Algoritmo de Estimación del Nivel de Confianza en Servicios 6G

Algorithm for Estimating the Level of Trust in 6G Services

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA DEL SOFTWARE
CURSO 2023–2024

Jesús García Peña
José Otegui Marín

Directores

Luis Javier García Villalba
Jesús Ángel Alonso López

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Mayo de 2024

Agradecimientos

Queremos expresar nuestro agradecimiento a nuestros directores del Trabajo de Fin de Grado, Luis Javier García Villalba y Jesús Ángel Alonso López, por su respaldo y disposición brindada durante todo el desarrollo del proyecto. Adicionalemnte, queremos expresar nuestros nuestros agradecimientos al Grupo GASS (Grupo de Análisis, Seguridad y Sistemas) y en particular hacer mención a Elmira Saeedi Taleghani, Ronald Iván Maldonado Valencia por las ayudas y el conocimiento ofrecido durante la implementación del código y a Ana Lucila Sandoval Orozco por su apoyo durante el desarrollo de la memoria del trabajo.

Finalmente, deseamos mostrar nuestra profunda gratitud a nuestros seres queridos, familiares y amigos cercanos, por confiar en nosotros y brindarnos su apoyo incondicional a lo largo de nuestra trayectoria como estudiantes del Grado de Ingeniería del Software.

Índice General

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

1G	*First Generation*
5G	*Fifth Generation*
6G	*Sixth Generation*
ABI	*Application Binary Interface*
API	*Application Programming Interface*
API REST	Representational State Transfer Application Programming Interface
BTA	*Blockchain Transfer Arquitecture*
CLI	*Command Line Interface*
CTI	*Cyber Threat Inteligence*
DAG	*Directed acyclic graph*
dApp	*Decentralized Application*
dApps	*Decentralized Applications*
DLT	*Distributed ledger tecnology*
E2E	End-to-end
ETH	*Ether*

EVM	*Ethereum Virtual Machine*
FCL	*Fuzzy Control Language*
GASS	*Grupo de Análisis, Seguridad y Sistemas*
HTTP	Hypertext Transfer Protocol
IA	*Inteligencia Artificial*
IDE	*Integrated Development Environment*
IoE	*Internet of Everything*
IoT	*Internet of Things*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
JSON-RPC	*Remote Procedure Call Protocol encoded in JSON*
LoT	*Level of Trust*
MEC	*Multi-access Edge Computing*
MNOs	*Mobile Network Operators*
NFV	*Network Function Virtualization*
P2P	*Peer-to-Peer*
PoA	*Proof of Authority*

PoS	*Proof of Stake*
PoT	*Proof of Transit*
PoW	*Proof of Work*
PSO	*Particle Swarm Optimization*
QoE	*Quality of Experience*
QoS	*Quality of Service*
SDN	*Software-Defined Networks*
SLA	*Service Level Agreement*
TCP	*Transmission Control Protocol*
TTP	Trusted Third Party
WSNs	*Wireless Sensor Networks*

Abstract

As *Fifth Generation* (5G) technology expands globally, challenges related to information security and network stability have been identified, especially in environments where data integrity is critical. In response to these challenges, the proposed research focuses on exploring and improving 6G networks to overcome the limitations of 5G networks in terms of trust and security.

This project is based on the premise of estimating the trust level of an End-to-end (E2E) service in future 6G networks, with the aim of improving network trust. To achieve this, a two-part approach is employed. The first part involves the study and implementation of an algorithm with fuzzy logic that generates a representative value of the trust level. This fuzzy logic model allows for a more dynamic and adaptable evaluation of network trust, based on service metrics and real-time conditions. The second part focuses on the research and implementation of smart contracts on *Ethereum*, using them as a persistent and decentralized storage system for the algorithm's results. The integration of blockchain provides a robust solution for managing the results, leveraging its capability to offer complete traceability and resistance to tampering.

The implementation of this trust estimation software has been carried out through task management that simulates both critical events and periodic evaluations of network services. The proposed algorithm collects the 6G service metrics through the tasks, then evaluates them to generate a confidence level using fuzzy logic. The result of this estimation is stored in both a smart contract and a NoSQL database management system. The results of the algorithm indicate a feasible approach to the service confidence level and could represent an improvement in the reliability and security of communications, marking an advance over the current capabilities of 5G networks.

keywords: 6G, blockchain, Ethereum, fuzzy logic, network security, network trust, smart contracts, trust evaluation.

Resumen

A medida que la tecnología 5G se expande globalmente, se han identificado desafíos relacionados con la seguridad de la información y la estabilidad de la red, especialmente en entornos donde la integridad de los datos es crítica. En respuesta a estos desafíos, la investigación propuesta se centra en explorar y mejorar las redes 6G para superar las limitaciones de las tecnologías 5G en términos de confiabilidad y seguridad.

Este trabajo parte de la premisa de estimar el nivel de confianza de un servicio E2E de las futuras redes 6G, con el propósito de mejorar la confianza de la red. Para lograrlo, se emplea un enfoque de dos partes. La primera consiste en el estudio e implementación de un algoritmo con lógica difusa que genere un valor representativo del nivel de confianza. Este modelo de lógica difusa permite una evaluación más dinámica y adaptable de la confianza en la red, basado en métricas del servicio y condiciones en tiempo real. La segunda parte se centra en la investigación e implementación de contratos inteligentes en *Ethereum*, empleando estos como sistema de almacenamiento persistente y descentralizado para los resultados del algoritmo. La integración de la *blockchain* proporciona una solución robusta para la gestión de los resultados, aprovechando su capacidad de ofrecer trazabilidad completa y resistencia a manipulaciones.

La implementación de este software de estimación de confianza se ha realizado mediante una gestión por tareas que simula tanto eventos críticos como evaluaciones periódicas de los servicios de la red. El algoritmo propuesto recopila las métricas del servicio 6G a través de las tareas y las evalúa para generar un nivel de confianza utilizando lógica difusa. El resultado de esta estimación se almacena tanto en un contrato inteligente como en una base de datos NoSQL. Los resultados del algoritmo indican una aproximación factible al nivel de confianza del servicio que podría suponer una mejora en la confiabilidad y seguridad de las comunicaciones, marcando un avance sobre las capacidades actuales de la red 5G.

Palabras clave: 6G, blockchain, confianza en redes, contratos inteligentes, Ethereum, evaluación de confianza, lógica difusa, seguridad en redes.

Capítulo 1

Introducción

1.1. Motivación

La evolución constante de las tecnologías de comunicación inalámbrica ha sido una pieza clave en el desarrollo y transformación de nuestras sociedades modernas. Desde la introducción de las primeras redes móviles hasta la actualidad, cada generación de tecnología de comunicaciones ha traído consigo avances significativos en términos de velocidad, confiabilidad y capacidad de conectividad. Sin embargo, a pesar de los logros alcanzados por las redes 5G, existe la necesidad de avanzar hacia un nuevo estándar que aborde las limitaciones existentes y se adapte a las demandas emergentes de un mundo cada vez más conectado y digitalizado.

Una limitación importante de las redes 5G, en relación con el nivel de confianza, es su capacidad para garantizar la seguridad y la integridad de las comunicaciones en entornos altamente dinámicos y heterogéneos. Si bien las redes 5G han introducido avances significativos en términos de velocidad y capacidad, su enfoque en la eficiencia espectral y la latencia ha dejado lagunas en lo que respecta a la confianza en la conectividad. La creciente interconexión de dispositivos y sistemas críticos en áreas como el transporte autónomo, la atención médica remota y la infraestructura inteligente plantea nuevos desafíos en términos de seguridad y privacidad que las redes 5G no pueden abordar de manera óptima.

Por otro lado, las redes 6G se perfilan como un paradigma de comunicaciones inalámbricas que va más allá de simplemente mejorar la velocidad y la capacidad de datos. Se espera que esta sexta generación aborde estas limitaciones mediante el diseño de arquitecturas y protocolos que prioricen la confiabilidad, la seguridad y la privacidad en todos los niveles de la conectividad. Al integrar tecnologías emergentes como la inteligencia artificial, la computación cuántica y las DLT, las redes 6G tienen el potencial de establecer un nuevo estándar en las comunicaciones inalámbricas.

Esta sexta generación supondrá una gran evolución en las tecnologías de conectividad inalámbrica, siendo una parte fundamental para el futuro *Internet of Everything* (IoE). La nueva generación debe permitir la recopilación e intercambio de datos de una manera rápida, efectiva y confiable entre una amplia variedad de servicios, aplicaciones, máquinas y usuarios para formar un ecosistema digital en tiempo real. El desarrollo de esta sexta generación presenta un gran reto con una serie objetivos que se deben cumplir, como velocidades de datos extremadamente altas, baja latencia, mayor eficiencia energética, mayor capacidad de transmisión y una alta seguridad y privacidad entre otros.

En este contexto, la confiabilidad de los elementos de la red es un aspecto crítico a tener en cuenta en el diseño de las redes 6G. Cada uno de los elementos de la red debe cumplir con las expectativas que esta nueva era tecnológica demanda, asegurando no solo la conectividad constante y fluida, sino también la integridad de los datos, la resiliencia ante posibles fallas y la capacidad de adaptarse dinámicamente a las cambiantes condiciones del entorno de red.

La creciente sofisticación y número de ataques de ciberseguridad presenta serias dificultades a la hora de proteger la integridad de los elementos de la red. Lo que supone una rigurosa implementación de medidas de seguridad proactivas y robustas. Este aumento de la medidas de seguridad junto con la recopilación y procesamiento de datos por parte de los elementos de la red, pueden plantear un riesgo para la privacidad de los usuarios, quienes cada vez son más conscientes del tratamiento de sus datos y esperan medidas eficaces para garantizar su protección y anonimato.

La motivación detrás de este trabajo de investigación radica en la necesidad de tener una red confiable y robusta. Este enfoque lleva al diseño e implementación de un algoritmo que pueda estimar el nivel de confianza de los servicios de la red. Esta herramienta podría resolver la problemática de confiabilidad, estudiando y teniendo en cuenta diversos factores.

1.2. Contexto

El presente trabajo se ha desarrollado dentro del contexto de los proyectos ATESTA5G (TSI-063000-2021-0049) y TRAZA5G (TSI-063000-2021-0050') financiados por el Programa UNICO-5G I+D del Ministerio de AsuntosEconómicos y Transformación Digital y de la Unión Europea – Next Generation EU. Así mismo, ha contribuido al proyecto PRIVATEER, financiado por el programa de innovación Horizon Europe (grant agreement No.10109611).

1.3. Objeto de la Investigación

Las tecnologías de comunicación inalámbrica de sexta generación deberán permitir la conexión y el intercambio de información entre diversos elementos de la red en el futuro IoE. Estas tecnologías están destinadas a establecer un nuevo estándar en el uso de las tecnologías de comunicación y en las posibles aplicaciones derivadas de ellas. Considerando esta perspectiva, para mantener una óptima interacción en la red, es fundamental garantizar que sea segura y confiable.

Las conexiones y los servicios que forman parte de la red gestionan una gran cantidad de información sobre el funcionamiento de la conexión y otros aspectos relevantes para la confianza de la red, como la atestación de paquetes o la latencia de conexión. Llevando a cabo un estudio y análisis de estos valores, junto con otros relevantes, es posible hacer una estimación sobre cuánto de confiable es un servicio dentro de la red.

Este trabajo plantea un algoritmo *Level of Trust* (LoT) que, mediante la evaluación de las métricas necesarias, estime el nivel de confianza de los servicios de una red 6G. Junto con esto se propone también una solución para que la información resultante de la evaluación de confianza, sea almacenada dentro de una DLT, empleando así

la descentralización como primera medida de seguridad contra ataques cibernéticos y aumentando la confianza e inmutabilidad de los valores almacenados.

1.4. Plan de Trabajo

El desarrollo de este trabajo, como se puede observar en la Figura 1.1, se estructura en cinco fases:

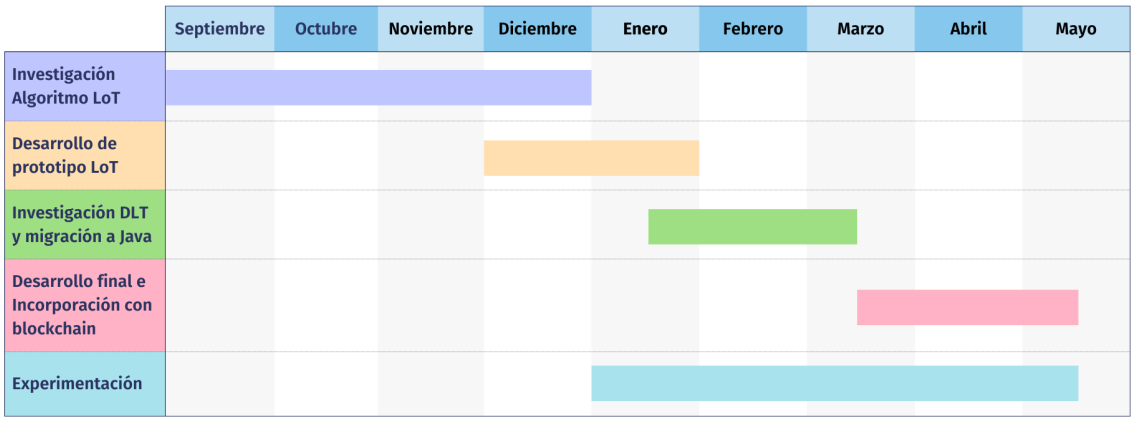

Figura 1.1: Digrama de Gantt para el plan de trabajo.

1. **Investigación del Algoritmo LoT**:

   Durante los primeros cuatro meses en los que se desarrolló esta fase, se concertaron reuniones semanales con el propósito de iniciar el estudio en la materia. En estos encuentros, primeramente se explicó el contexto de la investigación, se abordaron conceptos fundamentales sobre redes y se establecieron los objetivos a alcanzar e ideas principales en consideración en aquel momento. Además también se explicaron algunas herramientas pertinentes para la investigación y estudio de diversos artículos que estuvieran relacionados con el tema que concierne al trabajo, incluso sugiriendo ciertos documentos científicos como punto de partida.

   Durante los primeros meses del proyecto se llevó a cabo una meticulosa recopilación de información y estudio para poder ampliar los conocimientos en el área en la que se desarrolla el trabajo. Paralelamente, se continuaba con reuniones periódicas para efectuar un seguimiento del progreso del trabajo y aclarar ciertas dudas que surgían a medida que se avanzaba en la investigación.

   En este punto se realizó una investigación inicial acerca de las redes 6G y todos los aspectos relacionados con el diseño de una red confiable. Posteriormente se estudió el concepto principal que se tenía en esta etapa del proyecto, definiendo los requisitos del algoritmo, los aspectos a considerar y los resultados esperados. Adicionalmente se realizó un estudio de diferentes modelos de inteligencia artificial que se podrían usar en contextos como los de esta investigación.

   Tras comprender y evaluar todas las opciones disponibles, se tomaron diversas decisiones, en colaboración con el equipo, respecto a la implementación del algoritmo LoT. Esto incluyó la selección de los lenguajes de programación y bibliotecas adecuadas, la identificación de los parámetros de red a evaluar y la elección de la lógica difusa como enfoque para abordar la problemática planteada.

2. **Desarrollo del prototipo del Algoritmo LoT**:

   Después de una exhaustiva investigación en los fundamentos del área de estudio del trabajo, se procedió a iniciar la fase de desarrollo. En este período, que abarcó aproximadamente dos meses, se compaginó parte de la investigación con la implementación del algoritmo, dado que conforme el proyecto tomaba forma, surgían cuestiones en torno a su implementación. A lo largo de esta fase y durante el transcurso del resto del proyecto, se siguió manteniendo la dinámica de reuniones semanales para llevar un control sobre el avance del mismo.

   Inicialmente, se comenzó a desarrollar un prototipo del algoritmo utilizando el lenguaje de programación *Python*. Para poder comenzar con la implementación se hizo una investigación a fondo sobre lógica difusa para comprender su funcionamiento, puesto que es un pilar fundamental del algoritmo propuesto. Asimismo, para la implementación de dicho prototipo se llevó a cabo un análisis de diferentes bibliotecas de programación que fueron empleadas, como por ejemplo *SkFuzzy* para lógica difusa o *connectMongo* para el almacenamiento de datos con *MongoDB*.

3. **Investigación DLT y migración a *Java***:

   Esta etapa del proyecto, que tuvo una duración aproximada dos meses, se compaginó con la anterior fase una vez implementado gran parte del prototipo en *Python*. En esta fase se consideró la viabilidad de emplear una DLT como una alternativa de almacenamiento e incluso de desplegar completamente el código del algoritmo en un contrato inteligente. Por lo tanto, se emprendió una investigación preliminar acerca de las DLT y su funcionamiento, investigando artículos y documentación de diferentes arquitecturas DLT y su aplicación en ámbitos como los de este trabajo. Para poder continuar con el desarrollo del algoritmo, puesto que se decidió usar la plataforma *Ethereum* y el cliente de código abierto *Hyperledger Besu* compatible con *Java*, se decidió hacer una migración del código de *Python* a *Java*. Para esta implementación en *Java* se efectuó una investigación acerca de las bibliotecas que podrían resultar útiles para la implementación de la lógica difusa en *Java*. Posteriormente, con los conocimientos adquiridos se realizó la migración de la aplicación junto con diversas implementaciones de mejoras al mismo.

   Con el código implementado en *Java*, se realizaron varias pruebas para comprender el funcionamiento del cliente de *Ethereum*, procediendo a la configuración de una red privada de prueba, la cual se destinó para la fase posterior de desarrollo.

   En relación con *Ethereum* se investigó acerca de la *Ethereum Virtual Machine* (EVM). Esta investigación también abarcó un minucioso estudio sobre el lenguaje *Solidity*, uno de los varios lenguajes compatibles con la EVM y se realizaron diversas pruebas de codificación de contratos inteligentes en *Solidity* para comprender su funcionamiento.

4. **Desarrollo final del algoritmo e incorporación con *blockchain***:

   En esta fase, que abarcó aproximadamente dos meses, después de varias reuniones con el equipo, se determinó finalmente la implementación del algoritmo y a su vez un contrato inteligente que pudiera almacenar los resultados del mismo.

   Para esta tarea se investigaron varias librerías en *Java* que pudieran servir de nexo entre la *blockchain* de *Ethereum* y la aplicación *Java*. Con el conocimiento

adquirido, se procedió al desarrollo de un contrato inteligente que pudiera almacenar los resultados, así como de facilitar su posterior consulta.

Esta etapa del desarrollo culminó exitosamente con la implementación del código correspondiente al algoritmo LoT en *Java* y su integración con el contrato inteligente desplegado en la *blockchain*.

5. **Experimentación**:

   Esta sección del trabajo se estuvo desarrollando desde la segunda fase del plan de proyecto. En un principio, se procedió a evaluar la viabilidad de la aplicación de la lógica difusa en un proyecto de tal envergadura, realizando diversos ajustes en el código con el fin de obtener diferentes visualizaciones gráficas de los resultados. A través del primer prototipo del algoritmo LoT desarrollado en *Python*, se comenzaron a obtener resultados preliminares que permitieron identificar mejoras necesarias en el código. Más adelante, con la implementación del algoritmo en *Java*, se llevaron a cabo comparaciones y análisis de los resultados, contrastándolos con datos de entrada.

   Adicionalmente en la fase 3 y 4, se investigaron los tiempos de reacción, coste y otras variables relevantes asociadas a la tecnología *blockchain* y los contratos inteligentes. Esta etapa de la experimentación condujo a modificaciones en el código y a la revisión de decisiones durante el desarrollo motivadas por las observaciones de los resultados.

1.5. Estructura del Trabajo

El trabajo restante se encuentra estructurado en ocho capítulos, según se detalla a continuación:

El capítulo 2 se dedica a la introducción de varios conceptos fundamentales para una comprensión adecuada acerca del ámbito del proyecto. En este apartado se describe el marco teórico del trabajo incluyendo la descripción de redes 6G y el concepto de evaluación de confianza. Además se realiza una explicación de la lógica difusa, la cual constituye un pilar fundamental del algoritmo. En este capítulo también se aborda la descripción de diversas tecnologías empleadas en la implementación del algoritmo. Dentro de este marco teórico, se proporcionará una explicación técnica sobre las *Distributed Ledger Tecnologies* y la *blockchain*, además de la descripción de una implementación de estas, en concreto *Ethereum*. Esto incluye un análisis del funcionamiento y uso de los contratos inteligentes, explicando su comportamiento dentro del contexto de la plataforma *Ethereum*.

En el Capítulo 3 se exponen una serie de investigaciones relacionadas en diferentes aspectos con la evaluación de la confianza de nodos y servicios en las redes 6G, además de explorar el uso de la DLT y los contratos inteligentes, para establecer su utilidad con respecto a las futuras redes 6G y el objetivo principal de esta investigación.

En el Capítulo 4 se realiza la explicación de toda la investigación, diseño e implementación del algoritmo de estimación del nivel de confianza. En este capítulo se describen las contribuciones realizadas al desarrollo del software que compone el algoritmo LoT.

El Capítulo 5 expone los resultados derivados de los diversos experimentos llevados a cabo para evaluar la confianza de un servicio en la red y la implementación del contrato inteligente. En este apartado se explica la consistencia de los resultados obtenidos

incluyendo los objetivos de esta experimentación. En esta sección se examina la viabilidad y eficacia del algoritmo en posibles escenarios reales.

El Capítulo 6 presenta las conclusiones principales de este trabajo, abordando la utilidad práctica del algoritmo desarrollado y las tecnologías empleadas. Adicionalmente se abordan las posibles investigaciones futuras, enfocadas en ampliar y mejorar el alcance del algoritmo, ofreciendo una visión del avance que puede suponer la investigación en este campo.

El Capítulo 7 describe las contribuciones de cada alumno al proyecto. En esta parte se describe el trabajo de estudio e investigación del alumnado así como las implementaciones que se han llevado a cabo en este trabajo.

Los Capítulos 8 y 9 corresponden con la traducción al inglés de la introducción y las conclusiones de este trabajo.

Capítulo 2

Marco Teórico

Este capítulo se ha dividido en tres secciones que se consideran esenciales para una comprensión integral del área del trabajo. Por un lado se explica el contexto de las redes 6G, la necesidad de evolución a estas tecnologías y aspectos relevantes de confianza y seguridad. Por otro lado, se proporciona una explicación sobre la lógica difusa, un método de inteligencia artificial que se ha empleado en el algoritmo. Finalmente, se presenta un análisis de las DLT y en concreto la plataforma *Ethereum*, como implementación de esta tecnología, exponiendo en este contexto una explicación detallada de los contratos inteligentes.

2.1. Redes 6G y Seguridad

Para comprender el contexto del algoritmo propuesto, es fundamental destacar la transición hacia la sexta generación de redes y la importancia de contar con una infraestructura de red confiable. En los siguientes apartados, se analizarán estas cuestiones con el propósito de establecer un marco teórico que permita comprender la problemática abordada en este trabajo.

