Tolerance*.

Códigos UNESCO

120308 - Código y Sistemas de Codificación

120349 - Blockchain

120356 - Internet de las Cosas

120361 - Ciberseguridad

ÍNDICE GENERAL

Capítulo 1

INTRODUCCIÓN

A día de hoy, y cada vez más, las personas interactúan continuamente con entornos inteligentes en casa, en el trabajo y en espacios públicos. Esta situación demuestra la necesidad de integración de redes y servicios para ofrecer aplicaciones sin fisuras mientras los usuarios cambian de ambiente y se mueven en sus entornos vitales. Dicha interacción de los usuarios genera enormes cantidades de datos personales, lo que plantea el reto de proteger la privacidad del individuo y la propiedad de los datos. En este contexto, las redes *Internet of Things* (IoT) juegan un papel importante.

1.1. Redes *Internet of Things*

Este tipo de red se refiere al conjunto de dispositivos, vehículos, electrodomésticos y otros objetos físicos dotados de sensores, software y conectividad de red que les permite recopilar y compartir datos entre sí. Estos dispositivos pueden ir desde sistemas para el hogar, como termostatos inteligentes, pasando por sistemas vestibles, como relojes inteligentes, hasta maquinaria industrial compleja y sistemas de transporte.

Una red IoT permite que estos dispositivos inteligentes se comuniquen entre sí y con otros dispositivos con acceso a *Internet*, como teléfonos inteligentes, creando una amplia red de dispositivos interconectados que pueden intercambiar datos y realizar diversas tareas de forma autónoma. Esto puede incluir desde el control de las condiciones ambientales en las granjas hasta la gestión de los patrones de tráfico con coches y otros dispositivos de automoción inteligentes, pasando por el control de máquinas y procesos en fábricas o el seguimiento de inventarios y envíos en almacenes.

Las aplicaciones potenciales de IoT son variadas y su impacto ya se deja sentir en una amplia gama de sectores, como la fabricación, el transporte, la sanidad y la agricultura. A medida que aumente el número de dispositivos conectados a Internet, es probable que IoT desempeñe un papel cada vez más importante en la configuración de nuestro mundo y en la transformación de nuestra forma de vivir, trabajar e interactuar.

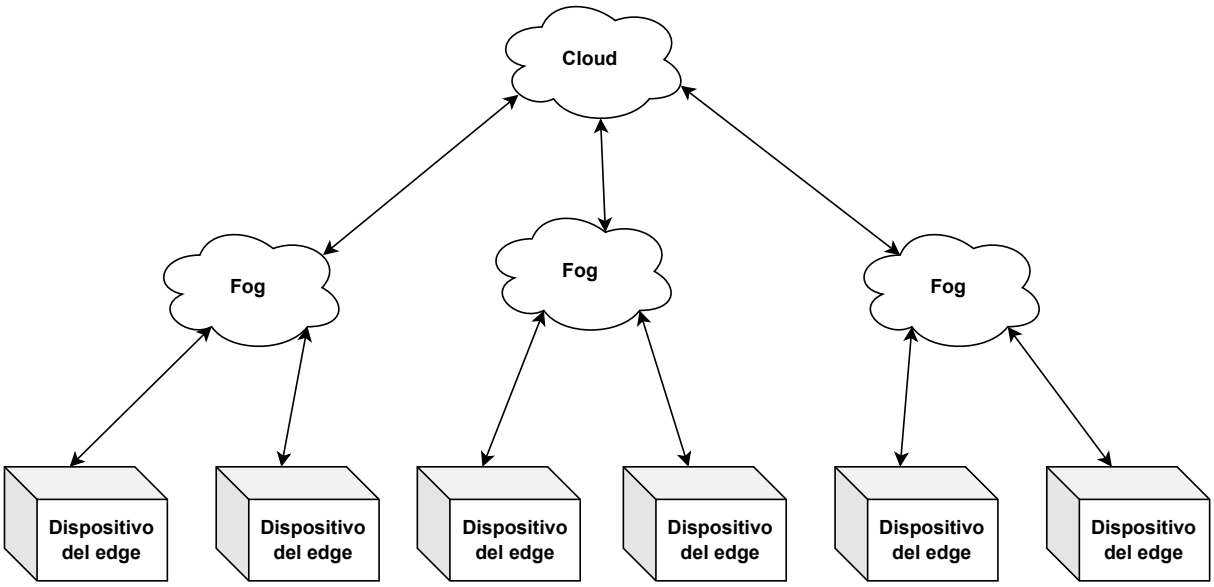


Figura 1.1: Esquema de la estructuración típica por capas de una red IoT

En el contexto empresarial, los dispositivos IoT se utilizan para controlar una amplia variedad de parámetros como la temperatura, la humedad, la calidad del aire, el consumo de energía y el rendimiento de las máquinas. Estos datos pueden analizarse en tiempo real para identificar patrones, tendencias y anomalías que pueden ayudar a las empresas a optimizar sus operaciones y mejorar sus resultados.

Tal y como se expone en [1], una infraestructura de IoT tiene típicamente una disposición por capas como la mostrada en la figura 1.1:

1. La capa inferior, llamada ***edge***, formada por dispositivos que funcionan como nodos de una red, que suele ser inalámbrica. Su función es la interacción directa con el entorno de aplicación del sistema, ya sea obteniendo información a través de sensores o llevando a cabo acciones mediante actuadores.

2. La capa superior, llamada ***cloud***, que es el lugar donde se encuentran servidores con capacidad computacional para realizar el procesado de los datos recogidos por los nodos del *edge*, y donde se alojan otro tipo de aplicaciones como interacción con los usuarios o configuración de los sistemas.

3. La capa intermedia, llamada ***fog***, es el nivel entre los dos anteriores y que generalmente sirve de mediador, recopilando información desde el *edge* para enviarla al *cloud*, adaptando dicha información previo al envío si fuera necesario. Esta capa no tiene por qué existir siempre, depende de cada red y su estructura particular.

En este proyecto, la red IoT que se desarrolla usa nodos Cookie para formar el *edge* de la red. Estas Cookies son dispositivos *hardware* creados específicamente para aplicaciones de IoT por el Centro de Electrónica Industrial de la Universidad Politécnica de Madrid (CEI-UPM), mostrada en la figura 1.2. Dichas Cookies llevan incorporadas un microcontrolador EFR32MG12 [2] de la empresa Silicon Labs cuyo microprocesador es un ARM Cortex-M4 de 32 bits. Además de los elementos de procesamiento, entre

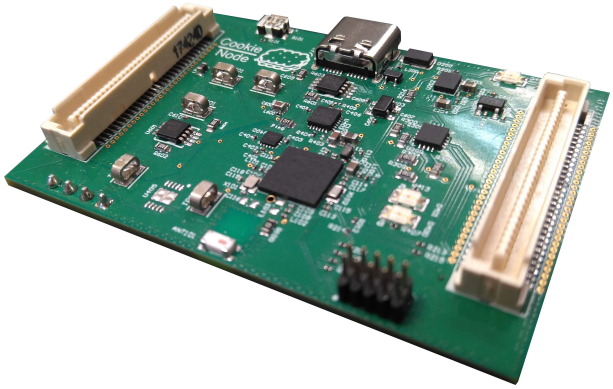


Figura 1.2: Cookie desarrollada por CEI-UPM [4]

los que se encuentra el microcontrolador mencionado, las Cookies también incluyen distintos sensores para interactuar con el entorno físico y una parte de alimentación para proporcionar la tensión necesaria para que el dispositivo funcione. Todo lo anterior se refiere al *hardware* de la Cookie; en cuanto al *software*, las placas estarán programadas con el sistema operativo Contiki-NG [3], creado específicamente para dispositivos IoT con recursos limitados, y el código específico de este trabajo se implementará en el lenguaje de programación C.

Por otra parte, en este trabajo se opta por sustituir la capa del *cloud*, un sistema de almacenamiento central, por una *blockchain*, un sistema de almacenamiento descentralizado y distribuido entre todos los nodos del *edge*.

1.2. *Blockchain*

Como se ha dicho hace un momento, una *blockchain* es un sistema de almacenamiento descentralizado y distribuido entre los nodos de la red, es decir, los datos no se guardan en una única ubicación, como puede ocurrir en una red IoT con la capa del *cloud* si tiene la estructuración típica comentada anteriormente, ni son controlados por una única entidad.

Cada bloque de datos que forma la cadena está conectado con el anterior para formar dicha cadena. Este detalle es muy importante ya que, junto al hecho de tratarse de un único registro pero con copias sincronizadas en todos los nodos, no se pueden eliminar o manipular bloques, lo que aporta gran fiabilidad y seguridad a los datos que almacena.

Debido a todo esto, resulta interesante implementar una *blockchain* en una red IoT para proporcionar mayor seguridad a los datos al pasar de un sistema de almacenamiento central a uno descentralizado.

1.3. Objetivos del proyecto

El objetivo central de este proyecto es el desarrollo de una aplicación para el sector sanitario donde la privacidad de los datos personales de cada usuario es de extrema importancia. Dicha aplicación estará basada en una red IoT en la que se opta por la implementación de una *blockchain* en sustitución del *cloud* de la estructuración típica de IoT, para proporcionar la seguridad necesaria a esos datos. Esta decisión se debe a la seguridad que ofrece la descentralización que la tecnología *blockchain* conlleva.

En este trabajo se probarán dos modelos de *blockchain*, ambos explicados más adelante, para analizar el comportamiento de cada uno en nuestro tipo de aplicación objetivo. Para realizar dicho análisis, se tomarán medidas de consumo de energía y tiempo y se realizarán las deducciones y cálculos necesarios. El primer modelo trata de una red pública, en la que cualquiera puede participar, basada en el algoritmo de consenso *Proof of Work*, y el segundo modelo es una red privada, en la que es necesario autenticarse para acceder a ella, basada en el método de consenso *Practical Byzantine Fault Tolerance*.

Capítulo 2

BLOCKCHAIN

Una *blockchain*, o cadena de bloques, es una base de datos distribuida y a prueba de manipulaciones que no controla una única identidad, sino que es compartida y accedida por todos los dispositivos de la red. Se pueden añadir nuevos registros, llamados bloques, a la cadena siempre y cuando el nuevo bloque sea aprobado por el resto de miembros de la red y, además, una vez registrados los bloques no es posible modificarlos o borrarlos. En cada bloque se guarda el *hash* del bloque anterior, relacionando así un bloque con el anterior y el siguiente y, por lo tanto, formando la cadena deseada. [5]

2.1. Función *hash*

Una función *hash* es un algoritmo matemático que transforma un conjunto de datos de entrada en un código único de tamaño fijo, aunque hay algunas funciones que admiten salidas de longitud variable. Los valores devueltos por estas funciones se denominan *digest*, aunque habitualmente se les denomina también como *hash*.

Las funciones *hash* se utilizan en aplicaciones de almacenamiento y recuperación de datos para acceder a éstos en un tiempo pequeño y prácticamente constante por recuperación, ya que es una forma de acceso a los datos eficiente desde el punto de vista computacional y del espacio de almacenamiento.

Esta transformación de los archivos es unidireccional, es decir, no es reversible, por lo que un usuario puede conocer un *hash* pero no los datos reales a partir de dicho *hash*. Hay distintos tipos de funciones en función del tamaño, siendo muy utilizado el SHA-256, un algoritmo de *hash* de longitud fija de 256 bits que se usa para seguridad criptográfica al generar *hashes* irreversibles. A través del uso de estos algoritmos seguros, sofisticados y que requieren muchos cálculos, se logra la integridad de los datos, lo que impide que se borren o manipulen y que no se registre información no válida.

En este trabajo, todo *hash* utilizado es un SHA-256 debido a que es el más usado en las aplicaciones de *blockchain*.

2.2. Tipos de *blockchain* y algoritmos de consenso

A continuación, se explican los tipos de *blockchain* y de algoritmos de consenso, todo ello basado en la información expuesta en [6].

En función de como se unan los participantes a la red, se pueden distinguir dos tipos fundamentales de *blockchain*: públicas y privadas.

1. Las *blockchain* **públicas**, como su nombre indica, son públicas, es decir, cualquiera puede acceder a la red desde donde quiera. En este tipo de redes cualquiera puede ver los mensajes que se mandan y leer, escribir y verificar la *blockchain*. Los ejemplos más conocidos de este tipo de *blockchain* son Bitcoin y Ethereum.

2. Las *blockchain* **privadas** requieren a los participantes autenticarse antes de poder acceder a la red. Las redes con este tipo de *blockchain* suelen ser más pequeñas que las públicas y las empresas suelen usarlas en aplicaciones donde prima el control y la privacidad de los datos frente a otros aspectos técnicos.

Ambos tipos de *blockchain* se basan en algún algoritmo de consenso, el cual es un mecanismo que permite a los distintos dispositivos coordinarse y tomar decisiones bajo un mismo criterio, para determinar si un nuevo bloque es válido o detectar anomalías. Por lo tanto, un sistema basado en *blockchain* es tan seguro y robusto como su método de consenso subyacente.

Existe una gran cantidad de algoritmos de consenso, por lo que, a continuación, se explican los métodos más comunes hasta el momento:

1. El algoritmo de consenso ***Proof of Work*** (PoW) se basa en el principio de que cuanto mayor sea el poder computacional de un nodo, mayores serán sus posibilidades de generar bloques. Este método implica resolver un problema matemático mediante prueba y error, lo que conlleva un alto coste computacional para añadir nuevos bloques a la cadena. Este proceso se conoce coloquialmente como minería y los nodos de la red que se dedican a ello son los mineros. El primer participante que encuentre una solución distribuye a la red dicha solución junto con el bloque que quiere añadir para que el resto la valide y se incluya dicho bloque en la cadena. En concreto, ese problema matemático consiste en hallar un número, llamado *nonce*, con el cual al hacer un *hash* al conjunto bloque-*nonce* se obtiene un determinado número de ceros seguidos en dicho *hash*; la cantidad de ceros necesaria es acordada por cada red, aumentando la dificultad según aumenta dicha cantidad.

