Análisis de IOTA y Ethereum para comunicación de dispositivos IoT

Gerard Capdevila Solanich
Máster Universitario en Ciberseguridad y Privacidad
Sistemas de blockchain

Profesor Alberto Ballesteros Rodríguez

31 de mayo de 2022

Índice

1. Introducción

Internet of Things (IoT) busca conectar y comunicar miles de dispositivos entre sí. Por esta razón, IoT interconecta dispositivos con tecnologías antiguas como son Bluetooth o WiFi (en la capa de transporte) o con HTTP o Websockets (en la capa de aplicación).
En consecuencia, el tema de estudio de este proyecto buscará implementar un ecosistema IoT usando unas tecnologías más modernas como son blockchain y DAG (***D****irected* ***A****cyclic* ***G****raph*).
Así, se presentaran las diferentes ventajas e inconvenientes del uso de ambas tecnologías frente a las opciones existentes en el mercado actual.

Como objetivo se buscará consolidar una comparación entre dos DLT («Distributed Ledger Technology») diferentes, una de propósito general, como puede ser **Ethereum** y una red orientada para dispositivos IoT, como **IOTA**. La red de IOTA, basada en un DAG («Directed Acyclic Graph») propio llamado TANGLE presenta varias ventajas frente a Ethereum para las redes con dispositivos IoT.

Por último, en el estudio se compararán diferentes métricas demostrando las ventajas e inconvenientes que nos permitirán priorizar el uso de una frente a la otra.

1.1. Contexto y justificación del Trabajo

La adopción y la implementación de un mundo conectado a través del Internet de las cosas, juntando personas, objetos y lugares ha sido uno de los desafíos que los ingenieros han intentado resolver durante estas últimas décadas. Las redes IoT están compuestas, entre otros, de chips sofisticados, sensores, personas, dispositivos móviles. Éstos son los responsables de mandar las informaciones a la red IoT de la cual forman parte.

Las posibilidades de este nuevo mundo conectado son extensas, muchas aún por implementar. Eso es debido a una gran multitud de problemas técnicos y de seguridad que siguen sin resolverse. Por eso, se verá que la tecnología blockchain permite resolver algunos de los temas que más riesgo presentan frente a las soluciones actuales de redes IoT.

Algunas de las ventajas incluyen un aumento de la seguridad entre las transacciones de los dispositivos conectados, la codificación de las comunicaciones con métodos de cifrado seguros, la reducción de costes en la gestión de la infraestructura IoT, la descentralización de la gestión de los dispositivos IoT, etc.

En primer lugar, con la tecnología blockchain se puede legitimar el origen de las transacciones, lo cual se logra con el uso de registros. De ésta forma, se asegura que las fuentes de información provengan de fuentes conocidas siendo un beneficio debido al elevado número de dispositivos IoT conformando la red. En segundo lugar la reducción de costes operacionales gracias a la posibilidad de transmisión de datos «peer-to-peer» sin una entidad de control que centralice la red. También se mejora la transparencia frente a otros usuarios en la red, suponiendo una ventaja frente a auditorías de seguridad. Por último, la mejora de la eficiencia de la red distribuida. Si se eliminan el organismo central encargado de recibir, validar y ejecutar los cambios en la red se tendrá una mejor velocidad en las transacciones.

En este apartado se han comentado algunas de las ventajas de estas nuevas tecnologías. Sin embargo, será durante la realización de este trabajo que se buscará comparar e implementar un sistema IoT con el uso de blockchain y DAG para demostrar cuál de ellas es mejor frente a las nuevas necesidades que presenta el Internet of Things.

1.2. Objetivos del Trabajo

Durante la ejecución del estudio se buscará demostrar las ventajas del uso de una tecnología que ha sido concebida especialmente para las redes con dispositivos IoT. Por eso el estudiante:

1. Comprenderá e usará una red blockchain de propósito general como Ethereum;

2. Comprenderá e usará una red dedicada a dispositivos IoT como es la de IOTA;

3. Realizará una comparación entre ambas redes. Las métricas que se tomaran en cuenta incluirán medidas de interés general en el ámbito de las telecomunicaciones. Algunas de las medidas que se realizarán son el cálculo de:

    a) Eficiencia

    b) Rendimiento

    c) Costes

    d) Seguridad de la información

    e) Ancho de banda

1.3. Metodología a seguir

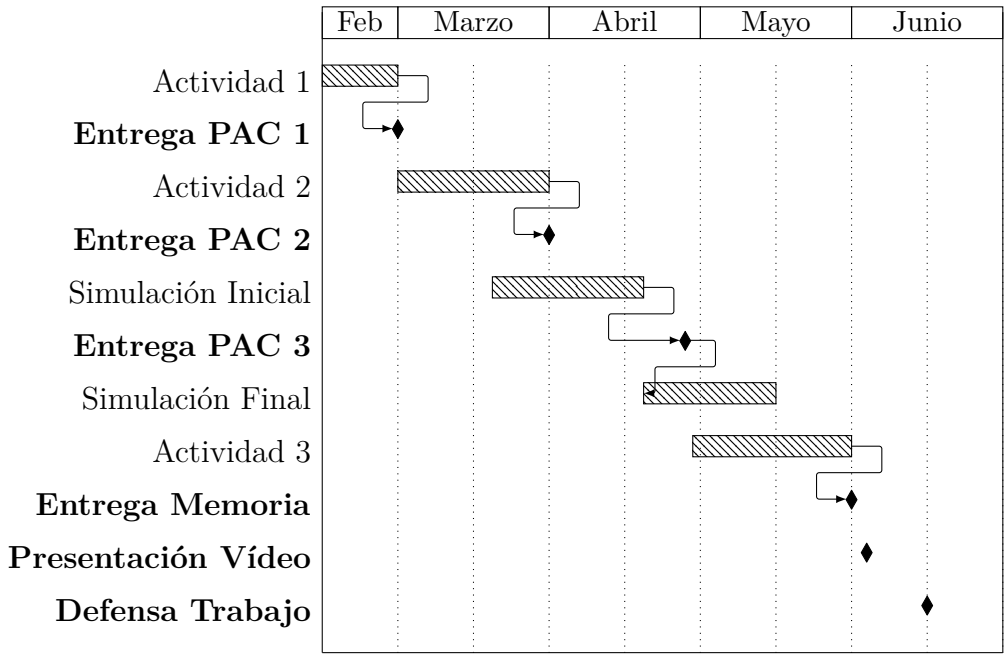
Para poder llevar a cabo dicho estudio, la simulación de un red IoT basada en Ethereum y otra en IOTA será realizada. En primer lugar, el estudiante recibe la necesidad de aprender las diferentes tecnologías que se requieren para completar el trabajo. Así, empezará por un estudio de la tecnología blockchain y la comparará de forma teórica con una red IoT tradicional.

En segundo lugar, se estudiará la tecnología de Ethereum e IOTA. Al tratarse de tecnologías diferentes entre ellas se aportarán los puntos de interés para que se pueda proceder a la implementación de un sistema real.

En último lugar, se usará la red IoT con ambas tecnologías y se comparará las ventajas e inconvenientes de priorizar una de ellas.

1.4. Planificación del Trabajo

La planificación del trabajo ha sido representada acorde al diagrama de Gantt que se encuentra a continuación y su respectiva leyenda. Además, se incluye una descripción de los objetivos a completar para cada actividad.



Actividad 1 (16/02 a 01/03)	*Organización del trabajo, creación de los objetivos, planificación del tiempo dedicado a cada punto del estudio, determinación de los objetivos teóricos del estudio.*
Actividad 2 (02/03 a 29/03)	*Comprensión, análisis y estudio al estado del arte de las diferentes tecnologías que se usaran durante el trabajo.*
Simulación Inicial (21/03 a 17/04)	*Implementación de las primeras redes con Ethereum e IOTA.*
Simulación Final (18/04 a 15/05)	*Implementación de la red que se usará para las simulaciones del estudio y toma de mediciones.*
Actividad 3 (27/04 a 31/05)	*Corrección de la memoria. El documento incluirá todas las simulaciones realizadas durante el transcurso del estudio y una explicación teórica detallada de cada uno de los conceptos utilizados.*

1.5. Recursos necesarios para la implementación del estudio

Para la realización del estudio los únicos requerimientos son el acceso a un ordenador con una distribución de UNIX como sistema operativo.

En el estudio se ha usado **Ubuntu 20.04.4** . Sin embargo, para trabajar con Ethereum e IOTA se deben instalar todas las librerías necesarias. Los componentes principales y sus versiones se incluyen a continuación:

1. **npm** versión 8.1.0 y **node** versión 16.13.0
2. **Librería web3 JavaScript** versión 1.5.3
3. **Suite Truffle** versión 5.5.11
4. **Truffle Box Webpack** versión 5.72.0
5. **HDWalletProvider para Web3 de JavaScript** versión 2.0.8
6. **Framework iota/streams para JavaScript** versión 0.1.2
7. **Wrapper de iota/client-wasm para Node.js** versión 1.0.4
8. **Binding de Node.js para iota/client** versión 2.2.3
9. **Infura**
10. **API Etherescan para Ropsten**

Para la parte de Ethereum se usará la librería Web3 de JavaScript para interactuar con el contrato inteligente que se encuentra en la blockchain. Para poder realizar esta comunicación necesitaremos también de truffle e **Infura**, además de **NodeJS** en el ordenador para ejecutar el código JavaScript.

La parte de IOTA requiere el uso de las librerías Streams y su correspondiente «wrapper» en JavaScript.

Además, serán requisito las suites de Truffle y los documentos de desarrollador de IOTA para aprender a usar todas estas herramientas. Adicionalmente, se deberá poseer una cuenta en Infura para su despliegue en la red de pruebas.

Por último, se utilizará un dispositivo propio para la ejecución de los métodos simulados, lo cual permitirá alcanzar las conclusiones finales del trabajo.

2. Estado del arte

2.1. Internet of Things

Por primera vez, en el año 1999 el término Internet of Things (IoT) fue empleado por el pionero británico Kevin Ashton para describir un sistema en el cual los objetos del mundo físico se podían conectar a Internet por medio de sensores.

El Internet de las Cosas, Internet of Things o IoT es un paradigma que combina aspectos y tecnologías provenientes de diferentes enfoques. La computación e informática ubicua, los protocolos que conforman el Internet actual, las tecnologías de comunicación que permiten comunicar los sensores con Internet y los dispositivos que permiten dicho intercambio de información. El objeto inteligente es el componente básico de la visión IoT. Al colocar inteligencia en los objetos cotidianos, se convierten en objetos inteligentes capaces no solo de recopilar información del entorno e interactuar/controlar el mundo físico, sino también de estar interconectados entre sí a través de Internet para intercambiar datos e información.

Se observa que el Internet de las cosas ha evolucionado debido a la convergencia de múltiples tecnologías. Por ejemplo, tecnologías como: la blockchain, IOTA, aplicación de sensores en productos básicos, reducción del coste de los sistemas integrados, sistemas de aprendizaje automático y uso de la inteligencia artificial para ligar dichas redes, etc. Por esta razón, durante los últimos años, las investigaciones se han basado en la escalabilidad, la seguridad y el mantenimiento de las redes IoT.

En el futuro, los objetos se encontrarán conectados entre ellos para intercambiar datos vía Internet. Por ejemplo: los vehículos, los edificios, los electrodomésticos que tenemos en casa, etc. El avenir del IoT creará largos ecosistemas de objetos conectados con una infinidad de posibles funcionalidades. Con las cifras de previsiones en 2020 de superar los 50 billones de dispositivos conectados según los expertos del sector [17].

Así, el IoT conduce los servicios actuales a una nueva dimensión con el potencial de mejorar la calidad de vida de sus consumidores. Puede ser una solución eficiente para una gran multitud de áreas de uso diario como la subministración de la energía, el sistema bancario, transporte, administración pública, seguridad, sanidad, educación y muchos otros.

Para las empresas, el sector IoT ofrece oportunidades de crear nuevos negocios. Por ejemplo, en el futuro se habla de pagos entre máquinas (M2M), procesos industriales que serán automatizados completamente sin la necesidad de intervención humana. Por ejemplo, el coche podrá realizar pagos directamente en los peajes, garajes privados, recarga de las baterías eléctri-

cas del coche en una estación de carga o en una gasolinera tradicional usando una cartera integrada en el vehículo.
Es por esta razón que con todas estas nuevas posibilidades tecnológicas a nuestro alcance surge un nuevo mercado emergente con gran potencial como es el «Internet of Things» [2].

2.1.1. Aplicaciones IoT en la industria

1. **Industria automovilística**
   Soluciones automatizadas para soluciones a larga escala, automatización de robots industriales y monitorización de plantas de producción.

2. **Ingeniería mecánica**
   Un concepto de «pagar para usar» puede ser posible. Los usuarios actúan a la vez como operadores y proveedores. Los sistemas en producción que requieran mantenimiento y monitorización a distancia pueden beneficiar de IoT.

3. **Industria electrónica**
   Una larga variedad de modelos y una optimización de los sistemas de producción requieren más flexibilidad. Se puede lograr, por ejemplo, con el uso de IoT

4. **Industria metalúrgica**
   En la industria metalúrgica, los fallos no pueden existir. El IoT industrial puede ayudar a hacer la producción más sostenible y precisa para evitar errores y documentarlos en caso de ser necesario.

5. **Agricultura, construcción y gestión forestal**
   En éstos sectores, los sensores tecnológicos IoT se usan para obtener información en tiempo real sobre los diferentes estados del terreno.

6. **Utilidades domésticas (electricidad, gas, agua)**
   Para estos trabajos utilitarios, los beneficios cuentan con la posibilidad del mantenimiento remoto y monitorización de las turbinas de viento u otro tipo de sistemas de producción de energías. El uso de tecnologías IoT conduce a una mejor eficiencia en la cadena de suministros desde la producción hasta el usuario final.