2.1.1. Redes 6G

El constante avance de las tecnologías en el ámbito de las comunicaciones móviles ha sido un pilar fundamental para la evolución de las redes inalámbricas. Cada generación de redes desde la implantación de la tecnología *First Generation* (1G) hasta el actual 5G han marcado un hito en la forma que tenemos de comunicarnos como sociedad. Sin embargo, la acelerada innovación y crecientes demandas de conectividad y rapidez de conexión, implican la necesidad de prepararse para una siguiente etapa: las redes 6G.

Como se comenta en diversos artículos científicos como [GPM+20] o [ASa21], la confianza en la conectividad es primordial en entornos altamente dinámicos, donde la seguridad y la integridad de las comunicaciones son esenciales. Una limitación significativa de las redes 5G radica en su capacidad para garantizar esta confianza en escenarios cada vez más complejos.

El prometedor *Internet of Things* (IoT) [Eva11] [REC15] se está convirtiendo cada vez más en una realidad y un elemento central de nuestras vidas. Esto implica una variedad cada vez mayor de dispositivos conectados a internet que abarcan desde dispositivos

médicos a coches autónomos o simples electrodomésticos. El IoT es concebido como una estructura compuesta por múltiples redes, donde una variedad de dispositivos asociados con diversos aspectos de la vida cotidiana poseen sus propias redes individuales y, a su vez, se interconectan entre sí con el fin de optimizar, por ejemplo, la seguridad o la gestión.

Esta expansión significativa de dispositivos genera ciertos obstáculos a sortear, como la enorme cantidad de datos que requieren y el uso de una red robusta, segura y confiable para el procesamiento y análisis de esos datos en tiempo real. El IoT requerirá una nueva arquitectura de red con un grandes reestructuraciones en el diseño de las redes móviles actuales. En este contexto las redes 6G aparecen como una mejora de su predecesor el 5G, ofreciendo la capacidad y confiabilidad necesarias para satisfacer las necesidades cada vez mayores de este ecosistema digital.

Para abordar estas demandas de las redes móviles, se requerirán entre otros: *Quality of Service* (QoS) (Calidad de Servicio), altas velocidades de datos, baja latencia y *Quality of Experience* (QoE) (Calidad de experiencia). Para ello las redes 6G deberán construirse bajo nuevos paradigmas como la desagregación y virtualización de equipos de red o diseño de redes heterogéneas. Además el desarrollo de esta nueva sexta generación implicará también serios desafíos relacionados con el consumo energético. Dado que en el IoT se contemplarán una gran cantidad de dispositivos conectados es necesario el diseño de mecanismos novedosos con el fin de reducir el coste energético y mantener la calidad de experiencia, como se comenta en el artículo [ASa21].

Por otra parte, las redes 6G beneficiarán enormemente la integración de tecnologías como la inteligencia artificial. Las altas capacidades que se esperan de esta sexta generación permitirían una optimización de los recursos de la red, mayor capacidad de adaptación a las cambiantes condiciones y una gestión más inteligente del tráfico.

La inteligencia artificial en este contexto proveerá un pilar fundamental para la optimización del rendimiento de la red y la toma de decisiones automatizadas para mejorar la experiencia de los usuarios. Las futuras redes 6G deben ser capaces de proporcionar una conectividad ininterrumpida y cumplir con una variedad de requisitos QoS para una gran cantidad de dispositivos. En un estudio realizado acerca de esta área [YAX+20] se proponen técnicas de *Inteligencia Artificial* (IA) con capacidades avanzadas de análisis, aprendizaje, optimización y reconocimiento inteligente, las cuales pueden ser empleadas en las redes 6G para llevar a cabo tareas inteligentes como optimización de rendimiento, descubrimiento de conocimiento, aprendizaje sofisticado, organización estructurada y toma de decisiones automatizadas y complejas.

2.1.2. Confiabilidad

La confianza en la red es uno de los mayores retos a conseguir con esta nueva generación de redes inalámbricas. Para abordar esta problemática se pueden encontrar diversos enfoques, uno de ellos, propuesto en el artículo [AL23], contempla una estimación de confianza basado en las siguientes tecnologías clave:

1. *Atestación de componentes*: La atestación es al proceso mediante el cual un servicio o componente de red verifica su configuración de hardware y software. Esta tecnología es fundamental para asegurar la integridad y autenticidad de los elementos de la red. Los autores del artículo proponen la tecnología *blockchain* para mejorar la seguridad de la atestación, permitiendo trazabilidad e inmutabilidad.

2. *Proof of Transit* (PoT) (Pruebas de Tránsito): Las PoT garantizan que el tráfico de datos sigue un camino preestablecido a través de la red, lo cual es vital para prevenir manipulaciones o desvíos maliciosos. Este método fortalece la confianza en la red al asegurar que los datos no sean interceptados o alterados durante su transmisión.

3. *Uso de Contratos Inteligentes*: Los contratos inteligentes juegan un papel crucial en la gestión automática y transparente de las operaciones de red y los *Service Level Agreement* (SLA) (Acuerdos de nivel de servicio). Estos contratos permiten ejecutar acuerdos preprogramados almacenados en la *blockchain*, lo que proporciona una trazabilidad y cumplimiento eficiente de los QoS prometidos.

En la Figura 2.1 se puede observar un esquema de la estimación del nivel confianza incluyendo los tres componentes descritos.

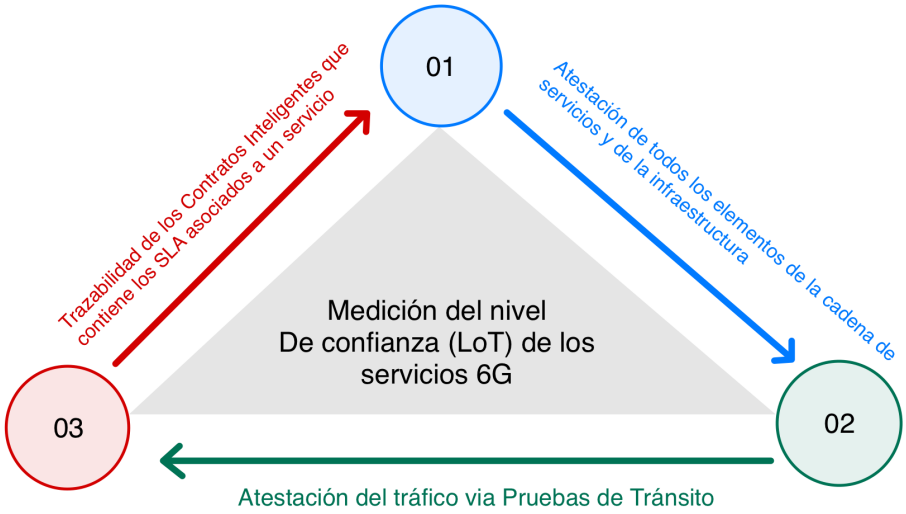

Figura 2.1: Esquema de la estimación del nivel confianza de servicios 6G.

La confianza es un pilar fundamental en el desarrollo y despliegue de redes 6G. Las tecnologías de atestación, las pruebas de tránsito y el uso de contratos inteligentes son esenciales para construir una infraestructura de red en la que los usuarios puedan confiar. En el artículo [AL23] destaca la importancia de integrar diversas tecnologías de manera efectiva para abordar los desafíos de seguridad inherentes a las redes 6G y garantizar su éxito en el futuro cercano. Para consolidar el entendimiento de la mejora de la confiabilidad en las redes 6G se exponen una serie de elementos que pueden contribuir a esta mejora de la seguridad:

- **Orquestación Inteligente de Servicios**: Para manejar eficientemente los servicios en redes 6G, se utiliza un sistema de orquestación que incorpora técnicas avanzadas de inteligencia artificial. Esta orquestación no solo mejora la eficiencia operativa sino que también refuerza la confianza al permitir una gestión más precisa y segura de los recursos de la red. La adopción de aprendizaje automático y otras técnicas de IA permite la automatización de las tareas administrativas y la toma de decisiones en tiempo real, lo que resulta esencial para mantener la integridad y confiabilidad de la red.

- **Algoritmo de confianza**: El algoritmo de confianza se refiere a la evaluación de la confianza obtenida a través de la autobservación o de la retroalimentación de la reputación de otro objeto de IoT. Los principales algoritmos y enfoques utilizados para evaluar la confianza incluyen la inferencia bayesiana, la lógica difusa, cadenas de markov, etc. Los autores del artículo [NSW19] en relación con este ámbito, han propuesto algoritmos de inspiración biológica como la colonia de hormigas y Optimización del *Particle Swarm Optimization* (PSO) (Enjambre de partículas). Últimamente ha surgido un nuevo algoritmo que se utiliza para calcular la confianza en IoT, que se basa en el aprendizaje automático.

  La evaluación de confianza se basa en el análisis de las probabilidades condicionales. Cuando el confiado necesita servicios confiables en el tiempo *t+1*, se evalúa si el confiador tiene la capacidad de prestar el servicio en ese momento. Este análisis se hace teniendo en cuenta los valores actuales y pasados de las variables relevantes en la red que afectan a esta capacidad. En otras palabras, se considera el estado de la red y de los componentes relevantes para determinar si el servicio será confiable en el futuro. Para esta evaluación de confianza se pueden emplear diferentes métodos de inteligencia artificial como por ejemplo:

  1. *Cadena de Markov*: Es un método relativamente común y relativamente simple para el modelado estadístico de variables estocásticas. El modelo de confianza en una cadena de Markov tiene una propiedad sin memoria, porque el estado de confianza futuro depende solo del estado actual de un nodo y no del estado anterior. Cada nodo evalúa el modo de confianza de sus vecinos y envía información sobre él a los mismos. La Figura 2.2 muestra un esquema de este modelo.

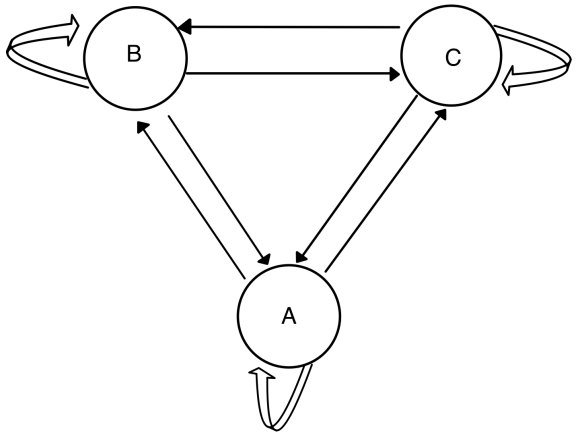

Figura 2.2: Ejemplo de una cadena de Markov con tres nodos.

  2. *Lógica difusa*: Ofrece una oferta flexible, de esta manera, la cantidad de incertidumbre puede ser determinada por cada situación. Se basa en la toma de decisiones humana y al introducir el concepto de grado de verdad al confirmar una condición, proporciona flexibilidad en el razonamiento para dar cabida a imprecisiones e incertidumbres.

- **Propagación de confianza**. La propagación de confianza esta relacionada con proceso de propagación de la puntuación de resultados de evaluación de confianza a otros nodos o sistemas. La propagación se puede gestionar a nivel de nodo o a nivel de *cluster* como se describe en el artículo [NSW19].

    1. *Nivel de nodo*. En este modelo de propagación, los dispositivos de IoT propagan de forma autónoma la puntuación de confianza a otros dispositivos de IoT sin el uso de un coordinador o un jefe de *cluster*.
    2. *Nivel de grupo*. La propagación de confianza en un sistema de IoT basado en *cluster* la consolida el coordinador del *cluster*.

2.2. Lógica Difusa como Técnica para la Estimación de Confianza

El Algoritmo LoT, que se basa en la lógica difusa, utiliza este enfoque para calcular el nivel de confianza en una red. La lógica difusa se origina en la teoría clásica de conjuntos, la cual establece que un elemento puede pertenecer a un conjunto con un grado de pertenencia que varía más allá de los valores absolutos 0 y 1.

La lógica difusa es una rama que maneja expresiones que no son ni completamente ciertas ni totalmente falsas. Dicha lógica aplica a conceptos que pueden asumir cualquier valor de veracidad dentro de un espectro que va desde la verdad absoluta hasta la falsedad total. Por lo tanto, los sistemas de control que emplean lógica difusa combinan variables de entrada, las cuales se definen en términos de conjuntos difusos, a través de grupos de reglas que producen uno o varios valores de salida.

Esta aproximación permite que los valores representen cualquier grado de verdad entre 0 y 1, en contraste con la lógica binaria tradicional, donde las variables solo pueden ser verdaderas o falsas (1 ó 0). Esto la hace particularmente útil para manejar información imprecisa y para modelar situaciones del mundo real donde las respuestas no se limitan a un simple sí o no. Esta comparativa entre la lógica binaria tradicional y la lógica difusa se puede observar en la Figura 2.3.

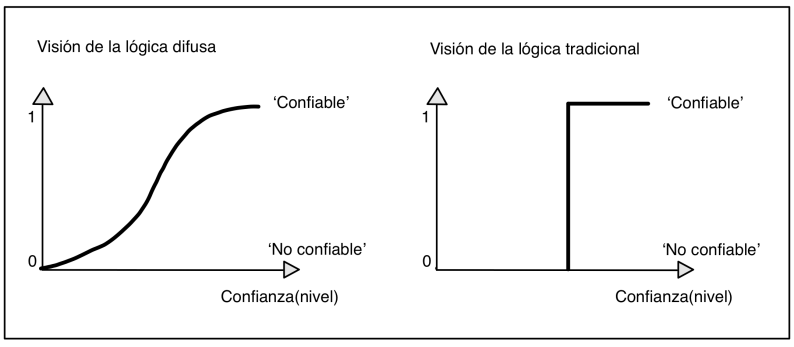

Figura 2.3: Comparativa entre lógica difusa y lógica binaria tradicional.

El modelo de lógica difusa se divide en tres bloques principales:

1. *Bloque fuzzificador*: Asigna a cada variable de entrada un grado de pertenencia a los conjuntos difusos previamente definidos, utilizando funciones características. Las entradas de este bloque son valores concretos de las variables y las salidas son los grados de pertenencia.
2. *Bloque de Inferencia*: Este bloque aplica mecanismos de inferencia para relacionar los conjuntos difusos de entrada con los de salida, basándose en las reglas que definen el sistema. Las entradas son los grados de pertenencia a conjuntos difusos y las salidas son también conjuntos difusos asociados a la variable de salida.
3. *Bloque deffuzificador*: Transforma el conjunto difuso resultante del bloque de inferencia en un valor real para la variable de salida, utilizando métodos matemáticos de deffuzificación.

Dos de los métodos de lógica difusa, aplicados en el algoritmo LoT, son la función de membresía triangular para el bloque de inferencia y el método del centroide para la defuzzificación:

1. *Función de Membresía Triangular*: Una función de membresía triangular se caracteriza por tres puntos: el punto de inicio a, el punto máximo b, y el punto final c. La función aumenta linealmente desde a hasta b y luego disminuye linealmente de b a c. Un ejemplo de esta se puede observar visualmente en la Figura 2.4.

   Este tipo de función se utiliza para modelar conceptos difusos como *malo*, *bueno*, o *excelente* por ejemplo, dependiendo del contexto del objeto de estudio.

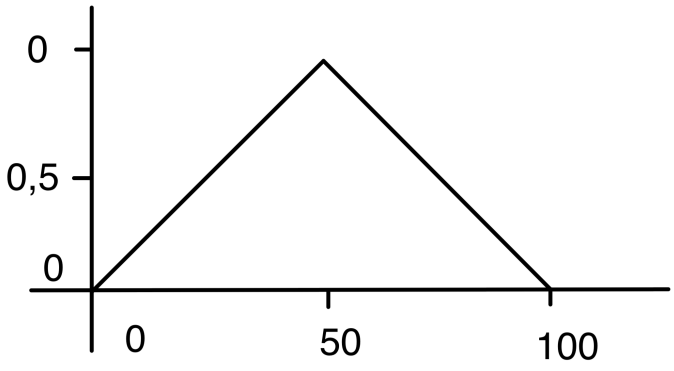

Figura 2.4: Ejemplo gráfico de una función de membresía triangular.

2. *Método del Centroide para la Defuzzificación*: El método del centroide es uno de los métodos más comunes para la defuzzificación. Este es el proceso de convertir la salida de un sistema, el cual es un conjunto difuso, en un número real. Esto es necesario para que los sistemas difusos puedan interactuar efectivamente con el mundo real y otras lógicas no difusas. Un ejemplo gráfico de este método se muestra en la Figura 2.5.

   El método del centroide proporciona un resultado que es intuitivamente atractivo porque refleja el ”promedio ponderado”de todas las posiciones posibles, ponderado por sus grados de membresía.

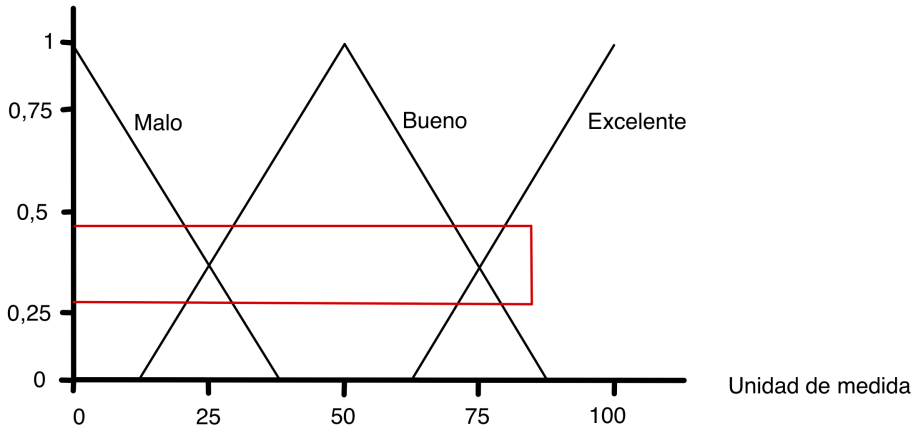

Figura 2.5: Ejemplo gráfico del método del centroide.

2.3. Ethereum Blockchain y las DLT

Las tecnologías que serán detalladas en los apartados siguientes constituyen un tipo de infraestructura tecnológica cuya implementación presenta el potencial de mejorar la confiabilidad de la red. En estos apartados, se proporciona una explicación tanto de las DLT como de la plataforma *Ethereum*, que implementa este tipo de tecnología. En este ámbito se realiza también una explicación del funcionamiento de los contratos inteligentes, ya que es una de las tecnologías que también han sido aplicadas en el presente trabajo.

2.3.1. Distributed Ledger Technology (DLT)

Una de las tecnologías que se han empleado en el proyecto ha sido la plataforma *Ethereum*, la cual se fundamenta en una DLT, que representa un sistema de registro y validación descentralizado. Este enfoque, mediante el uso de técnicas criptográficas avanzadas y la descentralización, asegura la integridad y seguridad de los datos almacenados en ella, garantizando la inmutabilidad de los registros evitando la manipulación o falsificación.

Este tipo de sistemas funcionan con una red *Peer-to-Peer* (P2P), en la cual, cada nodo participa en la validación de eventos o transacciones que emergen en la red [N+08] [KLDS20]. En este contexto la DLT distribuye la responsabilidad entre todos los nodos de la red, creando así una infraestructura en la que todas o algunas de sus características funcionan sin necesidad de servidores fijos, lo que significa que no hay una entidad central que supervise o controle la red. Esta implementación implica que cada nodo actúe como servidor y cliente simultáneamente respecto al resto de nodos, como se puede observar en la Figura 2.6, que muestra una comparación entre la arquitectura centralizada y la descentralizada.

Las *Distributed Ledger Technologies* funcionan con un libro mayor distribuido, el cual se propaga entre todos los nodos de la red. Cada participante de la DLT guarda una réplica del libro mayor por completo. Para añadir nuevas entradas al libro debe de enviarse una nueva transacción a la red P2P y ser validada por el resto de participantes.

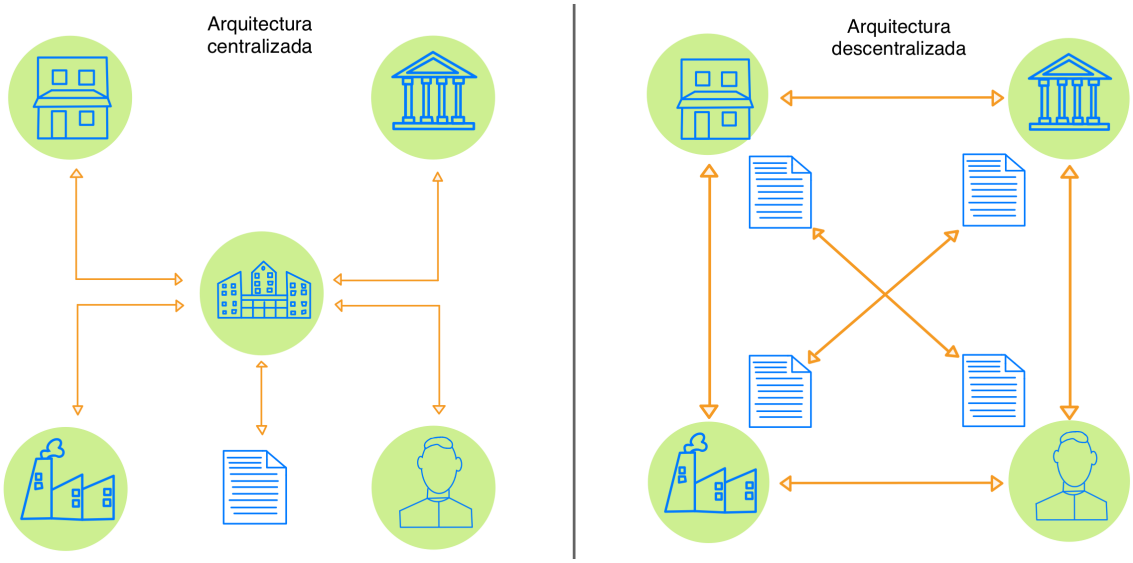

Figura 2.6: Comparación entre arquitectura centralizada y descentralizada.

Como se comenta en el capítulo 9 del libro [Sun20], dentro de los sistemas DLT se comprenden los diseños públicos o diseños privados. Las DLT públicas, debido al alto número de nodos que participan, tienen un gran nivel de disponibilidad y deben tener diseños escalables para priorizar el rendimiento ante el aumento arbitrario de nodos en la red. Sin embargo en los diseños de DLT privadas el número de nodos es conocido y cada uno de ellos es identificable por otros miembros de la red.

