Facultat d'Informàtica de Barcelona (FIB-UPC)

Design and Implementation of the Persistence layer of a Blockchain to secure IP prefixes

Autor: Miquel Ferriol Galmés

Director: Albert Cabellos Ceballos

Abstract

BlockChain technology is currently one of the topics that is giving more talk, as it offers security, decentralization, transparency and anonymity, among other qualities.

This project aims to bring this technology to the field of Information Technology and, more specifically, to offer a parallel implementation of the distributed database that defines LISP.

The complete development of the system, exceeds the scope for a single technical engineering project. Being such a broad study, the work has been divided into four distinct sections. In particular, this section is based on the implementation of the persistence layer that allows securing the existing IP addresses through the block chain.

Then, the whole development process of this one is shown. Indicate that, to carry out this task, the work has been based on existing implementations of other known BlockChains, adapting them to the needs that are defined by LISP.

Resum

La tecnologia BlockChain actualment és un dels temes que més està donant què parlar, ja que ofereix seguretat, descentralització, transparència i anonimat, entre d'altres qualitats.

Aquest projecte intenta dur a aquesta tecnologia a l'abast de les Tecnologies de la Informació i, més concretament, ofereix una implementació paral·lela de la base de dades distribuïda que defineix LISP.
El desenvolupament complet del sistema, sobrepassa l'abast per a un únic projecte d'enginyeria tècnica. Com que és un estudi tan ampli, s'ha dividit el treball en quatre apartats ben diferenciats. En concret, aquest apartat es basa en la implementació de la capa de persistència que permet assegurar les adreces IP, a través de la cadena de blocs.

A continuació, es mostra tot el procés de desenvolupament d'aquest. Cal indicar que, per dur a terme aquesta tasca, el treball s'ha basat en implementacions ja existents d'altres BlockChains conegudes, adaptant-les a les necessitats que vénen marcades per LISP.

Resumen

La tecnología BlockChain actualmente es uno de los temas que más está dando que hablar, ya que ofrece seguridad, descentralización, transparencia y anonimato, entre otras cualidades.

Este proyecto intenta llevar dicha tecnología al ámbito de las Tecnologías de la Información y, más concretamente, ofrecer una implementación paralela de la base de datos distribuida que define LISP.

El desarrollo completo del sistema, sobrepasa el alcance para un único proyecto de ingenieria técnica. Al ser un estudio tan ámplio, se ha dividido el trabajo en cuatro apartados bien diferenciados. En concreto, este apartado se basa en la implementación de la capa de persistencia que permite asegurar las direcciones IP existentes mediante la cadena de bloques.

A continuación, se muestra todo el proceso de desarrollo de éste. Indicar que, para llevar a cabo esta tarea, el trabajo se ha basado en implementaciones ya existentes de otras BlockChains conocidas, adaptándolas a las necesidades que vienen marcadas por LISP.

1- Contexto

1.1 - Introducción

En el mundo de las tecnologías de la información, las redes son una parte fundamental para el funcionamiento de equipos y la comunicación entre ellos. A medida que, poco a poco, las redes se van extendiendo, un cambio en la topología puede ser muy costoso e incluso, a veces, inalcanzable.

Las Overlay Networks han sido y están siendo usadas para evitar este tipo de problemas. Mediante ellas es posible añadir una capa lógica sobre la física. Estas redes han demostrado ser útiles para una amplia gama de casuísticas de uso, tales como multicast [1], ingeniería de tráfico [2], redes elásticas [3] o redes peer-to-peer [4].

Además, con el advenimiento de Software Defined Networking (SDN), se han convertido en una herramienta para habilitar las capacidades SDN sobre equipos de red heredados [5].

Entre las diferentes opciones que existen para añadir esta capa lógica, el Locator / ID Separación Protocolo (LISP) [6] ha ido ganando fuerza en la industria y el mundo académico [7], [8].

En las redes LISP el tráfico es encapsulado en paquetes basados en un identificador (LocatorID) y enrutados a través de la capa física. LISP además, aprovecha una base de datos pública para guardar los mapeos que hacen posible esta superposición y un mecanismo de PULL para recuperar estos mapeos según sean necesarios.

Esta base de datos distribuida, necesaria para que LISP funcione, ha sido objeto de estudio por diferentes partes. El sistema actual que ofrece LISP (DDT) es complicado de gestionar (son necesarios varios nodos y los cambios deben realizarse manualmente) y, aunque se dice que la base de datos es distribuida, el modelo es más bien centralizado y la seguridad se basa en certificados digitales.

Estudios realizados por CISCO y la UPC [9] argumentan que la tecnología BlockChain parece ofrecer las características necesarias para, además de mantener una base de datos distribuida, simplificar su gestión basándose en un sistema de transacciones como el que podría existir en Bitcoin.

El proyecto que se va a describir a continuación, pretende estudiar la viabilidad del uso de la tecnología BlockChain para mantener la base de datos de mapeos que describe LISP, así como la implementación de un algoritmo de consenso basado en PoS (Proof of Stake).

1.2 - Partes Interesadas

Como en todo proyecto, existen partes interesadas en la realización de éste. A parte de la propia experiencia y conocimientos que adquiere el alumno en el trabajo realizado, existen otros entes a los que les pueda interesar su desarrollo y realización:

- **UPC:** como se ha citado anteriormente, la UPC ha realizado estudios sobre la viabilidad del uso de la tecnología BlockChain sobre LISP. La realización y éxito de este proyecto sería un refuerzo más a los anteriores.
- **CISCO:** la empresa de TI ya ha incorporado LISP a su tecnología. Todo estudio que permita encontrar mejoras, añadir seguridad, ofrecer nuevas características…, puede hacer que su producto destaque sobre otros.

Por otro lado, se espera poder realizar un artículo académico sobre los resultados obtenidos en este proyecto, por lo que a cualquier interesado en el estudio de redes le podría resultar de especial interés.

2 - Estado del Arte

A continuación se detallan el conjunto de tecnologías que forman parte de todo el proyecto.

2.1 - LISP

Aunque LISP posee muchas ramas, en este documento nos vamos a centrar en la explicación de la arquitectura y, sobre todo, en la base de datos citada anteriormente.

2.1.1- Arquitectura [10]

LISP separa el núcleo del "resto" de internet creando dos espacios de nombres separados:

- Endpoint Identifiers (EIDs)
- Routing Locators (RLOCs)

El "resto" de internet consiste en sitios LISP (un Sistema Autónomo, por ejemplo) que utilizan direcciones EID. Los EID son direcciones IPv4 o IPv6 que identifican de forma única los hosts finales de comunicación.

Los sitios LISP se encuentran conectados a través del núcleo de Internet usando túneles entre los routers con capacidades LISP. Así, podemos identificar dos tipos principales de túneles en una red LISP:

- **Ingress Tunnel Router (ITR):** cuando los paquetes van de un sitio LISP al núcleo de Internet.
- **Egress Tunnel Router (ETR):** cuando los paquetes van desde el núcleo de Internet a un sitio LISP.

Con LISP, el núcleo utiliza RLOCs, un RLOC es una dirección IPv4 o IPv6 asignada a una interfaz de red orientada a Internet a través de un ITR o ETR.

2.1.2- Mapping System [11]

Como se ha dicho anteriormente, LISP posee una base de datos donde se almacenan las asignaciones entre EIDs y RLOCs. Esta base de datos distribuida se denomina Mapping System.

Dichas asignaciones relacionan la identidad de los dispositivos conectados a los sitios LISP (EID) con el conjunto de RLOCs configurados en los routers con capacidad LISP que dan servicio al sitio. Así mismo, éstas también incluyen políticas de ingeniería de tráfico y se pueden configurar para lograr el multihoming y el balanceo de carga deseado.

La LISP-DDT es conceptualmente similar al DNS, un directorio jerárquico cuya estructura interna representa el espacio de direcciones EID. La DDT está compuesta por nodos-DDT que forman una estructura de árbol, donde las hojas representan Map-Servers (componentes que publican los mappings).

En el momento en que se desea resolver una query, DDT opera igual que DNS pero solamente soporta lookups iterativos.

El problema que existe en estas tablas es que la configuración se tiene que realizar de forma manual. Los resolvers están configurados manualmente con un conjunto de nodos que actúan como nodo raíz y los nodos se configuran para tener la delegación DDT apropiada.

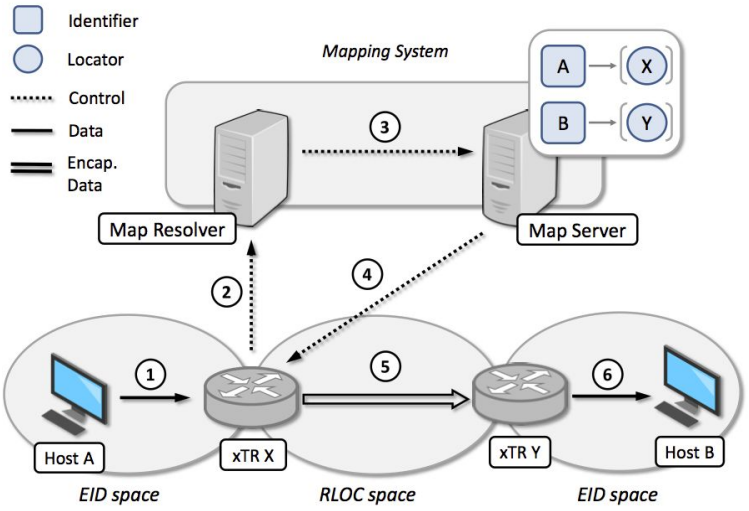


Figura 1: Lisp Overview

2.2 - BlockChain [12]

Blockchain es principalmente una tecnología para compartir información de forma distribuida, consensuada e inmutable. No deja de ser un libro mayor (un registro de transacciones de información) compartido entre múltiples nodos que basa la confiabilidad del mismo en el consenso y la replicación entre todos ellos.

La cadena de bloques se construye en base a propiedades criptográficas que buscan imposibilitar la modificación de la información que en ella se contiene. El bloque se compone de dos elementos: la cabecera, donde se encuentran los elementos necesarios para la validación, y el conjunto de transacciones.

Esta nueva tecnología se basa en tres principios fundamentales:

- DB distribuida: facilita que todos los nodos puedan comprobar la validez de los datos que se están manejando a partir de la información cifrada.
- Consensuada: con la información de las cabeceras, los distintos nodos pueden comprobar si la que ellos están manejando (y almacenando) es la misma que comparten la mayoría de los nodos.

- Inmutable: el algoritmo de cálculo de las claves de las cabeceras incluye, además de la información cifrada del propio bloque, el código cifrado de la cabecera del código precedente. Al estar compuesto por estos dos elementos, cualquier cambio en un bloque de la cadena afectaría a todos los que lo siguen, quedando así invalidados; pudiendo, de esta forma, detectar manipulaciones. El coste que supone tener que modificar un bloque es elevado y cuanto más atrás se encuentre en la cadena, esta complejidad se multiplica exponencialmente ya que supone tener que volver a crear los bloques siguientes con los nuevos códigos calculados. Por otra parte y por lo que hace referencia al consenso citado en el punto anterior, cualquier cambio que se intentara introducir desde un nodo individual implicaría que éste fuera rápidamente desechado por la mayoría de los nodos de la cadena.