2.1.2. Visión de IoT y blockchain

Con la acumulación de dispositivos IoT y los bastos flujos de información que generan la mayoría de empresas no se disponen de soluciones prácticas para aprovechar todos estos datos. Se demuestra que una gran parte de las empresas con servicios IoT no tienen la posibilidad de aprovechar todo su potencial. Por eso, el primer cambio pasa para integrar los datos IoT entre todas las empresas y generar una base de datos completa que incluya las informaciones recuperadas. Por eso, los problemas de seguridad que plantean estas redes deben resolverse antes de poder lanzarse en ambientes de producción sensibles.

Aunque ya existen investigaciones y soluciones centradas en las redes IoT, el uso de sistemas descentralizados puede ser una de las principales soluciones para que puedan implementarse a gran escala. Un sistema de consenso descentralizado como es blockchain o DAG para poder servir como una plataforma inmutable, transparente, descentralizada, económicamente posible, segura y adaptable. Por eso, hay múltiples estudios de como usar éstas tecnologías en un modelo especifico para los dispositivos IoT.

Sin embargo, como los protocolos públicos de blockchain tienen problemas de escalabilidad y un coste elevado para realizar las transacciones no son soluciones posibles en escenarios con sistemas IoT. Como alternativa, IOTA con su tecnología DAG presentan una solución real para la gran multitud de equipos IoT que se integran día a día. IOTA ofrece un token criptográfico que está diseñado y optimizado especialmente para IoT. Ésta tecnología ofrece la posibilidad de realizar transacciones con muy baja latencia de forma descentralizada y con una infraestructura M2M y P2P («Machine-to-Machine / Peer-to-Peer»). La combinación de dispositivos y blockchains transforma el uso de éstas máquinas en sistemas económicamente independientes capaces de gestionar sus propias comunicaciones. Además de reducir costes operacionales del IoT, resolviendo problemas de seguridad y protegiendo la privacidad de los usuarios.

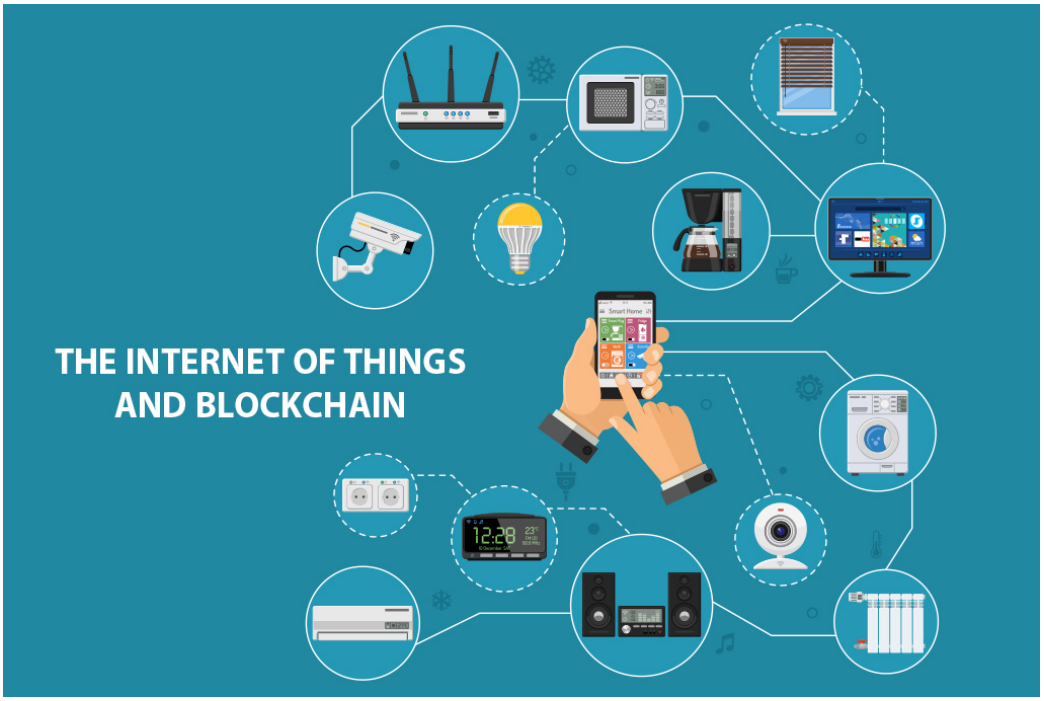


Figura 1: Fuente IEEE 2019 Disponible en: https://innovationatwork.ieee.org/the-potential-of-blockchain-for-iot/

2.2. Blockchain

El origen de la tecnología blockchain fue descrita por primera vez en el año 1991 en base a las investigaciones del profesor Stuart Haber y W. Scott Stornetta. Su objetivo fue de introducir una capa de protección para preservar la integridad en documentos digitales. Por eso, desarrollaron un concepto criptográfico para proteger las cadenas de bloques responsables de almacenar las fechas de modificación de los documentos.

En 2008, un experto en informática y criptografía conocido como Satoshi Nakamoto describió las blockchain distribuidas. A partir de las mejoras sustanciales que fueron creadas la Blockchain se popularizó con el uso de la primera criptomoneda que se conoce como Bitcoin. El diseño de Bitcoin inspiró la creación de otras aplicaciones y blockchains que presentan múltiples ventajas desde este primer protocolo.

La segunda criptomoneda más utilizada es Ethereum, desarrollada por Vitalik Buterin. La ventaja principal frente a Bitcoin es el uso de un lenguaje de Turing completo como mejora del protocolo. El propósito principal de Ethereum era de permitir una gran variedad de funciones, como los ser-

vicios de ordenador basados en aplicaciones descentralizadas (DApps) y la implementación de contratos inteligentes.

Debido a múltiples problemas de seguridad, y el robo de estas criptomonedas, se ha generado bastante controversia sobre ellas [18]. Sin embargo, debido a la gran comunidad de usuarios y la popularización del concepto «opensource» de las diferentes blockchains, los usuarios contribuyen en gran medida a mejorar los fallos de seguridad.

Nuevas versiones de estos protocolos salen de forma regular, incluyendo: mejoras en los tiempos de respuesta, reducción en el tiempo de cálculo de los bloques, reducción del consumo de los nodos que conforman la red (llamados «miners») en su proceso del cálculo de bloques, etc.

Paro antes de hablar de Ethereum y comparar las ventajas que presenta IOTA con su tecnología de computación distribuida en las aplicaciones de sistemas IoT se debe entender cómo una blockchain funciona de forma general.

2.2.1. Características principales de una blockchain

Se imagina un grupo de personas con un interés común que pretende trabajar en equipo, sin embargo no se confía en los otros equipos y además tampoco en la persona que va a gestionar la comunicación. Por esa razón, la blockchain presenta una solución a éste problema. En primer lugar, ofrece las siguientes ventajas frente una solución convencional:

- **Transparencia:** Los usuarios de una red blockchain clásica, como Ethereum, pueden ver todas las transacciones. Se trata de un sistema completamente transparente.
- **Transacciones instantáneas:** Las transacciones blockchain requieren de menos tiempo que las transacciones que involucran una tercera persona para ser validadas.
- **Descentralización:** No hay un único responsable que controle la red, todos los participantes son iguales.
- **Prueba de la transacción:** No se puede gastar dos veces el mismo recurso.
- **Prueba de inmutabilidad:** Existe una sola fuente de información disponible para todos los usuarios que asegura la integridad de cada transacción en la blockchain.

Por esa razón, una blockchain se puede considerar como un simple libro de registros. Tan pronto como una transacción existe entre un remitente y un receptor se incluye en el libro de registros como una nueva transacción que además son disponibles a todos los usuarios en el mundo entero. Una blockchain actúa como una base de datos que controlan las transacciones sin un sistema central, sin necesidad de confianza mutua y de forma completamente transparente que además incluye una prueba de la operación.

Desde que el estado general de un sistema blockchain descentralizado no se controla con una unidad central, se necesita los mineros lo hagan para nosotros. A mayor número de mineros, más estable es la red blockchain y menos probabilidades de ser vulnerable al ataque del 51 % permitiendo comprometer por completo la blockchain.

Los mineros se aseguran de mantener la blockchain en un estado idéntico general. Son los responsables de verificar si las nuevas transacciones cumplen con el protocolo y las confirman. Además son los responsables de asegurar que nadie hace una doble transacción de pago o una violación del protocolo blockchain. Por eso, todas las actividades deben ser registradas y controladas para que puedan ser rastreadas por todos los usuarios de la blockchain. Con ese proposito, todas las transacciones ocurridas en el protocolo se resumen en un bloque. Cuando un bloque llega a su capacidad límite, se cierra y se une a la blockchain. Sin embargo, la confirmación de las transacciones se hace a partir de un mecanismo de consenso, como en el caso de Bitcoin que se llama «Proof-of-Work». Para validar las transacciones con PoW los mineros deben resolver un problema matemático difícil con el uso de su poder computacional. La idea detrás del PoW es de crear la cadena más larga donde, al menos, el 50 % de los mineros la aceptan como buena.

Aún y siendo un buen avance, la tecnología blockchain tiene diferentes puntos a mejorar. La idea de construir redes descentralizadas, basadas en transacciones «peer-to-peer» sin intermediarios, no se ha implementado completamente con las opciones disponibles actualmente. Por eso, existen una serie de desventajas en el uso y la escalabilidad de éstas tecnologías. En una blockchain clásica se presentan las siguientes desventajas:

- **Coste de las transacciones:** La volatilidad del uso de estas redes es muy elevado.
- **Elevados costes de electricidad:** Para poder cubrir los costes computacionales en el minado se requiere de un consumo elevado de electricidad.
- **Tiempos más elevados para completar las transacciones:** Las transacciones se ejecutan a lo largo de unos cuantos minutos.

- **Dispositivos con altas capacidades de almacenamiento:** Almacenar todas las transacciones de la blockchain requiere una buena capacidad de almacenamiento dedicada al sistema blockchain.
- **Grupos de minado abusivos:** Grupos gigantes de minado dificultan la idea de la descentralización, ocupando más de un 51 % del poder computacional.
- **Reducción de la producción:** A mayor número de participantes menor capacidad de respuesta a todos ellos.

Por último, se traduce que la blockchain actual no es adecuada para el Internet of Things por culpa de las desventajas anteriores. Sin embargo, en el futuro no se conoce como podrán avanzar estas tecnologías, ya que su desarrollo no se detiene. La industria ya confía en el uso de blockchains centralizadas mejorando así los tiempos de respuesta de las transacciones.

2.3. Ethereum

Ethereum es una plataforma blockchain open source y descentralizada. El protocolo está compuesto por miles de nodos desplegados en todo el mundo formando una red con topología P2P. Ethereum se trata de una de las tecnologías de cadena de bloques más usadas permitiendo a los usuarios el acceso libre a su red descentralizada.

Una de las ventajas de Ethereum frente a Bitcoin es el uso de un lenguaje de programación Turing completo el cual permite crear todo tipo de aplicaciones. Así, los programadores pueden desplegar aplicaciones que usan la tecnología de Ethereum, que actuará como un tipo de ordenador universal guardando el estado de una multitud de aplicaciones en una misma cadena de bloques. Por eso, Ethereum con su red P2P en la que sus nodos contribuyen a distribuir las transacciones de los usuarios y los bloques de la cadena, podrá ejecutar los contratos inteligentes mediante el uso de la «Ethereum Virtual Machine», y guardará los resultados de las ejecuciones en la cadena de bloques. Sin embargo, si se desea usar las aplicaciones de Ethereum se requiere el uso de ***ethers (ETH)***, la moneda que sirve como sistema de cambio para poder desplegar y ejecutar contratos en la red.

Por último, Ethereum solo representa el nivel más básico de éste ecosistema. Una gran multitud de nuevas tecnologías basadas en Ethereum están surgiendo para dar respuesta a otras necesidades de las DApps, como por ejemplo, oráculos que permiten consultar datos provenientes de fuera de la cadena de bloques o soluciones de segundo nivel para facilitar el intercambio de tokens digitales de forma rápida y con bajos costos de transferencia.

2.3.1. Tecnología de Ethereum

A diferencia de lo que pasa con las arquitecturas cliente-servidor convencionales, en una red P2P los participantes se comportan iguales entre ellos y actúan a la vez como cliente y servidor respecto a los otros participantes. Dependiendo del tipo de participación que se quiera ejercer, un usuario podrá escoger entre diversos tipos de nodos. Por ejemplo la imagen siguiente ofrece una representación simplificada de los tipos de nodos [4].

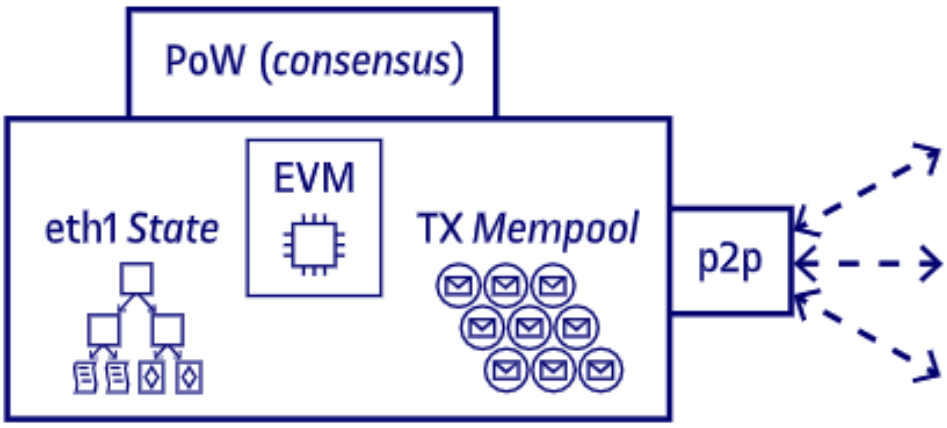


Figura 2: Fuente
https://ethereum.org/en/developers/docs/nodes-and-clients/

Se debe tener en cuenta que son los nodos de la red los que de forma conjunta ofrecen las diferentes funcionalidades de Ethereum. Como se ha comentado anteriormente, una de las funcionalidades de Ethereum es la ejecución de contratos inteligentes. Por esa razón los contratos inteligentes son lanzados en la red de Ethereum mediante los diferentes nodos, que implementan la máquina virtual Ethereum (EVM), encargada de ejecutar las funciones de los contratos inteligentes.