Además de esta distinción existen múltiples implementaciones de esta tecnología, en uno de los capítulos del libro [EIP18] se identifican 4 de los tipos de DLT más empleados:

1. *Blockchain*: Las cadenas de bloques se basan principalmente en el concepto de máquinas de estados replicadas, en las cuales cada nodo conserva una réplica local del registro distribuido en un estado específico *sn*, en el que si aparece una nueva transacción significa la evolución del estado *sn* al estado *sn+1*. En esta implementación cada bloque establece una conexión con su predecesor mediante un valor *hash* que hace referencia a dicho bloque.

2. *Tangle*: Este tipo de DLT es una infraestructura descentralizada para el almacenamiento de datos y un protocolo de consenso que se fundamenta en una estructura de datos conocida como *Directed acyclic graph* (DAG) (Grafo acíclico dirigido). En esta estructura, cada nodo, representa una transacción, y las relaciones entre transacciones representan las validaciones de las mismas.

3. *Hashgraph*: Con este diseño se implementa también una estructura de datos DAG, que se emplea para almacenar transacciones junto con un algoritmo de votación y un protocolo de difusión para lograr un consenso eficiente. El principal logro de *Hashgraph* radica en su protocolo de consenso, cuya eficacia se respalda mediante pruebas matemáticas, demostrando que facilita una replicación de datos considerablemente de manera más rápida que la tecnología *blockchain*.

4. *Sidechain*: La innovación de la arquitectura *sidechain* consiste en la fusión de dos estructuras *blockchain*. Este enfoque se basa en una cadena de bloques para

administrar solicitudes de acceso de los nodos participantes, seguida por un conjunto de cadenas de bloques privadas, que gestionan las transacciones locales. Esta configuración posibilita que cada cadena local mantenga su información de manera privada y comparta únicamente datos seleccionados con el resto de cadenas.

Estas diferentes implementaciones se pueden observar de manera visual en los ejemplos de la Figura 2.7.

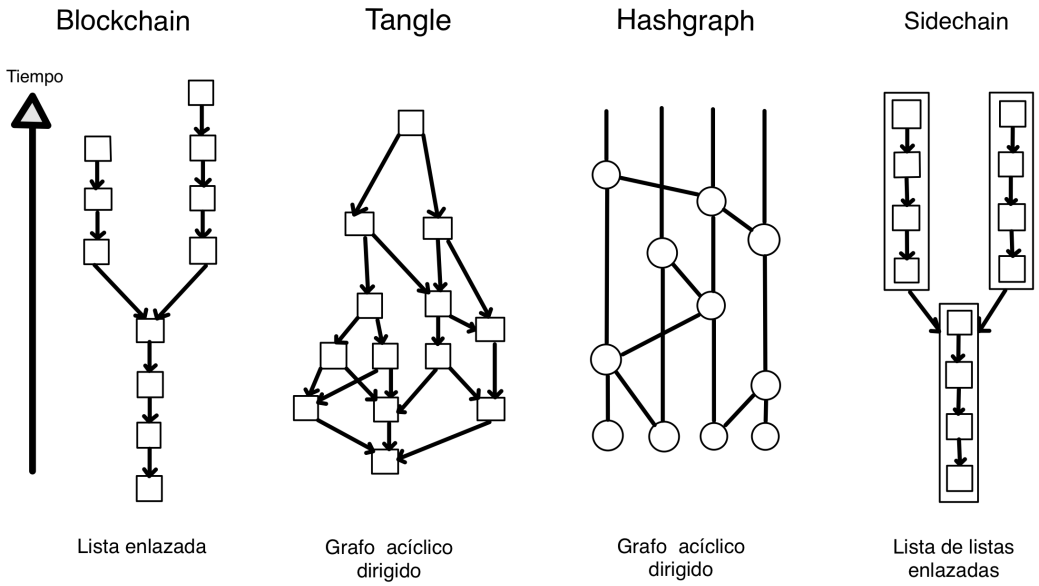

Figura 2.7: Distintos tipos de DLT.

En las infraestructuras fundamentadas en DLT los datos son almacenados de manera cronológica en el libro mayor en forma de transacciones. Cada una de estas transacciones contiene una serie de metadatos necesarios para la validación o cumplimiento de la misma, como por ejemplo el remitente de la transacción, una firma digital o el código de un contrato inteligente. Cuando un nodo envía una nueva transacción, es recibida por el resto de participantes y validada mediante pruebas basadas en firmas digitales y criptografía de clave pública y privada.

Estas tecnologías disponen de mecanismos de consenso diseñados para facilitar actualizaciones del libro mayor de manera segura y fiable. Un ejemplo de estos protocolos es la *Proof of Work* (PoW) (prueba de trabajo), aplicada por *Bitcoin* [N+08]. Esta metodología de validación implica el cálculo de un *hash* que cumpla con una cantidad predefinida de ceros al inicio, lo que implica un coste computacional específico para la resolución del problema.

En el diseño de la mayoría de las *Distributed Ledger Technologies*, el concepto de *hash* es un componente fundamental en el proceso de validación de transacciones. Las funciones *hash* son un pilar fundamental en los sistemas DLT ya que garantizan la integridad de los datos y la seguridad de la red, además de permitir el uso de firmas digitales. Estas funciones son algoritmos que producen resultados únicos para entradas específicas, generando siempre el mismo resultado para una serie de datos iniciales. Esto garantiza que sea extremadamente difícil para un atacante adivinar la entrada original a partir de

un valor *hash*, ya que las funciones *hash* en ningún caso revelan información sobre los datos originales. En el capítulo 9 del libro [Sun20], se sostiene que debido a la dificultad de encontrar una entrada para reproducir un *hash* objetivo, la reconstrucción del dato original mediante un ataque de fuerza bruta se vuelve prácticamente imposible.

Las transacciones representan registros o modificaciones de los datos almacenados en la DLT. La naturaleza específica de las transacciones puede variar según la aplicación o el tipo de DLT en uso, sin embargo, con la información recogida de libros, estudios y artículos científicos como: [N+08], [B+13], [Sun20] o [EIP18], se puede concretar que su validación generalmente involucra una secuencia de pasos específicos:

1. **Firma de la transacción**: La transacción se firma digitalmente por el remitente para garantizar la autenticidad e integridad de la misma.

2. **Envío de transacción**: El participante de la red envía la transacción al resto de nodos para añadirla al libro mayor. Esta transacción puede incluir por ejemplo transferencia de activos digitales o la dirección de un contrato inteligente para la ejecución de algunas de las funciones de su código.

3. **Validación de la transacción**: Los nodos de la red se encargan de validar la transacción con una serie de mecanismos de consenso dependientes del tipo de arquitectura de la DLT.

4. **Agregación al libro mayor**: Una vez la transacción ha sido validada se agrupa con el resto de transacciones válidas para formar parte del libro mayor.

5. **Propagación**: Con la transacción validada y añadida, esta se propaga por los nodos de la red para que actualicen su réplica del libro mayor. Una vez que todos los participantes de la red actualizan su libro mayor, la transacción se considera completada.

Este proceso garantiza que todas las transacciones que han sido almacenadas en el libro mayor sean válidas e inmutables además de visibles para todos los participantes de la red. Proporcionando así un registro seguro y confiable de todos los eventos de la red.

Este tipo de tecnologías son prácticamente imposibles de atacar, sin embargo como se explica en un apartado del artículo [KLDS20], hay algunas amenazas significativas, como por ejemplo la llamada el ataque del 51 %. Esta amenaza implica que si un participante de la red o grupo de participantes controlan más del 50 % de la red, tendrían la capacidad de influir en el proceso de validación comentado anteriormente. Para evitar este tipo de ataques la mayoría de las DLT implementan mecanismos de consenso robustos como la prueba de trabajo de *Bitcoin* o la prueba de participación de *Ethereum*. Estas pruebas hacen que sea extremadamente costoso adquirir una mayoría del poder computacional de la red, ayudando a la integridad y seguridad de la misma.

2.3.2. Ethereum

Toda la información recogida en esta sección sobre el funcionamiento de la plataforma *Ethereum* y su diseño de la *blockchain*, ha sido extraída y recopilada de diversos artículos científicos o documentación oficial como: [N+08], [B+13] ó [Eth23].

En el presente trabajo, se ha empleado la plataforma *Ethereum*, específicamente haciendo uso de la EVM para la ejecución de un contrato inteligente. *Ethereum* es una plataforma descentralizada de código abierto, lanzada en 2015, que facilita la construcción y despliegue de contratos inteligentes y *Decentralized Applications* (dApps) (Aplicaciones descentralizadas). Se sustenta en un tipo de DLT, en concreto la *blockchain* o cadena de bloques en español.

La *blockchain* es una tecnología DLT que proporciona un medio seguro y transparente para almacenar y verificar datos de manera descentralizada. Esta tecnología consiste en una secuencia de bloques de datos, donde cada bloque contiene un conjunto de transacciones y un *hash* criptográfico que apunta al bloque anterior, formando así una cadena como se muestra en la Figura 2.8.

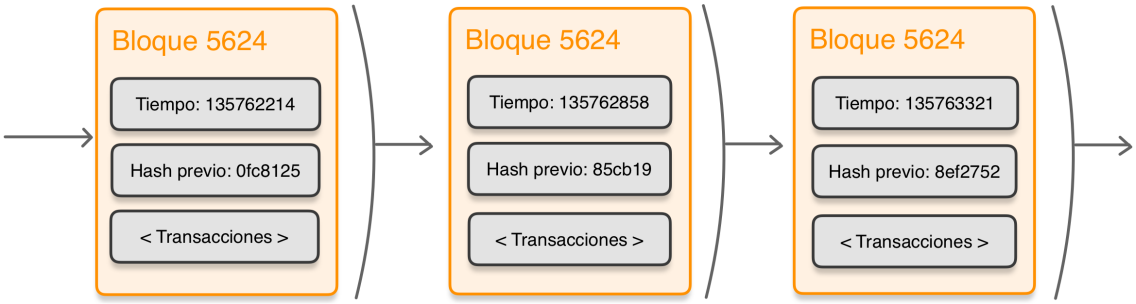

Figura 2.8: Ejemplo de una cadena de bloques.

La característica principal de la *blockchain* es que cada uno de los bloques que la componen están conectados con el anterior hasta llegar al bloque génesis, que es el primero que se propagó dentro de la red. Esta conexión se realiza mediante una "firma criptográfica" representada por un *hash*. Esto proporciona a la cadena una secuencia inmutable y cronológica de bloques, siendo esencial para la integridad de la misma, ya que cualquier intento de modificar un bloque anterior requeriría la modificación de todos los bloques posteriores. Como resultado, si se realiza un ataque de esta naturaleza, los demás participantes de la red detectarán una discrepancia entre los *hashes*, lo cual sugiere una alteración en la cadena de bloques. En ese caso se considera el bloque como inválido, se niega su aceptación y se comunica la decisión al resto de nodos durante el proceso de consenso.

Estas propiedades de la cadena de bloques sirven como fundamento para las transacciones en *Ethereum*, las cuales representan transferencias de valor y ejecución de contratos inteligentes en la red entre otras aplicaciones.

Las transacciones consisten en órdenes de cuentas firmadas criptográficamente. Una cuenta realizará una transacción con el fin de modificar el estado de la red. Las cuentas en *Ethereum* son entidades que poseen una cantidad específica de la divisa digital que maneja la plataforma, el *Ether* (ETH). Estas cuentas pueden ser controladas directamente por usuarios individuales, contratos inteligentes u otras aplicaciones descentralizadas.

Para poder llevar a cabo el procesamiento de una transacción, esta requiere de una tarifa además de su inclusión en un bloque válido. Esta pequeña tarifa está expresada en *gas*, el cual es una medida de la cantidad de trabajo computacional requerido por un validador para procesar una transacción. Esta forma de pago se utiliza como mecanismo para limitar la cantidad de esfuerzo computacional que una transacción puede emplear en

la red, evitando así ataques de *spam* y el bloqueos en bucles computacionales infinitos.

Estas tarifas de *gas* deben ser pagadas en ETH, y para ello se hace un cálculo con otra unidad de medida llamada *gwei* que representa una fracción muy pequeña de ETH. Por lo tanto el precio en ETH de una transacción es la cantidad de *gas* consumido multiplicado por el precio del *gas* (en *gwei*). Estas medidas implican que las transacciones en su envío lleven consigo una serie de parámetros referidos al *gas*, además de otros valores como puede ser la dirección de cuenta del remitente.

Para poder validar las transacciones por parte de los participantes de la red, estos deben de cumplir y llevar a cabo una serie de mecanismos de consenso. Actualmente *Ethereum* emplea la *Proof of Stake* (PoS) (Prueba de participación), un mecanismo con el cual los validadores de la red "apuestan" una cantidad de ETH transformado en un contrato inteligente. Si estos intentan engañar a la red, alguno o todo su capital apostado en forma de ETH puede ser destruido.

Las transacciones en *Ethereum* pueden utilizarse no solo para la transferencia de fondos, sino también para invocar la funcionalidad de un contrato inteligente o desplegarlo en la red. Las transacciones disponen de un campo especializado opcional que, en caso de ser una invocación a un contrato, permite especificar valores como el método que debe ser invocado dentro del código, o bien, si la transacción implica el despliegue del contrato en la red, dicho campo transportará su código.

Para poder ejecutar el código de estos contratos inteligentes la arquitectura de *Ethereum* implementa un componente central llamado Máquina Virtual *Ethereum* (EVM, por sus siglas en inglés). La EVM es un entorno de ejecución y es la responsable de procesar todas las transacciones que cambian el estado de la red. *Ethereum* está fundamentada en la tecnología DLT pero va más allá de este concepto, ya que no solo mantiene un libro mayor distribuido si no una máquina de estados distribuida. *Ethereum* es una arquitectura que no solo almacena datos de cuentas y sus respectivos saldos, además tiene la capacidad de ejecutar código. Este tipo de arquitectura permite el desarrollo de los contratos inteligentes.

En este trabajo, para la comunicación con la red *Ethereum* y el envío de transacciones, se ha empleado *Hyperledger Besu*. Esta tecnología es una implementación de un cliente *Ethereum* de código abierto. *Besu* proporciona opciones para ejecutar el nodo como cliente de la *Mainnet* de *Ethereum*, redes públicas de prueba o redes privadas, incluyendo documentación para la configuración de estas últimas.

Besu ofrece tanto una interfaz de línea de comandos como una *Application Programming Interface (API) Remote Procedure Call Protocol encoded in JSON (JSON-RPC)* que facilitan la gestión, mantenimiento, depuración y monitoreo de nodos en una red *Ethereum*. Esta API puede ser utilizada mediante a través de Hypertext Transfer Protocol (HTTP) o a través de un *WebSocket*. Además, en términos de funcionalidad, *Besu* abarca una amplia gama de casos de uso, incluyendo operaciones, desarrollo de dApps y contratos inteligentes. Para esto, se integra con populares herramientas como *Web3j*.

Web3j es una biblioteca de Java empleada en este proyecto de investigación en concreto y que permite interactuar con nodos de *Ethereum*, enviar o recibir transacciones y crear y desplegar contratos inteligentes. Además *Web3j* permite la compilación de contratos inteligentes escritos en *Solidity*, de tal manera que produce archivos *Java* con clases que representan el contrato inteligente, permitiendo la integración de las funcionalidades del contrato a la aplicación escrita en *Java*.

2.3.3. Contratos Inteligentes

Nick Szabo en sus articulos [Sza97] y [Sza96] introdujo ya en los años 1996 y 1997 el término de contrato inteligente. En estos artículos, Szabo propuso los contratos inteligentes como formalizaciones de relaciones que proporcionan seguridad y garantías al aprovechar el hardware y el software, sugiriendo que estos contratos sean dinámicos y proactivamente aplicados, lo que permite una mejor observación y verificación que las medidas convencionales.

Los contratos inteligentes son programas informáticos que se ejecutan automáticamente y regulan, verifican o hacen cumplir la negociación o ejecución de un contrato, o de una parte del mismo. Esta implementación supone la emisión segura de eventos o transacciones sin necesidad de terceros, como se explica también en el libro [Sun20]. El código del contrato una vez que está almacenado en una DLT es rastreable e inmutable, de tal manera que se garantiza siempre un determinado resultado para unas entradas específicas.

Dentro de esta definición, *Ethereum* es una de las plataformas que permiten la ejecución de código de contratos inteligentes en su EVM. Para ello la EVM necesita convertir el código del contrato a *bytecode*, además, el compilador también genera archivos con extensión *Application Binary Interface* (ABI), pudiendo ser utilizados por una aplicación externa para comunicarse con el contrato a través de transacciones.

Estos contratos inteligentes se almacenan en una cuenta de *Ethereum*, con una dirección específica y datos sobre su estado. *Ethereum* para la compilación de contratos inteligentes soporta dos lenguajes, *Solidity* [Sol] y *Vyper* [Vyp]. Con estos dos lenguajes se puede codificar un contrato con unas determinadas limitaciones dentro del marco de la EVM y el lenguaje escogido en sí. Los contratos inteligentes permiten en *Ethereum* almacenamiento persistente, funciones externas e internas, herencia, manejo de divisas o direcciones de pago entre otras características.

Conforme a la documentación oficial de *Ethereum* [Eth23], el procedimiento que conduce desde la codificación de un contrato inteligente hasta su eventual ejecución se describe de la siguiente manera:

- **Despliegue del contrato**: Una vez se escribe el código del contrato en uno de los dos lenguajes soportados por el compilador, se procede al despliegue del contrato en la red. El despliegue de un contrato requiere de cuenta asociada con suficiente ETH, dado que dicho despliegue conlleva una consumición de *gas* que reduce el saldo de la cuenta. Este proceso se lleva a cabo mediante una transacción enviada a través de un nodo participante en la red, durante la cual, el código viaja a través del campo dedicado para ello. Posteriormente, los nodos participantes en la red procesan y validan la transacción, asignando así una dirección única al contrato desplegado.

- **Interacción con el contrato**: Para poder ejecutar las funciones del contrato desplegado, se debe emplear un nodo activo en la red con suficiente saldo ETH en su cuenta. La interacción se produce con una transacción que lleva como argumentos la dirección del contrato y las llamadas a funciones o datos necesarios para la ejecución del mismo. El remitente de la transacción debe pagar la tarifa de *gas* que se ha establecido por la ejecución de esa parte del código. Además en *Ethereum* los contratos inteligentes son públicos, por lo que desde un mismo contrato se pueden iniciar otras transacciones para llamar a otros contratos, acción que aumentaría la tarifa de *gas*.

Por último cabe destacar que los contratos inteligentes presentan diversas limitaciones, entre las cuales se encuentra el tamaño máximo permitido para su implementación. Concretamente, un contrato inteligente no puede exceder los 24 KB de tamaño, dado que superar este límite resultaría en una insuficiencia de *gas* en la ejecución.

Otra limitación es la incapacidad de los contratos inteligentes para acceder a información proveniente de fuentes externas a la cadena de bloques, lo que impide su capacidad de respuesta ante eventos del mundo real. Esta limitación puede comprometer el consenso, que es un aspecto fundamental para garantizar la seguridad y la descentralización en el ecosistema de la *blockchain*.

No obstante, es esencial que las aplicaciones basadas en *blockchain* puedan acceder a datos externos para su funcionamiento. Para esa aplicación se puede recurrir a los oráculos como se muestra en la Figura 2.9, estas herramientas están diseñadas exclusivamente para procesar información proveniente de fuentes externas y son encargadas de facilitarla a los contratos inteligentes.

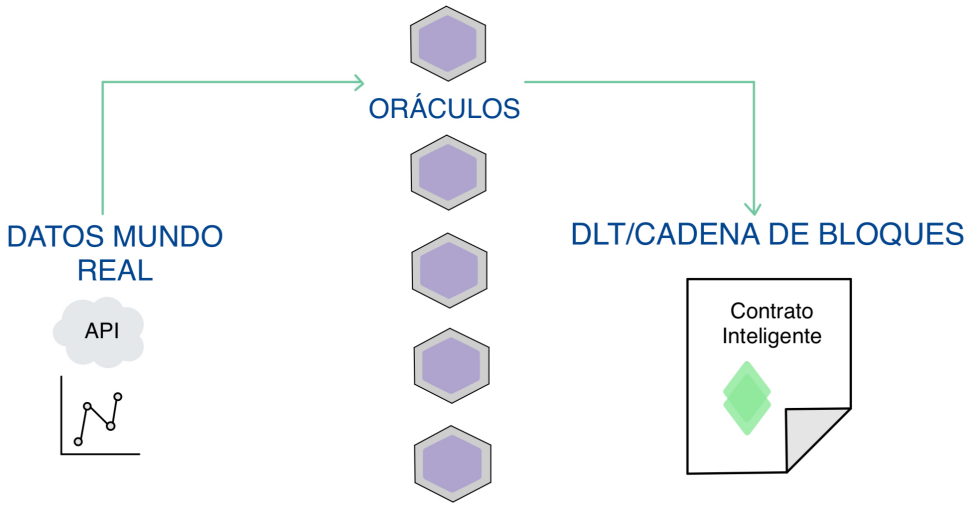

Figura 2.9: Esquema del uso de oráculos.

Capítulo 3

Estado del Arte

En este capitulo se presentan investigaciones relacionadas con el área que concierne al trabajo. Para ello, se realiza una subdivisión en dos aspectos. Por un lado, se analizan trabajos relevantes acerca de la seguridad y la confianza en las redes, mientras que por otro lado se exploran trabajos que detallan el uso de tecnologías DLT y contratos inteligentes en este ámbito de confianza de la red.

3.1. Confianza y Seguridad de las Redes 6G

Las redes 6G enfrentan desafíos significativos relacionados con la seguridad y la privacidad debido al aumento en el número de dispositivos conectados, la complejidad de las cadenas de servicio virtualizadas y el crecimiento en ataques de ciberseguridad. Estos desafíos requieren un enfoque robusto que garantice la protección de datos y la confiabilidad de la red.

La propuesta del artículo [AL23] incluye el uso de la atestación de componentes hardware y software para verificar la integridad y seguridad de los elementos de la red. Además, se emplea la PoT para asegurar que el tráfico de datos sigue rutas predefinidas, evitando desviaciones que puedan comprometer la seguridad. Los contratos inteligentes sobre plataformas de DLT permiten automatizar la gestión de los niveles de QoS y asegurar la ejecución de acuerdos de servicio. La arquitectura expuesta por los autores utiliza técnicas de aprendizaje automático y orquestación inteligente para optimizar la gestión de las cadenas de servicio. Esto incluye la adaptación dinámica de la red para cumplir con los requerimientos de privacidad y seguridad en tiempo real.

Adicionalmente, dada la naturaleza sensible de la información gestionada, se considera esencial la privacidad de los usuarios finales, operadores de servicio y operadores de infraestructura. Los autores proponen técnicas avanzadas como el Cifrado Homomórfico y la Encriptación Buscable para permitir operaciones seguras sobre datos cifrados.