2. El algoritmo de consenso ***Proof of Stake*** (PoS) se basa en el principio de que cuanto mayor sea la participación de un nodo en la red, tendrá mayores posibilidades y legitimidad para validar bloques. Con este método, los nodos actúan como validadores en vez de como mineros, creando el bloque simplemente con la información que se quiere guardar y validando a través de la confirmación de que dicha información es correcta. Los nodos son seleccionados aleatoriamente para validar bloques. La probabilidad de que un nodo sea seleccionado depende del

número de monedas que posea; hablamos de monedas ya que se suele usar este algoritmo con criptomonedas como actualmente Ethereum. Los validadores deben hacer un depósito de seguridad, siendo más probable su selección cuanto mayor sea su depósito. Por ejemplo, un nodo que depositara dos monedas tendría el doble de opciones de ser elegido frente a un nodo que deposite una moneda. Tras la selección del validador y la validación del bloque por parte de éste, la red vota para decidir si añadir dicho bloque o no.

3. El algoritmo de consenso ***Delegated Proof of Stake*** (DPoS) se basa en el princicpio de que cuanto mayor se la participación de un nodo en la red, más votos podrá delegar para que otro nodo fiable haga de validador. Este método es igual que PoS con la diferencia de que en vez de intentar simultáneamente todos los nodos ser elegidos validadores, solo lo intentan algunos de ellos siendo votados éstos por otros nodos.

4. El algoritmo de consenso ***Practical Byzantine Fault Tolerance*** (PBFT) se basa en el problema de los generales bizantinos, en el cual éstos tienen que llegar a un consenso sobre la estrategia de ataque con la suposición de que algunos generales pueden ser traidores e intentar adoptar acciones para evitar que los generales leales lleguen a un consenso y conquistar la ciudad. En este método, todos los nodos deben participar en el proceso de votación para añadir el siguiente bloque y el consenso se alcanza cuando más de dos tercios de la red está de acuerdo con ese bloque. Por lo tanto, en este mecanismo el número máximo de nodos que pueden fallar en el proceso es $f = (n-1)/3$ siendo n el número total de nodos de la red. Esta capacidad de tolerancia proporciona un nivel de seguridad adecuada para cualquier red. Todo esto se debe a que en cualquier sistema tolerante a fallos, los mensajes pueden sufrir pérdidas, corrupción, un alto nivel de latencia y repetición. Además, el orden de transmisión puede no coincidir con el orden de recepción de mensajes. Las actividades de los nodos también son impredecibles, ya que los nodos pueden entrar o salir de la red en cualquier momento o pueden perder información, falsificarla o simplemente dejar de funcionar. Aquí el consenso se alcanza de forma más rápida y económica en comparación con PoW. PBFT tiene un alto rendimiento y baja sobrecarga computacional, ambas deseables para redes IoT, pero, sin embargo, su elevada sobrecarga de la red hace que no sea escalable para grandes redes, por lo que solo podría aplicarse a redes pequeñas. Es por esto que este algoritmo es adecuado para *blockchain* privadas, en las que se puede controlar el tamaño de la red. En cambio, para públicas no es la mejor opción debido a su limitada escalabilidad.

Este trabajo se centra en los algoritmos de consenso PoW y PBFT. La elección de PoW se debe a que es el mecanismo del que más información había en el momento de empezar el trabajo y así se podía conseguir una visión más completa para poder afrontar los problemas que pudieran surgir a la hora de desarrollar el trabajo. Tras desarrollar por completo PoW y observar su comportamiento en nuestro sistema, se decidió optar por PBFT buscando directamente un algoritmo que, en principio, sea adaptable a las características de nuestro sistema.

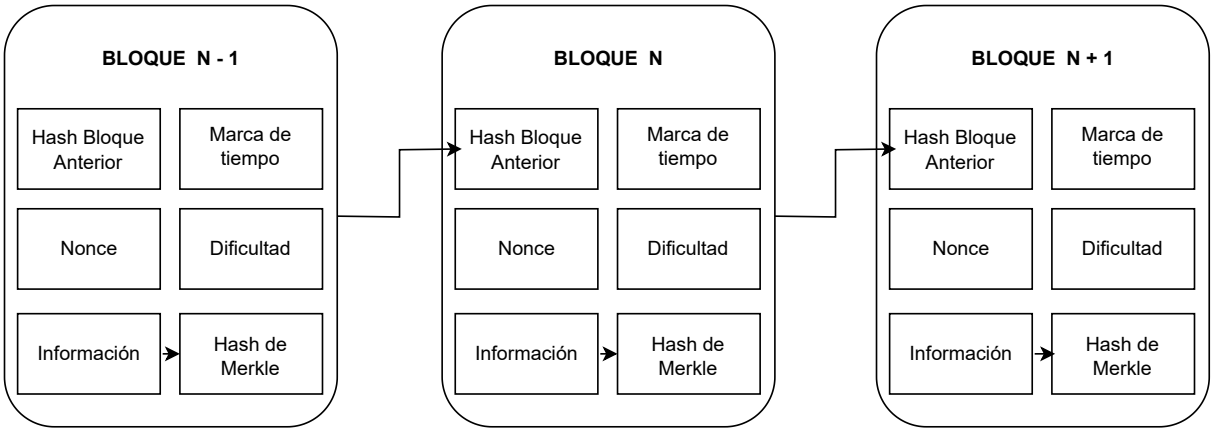


Figura 2.1: Esquema de *blockchain* con PoW

2.3. Algoritmo de consenso PoW

En esta sección se detalla en mayor profundidad cuál es el funcionamiento del algoritmo de consenso *Proof of Work*.

Antes de detallar el procedimiento completo de la formación de un bloque, es necesario saber de que se componen los bloques. En la figura 2.1 se muestra un esquema de cómo es un bloque en PoW. En dicho esquema se observa que un bloque está formado por el ***hash* del bloque anterior**, para relacionar los bloques y formar la cadena, una **marca de tiempo**, que indica en que momento se ha creado el bloque, el ***nonce***, la **dificultad**, que es el número de ceros que tiene que cumplir el *hash* del bloque para resolver el problema matemático mencionado anteriormente, el ***hash* de Merkle**, que es un *hash* que representa la información que se quiere guardar, y dicha **información** que se quiere guardar en el bloque.

Se ha mencionado que cada bloque guarda el *hash* del bloque anterior para formar la cadena pero el primer bloque no tiene un bloque anterior. Para este primer bloque lo que se hace es crear antes un bloque origen, llamado **bloque génesis**, que contiene un *hash* que decide cada red al inicio de su funcionamiento para que todos tengan el mismo.

Para obtener el *hash* de Merkle se sigue el procedimiento mostrado en el esquema de la figura 2.2. En primer lugar, se realiza el *hash* (*Hash 0-0 .... Hash 1-1*) de cada grupo de datos que se quiera guardar en el bloque (L1 .... L4). Tras esto, se agrupan en parejas los *hashes* obtenidos y se hace el *hash* de cada pareja que se forme (*Hash 0 y Hash 1*). Una vez llegado al nivel en el que solo queda una pareja de *hashes*, se obtiene el *hash* de dicha pareja que es el *hash* de Merkle. En el ejemplo se muestra la formación de esta variable con cuatro grupos de datos iniciales, pero no siempre tiene que ser así. Tanto el número de grupos de datos como el tamaño de los mismos puede ser cualquiera, con el inconveniente de que cuanto mayor sea el número de grupos más *hashes* habrá que realizar y, por tanto, mayor será el consumo de tiempo y energía.

Como ya se ha expuesto, la dificultad es el número de ceros seguidos que decide la red que tienen que tener los *hashes* de los bloques al calcular el *nonce*. Cuánto mayor

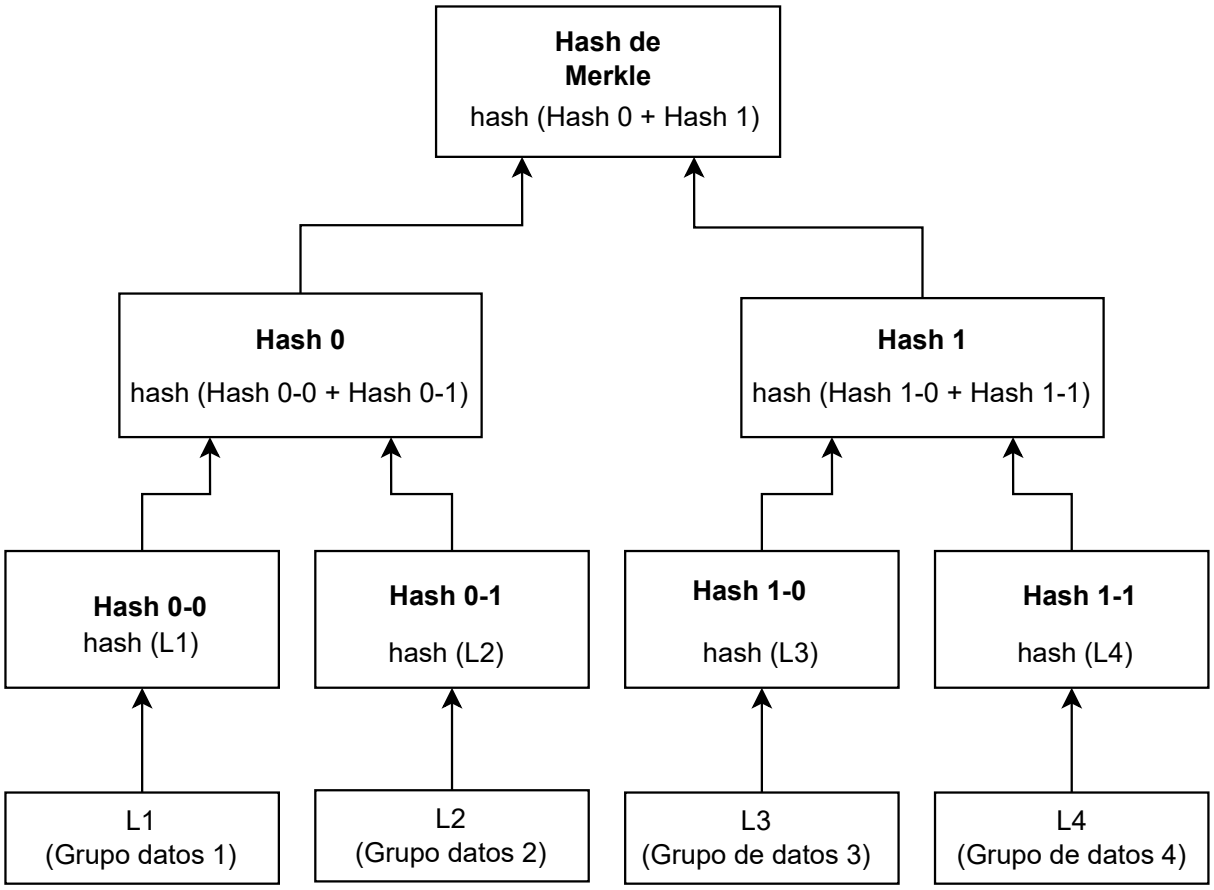


Figura 2.2: Esquema de generación de *hash* de Merkle

sea la dificultad, más complicado será hallar el *nonce* y, por lo tanto, minar un bloque. Este aumento de dificultad conlleva un aumento de gasto computacional al tener que realizar más operaciones para resolver el problema. Debido a esto, un aumento de la dificultad también conlleva un aumento de seguridad de la red ya que los posibles atacantes necesitarían mayor capacidad computacional también.

Por lo tanto, para formar un bloque y enviarlo a la red, el nodo tiene que obtener la información que quiere guardar en dicho bloque, calcular el *hash* de Merkle y guardar ambos datos en el bloque junto a la dificultad y el *hash* del bloque anterior. Una vez ya se ha guardado todo lo mencionado en el bloque, se procede a calcular por prueba y error el *nonce* y, una vez hallado, se guarda junto a la marca de tiempo en el bloque.

Tras ver como un nodo crea un bloque, es el momento de saber cómo validan el resto de nodos ese bloque creado por otro. Para decidir si un bloque es válido o no, lo único que tienen que hacer los nodos que lo reciben es calcular el *hash* de dicho bloque y comprobar que cumple las condiciones acordadas por la red (la dificultad) y que coincide con el *hash* calculado por el nodo que ha creado el bloque.

Hasta el momento, en esta sección se ha hablado continuamente de lo que realiza un solo nodo pero la realidad es que están todos los nodos de la red haciendo lo mismo a la vez, es decir, todos los nodos están intentando minar un bloque simultáneamente. Por eso, cuando un nodo mina un bloque y lo manda a la red, el resto de nodos detienen lo

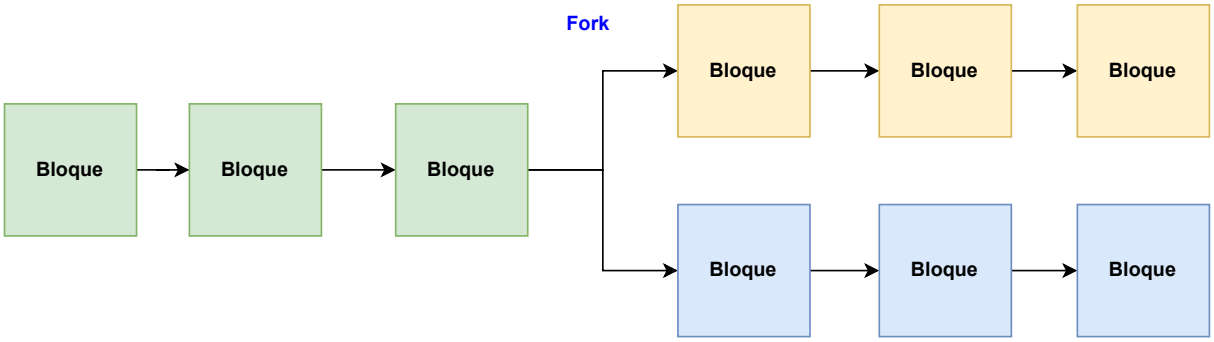


Figura 2.3: Esquema de un *fork* en *blockchain*

que estén haciendo, validan o no dicho bloque, actualizan la *blockchain* si la validación es exitosa y vuelven a intentar minar su bloque.