La ejecución de las funciones en los contratos y su resultado se guardan de forma consensuada entre todos los nodos de la red que conforman la cadena de bloques mediante el protocolo PoW (Proof-of-Work). De esta forma, la cadena de bloques mantiene el registro del estado global del sistema.

2.3.2. Estructura de los datos

En Ethereum se usa el árbol modificado de Merkel Patricia, cuya invención es un derivado del árbol de Merkle y el árbol de Patricia y algunas

mejoras. El árbol modificado de Merkle Patrica se usa como la principal estructura de datos en los árboles de recepción, árboles de estados, árboles de almacenamiento y árboles de transacciones de Ethereum [12].

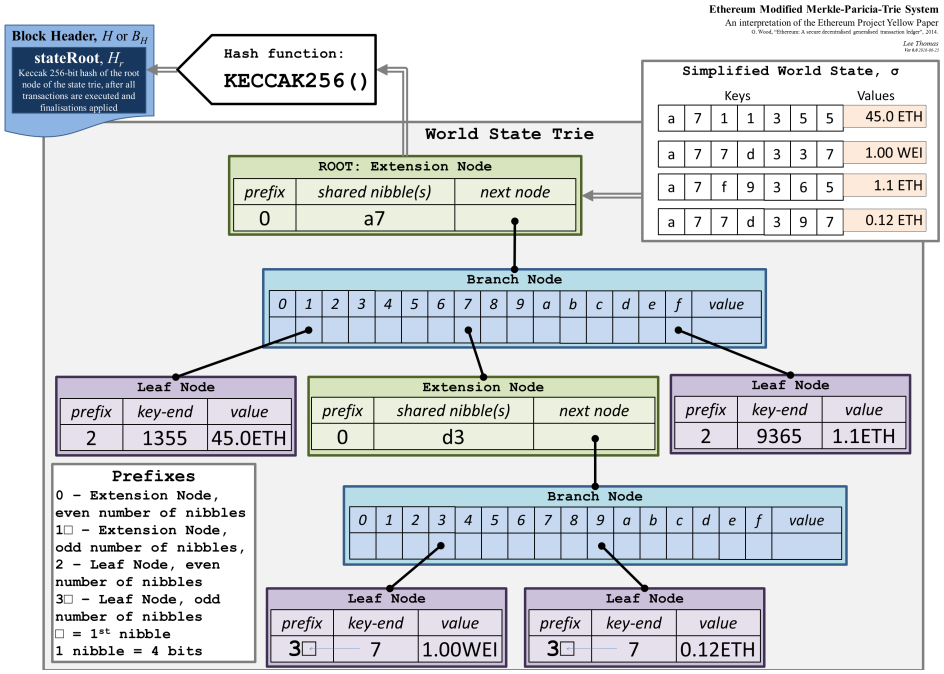


Figura 3: Fuente
https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture

En la imagen superior se muestra la estructura del árbol de Merkle Patricia. Se compone principalmente de tres tipos de nodos. Cada uno de los nodos se compone de valores hash *SHA3* con sus contenidos y el hash que se usa como identificativo.

Además, al tratarse de un proyecto *open source* existen diferentes protocolos personalizados que derivan de Ethereum. Con este método se evita que todos los clientes se vean afectados en caso de vulnerabilidad o fallos en los sistemas, permitiendo que la red pueda continuar funcionando correctamente y de forma segura.

Los nodos en la red de Ethereum pueden ejecutarse de 3 formas de cliente diferentes en función de los datos que procesan, dando lugar a los siguientes tipos de nodos:

1. **Nodo completo (branch node):** Encargado de validar y verificar los bloques, además verifica los estados. Almacena datos completos de la

cadena de bloques de Ethereum, por eso puede servir toda la información de la red y también es capaz de proporcionar datos bajo demanda.

2. **Nodo ligero (extension node):** Solo se encarga de almacenar las cabeceras de los bloques y solicita el resto de información a un nodo completo en caso de necesidad. Puede verificar la validez de los datos que recibe haciendo una comparación de las cabeceras de bloques que tiene almacenado sobre estos datos, siendo el hash obtenido a partir de todas las transacciones que se hayan ejecutado y finalizado.

3. **Nodo archivo (Leaf node):** Encargado de almacenar todos los datos del nodo completo. Es capaz de construir un archivo con los estado históricos de la red.

2.3.3. Estado de las transacciones

La información se almacena en Ethereum siguiendo los arboles de Merkle-Patricia [11]. Concretamente se guardará el estado de las cuentas, la variación de la información de un contrato inteligente o el balance de una cuenta.

En Ethereum hay cuatro árboles de estados, cada uno de ellos se construye con el árbol de Merkle Patricia y solo el nodo raíz es almacenado para ahorrar almacenamiento. Las diferentes estructuras que se encuentran en los bloques a partir de sus raíces:

1. **Receipt** (recibo de la transacción), la raíz cuadrada de la cual viene del parámetro *receipt root*.

2. **Transaction** (transacción), la raíz del cual viene del parámetro *transaction root*.

3. **State** (estado), la raíz de la cual viene del parámetro *state root*.

4. **Storage** (almacenamiento), la raíz de la cual viene del parámetro *storage root*.

Como se puede observar, Ethereum funciona como una máquina de estados, la cual dispone de un mecanismo para guardar y controlar estados dentro de la estructura de datos. El estado global del árbol almacena los estados de las cuentas los cuales representan el balance total de ésta. En segundo lugar, el árbol estado de transacciones almacena de forma inmutable las transacciones que pueden actualizar el estado global de las cuentas. Por último, el árbol de recibos permite obtener el resultado de una transacción.

2.3.4. Algoritmo de consenso

Como en la mayoría de protocolos basados en la blockchain, Ethereum usa el algoritmo de consenso basado en la prueba de trabajo (PoW). Su objetivo es el de asegurar el buen funcionamiento de la cadena de bloques mientras la red sigue de desplegada y descentralizada.

Sin embargo, respecto a otros protocolos con el mismo tipo de mecanismo de consenso (PoW), Ethereum presenta algunas diferencias en cuanto a funcionalidades y diseño se refiere. Dicho algoritmo se conoce como *Ethash*.

En cuanto a su funcionamiento, los mineros ejecutan un proceso de prueba y error compitiendo entre ellos para obtener el *nonce* para el siguiente bloque, del mismo modo que en Bitcoin. Sin embargo, Ethereum se centra en variar el parámetro *nonce* para calcular el *hash* de la cabecera de bloques tantas veces como haga falta hasta obtener un número más pequeño que un parámetro dado por la red (llamado Dificultad). En el algoritmo de Ethereum se usan otros parámetros además de los incluidos en la cabecera del bloque para añadir un coste computacional más alto. El uso de estos parámetros hace que se pueda resolver de forma eficiente usando una GPU, en lugar de CPU, como se hace en Bitcoin.

Éstos nuevos parámetros de Ethash son un conjunto de datos que forman un DAG, que aumenta con los nuevos bloques que se validan en el protocolo. Por eso, el proceso de minado consiste en tomar los fragmentos del DAG de forma aleatoria y mezclaros con los datos que forman la cabecera del bloque (raíz de Merkle de las transacciones, Dificultad, hash del bloque anterior, límite de gas, etc.) juntamente con el *nonce*. Una vez combinados, se modifica el valor del *nonce* hasta que se encuentra uno en el cual el hash de la combinación anterior está por debajo de un objetivo definido por el parámetro Dificultad.

2.3.5. Futuro de Ethereum

El futuro de Ethereum pasa por la actualización a Ethereum 2.0. Las primeras fases para migrar el protocolo a esta nueva versión ya han comenzado a implementarse. Cada una de estas fases tiene por objetivo mejorar la funcionalidad y el rendimiento de Ethereum de diferentes formas, aumentando la velocidad, eficiencia y escalabilidad, cosa que permitirá aumentar el nombre de transacciones por segundo. Uno de los mayores problemas en la implementación actual de Ethereum son la saturación y el retardo que provoca que las transacciones tarden mucho tiempo a incluirse en la cadena de bloques y las comisiones son muy elevadas.

Por eso, Ethereum 2.0 proporciona cambios fundamentales en el diseño y la estructura en comparación de sus primeras versiones. Los cambios principales son en la implementación de su mecanismo de participación y las cadenas compartidas («shared chains»):

1. **Mecanismo de participación:** Actualmente Ethereum 1.0. usa el mecanismo de consenso de PoW. La implementación de la prueba de participación (*Proof-of-Stake*, PoS) se trata de una actualización que permite mejorar la seguridad, escalabilidad y eficiencia energética. En PoS, los nodos que validen las transacciones y generen nuevos bloques deberán dejar un deposito en criptomonedas antes de poder proponer un nuevo bloque. En el caso de proponer un bloque incorrecto, este nodo perderá su deposito. Es importante destacar que este sistema, para incentivar la creación correcta de bloques y el consenso entre todos los nodos de la red, ya no se requiere la resolución de ejercicios criptográficos complejos y un alto consumo energético como pasaba con PoW. La prueba de participación se basa en verificadores y sus respectivos depósitos en ethers, en lugar de hacerlo en función de la potencia computacional de los mineros físcos.

2. ***Shard chains:*** Se trata de un mecanismo de escalabilidad que quiere mejorar el rendimiento de Ethereum. Actualmente, Ethereum tiene una sola cadena formada por bloques de consenso que ofrecen una alta seguridad y facilidad de verificación dela información. El hecho de procesar y validar cada transacción según ésta estructura de bloques consecutivos afecta la eficiencia en el procesamiento de transacciones, cosa que provoca a veces una congestión y saturación de las transacciones a la red de Ethereum. En el caso de las ***shard chains*** (o cadenas fragmentadas) [15], que son mecanismos mediante los cuales la cadena de bloque de Ethereum se divide entre si. Provocan que la responsabilidad del procesado de la información también se divida entre diversos nodos. Éstas cadenas permiten que las transacciones se procesen en paralelo en lugar de hacerlo consecutivamente, por lo tanto, un número más elevado de cadenas en paralelo comportará un aumento del rendimiento.

2.4. IOTA

La primera característica que se debe tener en cuenta es que IOTA no es una Blockchain. IOTA es un protocolo de comunicación «open-source» con tokens o criptomonedas que se usan para transferir valores. Se desarrolla y se incentiva su uso desde la organización sin ánimo de lucro *IOTA Foundation* con sus oficinas en Berlin.

El objetivo principal de IOTA Foundation es el de crear una capa de confianza para «Internet of Everything» (IoE), el cual requiere un intercambio de datos y de valores con inmutabilidad y libre de cargos. IOTA sobretodo quiere ofrecer su tecnología a todos los usuarios que quieran crear sus propias aplicaciones. Además, IOTA permite un intercambio de datos rápido, con un concepto a prueba de cambios y de forma descentralizada.

Sin embargo, surgen algunas cuestiones, como por ejemplo: ¿cómo es posible que IOTA pueda lograr a superponerse a protocolos ya existentes y que funcionan? En primer lugar, un usuario puede continuar a utilizar el protocolo TCP/IP y cifrar las comunicaciones al mismo tiempo. Pero además de poder ser víctimas de una posible interceptación de los mensajes, también se pierde la posibilidad de demostrar cuándo se han mandado los paquetes y que estos son los mismos que se han mandado a otros usuarios de la red.

Por eso, si se usa el **Tangle** de IOTA como medio de transporte, se puede beneficiar de éstas ventajas lo cual es crucial en ciertos escenarios. Por ejemplo: el uso de cadenas de suministros, oráculos, sistemas de control sincronizados,etc. En resumen, se beneficiaran las aplicaciones que requieran demostrar la integridad del emisor/receptor y que las comunicaciones entre ambos sean transparentes.

El hecho de que el protocolo de comunicaciones sea modular, nos permite una rápida y simple adaptación de actualizaciones en el futuro. Pero para que el protocolo funcione correctamente se necesitan muchos nodos. En el futuro cada coche, máquina, router, sensor, etc. puede devenir un nodo. Como más nodos existan en la red, más rápida y segura sera la comunicación.

2.4.1. Tecnología de IOTA

Para poder lograr al estado del arte de todos los aspectos que se han comentado anteriormente tenemos las siguientes características importantes:

- **Escalabilidad** Capacidad de procesar un número significativo de transacciones por segundo en redes de larga envergadura con tiempos de respuesta reducidos.

- **Producción Lean** Bajo rendimiento para que todos los dispositivos puedan participar en la red.
- **Libre de cargos** Las transacciones deben ejecutarse libre de cargos.

2.4.2. Estructura de los datos

Además de las tasas, las DLTs convencionales, como la blockchain, también hay otros factores limitantes y que por tanto, no son tan adecuados para lograr el objetivo que pretende IOTA.

Sin embargo, IOTA no presenta un hilo de bloques y tampoco requiere mineros que validen las transacciones. El núcleo de la estructura de datos de IOTA es altamente escalable, siendo posible con una sola norma: cada transacción controla y aprueba dos transacciones existentes. Ésta norma define la estructura de datos de IOTA, llamada Tangle que es un concepto matemático que se conoce como «directed acyclic graph» (DAG):

- Un conjunto de transacciones conectados con múltiples caminos;
- Cada uno de estos caminos tiene una dirección definida, lo que lo convierte en un gráfico dirigido;
- Si no se puede encontrar ningún camino que vuelva a su punto de origen el gráfico es acíclico (acylic).

