Universidad Politécnica de Cataluña
Facultad de Informática de Barcelona

Grado en Ingeniería informática
Especialidad de Computación

Design and implementation of the Transactional and Communication layer of a Blockchain to secure IP prefixes

Autor:
Carlos Piris Yamuza

Director:
Albert Cabellos Aparicio

Codirector:
Jordi Paillissé Vilanova

Enero 2018

Índice general

Índice de tablas

Índice de figuras

1. Definición del alcance y contextualización

1.1. Contexto

1.1.1. Introducción

Este proyecto es un Trabajo de Final de Grado de la especialidad de Computación de la Facultad de Informática de Barcelona (Universidad Politécnica de Cataluña). Se trata de un proyecto de modalidad A en el cual los investigadores del Departamento de Arquitectura de Computadores (DAC) junto con la empresa Cisco Systems estudian el uso de la tecnología blockchain en las redes de comunicaciones[1].

El objetivo principal del proyecto es construir un prototipo funcional de blockchain que asegure la información del enrutamiento de Internet. Esta contendrá información sobre a quién le pertenece un rango de direcciones IP y datos del protocolo LISP[2] (Locator/ID Separation Protocol). Protocolo en el que trabajan los investigadores del DAC como posible solución al problema de escalabilidad en Internet debido a la creciente cantidad de dispositivos conectados.

Finalmente se probará y se analizará el funcionamiento de la blockchain junto con la red de pruebas LISP Beta Network que disponemos en la universidad.

1.1.2. Actores Implicados

A continuación, se definirán los principales actores en este proyecto, es decir, toda persona u organización que tenga algún tipo de relación con el proyecto o pueda estar interesada.

Director

El director de este proyecto es Albert Cabellos, su papel es el de supervisar que el proyecto alcance los objetivos marcados y que se cumpla el calendario estipulado. Además, guía al grupo de desarrolladores.

Codirector

El codirector de este proyecto es Jordi Paillissé, es el desarrollador jefe, es quien tiene conocimiento de todas las partes del proyecto y está en constante contacto con los desarrolladores.

Desarrolladores

El grupo de desarrolladores está formado por cuatro miembros, todos estudiantes de la FIB y como proyecto de final de carrera para cada uno de los miembros. Cada miembro tiene asignada una parte del prototipo independiente del resto de la cual es el responsable. Los miembros del proyecto son Hamid Latif, Miquel Ferriol, Eric Garcia y yo mismo.

Organizaciones

La empresa Cisco colabora con el proyecto y espera ver los resultados para ver si puede sacar un producto similar al mercado. Además, este proyecto ha sido presentado a la IETF (Internet Engineering Task Force), organización que vela por la arquitectura de internet y los protocolos que la conforman, como mejora para internet. El código fuente del proyecto será publicado bajo una licencia libre para que cualquier organización podrá contribuir.

1.1.3. Formulación del Problema

Internet ha ido creciendo exponencialmente en las últimas décadas lo que ha creado una gran y compleja infraestructura de redes y protocolos junto a él. A medida que Internet sigue creciendo va demandando cambios en la red que son muy costosos de realizar.

Los investigadores de la universidad trabajan en mejorar las redes de comunicaciones actuales, para ello colaboran en el desarrollo del protocolo LISP y en la plataforma OpenOverlayRouter [3] (OOR) que permite desplegar Overlay Networks (redes superpuestas).

LISP es un protocolo de encapsulación originalmente diseñado para reducir el rápido crecimiento del número de rutas en la tabla global de enrutamiento IPv4. Funciona separando semánticamente los dispositivos que se conectan en: EndPoint Identifiers (EIDs), los dispositivos finales, y Routing Locatos (RLOCs), los dispositivos de la capa de enrutamiento. Luego encapsula los paquetes EID con su RLOC, para ello necesita encontrar qué RLOC está asociado con cada EID.

Una Overlay Network es una red virtual de nodos que está construida sobre una o más redes subyacentes. Con el objetivo de implementar servicios que no están disponibles en las subyacentes. OpenOverlayRouter es una plataforma de código abierto que permite crear este tipo de redes utilizando el protocolo LISP.

Con estas tecnologías se puede modificar el comportamiento de una red, mejorándolo sin tener que modificarlo físicamente permitiendo así abaratar los costes.

El problema surge en la seguridad de los datos. Las direcciones IP de Internet no tienen ningún tipo de identificador para saber a quién pertenecen, ¿cómo podemos saber que no nos están engañando?

Actualmente, el protocolo LISP para localizar un RLOC sabiendo un EID cuenta con un sistema llamado DDT (Delegated Database Tree), es un directorio jerárquico compuesto por nodos en una estructura en árbol, que permite encontrar la ruta deseada. Conceptualmente equivalente al DNS. Este sistema es difícil de gestionar, se necesitan nodos diferentes y los cambios son manuales. Además es un sistema centralizado y para asegurar la información se utiliza un sistema de certificados digitales (PKI) que complica la configuración.

Los investigadores de la UPC vieron una forma mejor de asegurar los datos utilizando una blockchain en lugar del sistema DDT. La tecnología blockchain permite guardar información de forma muy segura y además por su diseño cuenta con un registro histórico de todos los cambios. También es un sistema descentralizado, no se depende de un organismo central, sino que los propios usuarios.

La tecnología blockchain también presenta una serie de retos, en parte debidos a la juventud de esta tecnología, por ello se pretende aprender sobre ella e investigar como se puede aplicar a las redes de comunicaciones.

1.2. Blockchain

Esta tecnología surgió en el año 2009 como parte de la criptomoneda Bitcoin[4]. Como su nombre indica, esta guarda la información en una cadena de bloques, diseñada para evitar su modificación una vez un dato ha sido publicado dentro de uno de estos bloques. Conceptualmente podemos decir que se trata de una gran base de datos segura y confiable.

Los datos almacenados son transacciones entre los usuarios. Un usuario adjunta datos en una transacción junto con su firma digital y su llave pública. Normalmente los datos adjuntos son algún tipo de activo o token, algo único que no se puede duplicar (ej, monedas en Bitcoin). A continuación, una transacción es emitida al resto de nodos en la red, estos la guardan temporalmente hasta que uno de ellos, en un tiempo fijado, la introduzca en un nuevo bloque que transmitirá de nuevo a la red. La red de conexión de los nodos es a través de una red Peer to Peer (P2P).

Cada nodo en la red guarda la blockchain entera de forma local y verifica todo su contenido. Además, la mayoría de blockchains dan una recompensa a los nodos cuando añaden un nuevo bloque a la cadena, aunque no es estrictamente necesario. La tabla 1.1 presenta una visión general de los elementos que contiene un bloque.

Número de bloque	Hash (bloque anterior)	Hash (todas las trans. del bloque)	Firma del creador del bloque
Transacción 1			
Transacción 2			
...			
...			
Transacción N			

Tabla 1.1: Estructura común de un bloque

Se usan dos mecanismos básicos para proteger la información contenida en la cadena, la cadena de firmas y el algoritmo de consenso.

1.2.1. Cadena de firmas

La cadena de firmas opera a nivel de transacción. Cada remitente y receptor tiene su pareja de llaves pública-privada. Para cambiar un token de propietario, el remitente debe generar una transacción donde firma los datos y el hash de la llave pública del receptor con su llave privada.

El remitente también comparte su clave pública. La tabla 1.2 presenta la estructura de una transacción.

Llave pública remitente	Firma remitente (Llave privada)	Datos	Hash (Llave Publica recetor)

Tabla 1.2: Estructura común de una transacción

Estas normas aseguran por un lado el dueño del token tiene todo en control y solo él puede autorizar una transferencia. Por otro lado, cuando se transfiere un token es irreversible ya que el receptor pasa a tener todo el control.