Debido a este funcionamiento simultáneo de los nodos de la red, también es posible que ocasionalmente dos nodos creen un bloque a la vez o que, por retardos en la comunicación, ambos bloques lleguen al resto de nodos al mismo tiempo y se produzca un *fork*. Un ***fork***, cuyo esquema se muestra en la figura 2.3, es una bifurcación de la *blockchain* en dos cadenas igual de válidas al generarse dos bloques a la vez y algunos nodos guardar primero un bloque y otros nodos guardar primero el otro. Para solucionar este problema se decide que la cadena que prevalecerá será la que crezca más rápido. En ese instante en el que una cadena crezca más rápido que la otra, los nodos que tuvieran la cadena que crece más lento tienen que sustituirla por la otra ya que se convertirá en la *blockchain* oficial de la red. Para poder hacer este cambio, los nodos tienen que guardar también la cadena que no les corresponde según se generan ambas cadenas. Este hecho supone un gran consumo de memoria y, por tanto, un gran inconveniente para nuestro tipo de aplicación, en la cual los dispositivos de la red tienen una memoria limitada de tamaño reducido.

2.4. Algoritmo de consenso PBFT

En esta sección se detalla en mayor profundidad cuál es el funcionamiento del algoritmo de consenso *Practical Byzantine Fault Tolerance*.

Es conveniente comentar qué campos componen los bloques en este tipo de *blockchain* para poder entender más tarde el funcionamiento de dicha *blockchain*. En la figura 2.4 se muestra un esquema de cómo es un bloque en PBFT. En ese esquema se aprecia que un bloque está formado por el ***hash* del bloque anterior**, para relacionar los bloques y formar la cadena, una **marca de tiempo**, que indica en que momento se ha creado el bloque, la **información** que se desea guardar en el bloque, el ***hash* del bloque**, que se obtiene realizando el *hash* de los tres campos mencionados anteriormente, un **número de secuencia**, que identifica el número de bloque que representa en la cadena, y la **dirección del creador** de dicho bloque.

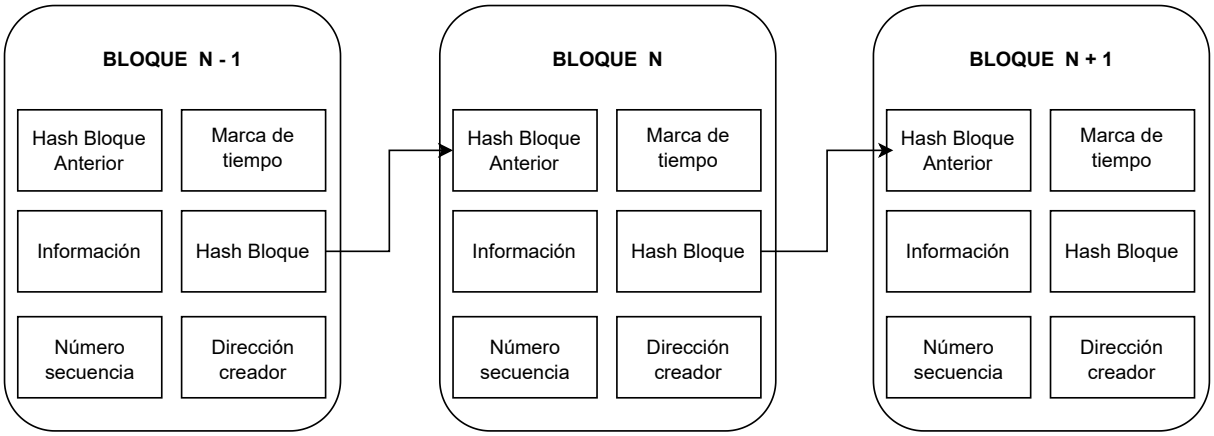


Figura 2.4: Esquema de *blockchain* con PBFT

En este caso, ocurre lo mismo que en PoW con el primer bloque y el *hash* del bloque anterior al no existir dicho bloque anterior. Por lo tanto, se procede de igual manera creando un bloque génesis en el cual se guarda un *hash* que decide la red al inicio de su funcionamiento y, además, se le asigna valor cero a su campo de número de secuencia. De esta manera, no solo tendrá un *hash* previo el primer bloque, sino que también su número de secuencia valdrá uno como debe suceder.

Por lo tanto, para formar un bloque y enviarlo a la red, el nodo tiene que guardar en el bloque la información que quiere guardar en dicho bloque junto al *hash* del bloque anterior y la marca de tiempo, y realizar un *hash* a todos eso datos y guardar dicho *hash* en el bloque también. Tras esto, simplemente hay que guardar en el bloque el número de secuencia que corresponda y la dirección del propio nodo que está creando el bloque.

Dado que en este proyecto se ha desarrollado una versión modificada del algoritmo original de PBFT [7], se explicará dicha versión. En esta versión, que funciona en una red privada, un nodo crea su bloque y lo manda al resto del nodos, para que valoren si los campos del número de secuencia y del *hash* del bloque anterior son correctos. No hace falta comprobar el resto de datos del bloque ya que al ser una red privada y tener que autenticarse para acceder, se confía en todos los participantes. Tras hacer cada nodo su valoración, mandan dicha valoración al resto para que confirmen si es correcta. Ya por último, se le manda al nodo que creó el bloque las confirmaciones o no confirmaciones de las valoraciones iniciales, y el número total de valoraciones positivas confirmadas tiene que ser superior a dos tercios de la red para añadirlo a la cadena. El esquema de esta versión, mostrado en la figura 2.5, muestra una red de cuatro nodos para que sea fácil de entender, pero la red puede ser del tamaño que se quiera. Por ejemplo, si nos centramos en el nodo azul del esquema, éste recibe el bloque del nodo creador y realiza su valoración. Tras realizar dicha valoración, la envía a los nodos verde y rojo, y recibe las valoraciones de dichos nodos. Entonces, confirma o no las valoraciones de los nodos verde y rojo, y le manda al nodo creador el número de confirmaciones de valoraciones para que este último haga el recuento final y compruebe si se puede añadir o no el bloque a la cadena.

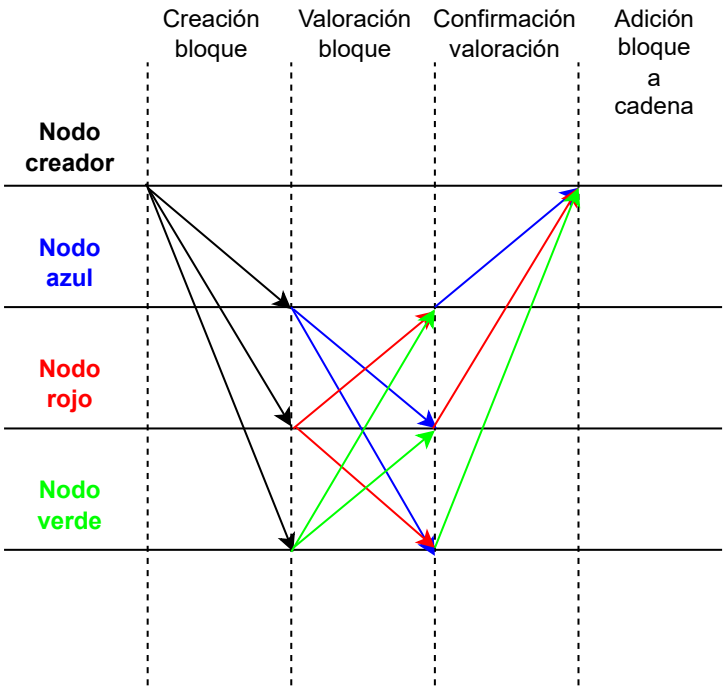


Figura 2.5: Esquema de versión de *blockchain* con PBFT modificada

Capítulo 3

IMPLEMENTACIONES DE BLOCKCHAIN

En este capítulo se explican las implementaciones de los dos tipos de *blockchain* desarrollados en este trabajo.

3.1. Sincronización temporal entre nodos

Antes de empezar todo el proceso propio de la *blockchain*, es necesario que los relojes de todos los nodos de la red marquen el mismo tiempo. Para lograr esto hay dos opciones, o conectarse a la hora de la zona a través de *Internet* o sincronizar los relojes de todos los nodos de manera local en la red. En este proyecto se ha optado por la segunda opción siguiendo el siguiente protocolo de sincronización [8], cuyo esquema puede observarse en la figura 3.1. Esta elección se debe a que los nodos no estaban configurados para conectarse a *Internet* en ese momento y, para estas pruebas, resultaba más sencillo realizar el protocolo que configurar dicha conexión a *Internet*. Para poder sincronizar los nodos se elige un nodo de referencia, al que llamaremos *root*, para que el resto se sincronice con él y, por tanto, el resto también estarán sincronizados entre sí. Teniendo esto en cuenta, se procede a explicar la sincronización entre dos nodos, un nodo N y el *root*. En primer lugar, el nodo N que quiere sincronizarse con el *root* le manda una solicitud a este último y registra el tiempo t_1 en el que lo manda según su hora local. Cuando el *root* recibe la solicitud registra el tiempo t_2 en el que llega según su hora local y le manda respuesta al nodo N, registrando el tiempo t_3 en el que lo manda según su hora local. Por último, el nodo N registra el tiempo t_4, según su hora local, en el que llega la respuesta y, a continuación, realiza los cálculos necesarios para sincronizarse con el *root*. Dichos cálculos son los siguientes:

Primero se relaciona el tiempo local del *root* (T) con el tiempo local del nodo N (t) según la siguiente expresión:

$$T = t + \delta + d \tag{3.1}$$

siendo δ la desviación relativa del reloj y d el retardo en la propagación de un mensaje.

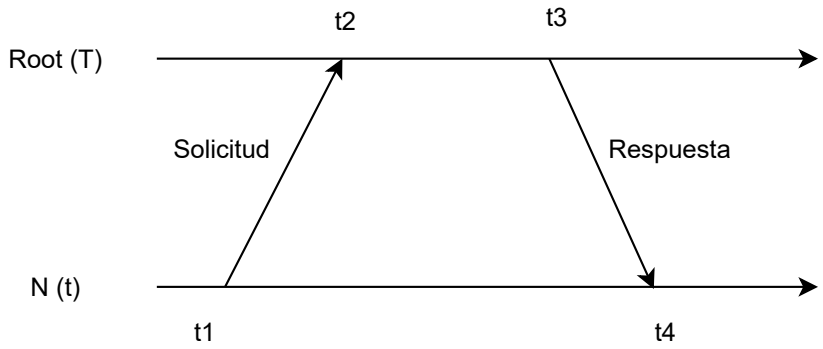


Figura 3.1: Esquema de protocolo de sincronización de dos nodos

A continuación, se muestran las ecuaciones para calcular los valores de δ y d:

$$\delta = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{3.2}$$

$$d = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{3.3}$$

Sabiendo todo esto, solo queda calcular el nuevo tiempo local del nodo N (t'), sincronizado con el *root*, usando la primera expresión y quedando:

$$t' = t + \delta + d \tag{3.4}$$

3.2. Implementación de *blockchain* basada en *Proof of Work*

En esta sección se explica el desarrollo de la implementación de la *blockchain* usando el algoritmo de consenso PoW.

Como se ha explicado en la sección anterior, antes de iniciar el proceso propiamente dicho de la *blockchain*, es necesario que los relojes de todos los nodos de la red marquen el mismo tiempo. Para ello se sincronizan los relojes de los nodos de manera local siguiendo el protocolo de sincronización descrito en dicha sección.

En esta *blockchain* se ha decidido que el *hash* del bloque génesis, mencionado y explicado en la sección 2.3, sea resultado de aplicar dicha función *hash* al conjunto de un número escogido por el programador y otro número aleatorio elegido mediante prueba y error, para que el resultado tenga el número de ceros que indica la dificultad acordada por la red, un problema análogo al que se resuelve para obtener el *nonce*. Dado que en esta *blockchain* no existe un orden predefinido para minar un bloque y están todos intentándolo a la vez, cada nodo crea su propio bloque génesis. Cuando un nodo consigue minar el primer bloque y que se añada a la cadena, el resto actualiza su bloque génesis al recibir ese primer bloque con el *hash* del bloque génesis almacenado.

Habiendo sentado ya las bases sobre la estructura de esta *blockchain*, es momento de explicar cómo se crea un bloque para enviarlo a la red y que el resto de nodos lo validen, tal y como se muestra en el flujograma de la figura 3.2:

1. En primer lugar, lo que tiene que hacer todo nodo al iniciar el sistema, además de sincronizarse con el resto, es crear el bloque génesis como se ha explicado previamente.

2. El siguiente paso consiste en, tras saber la información que se quiere guardar en el bloque, generar el *hash* de Merkle siguiendo el procedimiento descrito anteriormente. Para las pruebas de este trabajo, la información a guardar en el bloque han sido cuatro temperaturas medidas en el momento con los sensores de la Cookie.

3. Una vez ya se tiene el *hash* de Merkle se guarda en el bloque dicho *hash*, la información de la que proviene éste, la dificultad acordada por la red y el *hash* del bloque anterior, ya sea el *hash* del bloque génesis o de un bloque añadido a la cadena en otro momento según corresponda.