2.4.3. Estado de las transacciones

En vez de limitar a un único punto para unir las transacciones como se hace en la blockchain, con DAG hay diferentes puntos donde las transacciones pueden unirse. Los usuarios pueden unir nuevas transacciones en diferentes puntos de Tangle sin esperar la confirmación de otras transacciones uniéndose entre ellas al final.

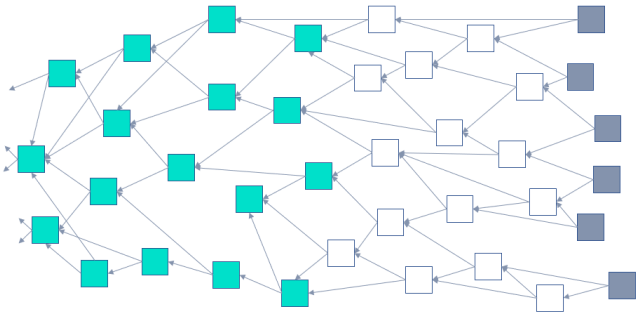


Figura 4: IOTA Beginners Guide - Estado de las transacciones en IOTA

verde = completamente confirmado / blanco = parcialmente confirmado / gris = sin confirmar

2.4.4. Mecanismo de consenso

En una red blockchain se dividen los participantes de la red en mineros y usuarios. Los mineros consumen una gran cantidad de energía de cálculo computacional para el «proof-of-work» (PoW) requerido para crear los bloques. Los mineros son incentivados con el pago de las tasas que los usuarios están dispuestos a pagar para tener sus transacciones reflejadas en un bloque.

En IOTA no existe diferencia entre mineros y usuarios. Todos los nodos pueden participar en el consenso del protocolo. Esto significa que un nodo en IOTA juega un rol muy diferente al de un minero de Ethereum o Bitcoin. Los nodos de IOTA solo realizan operaciones básicas que no necesitan de una gran capacidad de procesado. Los usuarios pueden establecer un nodo a costo mínimo y participar activamente en el consenso de la red, además de mejorar su seguridad.

Por eso, el mecanismo de consenso determina como los nodos aceptan las transacciones en que se quiere confiar, asegurando la continuidad de la red. En las versiones actuales los nodos solo deben confiar en las transacciones que están controladas y aprobadas por el coordinador. La figura del coordinador es el único aspecto centralizado del protocolo, sin embargo aún es necesario para asegurar la seguridad de la red que se encuentra al inicio de su vida.

Por último, decir que la figura del coordinador solamente puede validar transacciones. No tiene la capacidad de saltarse las normas, no puede crear, congelar o robar tokens y solo se utiliza como una herramienta de observación. La influencia del coordinador en el tangle es muy limitada, porqué el tangle de IOTA se encuentra en constante control por la parte de los otros nodos.

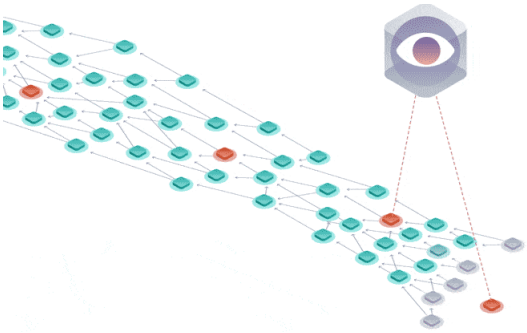


Figura 5: IOTA Foundation
https://www.iota.org/

2.4.5. Futuro de IOTA

El futuro de IOTA pasa por suprimir al coordinador. La versión Coordicide presentará un sistema sin ningún elemento de control. Por eso las nuevas características de esta versión serán acorde con el estado del arte del protocolo IOTA, manteniendo la escalabilidad, seguridad y descentralización.

Sin embargo, se deberán aplicar nuevos módulos de seguridad sin que éstos interfieran a la velocidad de las transacciones. El núcleo de la solución pasa por aplicar otra capa de mecanismo de verificación llamado «Shimmer», en el cual los nodos piden la opinión a los otros nodos para decidir cual de las transacciones puedan incluirse en el Tangle y cual debería ser eliminada.

Pero, para eliminar el coordinador un gran número de problemas deben ser resueltos. Debido a la complejidad de la solución, la versión Coordicide se ha separado en unos cuantos componentes. Esto significa que el futuro del protocolo va a ser modular. Cada módulo podrá ser reemplazado individualmente en caso de haber mejores opciones para un mismo módulo.

Figura 6: Blog IOTA Foundation
https://blog.iota.org/a-coo-less-testnet-879ad17ca1af/

2.5. Esquema de conocimiento del estudio

Por último, se incluye un esquema de conocimientos que une los diferentes puntos evocados en el estado del arte que permitirán llegar a la conclusión del estudio.

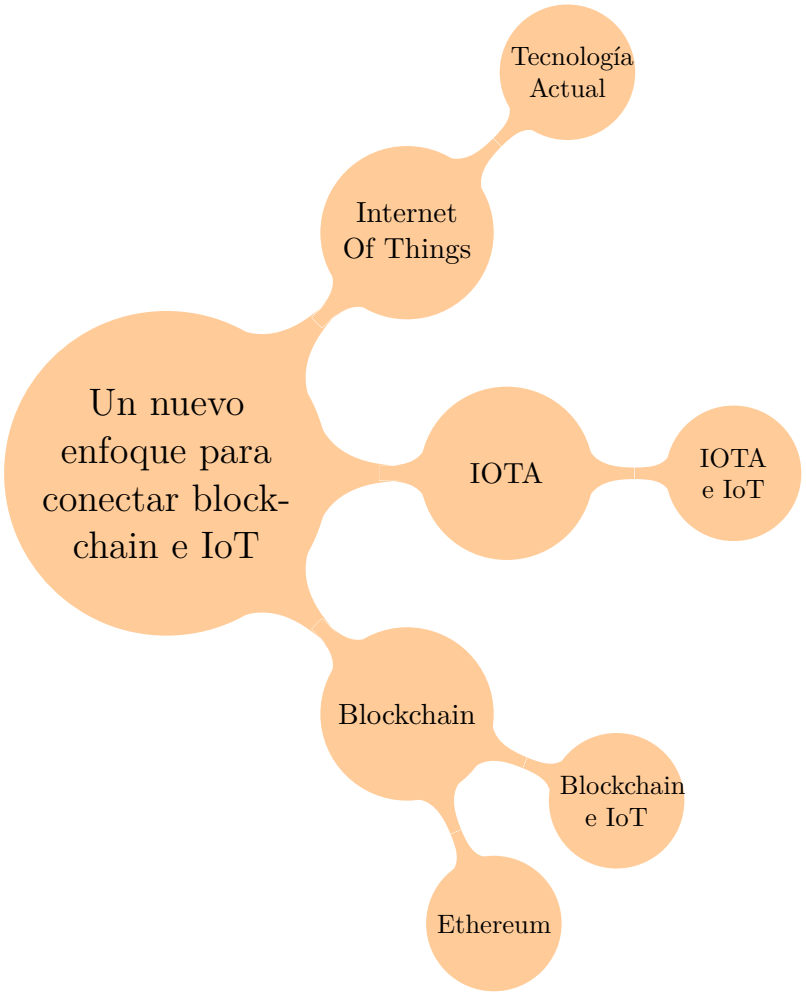


Figura 7: Mapa de conocimiento del estudio

3. Implementación IoT con Ethereum

El objetivo del trabajo es comparar dos redes IoT con tecnologías descentralizadas. Por eso, se ha llevado a cabo el estudio del protocolo Ethereum. Para ello se requiere el análisis en profundidad de los puntos que se encuentran en el estado del arte. En primer lugar, para el desarrollo de aplicaciones IoT basadas en Ethereum se deben contemplar dos puntos: «front-end» y la lógica en las cadena de bloques [5]. Las diferentes partes que constituyen el Front-end son aplicaciones Web, móvil o integradas que permiten a un usuario, mediante el uso de APIs, de comunicarse con la blockchain. Las aplicaciones Front-end son necesarias para crear interificies entre los usuarios y los dispositivos IoT que usan la blockchain. La inteligencia en las cadenas de bloques se refiere a los «smart contracts», los cuales son contratos inteligentes lanzados en la blockchain. Las ejecuciones de los smart contracts son validadas y ejecutadas en la blockchain permitiendo el uso de la descentralizada «Ethereum Virtual Machine» (EVM).

3.1. Smart Contracts

El uso de tecnologías descentralizadas que permitan realizar transacciones con un cierto grado de confianza, eliminando así la necesidad de confianza mutua entre autoridades centrales, es un punto fundamental para el desarrollo de futuras aplicaciones con datos sensibles como son los sistemas IoT. Sin embargo, algunas tecnologías en la blockchain van un paso más allá y permiten el uso de Smart Contracts. Los Smart Contracts permiten en la blockchain una ejecución programada en su misma red.

El uso de un Smart Contract es necesario para completar el estudio. Por eso, el contrato inteligente se ha desarrollado en el lenguaje de programación Solidity que es el más utilizado en Ethereum. El código es compilado y posteriormente desplegado en la blockchain.

Después de lanzarlo en la red de Ethereum, el Smart Contract es accesible solamente desde la dirección que ha lanzado la orden, siendo ésta su propietaria. Un Smart Contract presenta funciones que son abiertas a otros usuarios de la blockchain. Estas funciones exponen instrucciones listas a ser llamadas por otras cuentas del protocolo, como una API. Además, un Smart Contract recibe transacciones que van dirigidas al programa, con los parámetros especificados en el contrato. Finalmente, el Smart Contract procesará las peticiones entrantes según la lógica programada y en consecuencia realizará un evento.

3.2. Tecnologías Utilizadas

De este modo se realiza un contrato inteligente programado en Solidity que pretende simular una bombilla que se enciende y se apagada conectada en la red de pruebas Ropsten, una red de pruebas muy similar al funcionamiento de Ethereum con un modelo de consenso también basado en PoW.

El motivo por el cual se usa una red de pruebas, o *testnet*, es debido al elevado coste que supondría usar la red de Ethereum. Ropsten es la emulación más segura y parecida de la red principal, o «*mainnet*» de Ethereum. El uso de esta red con el mismo algoritmo de consenso permitirá obtener las mediciones y comparaciones más adecuadas con respecto al protocolo original Ethereum.

A continuación, se requiere el uso de Truffle e Infura para completar el funcionamiento de la simulación. Truffle es una herramienta de desarrollo, destinada a analizar y ejecutar blockchains usando la Ethereum Virtual Machine (EVM). Truffle permite compilar, enlazar y desplegar contratos inteligentes. Además, analiza de forma automatizada los contratos inteligentes, gestiona las redes de Ethereum (privadas o públicas) y proporciona una consola interactiva para establecer una comunicación directa con el contrato [6]. A partir de Truffle se publicará el contrato inteligente responsable de imitar el comportamiento de la bombilla.

Con todo este aún se requiere el uso de Infura [7]. Infura proporciona acceso instantáneo a las redes Ethereum e IPFS mediante el uso de HTTPS y WebSockets. Infura permite una conexión fácil a la Web3.0 evitando la configuración o la preocupación de mantener nuestra propia infraestructura a este nivel. Metamask, por ejemplo, es una herramienta basada en Infura y que integra todas las funciones de ésta, ofreciendo todas las funciones necesarias para realizar transacciones con las Dapps.

3.3. IoT y Ethereum

Solidity es un lenguaje de programación de alto nivel orientado a programar los contratos inteligentes. Su sintaxis es similar a la de JavaScript y está enfocado específicamente a la Máquina Virtual de Ethereum (EVM) que permite la recuperación de las cuentas de los usuarios, la ejecución de transacciones, la interacción con los Smart Contracts, etc [8].

Además, es necesario complementar el funcionamiento del Smart Contract en el momento de interactuar con Ethereum. Para eso existen librerías en JavaScript y Python. Con ellas se puede interactuar con un nodo local o remoto de Ethereum usando HTTP, IPC o WebSockets. Existe documentación detallada para el uso de estas librerías [13] y una gran comunidad de usuarios que sigue aportando nuevas actualizaciones.

En resumen, se aprovecha el potencial de la máquina virtual de Ethereum (EVM) gracias al uso de Solidity, que permite las interacciones con la blockchain y se mejoran sus capacidades con el uso de las librerías Web3 de JavaScript.

3.3.1. Análisis de requerimientos

Para la implementación de un sistema IoT basado en Ethereum en el cual se puedan analizar las métricas comentadas en el estado del arte para poder completar el sujeto de estudio se empieza por desplegar una instancia Truffle. Para la realización del estudio y las simulaciones se ha basado la primera versión de la bombilla conectada en el Webpack Truffle Box.

Se requiere que la simulación del objeto conectado proporcione un flujo de información de forma periódica sobre el estado de la bombilla. Además, el smart contract de presentar una función para encender y apagar la bombilla usando la blockchain de Ethereum y en consecuencia la blockchain recibirá una actualización de su estado.

El primer paso pasará por compilar y desplegar el contrato inteligente en la blockchain. Para eso existen diversas posibilidades. Para el estudio se ha escogido el uso de Truffle, responsable de desplegar el contrato en la red de pruebas Ropsten. Como se ha visto antes, no se puede realizar la simulación de una blockchain desplegada en una red local porque los tiempos de respuesta no serian realistas.

En segundo lugar se incluye el contrato inteligente programado en Solidity.

Por último, la creación de una pequeña interfaz web en la cual se pueden observar los parámetros de interés como la dirección propietaria del contrato inteligente. Las dos funciones que permiten encender y apagar la bombilla y por último el estado actualizado de la bombilla.