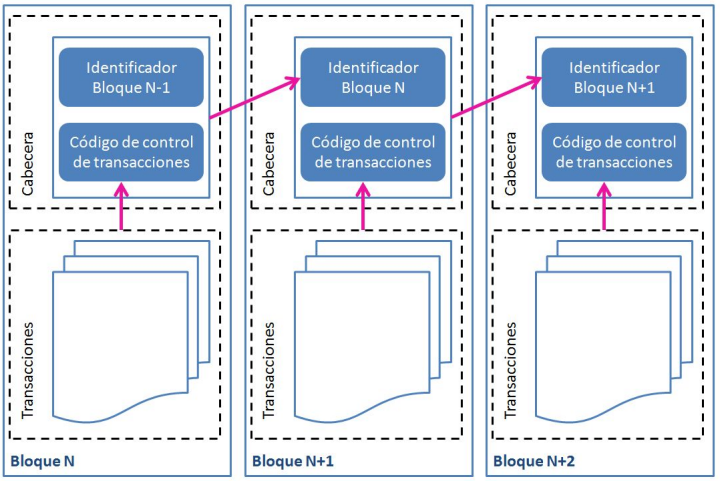


Figura 2: Estructura básica de la cadena de bloques.

2.2.1 - Inicios (Bitcoin [13])

El auge de esta tecnología surge con el Bitcoin. Una moneda digital que permite un pago rápido, instantáneo y descentralizado. Para entender su funcionamiento, se van a explicar a continuación dos de sus elementos principales:

2.2.1.1 - Direcciones

Todo participante de la red Bitcoin tiene una cartera electrónica que contiene un número arbitrario de claves criptográficas. A partir de la clave pública, se obtiene la dirección Bitcoin, que funciona como la entidad remitente y receptora para todos los pagos. Su clave privada correspondiente autoriza el pago solo para ese usuario. Las direcciones no tienen ninguna información sobre su dueño, son generalmente anónimas y no requieren de ningún contacto con los nodos de la red para su generación

2.2.1.2 - Transacciones

Los bitcoins contienen la dirección pública de su dueño. Cuando un usuario “A” transfiere algo a un usuario “B”, el primero (“A”) entrega la propiedad agregando la clave pública de “B” y firmando con su clave privada. “A” entonces incluye esos bitcoins en una transacción, y la difunde a los nodos de la red P2P a los que está conectado. Estos nodos validan las firmas criptográficas y el valor de la transacción antes de aceptarla y retransmitirla. Este procedimiento propaga la transacción de manera indefinida hasta alcanzar a todos los nodos de la red P2P. Finalmente la transacción es validada por un nodo minero y es incluida en un bloque de transacciones que es agregado a la cadena de bloques. Una vez que se encuentra en la cadena de bloques y ha sido confirmada por un número suficiente de bloques subsecuentes, la transacción es una parte permanente de la cadena de bloques y es aceptada por todos los participantes.

2.2.2 - Actualidad (Ethereum [14])

En la actualidad encontramos muchos usos de la tecnología BlockChain, pero principalmente esta tecnología es usada para las criptomonedas. Hoy en día, Ethereum es una de las principales ya que, como Bitcoin, plantea un sistema de pago con moneda pero además incluye la implementación de una característica nueva que llamamos Smart Contract. Los Smart Contract, básicamente, permiten la ejecución de código sobre la BlockChain; lo que facilita, asegura y hace cumplir acuerdos registrados entre dos o más partes.

2.2.3 - Algoritmo de Consenso

Una parte fundamental de esta tecnología es el algoritmo de consenso usado para decidir cuando un bloque es válido y se puede “emblocar” dentro de la cadena.

2.2.3.1 - Proof of Work

En la actualidad la cadena de bloques se forma a partir de un algoritmo de consenso que se basa en el trabajo. Generalmente, este trabajo consiste en realizar operaciones matemáticamente complejas que tengan una serie de características.

Para poner un ejemplo, en las criptomonedas, esta operación consiste en, una vez creado el bloque, encontrar un campo que haga que su hash tenga una serie de ceros al inicio. Este proceso se conoce como minado.

Una vez que algún nodo consigue minar un bloque, se le recompensa con una serie de monedas. De este modo, se incentiva a otros nodos a realizar este tipo de cálculos aumentando la seguridad de la cadena.

2.2.3.2 - Proof of Stake

Alternativamente al PoW surge el PoS. En el primer algoritmo, la seguridad recaía en la potencia de cálculo de la cadena entera, ahora ésta se centra en la importancia que tiene el nodo en la cadena.

En las criptomonedas, esta importancia normalmente viene definida por el número de monedas que tiene. Al ser tú una parte importante de la red, te interesa que ésta funcione correctamente.

Este tipo de consenso tiene una serie de ventajas sobre el algoritmo anteriormente descrito:

- Ahorro de energía: La cantidad total de energía requerida para procesar cada transacción Bitcoin asciende a 163 kWh, que equivale a la cantidad de electricidad empleada por un hogar durante unos 5 días y medio. Al suprimir esa necesidad de cálculo se reduce considerablemente el consumo de energía.

- Ataque del 51%: en el PoW si algún nodo es capaz de hacerse con el 51% de la potencia de cálculo de la red, sería capaz de cambiar la red a su antojo. En PoS, para hacerse con el 51% de las monedas se tendría que pagar tal cantidad de dinero que los costes superarían en mucho a los beneficios.

Poco a poco, van saliendo algunas propuestas que intentan implementar este tipo de algoritmos entre los cuales podemos destacar Algorand [15] y Ouroboros [16]. El primero explica una posible implementación que hace frente a los principales problemas de seguridad existentes en algoritmos PoS. El segundo, ofrece un algoritmo sencillo que aplica los conceptos básicos de este tipo de algoritmos de consenso.

4 - Alcance

4.1 - Formulación del problema

Cómo se ha comentado anteriormente, este proyecto tratará de estudiar la viabilidad de la aplicación de la tecnología BlockChain sobre el protocolo LISP para mantener la base de datos distribuida necesaria; y así, simplificar su funcionamiento, distribuir completamente la información y aportar la seguridad que ofrecen las cadenas de bloques.

4.1.1 - Objetivos

Los principales objetivos del proyecto son:

- Utilizar la tecnología BlockChain para construir una base de datos distribuida que pueda ser usada como Mapping System.
- Crear un sistema de transacciones basado en implementaciones existentes.
- Implementar una API que sea capaz de traducir el protocolo LISP actual a la realizada.
- Diseñar el sistema P2P para la conexión entre los diferentes nodos y crear un sistema de sincronización para mantener la base de datos consistente.
- Definir el algoritmo de consenso (basado en PoS) para la creación de los bloques.

- Utilizar varios modos de cifrado basados en PKI para cifrar las conexiones y cifrar tanto bloques como transacciones.

4.1 - Meta Final

A la finalización del proyecto, se espera tener un prototipo basado en los objetivos definidos en el apartado anterior y que pueda funcionar sobre una red LISP ya definida. Además de realizar una BlockChain basada en PoS que resulte lo suficientemente modular como para tener una base para la creación de otras.

La intención final es redactar un artículo académico con los resultados extraídos con las mediciones realizadas y ver, finalmente, si la aplicación de la BlockChain sobre LISP resulta útil.

4.2 - Posibles obstáculos

El PoS es un algoritmo nuevo que se ha planteado pero que presenta muchas dificultades a la hora de su implementación debido a la importancia que tiene éste sobre la cadena de bloques. No hay ningún modelo ya implementado en el que poderse basar que tenga las características deseadas.

Al ser un trabajo que abarca varios apartados y la realización no es individual, la organización va a ser un punto crítico. Aunque sean bloques separados, muchas veces existe dependencia entre ellos. Un retraso en el proyecto de cualquiera de las partes, puede significar otro en otro bloque.

5- Organización

5.1 - División del trabajo

Al ser un proyecto muy amplio y con varios estudiantes implicados, se ha decidido dividir el proyecto en 4 grandes bloques:

- **Persistencia:** El encargado del guardado de los datos así como el de la validación de las transacciones y bloques.
- **P2P:** Realización del protocolo P2P además de la sincronización de la cadena, envío de transacciones, ...
- **Algoritmo de Consenso:** Implementación del algoritmo PoS.
- **Mapping-System:** Traducción de la información que encontraremos en la BlockChain en la información necesaria para el protocolo LISP

Durante la fase de implementación cada estudiante realizará su bloque, teniendo en cuenta un conjunto de necesidades debido a las claras dependencias que existen entre bloques.

Durante la fase de testeo, primeramente se realizará un testing individual donde cada uno realizará un conjunto de pruebas para asegurar que el bloque funciona. Para terminar, se realizará un testeo del proyecto en común y se volcará sobre una red ya existente para así realizar las mediciones que sean necesarias.

5.2 - Metodología

Dentro del proyecto podemos destacar 3 grandes fases:

- **Información y documentación:** Una primera fase donde se investigan tecnologías existentes y dónde se elige el mejor camino a seguir para la implementación.
- **Implementación:** Donde cada miembro del grupo va a realizar el estudio y desarrollo de su bloque correspondiente.
- **Testeo:** Prueba de las implementaciones realizadas.

Aunque existen 3 fases bien diferenciadas se ha decidido usar una metodología Agile en lugar de Cascada por dos razones principales. Por una parte, al ser una tecnología tan novedosa la fase de documentación va en paralelo a la de implementación ya que, a medida que se va a ir desarrollando el proyecto, irán surgiendo dudas. Además, al ser varios miembros en el equipo el trabajo en paralelo es más que necesario. Y por la otra, la fase de testeo se realizará a medida que se vayan implementando pequeños sub-bloques para llegar a una etapa final de testeo del conjunto total del proyecto.

Para realizar el seguimiento del estudio, se realizarán reuniones semanales donde se explican los avances y los problemas/dudas que puedan surgir durante la implementación.

6- Planificación temporal

6.1- Planificación general

La duración estimada del proyecto es de unos 6 meses, empezando a mediados de Julio y terminando a mediados de Enero. La estimación de la carga de trabajo para este proyecto es de unas 360 horas aproximadamente. Este plazo, es intuitivo ya que, dependiendo del día de la presentación final, las fechas podrían variar para terminar de perfilar algún aspecto. Aún así, en el caso de que surgieran varios problemas a la hora de la realización del proyecto, este plazo se podría alargar hasta la siguiente fecha de entrega existente.

Al ser éste un trabajo realizado por varios miembros y con un bloque bien diferenciado para cada uno de ellos (tal y como se ha explicado en el apartado de alcance y contextualización), las diferentes etapas existentes se irán realizando, muchas de ellas, en paralelo. Eso sí, teniendo en cuenta que hay partes en que las dependencias entre bloques son claras y para ello la organización con el grupo es necesario.

6.2- Plan de acción y valoración de alternativas

Considerando el alcance del proyecto, se ha decidido presentar el proyecto en el turno existente de Enero. En este apartado se valoran los posibles imprevistos que se deben tener en cuenta a la hora de planificar la cantidad de trabajo que va a necesitar cada una de las etapas y que pueden hacer que la fecha de finalización varie.

Por una parte, el punto más crítico y problemático que existirá va a ser la parte de implementación. Al ser tanto BlockChain como LISP tecnologías nuevas y relativamente poco exploradas, el desarrollador se enfrenta a terrenos nunca antes trabajados. Por otra, durante la implementación, además, irán surgiendo dudas que no van a ser fáciles de resolver ya que la documentación en este tema es escasa. Ello puede conllevar a que las

estimaciones de duración varíen mucho, pudiendo llegar incluso a la necesidad de una ampliación del tiempo del proyecto.