4. Por último, antes de enviar el bloque a la red, se procede a calcular el *nonce*. Para ello, se le asigna un número aleatorio a dicha variable, se guarda el tiempo en el que se encuentra en ese instante y se realiza el *hash* de todo el bloque. Se repite este proceso continuamente hasta que el *hash* tiene al principio un número de ceros seguidos igual al número que marca la dificultad. En ese momento, se guarda en el bloque el *nonce* y la marca de tiempo en que se ha generado, ya que esta marca también es el tiempo en el que se ha creado el bloque.

5. Ya teniendo el bloque completo, se procede al envío al resto de nodos para que lo validen. Si la respuesta es positiva, se agrega dicho bloque a la cadena y se vuelve al paso en el que se obtiene la información que se querrá guardar en el siguiente bloque. En cambio, si la respuesta es negativa se vuelve al momento del cálculo del *nonce* para crear un nuevo bloque con la información que se quería guardar en el anterior intento.

Ahora es momento de explicar en qué se traduce el flujograma desarrollado al programarlo en Contiki-NG con el lenguaje de programación C:

1. En primera instancia, al iniciarse el sistema, se inicia un proceso en Contiki en el que se configuran todos los elementos de la Cookie necesarios para el funcionamiento propio y la comunicación con el resto de nodos, como puede ser la activación de sensores o el inicio de una pila de datos tipo LIFO donde se guardarán los bloques y que será la propia *blockchain*. También, se inician el resto de procesos en el orden que corresponda.

2. La sincronización de los relojes de los nodos corresponde a un proceso en Contiki en el que, en primer lugar, el nodo que quiere sincronizarse intercambia mensajes con el nodo de referencia para conocer sus tiempos y, posteriormente, realiza todos los cálculos descritos en el protocolo de sincronización desarrollado anteriormente.

3. Para la creación del bloque génesis y de los posteriores bloques se usa el mismo proceso en Contiki. La primera vez que se ejecuta el proceso es cuando se crea el *hash* del bloque génesis como se ha explicado antes y, tras esto, el proceso se pausa quedando a la espera de la información que se quiere guardar en el siguiente bloque. Una vez se tiene dicha información el proceso se reanuda y se calcula el *hash* de Merkle como se explicó en la sección 2.3. A continuación, se guardan todos los datos necesarios en el bloque y se calcula el *nonce* y se guarda también en el bloque junto a la marca de tiempo. Al final, el proceso queda a la espera de tener nueva información en el futuro para intentar minar otro bloque.

4. El envío del bloque al resto de la red corresponde a otro proceso en Contiki que está siempre a la espera de que se cree un nuevo bloque para enviarlo al resto de nodos para que lo validen.

5. En cuanto a la validación del bloque por el resto de la red, dicha acción corresponde a un proceso diferente en Contiki en el que se realiza la comprobación explicada en la sección 2.3.

6. La adición del bloque a la cadena en el caso de que sea validado, es decir, el guardado de los datos en la pila LIFO, la realiza cada nodo cuando confirma toda la red que es válido el bloque.

3.3. Implementación de *blockchain* basada en *Practical Byzantine Fault Tolerance*

En esta sección se explica el desarrollo de la implementación de la *blockchain* usando el algoritmo de consenso PBFT.

Como se menciona en los objetivos del proyecto expuestos en el capítulo introductorio, la red que funcionará con este tipo de *blockchain* será una red privada en la que los nodos tienen que autenticarse para poder acceder a ella. Para el desarrollo de este trabajo se supone que dicha autenticación ya se ha llevado a cabo.

Al igual que en el caso de PoW, antes de iniciar el proceso propiamente dicho de la *blockchain*, es necesario que los relojes de todos los nodos de la red marquen el mismo tiempo. En este caso, también se sincronizarán los relojes de los nodos de manera local en la red siguiendo el protocolo de sincronización explicado en la sección 3.1.

Para esta *blockchain* se ha decidido que el *hash* del bloque génesis, mencionado y explicado en la sección 2.4, se calcule simplemente a partir de un dato elegido por el programador. Esta decisión se debe a que, a diferencia de PoW, el *hash* de un bloque no tiene que cumplir ninguna norma específica y, por lo tanto, es irrelevante el valor que tenga para las pruebas.

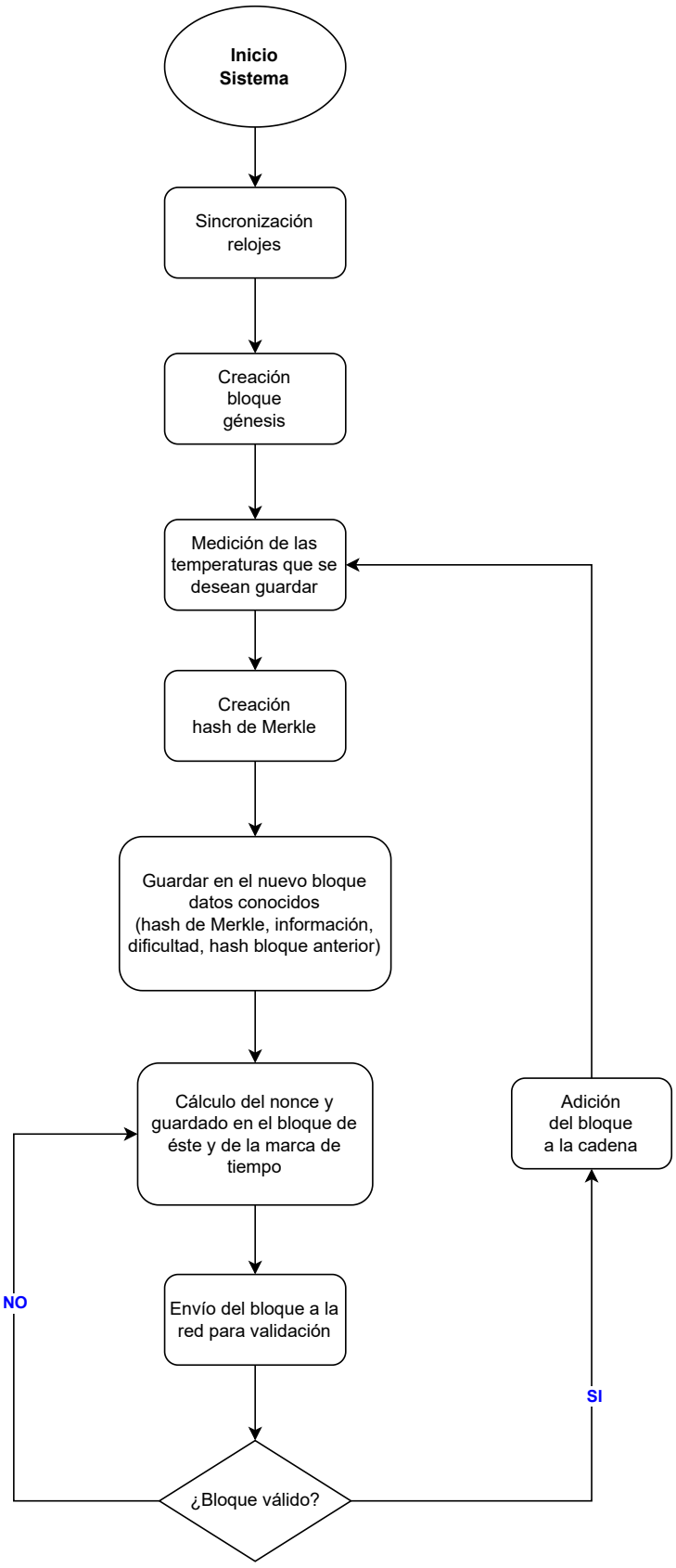


Figura 3.2: Flujograma de proceso en *blockchain* con PoW

El procedimiento de esta *blockchain*, desarrollado en la sección 2.4, conlleva mucho envío de mensajes simultáneamente y aumenta la probabilidad de saturar la red a medida que crece la misma, es por ello que se ha decidido que el proceso sea lo más secuencial posible y con tiempos de espera distintos para cada nodo, para no enviar tantos mensajes a la vez y así no saturar la red. Para conseguir esa secuencialidad buscada, al iniciar el sistema, la red decide aleatoriamente tanto el orden de los nodos para crear un bloque como el tiempo de espera para cada nodo.

Conociendo ya la estructura de esta *blockchain*, es momento de explicar cómo se desarrolla todo el proceso de dicha *blockchain*. En ella, dicho proceso se centra en un nodo que tenga que valorar el bloque creado por otro, ya que el nodo que crea el bloque solo tiene que sumar el número de valoraciones finales para decidir si se añade el bloque a la cadena o no.

En primer lugar, como se muestra en el flujograma de la figura 3.3, todos los nodos, tras autenticarse para unirse a la red, sincronizarse entre sí y decidir el orden y tiempos de espera para crear bloques, tienen que crear el bloque génesis de la cadena tal y como se ha explicado antes.

Tras esto, el nodo que crea un bloque, guarda en éste todos los datos necesarios, mencionados al explicar la estructura de un bloque, y lo manda a la red. Una vez que el resto de nodos han hecho todas las valoraciones necesarias se las mandan al nodo creador del bloque. Siendo n el número de nodos de la red, al nodo creador le llegan confirmaciones de $n-1$ nodos, a los cuales les llegaron valoraciones de $n-2$ nodos, por lo tanto, al creador le llegan en total $(n-2)(n-1)$ confirmaciones, ya sean positivas o negativas. Este nodo creador cuenta las valoraciones positivas confirmadas y si cumple la norma impuesta por la red de que el resultado de esa suma sea superior a dos tercios de la red, se añade el bloque a la cadena. Si el resultado de la suma no cumple el objetivo, el nodo cambia su orden de creación de bloque para volver a intentarlo después de que el resto de la red haya creado su bloque. Tras esto, haya conseguido añadir su bloque a la cadena o no, el nodo queda a la espera de que otro nodo cree un bloque para valorarlo. Todo ello mostrado en el flujograma de la figura 3.4.

Por otra parte, el resto de nodos queda a la espera de un bloque creado por otro nodo tras crear el bloque génesis. Una vez llega el bloque, el nodo comprueba que la información que contiene es correcta (*hash* del bloque anterior y número de secuencia) y manda su valoración al resto. A continuación, a este nodo llegarán las valoraciones del resto de nodos de la red, es decir, le llegarán $n-2$ valoraciones siendo n el número de nodos de la red. Entonces, el nodo confirma, o no, las valoraciones del resto, y le manda al nodo creador del bloque dichas confirmaciones para que compruebe si puede o no añadir el bloque. Tras el envío, el nodo comprueba si es su turno de crear un bloque. Si es su turno, inicia el proceso de la creación de bloque como se ha explicado en el párrafo anterior. Si no es su turno, el nodo queda a la espera de que otro nodo cree un bloque para valorarlo. Todo esto se muestra en el flujograma de la figura 3.5.

Ahora es momento de explicar en qué se traducen los flujogramas desarrollados al programarlos en Contiki-NG con el lenguaje de programación C.

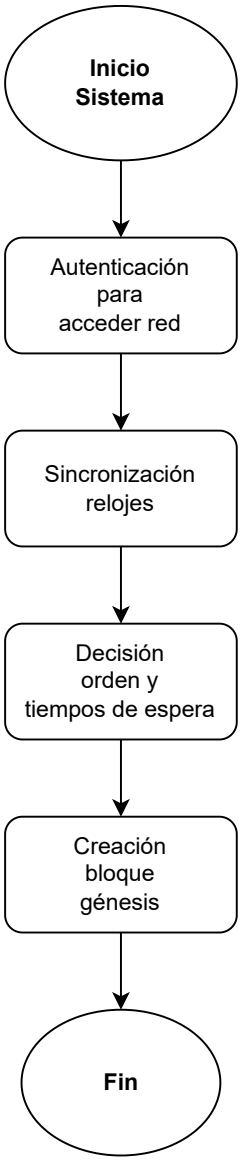


Figura 3.3: Flujograma de inicio de la red previo proceso de la *blockchain* con PBFT

En primer lugar, en cuanto al flujograma de la figura 3.3 se tiene lo siguiente:

1. En primera instancia, al iniciarse el sistema, se inicia un proceso en Contiki en el que se configuran todos los elementos de la Cookie necesarios para el funcionamiento propio y la comunicación con el resto de nodos, como puede ser la activación de sensores o el inicio de una pila de datos tipo LIFO donde se guardarán los bloques y que será la propia *blockchain*. También, se inician el resto de procesos en el orden que corresponda.

2. Respecto a la autenticación para acceder a la red, como ya se ha explicado antes, en este trabajo se supone realizada para centrarnos en la *blockchain* propiamente dicha, por lo que no actualmente aparece en el código de la aplicación.