1.2.2. Algoritmo de consenso

El algoritmo de consenso es la parte central de la blockchain y se encargar del encadenamiento de los bloques. Su objetivo principal es proveer de un conjunto de normas bien definidas que deben cumplir todos los usuarios para mantener la consistencia de la cadena. Este debe actuar frente a los forks (bifurcaciones de la cadena), los participantes deben ponerse de acuerdo sobre que cadena es la válida. También debe decidir que participante puede o no añadir un nuevo bloque a la cadena.

Los algoritmos de consenso más populares son PoW (proof of work) y PoS (proof of stake)

Proof of Work (PoW)

En Proof of Work (prueba de trabajo) los nodos tienen que resolver un problema matemático para poder añadir un bloque a la cadena. La cadena valida es aquella que más bloques contenga. Si alguien intentase modificarla necesitaría un poder computacional más elevado que el resto de nodos de la red juntos, algo poco probable y requeriría de mucha energía eléctrica.

El principal inconveniente es su enorme gasto de recursos, haciendo que no sea viable de aplicar a nuestro prototipo. Además, requiere que se actualice el hardware frecuentemente.

Proof of Stake (PoS)

En Proof of Stake (prueba de participación), los participantes con más participación en la red, es decir, aquellos que más tokens poseen, tienen más probabilidad de añadir un bloque a la cadena. Se asume que un usuario con mucha participación velará por la seguridad de la cadena (no se perjudicaría a el mismo) de este modo los bloques que genere no serían maliciosos.

PoS presenta una serie de ventajas frente a PoW:

- No requiere de hardware especial ni se gastan recursos en resolver cálculos matemáticos que añadan dificultad.
- Un atacante necesitaría de mucha participación para poder atacar la red, independientemente del poder computacional que tenga.

Por otro lado, PoS añade una seria de inconvenientes que permiten ataques si no se ponen mecanismos para evitarlo. Investigaremos posibles formas de minimizar ataques.

Se propone usar PoS para el prototipo debido a sus ventajas frente a PoW, aunque este aumente la complejidad del proyecto debido a que requiere de una serie de algoritmos que lo protejan frente a los ataques.

1.2.3. Beneficios

El uso de la blockchain ofrece un nivel de seguridad en los datos equivalente a los certificados (PKI) usados por DDT en LISP y además aporta facilidades en el manejo como por ejemplo en el rekeying.

Al ser descentralizado una empresa puede gestionar sus direcciones IP a su manera sin necesidad que pedir certificados a otros organismos.

Toda la información es pública, se envida cualquier tipo de censura y queda constancia de todos los movimientos permitiendo así la auditoria.

1.2.4. Contras

Es necesario que cada usuario guarde una copia de toda la cadena, podría llegar a ocupar mucho espacio, las estimaciones previstas no indican que pueda ser un problema. Para arrancar un nodo nuevo en la red necesita descargar todos los bloques de la cadena, provocando una espera en el inicio.

La veracidad de la cadena depende de que sean los propios usuarios los que se porten correctamente. Si muchos usuarios se comportan mal, acabaría perjudicando a la blockchain.

Independientemente de la seguridad de la cadena también hay que velar por la seguridad de la red P2P que comunica los nodos. Ya que usando un algoritmo de PoS no podemos permitir que algún nodo quede aislado y no pueda generar un nuevo bloque. Hay que evitar a toda costa un ataque de denegación de servició (DoS).

En nuestra blockchain no se puede permitir que una dirección de internet quede perdida, si un usuario perdiese sus llaves o emite una transacción a una dirección sin dueño hay que permitir que se vuelvan a emitir esas direcciones que han quedado perdidas.

1.3. Objetivos

Debido a la magnitud del proyecto, este se divide en cuatro módulos independientes uno para cada estudiante como mencione anteriormente. La figura 1.1, muestra un diagrama del proyecto.

Las partes que debo desarrollar son las siguientes:

- Montar una transacción (llaves públicas-privadas, firmas, etc) que incluya las direcciones de Internet junto con los EID y RLOC de LISP en lugar de dinero digital.

- Crear una red de comunicación P2P que permita a los nodos de la red comunicarse, enviar y recibir transacciones y bloques además del bootstrap inicial.
- Investigar sobre la seguridad de la red P2P en una blockchain tipo PoS.

Mis compañeros desarrollaran las siguientes partes:

- Hamit Latif: La interfaz con OOR, la entrada del usuario y el keystore.
- Èric Garcia: Algoritmo de consenso.
- Miquel Ferriol: La chain y el diseño de los bloques.

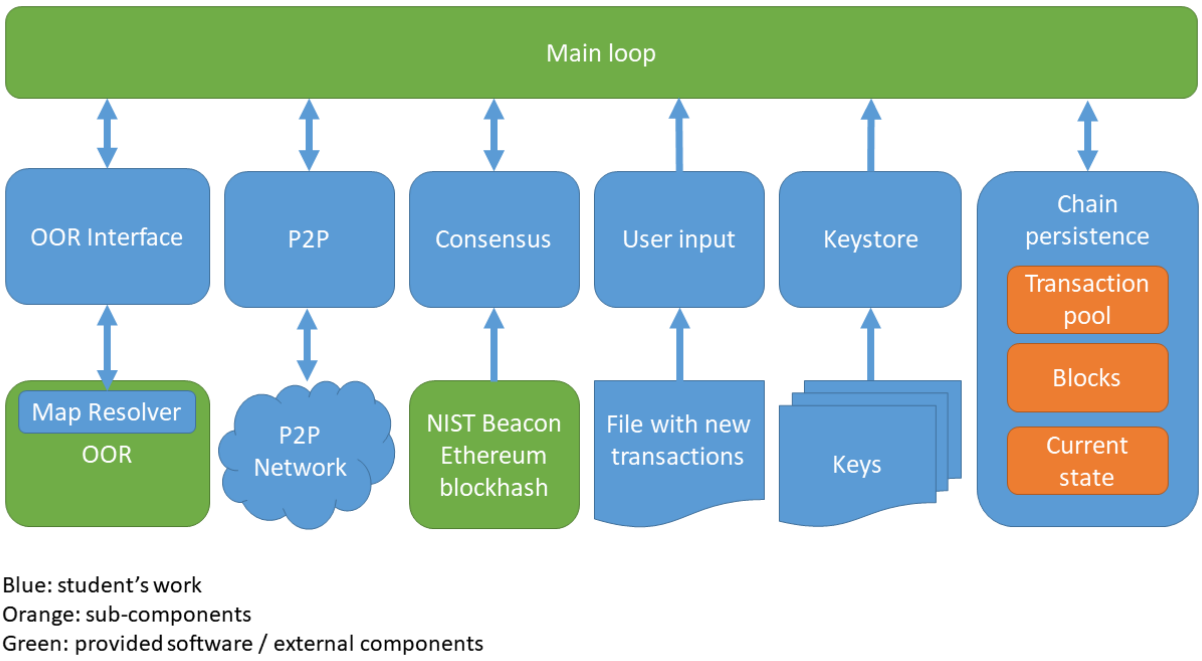

Figura 1.1: Diagrama del proyecto

1.4. Retos

La blockchain que queremos construir presenta una serie de problemas con los que hay que lidiar a diferencia de las blockchains que implementan las criptomonedas.

Cuando un usuario de una criptomoneda emite una transacción, este debe pagar una tasa al usuario que la incluye en el bloque. La tasa es un incentivo para que la transacción sea incluida en la cadena, esta puede ser cero, pero eso hará que quizás no se incluya nunca, en cambio una tasa alta hará que entre más rápido. Nosotros no trabajamos con ningún tipo de dinero virtual sino con direcciones de internet, lo cual hace imposible el pago de algún tipo de tasa. Esto podría provocar que no se incluyesen transacciones en un bloque, aunque es poco probable ya que los

usuarios son empresas que se beneficiarían de la seguridad que ofrece incluir la información. El problema estaría en que dos usuarios podrían emitir la misma transacción entre ellos sin parar ya que no existe ninguna tasa y llenarían la blockchain de datos inútiles.