Para intentar solventar este problema, durante la fase de especificación se intentará resolver el máximo de dudas que puedan ir surgiendo además de priorizar siempre la facilidad de implementación a otros posibles aspectos como puedan ser la eficiencia o la seguridad (siempre ciñéndose a unos mínimos).

Si se diera el caso de una necesidad extrema de tiempo, se optaría por eliminar algunos módulos que no fuesen estrictamente necesarios para el funcionamiento del prototipo final. Además, siempre cabe la posibilidad de realizar juegos de prueba más sencillos y de unas mediciones más simples.

Por el contrario, en el caso que el tiempo no sea un problema y llegue a sobrar, se optaría por seguir la especificación incluyendo una mejor documentación o por añadir funcionalidades extras anteriormente no contempladas.

7- Recursos

7.1- Recursos personales

Este proyecto, según lo dicho con anterioridad, lo componen cuatro miembros que tienen asignado un bloque de trabajo cada uno. Aún así, la especificación del estudio se realizará en conjunto para tener una visión global del proyecto.

Se estima que el tiempo de dedicación sea de unas 15-20h semanales para cumplir con los límites de tiempo establecidos. Esta cantidad, puede ir variando dependiendo de la carga de trabajo que cada uno de los miembros tenga, pudiendo tanto aumentar como disminuir a lo largo de la realización del trabajo.

7.2- Recursos materiales

A continuación se enumeran los distintos recursos materiales necesarios para la realización del proyecto.

1. **Ordenador de sobremesa:** Herramienta hardware para el desarrollo del proyecto. Con él se realizará toda la parte de implementación, seguimiento y redacción de la memoria del proyecto.

2. **JetBrains PyCharm Community Edition:** Herramienta software para la escritura de código del proyecto.

3. **Git:** Herramienta software para el control de versiones del proyecto.

4. **GitHub:** Servidor donde se guarda el código escrito del proyecto.

5. **Docs, Calc, Word y PowerPoint:** Herramientas software para la escritura de la memoria, realización de tablas y presentaciones del proyecto.

6. **Skype:** Herramienta software para la realización de las reuniones programadas con todos los miembros del equipo.

7. **Gantter:** Herramienta software usada para realizar una posible estimación de cada tarea del proyecto.

8. **VMs:** Software para la simulación de varios equipos para la realización de las pruebas.

9. **LISP Beta Network [17]:** Red donde se probará el prototipo final y se realizarán las mediciones necesarias.

10. **Conexión a Internet:** Herramienta obligada para la investigación, además de ser indispensable para muchos de los puntos enumerados arriba como pueden ser Github, Skype, Grantter...

8- Calendario del proyecto

8.1- Estimación de horas por tarea

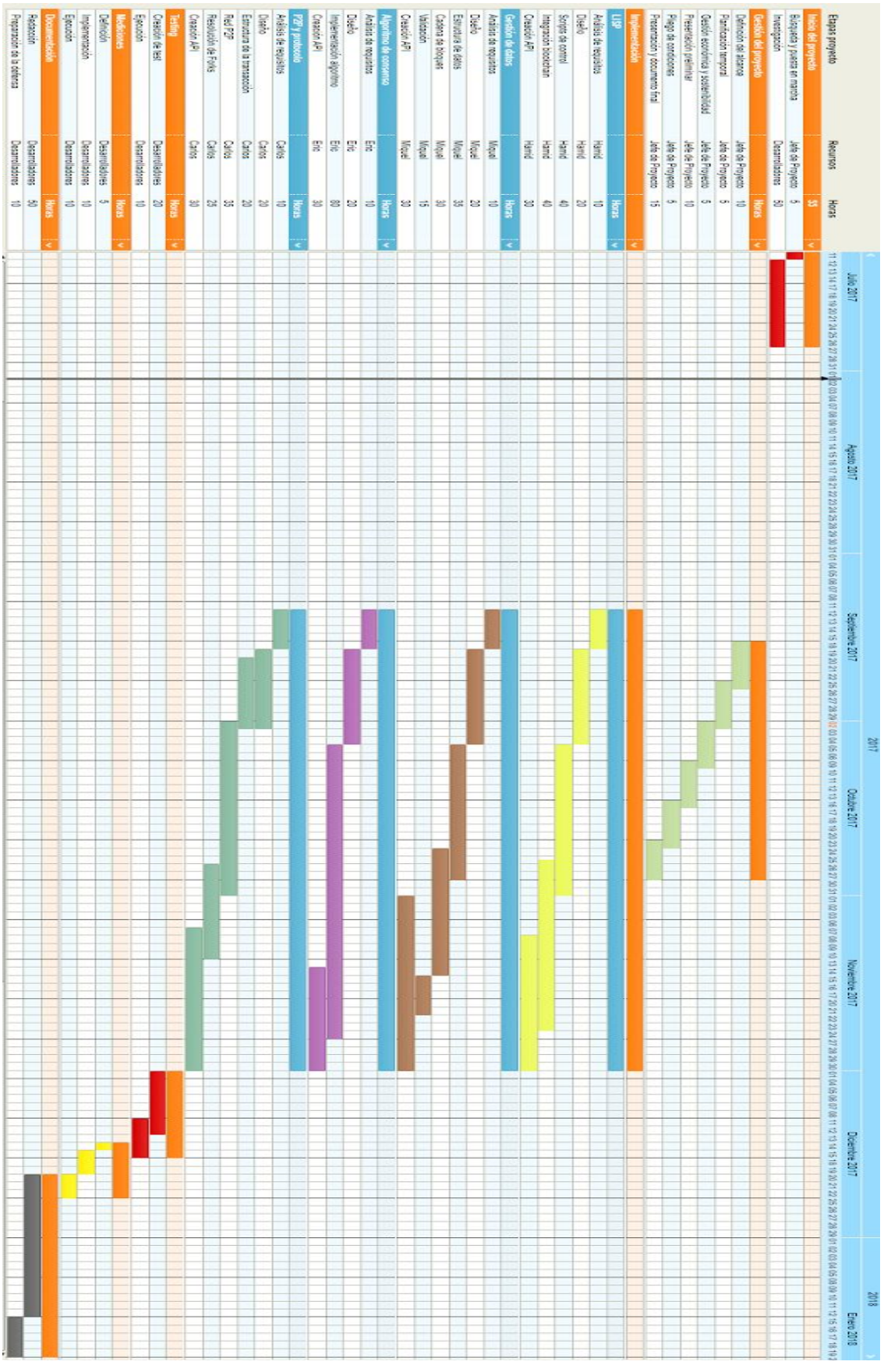
A continuación se enumeran las tareas con la estimación del tiempo necesario para su realización, para tener una pequeña estimación de la carga total del proyecto.

Tarea	Responsable	Horas
Inicio del proyecto		55
1. Búsqueda y puesta en marcha	Jefe de Proyecto	5
2. Investigación	Desarrolladores	50
Gestión del proyecto		50
1. Definición del alcance	Jefe de Proyecto	10
2. Planificación temporal	Jefe de Proyecto	5
3. Gestión económica y sostenibilidad	Jefe de Proyecto	5
4. Presentación preliminar	Jefe de Proyecto	10
5. Pliego de condiciones	Jefe de Proyecto	5
6. Presentación y documento final	Jefe de Proyecto	15
Implementación		560
LISP		140
1. Análisis de requisitos	Desarrollador (Hamid)	10
2. Diseño	Desarrollador (Hamid)	20
3. Scripts de control	Desarrollador (Hamid)	40
4. Integración BlockChain	Desarrollador (Hamid)	40
5. Creación API	Desarrollador (Hamid)	30
Gestión de Datos		140
1. Análisis de requisitos	Desarrollador (Miquel)	10
2. Diseño	Desarrollador (Miquel)	20
3. Estructuras de Datos	Desarrollador (Miquel)	35
4. Cadena de Bloques	Desarrollador (Miquel)	30
5. Validación	Desarrollador (Miquel)	15
6. Creación API	Desarrollador (Miquel)	30
Algoritmo de Consenso		140
1. Análisis de requisitos	Desarrollador (Eric)	10
2. Diseño	Desarrollador (Eric)	20
3. Implementación Algoritmo	Desarrollador (Eric)	80
4. Creación API	Desarrollador (Eric)	30
P2P + Protocolo		140
1. Análisis de requisitos	Desarrollador (Carlos)	10
2. Diseño	Desarrollador (Carlos)	20
3. Estructura de la Transacción	Desarrollador (Carlos)	20

4. Red P2P	Desarrollador (Carlos)	35
5. Resolución de Forks	Desarrollador (Carlos)	25
6. Creación API	Desarrollador (Carlos)	30
Testing		30
1. Creación de tests	Desarrolladores	20
2. Ejecución de tests	Desarrolladores	10
Mediciones		25
1. Definición	Ingeniero QA	5
2. Implementación	Desarrolladores	10
3. Ejecución	Ingeniero QA	10
Documentación		60
1. Redacción	Jefe de Proyecto	50
2. Preparación defensa	Jefe de Proyecto	10
TOTAL (per persona)		360

Tabla 1: Estimación de horas por tarea.

9- Gantt


Etapa proyecto
Recursos
Horas
Julio 2017
Agosto 2017
Septiembre 2017
Octubre 2017
Noviembre 2017
Diciembre 2017
2017
2018
Enero 2018
Inicio del proyecto
Bitácoras y puesta en marcha
Jefe de Proyecto
Desarrolladores
Investigación
Gestión del proyecto
Definición del alcance
Planificación temporal
Gestión económica y sostenibilidad
Presentación preliminar
Riesgo de contingencias
Preparación y documento final
Implementación
LSP
Análisis de requisitos
Diseño
Scripts de control
Integración blockchain
Creación API
Gestión de datos
Análisis de requisitos
Diseño
Estructura de datos
Cadena de bloques
Validación
Creación API
Algoritmo de consenso
Análisis de requisitos
Diseño
Implementación algoritmo
Creación API
P2P y protocolo
Análisis de requisitos
Diseño
Estructura de la transacción
Red P2P
Resolución de Forks
Creación API
Testing
Creación de test
Ejecución
Mediciones
Definición
Implementación
Ejecución
Documentación
Redacción
Preparación de la defensa

10- Gestión económica del proyecto

En el siguiente apartado se detallan los diferentes costes que se deberán tener en cuenta a la hora de realizar el presupuesto total del proyecto.

Al desarrollarse un software para ofrecer a los clientes, no se considerarán los costes relativos al volumen de producción ya que no estamos produciendo ningún número específico de unidades.

Para realizar la gestión de gastos se considerará la siguiente clasificación de costes: recursos humanos, recursos materiales (hardware y software) y gastos generales. Además se tendrán en cuenta costes de contingencia y de imprevistos.

10.1- Estimación de costes

10.1.1 - Costes de recursos humanos

Siguiendo la tabla mostrada en el apartado de estimación de horas por tareas, tenemos una referencia con la que guiarnos para calcular los costes en recursos humanos. Aunque existan varios roles que realizan las diferentes tareas (jefe de proyecto, analista, diseñador, desarrollador...) todas ellas van a ser realizadas por estudiantes, por lo que, en su totalidad, se calcularán con un precio de 8€/hora.