3. La sincronización de los relojes de los nodos corresponde a un proceso en Contiki en el que, en primer lugar, el nodo que quiere sincronizarse intercambia mensajes

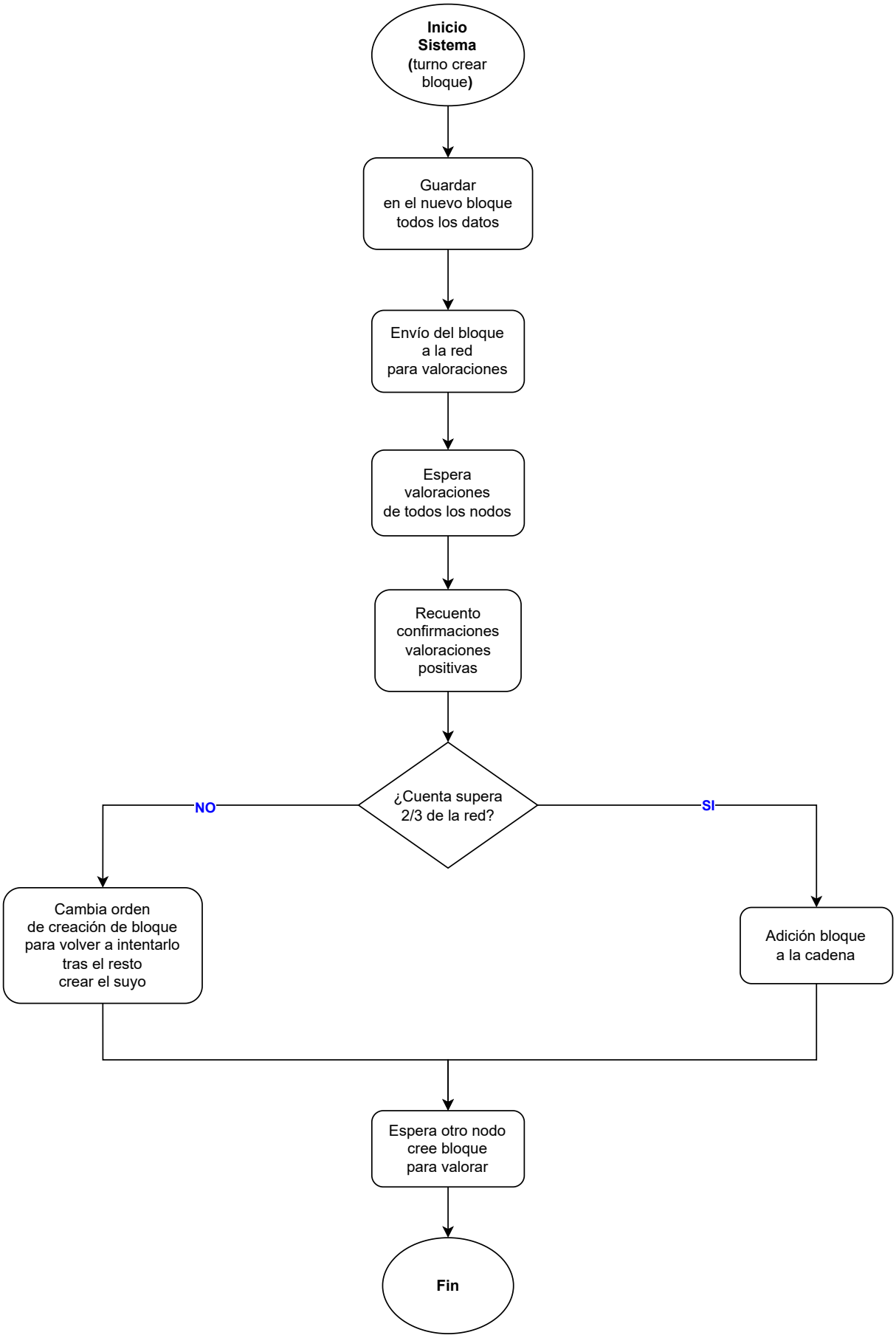


Figura 3.4: Flujograma de creación y adición de bloque en PBFT

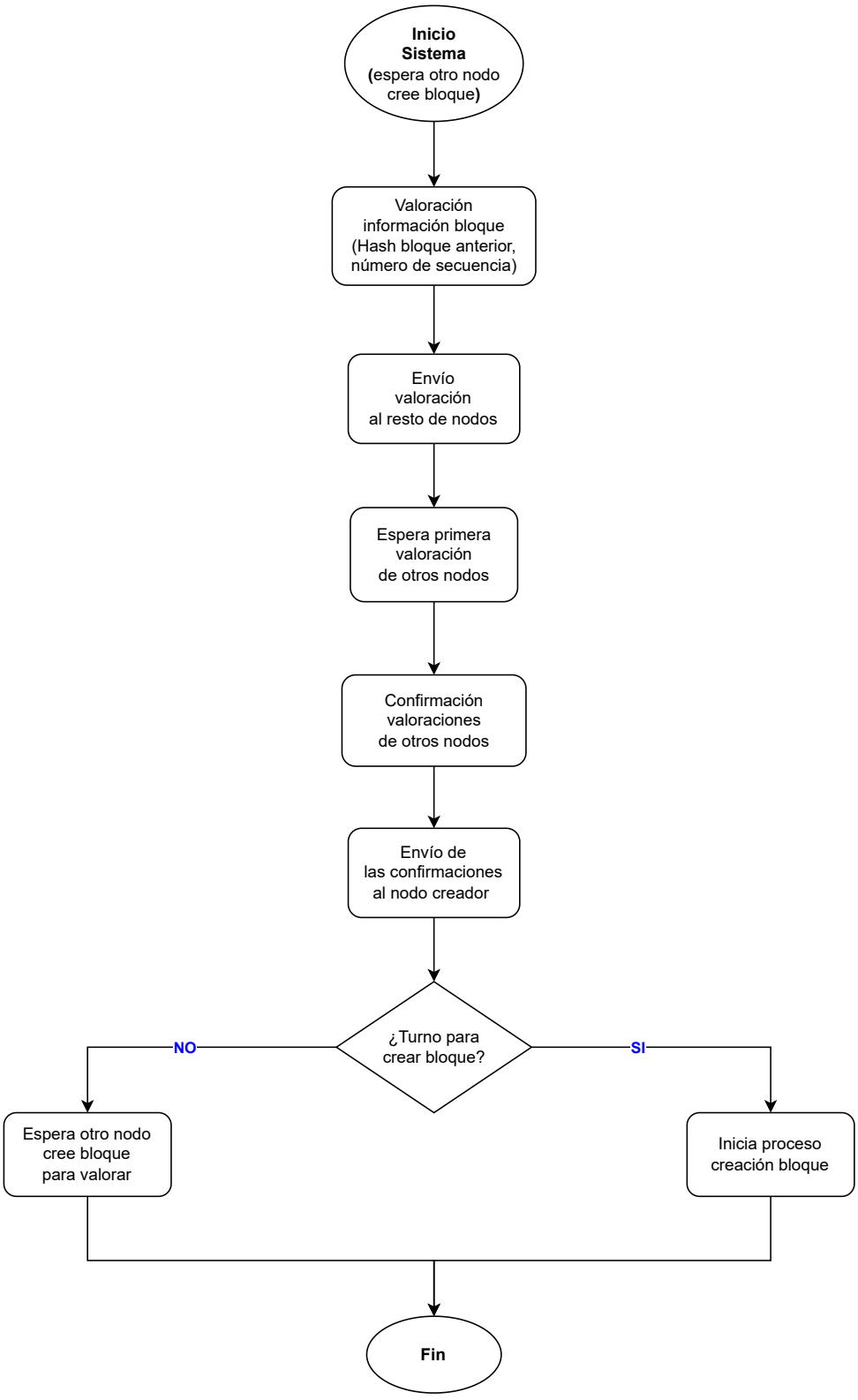


Figura 3.5: Flujograma de valoración de bloque en PBFT

con el nodo de referencia para conocer sus tiempos y, posteriormente, realiza todos los cálculos descritos en el protocolo de sincronización desarrollado anteriormente.

4. Para la asignación del orden y los tiempos de espera no hay un proceso específico en Contiki, sino que se aprovechan los mensajes del proceso de la sincronización para realizar la tarea.

5. Por último, para la creación del bloque génesis, hay un proceso en Contiki en el cual se genera el *hash* y se le da valor al número de secuencia como ya se ha explicado.

Por otro lado, en cuanto al flujograma de la figura 3.4 se tiene lo siguiente:

1. En primer lugar, existe un proceso en Contiki para comprobar, cada vez que se actualiza el turno para crear un bloque, si es el turno del nodo correspondiente. En el caso de este flujograma, esa comparación sería correcta y sería el turno de este nodo.

2. Para la creación del bloque, existe otro proceso en Contiki en el cual primero se guardan todos los datos mencionados en la sección 2.4, para después hacer el *hash* del grupo de datos formado por la información que se quiere almacenar en el bloque, la marca de tiempo y el *hash* del bloque anterior, y así guardar ese nuevo *hash* también en el bloque.

3. El envío del bloque al resto de la red corresponde a otro proceso en Contiki que está siempre a la espera de que se cree un nuevo bloque para enviarlo al resto de nodos para que realicen sus valoraciones.

4. Por último, respecto al recuento de valoraciones del resto de nodos sobre el bloque creado, existe un proceso en Contiki en el cual se van sumando todas las confirmaciones y, si al mandar todos sus valoraciones finales la suma de confirmaciones cumple la norma impuesta por la red, se añade el bloque a la cadena, es decir, se realiza el guardado de los datos en la pila LIFO, y se avisa al resto de nodos para que hagan lo propio.

En último lugar, en cuanto al flujograma de la figura 3.5 se tiene lo siguiente:

1. Para cuando un nodo crea un bloque y lo envía a la red para que lo valoren, hay un proceso en Contiki que realiza esa primera valoración sobre el bloque y la envía al resto de nodos tras esperar el tiempo asignado al principio del funcionamiento de la red.

2. Respecto a la confirmación de las valoraciones del resto de nodos, existe otro proceso en Contiki que realiza dicha tarea comprobando que son correctas las valoraciones.

3. Además, hay otro proceso en Contiki que se dedica a contar cuántas confirmaciones realiza el proceso anterior para, cuando todos los nodos hayan mandando sus valoraciones, realizar el recuento total y enviárselo al nodo creador del bloque para que compruebe si se puede añadir o no el bloque a la cadena.

4. Por último, tras enviar las confirmaciones se ejecutaría el primer proceso explicado en el flujograma anterior para comprobar si es el turno del nodo de crear un nuevo bloque.

Capítulo 4

EVALUACIÓN DE LAS IMPLEMENTACIONES PROPUESTAS

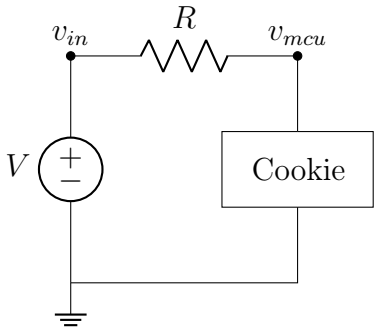
En este capítulo se evalúan las implementaciones propuestas en el capítulo anterior en lo referente a su impacto energético, detallando las mediciones y cálculos realizados en el proceso.

4.1. Medición experimental del consumo energético

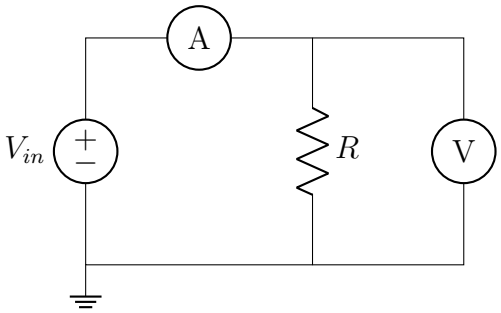
Para realizar las medidas de ambas implementaciones se conecta una fuente de tensión V de 3.3 V con nuestra resistencia R y la Cookie en serie, y se usa un osciloscopio para medir tres tensiones en tres puntos distintos del circuito, la tensión de entrada v_{in}, la tensión de la Cookie v_{mcu} y la tensión de un pin de la Cookie v_{ref} que se activará en los momentos que se desea medir, como se muestra en la figura 4.1a.

En primer lugar, se identifica en las medidas tomadas de v_{ref} en qué momentos está activado el pin de la Cookie para operar solo con los datos de las diferentes tensiones de los correspondientes tiempos. Una vez se sabe con qué datos hay que trabajar, se procede a calcular todo lo necesario para hallar el consumo de energía.

La resistencia R empleada presenta un valor nominal de 10 Ω con una tolerancia indicada por el fabricante. Sin embargo, es posible conocer con mayor detalle su valor real realizando una serie de medidas experimentales que, además, permitan reducir la incertidumbre sobre el mismo. Para hallar dicho valor se conectará la resistencia R con una fuente de tensión V_{in}, además de un amperímetro y un voltímetro, como se muestra en la figura 4.1b. Una vez todo conectado, se miden las intensidades que se obtienen con distintas tensiones y se haya la recta de regresión que forman todos los puntos de la forma (I_R, V_R). La pendiente de dicha recta será el valor real aproximado de la resistencia. Como se puede observar en la tabla 4.1 y en la figura 4.2, con nuestros datos se obtiene que el



(a) Circuito para medir consumo de la Cookie



(b) Circuito para medir el valor de la resistencia

Figura 4.1: Circuitos para realizar medidas

$\boldsymbol{I_R \pm 0.0001}$ **mA**	$\boldsymbol{V_R \pm 0.1}$ **mV**
2.4735	24.4
3.6700	36.3
4.5671	45.2
5.7769	57.2
7.7523	76.8
Resistencia	$(9.925 \pm 0.003)\,\Omega$

Tabla 4.1: Valor real aproximado de la resistencia

valor es $R = (9.925 \pm 0.003)\,\Omega$.

Una vez es conocido el valor de nuestra resistencia, ya se puede proceder a medir el consumo de cada implementación en funcionamiento para, posteriormente, poder sacar conclusiones sobre cada una.

La energía eléctrica E consumida en un tiempo t se define como:

$$E = \int_{t_1}^{t_n} P\,dt \tag{4.1}$$

por lo que para calcularla, se hallará primero la potencia P en cada instante y después se aproximará el valor de la integral usando un método de integración numérica. En este caso se ha decidido usar el método trapezoidal.