Una blockchain no contempla la opción de prestar un token, en nuestro caso si una empresa de telecomunicaciones quiere asignar una dirección IP a un cliente, este sería el dueño de la dirección para siempre. Hay que añadir una nueva funcionalidad a la blockchain que permita asignar una dirección a un cliente sin que este sea el propietario. No se pueden perder direcciones IP, cuando un usuario de una criptomoneda pierde sus llaves o envía el dinero a un lugar equivocado este no puede recuperarse de ninguna forma.

Criptográficamente una blockchain es muy segura pero una red de comunicaciones puede sufrir un ataque como por ejemplo un *denial of sevice* (DoS) o un *man in the middle*. Actualmente existen estudios como el de Maria Apostolaki et al.[5] que demuestran que se pueden aislar nodos en la criptomoneda Bitcoin. Poder realizar este ataque en una blockchain tipo PoS como la nuestra sería muy grave ya que aislando pocos nodos que tengan mucha participación disminuiría la seguridad de nuestra cadena. Buscaremos métodos para impedir que esto pueda ocurrir.

1.5. Estado actual

Actualmente ya se está usando y comercializando el protocolo LISP y las Overlays programables, pero ninguno cuenta con algún tipo de blockchain que asegure y descentralice sus datos. La mayoría de blockchains que existen se usan principalmente para crear monedas digitales, aunque poco a poco se empiezan a usar con otros fines.

No he encontrado una blockchain que haya tratado de incluir direcciones IP en las transacciones. Tampoco parece existir una solución a la seguridad de la red aunque actualmente existen modelos que garantizan bastante seguridad.

1.6. Contextualización

En este proyecto se desarrollará un sistema blockchain basado en PoS, debido a la inviabilidad de usar otro tipo de algoritmo de consenso, para reemplazar el sistema DDT del protocolo LISP. Me corresponden las tareas montaje de las transacciones y creación de una red P2P. Además investigar como minimizar los ataques a la red.

1.7. Estado del arte

Actualmente existes diversas blockchains en el mundo. Pero ninguna de ellas está pensada para la gestión de direcciones de internet, la gran mayoría de ellas son en forma de criptomoneda.

Para realizar nuestro proyecto hemos analizado las criptomonedas más populares en busca de adaptar alguna de ellas a nuestras necesidades. En concreto hemos analizado dos de ellas Bitcoin y Ethereum, en las que basamos para crear nuestra propia blockchain.

También hemos analizado la red P2P de la criptonomeda Nem, que puntúa los nodos para detectar posibles nodos maliciosos.

1.7.1. Bitcon

La tecnología blockchain nació con Bitcoin[4] y actualmente es la criptomoneda más popular. Bitcoin es la prueba de que la blockchain funciona y es muy segura.

El hecho de ser la primera forma de blockchain hace que tenga algunos problemas en su diseño. Utiliza un algoritmo de consenso tipo PoW, lo que hace que tenga un gasto eléctrico desproporcionado y requiere hardware especializado. También tiene problemas de escalabilidad, de media solo llega a unas 7 transacciones por segundo que algunas veces ha generado congestiones en la red, aunque en las últimas actualizaciones prometen mejorarlo.

Las transacciones de Bitcoin (Figura 1.2) están entrelazadas entre ellas, para poder emitir dinero debes referenciar a una o varias transacciones anteriores en las que hayas recibido dinero. Para ello incluyen un script de entrada y salida, el script de entrada debe desbloquear el script de salida de la transaccion anterior y el script de salida son las condiciones que se deben cumplir para desbloquear el dinero que estamos emitiendo.

Bitcoin cuenta con todo un lenguaje de scripting para poder generar todo tipo de condición a la hora de desbloquear una transacción como por ejemplo multifirmas.

El sistema de transacciones de Bitcoin es complejo de implementar. Además al no existir el concepto de balance, siempre se emite todo el contenido de las transacciones a las que se hace referencia como entrada, emitiendo a uno mismo la parte que se quiere preservar. Dando como resultado unas transacciones muy complejas y grandes, ademas, difciles de gestionar en una base de datos y mas uso de banda ancha.

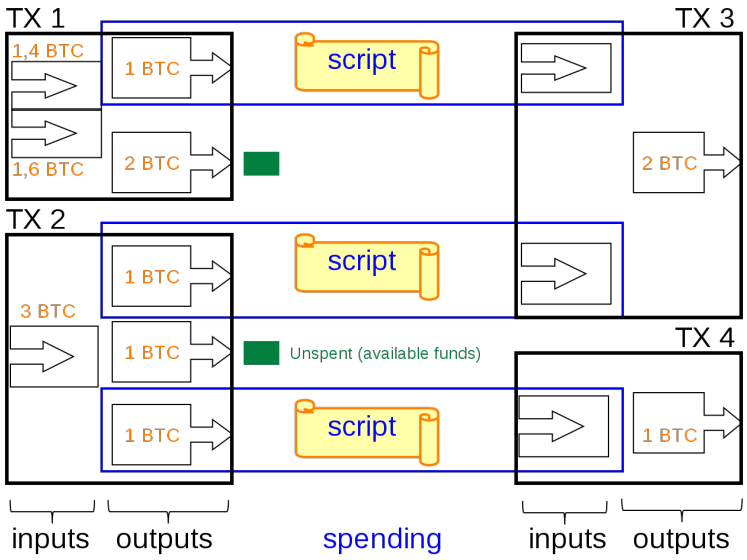

Figura 1.2: Esquema de una transacción de Bitcoin

1.7.2. Ethereum

Ethereum[6] es la segunda criptomoneda más popular y una de las blockchains más avanzadas. Su diseño mejora en gran parte al de Bitcoin.

Si bien también utiliza un algoritmo de consenso tipo PoW trabajan para migrar a uno de tipo PoS. Su sistema de transacciones ha sido simplificado a un sistema de cuentas, parecido al sistema bancario que utilizamos. Esto le dota de mayor escalabilidad y un gasto menor en espacio.

Al existir el concepto de balance, a la hora de enviar una transacción basta con indicar la cantidad y el receptor. Toda la parte de verificación y validación no se incluye en la transacción como en Bitcoin, esta parte la debe realizar la chain. Como resultado las transacciones ocupan muy poco espacio y son muy fáciles de crear.

Ethereum permite incluir programas informáticos en la propia cadena, llamado Smart Contrat, dotando sistema de la capacidad de trabajar no solo con monedas sino con cualquier cosa. Estos programas son muy superiores a los scripts que se pueden generar en Bitcoin.

Además Ethereum cuenta con una serie de librerías como RLP (Recursive Length Prex) que permite serializar objetos en datos binarios y también decodificarlos. También cuenta con librerías para generar las firmas.

En cuanto a la red P2P Ethereum implementa una tabla de hash distribuida (DHT), en concreto el algoritmo Kademlia[7]. Este algoritmo tiene mucha escalabilidad, permitiendo conectar miles de nodos en la red ofreciendo baja latencia y gran velocidad de transmisión. Este algoritmo genera una red muy robusta, aunque sigue existiendo la posibilidad de aislar un nodo, a su vez es muy difícil de implementar.

1.7.3. Nem

En 2015 aparece la criptomoneda Nem[8] incorporando una novedad en la red de comunicación. Esta es consciente de la posibilidad de que un nodo malicioso pueda intentar engañarnos o aislarnos de la red.