Con todos estos puntos finalizados se podrá desplegar la aplicación descentralizada con la instrucción Truffle *truffle migrate*. Sin embargo, se requiere el acceso y la gestión de las comunicaciones entre las redes Ethereum e IPFS, siendo posible gracias al uso de Infura.

3.3.2. Implementación del sistema IoT

La primera parte es la programación en Solidity. Se ha desarrollado con el siguiente planteamiento:

1. Una función *view*, que no consume GAS en la blockchain, permite acceder al estado de la bombilla;

2. Una función para encender y otra para apagar la bombilla, que consumirán GAS debido al cambio de estado de una variable en la Blockchain.

La implementación de la bombilla conectada para realizar una simulación se desarrolla de la siguiente forma. En primer lugar, se lleva a cabo el desarrollo de un Smart Contract programado en Solidity. Éste se compone de las funciones principales que permiten interactuar con la blockchain y analizar los parámetros listados anteriormente. Además, las funciones serán accesibles desde una interfaz Web implementadas con las tecnologías HTML y JavaScript con la librería Web3.

Antes de realizar la migración del contrato inteligente, mediante el uso de Infura, se da por supuesto que la aplicación descentralizada esta lista para ser desplegada y el contrato compila sin errores.

Antes de poder usar Infura se requiere a sus usuarios que se registren en la plataforma obteniendo un nuevo identificador de proyecto (*Project ID*) y Secreto, copiando las llaves de forma segura y seleccionando el *endpoint* de red adecuado.

El siguiente paso será de modificar el archivo *truffle-config.js* para usar la librería HDWalletProvider previamente instalada y responsable de ofrecer todas las configuraciones necesarias para desplegarse en Ropsten. El acceso al documento de configuración puede encontrarse en los anexos 8.1.

A continuación, se debe instalar y configurar Metamask en el navegador. Se trata de una extensión que permite gestionar las carteras digitales conteniendo los Ethereum de cada una de las cuentas. Para lograr el pago se usará un Faucet de la red Ropsten para atribuir Ethers a la cuenta de pruebas. Sin embargo, estos Ethers no tienen ningún valor en el mundo real y se limitan al uso de la testnet de Ethereum [14].

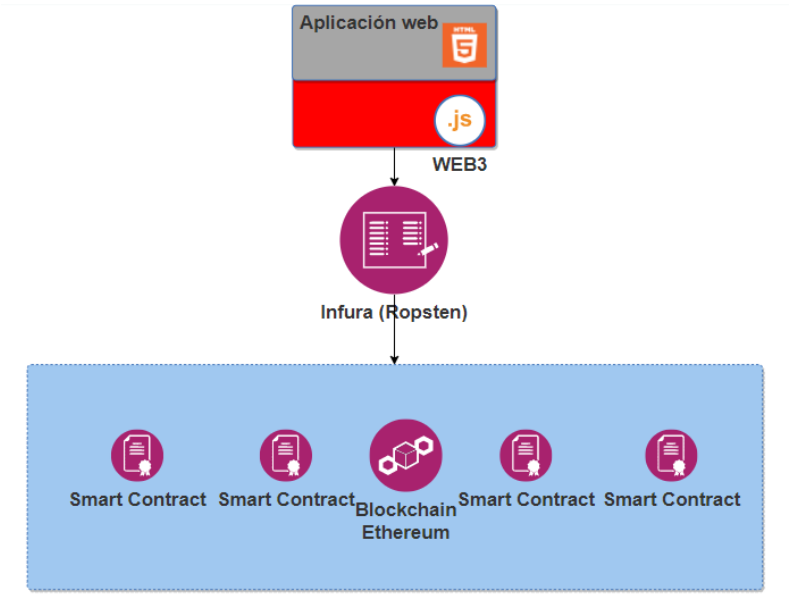


Figura 8: Capas de la implementación IoT con Ethereum

3.3.3. Análisis del Smart Contract

El primer paso para completar la última parte es el despliegue del contrato inteligente en la red Ropsten. Como las simulaciones deben realizarse en una testnet que simule la blockchain de Ethereum se debe modificar el archivo *truffle-config.js*. En el archivo se debe contemplar un valor de gas suficiente para soportar el lanzamiento del smart contract. Sin el suficiente gas de margen para ser desplegado el contrato se quedaría en espera hasta que un minero acepte el valor máximo de gas que se ha indicado en la configuración. A continuación se incluyen los valores en el momento de publicar el smart contract:

```
    ropsten: {
      provider: () => new HDWalletProvider(mnemonic,
      "https://ropsten.infura.io/v3/KEY",0),
      network_id: 3,
      gas: 4700000,
      gasPrice: 200000000000,
      timeoutBlocks: 200
}
```

Figura 9: Fragmento de la configuración de Ropsten en truffle-config.js

Cuando se obtiene la confirmación de diversos bloques en la blockchain el smart contract quedará publicado en Ropsten. La cuenta que dispone de suficientes Ethers es 0xcFD8169B72465a16240f1EAf35248F3Ea89e7850.

Además, el repositorio completo con los archivos para hacer funcionar la Dapp se puede encontrar en los anexos 8.1. Así, para situar al lector, se incluye la estructura en árbol del proyecto:

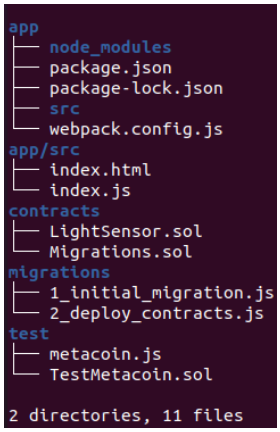


Figura 10: Estructura del proyecto IoT con Truffle

Por otra parte, una vez el contrato esté publicado en Ropsten deberemos albergar la aplicación en el ordenador de pruebas. Con la ejecución de NodeJS se obtiene el siguiente resultado de la Dapp:

Figura 11: Aplicación web usada para interactuar con el contrato inteligente

Por último, antes de realizar las mediciones se debe comprobar que el contrato funciona correctamente. Las interacciones con las diferentes funciones pueden ejecutarse desde la interfaz web que se ha programado. Posteriormente las transacciones de los cambios de estado de la blockchain se recuperan desde `https://ropsten.etherscan.io/`. Algunas de las transacciones que se han usado para probar el contrato inteligente se incluyen a continuación y podrán ser consultadas en la página de **Etherscan**:

- Creación del contrato inteligente:
  `0xC93c140e9a8435C91513E8c3E6Cfb9F98231a337`
- Transacción en el contrato inteligente:
  `0x7b37e5e043fbd65c9c3a3f4a89f18648f9d8d4867b348d9a29754e0464211191`

De igual importancia, hay dos formas para interactuar con el contrato. Siendo la segunda opción una forma de comunicar directamente con la consola de *truffle* (desde el terminal de nuestro sistema operativo).

De ahí, el mismo repertorio donde se ha migrado el programa, se ejecuta: `truffle console --network ropsten`.
Truffle permite con esta instrucción acceder e interactuar con el contrato inteligente desplegado sin pasar por la aplicación descentralizada y Metamask como se ha mostrado en el caso anterior.

Por último, se incluye un esquema de como los diferentes elementos de nuestro proyecto (Infura, Truffle, framework Web3 de JavaScript y Solidity) se unen y se establecen como los actores principales de la red:

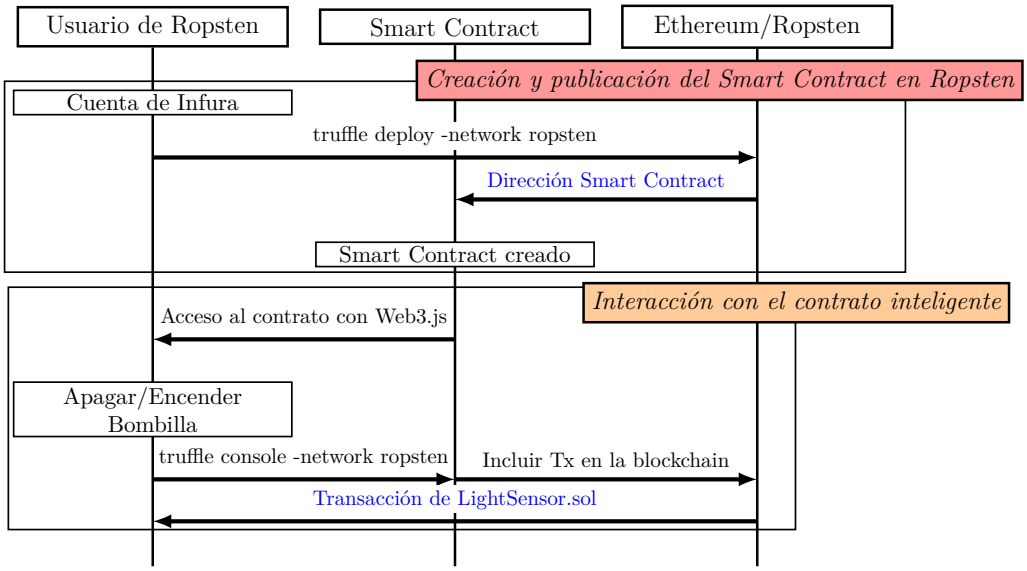


Figura 12: Esquema de procedimientos para controlar una bombilla conectada en Ethereum

3.3.4. Cálculo de métricas y análisis de los resultados

Desde que la propuesta del trabajo consistía en la integración de las criptomonedas para el control de dispositivos IoT, la planificación de los experimentos es la etapa mas crucial del proceso. Además, como no se pueden aplicar cambios en la blockchain, los análisis serán destinados a demostrar la viabilidad de usar Ethereum o IOTA [16]. Se focalizará en dos grupos de experimentos, el primero que buscará detectar que pasa en caso de subidas de carga y el otro realizará las pruebas de rendimiento.

El primer grupo será el encargado de identificar los problemas funcionales, la calidad del canal, su disponibilidad, robustez y estabilidad. El segundo grupo, buscará calcular los tiempos de respuesta, ancho de banda y recursos de utilización. Por la arquitectura propuesta se podrán obtener estos valores a través de las mediciones del tiempo de confirmación entre las transacciones. Para eso hay que obtener los valores de las transacciones antes de analizar los resultados. Se usará la API de Etherscan en su versión gratuita para automatizar la obtención de datos.

Después de consultar las primeras transacciones con el contrato inteligente y las informaciones que devuelve se ha decidido realizar las pruebas que se presentan a continuación:

Gas consumido

En primer lugar tenemos el efecto económico. La moneda digital Ethereum tiene una propiedad que la distingue de IOTA y que brinda a sus usuarios la posibilidad de controlar el precio ofrecido por las instrucciones que se quieren ejecutar (GAS) en las transacciones. El GAS es un parámetro controlado por el usuario y su variación puede influenciar el precio final pagado así como el tiempo de espera antes que una transacción sea confirmada en la red. Existen a disposición de sus usuarios librerías que informan del precio medio de la red al momento de la transacción.

Por eso, se han realizado los análisis en dos momentos diferentes. Durante el fin de semana el precio medio pagado para el GAS en Ropsten era de *673.24 Gwei*, equivalente a $673{,}24 * 10^9$ WEI o lo mismo que $673{,}24 * 10^{-9}$ ETH con un precio de mercado en mayo de 2022 de 0,0000018€ por Gwei y durante la semana y a primera hora del domingo el precio medio del GAS era mucho menor, de aproximadamente *2.5 Gwei*. Los precios medios del gas se han obtenido en base a la siguiente página que permite extraer gráficas de los distintos valores de GAS:

Figura 13: Valor medio del GAS obtenido el domingo de la realización de los experimentos

Figura 14: Valor medio del GAS durante el sábado de la realización de los experimentos

Para la red de Ethereum las hipótesis eran que, a menor precio que se pague para el gas mayores los tiempos de espera hasta que se confirme la transacción en la blockchain.

Entonces, para realizar las medidas se decidió ejecutar las transacciones con los siguientes valores: 2 Gwei, 12 Gwei, 24 Gwei, 100 Gwei, 500 Gwei, 1000 Gwei y 10000 Gwei.

La misma hipótesis hace pensar que la selección del precio del gas podría tener un mayor impacto en los tiempos de repuesta máximos para completar las transacciones, pero sin contemplar el tiempo mínimo. El motivo de no contemplar el tiempo mínimo es porqué se encuentra directamente influenciado con el tiempo mínimo necesario que se tarda en incluir un nuevo bloque en la red.

GAS(Gwei)	Tiempo1(s)	Tiempo2(s)	Tiempo3(s)	Tiempo4(s)	Tiempo5(s)
2	46	14	52	58	61
12	33	27	36	35	44
24	35	33	46	45	37
100	120	34	40	38	60
500	7	50	70	42	39
1000	34	35	53	32	33
10000	32	33	6	22	25

Figura 15: Tiempos empleados a validar 5 transacciones

Los datos anteriores son los diferentes tiempos que ha tardado en validarse una transacción. El procedimiento se ha realizado 5 veces y vemos que en el caso de pagar el GAS a 2 Gwei tardamos una media de 46 segundos a obtener la respuesta. Si se paga el GAS a 10000 Gwei la media es 24s, lo que significa que es un 52 % inferior. Añadir que se han realizado los cálculos en un momento de bajo uso de la red y que, durante las diferentes transacciones mandadas una por una, el 100 % de los paquetes han sido procesado por la red.

A continuación, la siguiente tabla muestra que existe una gran diferencia en el precio pagado por parte del usuario de Ethereum. La diferencia entre establecer el gas a 2 Gwei y a 10000 Gwei es de 5000 %. Se compara así los precios para cada una de las configuraciones anteriores, que empiezan a aproximadamente 0,8€ por el conjunto de 5 transacciones y finalizan con un valor de 3206€.