Por un lado, usaremos v_{in} y v_{mcu} junto con R para hallar la corriente I que circula en cada momento por nuestro circuito con la siguiente expresión:

$$I = \frac{(v_{in} - v_{mcu})}{R} \tag{4.2}$$

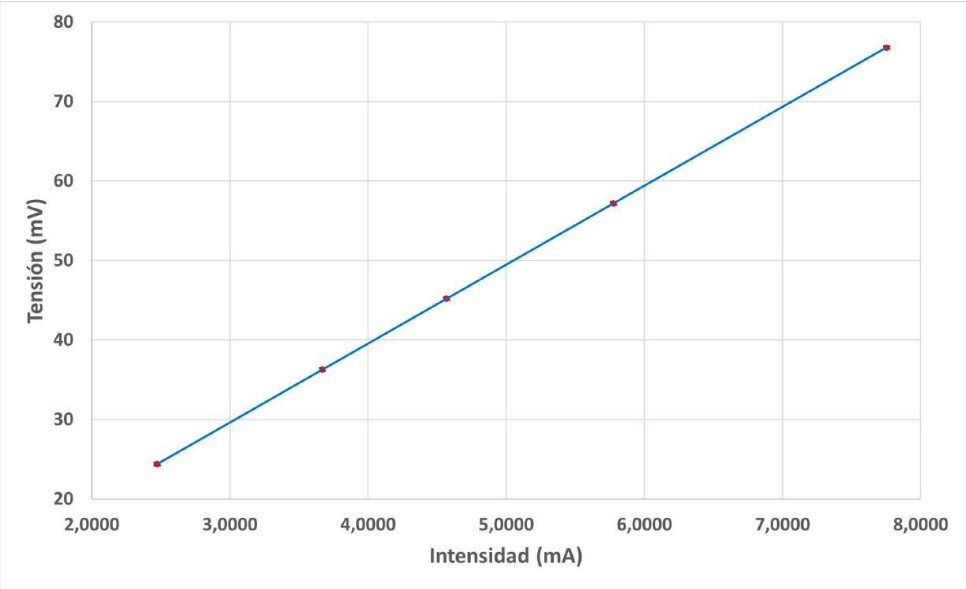


Figura 4.2: Recta de regresión para el cálculo de la resistencia

Una vez hallada la intensidad I, se calcula junto a la tensión de la Cookie v_{mcu}, su potencia P en cada instante a través de la siguiente ecuación:

$$P = v_{mcu} \cdot I \tag{4.3}$$

Ahora es momento de explicar el método trapezoidal. En este método de integración numérica, para aproximar el valor de la integral se forman trapezoides uniendo los valores máximos y mínimos de la función en cada subintervalo, se halla su área y se calcula la suma de todas las áreas.

El área de un trapezoide está definido por la siguiente ecuación:

$$A = \frac{h(b_1 + b_2)}{2} \tag{4.4}$$

donde b_1 y b_2 son los lados paralelos y h es la altura. En este caso b_1 y b_2 se corresponden con los valores de la función, es decir con P_i y P_{i+1}, y h se corresponde con Δt, es decir con $(t_{i+1} - t_i)$. Por lo que si el área de uno de estos trapezoides lo denominamos como e_i, queda la siguiente expresión:

$$e_i = \frac{(P_i + P_{i+1})}{2} \cdot (t_{i+1} - t_i) \tag{4.5}$$

Por lo tanto, para conseguir el valor de la energía total E consumida en un determinado tiempo t solo hay que sumar todas las áreas e_i que corresponden a ese tiempo t. Si el tiempo t, en el que se quiere calcular la energía consumida, está formado por n instantes, la ecuación para hallar dicha energía E será:

$$E = \sum_{i=1}^{n-1} e_i \tag{4.6}$$

Se observa que el sumatorio de la ecuación 4.6 llega hasta $n-1$ en vez de hasta n. Esto es debido a que el instante n ya está incluido en el término e_{n-1} como puede deducirse de la expresión 4.5.

Posteriormente, se realizan varias medidas, y tras realizar todos los cálculos descritos, se toma la media como valor representativo.

A continuación, se explica qué energías se han medido en cada implementación y se muestran los resultados y cálculos específicos necesarios de cada una.

4.2. Medidas de consumo con PoW

En esta sección, se detalla qué momentos son críticos para el consumo y, por lo tanto, en qué momentos se han tomado las medidas en el laboratorio para la implementación de la *blockchain* basada en PoW, además de posibles cálculos necesarios.

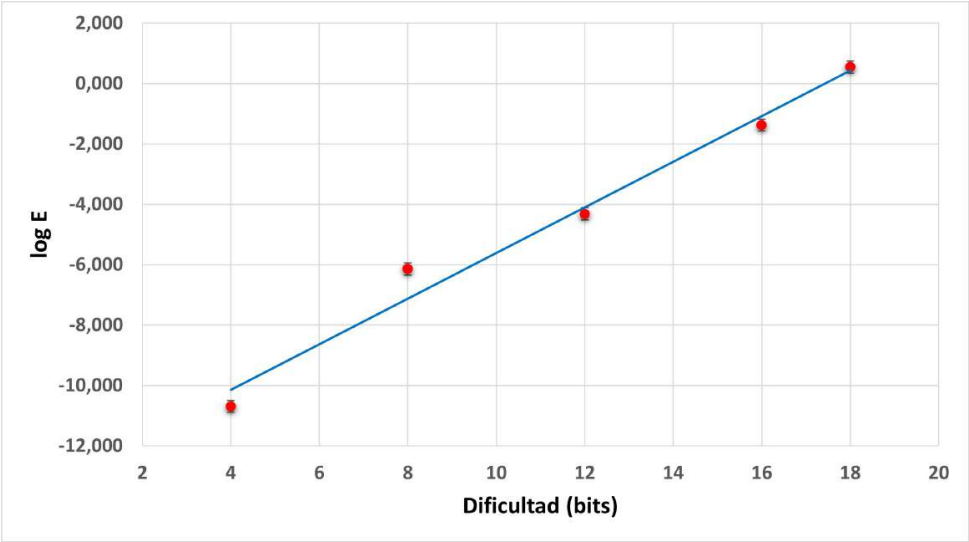
En el caso de esta implementación, como se explicó en la sección 3.2, la validación del bloque por parte de la red es muy rápida y no requiere casi recursos en comparación con la creación del bloque, por lo que el pin de la Cookie estará activo para medir las tensiones durante todo el proceso de creación, desde la obtención de la información que se desea guardar hasta el cálculo y guardado del *nonce* y la marca de tiempo, como se veía en la figura 3.2.

También, como se mencionó en la información sobre PoW en el capítulo 2, cuando aumenta el número de ceros que decide la red que tiene que tener el *hash* del bloque, aumenta la dificultad para minar dicho bloque, razón por la que ese número recibe el nombre de dificultad. Debido a esto, se han tomado medidas de energía y tiempo con distintas dificultades para observar la dependencia real del consumo en función de la dificultad, como se ve en la tabla 4.2 y en las figuras 4.3a y 4.3b.

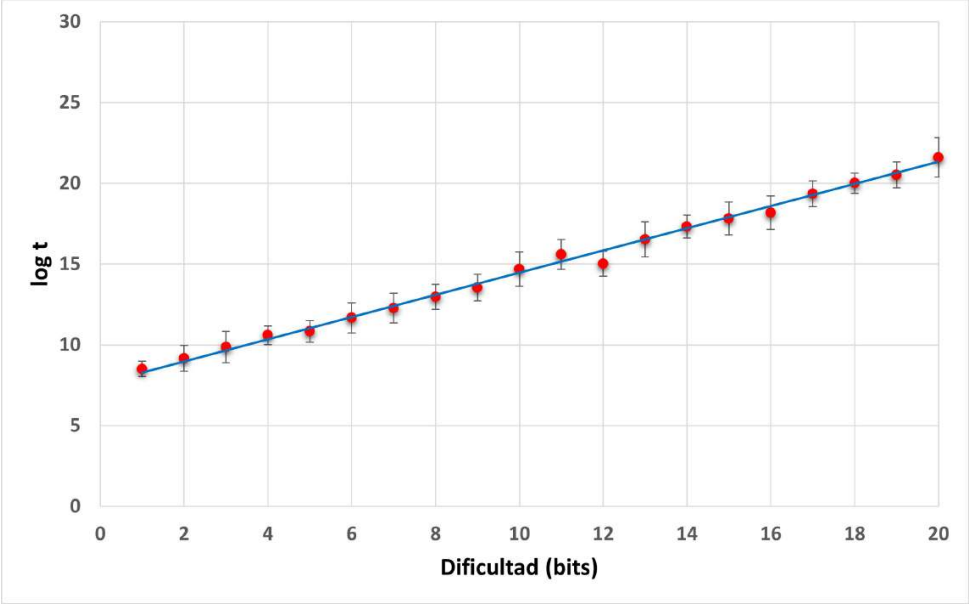
Si se miran con atención la tabla y las gráficas adjuntas sobre el consumo, se observa que los datos de energía y tiempo se muestran con el valor de su logaritmo natural. La energía está calculada en julios (J) y el tiempo en ciclos de reloj y con estas unidades es muy difícil apreciar adecuadamente la evolución de ambas variables ya que tienen un crecimiento exponencial según el nivel de dificultad aumenta. Es por esto por lo que se usa el logaritmo, para poder observar de manera idónea dicha evolución de las variables en función de la dificultad.

Dificultad	log (E/1J)	log (t)
4	$-10.6920000 \pm 0.0000013$	11.0 ± 0.6
8	-6.138000 ± 0.000001	13.0 ± 0.8
12	-4.3200000 ± 0.0000005	15.0 ± 0.8
16	-1.3760000 ± 0.0000005	18 ± 1
18	0.5440000 ± 0.0000005	20.0 ± 0.6

Tabla 4.2: Valores de energía y tiempo según la dificultad en PoW



(a) Energía frente a dificultad en PoW



(b) Tiempo frente a dificultad en PoW

Figura 4.3: Representación gráfica y ajuste de los valores de energía y tiempo según la dificultad en PoW

Además, haciendo uso de esta representación con el logaritmo, se puede hallar una recta de regresión con los puntos obtenidos y obtener así un modelo matemático que estime el valor de la energía o del tiempo, según corresponda, en función de la dificultad.

Si representamos la dificultad por la letra d, para la energía E y para el tiempo t quedarían expresiones de la siguiente forma:

$$\log(E) = m \cdot d + b$$
$$\log(t) = m \cdot d + b$$

Para la energía E, m tiene un valor de 0.75 ± 0.06 y b tiene un valor de -13.15 ± 0.76. Respecto al tiempo t, m tiene un valor de 0.63 ± 0.05 y b tiene un valor de 8.08 ± 0.62.

4.3. Medidas de consumo con PBFT

En esta sección, se detalla qué momentos son críticos para el consumo y, por lo tanto, en qué momentos se han tomado las medidas en el laboratorio para la implementación de la *blockchain* basada en PBFT, además de posibles cálculos necesarios.

En el caso de esta implementación, como se explicó en el la sección 3.3, la valoración de los bloques por parte de los nodos de la red es muy sencilla y no requiere recursos en comparación con la cantidad de mensajes que se envían en el proceso, por lo que aquí será de mayor importancia la energía consumida en la creación del bloque más la energía consumida en cada envío de mensaje. De este modo, por un lado, se tomarán medidas de las tensiones activando el pin de la Cookie durante el guardado de la información en el bloque, y, por otro lado, se tomarán medidas de las tensiones activando el pin de la Cookie durante la acción de envío de mensaje, cuyos valores se aprecian en la tabla 4.3.

Como ya se ha mencionado, lo que marca la diferencia en este modelo de *blockchain* en cuanto al consumo es el número de mensajes, así que se procede a calcular cuál es el gasto energético aproximado por parte de toda la red en la creación de un bloque y todas las valoraciones.

Para ello, lo primero es saber cuántos mensajes se envía en todo el proceso completo. Como aparece mostrado en la figura 4.4, si el número de nodos de la red es n, en la primera fase, en la que un nodo crea su bloque y lo manda al resto de la red, se envían $n-1$ mensajes, ya que es un nodo mandando esos $n-1$ mensajes; en la segunda fase, en la que todos los nodos valoran el bloque y mandan dichas valoraciones al resto salvo al creador del bloque, se envían $(n-1)(n-2)$ mensajes, ya que son $n-1$ nodos mandando $n-2$ mensajes; y la tercera fase, en la que los nodos confirman o desmienten las valoraciones del resto de nodos y le mandan al creador cuantas confirmaciones hay, se envían $n-1$ mensajes, ya que son $n-1$ nodos enviando un único mensaje.

Ahora que ya se sabe el número de mensajes que se envían en todo el proceso y los valores de la energía en cada envío y en la creación del bloque, ya sí se puede calcular el

Energía creación bloque	$(40.270\,00 \pm 0.000\,05)$ µJ
Energía envío mensaje	(586.8000 ± 0.0011) µJ

Tabla 4.3: Valores de energía en creación de bloque y envío de mensaje en PBFT

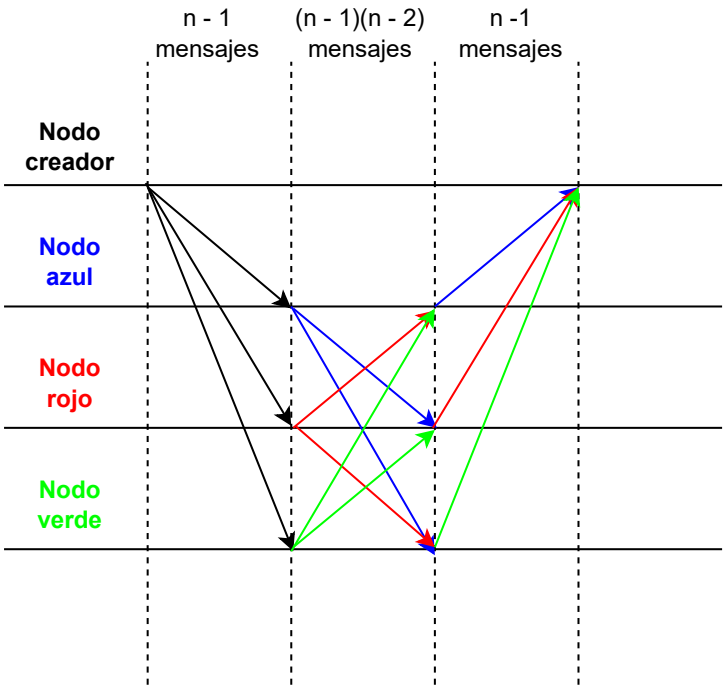


Figura 4.4: Número de mensajes en cada fase de PBFT en función del número de nodos de la red n

consumo total.