Los nodos son puntuados con un valor de confianza que influye a la hora de elegir los peers con los que comunicarse. Además cuenta con una lista negra por si detecta nodos maliciosos. Por ejemplo un nodo bastante ocupado o con mucha latencia obtendrá peor puntuación.

Nem periódicamente atendiendo a los cambios que se producen en la red, preguntando directamente a sus peers, va actualizando su lista de nodos priorizando todos aquellos con los que tiene más confianza.

De esta forma Nem consigue una red más resistente a posibles ataques que puedan ocurrir en una blockchain tipo PoS como la nuestra.

1.8. Alcance

Pretendemos poner en funcionamiento la blockchain y testear su funcionamiento si puede ser con servidores en la nube alrededor del mundo para mostrar que es posible mejorar la seguridad de internet usando esta nueva tecnología. El proyecto debe ser una prueba de concepto.

En base al éxito esperamos que la IETF acepte como un nuevo estándar para internet el uso de blockchains y que empresas como Cisco que colaboran en el desarrollo junto con otras desarrollen una versión comercial del prototipo para su uso a nivel mundial, haciendo así un internet más seguro y descentralizado.

Queremos publicar un paper explicando los resultados obtenidos en las pruebas y un estudio de los problemas que presenta el uso de una blockchain con una posible solución.

1.9. Metodología y rigor

El método empleado para este proyecto es iterativo. Es decir, se hacen una reunión semanal con todos los desarrolladores y el director del proyecto. Además, si es necesario de hace una pequeña reunión con el ponente. Algunos de los desarrolladores tenemos un PC asignado en un aula de la facultad, lo que permite que en caso de dudas podamos hablar con el compañero.

La metodología de trabajo que se seguirá la metodología ágil Scrum que nos permitirá gestionar el proyecto de una forma más amena y motivadora que con un método mas tradicional.

Todo el código se comparte con el resto de desarrolladores a través de la plataforma GitHub que además es un control de versiones. También se usa Google Drive para compartir documentos y esquemas.

2. Planificación temporal

2.1. Programa

El proyecto tiene una duración estimada de 5 meses, desde principios de setiembre a finales de enero, aunque durante los meses de verano de julio y agosto ya estuve trabajando en la parte de investigación debido su complejidad y magnitud.

2.2. Consideraciones globales

Cabe destacar que este proyecto se lleva a cabo por 4 desarrolladores junto con el desarrollador jefe, por lo tanto, se realizaran tareas en paralelo. Una vez estas tareas se vayan completando hay que juntarlas en un único programa para lograr un resultado satisfactorio, poder testear e investigar sobre las ventajas y desventajas que aporta una blockchain sobre el sistema actual.

Si algún desarrollador no terminase en el plazo previsto no podríamos probar la blockchain, aun así, cada parte es un TFG separado lo que no afectaría a los otros compañeros.

2.3. Plan de acción y valoración de alternativas

Se pueden producir desviaciones temporales en el diseño debido a que es un trabajo de investigación, no se han implementado todavía blockchains en el contexto en el que trabajamos, lo que puede producir que sobre la marcha nos encontremos con nuevos problemas que obliguen a rehacer partes del proyecto. Además, se hace muy difícil de predecir cuantas horas que llevara a una parte en desarrollarse.

2.4. Descripción de las tareas

Búsqueda y puesta en marcha

El jefe del proyecto busca información sobre el dominio del proyecto y prepara la repartición de trabajo entre los 4 desarrolladores. Se resolverán las dudas y se expondrán todas las propuestas encontradas. Esta fase no tiene ninguna dependencia de precedencia.

Investigación

Cada desarrollador investiga sobre todas las partes aunque no sea el encargado de programarla, se analizan las diferentes blockchains que hay en la actualidad para ver cuál de adapta mejor y ver qué cambios son necesarios para adaptarlas a nuestras necesitadas. Esta parte depende de la repartición de trabajo.

Análisis de requisitos y diseño

Se analizarán los requisitos necesarios para realizar cada una de las partes que nos han tocado y su diseño. Esta parte depende de la investigación.

Estructura de la transacción

En esta parte se desarrollará la estructura de la transacción, como y que datos se incluirán en cada una de ellas.

Red P2P

Independientemente del resto de partes, el proyecto necesita de una red de comunicación tipo P2P para comunicar los ordenadores entre sí. En esta parte de desarrollada esta red.

Resolución de forks

En esta parte se estudiará un algoritmo que gestione la resolución de forks que se puedan producir en la blockchain. Esta parte depende de que la red P2P esté terminada.

Creación de la API

Esta parte es la última del desarrollo, una vez programadas las otras partes se pone a disposición de los demás compañeros una API para poder interactuar con su código.

Documentación y Testeo

Una vez finalizadas las partes anteriores se terminará la documentación en la que se incluirán los resultados de las pruebas y la experiencia obtenida durante el desarrollo. A parte, se preparará la defensa del proyecto.

2.5. Recursos

Para la realización del proyecto se dispone de diferentes recursos hardware y software, se marcan en la siguiente tabla.

PC con Linux	Dispongo de un PC asignado en la facultad para poder trabajar desde allí y estar en contacto los desarrolladores.
Portátil personal	Para trabajar desde casa, dispongo de un ordenador portátil.
LISP beta network	Red LISP que utiliza la universidad para realizar las pruebas.
Servidor en la nube	Se espera disponer de máquinas virtuales en la nube distribuidas en diferentes países para probar la blockchain en un entorno de pruebas que simule la latencia real de la red.
GitHub	Servicio para albergar el código fuente del programa, que permite compartir el código con los compañeros y publicarlo bajo licencia libre. Además, cuenta con un gestor de versiones.

Librerías Python	Python dispone muchas librerías opensource las cuales podemos utilizar.
VS Code	Editor de texto para desarrollar la aplicación.
Sphinx	Herramienta para documentar la aplicación.
Google Docs	Para compartir documentos entre los desarrolladores.
LaTex	Para escribir el paper sobre el proyecto.
Libre Office	Para realizar la documentación del proyecto