Gwei	2	100	500	1000	10000
ETH	0.000439094	0.008755235	0.0877765	0.153697	1,77246
€(1Eth=1800€)	0.796021	15.85	159.10	278.66	3206.56

Figura 16: Comparación de los diferentes precios para realizar 5 transacciones

Transacciones enviadas vs confirmadas

En este caso se busca mandar el máximo número de transacciones en el intervalo de un minuto para saber cuantas de éstas transacciones se procesarán correctamente. La importancia de este análisis, recae en el hecho de que un usuario probablemente requiere de inmediatez cuando quiera interactuar con un dispositivo conectado.

Por esta razón, la siguiente gráfica muestra el porcentaje de transacciones que se llegan a validar en 1 minuto para una función de «Encendido/Apagado».

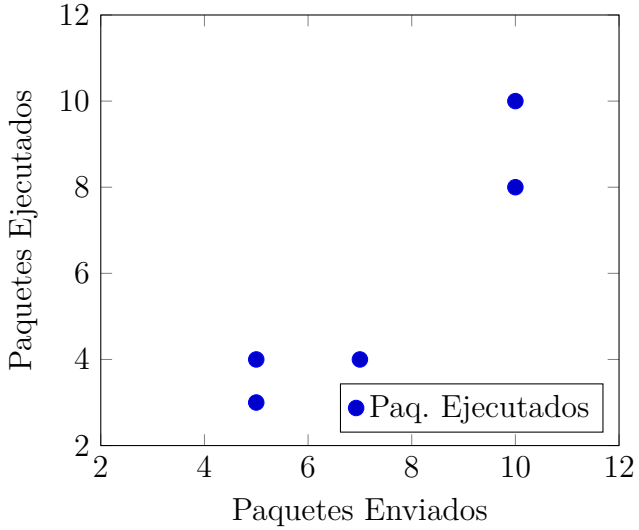
Para poder ejecutar las llamadas una detrás de otra se prepara un conjunto de 10 terminales con la consola de Truffle conectada a Ropsten.

De forma síncrona se ejecutan todas las transacciones. Existirán el mismo número de llamadas para encender la bombilla que para apagarla. Sin embargo, en algunos momentos, se observa que aparecen algunos errores y no se procesan todas las transacciones en la blockchain. El primero de esos errores indica que el *Nonce* es demasiado pequeño, ya que seguramente significa que aún hay mensajes que siguen en espera de ser procesados.

Se observa así los siguientes resultados obtenidos en los 5 intentos realizados:

Intento	Enviados	Ejecutados
1	7	4
2	5	3
3	5	4
4	10	10
5	10	8

Figura 17: Resumen de los 5 intentos realizados



Los resultados anteriores muestran que de todos los intentos que se han realizado solamente se han mandado a la blockchain un **74 %** de las llamadas a las funciones de «TurnON» y «TurnOff». De todas ellas, se han ejecutado

el **78,37 %**. De aquí, podemos extraer que Ethereum no funciona de forma asíncrona y que en caso de que dos personas quieran acceder a la bombilla desde terminales distintos al mismo momento, no siempre se podrá asegurar la disponibilidad del servicio. El hecho de haber seleccionado una transacción en una consola de Truffle no nos asegura que todas se ejecuten correctamente una detrás de otra.

También se observa que, sin especificar ningún valor de GAS, el comportamiento de la Blockchain puede cambiar. Truffle se adapta para que en los momentos en los que esté más saturada, debido a débitos mas altos de utilización, ofrezca un mayor precio para el GAS. También, con los datos de antes vemos que la diferencia del precio del GAS puede llegar a ser de hasta un 26929 % superior. Puede parecer una barbaridad, pero se han cogido valores extremos para poder apreciar les grandes diferencias que existen en la red de pruebas. Por esa razón en momentos específicos existe la posibilidad que la transacción nos devuelva otro error. En este caso la petición habrá fallado por un problema de "fondos insuficientes"debido a las altas comisiones que se pueden llegar a pagar y a un precio del GAS más elevado de lo normal. Se precisa que los valores estándares del GAS lo suelen situar a los 60 Gwei.

Existe también el caso donde las 10 transacciones se han realizado correctamente. Durante el ensayo se ha encontrado un porcentaje de uso particularmente bajo de la red Ropsten, situando así el precio del GAS en mínimos. Esté efecto ha permitido completar todas las transacciones sin ningún error.

Tiempo que se tarda a mandar los paquetes en función del tamaño en bytes del mensaje

El último punto del estudio busca analizar si las transacciones con un tamaño variable en la blockchain añaden tiempos de espera superiores. La hipótesis es que los paquetes tienen una estructura predefinida como se observa en los *Papers de Ethereum* y que si el usuario dispone de suficientes ethers para pagar el gas de la transacción será suficiente para ejecutar la petición con prioridad. Para eso se ha seleccionado las 10 últimas peticiones enviadas al smart contract y se ha comparado su tamaño en Bytes. Se colaciona la información con los datos teóricos y se determinará así si los paquetes se ven afectados por el contenido del contrato inteligente.

A través de mirar los paquetes en el navegador se observa que son fijos con las variables definidas en el SmartContract. Por eso se economiza GAS cuando la mayor parte del procesado de la información se realiza al lado del cliente, es decir, en el código con Web3 de JavaScript.

4. Implementación IoT con IOTA

La concepción de un sistema IoT basado en el protocolo de comunicaciones IOTA permite serías ventajas según se demuestra en el estado del arte. Es por este motivo que siguiendo la documentación del wiki oficial de IOTA se implementa una simulación de un dispositivo IoT conectado a Tangle [9]. Para poder comunicar con la red de IOTA, el dispositivo debe conectarse e interactuar con un nodo. La tarea puede ser simplificada gracias a las diferentes librerías que se encuentran disponibles en la página oficial de la organización IOTA.

Asimismo, cuando un dispositivo realiza la conexión inicial a un nodo, la librería de IOTA no tiene ningún estado. Las operaciones solo realizarán la información que el cliente manda durante una petición y no tienen ningún efecto en el software a parte de devolver un valor. El usuario se encuentra en control total de la información que se genera en la aplicación.

IOTA proporciona a sus usuarios unas librerías escritas en Rust y ofrece las mismas funciones en varios lenguajes de programación conocidos: JavaScript, Python, Java y Wasm. Durante este trabajo se ha priorizado el uso de las implementaciones en el lenguaje Python, manteniendo el mismo tipo de comunicación con el API. Por eso, la API sigue siendo igual de efectiva en todos los lenguajes listados anteriormente.

4.1. Aplicaciones en la red de IOTA

Las aplicaciones se comunican con *iota.rs* a través de Rust o en Python (lenguaje de programación del estudio). La librería de iota.rs convierte las peticiones como si fuera una «API REST» [10], transfiriendo una representación del estado del recurso requerido a quien lo haya solicitado, en este caso el nodo de IOTA a través de Internet. El nodo, en su caso interactúa con el resto de la red IOTA, el cuál puede ser la «mainnet» o la «testnet».

En el caso de la simulación presentada en el estudio, donde se simula un dispositivo IoT, se obtiene el siguiente esquema:

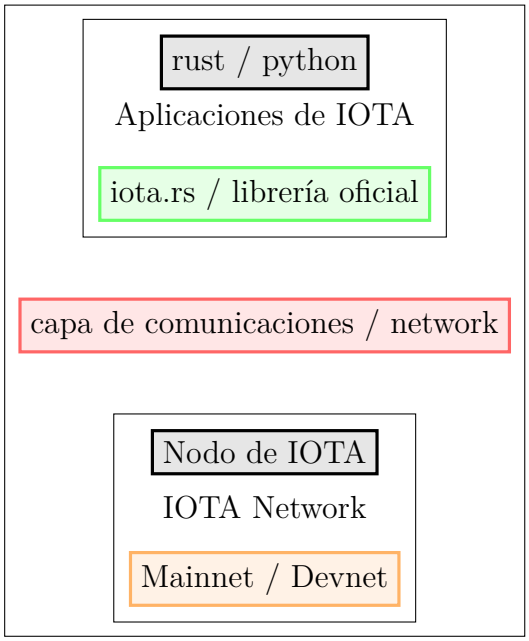


Figura 18: Estructura de un proyecto IOTA

4.2. IOTA Streams

Los «streams» funcionan como una herramienta que permite organizar, estructurar y navegar de forma segura entre los datos de Tangle. Los Streams organizan los datos ordenándolos de forma uniforme y con una estructura interoperable.

Además, streams es un framework que permite implementar protocolos criptográficos para su uso en mensajería. Streams se compone de un protocolo ya construido llamado *Channels* para mandar mensajes con un control de acceso entre dos o mas miembros de Tangle y adicionalmente permitiendo a los desarrolladores de construir sus propios protocolos según sus necesidades.

El protocolo *Channels* presenta una gran cantidad de funciones ya implementadas como se verá a continuación, sin embargo en el caso de que la implementación actual no sea ideal se podrá modificar el programa para construir su propia implementación.

Los canales pueden estructurarse en una multitud de formas siguiendo cualquier combinación arbitraria entre autores y suscriptores, aunque cada canal solo puede ser albergado por un único autor de la instancia.

El protocolo *Channels* proporciona las herramientas de alto nivel necesarias para los autores y suscriptores de tal forma que pueden generarse e interactuar con el Tangle de forma sencilla. También presenta una gran can-

tidad de nuevas funciones, como: el hecho de que un autor pueda firmar un mensaje y que una gran cantidad de suscriptores puedan recibirlos y que ellos mismos también puedan publicar mensajes no signados. Streams está compuesto de las siguientes características:

1. **Autores** Un autor de canal es responsable de la generación de un nuevo canal con la configuración de la estructura de datos de dicho canal (una sola rama vs múltiples ramas). Un autor del canal tendrá la capacidad de establecer restricciones de acceso a las diferentes ramas dentro de la estructura del canal, además de aceptar y gestionar la subscripción de los mensajes de la parte de los diferentes usuarios.

2. **Suscriptores** Un subscriptor de canal es cualquier usuario en un canal que no sea el autor de éste. Un suscriptor puede ser generado independientemente sin verificación alguna por un autor, pero si quiere escribir en una rama, o procesar cualquier flujo privado, necesitaran suscribirse en el canal y obtener la aceptación del autor para procesar dicha suscripción. Un suscriptor también puede usar claves pre-compartidas en cambio de suscribirse como forma de interacción con un *stream* sin tener que establecer un proceso de suscripción.

3. **Branching** Las ramas pueden ser definidas como cualquier grupo secuencial de mensajes de forma que sean unidas o asociadas al anuncio del mensaje. Las ramas pueden ser creadas bien con un paquete de mensajes firmados o con un mensaje con clave para comunicaciones públicas y privadas. Un canal puede asumir dos formas diferentes:

   - ***Single branch***: una secuencia lineal de mensajes con cada mensaje asociado al anterior;
   - ***Multi branch***: una secuencia de mensajes que no funciona en la unión secuencial de mensajes. Cuando se genera un canal, el autor debe decidir si el canal usará una sola rama o una multitud de ellas, de esta forma se informará de que forma la instancia de los Streams debe comunicar. Los suscriptores también serán informados como proceso de transmisión de mensaje, entonces las instancias conocen la forma de secuencias apropiadas.

4. **Keyloads** Un mensaje de *keyload* permite gestionar y restringir los accesos que permiten a un autor de descifrar cualquiera de los mensajes que lo siguen. Existen dos formas para generar un mensaje de keyload:

   - **Claves públicas de suscriptores:** Durante el proceso de suscripción de los mensajes, las claves públicas estarán enmascara-

das y se entregarán al autor para que sean almacenadas en su instancia. Entonces, el autor podrá especificar cual de éstos usuarios puede acceder a los mensajes subsecuentes incluyendo la clave pública en un mensaje de *keyload*.

- **Claves pre-compartidas:** Una clave predefinida es compartida entre los usuarios por otros medios que el proceso de suscripción comentado arriba. Estas claves pueden ser usadas para acceder a flujos restringidos sin necesidad de realizar el proceso de suscripción.

5. **Secuenciación de mensajes** La secuenciación de los mensajes es una metodología construida dentro del protocolo de los Streams que permite generar los identificadores de los mensajes de forma secuencial sin importar la forma del canal.

4.3. IoT e IOTA

Mediante los conceptos presentados anteriormente, se busca implementar una conexión con la «testnet» de IOTA para poder aprovechar el potencial del framework Streams. El framework dispone de *wrappers* que permiten de crear el programa en tres lenguajes de programación diferentes.

Aunque en Ethereum se ha programado un smart contract para controlar la bombilla conectada y existan nuevas versiones que intenten simular la creación de contratos inteligentes para Tangle, el objetivo de este estudio no es de explorar estas nuevas capacidades sino de establecer una conexión entre nuestra bombilla y el protocolo de IOTA.

Para eso, durante la concepción del estado del arte se ha hablado de las diferentes ventajas que presenta *DAG* y no blockchain para interconectar millones de dispositivos que deben estar visibles entre si. Como se observa en las métricas de la primera simulación, en el caso de usar una blockchain de propósito general no se puede obtener la inmediatez en la recepción de los paquetes. En consecuencia, no es la solución ideal para ser el protocolo responsable de transportar la información y ponerla a disposición de los otros usuarios.

Por este motivo, uno de los mayores objetivos de la comunidad de IOTA ha sido su vulgarización para su uso en medios con sistemas IoT. Entonces, siendo así el sujeto de estudio que se quiere demostrar, IOTA aporta grandes ventajas frente a otras tecnologías ya existentes.

4.3.1. Análisis de requerimientos

Para realizar la simulación del dispositivo IoT y el cálculo de las métricas vistas en el estado del arte se busca obtener un sistema de lo más parecido posible a la aplicación descentralizada basada en Ethereum.