En primer lugar, el gasto energético total derivado del envío de mensajes E_{msg}, en función del número de nodos de la red n, será igual a la energía consumida en un solo envío e_{send} multiplicada por el número total de mensajes, es decir:

$$E_{msg} = e_{send} \cdot [(n-1) + (n-1) \cdot (n-2) + (n-1)]$$
$$E_{msg} = e_{send} \cdot [(n-1) \cdot (1 + n - 2 + 1)]$$
$$E_{msg} = e_{send} \cdot (n-1) \cdot n$$

Por lo que la energía total gastada debida al envío de mensajes en función del número de nodos de la red n obedece la siguiente expresión:

$$E_{msg} = e_{send} n^2 - e_{send} n \tag{4.7}$$

Por último, para hallar la energía total consumida en todo este proceso E, hay que sumar a la energía calculada en la expresión 4.7, la energía consumida en la creación del bloque e_{crt}, es decir:

$$E = E_{msg} + e_{crt}$$

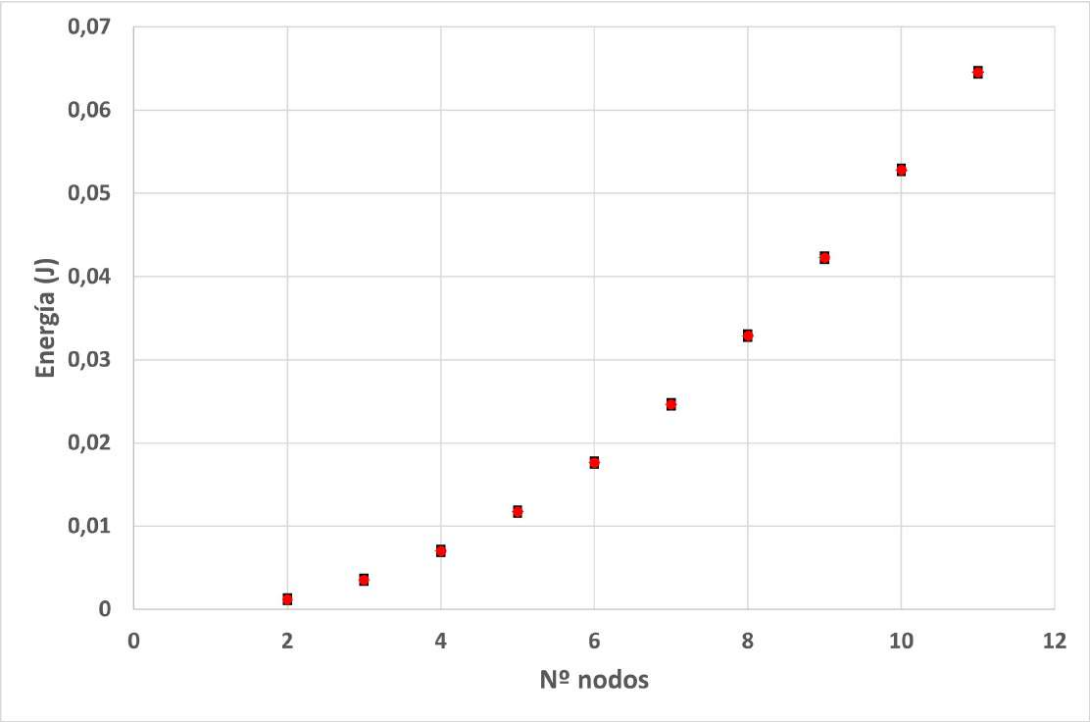


Figura 4.5: Energía frente a número de nodos en PBFT (Error pequeño y prácticamente inapreciable pero puede consultarse sus valores en la tabla 4.4)

Número nodos (n)	**Energía**
4	$(7.100\,000 \pm 0.000\,013)$ mJ
5	$(11.800\,00 \pm 0.000\,02)$ mJ
6	$(17.600\,00 \pm 0.000\,03)$ mJ
7	$(24.700\,00 \pm 0.000\,05)$ mJ
8	$(32.900\,00 \pm 0.000\,06)$ mJ
9	$(42.300\,00 \pm 0.000\,08)$ mJ
10	(52.8000 ± 0.0001) mJ

Tabla 4.4: Valores de energía según el número de nodos de la red

Por lo que la energía total consumida en el proceso obedece la siguiente expresión:

$$E = e_{send}n^2 - e_{send}n + e_{crt} \tag{4.8}$$

Ahora, tras ya tener la expresión 4.8, se sustituyen los valores de las energías contenidos en la tabla 4.3 y se obtiene la ecuación específica para este caso:

$$E = (586800.0 \pm 1.1) \times 10^{-9}n^2 - (586800.0 \pm 1.1) \times 10^{-9}n + (4027000 \pm 5) \times 10^{-11}\ \text{J} \tag{4.9}$$

Una vez ya obtenida la ecuación 4.9 que describe el consumo de la aplicación, se sustituye el valor de n por distintos valores para ver como evoluciona el consumo de energía según aumenta el número de nodos de la red, siendo mostrados algunos de los

valores en la tabla 4.4 y en la figura 4.5.

4.4. Comparación de consumo entre PoW y PBFT

En esta sección, se comparan los datos obtenidos y explicados en la secciones anteriores sobre ambas implementaciones, PoW y PBFT.

Como se pudo ver en la sección 4.2, el consumo de energía en PoW depende fundamentalmente de la creación del bloque y la dificultad de éste, mientras que en el caso de PBFT depende fundamentalmente del número de nodos de la red y los mensajes que intercambian entre ellos. Debido a esto es difícil hacer una comparación directa pero si se atiende a los datos numéricos, se comprueba que el gasto energético es mayor en PoW. Por ejemplo, si escogemos una red compuesta por 10 nodos, la tabla 4.4 nos muestra que la red consumiría 0.0528 J en el caso de PBFT, mientras que para la misma red en PoW con una dificultad de valor 16, no muy alta dado que son 16 bits de 256, la tabla 4.2 nos muestra que la red consumiría aproximadamente 0.2526 J.

Capítulo 5

CONCLUSIONES

Tras todo lo expuesto en este trabajo, en este capítulo se extraen algunas conclusiones importantes al respecto.

En primer lugar, entender el motivo por el cuál es interesante mezclar los campos de *blockchain* y redes IoT. En el tipo de aplicación objetivo, como es una para el ámbito sanitario en la que la privacidad de los datos es primordial, la descentralización que conlleva la tecnología *blockchain* puede ser muy importante ya que aporta mayor seguridad a esos datos.

En contraposición al aumento de seguridad, el uso de *blockchain* también produce un gasto energético considerable que hay que tener en cuenta debido a que los dispositivos de una red IoT no suelen tener mucha capacidad energética.

Siguiendo la línea del gasto energético, en este trabajo se han tomado medidas de consumo de dos implementaciones distintas, cada una con un tipo de *blockchain* diferente. Se ha visto que, por lo general, la implementación con el algoritmo de consenso PoW tiene un consumo mayor que la implementación con el algoritmo de consenso PBFT.

Por otro lado, en la implementación con PoW dicho consumo está asociado a la creación del bloque principalmente, creciendo exponencialmente según aumenta la dificultad, y casi no hay envío de mensajes, mientras que en la implementación con PBFT el consumo se asocia fundamentalmente al envío de mensajes debido a la inmensa cantidad de mensajes, creciendo siguiendo una expresión cuadrática según el número de nodos de la red aumenta.

También, debido a las características de nuestro tipo de red, entender que es posible que haya fallos en las comunicaciones por una saturación de la red si hay muchos nodos enviando mensajes, motivo por el cual se ha decidido secuenciar el proceso lo máximo posible y poner tiempos de espera.

Por lo tanto, la elección de implementar una *blockchain* en una red IoT para aumentar la seguridad, parece adecuada siempre y cuando se escoja un modelo en el que el consumo energético sea lo mínimo posible debido a las características propias de los dispositivos

de dichas redes. Por esta razón entre los dos modelos de este trabajo se escoge aquel con PBFT.

5.1. Líneas futuras

En esta sección se enumeran posibles cambios a realizar o nuevas líneas de desarrollo a seguir en cada una de las implementaciones.

En ambas implementaciones se podría desarrollar un sistema de almacenamiento descentralizado adicional para guardar los datos de salud de los usuarios y en la *blockchain* guardar los datos de acceso de los usuarios para acceder a ellos. Además, otra línea futura para ambas implementaciones podría ser mejorar la seguridad de la red adaptándola a un escenario post-cuántico.

5.1.1. Implementación con PoW

En este apartado se mencionan posibles mejoras que puedan realizarse en la *blockchain* desarrollada con el algoritmo de consenso PoW:

1. Podría llevarse a cabo un cambio en la sincronización de los relojes de los nodos eliminando el protocolo descrito en la sección 3.1 y conectando los nodos a *Internet* para obtener la hora del lugar y tener todos esa hora. Este cambio ahorraría a los nodos tener que realizar todos los cálculos del protocolo.

2. También podría modificarse la manera en la que hallar el *hash* del bloque génesis para que fuera totalmente aleatoria y no hubiera ninguna variable fija.

3. Además, otro cambio podría ser realizar el *hash* de Merkle solo con dos grupos de datos, en vez de con cuatro, para así reducir el tiempo de minado de un bloque y su consumo de energía.

4. Otra posible mejora sería dividir el proceso de Contiki que crea el bloque en varios procesos para que el nodo pueda realizar más tareas de manera paralela. Por ejemplo, podría realizarse la creación del bloque génesis en un proceso, el guardado de los datos en otro y el cálculo del *nonce* y su guardado junto a la marca de tiempo en otro más.

5.1.2. Implementación con PBFT

En este apartado se mencionan posibles mejoras que puedan realizarse en la *blockchain* desarrollada con el algoritmo de consenso PBFT:

1. Respecto a la sincronización de los relojes de los nodos, podría realizarse el mismo cambio mencionado en el apartado anterior con PoW.

2. En el caso de esta *blockchain*, habría que desarrollar un protocolo de autenticación para acceder a la red, que en este trabajo se ha dado por supuesto.

3. Otro posible cambio trataría de crear un proceso en Contiki específico para la asignación del orden, para la creación de un bloque, y de los tiempos de espera.

4. También, al igual que en PoW, se puede aleatorizar la obtención del *hash* del bloque génesis en vez de ser una variable fija.

BIBLIOGRAFÍA

[1] N. Mohan and J. Kangasharju, "Edge-Fog cloud: A distributed cloud for Internet of Things computations," in *2016 Cloudification of the Internet of Things (CIoT)*, pp. 1–6, 2016.

[2] Silicon Labs, *EFR32MG12 Gecko Multi-Protocol Wireless SoC Family Data Sheet*, Rev 1.8.

[3] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The Contiki-NG open source operating system for next generation IoT devices," *SoftwareX*, vol. 18, p. 101089, 2022.

[4] P. Merino, G. Mujica, J. Señor, and J. Portilla, "A Modular IoT Hardware Platform for Distributed and Secured Extreme Edge Computing," *Electronics*, vol. 9, no. 3, 2020.

[5] M. Salimitari and M. Chatterjee, "A survey on consensus protocols in blockchain for iot networks," *arXiv preprint arXiv:1809.05613*, 2018.

[6] D. Hellwig, G. Karlic, A. Huchzermeier, *et al.*, *Build your own blockchain.* Springer, 2020.

[7] G. Xu and K. Shi, "An Improved Practical Byzantine Fault Tolerance Consensus Algorithm," in *2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, pp. 1083–1089, IEEE, 2022.

[8] G. Seth and A. Harisha, "Energy efficient timing-sync protocol for sensor network," in *2015 International Conference on Computing and Network Communications (CoCoNet)*, pp. 912–916, IEEE, 2015.

PLANIFICACIÓN Y PRESUPUESTO DEL PROYECTO

En este apartado del trabajo se reflejan tanto la planificación del proyecto, con su respectivo diagrama, como el presupuesto estimado del mismo.

1. Fases del proyecto

Definición del proyecto

Este Trabajo Fin de Grado aparece debido al interés creciente en la sociedad actual por la tecnología *blockchain* y la seguridad que ofrece la descentralización que dicha tecnología conlleva. En particular, aquí se pretende usar *blockchain* en un sistema de *Internet of Things*.

Por lo tanto, el interés principal es intentar mejorar la seguridad de una red IoT haciendo uso de una *blockchain* y haciendo frente a los retos que supone unir ambos campos, como el consumo de energía o de memoria.

Planificación y estructuración

La planificación inicial del proyecto estableció una estructuración con las siguientes fases:

1. Una primera etapa de investigación y aprendizaje sobre los diferentes modelos de *blockchain* y algoritmos de consenso y sus ventajas e inconvenientes.

2. Adaptación al funcionamiento del sistema de *Internet of Things* (Contiki-NG) con el que, posteriormente, se va a trabajar.

3. Selección de dos tipos distintos de *blockchain*, y sus respectivos algoritmos de consenso, y estudio en detalle de los mismos.

4. Implementación de las *blockchain* escogidas.

5. Propuesta de posibles mejoras a lo implementado anteriormente.

6. Análisis de los resultados obtenidos.

Investigación sobre *blockchain*

Esta fase se concentra en la lectura y el entendimiento de varios modelos de *blockchain* y algoritmos de consenso para, posteriormente, poder hacer una elección adecuada de uno de ellos que se adapte a las necesidades y limitaciones de nuestro sistema.

Aprendizaje de Contiki-NG

En esta ocasión, se trata de aprender como funciona el sistema operativo que se va a utilizar en nuestra red IoT y habituarse a él, para poder implementar correctamente en el futuro la *blockchain* y todas las comunicaciones necesarias para el funcionamiento de nuestro sistema.

Selección de *blockchain* y algoritmo de consenso

Tras la búsqueda de información sobre *blockchain* y algoritmos de consenso y el aprendizaje sobre nuestro sistema operativo, se procede a elegir qué dos tipos son convenientes para el proyecto y se estudian en profundidad para poder llevar a cabo la implementación de las mismas correctamente.