Tabla 2.1: Recursos materiales

2.6. Estimación de horas

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 2.2: Estimación de horas

2.7. Diagrama de Gantt

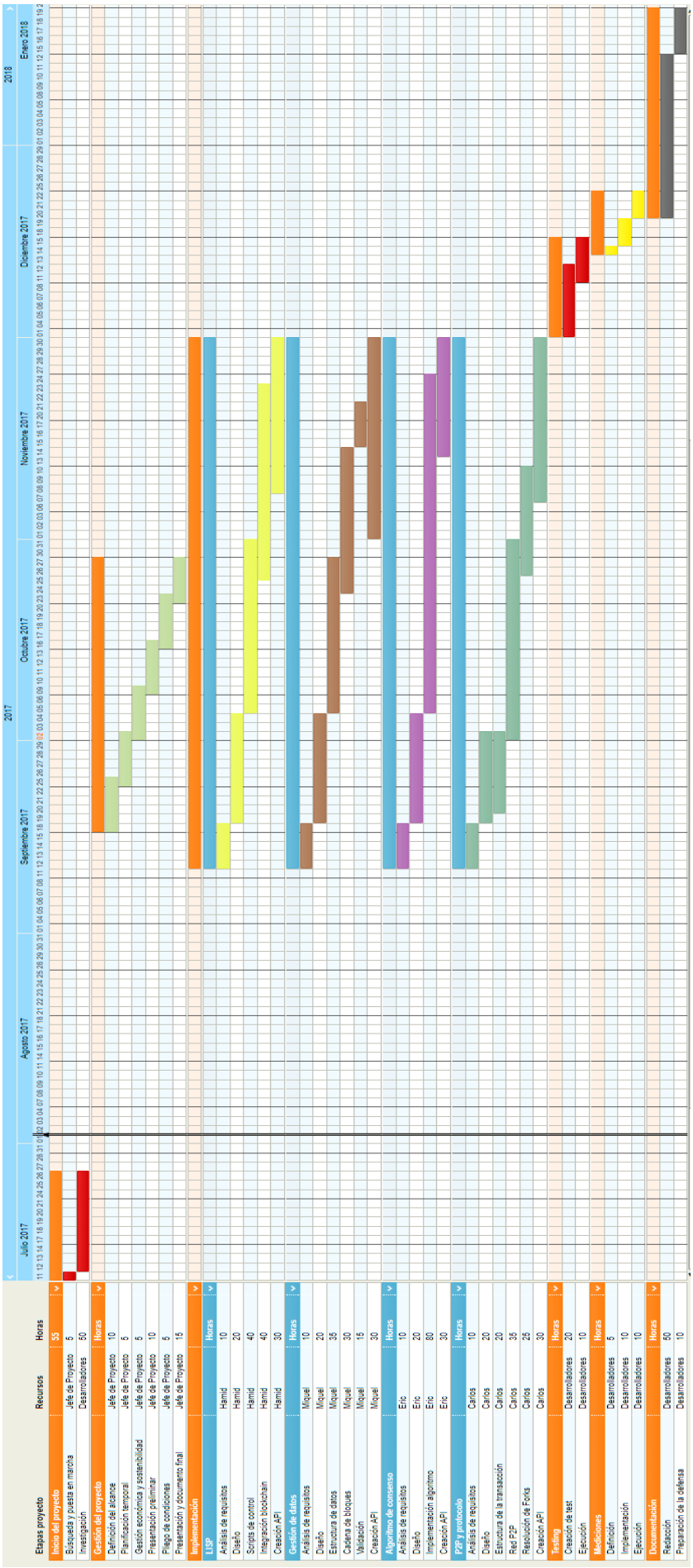

Figura 2.1: Diagrama de Gantt del proyecto

3. Gestión Económica y Sostenibilidad

3.1. Gestión Económica

Una vez planteado el problema y diseñado una solución, vamos a realizar un estudio económico y un estudio del impacto social y ambiental, para determinar la viabilidad del proyecto.

3.1.1. Consideraciones Iniciales

Para realizar este proyecto se utilizarán todos los elementos detallados en el apartado de recursos comentados en el apartado anterior. Vamos a estimar el coste del proyecto de los recursos humanos, hardware, software y costes indirectos.

Debido a que este proyecto se realiza dentro de la universidad con el material del que ya se dispone, no implica comprar material nuevo, lo que se traduce a que la mayor parte de los recursos ya están amortizados.

3.1.2. Presupuesto de recursos humanos

Debido a la naturaleza del proyecto, los recursos humanos somos estudiantes de la facultad que no disponemos de una beca, por lo tanto, no cobramos un salario (estimaremos el coste). Por otro lado, el jefe del proyecto sí que cobra un suelto de becario.

Vamos a analizar el coste de cada una de las partes del diagrama de Gantt: Inicio del proyecto, Gestión del proyecto, Implementación, Testing, Mediciones y Documentación.

Coste del inicio del proyecto: 1700€

Rol	Horas	Precio	Precio Total
Jefe de Proyecto	5	20€/h	100 €
Desarrollador 1	50	8€/h	400 €
Desarrollador 2	50	8€/h	400 €
Desarrollador 3	50	8€/h	400 €
Desarrollador 4	50	8€/h	400 €

Gestión del proyecto: 1000€

Rol	Horas	Precio	Precio Total
Jefe de Proyecto	50	20€/h	1000 €

Coste de la implementación: 4480€

Rol	Horas	Precio	Precio Total
Desarrollador 1	140	8€/h	1120 €
Desarrollador 2	140	8€/h	1120 €
Desarrollador 3	140	8€/h	1120 €
Desarrollador 4	140	8€/h	1120 €

Coste del testing: 960€

Rol	Horas	Precio	Precio Total
Desarrollador 1	30	8€/h	240 €
Desarrollador 2	30	8€/h	240 €
Desarrollador 3	30	8€/h	240 €
Desarrollador 4	30	8€/h	240 €

Coste de las mediciones: 720€

Rol	Horas	Precio	Precio Total
Ingeniero QA	5	20€/h	100€
Desarrollador 1	10	8€/h	80 €
Desarrollador 2	10	8€/h	80 €
Desarrollador 3	10	8€/h	80 €
Desarrollador 4	10	8€/h	80 €

Coste de la documentación: 1200€

Rol	Horas	Precio	Precio Total
Jefe de Proyecto	60	20€/h	1200€

El coste total de todas las partes de RR. HH. es de 10.060€.

3.1.3. Presupuesto de hardware

Para llevar a cabo la implementación del proyecto, necesitamos una serie de elementos hardware para desarrollar y documentar el proyecto.

Producto	Precio	Unidades	Vida útil
Ordenador	500€	5	5 años
Portátil	500€	5	5 años
Servidores AWS	20€	6	1 mes

Total 5120€

3.1.4. Presupuesto de software

En cuanto al software, se necesitarán las siguientes herramientas de software. Todas las utilizadas son gratuitas.

Producto	Precio	Vida útil
Libre Office	gratuito	...
Google Docs	gratuito	...
LaTex	gratuito	...
VS Code	gratuito	...
GitHub	gratuito	...
Python	gratuito	...
Sphinx	gratuito	...

Total 0€

3.1.5. Costes Indirectos

Dado que este proyecto se realiza en la universidad, el consumo energético, el coste de internet, el coste en papel, etc. están incluidos en los gastos de la universidad.

Producto	Precio	Unidades	Precio Total
Electricidad	0,07€kW/h	1500 kWh	105€
Internet	50€/mes	4 meses	200€

Total 305€

En el caso de los desarrolladores no se cubre el gasto del transporte que lo deben aportar a nivel personal.

3.1.6. Imprevistos

Avería en uno de los ordenadores: es poco probable, supondría substituir o reparar el equipo, pero no retrasaría el proyecto ya que disponemos de más equipos.

Retraso en el desarrollo: podría ocurrir que se retrase el proyecto debido a su complejidad, aunque es poco probable que se salga del plazo de los 4 meses. Supondría incrementar los gastos de RR. HH. y los costes indirectos, no supondría incrementos en los costes de hardware y software.

3.1.7. Presupuesto total

Concepto	Precio Estimado
RR. HH.	10.060€
Hardware	5120€
Software	0€
Indirectos	305€

Total 15.480€

3.2. Sostenibilidad y compromiso social

3.2.1. Área económica

El proyecto es muy viable económicamente ya que no requiere de comprar ningún tipo de hardware o software. Solo requiere de un ordenador por desarrollador que la universidad ya dispone de ellos, no necesita comprar ningún equipo nuevo. En cuando al software es totalmente gratuito.

En cuanto al tiempo de desarrollo es difícil de predecir si se podrían reducir los tiempos debido a que se trata de un trabajo de investigación, no disponemos de un modelo a seguir.

3.2.2. Área social

Este proyecto aspira a tener un gran impacto en el mundo de internet, siendo uno de los primeros en estudiar el uso de la nueva tecnología blockchain en los protocolos de enrutamiento. Pretendiendo así, impulsar el uso de esta nueva tecnología.

Dependiendo del éxito final del proyecto esta propuesto para ser entrar en la IETF (The Internet Engineering Task Force) y poder ser usado como estándar para el uso en la infraestructura de internet. Podría llegar a suponer que empresas como Cisco que colaboran con el proyecto sacasen una versión comercial del producto.