Cabe destacar que, al ser un estudio realizado por 4 integrantes, la gestión de éste y la documentación final se contabilizarán como si sólo las realizara uno de los miembros ya que el trabajo es el mismo. En los otros casos, se computarán las horas totales realizadas entre los 4 estudiantes.

Tarea	Responsable	Horas	Coste (€)
Inicio del proyecto		55 (x4)	1760
1. Búsqueda y puesta en marcha	Jefe de Proyecto	5 (x4)	160
2. Investigación	Desarrolladores	50 (x4)	1600
Gestión del proyecto		50	400
1. Definición del alcance	Jefe de Proyecto	10	80
2. Planificación temporal	Jefe de Proyecto	5	40

3. Gestión económica y sostenibilidad	Jefe de Proyecto	5	40
4. Presentación preliminar	Jefe de Proyecto	10	80
5. Pliego de condiciones	Jefe de Proyecto	5	40
6. Presentación y documento final	Jefe de Proyecto	15	120
Implementación		560	4480
LISP		140	1120
1. Análisis de requisitos	Desarrollador (Hamid)	10	80
2. Diseño	Desarrollador (Hamid)	20	160
3. Scripts de control	Desarrollador (Hamid)	40	320
4. Integración BlockChain	Desarrollador (Hamid)	40	320
5. Creación API	Desarrollador (Hamid)	30	240
Gestión de Datos		140	1120
1. Análisis de requisitos	Desarrollador (Miquel)	10	80
2. Diseño	Desarrollador (Miquel)	20	160
3. Estructuras de Datos	Desarrollador (Miquel)	35	280
4. Cadena de Bloques	Desarrollador (Miquel)	30	240
5. Validación	Desarrollador (Miquel)	15	120
6. Creación API	Desarrollador (Miquel)	30	240
Algoritmo de Consenso		140	1120
1. Análisis de requisitos	Desarrollador (Eric)	10	80
2. Diseño	Desarrollador (Eric)	20	160
3. Implementación Algoritmo	Desarrollador (Eric)	80	640
4. Creación API	Desarrollador (Eric)	30	240
P2P + Protocolo		140	1120
1. Análisis de requisitos	Desarrollador (Carlos)	10	80
2. Diseño	Desarrollador (Carlos)	20	160
3. Estructura de la Transacción	Desarrollador (Carlos)	20	160
4. Red P2P	Desarrollador (Carlos)	35	280
5. Resolución de Forks	Desarrollador (Carlos)	25	200
6. Creación API	Desarrollador (Carlos)	30	240
Testing		30 (x4)	960
1. Creación de tests	Desarrolladores	20 (x4)	640
2. Ejecución de tests	Desarrolladores	10 (x4)	320
Mediciones		25 (x4)	800

1. Definición	Ingeniero QA	5 (x4)	160
2. Implementación	Desarrolladores	10 (x4)	320
3. Ejecución	Ingeniero QA	10 (x4)	320
Documentación		60	480
1. Redacción	Jefe de Proyecto	50	400
2. Preparación defensa	Jefe de Proyecto	10	80
Total		1110	8880

Tabla 2: Costes directos por actividad.

La tabla 2 nos ofrece una visión más específica de dónde irá a parar el dinero destinado a los recursos humanos y, apoyándose en el diagrama de Gantt, se podrá realizar una estimación del momento.

10.1.2 - Costes de recursos materiales

En este apartado se incluirán los costes de los recursos hardware y software utilizados para la creación y desarrollo del proyecto. Estos gastos son considerados indirectos ya que no dependen de éste.

Tal y como se explicó en el apartado de recursos materiales existen varias herramientas de Software que se van a ir utilizando durante la elaboración del trabajo. Al ser gran parte de ellas gratuitas, no existe un coste directo. Tanto PyCharm como Word y PowerPoint son herramientas software privativas que necesitan subscripción. La primera (PyCharm) tiene licencia gratuita para estudiantes y las dos últimas vienen incluidas en la compra del PC de sobremesa. Por todo ello, se considera que la parte software no tiene ningún coste directo sobre el estudio.

Por lo que hace referencia a los recursos hardware existentes dentro del proyecto, será necesario un ordenador de sobremesa montado por piezas cuyo coste (teniendo en cuenta torre, pantalla, ratón y teclado) fué de 1458,80€ el 15 de Octubre de 2014.

Por otra parte, es destacable el uso de la LISP Beta Network (Map-Servers, Map-Resolvers, Proxy-Routers, etc.) que contiene los elementos necesarios para realizar las pruebas. Al ser una red en la cual participa la UPC, se tendrá acceso gratuito a ella.

Además, y ya para finalizar, se debe tener en cuenta el coste de la impresión del TFG, así como sus respectivas copias y encuadernación. Observando algunos foros de estudiantes por internet [18] parece que un precio estándar por copia del TFG ronda alrededor de los 50€. Existirán 5 copias, tres para los miembros del tribunal, una para el director y otra para el co-director.

Producto	Precio/u. (€)	Unidades	Coste estimado
PC Sobremesa	1458,80	1,00	57,55
Impresión	5,00	50,00	250,00
Total			307,55

Tabla 3: Costes indirectos por recursos materiales

La estimación de coste para el PC de sobremesa se ha realizado teniendo en cuenta el cómputo total de horas por persona (360€) y la vida útil del PC de 5 años y una media de utilización de unas 7 horas al día.

10.1.3 - Costes generales indirectos

Dentro de los costes generales encontramos todos los costes indirectos en los que se engloban todos aquellos costes adicionales que no son ni recursos humanos ni materiales.

En la siguiente tabla se desglosan los citados costes:

Gasto	Precio/mes (€)	Coste estimado
Internet	25,00	125,00
Luz	20,00	100,00
Agua	10,00	50,00
Gas	5,00	25,00
Alquiler	250,00	1250,00
Desplazamientos	65,00	325,00
Total	375,00	1875,00

Tabla 4: Costes indirectos por recursos materiales

El coste de desplazamiento se ha calculado teniendo en cuenta los movimientos de larga distancia, que se van a realizar 1 vez al més para las reuniones y/o presentaciones

presenciales relacionadas con el seguimiento del proyecto; más el desplazamiento interno por Barcelona.

El coste medio de un vuelo Palma de Mallorca - Barcelona / Barcelona - Palma de Mallorca (descuento de residente -50%- + Familia Numerosa General - 5%-) es de unos 30€ de media por trayecto. Para el coste de desplazamiento interno por la ciudad se ha calculado que será suficiente una T10 (9,95€) cada 2 meses.

10.1.4 - Contingencia

Al ser todo ello una estimación se reservará un porcentaje del presupuesto por si surge algún problema en el transcurso del proyecto que haga que aumente el coste previsto.

La partida de contingencia prevista para el proyecto será del 15% de la totalidad de los costes (humanos, materiales y generales).

Tipo	Coste estimado (€)	Porcentaje (%)	Coste contingencia (€)
Recursos humanos	8880,00	15	1332,00
Recursos materiales	307,55	15	46,13
Generales	1875,00	15	281,25
Total			1659,38

Tabla 5: Costes de contingencia.

10.1.5 - Imprevistos

Durante el desarrollo del proyecto, podrán ir surgiendo distintos contratiempos que pueden hacer variar el coste total de éste. A continuación se citan los principales problemas que pudieran surgir:

1. **Avería PC sobremesa:** Al ser un PC de sobremesa montado por piezas se deben de tener en cuenta dos tipos de avería. Una avería en la que el error fuera debido a un componente intercambiable, o una avería que comprometiera todo el ordenador y fuera necesaria la compra de uno nuevo. En el primer caso, se le asigna una

probabilidad de un 5% sobre un coste máximo de 350€. En el segundo, calculamos una probabilidad de un 1% sobre un coste máximo de reposición de 1500€.

2. **Retraso en la implementación:** Un aspecto bastante probable a tener en cuenta es la posibilidad de que alguno de los bloques se retrase en la implementación. Para el cálculo de costes imprevistos, estimaremos un retraso de 15 días por bloque, es decir, 75 horas en total que multiplicadas por 8€/h resultan 600€; y una probabilidad del 20% sobre dicho importe.

Imprevisto	Probabilidad (%)	Precio (€)	Coste (€)
Avería parcial	5	350	17,5
Avería total	1	1500	15,0
Retraso	20	600	120,0
Total			152,5

Tabla 6: Costes de imprevistos.

10.1.6 - Presupuesto Final

Para terminar y poniendo en común todas las partidas a tener en cuenta nos queda un presupuesto final de 12.874.44€

Concepto	Coste (€)
Recursos humanos	8.880,00
Recursos materiales	307,55
Costes generales	1.875,00
Contingencia	1.659,38
Imprevistos	152,50
Total	12.874,44

Tabla 7: Presupuesto final.

Al ser un proyecto de investigación no se espera obtener beneficio alguno de la realización de éste.

Para entender el presupuesto, se debe tener en cuenta que los costes (excepto los humanos) se han calculado individualmente, ya que la situación de cada miembro que realiza el proyecto es diferente.

9.2- Control de gestión

Existen pocos aspectos que se puedan controlar en este trabajo . Los recursos materiales que se utilizan ya tienen su antigüedad y su propósito no es específico para la realización del proyecto. Paralelamente, los gastos generales tampoco van muy en relación al proyecto ya que, exceptuando los costes de desplazamiento, existirían igualmente. Por lo tanto, la parte más controlable, presupuestariamente hablando, es la de recursos humanos.

Al final de cada fase del proyecto se comprobará si el total de horas se asemeja de alguna forma al total estimado en los apartados anteriores. Lo ideal sería que no existiese una desviación muy pronunciada con las horas estimadas (ya fuera por déficit o por superávit).

Si la suma de todos los gastos reales (calculados con las horas exactas) supera el presupuesto inicialmente asignado, se utilizará la partida del fondo de contingencia para cubrir los gastos extra.

El cálculo de las desviaciones que se puedan producir en el proyecto se realizará de la siguiente manera:

- Recursos Humanos = (coste estimado - coste real) x horas reales.
- Realización de tarea = (coste estimado - coste real) x horas reales.
- Total realización de tarea = coste total estimado de tareas - coste total real de tareas.
- Total en recursos = coste total estimado en recursos - coste total real de recursos.

11- Sostenibilidad y compromiso social

11.1- Dimensión económica

Es destacable que en la evaluación de los gastos, realizada en los apartados anteriores, no se ha tenido en cuenta el coste que tendrían los ajustes, actualizaciones y/o reparaciones durante la vida útil del proyecto. Ello se debe a que este trabajo se engloba en un estudio sobre la viabilidad de la aplicación de la tecnología BlockChain en la seguridad de las direcciones IP. Si el proyecto prosperase, se debería realizar de nuevo una evaluación de costes.

El punto fuerte de este proyecto recae en la seguridad que ofrece la cadena de bloques y en la descentralización que tiene. Por lo que cabe la posibilidad de que a una empresa le pueda interesar adoptar esta tecnología por varias razones. Una de ellas es la seguridad, pero otra, aunque use PKI (Public Key Infrastructure), es que no es necesaria ninguna autoridad certificadora ni certificados digitales para su autenticación.

Este proyecto está englobado dentro del LISP en colaboración con CISCO. Al ser un estudio, las horas dedicadas a cada tarea son las mínimas para poder ofrecer unos requisitos mínimos tanto de seguridad como de eficiencia. Para ello, se han reutilizado partes de código de otras cadenas de bloques para disminuir las horas de trabajo.