Implementación de las *blockchain*

En esta fase, se implementará todo lo necesario para el funcionamiento de cada *blockchain* en nuestro sistema de *Internet of Things*, todo ello desarrollado en el lenguaje de programación C. Se realizarán todas las pruebas necesarias para asegurar el correcto funcionamiento de lo implementado y poder extraer conclusiones adecuadas sobre el rendimiento de dicha implementación en nuestra plataforma.

Propuesta de mejoras

Tras todo lo desarrollado anteriormente, se procede a pensar posibles mejoras que podríamos implementar en nuestro sistema para aumentar cualquier característica como podría ser la seguridad o la eficiencia.

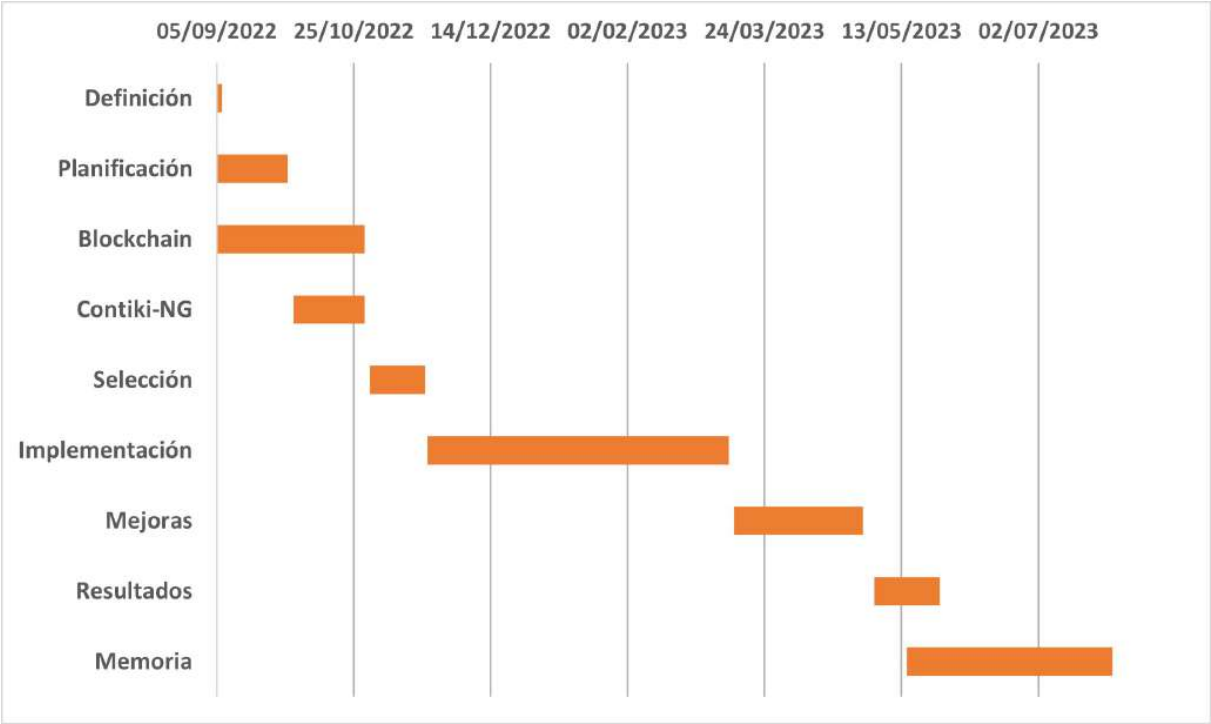


Figura 5.1: Diagrama de Gantt del proyecto

Ánalisis de los resultados

Una vez acabado todo el trabajo y planteadas las posibles mejoras, se deben sacar conclusiones sobre la viabilidad de la aplicación de cada *blockchain* sobre nuestra plataforma y establecer posibles pasos a seguir en el futuro si el proyecto continuara.

2. Estructuración temporal del proyecto

En la tabla 5.1 y en el Diagrama de Gantt, figura 5.1, se muestra la división temporal que se hace para cada tarea planificada en el proyecto. Se observa que la suma total de horas se aproxima a las 300 que corresponde a un Trabajo Fin de Grado según lo impuesto en la normativa.

A pesar de que lo ideal sería una mayor paralelización entre tareas, se aprecia que en este caso es escasa debido a que ciertas fases del proyecto dependen de los resultados de fases anteriores para poder ser llevadas a cabo. Por lo tanto, la mayoría de etapas se dan de forma secuencial. Las partes que se pueden realizar de forma simultánea son las relacionadas con búsqueda de información o elaboración de la memoria final.

Fase	Duración (horas)	Fecha Inicio	Fecha Fin
Definición	2	05/09/2022	05/09/2022
Planificación	10	05/09/2022	30/09/2022
Blockchain	30	05/09/2022	28/10/2022
Contiki-NG	15	03/10/2022	28/10/2022
Selección	10	31/10/2022	18/11/2022
Implementación	90	21/11/2022	10/03/2023
Mejoras	40	13/03/2023	28/04/2023
Resultados	25	03/05/2023	26/05/2023
Memoria	80	15/05/2023	28/07/2023

Tabla 5.1: Tabla de planificación temporal del proyecto

3. Presupuesto

Por último, en esta sección, se realiza una estimación del presupuesto del proyecto, resumido en la tabla 5.2, valorando tanto los recursos materiales como humanos.

Por un lado, los recursos materiales están formados por un ordenador con el que trabaja el alumno y todo el material de laboratorio que sea necesario, como son los aparatos de medida y el hardware con el que se trabaja, nodos Cookie principalmente. El coste que éstos suponen son sus amortizaciones y el coste de la luz gastada.

La amortización de los objetos se calcula multiplicando su precio de compra por su tiempo de uso y dividiendo por la vida útil de los mismos. Se estima que el ordenador tiene un precio de compra de 900€, un tiempo de uso de unas 275 horas y una vida útil de 5 años con 8 horas de uso diarias, por lo que su coste será de 16.95€. En cuanto al resto de material, se estima que su conjunto tiene un precio de compra de 1000€, un tiempo de uso de 155 horas de acuerdo a la planificación mostrada anteriormente y una vida útil también de 5 años con 8 horas de uso diarias, por lo que su coste será de 10.62€. Por lo tanto, la amortización total de todos los equipos será de 27.57€.

Ahora se calcula el coste eléctrico que conlleva el uso de dichos equipos. Dado que el precio de la electricidad no es constante, varía según día y hora, se opta por tomar el precio medio de uno de los meses anteriores al comienzo del trabajo, siendo así dicho valor 0.3071€/kWh. Para el ordenador se usará una fuente de alimentación de 300W y para el resto de dispositivos se estima la necesidad de 150W. Teniendo en cuenta el tiempo de uso de cada objeto, la potencia necesaria para cada uno y el precio medio de la electricidad, se obtiene un coste eléctrico de 32.48€.

También habría que tener en cuenta el coste de todas las licencias necesarias para desarrollar el proyecto, principalmente de software. En este caso, dicho coste es nulo ya que todas están disponibles, ya sea por distribuirse de forma gratuita por parte de la propia empresa o por existir en formato académico.

Por otro lado, los recursos humanos están compuestos por el alumno y su tutor. El

Coste ordenador	16.95€
Coste equipo	10.62€
Coste total material	27.57€
Coste eléctrico ordenador	25.34€
Coste eléctrico equipo	7.14€
Coste eléctrico total	32.48€
Salario alumno	2265€
Salario tutor	2500€
Coste total personal	4765€
Coste total	4825.05€
Presupuesto (coste + IVA)	**5838.32€**

Tabla 5.2: Resumen del presupuesto del proyecto

alumno se estima que trabajará 302 horas, como se ha especificado en la tabla 5.1 de planificación temporal, y tendrá un sueldo de 7.50€la hora. En cuanto al tutor, se espera que trabaje en las partes más teóricas del proyecto y quizá guiando en alguna parte práctica, por lo que se estima que trabajará unas 100 horas, aproximadamente un tercio del tiempo del alumno, y tendrá un sueldo de 25€la hora. Teniendo todo esto en cuenta, y sabiendo que el salario total de una persona es su sueldo multiplicado por el tiempo de trabajo, el coste total del personal será de 4765€.

Por último, hay que sumar todos los costes hallados previamente para aplicar un 21 % de IVA y obtener así el presupuesto aproximado del proyecto. Realizando dicha suma, se obtiene un coste total de 4825.05€y, por lo tanto, un presupuesto para el proyecto de 5838.32€.

EVALUACIÓN DE IMPACTOS Y OTROS ASPECTOS

En este apartado se sopesan los posibles impactos que podría tener este proyecto, ya sean de aspecto social, económico, medioambiental u otros aspectos.

Por un lado, no habrá un gran impacto social ya que se trata de un proyecto para cambiar el modo de almacenar unos datos en un tipo de aplicación ya existente, por lo que la sociedad no notaría un gran cambio, pero las personas estarán más tranquilas sabiendo que sus datos están más seguros.

Respecto al impacto económico, su efecto dependerá de a qué escala se quiera realizar el proyecto. Cuantos más nodos se pretendan tener en la red, más dispositivos habrá que adquirir y habrá mayor gasto de energía y, por lo tanto, mayor gasto económico. Si se usa para redes privadas no muy grandes, como es el ejemplo de la implementación con PBFT, el impacto económico no será elevado.

Por otro lado, el impacto medioambiental, al igual que el económico, depende de la escala del proyecto. Cuantos más nodos se pretendan tener en la red, más dispositivos tendrán que fabricarse junto con las baterías y otros productos necesarios, y más energía habrá que producir para poder alimentar dichos elementos. Por lo tanto, al igual que para el caso de la economía, para redes no muy grandes el impacto medioambiental no será elevado.

Si tenemos en cuenta aspectos legales y éticos, en este trabajo habría que tener especial cuidado con el tratamiento de los datos, debido a que el objetivo es llegar a implementar lo desarrollado en una aplicación de salud con datos sensibles de las personas cuya seguridad y privacidad es primordial.

Por último, se relacionan las características de este proyecto con los Objetivos de Desarrollo Sostenible (ODS). Los ODS son 17 objetivos globales interconectados, mostrados en la figura 5.2, diseñados para ser un plan para lograr un futuro mejor y más sostenible para todos. En la figura 5.3 se pueden ver los ODS con los que cumple este proyecto. El ODS 3 tiene que ver con garantizar una vida sana y promover el bienestar de todos a todas las edades, por eso, este proyecto está relacionado con este objetivo al estar orientado hacia una aplicación de salud para proteger los datos sanitarios de las personas de manera segura. El ODS 9 tiene que ver con construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación, por eso, este

Figura 5.2: Objetivos de Desarrollo Sostenible

Figura 5.3: ODS relacionados con este proyecto

proyecto está relacionado con este objetivo al innovar en la forma de almacenar datos en una red IoT e intentar desarrollar una infraestructura segura para guardar dichos datos. El ODS 11 tiene que ver con lograr que las ciudades y los asentamientos humanos sean inclusivos, seguros, resilientes y sostenibles, por eso, este proyecto está relacionado con este objetivo por la misma razón que los otros dos, es decir, por crear un entorno seguro para guardar los datos privados de salud de las personas.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS, UNIDADES Y ACRÓNIMOS

DPoS - *Delegated Proof of Stake*

IoT - *Internet of Things*

IVA - Impuesto sobre el Valor Añadido

LIFO - *Last In, First Out*

ODS - Objetivos de Desarrollo Sostenible

PBFT - *Practical Byzantine Fault Tolerance*

PoS - *Proof of Stake*

PoW - *Proof of Work*

SHA - *Secure Hash Algorithm*

A - Amperio (unidad de intensidad)

€ - Euro (unidad monetaria)

h - hora (unidad de tiempo)

J - Julio (unidad de energía)

W - Vatio (unidad de potencia)

V - Voltio (unidad de tensión)

GLOSARIO

A

Actuador Dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado.

Algoritmo Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas.

Autenticación Capacidad de demostrar que un usuario o una aplicación es realmente quién dicha persona o aplicación asegura ser.

B

Bit En informática o teoría de la información, corresponde a un dígito del sistema de numeración binario y representa la unidad mínima de información.

F

Flujograma Representación gráfica de un proceso, ya sea para su aplicación en una empresa, en economía, programación o sistemas industriales.

H

Hardware Aquellos elementos físicos o materiales que constituyen una computadora o un sistema informático.

M

Microcontrolador Circuito integrado que contiene un núcleo procesador, memoria y periféricos de entrada/salida programables.

Microprocesador Circuito integrado programable de uso general que incorpora las funciones de una unidad de control de proceso de un computador.

N

Nodo En una red de comunicaciones, es un punto de conexión que puede recibir, crear, almacenar o enviar datos a lo largo de rutas de red distribuidas.

P

Proceso Conjunto de tareas como el ensamblaje, compilación, generación, interpretación, computación y otras acciones sobre la información en un ordenador.

Protocolo Secuencia detallada de un proceso de actuación científica, técnica, médica, etc. Conjunto de reglas que se establecen en el proceso de comunicación entre dos sistemas.

S

Sensor Dispositivo que está capacitado para detectar acciones o estímulos externos y responder en consecuencia. Estos aparatos pueden transformar las magnitudes físicas o químicas en magnitudes eléctricas.

Servidor Sistema que proporciona recursos, datos, servicios o programas a otros ordenadores, conocidos como clientes, a través de una red.

Sistema de almacenamiento descentralizado Modo de almacenamiento en el que los datos no se guardan en un solo lugar, sino que se dividen y se distribuyen a través de una red de múltiples nodos.

Sistema operativo Conjunto de programas que ayudan a la explotación de un computador, simplifican su uso y permiten obtener un buen rendimiento del mismo, actuando como un interfaz entre el usuario y el hardware del computador.

Software Conjunto de reglas o programas que dan instrucciones a un ordenador para que realice tareas específicas.