3.2.3. Área ambiental

Por lo que hace al área ambiental, este proyecto no tiene una gran intrusión, ya que se desarrolla un software que no requiere de ningún tipo de hardware especializado para ejecutarse, a su vez, tampoco requiere de un ordenador potente. Como se indica en la contextualización uno de los objetivos principales del proyecto era usar un modelo de blockchain que solventara los graves problemas ambientales, un gasto enorme de electricidad y recursos hardware, que traen consigo las primeras versiones de esta tecnología.

3.2.4. Puntuación sostenibilidad

	PPP	Vida Útil	Riesgos
Ambiental	Consumo del diseño	Huella Ecológica	Riesgos ambientales
	10/10	19/20	-1/-20
Económico	Factura	Plan de viabilidad	Riesgos económicos
	10/10	15/20	-5/-20
Social	Impacto Personal	Impacto Social	Riesgos Sociales
	7/10	19/20	0/20
Rango Sostenibilidad	27/30	53/60	-6/-60
	74/90		

Tabla 3.11: Matriz de sostenibilidad

4. Transacción

4.1. Requisitos

Una transacción debe indicar las IPs que se están emitiendo junto con la información que requiere el protocolo LISP si esta es necesaria. Estas deben estar firmadas por el emisor y indicar la llave pública del receptor.

Para la realización de este proyecto se ha elegido usar el lenguaje de programación Python. Por ello he elegido la librería de manipulación de IPs de Google *ipaddr* que me permite pasarlas a binario y viceversa.

4.2. Diseño

Todas las blockchains que hemos analizado trabajan con monedas virtuales, para hacerlo, simplemente indican la cantidad emitida. En nuestro caso debemos indicar que o cuales IP se intercambian ya que cada una de ellas tiene su número y no se pueden repetir, es decir, solo un usuario puede poseer una misma IP en un momento determinado.

Dado que IPv4 posibilita 2^{32} direcciones e IPv6 2^{128} es inviable indicar en una transacción todas las IP emitidas una a una. La mejor solución es usando rangos de direcciones, que es la forma habitual con la que se trabaja, bastará con dar una IP y una máscara. De esta forma si un usuario posee el rango 10.0.0.0/8, quiere decir que posee las IPs entre la 10.0.0.0 y la 10.255.255.255. Si se quiere hacer referencia a una única IP basta con usar la máscara 32 o 128 en el caso de IPv4 e IPv6 respectivamente. Para distinguir entre IPv4 e IPv6 hemos añadido un campo llamado Afi que toma valor 1 en caso de IPv4 y 2 en caso de IPv6.

Cuando un usuario de una criptomoneda envía dinero a otro, este pierde por completo el control de esa cantidad a partir de ese momento. En nuestro caso una empresa puede asignar a un cliente una dirección IP pero esta tiene que seguir siendo la propietaria. Para poder dotar a nuestra blockchain de esta propiedad, vamos a indicar con un campo en la transacción que el emisor esta prestando las IPs. Además se ha decidido que el emisor indicará con una fecha dentro de la transacción el día y hora en el cual recuperará todas las direcciones IP emitidas. Durante el tiempo que el receptor posee las IPs estas están bloqueadas.

Esta blockchain pretende substituir el sistema DDT de Lisp, por lo que debe almacenar toda la información que este contiene, en él se encuentran las direcciones de los MapServer y Locator. Estas direcciones no tienen ningún efecto en la blockchain, se han decidido guardar en su interior por la seguridad que ofrece. Esta información se guardará en forma de metadatos, un usuario enviará una transacción a si mismo con el rango de IPs al que se le asignarán los datos. Se pueden emitir tantas transacciones con metadatos como se desee, la última de todas, al contener

la información más actualizada es la válida. A un usuario al que le hayan prestado direcciones IP también puede emitir estas transacciones para asignar sus MapServer y Locator.

Finalmente distinguimos las transacciones en cuatro categorías:

- **Allocate**: El receptor es el nuevo propietario del rango de IPs.
- **Delegate**: El emisor presta durante un tiempo un rango de IPs al receptor.
- **MapServer**: Añadimos información de los MapServer a un rango de IPs
- **Locator**: Añadimos información de los Locator a un rango de IPs

En el grafo de la figura 4.1 se muestra cuando un usuario puede generar cada uno de los diferentes tipos de transacciones. Un usuario propietario de un rango de IPs puede emitirlo a otro usuario (Allocate), puede prestarlo (Delegate) y puede asignar metadatos (MapServer y Locator). Si se poseen unas IPs prestadas estas no se pueden ni emitir (Allocate) ni prestar (Delegate). Para identificar el tipo de transacción se les ha asignado los siguientes números: 0 → Allocate, 1 → Delegate, 2 → MapServer, 3 → Locator.

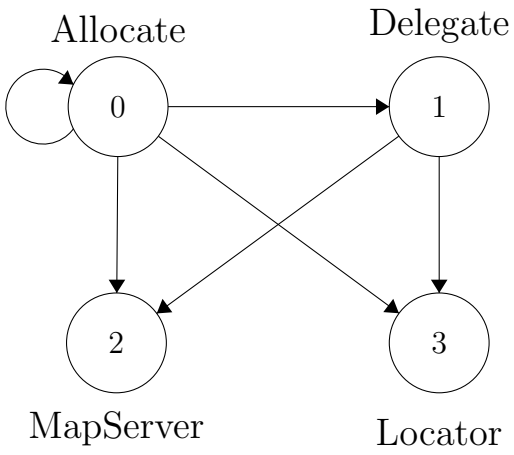

Figura 4.1: Grafo de transacciones

4.3. Implementación

En el inicio del proyecto se tenía en mente seguir la estructura de la criptomoneda Bitcoin, lo que trajo problemas a la hora de ver como implementábamos los scripts de entrada y salida que pretendíamos simplificar. Además la complejidad que suponía buscar y almacenar las transacciones en la base de datos junto con el tamaño de estas nos llevo a dejar de lado el modelo de Bitcoin.

Se decide centrarnos en el modelo de Ethereum, debido a los problemas presentados y por disponer de mejores herramientas con una implementación libre en Python. Esta cryptomoneda, ofrece una solución muy simple al problema de las transacciones.

En Ethereum si que existe el concepto de balance, por lo tanto, a la hora de enviar una transacción basta con indicar la cantidad (en nuestro caso las IPs) y el receptor. Toda la parte de

verificación y validación ya no se incluye en la transacción, esta parte ahora la deben realizar mis compañeros en la chain.

Los Smart Contrats no han sido implementados por no ser necesarios ya que estamos creando nuestra propia blockchain dedicada.

Ethereum cuenta con la librería RLP (Recursive Length Prefix) que permite serializar objetos en datos binarios y también decodificarlos. Creamos una lista con todos los campos de la transacción y RLP lo codifica. Utilizamos la misma estructura de Ethereum adaptada a nuestras necesidades y los mismos algoritmos de hash (keccak-256) y encriptación (curvas elípticas).

Los campos que contiene una transacción son los siguientes:

- **Nonce**: Número de transacción de esta cuenta.
- **Category**: Tipo de transacción (Allocate, Delegate, MapServer, Locator).
- **To**: Cuenta del usuario receptor.
- **Afi**: Distingue IPv4 de IPv6.
- **Value**: Rango de IPs emitido.
- **Metadata**: Campo opcional con la información de los MapServer y Locator.
- **Time**: Campo opcional con el tiempo Unix para indicar el fin de un Delegate.

En el campo de metadatos si se trata de una transacción de tipo MapServer incluye una lista de IPs con su Afi y una llave pública, esta no influye en la blockchain, para que el servidor pueda firmar los paquetes emitidos. En el caso de Locator incluye una lista de IPs con su Afi.