Para poder llegar a la aplicación el proyecto, sería necesario que alguna de las grandes empresas dedicadas al sector de las TI estuvieran dispuestas a adoptar este sistema. LISP aún es una red en constante cambio y, aunque es usada en algunos ámbitos, no está ampliamente aceptada como un estandard.

Puede que la aplicación del proyecto en sí no sea una realidad, pero sí que puede ser la base (el uso de la tecnología BlockChain) para aplicar a proyectos futuros.

11.2- Dimensión social

El aspecto social del proyecto es el que menos destaca ya que no se mejora, socialmente hablando, la calidad de vida a nadie. Si este proyecto se pone en funcionamiento la situación seguirá siendo la misma de antes; la gente conoce que existe internet, pero son pocas las personas a las que les interesa su funcionamiento. Además, éste es un estudio que se centra en el sector de Internet, pero de las grandes compañías.

¿Es necesario el proyecto?, si se considera cualquier mejora una necesidad en este caso sí. Una mejora en la seguridad de internet puede ser considerada una necesidad real para algunos, pero para otros pueda que el coste no sea asumible.

La tecnología blockChain es enemiga de unos y amiga de otros. La descentralización digital hace que no pueda existir una buena regulación por parte de las autoridades y ello puede perjudicar a algunos.

El proyecto podría ser perjudicial para muchos, pero a la vez, beneficioso para muchos otros. Por una parte, la no necesidad de Certificados Digitales y, en consecuencia, de Autoridades Certificadoras, podría quitar el trabajo a muchas personas. Por otra, para el usuario final, la descentralización, la transparencia, la posibilidad de participar, etc., pueden ser puntos a favor.

11.3- Dimensión ambiental

Aunque no lo parezca, el punto ambiental ha sido pensado desde el principio. Como se explicó en el apartado del estado del arte, actualmente todas las BlockChains se basan en un algoritmo de consenso basado en PoW. La potencia de cálculo necesaria para el mantenimiento de éstas ha llegado a superar el consumo de países enteros. Por ello se ha pensado en desarrollar un algoritmo basado en PoS que fuera capaz de dar la misma seguridad pero que no fuese necesaria una gran potencia de càlculo en cada uno de los nodos existentes interesados en participar en la plataforma.

El desarrollo del proyecto no requiere en sí mismo un consumo extra; el problema viene a la hora de la implementación real. Para mantener una cadena de bloques son necesarios

muchos nodos que corran realizando comprobaciones, propagando datos...; lo que conlleva a que el consumo vaya creciendo más y más a medida que va creciendo la cadena.

Para realización de este trabajo, se ha tenido en cuenta la posible reutilización del proyecto para otros campos. La idea principal es crear una base para que otros la puedan reutilizar si se desea crear alguna otra BlockChain con la sola necesidad de modificar algunos pequeños módulos.

Al ser un estudio basado en Software, el mayor impacto que puede producir un cambio es la masiva utilización. Para ello, se decidió crear el algoritmo de consenso PoS. Al no tener la necesidad de uso de Hardware dedicado, sería posible realizar la función con cualquier tipo de máquina que ya estuviese en funcionamiento.

11.4- Matriz de sostenibilidad

Dimensión	PPP	Vida útil	Riesgos
Económica	9	9	-5
Social	1	1	-8
Ambiental	9	7	-5
Sostenibilidad	18		

Tabla 8: Matriz de sostenibilidad.

12 - Principales decisiones

A lo largo del trabajo han ido surgiendo diferentes opciones y formatos que eran igual de válidos para conseguir la finalidad del proyecto. En este apartado se especifican las diferentes opciones y el porqué de su elección final.

12.1 - UTXO vs. Account/Balance

Como ya se ha dicho anteriormente el proyecto se ha basado en dos BlockChains ya existentes (Bitcoin y Ethereum). En el inicio del proyecto existían dudas entre cuáles de los modelos seleccionar, el modelo UTXO de Bitcoin o el modelo Account/Balance de Ethereum.

Estudiando sus características, se observaron los siguientes puntos a favor y en contra de cada una de ellas.

12.1.1 - UTXO

Las principales ventajas de un modelo basado en UTXO:

- **Mayor nivel de privacidad:** Si un usuario utiliza una nueva dirección para cada una de las transacciones es muy difícil relacionar las diferentes cuentas a cada una de ellas.
- **Escalabilidad:** este modelo es teoricamente más fácilmente escalable, ya que no guarda el estado de todos los nodos.

Por el contrario, la principal desventaja de este modelo es la dificultad para su implementación. Para cada transacción es necesaria la existencia de uno o más inputs cuyo valor total sea igual o mayor a la cantidad que se quiera transferir. Además, cada uno de esos inputs tiene la necesidad de estar firmado, y, esa firma debe coincidir con el remitente. Una vez realizada la transacción se deben crear los correspondientes outputs que servirán en un futuro como nuevos inputs (ello en Bitcoin no es un problema grave, pero en este caso, al trabajar con IPs, el cálculo de los outputs podría tener una gran dificultad).

12.1.2 - Account/Balance

Las principales ventajas del modelo Account/Balance:

- **Simplicidad:** Este modelo además de ser más sencillo de programar, es mucho más fácil de entender. Cada cuenta puede ser entendida como una cuenta bancaria.
- **Disminución de espacio:** Cada transacción, de una manera simplificada, solo necesita la cantidad que va a ser transferida y a quién va dirigida. No son necesarios todo el conjunto de outputs para verificarla.
- **Facilidad en el acceso de los datos:** Cualquier nodo, puede conocer de una manera sencilla toda la información de la cadena de bloques solo consultando el State Trie.

Por otro lado, este sistema cuenta con un conjunto de desventajas:

- **Privacidad:** Cada nodo puede conocer toda la información con solo conocer la dirección.
- **Nonce:** Para evitar ataques de double-spending, existe la necesidad de tener algún campo usado para conocer si una transacción ya ha sido realizada o no. Para ello es necesario un campo adicional llamado Nonce.

12.1.3 - Decisión final

Viendo las ventajas y desventajas que ofrece cada modelo, al final se decidió por usar el modelo de account/balance, principalmente por la facilidad que nos ofrecia, tanto en el caso de su implementación, como en el caso de su entendimiento.

12.2 - LevelDB

Al estar todos los modelos de datos usados en el proyecto referenciados por funciones Hash, desde un principio se tenía claro que la mejor opción era usar una base de datos Key-Value.

La decisión final fue usar LevelDB por las siguientes razones:

- No tiene necesidad de instalación de Software Adicional.
- Implementación nativa en Python.
- Usada por Ethereum. Al tener un modelo basado en Ethereum, se aseguraba no tener problemas de compatibilidad.

12.3 - RLP Encode

Al tener gran variedad de modelos de datos para guardar en la base de datos, se optó por elegir un protocolo de codificación.

RLP pretende ser un formato de serialización altamente minimalista. Su único propósito es almacenar matrices de bytes anidadas sin necesidad de definir ningún tipo de datos específicos como boolenos, floats, doubles...

Una alternativa podría haber sido algún otro algoritmo como Protobuf o BSON, sin embargo se eligió RLP debido a su simplicidad de implementación y su consistencia a byte perfecta.

12.4- Secp256k1

De relevada importancia es la firma de transacciones y bloques que nos permite verificar que quién nos envía alguna información es realmente quién es.

Para ello, usamos una firma basada en ECDSA (Elliptic Curve Digital Signature Algorithm). En este caso utilizamos la llamada Secp256k1 (E: $y^2=x^3+7$) curva usada en Bitcoin.

Se decidió por usar ésta por varias razones, entre las cuales destacan:

- Implementación en Python.
- Eficiencia: Al contrario de otras curvas, ésta no tiene una estructura totalmente aleatoria, si no que ha sido construida de una manera especialmente eficiente para la computación.
- Seguridad: Las constantes de secp256k1 se seleccionaron de manera predecible, lo que reduce significativamente la posibilidad de que el creador de la curva inserte algún tipo de puerta trasera en la curva.

13 - Modelo de datos

En esta sección se explican los diferentes modelos de datos usados en el proyecto. Están ordenados de mayor a menor simplicidad.

13.1 - Address

Cada dirección de la blockChain viene identificada por una address. Cada una de ellas tiene una longitud fija de 20 Bytes y se crea escogiendo los últimos 20 Bytes del Hash-256 de la clave pública de cada usuario.

13.2 - Balance

Este modelo contiene toda la información sobre las IPs que posee cada una de las diferentes cuentas, además de la información necesaria para crear las diversas respuestas a las llamadas procedentes de OOR.

A continuación se describen sus principales campos:

- **own_ips:** Contiene las diferentes direcciones IPs que posee la cuenta.
- **delegate_ips:** Contiene las diferentes IPs y a que cuenta han sido delegadas.
- **received_ips:** Contiene las diferentes IPs y de que cuenta han sido recibidas.
- **map_server:** La información de los diferentes Map Servers que posee esa cuenta.
- **locator:** La información de los diferentes Locators que posee esa cuenta.

13.3 - Account

Cada dirección tiene asociada una account. En ella se guardan básicamente dos campos:

- **Balance:** Explicado en el apartado anterior.
- **Nonce:** Este campo indica cuantas transacciones ha realizado ya esta cuenta. Este apartado es necesario para evitar posibles ataques de double-spending.

13.4 - Trie

Para almacenar los datos se ha usado una estructura conocida como Trie. Ésta no es más que un Merkle Patricia Trie (al cual se la han realizado algunas mejoras que se explicarán más adelante).

Existirán dos tipos de Trie en nuestro caso.

Por una parte tenemos el Transaction Trie, que va a ser usado para almacenar las transacciones dentro de un bloque. Existe un Transaction Trie para cada bloque creado. Una vez se ha creado el bloque, éste ya no se modifica.

Y por otra, está el World State Trie, que contiene todas las Accounts que existen en la BlockChain. Existe un World State Trie global, que se modifica cada vez que se aplica un nuevo bloque.

13.5 - Pytricia

Para poder consultar a quién pertenece una IP es necesario tener otra estructura diferente al World State ya que, para encontrar una llave, sería necesaria una búsqueda iterativa entre las diferentes hojas del State.

Para ello se ha usado una implementación de un IP-Subnet-Tree, llamada py-Radix

13.6 - Keystore

Para guardar las diferentes claves usadas para firmar tanto transacciones como bloques se usa un fichero que se guarda en formato JSON. Este fichero contiene la llave privada cifrada.

Para crear un Keystore se crea un par de claves pública/privada, que se cifran usando un esquema de cifrado por bloques (AES en este caso). Para crear la clave usada para cifrar el AES se pide al usuario una clave que, posteriormente es adaptada usando una función KDF (función de derivación de clave).

Un ejemplo sería:


```
{"version": 3,
  "crypto": {
    "ciphertext": "9f5fb32d0c61d392ffa15363d72228d2cfa30cc74de5d765eaa5caf1f116a9c0",
    "cipherparams": {
      "iv": "dc93bfcf01fdada27f1260523c882190"
    },
    "kdf": "pbkdf2",
    "kdfparams": {
      "dklen": 32,
      "salt": "fe328c58b0d15c4318ecbee3902d59aa",
      "rounds": 262144
    },
    "mac": "f2805d4d713c343ba57162533c1705a832ede58e4314b431a75c654b4cbb8988",
    "cipher": "aes-128-ctr",
    "version": 1
  },
  "address": "7719818983cb546d1badee634621dad4214cba25"
}
```


Figura 3: Ejemplo de Keystore.