Sin embargo, del hecho de usar dos protocolos diferentes, no se puede hacer una copia idéntica del anterior. Pero se buscará programar una aplicación que actualice el estado de la bombilla cada 30 segundos y de tal forma mandar un flujo de datos constantes provenientes del objeto conectado hacia el Tangle de IOTA.

Para lograrlo, se ha usado el framework *Streams* de IOTA del cual se ha hablado antes, permitiendo mandar mensajes con información cifrada. Como existen diversas opciones se escoge el uso de sus librerías en WASM debido a la posibilidad de integración en un código JavaScript.

En esta segunda simulación los resultados de la interacción entre la bombilla y Tangle serán presentados solamente en formato de respuesta en el terminal. También podría ser en la consola de un navegador web, o directamente en el terminal de Ubuntu donde se ejecuta NodeJS.

Por último, la forma de hacer el cálculo de las métricas será en base a las respuestas que nos devolverán la funciones implementadas en el propio código. Entonces los resultados deberán presentarse de forma que se pueda comparar con los experimentos anteriores.

4.3.2. Implementación del sistema IoT

La primera parte consiste en instalar todos los requerimientos y librerías necesarias para poder usar el complemento Streams de IOTA. Además, como se trata de código en JavaScript la única forma de ejecutarlo en local será con NodeJS siendo necesario su uso para ejecutar un programa JavaScript en el lado del servidor.

Vinculado a esto se ha priorizado el aprendizaje de como usar el «*wrapper*» de Streams en WASM para generar las comunicaciones entre la bombilla y el Tangle durante la realización del trabajo. WASM corresponde a la abreviatura de WebAssembly, siendo un formato de código binario portátil. Su objetivo es la ejecución íntegra en el navegador (lado del cliente) de scripts. Se trata de un lenguaje de bajo nivel, diseñado inicialmente como formato destino en la compilación de C y C++.

Es necesario resaltar que la complejidad de programación del código fuente no presenta grandes dificultades. El programa limita su ejecución a mandar un mensaje cada 30 segundos y obtener los detalles de la transacción. Sin embargo, no hay que olvidar que IOTA presenta mejoras y nuevas funciones de

forma regular y por ese motivo el programa podría haberse implementado de diferentes formas.

Para terminar, se incluye un esquema de la arquitectura de la aplicación:

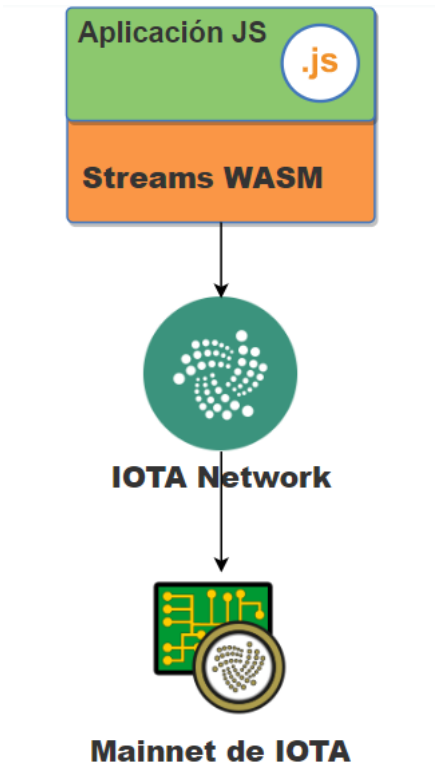


Figura 19: Arquitectura de la aplicación IoT con JavaScript para interactuar con la mainnet de IOTA

4.3.3. Análisis del código de IOTA

Antes de poder realizar las mediciones con la red principal de IOTA se comprueba que el programa funciona correctamente. El código incluido en el repositorio accesible en los anexos 8.2 se puede representar en forma de diagrama. Su procedimiento para mandar mensajes y comunicar con la red de IOTA es el siguiente:

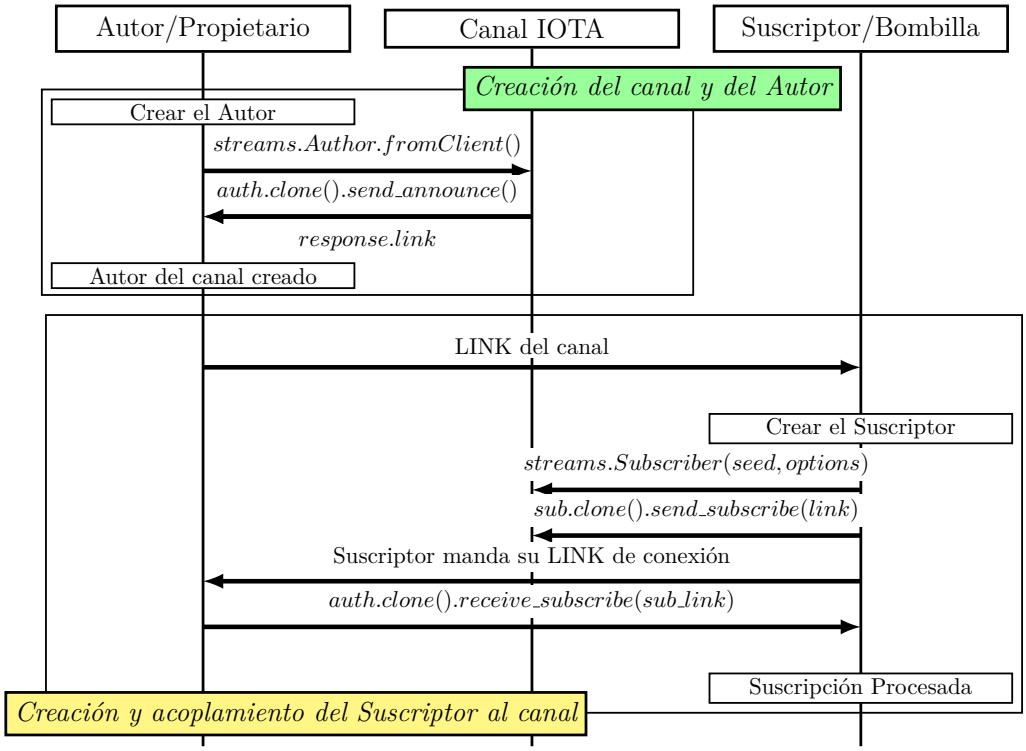


Figura 20: Esquema de procedimientos para establecer un canal con IOTA

En el esquema anterior se observan los diferentes procedimientos que permiten inscribir a un suscriptor en el canal de IOTA y sus respectivas funciones en código JavaScript que lo permiten. Así, podemos resumir el modelo de comunicación que propone Streams de IOTA.

En primer lugar, se configura un canal con las diferentes opciones disponibles. Posteriormente, se asocia la configuración del canal a un autor y se ejecuta la creación del autor. Cuando disponemos de un autor se pueden añadir usuarios al canal, dicho procedimiento es la segunda parte del esquema. El autor será el responsable de mandar la dirección, o LINK, al próximo suscriptor que quiera unirse al canal. Cuando el autor acepta la petición de unión de la parte del suscriptor éste último estará listo para empezar a mandar mensajes en el canal. Entonces, el propietario del canal podrá recuperar y leer los mensajes mandados por sus suscriptores.

Al fin, cuando se analiza el código, vemos que el Autor crea un canal «*Single Branch*» (rama única) y también se realiza la unión de un suscriptor en el canal. El usuario, en este caso la bombilla simulada, mandará mensajes con un «payload» público y otro privado. De esta forma se puede mandar información cifrada. La información cifrada estará protegida y disponible únicamente para el responsable del canal. Por ello, el autor será el único

que podrá leer las informaciones sensibles que envía la bombilla conectada al canal.

4.3.4. Cálculo de métricas y análisis de los resultados

Para el cálculo de métricas se ha intentado realizar los mismos experimentos que en el caso de Ethereum. Pero como se ha mostrado antes, para interactuar con el Tangle de IOTA se ha usado un programa en JavaScript que se ejecutará con NodeJS. En este caso, la obtención de las métricas será a través del terminal donde se ejecuta el programa. Además, con el uso de un script en Node.js se podrá conectar al cliente iota.rs y de este modo parsear mas fácilmente los paquetes de Tangle. Esta vez no se ha requerido el uso de una aplicación web accesorio que nos muestre los mensajes.

Payload público vs Payload privado

Como se ha visto antes, en el caso de Ethereum, las transacciones incluidas siempre presentan una estructura similar con su peso en Bytes. En cambio, se observa que Tangle procesa los paquetes de una forma completamente diferente. En este caso un mensaje debe procesar otros dos mensajes y a la vez transporta información. La hipótesis lleva a pensar que los paquetes tendrán un peso variable en la red y que en función del mensaje enviado ocupará más o menos espacio.

Como el suscriptor del canal de IOTA creado transporta un mensaje público y otro cifrado se busca analizar las diferencias que existen al momento de mandar un mensaje. La hipótesis actual es que el tamaño de los mensajes va a ser superior si se transporta más información cifrada que pública. Adicionalmente y considerando éste efecto se busca conocer si en el caso de tener mensajes más largos también conllevarán un tiempo de confirmación más elevado.

Se han mandado dos conjuntos de 10 paquetes, el primero de ellos con un payload público mucho más extenso que el privado y viceversa. A continuación se incluyen los dos payloads utilizados.

El primero de ellos con el mensaje largo siendo el payload público:

- **Contenido público:** `Estado Bombilla Esta encendida? (true/false)`
- **Contenido privado:** `timestamp:15:58 / 2022-05-28`

Del primer grupo de análisis obtenemos que los paquetes son todos de **282 Bytes**. Al usar un convertidor online de Hexadecimal a codificación Ascii se observa el siguiente mensaje:

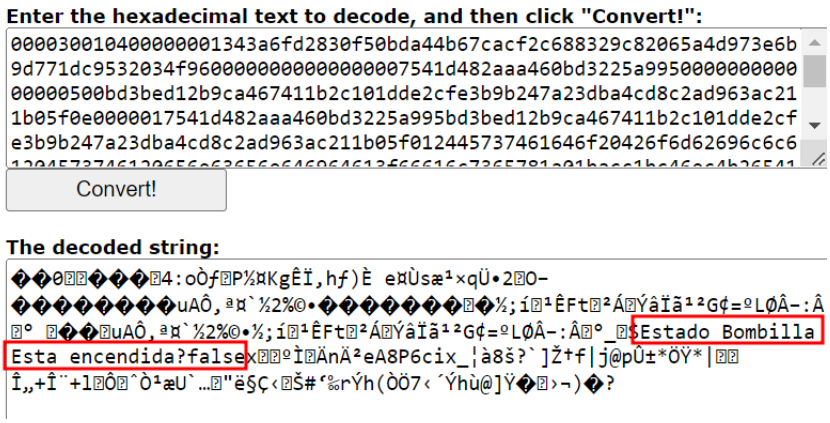


Figura 21: Contenido del mensaje con payload público con más información

El segundo conjunto de paquetes tendrá un pequeño payload público y la mayor parte de la información se mandará como el contenido cifrado.

- **Contenido público:** `Estado Bombilla`
- **Contenido privado:** `Esta encendida?:(true/false), timestamp:15:58 / 2022-05-28`

Se observa que en este caso el contenido del mensaje es fijo a **290 Bytes** para las 10 transacciones y en caso de convertirlo a Ascii se tiene el siguiente contenido:

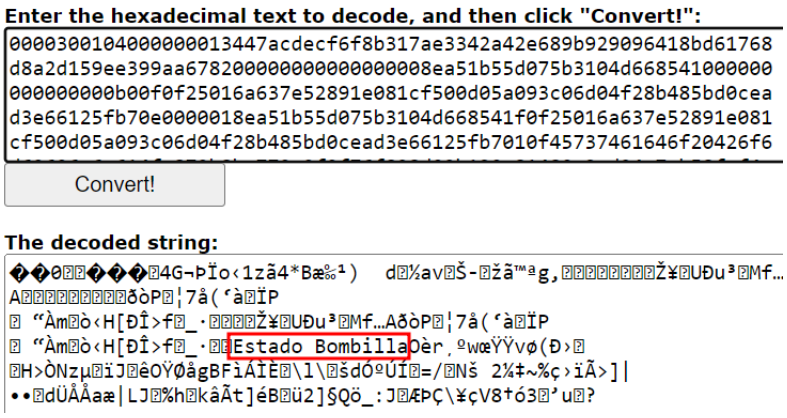


Figura 22: Contenido del mensaje con el payload público más pequeño

La diferencia que existe entre ambos es de solamente 8 Bytes. Lo que efectivamente confirma nuestra hipótesis, pero solamente con **2,84 %** menor para el payload público más largo. También se ha comprobado que en caso

de crear un nuevo canal la clave de cifrado es diferente, lo que implica que el peso de los paquetes pude cambiar de algunos bytes.

El mismo experimento es realizado 4 veces más para obtener la media de los resultados: **3,18 %**, **3,5 %**, **2,81 %** y **2,46 %**. Por lo que ahora, se puede asegurar que para un pequeño mensaje en caso de compartirlo como mensaje público se ahorra un **2,96 %** de su peso en bytes.

Los resultados anteriores muestran que hay una correlación entre el peso de los paquetes y su contenido. Analizando el tiempo de confirmación del grupo de 10 paquetes se verá si uno de los dos grupos es procesado antes por el Tangle de IOTA. A continuación se presentan los resultados de la siguiente hipótesis.