El IoT integra una amplia variedad de dispositivos, desde *wearables* hasta aplicaciones industriales, y juega un papel crucial en la vida cotidiana al interconectar dispositivos y facilitar servicios inteligentes. En este contexto, la confianza es fundamental para garantizar la seguridad, la privacidad y la eficacia de las interacciones entre dispositivos, siendo un componente esencial para la implementación exitosa de sistemas IoT.

En el artículo [NSW19] se presenta un estudio exhaustivo de los modelos de cálculo de confianza aplicados en IoT, resaltando diferentes metodologías usadas por los investigadores para evaluar la confianza entre dispositivos. Se desarrolla una clasificación que categoriza los modelos de confianza utilizando cinco parámetros: métricas de confianza, fuente de confianza, algoritmos, arquitectura y propagación de la confianza.

Dentro de las métricas de confianza, se distinguen dos principales categorías: basadas en la QoS y confianza social. La confianza directa y la indirecta (o reputación) son identificadas como las principales fuentes, con algunos modelos utilizando un enfoque híbrido que combina ambos tipos para una evaluación más robusta.

Los autores destacan varios algoritmos para el cálculo de la confianza, incluyendo inferencia bayesiana, lógica difusa y aprendizaje automático, adaptados para manejar las características dinámicas y heterogéneas de los entornos IoT. En términos de arquitectura, se exploran enfoques centralizados, descentralizados y basados en la nube para facilitar la gestión de la confianza en dispositivos IoT distribuidos.

La propagación de la confianza se trata tanto a nivel de nodo como de *cluster*, donde se evalúa cómo los dispositivos individuales o los cabezales de *cluster* diseminan las evaluaciones de confianza a otros nodos. Esta propagación es crucial para mantener la integridad y la fiabilidad del sistema en operaciones a gran escala.

En otro artículo relacionado, [CFLW15], los autores defienden a las *Wireless Sensor Networks* (WSNs) como sistemas críticos en campos como el militar, ambiental y la salud, entre otros. Dada su naturaleza, enfrentan desafíos significativos de seguridad debido a su vulnerabilidad a ataques y a su medio de comunicación abierto. Esto requiere un manejo de la confianza eficiente que complemente las medidas de seguridad tradicionales, como la criptografía, que pueden resultar inadecuadas por el alto consumo de energía y uso de memoria en dispositivos con recursos limitados.

Los autores exponen que la teoría Bayesiana se puede emplear para calcular la confianza directa basada en interacciones exitosas y fallidas, mientras que la teoría de la Entropía ayuda a distribuir pesos de manera objetiva entre diferentes valores de confianza indirecta. Esto aborda el problema de la asignación subjetiva de pesos en modelos tradicionales y mejora la adaptabilidad del modelo. El enfoque de este modelo propuesto por los autores reduce significativamente el consumo de energía al evitar cálculos innecesarios y mejora la eficiencia general del sistema de gestión de confianza.

3.2. Blockchain y Descentralización en el Contexto de las Redes 6G

En relación con este área de investigación, se han realizado diversos estudios sobre cómo los contratos inteligentes pueden desempeñar un papel fundamental en la mejora del nivel de confianza en las futuras redes 6G, destacando tanto los desarrollos actuales como los retos a superar. En esta sección, se presentan una serie de artículos relacionados con la integración de la DLT en redes.

Uno de estos artículos, [SG23], expone el uso de tecnologías de DLT, particularmente la arquitectura conocida como *blockchain*, esta tecnología es un relevante tema de discusión en numerosas investigaciones y estudios que abordan tanto las redes 5G como las redes 6G. En el artículo mencionado, los autores exploran la aplicación de estas tecnologías

para mejorar la capacidad y las características de las redes, así como para examinar su viabilidad en la implementación práctica.

En el artículo se comenta que la evolución de las redes hacia la 5G hacia la 6G, surgen nuevos desafíos relacionados con la seguridad, la privacidad, y la trazabilidad de los servicios ofrecidos. Estas redes se caracterizan por integrar una gran cantidad de componentes de red virtualizados y procesamiento masivo de datos, aumentando la complejidad y la vulnerabilidad a ataques cibernéticos. La integración de tecnologías emergentes y el aumento en la digitalización de los procesos demandan un nivel de confianza mejorado en las infraestructuras de red.

Para abordar estos desafíos, se ha propuesto por los autores del artículo [SG23], el uso de contratos inteligentes implementados en plataformas de cadena de bloques como *Ethereum*. Los contratos inteligentes permiten automatizar la monitorización del cumplimiento de los Acuerdos de Nivel de Servicio (SLA) y ofrecen una ejecución transparente y descentralizada, lo cual es crucial para mejorar la trazabilidad, privacidad y seguridad de los servicios en redes 6G.

En el artículo se propone una arquitectura que utiliza plantillas de contratos inteligentes para automatizar la trazabilidad y la verificación de los servicios ofrecidos en las redes 6G. Estos contratos interactúan con oráculos descentralizados para integrar datos del mundo real, permitiendo una compensación automática a los usuarios cuando los niveles de servicio acordados no se cumplen. Adicionalmente, los autores exponen una prueba de concepto para demostrar la viabilidad de esta solución. Utilizando la herramienta de monitorización *Zabbix* y la plataforma de oráculos *Chainlink*. Con ella se ha establecido una comunicación efectiva entre los contratos inteligentes y los datos de rendimiento de la red, lo que permite una ejecución automatizada y la compensación basada en el desempeño real del servicio.

En otro de los artículos sobre esta área [PDKJ22], los autores sostienen el uso de las tecnologías *blockchain* como solución a varios de los problemas a los que una evaluación de confianza centralizada se enfrenta. Estos problemas que presentan en el artículo, están relacionados con los riesgos que implican la necesidad de depender de un tercer componente centralizado. En una arquitectura centralizada, el tercer componente de confianza (Trusted Third Party (TTP), como lo denominan en el artículo), es el encargado de recibir los datos necesarios para más adelante procesarlos y almacenarlos. Este tipo de estructura puede generar ciertos riesgos, dado que al depender de un único punto de acceso, el TTP podría enfrentar diversos inconvenientes que afecten su funcionamiento óptimo y lo pongan en riesgo de compromiso. Esta situación da lugar a incertidumbre en cuanto a la integridad de los datos, tanto aquellos que han sido evaluados como aquellos que están pendientes de evaluación.

Los autores del artículo proponen un modelo de gestión de la confianza de manera distribuida. Este tipo de arquitectura elimina por completo la necesidad de un tercer componente, ya que se basa en la colaboración de todos los nodos de la red con el propósito de proporcionar confianza de entre ellos. Con la tecnología de libro mayor distribuido y mecanismos de consenso, el proceso de evaluación y almacenamiento de resultados sería prácticamente imposible de atacar, estando siempre disponible y manteniendo la integridad y seguridad de los datos. Los autores sustentan sus declaraciones exponiendo algunos ejemplos de éxito de tecnologías como por ejemplo *VANET* [HSBL17], relacionada con el manejo de la redes de vehículos para mejorar la seguridad vial o la eficiencia del tráfico.

La arquitectura propuesta por los autores se fundamenta en la *blockchain* de consorcio semi-distribuida y consta de diversas entidades distribuidas, como servidores perimetrales, usuarios finales y dispositivos de IoT. En esta cadena de bloques cada integrante es distinguible por medio de una clave pública que se encuentra enlazada con una cuenta de criptomonedas. En la arquitectura propuesta, se promueve un enfoque jerárquico con subredes donde cada una de estas tenga su propio modelo de confianza interoperable. En este modelo la *blockchain* facilitaría esta interoperabilidad con transacciones entre diferentes cadenas.

En este estudio se llevaron a cabo dos experimentos para evaluar la efectividad de un sistema de gestión de reputación basado en *blockchain* en el contexto del intercambio de recursos en redes 6G. Para los experimentos se consideró computación de borde en el cual dispositivos con capacidades limitadas transfieren tareas a nodos que poseen mayores recursos y capacidades. En el primer ensayo se midió el cumplimiento de los nodos con los acuerdos de servicio. En el segundo, se evaluó cómo los nodos confiables afectaban la eficiencia en el uso de recursos, destacando la utilidad del sistema de gestión de reputación.

Otra investigación relacionada con el uso innovador de la tecnología *blockchain* en redes [THD+20], examina cómo pueden mejorar estas tecnologías una gran variedad de servicios en diferentes áreas de las redes 5G. Presentan una categorización de aplicaciones de la *blockchain* que podrían resolver, con su integración, una serie de problemas.

En el estudio se hace una distición por aplicaciones que puede ofrecer la *blockchain* en el diseño de las redes 5G y el desarrollo del IoT, en concreto diferencian su utilidad para la gestión de la red, gestión de ubicación de recursos de almacenamiento, gestión de la comunicación, seguridad, privacidad y aplicaciones o servicios.

En una de esas distinciones, la gestión de redes, se discute acerca de las *Software-Defined Networks* (SDN), la virtualización de los componentes de red (*Network Function Virtualization* (NFV)) y el *network slicing*. Haciendo especial mención a la eficiencia, optimización y seguridad de la red. Por parte de los autores se sugiere una arquitectura de control descentralizada de una red SDN para mitigar los riesgos de seguridad y resolver problemas de gestión de redes como puede ser la mejora de confiabilidad, asegurando la transparencia e integridad de la información, y a su vez, evitar puntos únicos de falla que interrumpan o colapsen el sistema. Adicionalmente se mencionan algunas propuestas de SDN, como por ejemplo *DistBlockNet* [SSJP17].

Por otro lado en relación con la gestión informática los autores exponen diversos desafíos de gestión de datos en la nube, se proponen soluciones basadas en tecnologías *blockchain*. Una de ellas es la arquitectura de autenticación basada en *blockchain* [YZZ+17] (*Blockchain Transfer Arquitecture* (BTA)), un sistema que utiliza la tecnología de cadena de bloques para garantizar la seguridad y la integridad de los procesos de autenticación, proporcionando un registro inmutable y seguro de identidades digitales. Otro de estos ejemplos que se plantean en el artículo es la arquitectura basada en *blockchain* diseñada para abordar el desafío de establecer confianza en entornos heterogéneos de *Multi-access Edge Computing* (MEC), en donde la *blockchain* garantiza la seguridad y la privacidad al registrar información sobre la topología de la red de forma que todos los participantes puedan confiar en ella.

En el artículo tras comentar esta clasificación de aplicaciones se aborda como los operadores de redes están actualmente integrando estas tecnologías *blockchain* en sus sistemas, destacando algunos casos de prueba y proyectos significativos en la industria de la telecomunicación. Algunos de estos casos son por ejemplo, la colaboración de IBM y

Telefónica para la mejora de transparencia en el registro y gestión de datos de llamadas internacionales, o una red 5G introducida por el proveedor líder de telecomunicaciones en Corea del Sur impulsada por *blockchain*, donde se emplea esta tecnología para proteger y ocultar direcciones *Internet Protocol* (IP) de dispositivos IoT.

Otra de estas investigaciones en el ámbito de las DLT, [MGV+20], propone la integración de la *blockchain* dentro de la infraestructura de las redes móviles 6G con implementación de contratos inteligentes para acuerdos SLA. En este estudio se aborda la posibilidad de diseñar un modelo fundamentado en la tecnología *blockchain* para una red 6G descentralizada que facilite la gestión de la red por parte de múltiples operadores de red (*Mobile Network Operators* (MNOs)). El modelo que se expone en el artículo emplea la *blockchain* para crear un sistema descentralizado y transparente que sea capaz de asignar, compartir y negociar el uso de frecuencias entre los diferentes operadores de red móvil, lo que podría beneficiar tanto a los operadores como a los usuarios finales.

En este modelo propuesto por los autores, exploran la capacidad de los contratos inteligentes, describiendo varios ejemplo del uso de los mismos. Estos contratos inteligentes permitirían a los operadores de red móvil, los reguladores de espectro y los usuarios finales establecer acuerdos transparentes y automáticos sobre el uso de la infraestructura, el espectro, y la provisión de servicios. En la investigación se explica cómo el usuario final negocia un acuerdo de nivel de servicio (SLA) con los MNOs, basado en la potencia de la señal recibida, y selecciona el contrato inteligente mas conveniente. Según la arquitectura propuesta, el acuerdo SLA se ejecuta automáticamente y los MNOs no podrían hacer cambios en el contrato inteligente ya desplegado en la red.

Capítulo 4

Algoritmo de estimación de nivel confianza LoT

En este capítulo se exponen, en distintas secciones, los procedimientos seguidos en el diseño e implementación para el desarrollo del algoritmo de estimación de confianza. En primer lugar, se realiza una explicación general del software implementado seguida de un apartado dedicado al diseño del algoritmo. A continuación, se detallan, en diferentes secciones, los pasos para el desarrollo del algortimo y la implementación de elementos software que lo componen.

4.1. Descripción General

La solución propuesta en este trabajo al problema de la confiabilidad en redes 6G se fundamenta en un sistema compuesto por una serie de elementos software que, en su conjunto, permiten estimar el nivel de confianza para los servicios 6G.

Este algoritmo intenta resolver el problema mediante un procesamiento de métricas relacionadas con varias dimensiones que incluyen aspectos de seguridad y de calidad de servicio, permitiendo generar una estimación del nivel de confianza. Este procesamiento de las métricas se realiza a través de la lógica difusa, la cual devuelve un resultado numérico que representa la confianza para dicho servicio. La estimación de confianza se realiza a través de una gestión de tareas de evaluación para cada servicio. A través de estas tareas se recogen las métricas necesarias para llevar a cabo la estimación. Una vez terminada, cada una de las tareas finalizan con el almacenamiento del nivel de confianza resultante en una base de datos NoSQL, además de en un contrato inteligente desplegado en una red de prueba basada en *Ethereum*.

En primer lugar, se desarrolló una primera versión del software en *Python* que sirvió como punto de partida para, a continuación, migrar el código a *Java* pensando en su posible ejecución futura desde un contrato inteligente. El software propuesto se basa en dos aplicaciones principales. La primera de ellas esta destinada a la recogida y publicación de tareas de evaluación de los diferentes servicios. Por otro lado, la segunda aplicación recoge esas tareas y elabora los cálculos necesarios mediante lógica difusa para la estimación del nivel de confianza final del servicio, almacenando ese valor en un contrato inteligente.

4.2. Diseño del Algoritmo

Una de las primeras cuestiones a abordar durante el de diseño del algoritmo LoT consistió en determinar el tipo de tecnología a emplear en su implementación. Con este propósito, se realizó un estudio sobre la definición, modelo matemático de las redes bayesianas, cadenas de Markov y lógica difusa.

Finalmente se decidió adoptar un modelo basado en lógica difusa, motivado por su enfoque acerca de los grados de pertenencia a conjuntos. Esto es una característica muy útil en este contexto, debido a que las decisiones que se deben tomar no pueden ser simplemente clasificadas como confiables o no confiables, sino que implican un nivel de incertidumbre y ambigüedad que la lógica difusa es capaz de manejar de manera efectiva.

Por otro lado, se llevó a cabo una serie investigaciones sobre artículos y otros proyectos relacionados con el cálculo de confiabilidad tanto en redes 5G como en redes 6G. Tras esta investigación, se realizó un estudio de las diferentes alternativas de dimensiones y métricas de servicios en redes que deben ser consideradas en la evaluación de confianza. En esta etapa se estudió una de las definiciones de nivel de confianza proporcionadas por el *Grupo de Análisis, Seguridad y Sistemas* (GASS) y se tomaron decisiones con respecto a las métricas que se podían emplear en el algoritmo.

Estas dimensiones son empleadas por el sistema de inferencia difuso para el cálculo de la confianza. La fuzzificación y ejecución de las reglas difusas sobre cada una estas dimensiones devuelve resultados que representan la confiabilidad entre las siguientes variables lingüisticas: *low*, *medium*, *high*. Con la defuzzificación, esta representación se convierte a un valor real que indica el grado de confianza de esa dimensión. Este procedimiento se pueden apreciar en el diagrama de secuencia de la Figura 4.1.

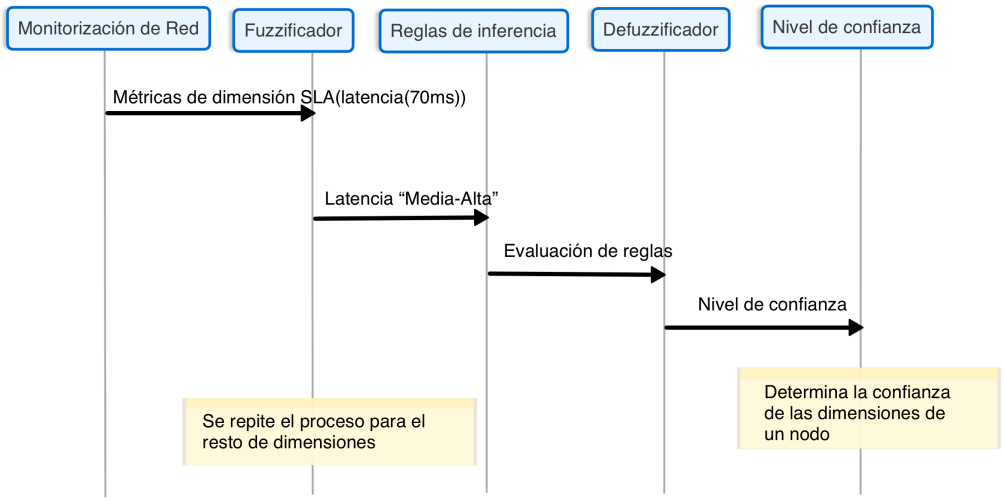

Figura 4.1: Diagrama de secuencia del cálculo de confianza de las dimensiones.

Posteriormente a la investigación de las diferentes dimensiones, se decidió evaluar tres de ellas en este trabajo:

- *Service Level Agreement* SLA: Esta dimensión está basada específicamente en dos métricas: la pérdida de paquetes (*Packet loss*) y la latencia (*Latency*). Con estas dos variables se puede cuantificar si realmente se está llevando a cabo lo estipulado en el

acuerdo de servicio. Estas dos métricas que representan la dimensión SLA se evalúan dentro de a la lógica difusa con categorías como *non acceptable*, *fair* y *acceptable* para la latencia o *bad*, *decent*, *great* para la pérdida de paquetes.

- *Cyber Threat Inteligence* (CTI): Se estudian datos de las variables de nivel de amenaza en la red (*Threat level*) y de la automatización (*Automation*) de la recopilación y análisis de información relacionada con amenazas cibernéticas. En este caso, las categorías que representan funciones de membresía de lógica difusa son: *no-risk*, *low-risk*, *medium-risk* y *high-risk* para el nivel de amenaza; mientras que para el nivel automatización son *unsupervised*, *reviewed*, y *manual*.
- *Peer-to-Peer Communication* P2P : En esta dimensión se estudian dos valores cuantificables, concretamente la información directa (*directly info*) e información indirecta (*indirectly info*). Estas variables hacen referencia a la reputación del nodo o el servicio de estudio en cuestión. Las categorías de estas métricas son: *non acceptable*, *fair*, y *acceptable* para la primera, y *bad*, *decent*, y *great* para la segunda.

El funcionamiento del algoritmo, que emplea la lógica difusa y estas métricas, se visualiza a alto nivel en la Figura 4.2 y puede describirse en dos fases:

- **Fase 1**: Se evalúan las métricas de las dimensiones SLA, CTI y P2P por medio del sistema de inferencia difuso. Esta evaluación da como resultado un número real que representa el grado de confianza para esa variable en concreto.
- **Fase 2**: Se recogen los resultados de la fase anterior y se emplean para el cálculo del valor final de confianza. Para esta segunda evaluación se toman como entradas del sistema de lógica difusa los resultados de las estimaciones de las dimensiones. En este sistema se tienen 3 categorías para cada una de las variables: *bad*, *recent*, *great*. Este nuevo modelo se ejecuta y produce como resultado un valor que representa el grado de confianza final para el servicio.

4.3. Desarrollo del Prototipo del Algoritmo LoT

Inicialmente, para la codificación del algoritmo se decidió emplear el lenguaje de programación *Python*, motivado por su versatilidad, flexibilidad y la extensa disponibilidad de bibliotecas especializadas. Estas características facilitan de manera considerable la implementación eficiente de diversos algoritmos y técnicas de inteligencia artificial. De esta manera, por ejemplo en el contexto de este proyecto, se evita la necesidad de realizar la implementación del modelo matemático de la lógica difusa desde cero.

Por parte del equipo GASS se propuso la opción de gestionar el almacenamiento persistente con una base de datos *NoSQL*, en concreto *MongoDB*. Esta decisión fue motivada por las diferentes ventajas en términos de privacidad y seguridad que ofrece *MongoDB*. Uno de estos ejemplos es la encriptación de datos en reposo y en tránsito, lo que proporciona una capa adicional de seguridad para proteger los datos sensibles contra accesos no autorizados y violaciones de privacidad, herramienta que podría ser muy útil en un futuro no muy lejano del algoritmo ya que se puede estar tratando con datos de carácter altamente sensible.

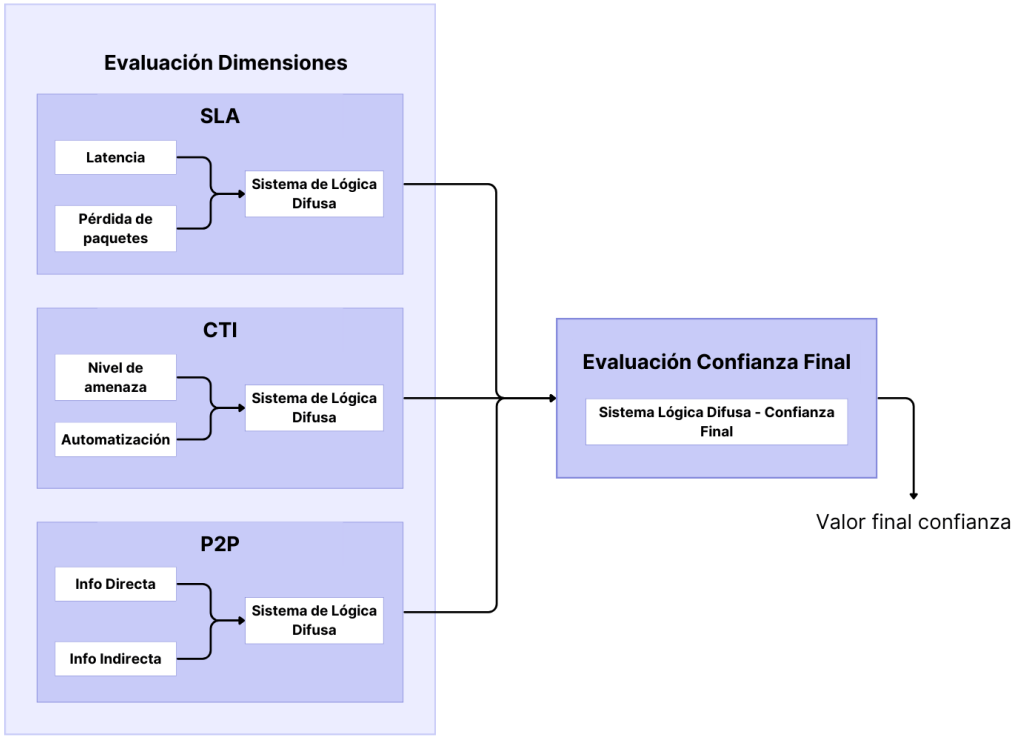

Figura 4.2: Diagrama del sistema de inferencia difuso.