Destacar los principales campos:

- **Crypto**: Contiene la clave cifrada y los parámetros necesarios para poder descifrarla usando el algoritmo correspondiente.
- **Kdf y KdfParams:** Especifica el algoritmo de cifrado, así como los parámetros.
- **Cipher:** Especifica el algoritmo usado para cifrar la clave privada.
- **Address:** Dirección a la cual pertenece esa clave.

13.7 - Cabecera de Bloque

Antes de pasar al bloque, vamos a definir la cabecera del bloque en cuestión. Esta última está formada principalmente por:

- **PrevHash:** Referencia al bloque anterior. Usado para mantener la consistencia del bloque.
- **Timestamp:** Tiempo en formato UNIX que indica el momento de la creación del bloque.
- **ExtraData:** Datos extras que se pueden añadir al bloque.

- **StateRoot:** Nodo raíz del World State Trie, usado para asegurar la correcta aplicación del bloque.
- **TransactionRoot:** Nodo raíz del TransactionRoot referente al bloque.
- **Number:** Indica el número de bloque.
- **Coinbase:** Indica el nodo que ha creado el bloque y que ha sido elegido por el algoritmo de consenso.

13.7 - Bloque

Cada bloque está formado por 3 elementos principales:

- **Header:** Cabecera del bloque.
- **Transactions:** Lista de todas las transacciones que existen dentro del bloque.
- **(v,r,s)**: Usados para crear la firma ECDSA.

14 - Implementación

En este apartado se va ha hablar de las diferentes implementaciones que se han realizado para el guardado de los datos y el mantenimiento de la consistencia de la cadena de bloques.

Podemos resumir las diferentes etapas implementadas en:

- Cargado inicial de la cadena (inicialización)
- Validación de transacción y bloque
- Aplicación de estos dos conceptos sobre la cadena (guardado de datos y modificación del estado).

14.1 - Conceptos previos

14.1.1- Bloque génesis

Para la inicialización de la cadena, es necesario un bloque inicial que permita crear el estado de inicio para todos los nodos.

En este proyecto, esta información viene definida en el fichero genesis.json.


```
{
  "coinbase": "0x0000000000000000000000000000000000000000",
  "timestamp": "0x5A4BB9B4",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "alloc": [
  {"be30c5eb90498848da5a5d822b0e1b4939f6fe74":{
      "balance":{
           "own_ips":["0:0:0:0:0:0:0:0/0"]
      }
  }},
  {"033351479035eda12e98cadd0a75a455173cfc52":{
       "balance":{
           "own_ips":["0.0.0.0/8"]
       }
   }},
  {"057383c1b0dd99e1293c4ce7a3cc14363c7238d4":{
       "balance":{
           "own_ips":["1.0.0.0/8"]
       }
   }},
  {"060dbcd5f1d7728334e92b22895bae7e4c3a2343":{
       "balance":{
           "own_ips":["2.0.0.0/8"]
       }
   }},
  {"067a7e73674e4f985b64cedcb4de34f41fa9595e":{
       "balance":{
           "own_ips":["3.0.0.0/8"]
       }
   }},
```


Figura 4: Fracción de definición del bloque génesis.

Podemos destacar 4 campos principales en la definición:

- Coinbase: La dirección del creador del bloque. Para ello nos reservamos la dirección 0. Al ser el bloque génesis, no hay creador.

- Timestamp: Donde se indica el tiempo en el que se creó el bloque.
- Alloc: En esta estructura viene definida toda la información de las diferentes cuentas que existen al inicio. Destacar que, para que el bloque génesis sea válido, debe tener todas las IPs existentes ya marcadas.

14.1.2- Transaction Pool

Cuando un usuario crea una transacción o la recibe de un nodo externo ésta no puede ser guardada directamente ya que no forma parte de ningún bloque. Por lo tanto, no es posible verificar su validez dentro de la cadena.

Para almacenar todas estas transacciones existe la Transaction Pool. A la vez que de ella se extraen las transacciones que van a ser utilizadas para la creación de un bloque.

14.1.3- Transaction

Aunque la transacción no forma parte de este proyecto, destacar que existen varios tipos de transacciones. Cada tipo viene identificado por un entero. Encontramos 4 tipos de transacciones:

- **Allocate (0)**: Esta transaccion se usa para hacer cambios entre direcciones IP. Tiene el efecto de cambiar el dueño de una IP.
- **Dellegate (1):** Es posible ceder un espacio de direcciones a otra cartera sin dejar de ser el dueño de éstas.
- **MapServer (2):** Esta transacción modifica la información a cerca de los MapServers que tiene una address.
- **Locator (3):** Esta transacción modifica la información a cerca de los Locator que tiene una address.

Dependiendo de el tipo de transacción que nos llegue, se aplicarán unos efectos u otros.

14.1.4- Trie

Cómo se ha explicado anteriormente, en la implementación se ha usado una estructura llamada Trie que nos permite guardar el estado global, así como asegurar una actualización de datos correcta.

Esta estructura se basa en un Merkle Patricia Tree, pero con algunas optimizaciones que permiten consultas más rápidas.

Para entender la implementación realizada primeramente debemos conocer los tipos de nodos. Mientras en un Merkle Patricia Tree existen tres tipos dentro del árbol (Rama, Raíz, Hoja), en la implementación se han creado 4 tipos de nodos (Raíz, Extensión, Rama, Hoja).

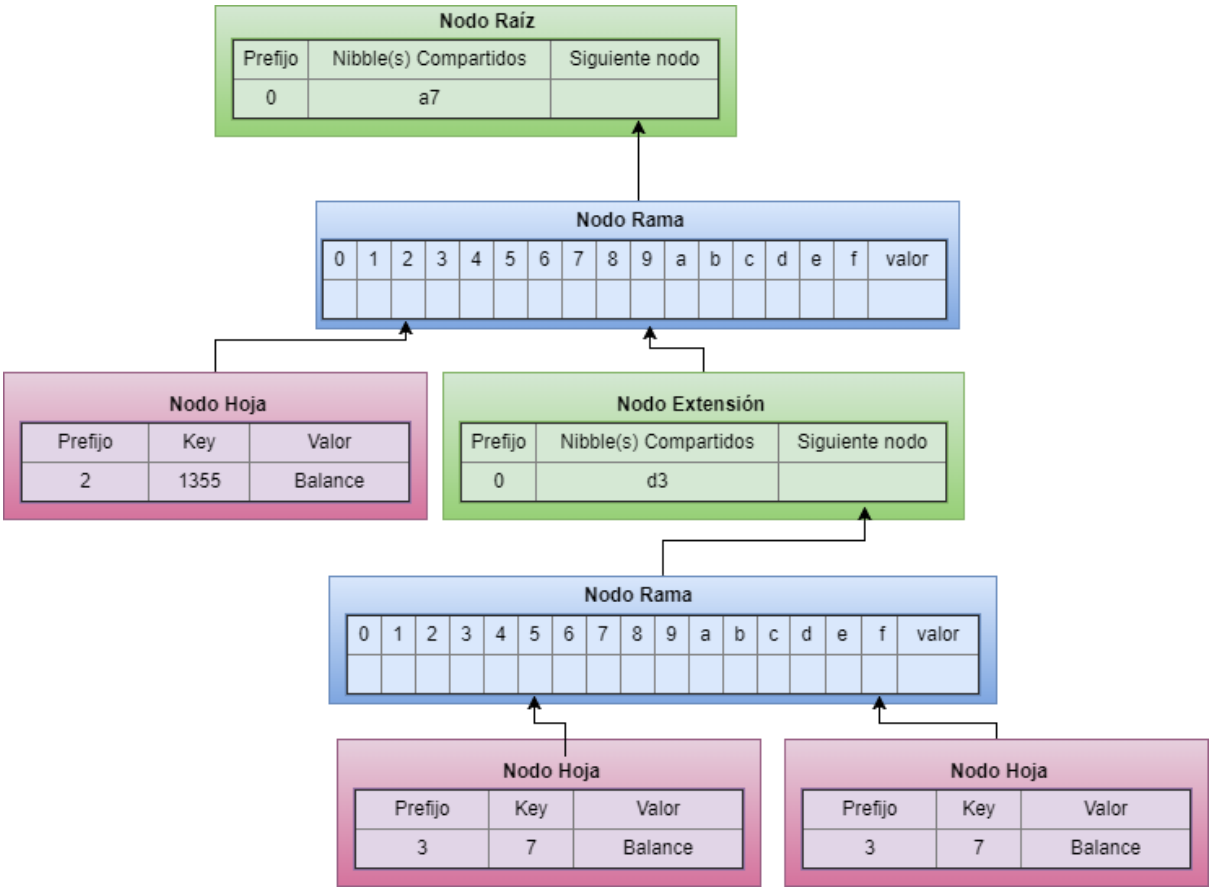


Figura 5: Construcción de Trie basada en Nodos.

Observando la figura 5, vemos la función de cada uno de los diferentes nodos que existen:

- Nodo Raíz: Nodo padre, del cual cuelgan todos los nodos.

- Nodo Rama: Éste tiene una estructura de 16 valores (referentes al código hexadecimal con el cual representamos los Hashes). De esta estructura cuelgan los diferentes nodos.

- Nodo Extensión: Este nodo se puede entender como una abreviatura. En lugar de crear otro nodo rama para identificar el camino a seguir, éste nos indica los nibbles (4 bits) que comparten los siguientes nodos.

- Nodo Hoja: Nodo que contiene el valor. En él encontramos un campo Key que indica el final de toda la palabra, y el valor (que en este caso sería el Balance).

Los prefijos solo existen para identificar los diferentes tipos de nodos.

14.2 - Creación de bloque

La creación del bloque es un apartado bastante sencillo, ya que no implica la modificación del estado. Si no que sólo implica consultas a la base de datos.

En primer lugar, se crea la cabecera asignando al bloque padre el último bloque de la cadena, al número de bloque se le asigna el del último de la cadena más uno, al coinbase se le establece la que nos ha indicado previamente el algoritmo de consenso y al timestamp el tiempo actual de la máquina en formato UNIX.

Una vez que se tiene la cabecera construida, se eligen las transacciones que formarán parte del bloque. Recordar que para los bloques pares sólo se pueden elegir transacciones IPv4 y para los impares transacciones IPv6. Existe un límite de transacciones, se añaden transacciones hasta que no quede ninguna en la pool de TX o se eligen hasta que el bloque tenga un tamaño máximo de 1MB.

Una vez se tienen las transacciones que formarán parte del bloque, se crea una cópia del estado en una Base de Datos provisional. Sobre este estado, se va a crear el Root Hash que se incluye en el bloque. Paralelamente, se creará el Transaction Trie que formará parte del bloque.

14.3 - Verificación

Para mantener la consistencia de la cadena de bloques y su confianza es necesario realizar las validaciones de bloques y transacciones antes de poder ser aplicadas al estado final.