Una vez rellenados todos los campos de una transacción, se hashea y se firma. Como resultado obtenemos los valores V, R y S. Con el método de firmas usado no es necesario indicar el emisor, está implícito en la firma. Ofreciendo así transacciones lo más pequeñas posibles.

En la tabla 4.1 del siguiente apartado se muestra una transacción de ejemplo con la longitud en bytes que ocupa cada campo. En la tabla 4.2 se muestra su codificación en binario resultado de usar la librería RLP. Se ha remarcado en color cada campo para que se pueda identificar.

4.3.1. Ejemplo

Descripción	Ejemplo	Longitud
Nonce	5	1+
Category (0 - 3)	2	1
To (destination addr)	0x8faae7a1e781df169bf47623c0ecbc3920cabc55	20
Afi	1	1
Value	c098000010 (192.152.0.0/16)	5 o 17
Metadata (Opcional) - MapServer (Afi, IP, PubKey) - Locator (Afi, IP)	1 01010101 (1.1.1.1) 0x54dbb737eac5007103e729e9ab7ce64a6850a310 1 02020202 (2.2.2.2) 0x89b44e4d3c81ede05d0f5de8d1a68f754d73d997 2 20010db80a0b12f000000000000 (2001:db8:a0b:12f0::1) 0x3a1e02d9ea25349c47fe2e94f4265cd261c5c7ca	6+
Time	0	1-4
V	1c	1
R	10466f7104039530d426d75ac53335929a467b985e2 0cc89f0d650791b787ed2	32
S	6a53f8f9c22d232849428d302d96dab065c8eab7fe0 47c5c9d655b8e29771cd2	32

Tabla 4.1: Parámetros de una transacción tipo MapServer

f8c1 05 02 94 8faae7a1e781df169bf47623c0ecbc3920cabc55 01 85 c098000010 f85d
01 84 01010101 94 54dbb737eac5007103e729e9ab7ce64a6850a31001 84 02020202 94
89b44e4d3c81ede05d0f5de8d1a68f754d73d99702 90 20010db80a0b12f000000000000
00001 94 3a1e02d9ea25349c47fe2e94f4265cd261c5c7ca 80 1b a0 10466f7104039530d4
26d75ac53335929a467b985e20cc89f0d650791b787ed2 a0 6a53f8f9c22d232849428d30
2d96dab065c8eab7fe047c5c9d655b8e29771cd2

Tabla 4.2: Transacción codificada con RLP

Nota: Como es una transacción de tipo MapServer el emisor y receptor son el mismo usuario.

5. Peer-to-Peer

5.1. Requisitos

Se requiere de una red de comunicación que conecte las 6 maquinas virtuales en la nube (Amazon Web Services) junto con la máquina local. Por esta red se enviarán tanto los bloques como las transacciones que se generen, a su vez, esta misma debe realizar el bootstrap cuando se arranque el sistema y responder a las peticiones del resto de nodos.

El bootstrap consiste en pedir a los peers los bloques que le faltan a la cadena y todas las transacciones que se han emitido pero aun no han entrado en la cadena, a lo que llamaremos el pozo de transacciones (transaction pool).

De la misma forma que las transacciones esta también debe ser programado en Python. Por ello se tuvieron en cuenta una serie de librerías para crear redes.

- pyP2P: Librería especialmente diseñada para redes P2P, se tuvo que descartar por no estar suficiente madura.
- Asyncio: Es la librería que trae python 3 por defecto para tratar funciones asíncronas, se tuvo que descartar porque finalmente tuvimos que usar python 2 al tener problemas con librerías de los módulos de mis compañeros.
- Twisted: Librería dirigida a programación por eventos, esta fue finalmente elegida por su facilidad en implementación.

Debido a la complejidad que supone implementar un algoritmo de consenso para una blockchain de tipo PoS y la difícil gestión de las bifurcaciones (forks) en la cadena, se ha decidido eliminar la posibilidad de que ocurran. Así tenemos una mejor modularidad del proyecto y más posibilidades de éxito.

5.2. Diseño

Debido a que no se conectarán muchos dispositivos en las pruebas que queremos realizar, hemos podido simplificar el diseño de la red para maximizar las probabilidades de éxito. La red tiene un nodo hardcodeado para conectarse al inicio y descubrir el resto de peers, en este caso elegimos el ordenador de la facultad como nodo inicial.

Cuando un nodo se conecta a la red descarga del nodo inicial toda la lista de peers y se conecta a ellos. Después este empieza el bootstrap, pregunta cuantos bloques tiene la cadena y descarga todos aquellos que le falten, luego descarga todo el pozo de transacciones.

Una vez realizada la etapa de bootstrap, cuando un nodo genera una transacción nueva o un bloque este lo envía por la red al resto. A su vez, cada 5 minutos un nodo hace un ping a todas sus conexiones para ver que todos los nodos sigan vivos, sino cierra la conexión.

5.3. Implementación

Para que la red pueda trabajar paralelamente con el resto de módulos, esta se ejecuta en un hilo separado. Dentro del bucle principal del programa se le pregunta si tiene alguna petición de ser atendida o se le notifican los datos que tiene que enviar.

Todos los mensajes se envían serializados en un JSON, indicando el tipo de mensaje con el campo *msgtype* y los datos.

Para poder realizar todas estas tareas utilizo las siguientes estructuras de datos. Además en forma de contador se tiene constancia del último bloque generado.

- La lista de Peers: se guarda en un diccionario.
- Bloques recibidos: se guardan en un diccionario, la llave es el número de bloque.
- Transacciones recibidas: se guardan en un conjunto.
- Lista de peticiones: se guardan en un diccionario, la llave es el identificador de un peer.

Cuando el hilo principal del programa pregunta si hay nuevas transacciones o bloques, este le responde y vacía el contenido. En caso contrario si otro modulo debe responder una petición de un peer este le notifica la petición.

La librería Twisted gestiona todas las conexiones de forma concurrente y cuenta con una serie eventos que se disparan cuando se reciben datos o se conecta o desconecta algún nodo. Cuenta con una clase global a la que pueden acceder todos los peers llamada *factory* donde se almacena la información (se guardan todas las estructuras de datos mencionadas) y por cada peer se ejecuta una instancia independiente del protocolo. Twisted ofrece los siguientes eventos.

- connectionMade: Se ejecuta cuando se estableces una conexión. Añadimos un nuevo peer a la lista.
- connectionLost: Se ejecuta cuando se pierde una conexión. Eliminamos el peer de la lista.
- dataReceived: Se ejecuta cuando recibimos datos de un peer. Decodificamos el JSON y ejecutamos la función pertinente.
- write: Envía los datos a otro nodo. Lo usamos para transmitir la información.
- LoopingCall: Ejecuta una función cada X tiempo. Se usa para hacer el Ping y consultar cuando ha acabado el bootstrap.
- deferLater: Ejecuta una función después de X tiempo. Se usa para comprobar en unos segundos si el peer nos ha enviado los datos.

5.4. Rendimiento

He realizado una prueba de rendimiento con 3 de nuestras máquinas en la nube. Para ello he elegido una en cada continente buscando así el peor escenario posible. La figura 5.1 muestra un mapamundi con la localización de las máquinas.

1. Frankfurt
2. North California
3. Sydney

Figura 5.1: Localización de las máquinas virtuales

A continuación la tabla 5.1 muestra el tiempo de respuesta (ping) entre las tres máquinas virtuales. Los resultados son bastante buenos, en el peor de los casos estamos por debajo de 300ms (Frankfurt-Sydney). La máquina de California es la que mejor conexión tiene tan solo 147ms con ambos continentes.