El punto de partida para la implementación del algoritmo fue un código proporcionado por el grupo GASS. Esta primera versión del algoritmo calculaba la confianza para una de las dimensiones de un nodo de la red. Con este ejemplo se tomo contacto con diferentes bibliotecas que se iban a emplear en el desarrollo, haciendo un estudio y comprensión de su documentación para su uso posterior. Estas bibliotecas son: *scikit-fuzzy* para implementación de un motor de lógica difusa, *matplotlib* para la visualización de los resultados sistema de lógica difusa, *connectMongo* para la gestión del almacenamiento en base de datos MongoDB y *numpy* para generar secuencias numéricas.

Con esta versión primitiva del algoritmo se realizaron diversas adiciones al código necesarias para poder implementar la arquitectura propuesta en la Figura 4.2, permitiendo así el cálculo final de confianza de un nodo de la red. Además se incluyó la evaluación de confianza final para una lista de nodos de red proporcionada. También se realizaron algunos cambios en las reglas del sistema de inferencia difuso. Aunque en esta etapa de desarrollo las reglas difusas son ejemplos para la comprobación de la correcta ejecución del algoritmo, ya que se necesitan unos altos conocimientos de redes para poder implementar reglas difusas que simulen comportamientos reales.

4.4. Uso de Ethereum como Tecnología DLT

Una de las ideas principales para el diseño del algoritmo era la incorporación de la tecnología DLT. Este enfoque de implementación se comentó en el diseño del algoritmo haciendo alusión a sus diferentes aplicaciones y la ventaja que supone emplear tecnologías distribuidas, pero no fue hasta el momento de esta etapa de desarrollo cuando se comenzó

realmente con su investigación. El objetivo final de usar una DLT es proporcionar almacenamiento seguro y trazable para la información del algoritmo LoT.

Con el primer prototipo desarrollado en *Python*. Se hizo una exhaustiva investigación y estudio acerca de las tecnologías DLT, repasando diversos artículos científicos y documentación que exponían este tema. Tras obtener conocimientos suficientes para entender el funcionamiento y aplicaciones de las tecnologías DLT, se llevó a cabo una estudio de diferentes proyectos y artículos que empleaban estas tecnologías en contextos como los del presentado por este trabajo. Después de esta búsqueda se hizo una comparación con diversas tecnologías basadas en arquitectura DLT y su viabilidad de incorporación al proyecto. En esta comparativa se hizo un estudio de las aplicaciones, arquitectura y documentación oficial, con el objetivo de comprender su funcionamiento y escoger la más adecuada para la problemática propuesta. De las cuatro tecnologías revisadas: *Hyperledger fabric* [ABB+18], *Ethereum* [B+13], *Corda* [HB16] e *IOTA* [Pop18], se decidió emplear la plataforma *Ethereum*. Esta implementación DLT fue escogida debido a su implementación de la *blockchain* y las diferentes ventajas que esta ofrece. Destacan además, la extensa documentación disponible sobre la plataforma y la amplia oferta de implementaciones de nodos clientes de código abierto, facilitando la conexión a la red de *Ethereum* y la realización de pruebas o despliegue de contratos inteligentes.

Con la decisión de emplear *Ethereum* como tecnología DLT, se repasaron algunas implementaciones de clientes para conectar con la *blockchain*, optando por utilizar *Hyperledger besu*. Este cliente *Ethereum* de código abierto, se seleccionó debido a su compatibilidad con *Java* y su posibilidad de ejecutar nodos de la red tanto en redes públicas, como puede ser la *Mainnet* de *Ethereum*, o en redes privadas, ya sea destinadas a entornos empresariales o de pruebas, característica realmente útil en el contexto de este trabajo.

En esta etapa de desarrollo la manera de incorporación de la DLT aún estaba por concretar, las opciones eran desplegar el código del algoritmo en la *blockchain* como una *Decentralized Application* (dApp) o utilizar la *blockchain* como almacenamiento persistente de los resultados del algoritmo. Por lo tanto, con la elección de *Hyperledger besu* como nodo cliente de *Ethereum* y su compatibilidad con *Java* se procedió a la migración del código del prototipo del algoritmo LoT de *Python* a *Java*, con una visión a futuro sobre la posible implementación del código en la cadena de bloques.

4.5. Desarrollo del Algoritmo LoT en Java

Debido a la implementación de *Hyperledger Besu* para la utilización de la DLT, se realizó una migración del código primigenio de *Python* a *Java*. El uso del lenguaje *Java* trajo nuevos desafíos. El primero de ellos es la orientación a objetos, antes inexistente. En esta migración se programaron clases que representan a cada una de las dimensiones involucradas en el modelo de negocio y se implementó una arquitectura con interfaces para la especificación de los métodos y dotar así al código de abstracción y polimorfismo. Adicionalmente, se siguió manteniendo la base de datos NoSQL *MongoDB*, por lo que se desarrolló el código necesario y se integró con el resto de componentes.

En esta migración se perdió la librería de *Python* que implementaba la lógica difusa, por lo que hubo investigar la existencia de una alternativa para esta problemática en *Java*. La librería *JFuzzyLoggic* fue la finalmente seleccionada para la implementación. Esta es

una librería *Java* de código abierto robusta y flexible que permite la implementación de la lógica difusa y la visualización gráfica de los resultados obtenidos de su ejecución. Además esta librería cuenta con su propio lenguaje de programación *Fuzzy Control Language* (FCL) empleado para la declaración de los modelos de lógica difusa.

Esta librería emplea un tipo de archivo con extensión *fcl*, escrito en su propio lenguaje, donde se almacena toda la información de las reglas de inferencia difusa, variables de entrada y de salida, funciones de membresía triangular y el método de defuzzificación. Con la elección de esta librería se realizaron las implementaciones necesarias en código FCL para definir los modelos de lógica difusa de cada dimensión y la confianza final.

Para comprender el uso y la implementación de esta librería con el lenguaje FCL, se expone el siguiente ejemplo del modelo de lógica difusa para la segunda fase de evaluación de la confianza:

1. Definición de las variables de entrada/salida:

```
VAR_INPUT
        sla : REAL;
        cti : REAL;
        p2p : REAL;
END_VAR
VAR_OUTPUT // Define output variable
        Final_result : REAL;
END_VAR
```

2. Definición las funciones de membresía y el método de defuzzificación (Ejemplo con SLA y la confianza final):

```
FUZZIFY sla     // Fuzzify input variable
    TERM bad := trian 0 0 50;
    TERM decent := trian 0 50 100;
    TERM great := trian 50 100 100;
END_FUZZIFY
DEFUZZIFY Final_result  // Defuzzify output variable
    TERM low := trian 0 0 50;
    TERM medium := trian 0 50 100;
    TERM high := trian 50 100 100;
        METHOD : COG; //"Center Of Gravity" defuzzification method
        DEFAULT := 60; // Default value is 0
END_DEFUZZIFY
```

3. Definición de las reglas de inferencia difusas (se muestran algunos ejemplos de estas):

```
RULEBLOCK No1
    AND : MIN;
    RULE  1 : IF sla IS bad AND cti IS bad AND p2p IS bad
    THEN Final_result IS low;
    RULE 2 : IF sla IS bad      AND cti IS bad AND p2p IS great
    THEN Final_result IS medium;
    RULE 3 : IF sla IS great AND cti IS decent AND p2p IS
    decent THEN Final_result IS high;
END_RULEBLOCK
```

La utilización de este lenguaje FCL, siendo en archivos totalmente separados de la lógica de negocio, permite que la modificación de las reglas se realice con un alto nivel de

abstracción, por lo que es más sencillo realizar cambios si se producen nuevas necesidades, sin la obligación de modificar el código fuente de *Java*.

Esta parte del desarrollo en *Java* y FCL culmina con la implementación del algoritmo con una arquitectura como la mostrada en la Figura 4.3.

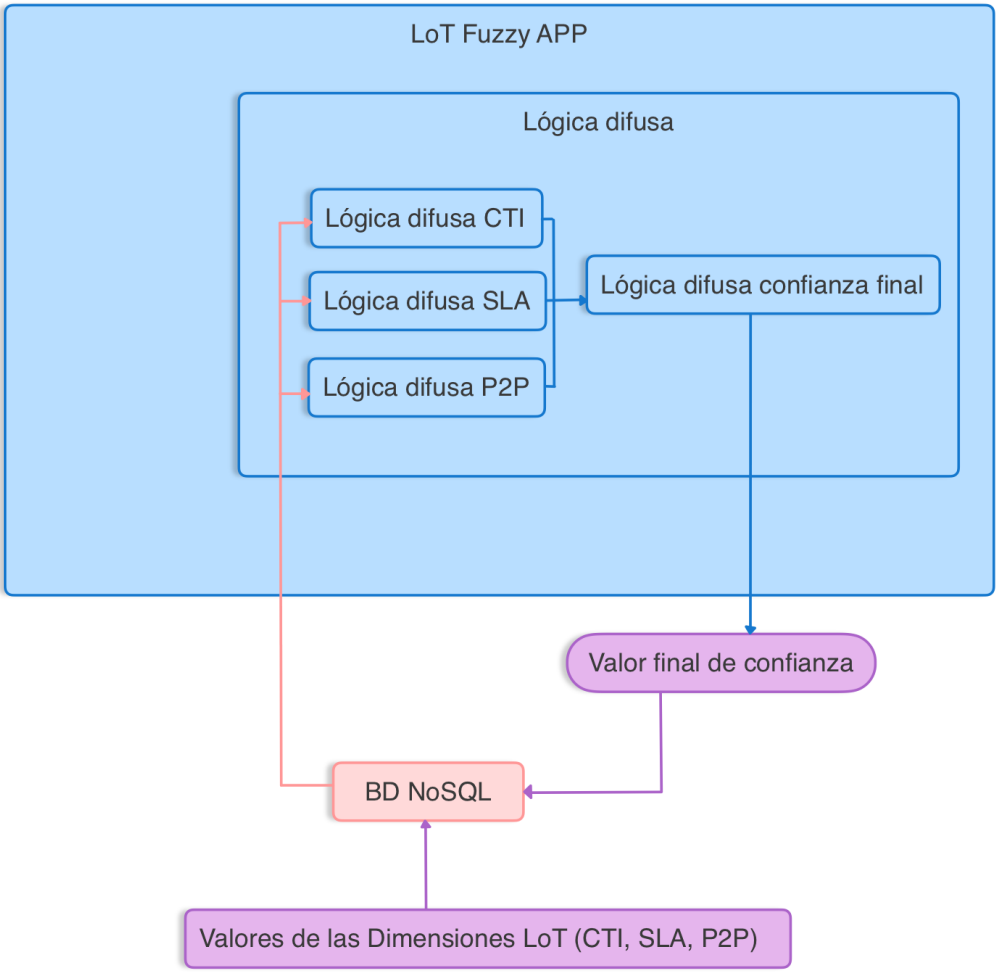

Figura 4.3: Diagrama de componentes de la primera implementación en *Java*.

4.6. Implementación de Tecnología *Blockchain* y Contratos Inteligentes

Esta sección se divide en tres subsecciones que explican el trabajo realizado acerca de la programación de contratos inteligentes e integración con la *blockhain* de *Ethereum*. Primeramente se explica la red privada de prueba que se ha configurado para simular un despliegue real del contrato inteligente LoT, a continuación se comentan las codificaciones de prueba que se realizaron con contratos inteligentes y por último se explica la implementación del contrato inteligente LoT destinado al almacenamiento de los resultados del algoritmo.

4.6.1. Configuración de Red Privada

El despliegue y el uso de contratos inteligentes en la red principal de *Ethereum* (*Mainnet*) supone un coste monetario real en ETH. Dado de que este trabajo es un proyecto de investigación se procedió a explorar diversas alternativas que puedan simular el comportamiento de la *Mainnet*, para poder realizar pruebas del despliegue y uso del contrato inteligente que sería integrado en el algoritmo LoT.

Las alternativas encontradas fueron las redes públicas de prueba llamadas *testnets* y las redes privadas. Las *testnets* en el contexto de *Ethereum* son redes que replican el software y la funcionalidad de la *Mainnet* pero están diseñadas específicamente para propósitos de prueba y desarrollo. Este tipo de redes permiten desplegar, probar y depurar contratos inteligentes sin usar fondos reales de una cuenta de *Ethereum*. Por otro lado, las redes privadas requieren de una configuración inicial, asignando tipos de protocolo de consenso, cuentas y límites de tarifas de *gas* entre otros, pudiendo servir tanto para entornos empresariales como para entornos de prueba.

En este trabajo se decidió configurar una red privada por dos motivos: El primero de ellos son las limitaciones encontradas al intentar emplear ciertas *testnets*, las cuales dificultaban la obtención de ETH, necesario para cubrir las tarifas de *gas*. Algunas de estas *testnets* que se probaron fueron *Goerli* y *Sepolia*. El segundo motivo es la gran disponibilidad de documentación que ofrece *Hyperledger Besu* para la configuración de redes privadas.

La red privada que se ha configurado para el proyecto se basa en el protocolo de consenso de *Proof of Authority* (PoA) IBFT 2.0. En este tipo de *blockchain* los validadores tienen la responsabilidad de verificar transacciones y bloques, alternándose entre ellos por turnos para la creación de los siguientes bloques.

La configuración de la red privada se implementó con una estructura de 4 nodos validadores en directorios dentro de un único ordenador. Esta disposición es necesaria para el almacenamiento de los datos de la *blockchain*. Adicionalmente en otro directorio, se redactó un archivo de configuración en extensión *JavaScript Object Notation* (JSON) con diversos parámetros necesarios para el correcto funcionamiento de la red, como por ejemplo el *gasLimit*, que representa el límite máximo de gas que se puede consumir en cada bloque de la cadena. Además con el uso de la línea de comandos proporcionada por *Hyperledger besu* se generaron las claves de cada uno de los nodos, las cuales son necesarias para el despliegue o interacción con contratos inteligentes a través de alguno de ellos.

Para poder hacer funcionar la cadena de bloques, se ejecuta cada uno de los nodos en diferentes terminales con las opciones ofrecidas por la *Command Line Interface* (CLI) de *Hyperledger besu* y además empleando diferentes puertos *Transmission Control Protocol* (TCP) para cada uno de ellos.

4.6.2. Prueba de Contratos Inteligentes

En el presente trabajo, se han incorporado las tecnologías *blockchain* y contratos inteligentes a través de la plataforma *Ethereum* y el despliegue de nodos clientes con *Hyperledger besu* en una red privada. Primeramente, para incorporar estas tecnologías se llevaron a cabo una serie de reuniones, para concretar el enfoque del uso de los contratos inteligentes. Además se realizó una investigación exhaustiva de la documentación oficial de *Ethereum* para comprender el alcance y las capacidades que pueden tener los contratos inteligentes en la red.

El análisis de documentación en relación con los contratos inteligentes no solo abarcó documentación de la plataforma, de la cual se obtuvieron conocimientos sobre el funcionamiento y coste de estos, si no que también implicó un estudio del lenguaje de programación de los contratos inteligentes en *Ethereum*, *Solidity*. Esta revisión de la documentación oficial del lenguaje *Solidity* sirvió como base para la decisión de la aplicación de contratos inteligentes en el algoritmo LoT.

Para las primeras pruebas con contratos inteligentes, programados con *Solidity*, se empleó el *Integrated Development Environment* (IDE) *online Remix*. Este IDE es un entorno web que permite codificar, probar y depurar contratos inteligentes en *Ethereum* y otras plataformas *blockchain* compatibles con *Solidity*. Con el uso de esta herramienta se programaron una serie de contratos inteligentes de prueba que ayudaron a comprender el funcionamiento del despliegue, transacciones asociadas, llamadas a funciones y el almacenamiento de variables, tanto persistente como en memoria dinámica. Además, se pudo comprobar el coste en términos de *gas*, derivado tanto del despliegue como de la llamada a funciones del contrato.

Tras revisar la documentación y realizar diferentes pruebas de programación de contratos inteligentes, se pudieron comprobar bastantes limitaciones tanto del lenguaje como de la plataforma en sí. Una de estas limitaciones que se encontraron era la imposibilidad de implementar el motor de lógica difusa, ya que no se encontraron librerías para este lenguaje que facilitaran su implementación. Por tanto, se decidió finalmente utilizar la tecnología *blockchain* como almacenamiento persistente del valor de confianza calculado por el algoritmo. Este almacenamiento sería implementado a través de un contrato inteligente desplegado en la red *Ethereum*, garantizando así la trazabilidad y la seguridad de los resultados del algoritmo.

4.6.3. Implementación del Contrato Inteligente LoT

Con el entorno de prueba de la red privada, donde poder desplegar y depurar el código del contrato inteligente y después de realizar las pruebas pertinentes con *Remix*, se procedió con la investigación de como integrar el algoritmo LoT desarrollado en *Java* con un contrato inteligente. Para esta integración se ha empleado la biblioteca de *Java Web3j*. Se eligió esta herramienta porque simplifica el proceso de despliegue de contratos inteligentes, permitiendo a los desarrolladores desplegar y gestionar contratos en la red sin necesidad de lidiar directamente con la complejidad de las transacciones en la *blockchain*. Al ser una librería *Java*, *Web3j* es compatible con cualquier plataforma que soporte la *JVM*. La librería *Web3j*, por otra parte, maneja la criptografía necesaria para interactuar con la *blockchain* de manera segura, gestionando claves privadas y firmando transacciones, lo que ayuda a proteger los activos y datos de la aplicación.

Para la integración de esta librería se estudió la documentación de la misma con el fin de comprender su funcionamiento y componentes que hacían posible esa interacción entre la cadena de bloques y la aplicación *Java*, además de realizar diversas pruebas de implementación.

Las primeras pruebas de integración de los dos sistemas, el algoritmo LoT y la *blockchain*, se realizaron con una implementación totalmente nueva. Esta nueva aplicación se implementó como un código totalmente ajeno al algoritmo LoT. En esta aplicación se programaron diversos contratos inteligentes y una aplicación en *Java* que empleara *Web3j* para el despliegue e interacción de los mismos. Esta codificación se llevó a cabo con la finalidad de comprobar el correcto funcionamiento tanto de *Web3j* como de la red privada configurada anteriormente. En estas pruebas se realizaron diversos despliegues e interacciones con contratos inteligentes a través de uno de los nodos de la red privada de prueba.

Tras la realización de las pruebas y la obtención del conocimiento necesario del lenguaje *Solidity* como de la biblioteca *Web3j*, se programó el código del contrato inteligente que almacena de forma persitente en la *blockchain* los valores de confianza resultantes del algoritmo LoT. En esta etapa de desarrollo, la evaluación de confianza se realizaba para diversos nodos de la red, por lo que el almacenamiento del nivel de confianza se realiza en el contrato inteligente con un mapa donde cada número que identifica a un nodo de la red 6G se utiliza como clave, mientras que la lista de sus respectivos valores de confianza se almacena como el valor correspondiente.

En este contrato inteligente se han implementado varias funciones adicionales además de aquella encargada de almacenar los valores de confianza. Entre estas funciones se incluye una para verificar el número de valores de confianza asociados a un nodo, así como otra destinada a recuperar los n últimos valores de confianza almacenados para dicho nodo.

4.7. Implementación Final del Algoritmo LoT

Para la implementación de esta última versión se llevaron a cabo varias reuniones con el propósito de determinar el enfoque del algoritmo. En estas reuniones con el equipo GASS se decidió realizar diversas implementaciones con el fin de mejorar el comportamiento del algoritmo LoT.

La primera de estas implementaciones consistía en la diferenciación de dos modalidades de ejecución del algoritmo: Una ejecución periódica (*Time Driven*) y una ejecución por eventos (*Event Driven*). Para el desarrollo de estas dos modalidades se decidió emplear la biblioteca *Kafka* en el código *Java*. *Kafka* es una plataforma de transmisión de datos de código abierto que se utiliza para la construcción de sistemas de mensajería en tiempo real y procesamiento de datos. En este trabajo se usa como servidor de tareas, al cual llegan tareas de evaluación con los datos de las métricas y dimensiones. Para la implementación del servidor *Kafka* y las dos modalidades de ejecución, se programó una nueva aplicación encargada de enviar las tareas de evaluación al servidor de *Kafka*. A esta nueva aplicación se le ha denominado *Data Collector APP*. Parte de la implementación de esta aplicación la desarrolló el grupo de investigación GASS. Además el equipo GASS realizó también la implementación del consumidor de tareas del servidor *Kafka* en el software de evaluación de confianza *LoT Fuzzy APP*. Esta parte del código recoge la tarea del servidor *Kafka* almacenada en un formato JSON y la convierte a una clase *Java* para poder proceder con

su evaluación.

A continuación se explican las dos modalidades de ejecución comentadas anteriormente:

- **Ejecución periódica**: Esta modalidad se basa en un envío de tareas al servidor de *Kafka* de manera periódica cada cierto tiempo determinado. Para realizar esta simulación de un entorno real, se implementó una generación de valores aleatorios para los datos de las métricas en las dimensiones. Adicionalmente se codificó un componente que cada cierto tiempo publicara en *Kafka* una tarea de evaluación para un servicio en concreto. Para poder realizar esta implementación se estudió y revisó la documentación oficial de *Kafka* para poder comprender el funcionamiento de la publicación de mensajes en su servidor. El funcionamiento de esta modalidad desde el envío de la tarea hasta la evaluación de confianza se puede ver en el diagrama de secuencia mostrado en la Figura 4.4.

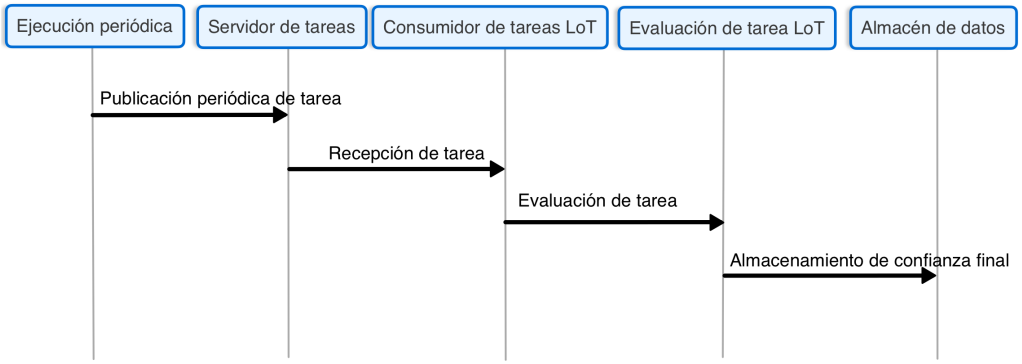

Figura 4.4: Diagrama de secuencia del modelo de ejecución periódica.