14.3.1 - Transacción

Los pasos de validación de una transacción son:

1. Verificar que la transacción se haya firmado y tenga un sender válido.
2. Verificar que el Nonce sea el mismo que tenemos guardado en el State. Para que el Nonce sea válido se debe cumplir: TxNonce(Sign. Address) = StateNonce(Sign. Address + 1); si esto no se cumpliera quiere decir que la transacción no es válida. Por otro lado, si el nonce es menor que el valor esperado, nos indicará que alguien está intentando gastar dos veces un mismo valor. Al contrario, si es mayor puede ser que aún no se hayan realizado los cambios en el estado global.
3. Comprobar que la transacción tenga una categoría válida. Para diferenciar entre los diferentes tipos de transacción, a cada una de ellas se les asigna una categoría. En el caso que ésta corresponda a un traspaso de IPs, una transacción se considerará válida si el Sender tiene en su poder esas IPs. Por otro lado, en la casuística que la transacción haga referencia a un guardado de Locators o Mapping Servers, sólo se considerará la transacción válida si el que envía la TX es el mismo que la recibe.

El procedimiento de validación de la transacción se puede visualizar con este diagrama de estados:

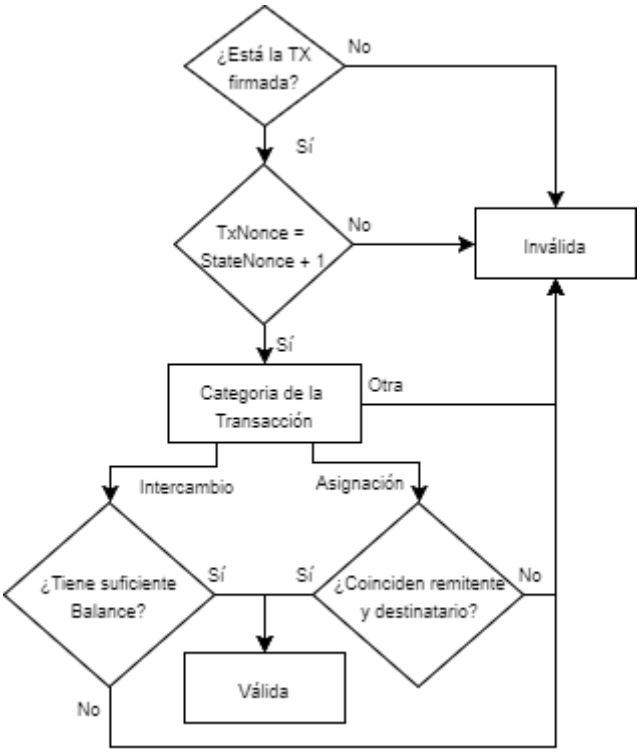


Figura 6: Proceso de validación de una transacción.

14.3.2 - Bloque

Los pasos de validación de un bloque son:

1. Verificar que el bloque se haya firmado y que la dirección que firma sea la indicada por el Consensus.

2. Validar el header. Para ello se realizan 3 comprobaciones:

    a. El Hash previo que indica el Header debe de ser igual al del último bloque de la cadena.

    b. El número de bloque tiene que ser igual al del último bloque más uno.

    c. El timestamp debe ser mayor al del último bloque.

3. Validar el Transaction Tree. Para ello, usando una Base de Datos temporal, volvemos a construir el Transaction Trie a partir de las transacciones que aparecen en el

cuerpo del bloque. Para que sea correcto, el resultado debe coincidir con el que indica el bloque.

4. Validar las transacciones. Se realiza el proceso descrito en el anterior apartado para cada una de las transacciones. Si alguna de las transacciones no es correcta, el bloque entero es descartado. Además, en este caso, cada una de las transacciones debe ser de un tipo (IPv4 o IPv6). Para bloques pares, transacciones IPv4. Para bloques impares, transacciones IPv6.

Podemos expresar la validación del bloque con este diagrama de estados:

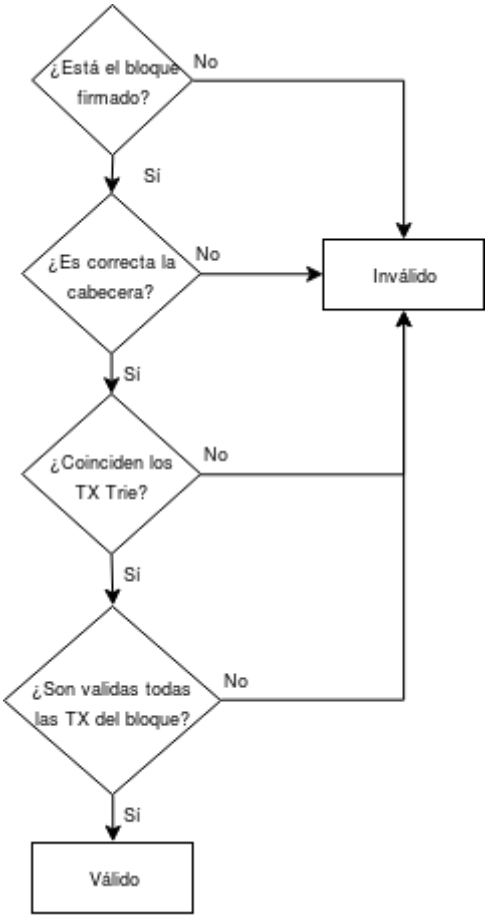


Figura 7: Proceso de validación de un bloque.

14.4 - Aplicación

Una vez se ha verificado la validez de los bloques y transacciones, se deben aplicar sobre la cadena.

En este apartado se explican los pasos que se siguen para ello.

14.4.1 - Transacción

La aplicación de la transacción es uno de los pasos más complejos ya que es la responsable de la modificación del estado global.

Para definir los pasos, debemos diferenciar los 4 tipos de transacciones ya que, cada una de ellas, tiene un efecto diferente sobre el estado.

14.4.1.1 - Allocate

Como se ha comentado anteriormente, esta transacción es la más parecida a la que encontramos en BlockChains, simplemente es un intercambio de direcciones de una cuenta a otra.

Esta transacción pero, tiene un efecto especial. Mientras lo obvio sería que sólo cambiáramos el estado de las dos cuentas a las que hacen referencia la transacción, al tener un método de delegación, es posible que algunas de las direcciones IP previamente delegadas a otra cuenta, se vean afectada por el cambio. Así, es necesario revocar esa delegación y ceder las direcciones al nodo receptor en cuestión.

Para ello, se consulta el balance del remitente y se buscan las IPs afectadas (recordemos que en el balance guardamos tanto IPs que nos ceden, cómo las que cedemos).

Una vez hecho esto, para cada una de las cuentas se les eliminan esas direcciones, se suman las IPs al receptor y se eliminan del remitente. Esto implica tanto modificar el World State Trie como el Patricia.

Finalmente, se incrementa el Nonce del remitente.

14.4.1.2 - Dellegate

Esta transacción permite delegar una IP (o rango) a otra cuenta sin perder su propiedad.

Igualmente que en el caso anterior, si se delega una IP que ya esté delegada a otra cuenta, esta se tiene que eliminar de la cuenta actual; y modificar el Balance para añadir al remitente las IPs delegadas, y al receptor las IPs recibidas.

Este tipo de transacción no tiene efecto sobre el Patricia ya que no cambia la propiedad de ninguna de las IPs.

De nuevo, finalmente, se incrementa el Nonce del remitente.

14.4.1.3 - Map Server y Locator

Estos dos tipos de transacciones son los más simples de aplicar ya que solo afectan al remitente. Además, esta transacción elimina la información que se tenia anteriormente de los Mappings y Locators. Así que no se debe hacer ninguna comprobación.

Se modifica el balance del remitente en cuestión añadiendo la información que indica la transacción.

Para terminar, y como en los otros casos, se incrementa el Nonce.

14.4.2 - Bloque

La aplicación del bloque es el paso más crítico de todos. Los bloques construyen poco a poco la cadena y un error en él puede provocar que toda la cadena vuelva inválida.

Antes de la validación del bloque, se buscará en la cabecera de éste cúal es el hash del bloque anterior al que debe ir concatenado. Si el Hash no coincide con el actual pero no existe en la DB, es posible que haya llegado antes que el bloque anterior, así que se añade a la cola de bloques pendientes. Si se da el caso de que el Hash no coincide con el actual pero existe en la DB, se descarta el bloque. Por otro lado, si el Hash coincide con el actual, se inicia la validación y una vez terminada se da inicio a su aplicación.

Ésta empieza por realizar una copia del estado en el que se encuentra la cadena. En caso de que por alguna razón, el proceso falle, se podrá recuperar el estado anterior. Igualmente, se realiza una copia del Patricia.

El primer paso a realizar es el más costoso del proceso. Se trata de la aplicación de todas las transacciones una a una. Si el proceso no falla, se guarda el bloque dentro de la base de datos, lo que conlleva 4 acciones principales:

- Guardar el bloque codificado en RLP en la LevelDB referenciado por su Hash.
- Cambiar el Head Hash de la cadena por el del bloque añadido.
- Añadir este bloque al anterior como su hijo.
- Aumentar el número de bloque para la creación.

Destacar que durante este paso, las transacciones se guardan referenciadas por dos coordenadas. Una es el número de bloque al que pertenece, y la otra la posición dentro de la lista de transacciones que tiene el bloque, lo que facilita el acceso a una transacción si no se dispone del Hash de ésta.

14.5 - Inicialización

Cuando se inicializa el prototipo la cadena se debe cargar. Es posible que ésta ya esté inicializada o se deba crear a partir del bloque génesis.

14.5.1 - Bloque Génesis

Si la cadena no tiene ningún bloque, es decir que se crea de 0, ésta debe ser inicializada a partir del bloque génesis.

Para ello, se crea un estado vacío. Se obtiene la estructura Alloc del fichero genesis.json del cual hablamos anteriormente. Ésta estructura contiene el Balance de todos los nodos en el estado inicial. Uno a uno, se van añadiendo y aplicando al State. Igualmente se crea el Patricia.

Una vez creado el State, se guarda el State Root y el Header Hash. Y se le asignan los valores al bloque.

Destacar que, al no tener creador. El bloque génesis es el único que se acepta sin estar firmado.

14.5.1 - Carga

El proceso anterior sólo se realizará durante la primera ejecución del prototipo o sólo si al usuario le interesa reinicializar la cadena. Pero la mayoría de las veces, la cadena será inicializada a partir de un estado ya existente.

Para ello, primeramente se carga la variable que contiene el Head Hash. Una vez obtenido, se consigue el bloque. Es posible que en algunos casos coincida con el génesis; si es así, repetimos el proceso descrito anteriormente; por el contrario, con los datos del bloque cargado, podemos reconstruir el estado anterior.

Resaltar que durante esta reconstrucción existe un parámetro llamado “Header Depth”, que nos indica, en el momento de la reconstrucción, hasta cuantos bloques atrás se debe verificar la integridad de la cadena.

En caso que la cadena no fuera correcta, se cargaría desde el último punto correcto.

15 - Experimentos

Para comprobar el rendimiento del módulo realizado en cuestión se han realizado un conjunto de mediciones.

15.1 - Definición

En este caso, nos interesa medir el tiempo de creación y aplicación de bloque. Para ello, se ha elegido un rango de transacciones por bloque (que irá de 0 a 250) y se van a crear y aplicar 15 bloques con ése número de transacciones.

Para simular el peor escenario posible, se van a crear transacciones de tipo Allocate y Dellegate ya que son las que más impacto y coste tienen sobre el sistema.