	Frankfurt	California	Sydney
Frankfurt	-	147ms	291ms
California	147ms	-	147ms
Sydney	291ms	147ms	-

Tabla 5.1: Tiempo de respuesta entre máquinas

5.4.1. Bloques

Para calcular el tiempo que tarda un bloque en ser transmitido, he generado 100 bloques del tamaño máximo permitido, en nuestro caso es 1MiB, y los he enviado entre todas las parejas de máquinas obteniendo así cien muestras de tiempo. La figura 5.2 muestra un boxplot de los resultados y la tabla 5.2 sus estadísticas.

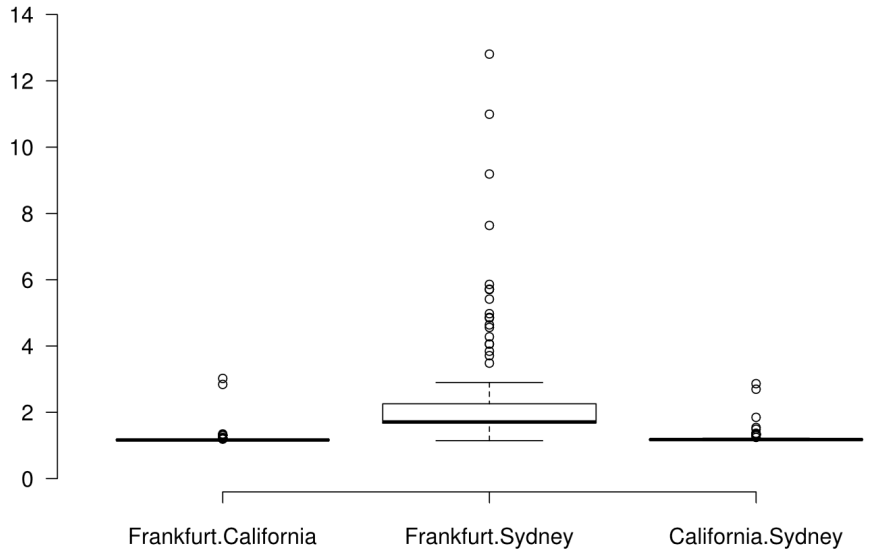

Figura 5.2: Muestras del tiempo de transmisión de un bloque

	Frankfurt-California	Frankfurt-Sydney	California-Sydney
Upper whisker	1.17s	2.90s	1.22s
3rd quartile	1.17s	2.26s	1.19s
Median	1.16s	1.71s	1.18s
1st quartile	1.16s	1.68s	1.17s
Lower whisker	1.16s	1.15s	1.15s
Nr. of data points	100.00	100.00	100.00

Tabla 5.2: Estadísticas del tiempo de bloque

Observamos que un bloque te tamaño máximo tarda entre 1.1 y 1.7 segundos en enviarse de media. Los bloques se generan cada minuto y en el peor de los casos un bloque ha tardado 13 segundos en llegar, seguramente por una congestión, estamos dentro de un intervalo de tiempo que podemos asumir. Si se quisiera disminuir el tiempo de creación de un bloque este intervalo no debería ser menor a 15 segundos. Podemos ver que la máquina de California vuelve a ser la que mejores resultados muestra.

5.4.2. Pozo de transacciones

Para medir el tiempo que tarda en enviarse el pozo de transacciones, he generado uno con 100mil transacciones y lo he enviado entre todas las parejas de máquinas 10 veces. He elegido

transacciones de categoría 2 ya que estas son las más grandes debido a que contienen metadatos. La tabla 5.3 muestra los resultados obtenidos.

Frankfurt-California	Frankfurt-Sydney	California-Sydney
7.39s	11.53s	9.81s

Tabla 5.3: Tiempo, transmisión del pozo de transacciones

Observamos unos resultados similares a los apartados anteriores, la máquina de California es la que rinde mejor. El pozo de transacciones solo se envía una vez al inicio por lo que esperar 12 segundos no es un problema, además lo normal es que sea mucho menor al usado en este test.

6. Conclusión

Hemos logrado introducir toda la información requerida por LISP en una transacción, MapServers y Locatos, también hemos ofreciendo la posibilidad de prestar direcciones a otros usuarios. A su vez, las transacciones se verifican correctamente y se incluyen perfectamente en la chain.

La red de comunicación funciona correctamente permitiendo el envío de información a todos los peers alrededor del mundo. Hemos analizado los problemas que supone usar una blockchain tipo Proof of Stake y sus posibles soluciones.

A nivel local hemos obtenido resultados satisfactorios a falta de probarla a nivel global por falta de tiempo. Han surgido bastantes problemas al querer implementar algo nuevo que con esfuerzo hemos ido solucionando.

Se ha cumplido nuestro objetivo de demostrar que era posible almacenar la información de enrutamiento de Internet en una blockchain. Una vez tengamos los resultados a nivel global se escribirá el paper y esperamos que se posicione como una posible mejora al Internet actual.

7. Trabajo futuro

7.1. Peer-to-peer

Como la cantidad de máquinas conectadas es muy pequeña, no ha requerido implementar una tabla de hash distribuida (DHT) como hace Ethereum con Kademlia. Para poder probar la blockchain con una cantidad mucho más grande de nodos, miles de ellos, se requeriría de la programación de una de ellas. También haría la red mucho más robusta.

La red P2P como he comentado en la contextualización puede recibir ataques que tenemos que minimizar, sobretodo si usamos un algoritmo tipo PoS. Sería interesante implementar algún método de puntuación de nodos como el que utiliza la criptomoneda Nem para detectar nodos maliciosos. De este modo podemos proteges a los nodos con más participación de ser excluidos de la red.

Por problemas de complejidad y modularidad se elimino la posibilidad de bifurcaciones (forks) en la cadena. Una futura mejora sería la capacidad de soportarlo. Durante estos meses si que analizamos como hacerlo pero era demasiado difícil.

Por otro lado se espera que se mejore el módulo del algoritmo de consenso lo que supondrá el envió de muchos mensajes entre peers. Se tendrá que incluir nueva lógica para tratarlos.

7.2. Pruebas

A fecha de hoy, No hemos podido ejecutar todos los módulos de la blockchain en todas las máquinas virtuales a nivel mundial por problemas de sincronización. En cambio, si que hemos podido ejecutarlas en los servidores de la facultad.

Se generaron varias máquinas virtuales en el servidor del DAC y se inserto en la cadena la información de las tablas BGP como prueba. La red de pruebas de LISP consultaba correctamente los metadatos de las transacciones y respondía a los routers.

Hay que conseguir que funcione finalmente la blockchain a nivel mundial para poder obtener datos de la ejecución y estudiar la viabilidad de esta almacenando la información de enrutamiento de Internet. Necesitamos obtener mediciones de la latencia total del sistema.

Finalmente el director y codirector redactarán un paper con el estudio realizado y la viabilidad del de las blockchains en los protocolos de enrutamiento.

Bibliografía

[1] Jordi Paillisse, Alberto Rodriguez-Natal, Vina Ermagan, Fabio Maino, and Albert Cabellos-Aparicio. An analysis of the applicability of blockchain to secure IP addresses allocation, delegation and bindings. https://tools.ietf.org/html/draftpaillisse-sidrops-blockchain-00, 2017.

[2] Albert Cabellos and Damien Saucez. An Architectural Introduction to the Locator/ID Separation Protocol (LISP). https://tools.ietf.org/html/draft-ietf-lispintroduction-13.

[3] Alberto Rodriguez-Natal et al. Programmable Overlays via OpenOverlayRouter. https://tools.ietf.org/html/draft-ietf-lispintroduction-13, 2017.

[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.

[5] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. https://arxiv.org/pdf/1605.07524.pdf, 2016.

[6] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. http://www.cryptopapers.net/papers/ethereum-yellowpaper.pdf, 2014.

[7] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf, 2002.

[8] Nem: Technical Reference. https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf, 2015.