Tiempo que tarda en ser confirmada una transacción según su tamaño (bytes)
A través del cálculo de la diferencia entre el *timestamp* al momento de mandar el mensaje y el momento de ser incluido en el Tangle para los dos grupos de payloads se tienen las siguientes tablas:

Intento Número	Tiempo Envío
1	5
2	5
3	9
4	7
5	2
6	11
7	3
8	1
9	6
10	8

(a) Payload público más grande

Intento Número	Tiempo Envío
1	5
2	16
3	5
4	8
5	15
6	6
7	4
8	6
9	6
10	8

(b) Payload privado más grande

Figura 23: Resumen de los 10 paquetes enviados

Como se observa el tiempo medio cuando el peso del paquete es menor se situa en **5,7 segundos** y con el paquete más pesado se tiene **7,9 segundos** lo que implica que es un **38,6 %** superior en el caso de mandar un mensaje cifrado.

Mensajes enviados vs ejecutados en 1 min

Para realizar las medidas como en el caso de Ethereum se buscará mandar el máximo número de paquetes durante 1 minuto para saber cuantos de estos se procesaran en el Tangle de IOTA.

Para llegar a la conclusión de éste experimento se debe tomar en cuenta que la versión de IOTA con la que se ha trabajado usa la figura del *Coordinador* para asegurarse de la validez de los mensajes. Para ello, un mensaje debe esperarse hasta que se complete un *Milestone* y de este modo se confirme su validez.

Por esa razón, se han mandado ráfagas de paquetes al canal con distintos intervalos. Los primeros 5 intentos de 1 minuto en los que el suscriptor (o bombilla) envia un solo mensaje a la vez. El segundo caso se han mandado 20 peticiones seguidas y luego se recuperan los mensajes que se han logrado incluir en el Tangle. Los resultados nos dan las siguientes tablas:

Intento	Enviados	Ejecutados
1	10	10
2	9	9
3	12	12
4	13	13
5	9	9

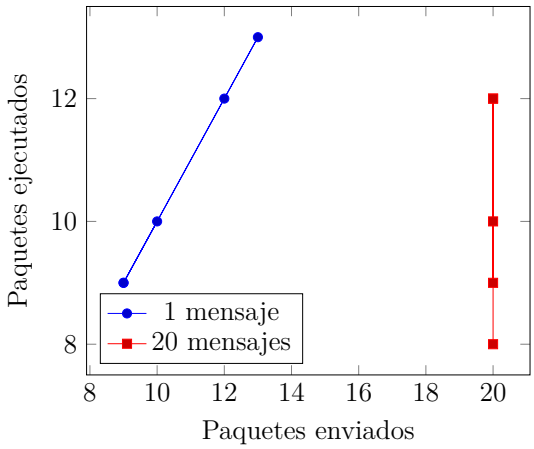
(a) 1 mensaje a la vez

Intento	Enviados	Ejecutados
1	20	8
2	20	12
3	20	9
4	20	12
5	20	10

(b) 20 mensajes a la vez

Figura 24: Resumen de los mensajes enviados vs los ejecutados

Así se podrá comparar el tiempo entre los paquetes que se han mandado a la red y los que se han ejecutado correctamente. Los resultados son los siguientes:



La gráfica muestra que en el caso de tener un solo suscriptor y a los efectos de haber usado JavaScript (que de base no permite usar el «multithreading») las transacciones confirmadas no difieren en caso de mandar además mensajes a la vez, causando incluso tiempos de envío adicionales.

Por último, en IOTA no existen más datos de interés que sean útiles para ser comparados con otras tecnologías y con todos éstos resultados se llega a las conclusiones del estudio.

4.4. IOTA vs Ethereum

En las gráficas y resultados que se han obtenido se observan claras diferencias entre ambas tecnologías. En primer lugar, IOTA es más rápido que Ethereum y no presenta diferencias de comportamiento en los momentos en que la red está congestionada.

Se observa que los rendimientos de IOTA pueden ser diferentes en función del tipo de payload deseado y en cambio en Ethereum se mantiene constante. Este efecto es debido a la estructura de los mensajes y a los protocolos que transportan la información. Así como en IOTA se usa Streams, un protocolo de mensajería sofisticado que cifra las comunicaciones en Ethereum se compila un Smart Contract y se mandan las informaciones y variables definidas en el.

En cambio, en Ethereum, el comportamiento de la bombilla cambia en función del precio del GAS haciendo imposible o muy cara el uso de la bombilla. Como punto importante IOTA es gratuito e incentiva su uso gracias a la forma como el protocolo ha estado concebido.

Por ultimo, en lo que respecta los recursos informáticos para usar ambas tecnologías no cambian mucho. Ambos requieren de JavaScript para funcionar, entre otras tecnologías que existen y además permiten su uso a través de aplicaciones web que facilitan el control y visualización de los dispositivos conectados.

Por último, para comparar los aspectos de seguridad se ha trabajado con los conocimientos prácticos adquiridos durante la realización del estudio y a través de comparar los diferentes aspectos teóricos vistos en el estado del arte. Entre ambas tecnologías existen diferencias de implementación que permiten reducir los errores que conllevarían a un problema de seguridad y afectarían la comunicación entre la bombilla y la red.

En caso de existir un error de seguridad en el código del contrato inteligente o tener una mala configuración de Truffle, Infura o el servidor donde se alberga la aplicación descentralizada puede ser peligroso para mantener la integridad, confidencialidad o disponibilidad de la bombilla inteligente y las

informaciones que comparte. Los puntos positivos existen y son todos los aspectos de seguridad que se han comentado en el estado del arte, como el uso de algoritmos de cifrado seguros, protecciones contra la modificación de los datos mandados a la blockchain e incluso dejar fuera de la red los dispositivos comprometidos.

En el caso de IOTA los aspectos de seguridad a considerar son los elementos de seguridad que existen al crear un canal. En tal caso se debe usar contraseñas que cumplan con los aspectos de seguridad recomendados (mínimo con una longitud de 12 caracteres, cifras, mayúsculas y caracteres especiales). Dicha información debe guardarse en un lugar seguro y en caso de compartirla debe hacerse a través de un canal protegido. Sino, IOTA ofrece una gran multitud de protecciones para dispositivos IoT, como la figura del Coordinador que debe validar la transacción, un nodo que resuelve complejos acertijos criptográficos antes de validar un mensaje, la atribución de pesos en los nodos que tienen un rol positivo en el Tangle y por último la comprobación de que nuestro mensaje haya procesado otros dos mensajes.

En resumen, se puede ver que IOTA ha puesto un gran peso en la seguridad de los dispositivos IoT y que frente a otra tecnología como Ethereum puede presentar menos problemas de seguridad en los ejemplos realizados.

5. Conclusiones

En este estudio se propuso como objetivo de nuestra investigación: *Demostración de las ventajas del uso de una tecnología especialmente concebida para las redes con dispositivos IoT*. Con el principio de probar las conclusiones se han realizado experimentos con IOTA y Ethereum que tenían como finalidad comparar ambas tecnologías y exponer cual de ellas presenta una mejor opción para dispositivos conectados.

Los análisis fueron realizados usando medios de producción reales. En el caso de Ethereum para evitar las altas comisiones que la red principal exige se usó la red de pruebas Ropsten y en IOTA la mainnet por la gratuidad del protocolo. También se ha considerado un único dispositivo conectado a la red y diferentes usuarios intentando ejecutar la misma transacción a la vez. Las mediciones también consideran diferentes momentos del día y la semana en los que las redes están mas congestionadas.

Con lo que respecta a los experimentos, los valores obtenidos han demostrado que pagando el precio de GAS más elevado (10000 Gwei) el tiempo medio para realizar una transacción con Ethereum es un **314,04 %** superior que en el caso de IOTA. La conclusión se obtiene a través de los resultados de IOTA habiendo más mensajes ejecutados en 1 minuto. También, un problema lateral en Ethereum son los altos costos para usar el protocolo comparado con IOTA que es gratuito.

Los resultados aquí obtenidos pueden servir de reflexión a otros investigadores y a empresas que busquen implementar de forma masiva redes de dispositivos IoT. En este sentido el uso de IOTA como protocolo de comunicación, puede ser un elemento impulsor para esta tecnología y mejorar sus capacidades actuales. Actualmente en desuso y muy por debajo de sus capacidades y ventajas teoréticas que se han demostrado en estudios anteriores.

No obstante el estudio presenta algunas limitaciones tales como las arquitecturas de los sistemas IoT realizados. En consecuencia, las simulaciones realizadas se ejecutan en un ordenador y solo incluyen un único dispositivo conectado. También existen restricciones en las muestras obtenidas para realizar los análisis, que se obtienen a partir de los diferentes navegadores de bloques. Así, las transacciones pueden verse limitadas por las restricciones de los protocolos usados.

Entonces, para hacer más extensivo el estudio y considerar todos los efectos del mundo real se puede implementar el mismo sistema usando componentes físicos. También repetir los experimentos con otros protocolos en la blockchain, como *Ripple*, puede usarse para encontrar el protocolo que mejor se adapte para su uso en sistemas IoT. Para concluir, IOTA presenta

actualizaciones regulares y realizar el mismo experimento en sus futuras versiones done la figura del Coordinador será eliminada permitirá de analizar como el hecho de suprimir dicho mecanismo de control reduce los tiempos de validación de las transacciones.

6. Glosario

Blockchain Conjunto de tecnologías que permiten llevar un registro seguro, descentralizado, sincronizado y distribuido de las operaciones digitales, sin necesidad de la intermediación de terceros.

TANGLE Tecnologia basada en *distributed ledger technology (DLT)* (libro de cuentas distribuido) que está básicamente basado para su en el Internet of Things (IoT). IOTA está basado en esta nueva tecnología DLT y no en un modelo blockchain tradicional.

DAG ***D****irected* ***A****cyclic* ***G****raph*

Proof of Work Algoritmo de consenso basado en el poder de cómputo para resolver un problema criptográfico.
sigla **PoW**

Proof of Stake Algoritmo de consenso basado en el poder de cómputo para resolver un problema criptográfico.
sigla **PoS**

distributed ledger technology Tecnología que habilita la posibilidad de almacenar, sincronizar y compartir datos de forma distribuida entre todos los nodos de la red sin depender de autoridades o nodos centrales a partir de un mecanismo de consenso.
sigla **DLT**

Smart Contract Un contrato inteligente es un programa informático que facilita, asegura, hace cumplir y ejecuta acuerdos registrados entre dos o más partes.

Solidity Solidity es un lenguaje de programación orientado a objetos para escribir contratos inteligentes.

Ropsten Es una testnet importante que contiene una serie de características que sirven como pruebas de código, imitando el modelo de consenso basado en PoW que utiliza Ethereum hoy en día.

Mainnet Es una redprincipal.[en] la que se ejecuta un proyecto de cadena de bloques.

Testnet En la tecnología blockchain, una red de prueba es una instancia de una cadena de bloques impulsada por la misma o una versión más reciente del software subyacente, que se utilizará para pruebas y experimentación sin riesgo para los fondos reales o la cadena principal.

hash Es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija. Independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud.

7. Bibliografía

Referencias

[1] https://uoc.edu/ (07/03/2018)

[2] https://iota-beginners-guide.com/ (07/03/2018)

[3] https://wiki.iota.org/ (07/03/2018)

[4] Josep Lluíos de la Rosa Esteva, Alberto Ballesteros Rodríguez (septiembre 2021), **Fundamentos de Ethereum**, FUOC.

[5] Matevž Pustišeka and Andrej Kos, **Approaches to Front-End IoT Application Development for the Ethereum Blockchain** https://www.sciencedirect.com/science/article/pii/S1877050918302308.

[6] https://www.kaleido.io/blockchain-platform/truffle

[7] https://trufflesuite.com/guides/using-infura-custom-provider/index.html

[8] https://solidity-es.readthedocs.io/es/latest/

[9] https://wiki.iota.org/iota.rs/overview

[10] https://www.redhat.com/es/topics/api/what-is-a-rest-api

[11] https://eth.wiki/fundamentals/patricia-tree

[12] https://medium.com/@eiki1212/ethereum-state-trie-architecture-explained-a30237009d4e

[13] https://web3js.readthedocs.io/en/v1.7.3/

[14] Roben Castagna Lunardi, Henry Cabral Nunes, Vinicius da Silva Branco, Bruno Hugentobler Lippert, Charles Varlei Neu and Avelino Francisco Zorzo. **Performance and Cost Evaluation of Smart Contracts in Collaborative Health Care Environments**, https://arxiv.org/pdf/1912.09773.pdf

[15] https://ethereum.org/en/upgrades/shard-chains/

[16] Bruno Machado Agostinho, Mario Antônio Ribeiro Dantas and Alex Sandro Roschildt Pinto, **Proposal of an Economy of Things Architecture and an Approach Comparing Cryptocurrencies**

[17] https://www.einfochips.com/blog/is-it-happening-for-real-50-billion-connected-devices-by-2020/

[18] Huaqun Guo and Xingjie Yu, **A survey on blockchain technology and its security** https://www.sciencedirect.com/science/article/pii/S2096720922000070.

8. Anexos

En este apartado se incluye el código y los scripts utilizados para completar el trabajo.

8.1. Implementación Ethereum e IoT

La siguiente URL dirige al lector al repositorio Github del autor del estudio.
https://github.com/CapSoG/UOC_TFM/tree/main/Projet
Los documentos que se incluyen son los necesarios para funcionar con el *Webpack Truffle Box*. También se incluye el documento de configuración de Truffle llamado *truffle-config.js* que ha permitido publicar el Smart Contract a la red Ropsten con Infura. Sin embargo, hay que poner una especial atención al hecho de que no es seguro publicar este documento públicamente a Internet con todas las informaciones necesarias para acceder a nuestra cartera de divisivas digitales y a nuestra cuenta de Infura.

8.2. Implementación IOTA e IoT

La siguiente URL permite acceder al repositorio donde se encuentra el código JavaScript que envía los mensajes a Tangle. También se incluye el script en JavaScript que se ha usado para extraer los datos de los experimentos. https://github.com/CapSoG/UOC_TFM