Destacar, que los bloques se han creado y aplicado continuamente. Así, la creación y aplicación de un nuevo bloque puede afectar al estado que había creado uno anterior.

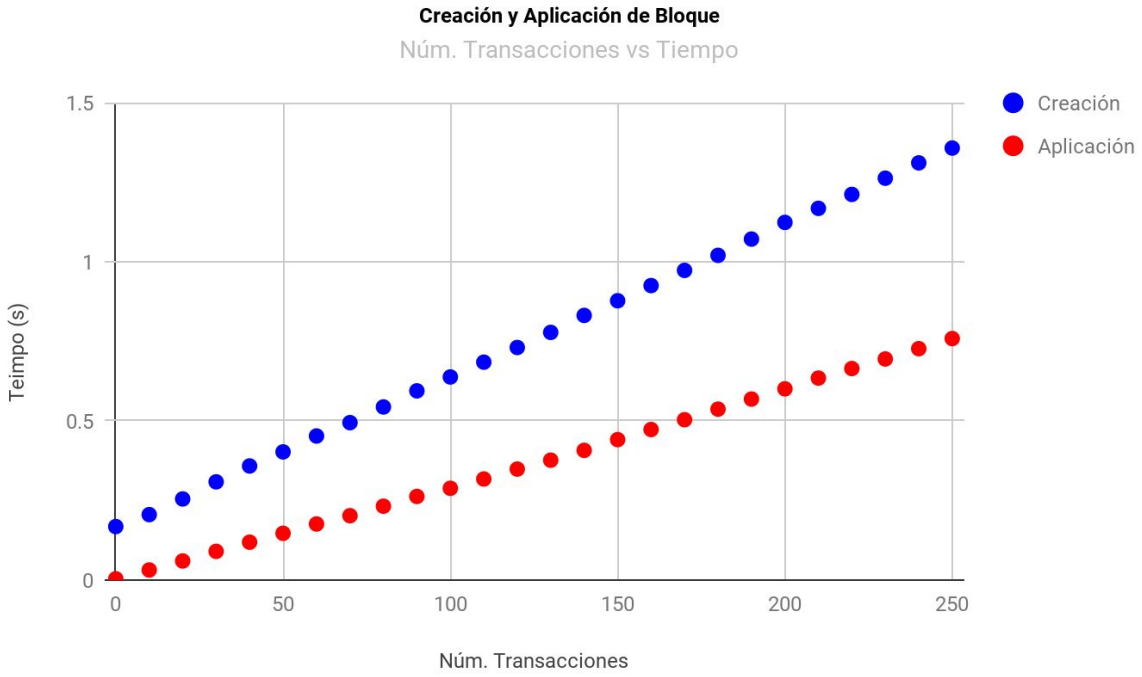
15.2 - Data Set

Para realizar los experimentos se ha elegido un ordenador de sobremesa con las siguientes características:

- **Procesador:** Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz
- **RAM:** 8094MB
- **OS:** Ubuntu 16.04.3 LTS
- **HDD:** TOSHIBA External USB 3.0

15.3 - Resultados

Una vez realizados los distintos conjuntos de pruebas se ha obtenido la siguiente gráfica:



Gráfica 1: Creación y aplicación de bloque (TX vs. Tiempo).

Como se puede observar en la gráfica los puntos que se encuentran en ella forman claramente una línea. Esto es debido a que existe una relación lineal entre Tiempo y número de transacciones por bloque (a razón de 0.04 segundos para cada 10 transacciones en la creación y 0.03 segundos en la aplicación).

No es apreciable en la gráfica, pero la desviación estándar es de milisegundos. Lo que nos indica que todos los datos que se han obtenido son muy representativos.

Si se piensa bien en el proceso de Creación y Aplicación de un bloque, se puede observar como prácticamente se realizan las mismas operaciones en uno y otro. Entonces, ¿dónde se origina la diferencia que se observa en la gráfica?. En este punto es cuando entra la copia temporal del estado que se realiza durante la creación.

En esta fase, se realiza una copia del estado (al igual que en la aplicación), pero ésta se aplica sobre una base de datos temporal (simulada con un diccionario). Esta aplicación, marca la diferencia notada en la gráfica.

Además, vemos que esta diferencia va creciendo a medida que aumentamos el número de transacciones. Ello es debido principalmente por dos razones:

- El estado, aunque no mucho, va creciendo a medida que se van aplicando los cambios (direcciones más repartidas, aparecen nuevas cuentas, ...). La copia y aplicación, por tanto crecen en tiempo.
- Al ser una Base de Datos simulada, las consultas no están tan optimizadas como las de la Base de Datos normal (LevelDB). Aunque la diferencia por consulta sea inapreciable, a mayor número de transacciones, más consultas, y por ello, más tiempo.

15.4 - Test

Para comprobar el correcto funcionamiento de cada uno de los módulos, se han creado un conjunto de scripts que simulan el comportamiento que debería tener el prototipo en un escenario real.

16 - Trabajo futuro

Aunque se ha conseguido realizar la implementación del sistema deseado, hay algunos aspectos que se podrían mejorar.

Primeramente, la entrada de datos ahora mismo es un poco engorrosa. Para poder crear transacciones, éstas deben ser definidas previamente en un fichero de texto. La idea principal sería, añadir para cada uno de los módulos una consola con la que fuese más fácil la interacción.

Un apartado del que no se ha incidido mucho ha sido el tamaño de bloque. En éste momento, el bloque tiene las transacciones que su tamaño le permite, pero no se tiene en cuenta el coste real de la aplicación de estas transacciones.

Es posible que una simple transacción modifique todo el estado de la cadena. Para poner un ejemplo (radical pero que podría suceder), supongamos que en el estado inicial una sola cuenta tiene todas las direcciones IPv4 (0.0.0.0/0). Esta cuenta, va delegando diferentes rangos de direcciones hasta que las tiene repartidas por toda la cadena. Si en algún momento la cuenta que ha delegado todas las direcciones decidiese enviarlas a otra cuenta, todo el estado de la cadena se vería afectado, ya que la delegación existente ya no sería válida.

Este movimiento tendría entonces un coste muy superior a cualquiera de las transacciones anteriores.

Para evitarlo, se podría implementar un sistema parecido al que existe en Ethereum, dónde no hay tamaño de bloque, pero si existe un límite de coste por bloque. Así, se podría calcular el coste que tiene una transacción y definir un límite para cada bloque.

17 - Conclusiones

El objetivo principal del proyecto era la creación de una BlockChain para conseguir asegurar las direcciones IP.

Cada módulo ha sido capaz de, utilizando tecnologías existentes, resolver los problemas planteados al principio.

Por lo que hace referencia a la capa de persistencia, se ha conseguido un sistema eficiente de creación, validación y aplicación de bloques; así como el guardado de datos necesario para mantener la cadena de bloques que permite construir la DDT que describe LISP. Para ello, se ha realizado una implementación basada en las dos BlockChains más conocidas del momento como son Bitcoin y Ethereum.

18 - Agradecimientos

Finalmente, agradecer toda la ayuda que nos ha ofrecido tanto el director del proyecto, Albert Cabellos, como Jordi Pallisé Vilanova (que realiza su doctorado con este estudio),

dedicando tiempo y esfuerzo para resolver los diferentes problemas que han ido surgiendo durante el transcurso de todo el trabajo.

Además, agradecer a Albert López, las explicaciones aportadas por lo que hace referencia al protocolo OOR, que nos han permitido realizar una implementación acorde a las características que describe LISP.

Y por supuesto, dar las gracias a Hamid Latif, Carlos Piris y Eric Garcia, los otros 3 integrantes que han formado parte del desarrollo, por su dedicación y compromiso por el bien del resultado final.

19 - Bibliografía y referencias

[1] Paul, Sanjoy. “Multicast Backbone of the Internet (MBone).” Multicasting on the Internet and its Applications, vol. 37, 1998, pp. 129–138., doi:10.1007/978-1-4615-5713-5_12.

[2] Andersen, David G., et al. “Best-Path vs. multi-Path overlay routing.” Proceedings of the conference on Internet measurement conference - IMC 03, 2003, pp. 91–100., doi:10.1145/948217.948218.

[3] Zhao, B.y., et al. “Tapestry: A Resilient Global-Scale Overlay for Service Deployment.” IEEE Journal on Selected Areas in Communications, vol. 22, no. 1, 2004, pp. 41–53., doi:10.1109/jsac.2003.818784.

[4] Lua, Eng Keong, et al. “A survey and comparison of peer-To-Peer overlay network schemes.” IEEE Communications Surveys & Tutorials, vol. 7, no. 2, 2005, pp. 72–93., doi:10.1109/comst.2005.1610546.

[5] Rodriguez-Natal, Alberto, et al. “LISP: a southbound SDN protocol?” IEEE Communications Magazine, vol. 53, no. 7, 2015, pp. 201–207., doi:10.1109/mcom.2015.7158286.

[6] D. Farinacci et al., “Locator/ID Separation Protocol (LISP),” Feb. 2013; https://tools.ietf.org/html/rfc6830

[7] Phung, Dung, et al. “The OpenLISP control plane architecture.” IEEE Network, vol. 28, no. 2, 2014, pp. 34–40., doi:10.1109/mnet.2014.6786611.

[8] Stockmayer, Andreas, et al. “JLISP: An Open, Modular, and Extensible Java-Based LISP Implementation.” 2016 28th International Teletraffic Congress (ITC 28), 2016, doi:10.1109/itc-28.2016.135.

[9] J. Paillisse “An analysis of the applicability of blockchain to secure IP addresses

allocation, delegation and bindings.”

[10] Cabellos, Alberto, and Damien Saucez. “An Architectural Introduction to the Locator/ID Separation Protocol .” 2 Apr. 2015, pp. 5–14., datatracker.ietf.org/doc/draft-ietf-lisp-introduction/?include_text=1.

[11] Cabellos, Alberto, and Damien Saucez. “An Architectural Introduction to the Locator/ID Separation Protocol .” 2 Apr. 2015, pp. 10–14., datatracker.ietf.org/doc/draft-ietf-lisp-introduction/?include_text=1.

[12] Morabito, Vincenzo. “The Blockchain Paradigm Change Structure.” Business Innovation Through Blockchain, 2017, pp. 3–20., doi:10.1007/978-3-319-48478-5_1.

[13] Nakamoto, Satoshi. “Bitcoin: A Peer-to-Peer Electronic Cash System.” 9 Jan. 2009, https://bitcoin.org/bitcoin.pdf

[14] Wood, Gavin. “Ethereum: A Secure Decentralized Generalised Transaction Ledger.” 7 Aug. 2017, ethereum.github.io/yellowpaper/paper.pdf.

[15] Gilad, Yossi, et al. “Algorand.” Proceedings of the 26th Symposium on Operating Systems Principles - SOSP 17, 2017, doi:10.1145/3132747.3132757.

[16] Kiayias, Aggelos, et al. “Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol.” Advances in Cryptology – CRYPTO 2017 Lecture Notes in Computer Science, 2017, pp. 357–388., doi:10.1007/978-3-319-63688-7_12.

[17] “Locator/ID Separation Protocol.” LocatorID Separation Protocol, www.lisp4.net/beta-network/

[18] “Cuanto me costará imprimir el PFC?” Cuanto me costará imprimir el PFC? • Foros de SóloIngeniería.NET, www.soloingenieria.net/foros/viewtopic.php?f=2&t=25679 Accessed 08 Oct. 2017