- **Ejecución por eventos**: El modelo de propuesto se fundamenta en la posibilidad de que ocurran ciertos eventos lo suficientemente significativos como para tener la necesidad de evaluar la confianza de un servicio sin esperar a la evaluación periódica. Este modelo funciona con una *Representational State Transfer Application Programming Interface (API REST)*, la cual recibe peticiones HTTP a un puerto determinado con la tarea de evaluación. Este tipo de ejecución fue programada por el equipo de investigación GASS.

Además de lo comentado sobre la ejecución periódica y la ejecución por eventos, el código previamente implementado del algoritmo LoT evaluaba sistemáticamente todos los nodos de la red, sin considerar la llegada de tareas. Por esta razón se implementó la lógica necesaria para la evaluación de confianza de tareas específicas que ingresaban al sistema mediante el consumidor de *Kafka* agregado al código. Esta implementación se puede observar a alto nivel de manera más visual con el diagrama de secuencia mostrado en la Figura 4.5.

Otra de las implementaciones llevadas a cabo fue la evaluación de servicios E2E en lugar de nodos de la red. Para esta codificación se realizaron las adiciones y cambios necesarios tanto en el código *Java* como en el contrato inteligente programado en *Solidity*. Con esta nueva implementación se sigue manteniendo todo el sistema de inferencia difuso implicado en el algoritmo, sin embargo, la lógica de almacenamiento de datos y la evaluación de tareas tiene algunos cambios ya que se emplean identificadores de servicios como cadenas

de caracteres, en lugar de un número entero que identificaba al nodo de red 6G. Por lo tanto, esto permite asociar valores de confianza a servicios E2E.

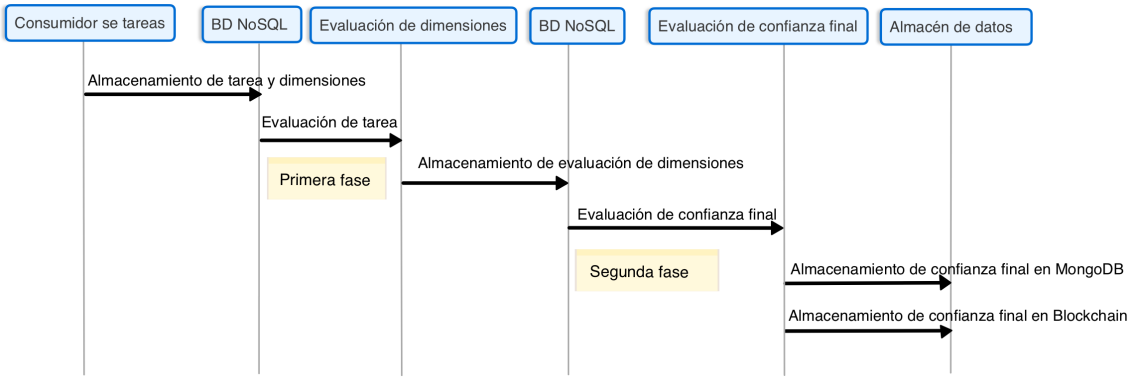

Figura 4.5: Diagrama de secuencia de la evaluación de una tarea.

Por último, para finalizar la implementación del algoritmo, se llevó a cabo la integración del código del contrato inteligente en la *LoT Fuzzy APP*. Esta integración se hizo de manera que la aplicación únicamente interactuara con el contrato inteligente llamando a funciones del mismo. Para la integración del código, las llamadas a las funciones del contrato que almacenan los resultados, se incluyeron directamente en el flujo de almacenamiento de datos junto con las llamadas de *MongoDB* como si de una base de datos tradicional se tratara. Esta etapa de implementación termina la versión final del algoritmo LoT. La arquitectura final del software propuesto en este trabajo, se puede observar en la Figura 4.7. Además, la estimación final de confianza de esta versión definitiva se puede apreciar, desde la segunda fase, con el diagrama de secuencia de la Figura 4.6.

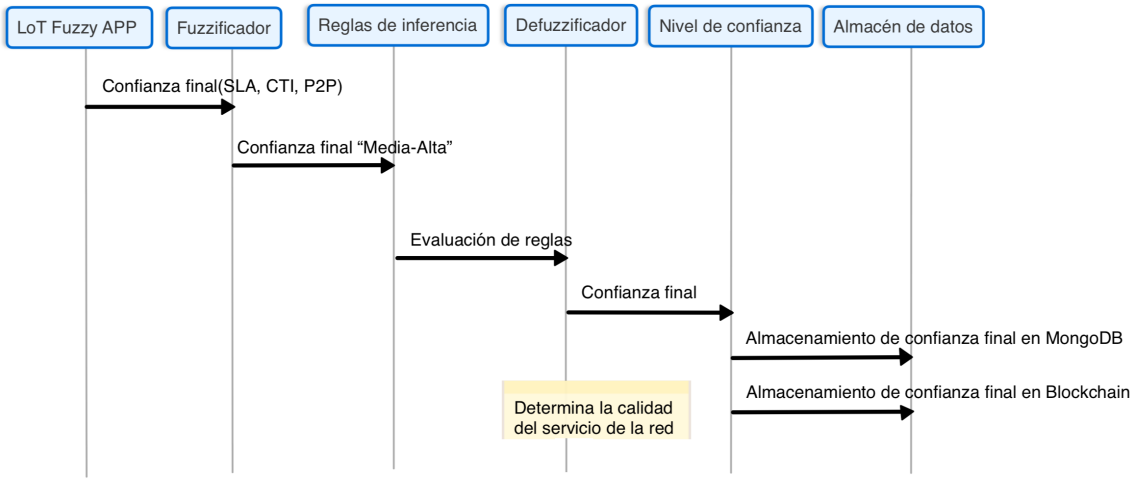

Figura 4.6: Diagrama de secuencia de la segunda fase de evaluación de confianza.

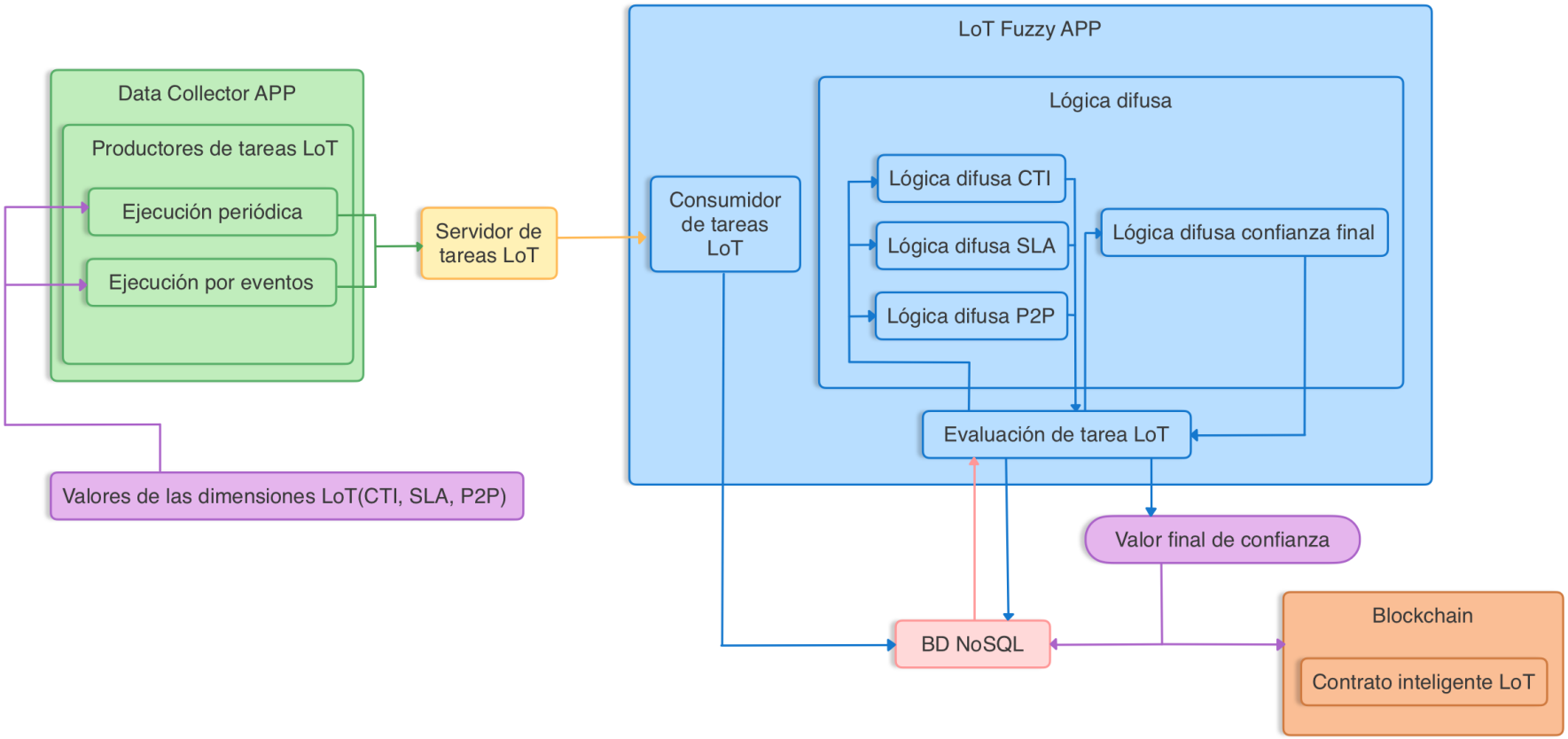

Figura 4.7: Diagrama de componentes Algoritmo LoT.

Capítulo 5

Experimentos y Resultados

En el próximo capítulo se presentarán los experimentos realizados con el fin de validar la eficacia del algoritmo propuesto. La experimentación llevada a cabo se puede dividir en tres partes, experimentación con el prototipo del algoritmo en *Python*, experimentación con la versión final del algoritmo y experimentación de contratos inteligentes. Cada una de estas experimentaciones han sido piezas clave en el desarrollo del trabajo. Gracias a ellas, no solo se ha verificado la eficacia del algoritmo, sino que también han contribuido en la toma de decisiones y en las modificaciones del algoritmo a lo largo del desarrollo.

5.1. Metodología de Evaluación y Objetivos de la Experimentación

La fase de experimentación se ha llevado a cabo en gran medida de manera concurrente con el desarrollo, con el objetivo de mejorar el código y su eficacia. Para llevar a cabo la experimentación, los resultados de las evaluaciones se comparan con las entradas proporcionadas para verificar el comportamiento del algoritmo. De este modo, se realiza una aproximación al resultado esperado con las entradas dadas y se contrasta con el resultado real proporcionado por el algoritmo.

La evaluación de los resultados del algoritmo ha dependido en todo momento del modelo de lógica difusa implementado en el código. Cabe destacar que al emplear lógica difusa, esta requiere de unas reglas de inferencia que se declaran para conseguir unos resultados esperados. Para poder simular entornos reales es necesario redactar unas reglas lo más concretas y coherentes posibles, estudiando en detalle las métricas que están evaluando y como deberían de devolver los resultados para realizar la defuzzificación con la que se obtiene el valor final de confianza. Para realizar una redacción de estas reglas lo mas fielmente posible a la realidad, se necesitan conocimientos avanzados en redes, por lo que estas reglas deberían ser escritas por los administradores de la red. Debido a las observaciones realizadas, se advierte que existe margen para mejorar las reglas de inferencia y pueden no representar la realidad al 100 %, sin embargo, el grupo de investigación GASS proporcionó una serie de reglas para cada una de las dimensiones que han sido útiles para comprobar la eficacia del algoritmo.

Por otro lado también se realizaron pruebas con contratos inteligentes con el fin de comprobar el comportamiento de estos y observar su funcionamiento con su integración al algoritmo. Esta experimentación se ha realizado para comprobar la viabilidad de

incorporar esta tecnología en un entorno real. De esta manera se pudo hacer una estimación de su rendimiento, además de sentar una base para posibles mejoras futuras de la integración de estas tecnologías.

5.2. Experimentación con el Prototipo de *Python*

En esta fase de la experimentación se llevaron a cabo una serie de pruebas para evaluar el comportamiento del algoritmo implementado en *Python*. En estas pruebas se comprobaron por medio de gráficas en *Python* como eran los resultados que se podían obtener con diferentes entradas. Para esta experimentación se empleó un *dataset* proporcionado por el grupo de investigación GASS. En esta base de datos se podían encontrar datos de diferentes parámetros de red aunque el algoritmo propuesto evalúa sólo las referidas a las tres dimensiones empleadas para el cálculo de confianza: CTI, SLA, P2P.

Primeramente, para establecer un contexto sobre estas pruebas, la manera de evaluar los resultados era a través de gráficas donde se pueden observar las funciones de membresía de la métrica de estudio. Además, en la gráfica donde se muestra el resultado, se observan tanto las funciones de membresía resultantes como el valor real del resultado, el cual es un valor de entre 0 y 100 que representa el grado o porcentaje de confianza. Se puede visualizar un ejemplo de estas gráficas en las Figuras 5.1, 5.2, 5.3, las cuales son un ejemplo de multitud de pruebas que se realizaron con el código del algoritmo, donde se puede observar en la línea negra de las figuras, el resultado de las evaluaciones de confianza y el grado de pertenencia de las funciones de membresía. Con el empleo de estas gráficas se puede identificar el grado de confianza de una manera visual y sencilla.

Las primeras pruebas eran dedicadas a la mejora del sistema de lógica difusa, estas primeras aproximaciones del nivel de confianza se hicieron con unas reglas de inferencia básicas propuestas por el grupo GASS. Estas pruebas consistían en la evaluación de una lista de nodos y con la ayuda de la herramienta *debug* se comprobaba la actuación del código.

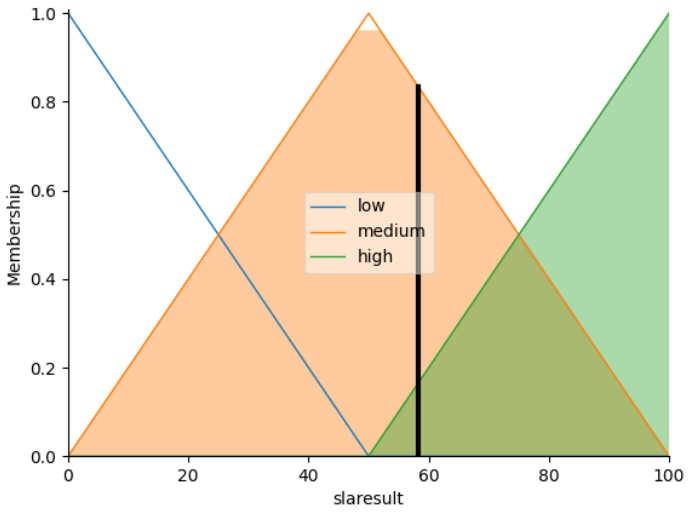

Figura 5.1: Gráfica del resultado de confianza de *SLA*.

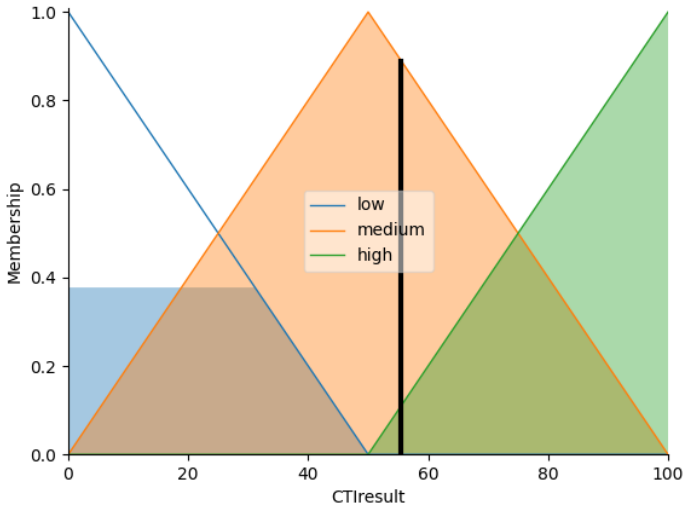

Figura 5.2: Gráfica del resultado de confianza de *CTI*.

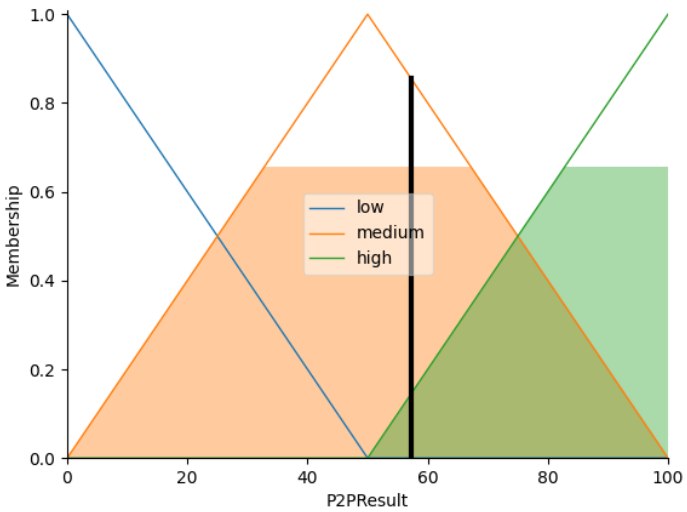

Figura 5.3: Gráfica del resultado de confianza *P2P*.

Durante las pruebas de experimentación de este prototipo, los resultados revelaron niveles de confianza siempre comprendidos en el rango del 40 % al 60 %. Estos resultados se compusieron con entradas de la base de datos de simulación, en la cual se podían encontrar datos de diversa índole que pudieran dar resultados muy diferentes. Por lo tanto se decidió realizar algunos ajustes en el código para la migración en *Java*, con el objetivo de conseguir un nivel de confianza mas ajustado a los datos de entrada, de tal manera que no solo se obtuvieran resultados entre 40 % y 60 %. Estos resultados sirvieron más adelante para ajustes en la lógica difusa cuando fue migrado el código a *Java* y algunos cambios en las reglas de inferencia.

En definitiva, debido a la implementación básica de las reglas de inferencia en este prototipo, los datos estudiados de los resultados no permitieron llegar a una decisión clara sobre la correcta estimación del nivel de confianza. Sin embargo si se pudo observar un cierto comportamiento que se reflejaba en los resultados. Los nodos evaluados con datos

de entrada que reflejaban un mal servicio de red, obtenían menores puntuaciones que los nodos que tenían un mejores métricas.

5.3. Experimentación con la Versión Final de *Java*

En esta fase de experimentación, el software implementa las dos modalidades de ejecución, la eventual y la periódica. Para simular la entrada de tareas de evaluación al sistema por el modelo de ejecución por eventos se han realizado envíos de los datos a través de la consola de comandos con peticiones HTTP a la *API REST* del aplicación *Data Collector*. Por otro lado, para simular la ejecución periódica se implementó un método que aleatoriamente generara valores para cada una de las métricas comprendidos siempre entre unos valores simulando un entorno real, de este manera se pudo comprobar el correcto funcionamiento de este modelo de ejecución. En las experimentaciones, el modelo de ejecución por eventos se empleó específicamente para comprobar los resultados frente a las entradas de datos encontradas en el *dataset* proporcionado anteriormente por el grupo GASS. Por otro lado, el modelo periódico ha servido para el estudio de los resultados de manera masiva, con envíos de tareas de evaluación en períodos cortos de tiempo y comprobación de la estimación frente a los datos de entrada.

Las pruebas que se realizaron en esta fase iban acompañadas de gráficas, gracias a la biblioteca *JFuzzyLogic* se pudieron representar gráficamente los resultados para observar el comportamiento del algoritmo. Además estas gráficas sirvieron para hacer una comparación entre la implementación de *Java* y la de *Python*, ya que en el caso de java las funciones de membresía eran declaradas manualmente con el lenguaje FCL, mientras que en *Python* se realizaban automáticamente gracias a la biblioteca *skfuzzy*. Uno de estos ejemplos de ejecución se puede observar en la Figura 5.4, la cual muestra las gráficas resultado del cálculo del nivel de confianza de las tres dimensiones en la primera fase de evaluación.

En esta fase de experimentación se revisaron multitud de resultados, los cuales procedían tanto de generación aleatoria como de envíos de tareas eventuales de forma manual. Estas pruebas se llevaron a cabo con el objetivo de determinar la utilidad del algoritmo para el cálculo de la confianza. A pesar de emplear reglas de inferencia difusa básicas que no representan fielmente la realidad, se sacaron varias conclusiones con respecto al modelo de lógica difusa y el cálculo del nivel de confianza.

El algoritmo exhibe resultados aceptables en lo que respecta el valor de confianza del servicio. El modelo propuesto demuestra la capacidad de generar una estimación del nivel de confianza apropiado con respecto a los servicios que evalúa. El algoritmo muestra una relación proporcional entre resultado y la calidad de los datos de entrada, es decir, ante la presencia de datos deficientes, se observa una disminución en el nivel de confianza generado, mientras que la utilización de datos de alta calidad resulta en un aumento correspondiente en dicho nivel de confianza. Como ejemplo de ello se muestra la Tabla 5.1, con tres tareas de evaluadas para el servicio *"TestingE2EService"*, en la tabla se puede observar el resultado de la confianza final para cada tarea con respecto a los valores iniciales de las métricas de cada dimensión.

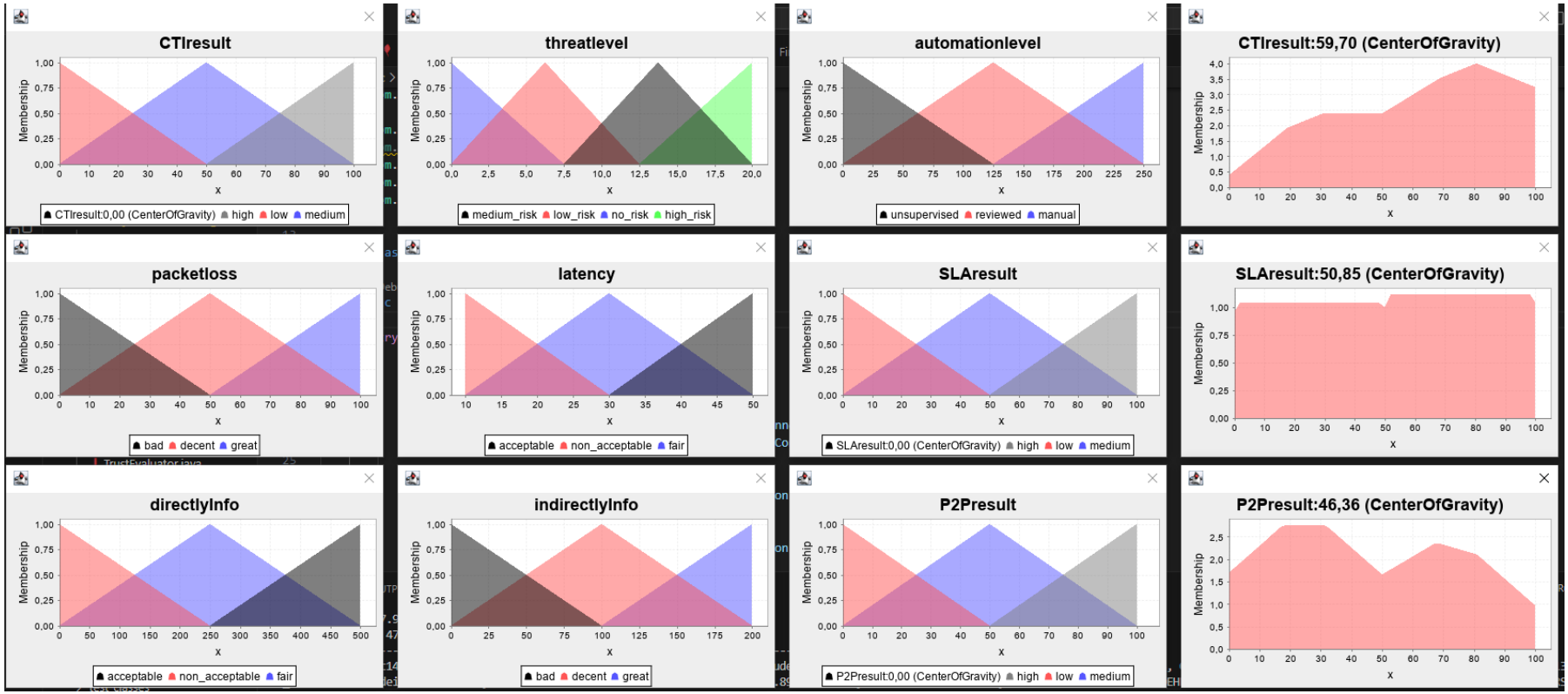

Figura 5.4: Gráficas resultado de evaluación de las dimensiones de un servicio.

Tabla 5.1: Tabla con resultados de la evaluación de tres tareas para un servicio

SERVICE	TASK	DIMENSIONS INPUT	DIMENSIONS INPUT VALUE	FINAL TRUST
TestingE2EService	663e49e4dba8e14e11aadcd2	threat_level	8,167538741	52,63118888
TestingE2EService	663e49e4dba8e14e11aadcd2	automation_level	166,4794304	
TestingE2EService	663e49e4dba8e14e11aadcd2	packet_loss	90,715527000	
TestingE2EService	663e49e4dba8e14e11aadcd2	latency	22,710730300	
TestingE2EService	663e49e4dba8e14e11aadcd2	directly_info	230,080477200	
TestingE2EService	663e49e4dba8e14e11aadcd2	indirectly_info	50,064751170	
TestingE2EService	663e4a20dba8e14e11aadcdd	threat_level	5,174678223	81,94858825
TestingE2EService	663e4a20dba8e14e11aadcdd	automation_level	96,333862040	
TestingE2EService	663e4a20dba8e14e11aadcdd	packet_loss	2,155951073	
TestingE2EService	663e4a20dba8e14e11aadcdd	latency	20,994252260	
TestingE2EService	663e4a20dba8e14e11aadcdd	directly_info	290,772084800	
TestingE2EService	663e4a20dba8e14e11aadcdd	indirectly_info	60,834780040	
TestingE2EService	663e4ad4dba8e14e11aadcfe	threat_level	14,649288160	28,61814942
TestingE2EService	663e4ad4dba8e14e11aadcfe	automation_level	102,327207400	
TestingE2EService	663e4ad4dba8e14e11aadcfe	packet_loss	25,606644530	
TestingE2EService	663e4ad4dba8e14e11aadcfe	latency	45,895358620	
TestingE2EService	663e4ad4dba8e14e11aadcfe	directly_info	155,765266100	
TestingE2EService	663e4ad4dba8e14e11aadcfe	indirectly_info	47,122363000	

5.4. Experimentación con Contratos Inteligentes

Además de la fase de experimentación del algoritmo para el estudio de sus resultados, cabe destacar las diferentes pruebas que se realizaron con la cadena de bloques de *Ethereum* y los contratos inteligentes.

Estas experimentaciones se llevaron a cabo para estimar los costes y tiempos del despliegue de contrato, llamada a funciones y operaciones lectura y escritura en la *blockchain*. Debido al uso de una red privada que implementa el mecanismo de consenso (PoA) frente al protocolo de la *Mainnet* de *Ethereum* (PoS), los resultados obtenidos no son del todo claros con como sería un despliegue real del contrato en *Ethereum*. A pesar de ello, a través del análisis de los gastos de ETH asociados a la cuenta utilizada para interactuar con el contrato, se pudo determinar que el costo de acceso a la *blockchain* o la ejecución del contrato se situaría en torno a 0,00019 ETH por cada diez mil accesos. Sin embargo, la tecnología *blockchain* esta sujeta fluctuaciones de las tarifas de transacciones o complejidad de operaciones entre otros. Cabe destacar que la evaluación se ha llevado siempre a cabo dentro del marco de la red privada empleada para las pruebas, por lo que en un escenario real es posible que las tarifas y los gastos de ETH difieran.

Adicionalmente se comprobaron diferentes tiempos a la hora de realizar interacciones con el contrato, dado que se necesita la realización del protocolo de consenso, estas operaciones pueden llegar a ser costosas en términos de tiempo. En el presente trabajo, se llegó a verificar que los tiempos de inserción de valores de confianza en el contrato inteligente alcanzaron hasta 5 segundos. Es importante señalar que estas mediciones pueden estar sesgadas debido a la utilización de una red privada y, sobre todo, a la concentración de los 4 nodos validadores en un único ordenador.

Capítulo 6

Conclusiones y Trabajo Futuro

En este capitulo se exponen las conclusiones derivadas del desarrollo e implementación del software propuesto y del análisis de los resultados obtenidos a lo largo de este trabajo. Además, se discuten las implicaciones prácticas proporcionando una visión crítica de algunas limitaciones de la implementación del proyecto. Finalmente, se proponen líneas de investigación futura que podrían ampliar el trabajo presente, sugiriendo posibles áreas de mejora.

6.1. Conclusiones

La llegada de las redes 6G traerá consigo una serie de desafíos y oportunidades significativas. En consecuencia, se requerirán nuevas arquitecturas y paradigmas que puedan afrontar con éxito estas problemáticas surgidas de una evolución hacia la sexta generación de redes móviles. Una de las cuestiones críticas en este contexto radica en la necesidad de establecer una arquitectura de red que garantice la seguridad y confiabilidad. El desarrollo de un algoritmo de estimación de confianza para servicios de redes 6G, representa un paso significativo hacia la construcción de ese ecosistema de comunicaciones inalámbricas más seguro y confiable para el futuro. El trabajo de investigación y desarrollo de esta herramienta a conducido a las siguientes conclusiones:

- La estimación de la confianza de servicios de la red es uno de los desafíos mas importantes a resolver con la implantación de la sexta generación de redes móviles. Esta evaluación de confianza puede permitir un entorno más seguro y privado para el futuro IoT donde se encontrará un análisis masivo de datos de los distintos dispositivos y usuarios conectados.

- La lógica difusa se muestra como un modelo apropiado para abordar problemáticas similares a la expuesta en el presente trabajo. La capacidad de la lógica difusa para manejar la imprecisión es excelente y especialmente en la evaluación de confianza puede ser un modelo realmente útil. La lógica difusa permite tomar en cuenta diferentes factores que interfieren en la confianza y a través de reglas difusas y variables lingüísticas se puede conseguir un nivel de evaluación más complejo y realista. Sin embargo, se ha concretado debido en gran parte al estudio de los resultados, que el modelo de lógica difusa debe de llevar un gran estudio del funcionamiento de las redes móviles. Para poder obtener resultados excepcionales

y concluyentes es esencial redactar unas reglas de inferencia lo suficientemente fieles al comportamiento de la red en un escenario real.

- La incorporación de las DLT y los contratos inteligentes tanto en la arquitectura de red como en el proceso de evaluación de confianza, podría representar un enfoque altamente prometedor para abordar la cuestión de la confianza en las redes 6G. Muchos estudios proponen este tipo de tecnología para ayudar a la mejora de integridad, análisis y seguridad de los datos. En este trabajo se refuerza esta afirmación, mediante la utilización de contratos inteligentes como medio para almacenar los valores de confianza. Con la integración de estas tecnologías, el sistema se volvería menos susceptible a ataques cibernéticos dado que los resultados estarían descentralizados y disponibles públicamente en una cadena de bloques. Sin embargo, el empleo de contratos inteligentes podría conllevar una demora en la evaluación de confianza, debido a que las transacciones para las interacciones con el contrato inteligente, deben someterse a los procesos de consenso de las cadenas de bloques antes de su validación.

- Los datos estudiados de la experimentación determinan que la solución propuesta en este trabajo demuestra ser una herramienta suficientemente eficaz para estimar el nivel de confianza de los servicios de las redes 6G. En este contexto, la selección de métricas destinadas a la evaluación de la red constituye un aspecto muy relevante para la estimación adecuada de diversos tipos de servicios. Al analizar métricas clave como la latencia o el nivel de amenaza, el algoritmo LoT proporciona una evaluación consecuente de la fiabilidad de los servicios, lo que permitiría tomar decisiones informadas para garantizar una interacción óptima en la red. A pesar de ello se conocen las limitaciones del algoritmo, siendo recomendable realizar implementaciones futuras con el objetivo de mejorar el trabajo presente, como se comenta en la siguiente sección.

6.2. Trabajo Futuro

En esta sección se proponen las siguientes áreas de investigación e implementación futuras:

- Implementación de la lógica del algoritmo un contrato inteligente. Esta integración se propone de tal manera que el código del algoritmo se ejecute dentro de una DLT. Este enfoque permite aumentar la confianza en los resultados obtenidos y la seguridad, integridad y trazabilidad tanto de las variables evaluadas como de los resultados, gracias a la descentralización.

- Implementación de un elemento de almacenamiento asíncrono para lidiar con las demoras de llamadas a contratos inteligentes. El desarrollo de este componente permitiría que la aplicación continúe funcionando y procesando datos mientras espera la confirmación de las transacciones en la cadena de bloques. Esto puede reducir significativamente los tiempos de espera percibidos y mejorar la eficiencia general del sistema.

- Estudio y adición de otras dimensiones al algoritmo. En el algoritmo propuesto se emplean métricas que representan tres dimensiones específicas. Sin embargo, la incorporación de diversas métricas, tales como un índice de privacidad, podría

potenciar los resultados obtenidos. Además, la incorporación de nuevas dimensiones podría generar la necesidad de implementar un mecanismo de anonimización de los datos. Con este elemento, se podría garantizar la preservación de la privacidad y confidencialidad de la información de los datos contenidos en estas métricas.

- Adición de pesos en las variables para conseguir la evaluación diferenciando tipos de servicios y redes. La integración de un sistema de pesos en el proceso de evaluación de confianza posibilita la asignación de más o menos importancia a ciertas dimensiones específicas en comparación con otras. Este enfoque hace posible la estimación de confianza en diferentes escenarios donde pueden ser más relevante unas métricas frente a otras, como por ejemplo un servicio que trabaje con datos altamente sensibles, la dimensión relacionada con ataques cibernéticos podría tener mas relevancia.
- Adición de una media histórica de valores de confianza. El algoritmo propuesto, durante su ejecución espera la recepción de todos los datos de las métricas para llevar a cabo la evaluación de confianza. No obstante, en situaciones reales, es posible que algunas de estas métricas lleguen al algoritmo desprovistas de datos. Por esta razón, la implementación de la lógica necesaria para una media ponderada sobre n valores antiguos de confianza, permite la evaluación correcta de tareas que se reciben sin valores para algunas dimensiones.

Capítulo 7

Contribuciones

En este capítulo se detalla la participación individual de cada integrante del equipo. Es relevante señalar que ambos miembros realizaron la mayoría de las actividades conjuntamente, incluyendo la investigación de artículos, estudio de documentación, diseño del código e implementación, experimentación y redacción de la memoria. La colaboración en equipo fue fundamental y se ejecutó con éxito, permitiendo que todos los miembros adquirieran significativos conocimientos y contribuyeran en todas las etapas del proyecto.

7.1. Jesús García Peña

Las primeras tomas de contacto con el trabajo se realizaron a través de una serie de reuniones didácticas con el equipo GASS. En estas reuniones educativas se adquirieron conocimientos básicos sobre el concepto del nivel de confianza en redes y conceptos relacionados sobre las redes 6G y el IoT. Adicionalmente en estas reuniones se aprendieron conceptos sobre la búsqueda de artículos científicos, específicamente haciendo hincapié en la herramienta *Google Scholar*. Con una base de conocimientos adquiridos me dispuse a investigar artículos científicos relacionados con las redes 5G y 6G. Prácticamente todos los estudios estaban relacionados con el tema que concierne al proyecto, la evaluación de confianza y el establecimiento de redes seguras y confiables. Aunque, además se revisaron algunos artículos relacionados con el IoT con la intención de comprender el término. En esta etapa investigué y estudié diferentes artículos con el fin de obtener suficientes conocimientos que permitieran el desarrollo del trabajo. En esta investigación se tomaron diferentes apuntes de los artículos revisados para su posterior consulta en caso de que fuera necesario.

Tras la investigación realizada y obtenidos los conocimientos necesarios para continuar, realicé un estudio de los diferentes modelos de inteligencia artificial que se podían emplear en el algoritmo. Gracias a artículos científicos y en parte a temario de diferentes asignaturas impartidas en otros años del Grado de Ingeniería del Software, pude llevar a cabo comparaciones objetivas entre estas tecnologías en colaboración con mi compañero y el equipo GASS, con el objetivo de seleccionar una de ellas para la implementación del algoritmo. La elección de lógica difusa fue una pieza clave en el desarrollo del trabajo. Primeramente se realizó una reunión educativa donde el equipo GASS hizo una propuesta de arquitectura para el algoritmo de lógica difusa. La reunión me proporcionó los fundamentos necesarios para realizar una investigación exhaustiva sobre el funcionamiento

y el modelo matemático de la lógica difusa. Tras la reunión realicé una investigación más a fondo acerca del modelo de lógica difusa, con la que adquirí los conocimientos necesarios para comenzar con el desarrollo del algoritmo.

Por parte del grupo GASS se nos proporcionó un ejemplo de código de implementación de la lógica difusa en *Python*, del que más adelante, se crearía el prototipo del algoritmo. Para poder realizar las implementaciones necesarias, realicé primeramente un estudio del código. Este estudio conllevó la revisión de la documentación oficial de las librerías *skfuzzy*, *numpy* y *matplotlib* además de la revisión del temario de la asignatura Ampliación de Bases de Datos. Este estudio del temario se hizo con el objetivo de repasar las bases de datos NoSQL. En concreto en el algoritmo se emplea *MongoDB*, para poder realizar las diferentes implementaciones y experimentación, revisé la documentación oficial de la biblioteca empleada para la conexión con *MongoDB* para el lenguaje *Python*. Después de llevar a cabo el estudio pertinente, junto a mi compañero José realicé el desarrollo del prototipo del algoritmo. Con una primera versión comencé parte de las experimentaciones, con el propósito de mejorar el algoritmo y el sistema de lógica difusa implementado.

Después de llevar a cabo la implementación del primer prototipo, se comenzó con las investigaciones de la viabilidad de incorporar las tecnologías DLT. En esta etapa hice revisiones de diversos artículos relacionados con estas las tecnologías DLT, la *blockchain* y los contratos inteligentes. En estas investigaciones no solo revisé artículos científicos, a su vez investigué y revisé documentación oficial sobre diferentes plataformas que ya implementan estas tecnologías, con el fin de escoger la más adecuada para el contexto del trabajo. Estas investigaciones sirvieron como base de conocimientos para entender la viabilidad de la incorporación de estas tecnologías.

Después de varias reuniones con el equipo GASS, conjuntamente se decidió emplear la plataforma *Ethereum* y su implementación de la *blockchain*, además del nodo cliente de *Hyperledger Besu*. Para poder comenzar con el desarrollo de contratos inteligentes, realicé un estudio de la documentación oficial tanto de la plataforma *Ethereum* como de *Hyperledger Besu*. En esta fase de investigación adquirí amplios conocimientos sobre la *blockchain* y *Ethereum* y realicé un estudio exhaustivo sobre la documentación que ofrecía la plataforma acerca del funcionamiento de los contratos inteligentes. Además, con el propósito de aprender los conocimientos necesarios para codificar contratos inteligentes, llevé a cabo un estudio de la documentación oficial del lenguaje de programación *Solidity*.

Una vez terminada esta fase de investigación acerca de las tecnologías DLT, con la elección de *Hypeldeger Besu* y su compatibilidad con *Java*, debimos realizar una migración del código del prototipo. Antes de poder comenzar con el desarrollo del algoritmo en *Java* se llevó a cabo una investigación de diferentes bibliotecas para la implementación de la lógica difusa en *Java*. Tras revisar varias de estas bibliotecas junto con mi compañero José, decidimos emplear *JFuzzyLogic*. Debido a que esta biblioteca tiene su propio lenguaje de programación para implementar modelos de lógica difusa, realicé un estudio de su documentación oficial. El resultado de esta investigación proporcionó los conocimientos necesarios para la implementación de la lógica difusa en *Java*. Adicionalmente, se revisó la documentación oficial de *MongoDB* en relación con su integración en *Java*. Tras adquirir los conocimientos necesarios, comencé con la implementación del código en *Java* y las diferentes experimentaciones realizadas para comprobar su correcto funcionamiento.

Después de completar la migración del código se comenzó con las pruebas de contratos inteligentes. En esta fase del proyecto codifiqué una serie de contratos inteligentes a modo de estudio del funcionamiento de los mismos. Este primer contacto se realizó

desde el IDE web *Remix*, para realizar diversas pruebas de despliegue e interacción con contratos inteligentes en una red *Ethereum*. Adicionalmente a las prueba de diferentes contratos inteligentes con *Remix*, me dispuse junto a mi compañero José a revisar la documentación que facilitaba *Hyperledger besu* acerca de las redes descentralizadas privadas. Esta documentación se empleó para realizar la configuración de la red privada de prueba. Además, en esta fase del proyecto revisé diferentes documentaciones de *Ethereum* e *Hyperledger besu* con el fin de entender el funcionamiento del protocolo de consenso empleado en la red de prueba. Tras la configuración de la red con los 4 nodos funcionales, realicé una investigación junto a mi compañero José sobre las diferentes librerías para *Java* que pudieran enlazar la red *Ethereum* con nuestra aplicación. Debido a esta indagación se eligió *Web3j*, por lo que realicé un estudio de su documentación. Este estudio de la documentación se acompañó con implementaciones de pequeños contratos inteligentes con el objetivo de comprobar el correcto funcionamiento de los mismos y de la librería *Web3j*. Después de esta etapa me dispuse a realizar la implementación del código del contrato inteligente final que se iba a emplear en nuestro algoritmo. Esta implementación llevó consigo una serie de experimentaciones para comprobar el funcionamiento y viabilidad de la integración del contrato inteligente.

En la fase final del trabajo, junto a mi compañero José, realicé las implementaciones comentadas en el apartado de la implementación final del algoritmo LoT del capítulo 4, con el propósito de obtener la última versión del algoritmo. Esta etapa de desarrollo brindó grandes actualizaciones y llevó a la realización de una serie de experimentos que realicé junto a mi compañero de proyecto. Estos experimentos se realizaron para comprobar la eficacia y funcionalidad del algoritmo y el contrato inteligente LoT.

Por último cabe mencionar que la redacción de la memoria del trabajo se ha realizado conjuntamente con mi compañero José. Para la realización de la memoria de este proyecto se llevó a cabo una reunión didáctica, donde por parte del grupo GASS se nos explicó el funcionamiento del sistema *Latex*, sobre el cual he realizado además una búsqueda de información para conseguir una estructuración y redacción correcta del texto de la memoria.

7.2. José Otegui Marín

Comenzando el proyecto se realizó una investigación exhaustiva de todo lo relacionado con la problemática, leyendo artículos de otros investigadores. Recibimos un taller de formación por parte del grupo *GASS* para poder comprender estos artículos y extraer la información verdaderamente relevante de ellos. En este se incluyó tanto comprender y sintetizar la información, la búsqueda en el *abstract* de la información clave que nos dice si ese artículo es de utilidad o no, y la búsqueda de más *papers* científicos en *Google Scholar*. Una vez recibida esta información se dispuso a leer dichos artículos, generar resúmenes y conclusiones para poder tener el conocimiento y las bases necesarias para el desarrollo correcto de nuestro trabajo. Todo este conocimiento fue utilizado a lo largo del proyecto. La información más relevante para nuestro trabajo fue extraída de todos aquellos artículos que trataban investigaciones relacionadas con la lógica difusa, la confianza en las redes 6g y los contratos inteligentes que servirían para comprender la problemática y poder desarrollar correctamente la implementación de estas tecnologías.

Una vez estudiados estos nuevos conceptos se procedió a comprender un primer prototipo de implementación de lógica difusa en *Python* desarrollado por uno de los

investigadores del grupo GASS, en el cual tuvimos que comprender el uso y funcionamiento de la librería *skfuzzy* de *Python* para implementar dicha lógica. Este prototipo utilizaba también una base de datos *NoSQL MongoDB*, por tanto tuvimos también que tratar de comprender la implementación de esta a la par que *matplotlib* y *numpy*, que se utilizaban para la visualización de los resultados. Una vez comprendidas estas tecnologías se realizaron adiciones al código para finalizar dicho prototipo del algoritmo en *Python*.

Una vez finalizada esta fase se procedió a la investigación más a fondo sobre la DLT y contratos inteligentes, ya que estas tecnologías se añadirían como propuesta por parte del grupo GASS ya que estos aportarían confiabilidad y aumentarían el nivel de confianza de las redes 6g. Tras reuniones con los investigadores, contextualizan sobre el flujo de trabajo y de implementación deseada donde irían ubicados los contratos inteligentes y las DLT en el algoritmo de nivel de confianza. En estas reuniones también se discute sobre distintas DLT y cuál podría ser la mejor opción, que finalmente concluyeron con la elección de *Ethereum*. *Hyperledger Besu* se elige como nodo cliente de la red de *Ethereum* el cual es compatible con Java.

Consecuentemente hubo que realizar una migración del prototipo realizado a *Java*. Tras esto, la librería *skfuzzy* de *Python* era inservible en este lenguaje, por lo que se procedió a investigar una alternativa. La alternativa encontrada es la biblioteca *JFuzzyLogic*, la cual emplea su propio lenguaje para definir las reglas de inferencia difusa y las variables o dimensiones que se emplean en la lógica difusa. Esta investigación fue exhaustiva ya que la documentación de esta librería es escasa en su página oficial y hubo que realizar muchas pruebas de concepto hasta que se logró implementar la lógica difusa en dicho lenguaje.

Al finalizar dicha migración se procede al estudio de la DLT elegida. Para ello se investigó el sitio oficial de *Hyperledger Besu* que, gracias a su documentación pudimos lograr lanzar una red privada con cuatro nodos. Esta tecnología tuvo una curva de aprendizaje muy compleja, ya que la red de cadena de bloques introduce conceptos abstractos que requieren de mucho estudio y comprensión. Además, a día de hoy es una tecnología en crecimiento de la cual todavía hay poca información y ejemplos de cómo utilizarla, por lo que el despliegue de estos nodos fue algo verdaderamente complejo, aunque finalmente, se logró con éxito.

Solidity es el lenguaje de programación de los contratos inteligentes en *Ethereum*, por lo tanto se procedió a la comprensión de este lenguaje para poder desplegarlos. Para hacer tests se utilizó *Remix*, un IDE web para ejecutar y depurar estos códigos en *Solidity*.

Tras implementar los contratos inteligentes de prueba se procede al despliegue y ejecución en un entorno privado el cual fue configurado con la documentación oficial de la ya mencionada *Hyperledger Besu*. Para esta configuración se hace uso de la librería Web3j de Java, es que está recomendada en esta documentación, por lo que se decide hacer uso de ella. *Web3j* se emplea para permitir la comunicación entre la red la cadena de bloques y la aplicación Java.

Después de estas pruebas exitosas, se desarrolla el contrato inteligente final que se implementa en nuestro algoritmo.

En la etapa final de este proyecto se procede a la integración del algoritmo que calcula el nivel de confianza con el código realizado del contrato inteligente, en el cual se almacenan los resultados de la ejecución del cálculo del nivel de confianza que concluyeron con una serie de pruebas y experimentos finales para asegurar la robustez del proyecto. Para la realización de esta memoria se ha recibido una pequeña guía por parte del grupo GASS

sobre *Latex*, un entorno de creación de documentos técnicos.

A lo largo del desarrollo de este proyecto he adquirido multitud de conocimientos y herramientas las cuales me han servido para aprender y conocer un mundo que en la actualidad está muy presente. En cuanto a mi compañero, Jesús, hemos trabajado conjuntamente en la realización de este proyecto y en la memoria, trabajando la comunicación y colaborando en las tareas de forma equilibrada y apoyándonos en las dificultades encontradas a lo largo del trabajo.

Capítulo 8

Introduction

8.1. Motivation

The constant evolution of wireless communication technologies has been a key factor in the development and transformation of our modern societies. From the introduction of the first mobile networks to the present day, each generation of communication technology has brought significant advancements in terms of speed, reliability, and connectivity capacity. However, despite the achievements of 5G networks, there is a need to move towards a new standard that addresses existing limitations and adapts to the emerging demands of an increasingly connected and digital world.

One significant limitation of 5G networks concerning trust is their ability to ensure the security and integrity of communications in highly dynamic and heterogeneous environments. While 5G networks have introduced significant advancements in terms of speed and capacity, their focus on spectral efficiency and latency has left gaps regarding connectivity trustworthiness. The growing interconnection of critical devices and systems in areas such as autonomous transportation, remote healthcare, and smart infrastructure presents new challenges in terms of security and privacy that 5G networks cannot optimally address.

On the other hand, 6G networks are envisioned as a wireless communication paradigm that goes beyond merely improving data speed and capacity. This sixth generation is expected to address these limitations by designing architectures and protocols that prioritize reliability, security, and privacy at all levels of connectivity. By integrating emerging technologies such as artificial intelligence, quantum computing, and distributed ledger technologies (DLT), 6G networks have the potential to establish a new standard in wireless communications.

This sixth generation will represent a significant evolution in wireless connectivity technologies, becoming a fundamental part of the future Internet of Everything (IoE). The new generation must enable the rapid, effective, and reliable collection and exchange of data among a wide variety of services, applications, machines, and users to form a real-time digital ecosystem. Developing this sixth generation presents a significant challenge with a series of goals that must be met, such as extremely high data speeds, low latency, increased energy efficiency, higher transmission capacity, and high security and privacy, among others.

In this context, the reliability of network elements is a critical aspect to consider in the design of 6G networks. Each network element must meet the expectations of this new technological era, ensuring not only constant and smooth connectivity but also data integrity, resilience to potential failures, and the ability to dynamically adapt to changing network environment conditions.

The increasing sophistication and number of cybersecurity attacks present serious challenges in protecting the integrity of network elements. This necessitates the rigorous implementation of proactive and robust security measures. This increase in security measures, along with the data collection and processing by network elements, may pose a risk to user privacy, as users are becoming more aware of how their data is handled and expect effective measures to ensure its protection and anonymity.

The motivation behind this research lies in the need for a reliable and robust network. This approach leads to the design and implementation of an algorithm that can estimate the trust level of network elements. This tool could address the reliability issue by studying and considering various factors.

8.2. Object of the Investigation

Sixth-generation wireless communication technologies should enable the connection and exchange of information among various elements of the network in the future Internet of Everything (IoE). These technologies are intended to set a new standard in the use of communication technologies and the potential applications derived from them. Considering this perspective, it is crucial to ensure that the network is secure and reliable to maintain optimal interaction.

The connections and services that are part of the network manage a large amount of information about the functioning of the connection and other aspects relevant to network trust, such as packet attestation or connection latency. By conducting a study and analysis of these values, along with other relevant ones, it is possible to estimate how reliable a service is within the network.

This work proposes a Level of Trust (LoT) algorithm that, by evaluating the necessary metrics, estimates the trust level of services within a 6G network. Alongside this, a solution is proposed for storing the resulting trust evaluation information within a Distributed Ledger Technology (DLT), thus employing decentralization as a primary security measure against cyber-attacks and enhancing the trust and immutability of the stored values.

8.3. Workplan

The development of this work, as can be seen in Figure 8.1, is structured into five phases:

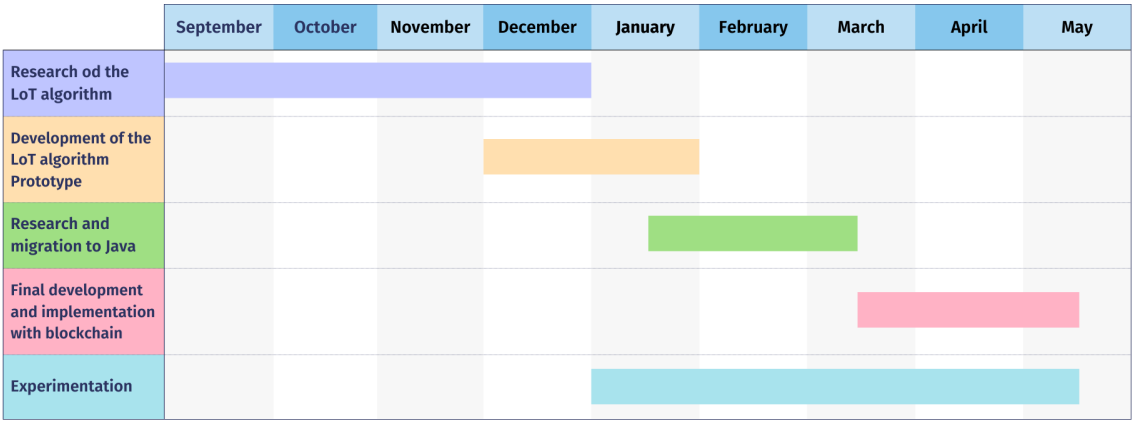

Figura 8.1: Gantt chart for the work plan.

1. **Research of the LoT Algorithm**:

   During the first four months in which this phase was developed, weekly meetings were scheduled with the purpose of initiating the study on the subject. In these meetings, the context of the research was first explained, fundamental concepts about networks were addressed, and the objectives to be achieved and main ideas under consideration at that time were established. Additionally, some relevant tools for the research were explained, and various related articles were studied, even suggesting certain scientific documents as a starting point.

   During the first months of the project, meticulous information gathering and study were carried out to expand knowledge in the area in which the work is being developed. Concurrently, periodic meetings were held to monitor the progress of the work and clarify certain doubts that arose as the research advanced.

   At this point, an initial investigation was conducted on 6G networks and all aspects related to the design of a reliable network. Subsequently, the main concept at this stage of the project was studied, defining the algorithm requirements, the aspects to be considered, and the expected results. Additionally, different artificial intelligence models that could be used in contexts like this research were studied.

   After understanding and evaluating all available options, various decisions were made, in collaboration with the team, regarding the implementation of the LoT algorithm. This included the selection of appropriate programming languages and libraries, the identification of network parameters to be evaluated, and the choice of fuzzy logic as the approach to address the proposed problem.

2. **Development of the LoT Algorithm prototype**:

   After thorough research into the fundamentals of the area of study, the development phase began. During this period, which lasted about two months, part of the research was combined with the implementation of the algorithm, as issues around its implementation emerged as the project took shape. Throughout this phase and

during the course of the rest of the project, the dynamic of weekly meetings continued to keep track of its progress.

Initially, a prototype of the algorithm was developed using the *Python* programming language. To begin the implementation, in-depth research on fuzzy logic was conducted to understand its workings, since it is a fundamental pillar of the proposed algorithm. Likewise, for the implementation of this prototype, an analysis of different programming libraries that were implemented was carried out, such as *SkFuzzy* for fuzzy logic or *connectMongo* for persistent storage of results.

3. **Research and migration to *Java***:

   This stage of the project, which lasted about two months, partly overlapped with the previous one once much of the algorithm prototype in *Python* had been implemented. During this phase, the feasibility of using a Distributed Ledger Technology (DLT) as a storage alternative, and even fully deploying the algorithm code in a smart contract, was considered. Therefore, preliminary research on *DLTs* and their operation was undertaken, investigating articles and documentation on different *DLT* architectures. To continue with the development of the algorithm, since the *Ethereum* platform and the open-source *Hyperledger Besu* client compatible with *Java* were chosen, a migration of the code started in *Python* to *Java* was decided. For this implementation in *Java*, research was conducted on libraries that could be useful for implementing fuzzy logic in *Java*. Subsequently, with the knowledge acquired, the migration of the application was carried out along with various additions and improvements to it.

   With the code already implemented in *Java*, several tests were conducted to understand the functioning of the *Ethereum* client, proceeding to the implementation of a private test network, which was designated for the subsequent development phase.

   Regarding *Ethereum*, research was conducted on the *Ethereum Virtual Machine* (EVM). This research also included a thorough study of the *Solidity* language, one of several languages compatible with the EVM and the one chosen for the implementation of the *Smart Contract*.

4. **Final development of the algorithm and implementation with blockchain**: In this phase, which lasted approximately two months, after several meetings with the team, the final implementation of the algorithm was determined, along with a smart contract capable of storing the algorithm's results.

   For this task, various *Java* libraries that could serve as a link between the *Ethereum* blockchain and the *Java* application were investigated. With the acquired knowledge, a smart contract was developed to store the results and facilitate their subsequent retrieval.

   This stage of development culminated successfully with the implementation of the LoT algorithm code in *Java* and its integration with the smart contract deployed on the blockchain.

5. **Experimentation**:

   This section of the work was being developed from phase 2 of the project plan. Initially, the viability of applying fuzzy logic in a project of such magnitude was evaluated, making various adjustments to the code in order to obtain different graphical visualizations of the results. Through the first prototype of the LoT

algorithm developed in *Python*, preliminary results began to be obtained that allowed for identifying necessary adjustments in the code. Later with the implementation of the algorithm in *Java*, comparisons and analysis of the results were carried out, contrasting them with real input data.

Additionally, during phases 3 and 4, reaction times, cost, and other relevant variables associated with blockchain technology were investigated. This stage of experimentation led to modifications in the code and a review of decisions during development motivated by observations of the results.

8.4. Struture of the Work

The remaining work is structured into seven chapters, as detailed below:

The chapter 2 is dedicated to introducing several fundamental concepts for a proper understanding of the project's scope. This section describes the theoretical framework of the work, including the description of 6G networks and the concept of trust evaluation. Additionally, an explanation of fuzzy logic is provided, which constitutes a fundamental pillar in this context. This chapter also addresses the description of various technologies used in the implementation of the algorithm. Within this theoretical framework, a technical explanation of blockchain networks, specifically *Ethereum*, will be provided. This includes an analysis of the technologies used for the implementation of smart contracts and the test infrastructure established through *Hyperledger Besu*.

The chapter 3 presents a series of related research on different aspects of trust evaluation of nodes and services in 6G networks, as well as exploring uses of different technologies to establish their utility with respect to future 6G networks or the main objective of this research.

The chapter 4 explains all the research and implementation of the trust level estimation software. This chapter describes the contributions made in terms of research, design, and development of the software that comprises the LoT algorithm.

The chapter 5 presents the results derived from various experiments conducted to evaluate the reliability of a service on the network. This section explains in detail the reliability of the results and the contrast with different metrics and data inputs.

The chapter 6 presents the main conclusions of this work, addressing the practical utility of the developed algorithm, examining its feasibility and effectiveness in real scenarios. It also discusses possible future research, focused on expanding and improving the scope of the algorithm, offering a clear vision of the progress that research in this field can represent.

The chapter 7 describes the contributions of each student to the project. In this part, the study and research work of the students is described, as well as the implementations that have been carried out in this work.

The chapters 8 and 9 correspond to the English translation of the introduction and conclusions of this work.

Capítulo 9

Conclusions and Future Work

9.1. Conclusions

The arrival of 6G networks will bring a series of significant challenges and opportunities. Consequently, new architectures and paradigms will be required to successfully address these issues arising from an evolution towards the sixth generation of mobile networks. One of the critical issues in this context lies in the need to establish a network architecture that ensures security and reliability. Developing a trust estimation algorithm for 6G networks represents a significant step towards building a safer and more reliable wireless communications ecosystem for the future. The research and development of this tool have led to the following conclusions:

1. Trust estimation of network services is one of the most important challenges to be solved with the implementation of the sixth generation of mobile networks. This trust assessment can allow a safer and more private environment for the future IoT where massive data analysis from various connected devices and users will occur.

2. Fuzzy logic is shown as an appropriate model to address issues similar to those presented in this work. The ability of fuzzy logic to handle imprecision is excellent and particularly in trust evaluation, it can be a really useful model. Fuzzy logic allows for the consideration of different factors that interfere in trust and through fuzzy rules and linguistic variables, a more complex and realistic evaluation level can be achieved. However, it has been specified largely due to the study of the results, that the fuzzy logic model must undergo a thorough study of the operation of mobile networks. To obtain exceptional and conclusive results, it is essential to draft inference rules faithful enough to the network behavior in a real scenario.

3. The incorporation of DLT and smart contracts both in the network architecture and in the trust evaluation process could represent a highly promising approach to addressing the issue of trust in 6G networks. Many studies propose this type of technology to help improve data integrity, analysis, and security. This claim is reinforced in this work, through the use of smart contracts as a means to store trust values. With the integration of these technologies, the system would become less susceptible to cyber attacks as the results would be decentralized and publicly available on a blockchain. However, the use of smart contracts could lead to delays in trust evaluation, as transactions for interactions with the smart contract must

undergo blockchain consensus processes before validation.

4. The data studied from the experimentation determine that the proposed solution in this work proves to be an effective tool for estimating the level of trust of services within 6G networks. In this context, the selection of metrics aimed at evaluating the network constitutes a very relevant aspect for the proper estimation of various types of services. By analyzing key metrics such as latency or the threat level, the LoT algorithm provides a consistent assessment of the reliability of the services, which would allow informed decisions to ensure optimal network interaction. Despite this, the limitations of the algorithm are known, and it is advisable to carry out future implementations aimed at improving the present work, as discussed in the following section.

9.2. Future Work

In this section, the following areas of research and future implementation are proposed:

1. Implementation of the algorithm logic into a smart contract. This integration is proposed in such a way that the algorithm code runs within a DLT. This approach increases trust in the results obtained and the security, integrity, and traceability of both the evaluated variables and the results, thanks to decentralization.

2. Implementation of an asynchronous storage element to deal with delays in calls to smart contracts. The development of this component would allow the application to continue functioning and processing data while waiting for the confirmation of transactions on the blockchain. This could significantly reduce perceived waiting times and improve the overall efficiency of the system.

3. Study and addition of other dimensions to the algorithm. In the proposed algorithm, metrics representing three specific dimensions are employed. However, the incorporation of various metrics, such as a privacy index, could enhance the results obtained. In addition, the incorporation of new dimensions might necessitate the implementation of a data anonymization mechanism. With this element, the preservation of privacy and confidentiality of the information contained in these metrics could be guaranteed.

4. Addition of weights in the variables to achieve differentiated evaluation of types of services and networks. The integration of a weighting system in the trust evaluation process enables the assignment of more or less importance to certain specific dimensions compared to others. This approach makes it possible to estimate trust in different scenarios where certain metrics may be more relevant than others, such as a service that deals with highly sensitive data, the dimension related to cyber attacks could have more relevance.

5. Addition of a historical average of trust values. The proposed algorithm, during its execution, awaits the reception of all the data from the metrics to carry out the trust evaluation. However, in real situations, it is possible that some of these metrics arrive at the algorithm devoid of data. For this reason, the implementation of the necessary logic for a weighted average over n previous trust values, allows for the correct evaluation of tasks received without values for some dimensions.

Bibliografía

[ABB+18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, New York, NY, USA, 2018. Association for Computing Machinery.

[AL23] Ana Lucila. García Villalba Luis Javier Alonso López, Jesús Ángel. Sandoval Orozco. Nivel de confianza y gestion de la privacidad en redes 6g basadas en intenciones para escenarios verticales. *XXXVIII Simposio Nacional de la Unión Científica Internacional de Radio (URSI2023)*, 2023.

[ASa21] Md. Jalil Piran and Mingzhe Chen and Shuguang Cui Amin Shahraki and, Mahmoud Abbasi and. A comprehensive survey on 6g networks: Applications, core services, enabling technologies, and future challenges. *CoRR*, abs/2101.12475, 2021.

[B+13] Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 1:22–23, 2013.

[CFLW15] Shenyun Che, Renjian Feng, Xuan Liang, and Xiao Wang. A lightweight trust management based on bayesian and entropy for wireless sensor networks. *Security and Communication Networks*, 8(2):168–175, 2015.

[EIP18] Nabil El Ioini and Claus Pahl. A review of distributed ledger technologies. In Hervé Panetto, Christophe Debruyne, Henderik A. Proper, Claudio Agostino Ardagna, Dumitru Roman, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*, pages 277–288, Cham, 2018. Springer International Publishing.

[Eth23] Ethereum. Documentación oficial de ethereum. https://ethereum.org/es/developers/docs/, Septiembre 2023.

[Eva11] Dave Evans. The internet of things. *How the Next Evolution of the Internet is Changing Everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG)*, 1:1–12, 2011.

[GPM+20] Marco Giordani, Michele Polese, Marco Mezzavilla, Sundeep Rangan, and Michele Zorzi. Toward 6g networks: Use cases and technologies. *IEEE Communications Magazine*, 58(3):55–61, 2020.

[HB16] Mike Hearn and Richard Gendal Brown. Corda: A distributed ledger. *Corda Technical White Paper*, 2016:6, 2016.

[HSBL17] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Vanet security challenges and solutions: A survey. *Vehicular Communications*, 7:7–20, 2017.

[KLDS20] Niclas Kannengießer, Sebastian Lins, Tobias Dehling, and Ali Sunyaev. Trade-offs between distributed ledger technology characteristics. *ACM Computing Surveys (CSUR)*, 53(2):1–37, 2020.

[MGV+20] Taras Maksymyuk, Juraj Gazda, Marcel Volosin, Gabriel Bugar, Denis Horvath, Mykhailo Klymash, and Mischa Dohler. Blockchain-empowered framework for decentralized network management in 6g. *IEEE Communications Magazine*, 58(9):86–92, 2020.

[N+08] Satoshi Nakamoto et al. Bitcoin. *A peer-to-peer electronic cash system*, 21260, 2008.

[NSW19] Warsun Najib, Selo Sulistyo, and Widyawan. Survey on trust calculation methods in internet of things. *Procedia Computer Science*, 161:1300–1307, 2019. The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia.

[PDKJ22] Guntur Dharma Putra, Volkan Dedeoglu, Salil S Kanhere, and Raja Jurdak. Toward blockchain-based trust and reputation management for trustworthy 6g networks. *IEEE Network*, 36(4):112–119, 2022.

[Pop18] Serguei Popov. The tangle. *White paper*, 1(3):30, 2018.

[REC15] Karen Rose, Scott Eldridge, and Lyman Chapin. The internet of things: An overview. *The internet society (ISOC)*, 80(15):1–53, 2015.

[SG23] J. A. Alonso López S. Guevara, L. J. García Villalba. Trazabilidad para servicios 6g mediante contratos inteligentes. *XXXVIII Simposio Nacional de la Unión Científica Internacional de Radio (URSI2023)*, 2023.

[Sol] Solidity. Documentación oficial de solidity. https://docs.soliditylang.org/en/v0.8.26/.

[SSJP17] Pradip Kumar Sharma, Saurabh Singh, Young-Sik Jeong, and Jong Hyuk Park. Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks. *IEEE Communications Magazine*, 55(9):78–85, 2017.

[Sun20] Ali Sunyaev. *Distributed Ledger Technology*, pages 265–299. Springer International Publishing, Cham, 2020.

[Sza96] Nick Szabo. Smart contracts: Building blocks for digital markets. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html, 1996.

[Sza97] Nick Szabo. The idea of smart contracts. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html, 1997.

[THD+20] Mohammad Tahir, Mohamed Hadi Habaebi, Mohammad Dabbagh, Amna Mughees, Abdul Ahad, and Kazi Istiaque Ahmed. A review on application of blockchain in 5g and beyond networks: Taxonomy, field-trials, challenges and opportunities. *IEEE Access*, 8:115876–115904, 2020.

[Vyp] Vyper. Documentación oficial de vyper. https://docs.vyperlang.org/en/stable/.

[YAX+20] Helin Yang, Arokiaswami Alphones, Zehui Xiong, Dusit Niyato, Jun Zhao, and Kaishun Wu. Artificial-intelligence-enabled intelligent 6g networks. *IEEE Network*, 34(6):272–280, 2020.

[YZZ+17] Hui Yang, Haowei Zheng, Jie Zhang, Yizhen Wu, Young Lee, and Yuefeng Ji. Blockchain-based trusted authentication in cloud radio over fiber network for 5g. In *2017 16th International Conference on Optical Communications and Networks (ICOCN)*, pages 1–3, 2017.