Escuela
Politécnica
Superior

Seguridad en robótica de enjambre mediante blockchain

Máster Universitario en Ciberseguridad

Trabajo Fin de Máster

Autor:
Miguel Ángel García Picó
Tutor:
Francisco A. Pujol López

Junio 2022

Universitat d'Alacant
Universidad de Alicante

Seguridad en robótica de enjambre mediante blockchain

Autor
Miguel Ángel García Picó

Tutor
Francisco A. Pujol López

Máster Universitario en Ciberseguridad

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Junio 2022

Resumen

La motivación tras la propuesta de este trabajo es el análisis de la seguridad en los sistemas de enjambres de robots. Este tipo de redes están compuestas por un gran conjunto de nodos de características semejantes que se comunican e interactúan entre sí para llevar a cabo tareas más complejas que no podrían desempeñar si no fuese por esta cooperación. Por su naturaleza, un factor esencial en estas redes es la comunicación entre los componentes para tomar decisiones en conjunto y actuar de forma armónica. Por ello, en la mayoría de los casos, será necesario que esta comunicación y que la red en su conjunto posea unos altos requerimientos en cuanto a seguridad para que posibles atacantes externos no sean capaces de llegar a afectar el correcto funcionamiento del sistema. Esto es de suma importancia ya que, en función de la tarea que vaya a realizar el enjambre, las consecuencias pueden llegar a ser catastróficas, como podría ser el caso de un ataque que tomase el control sobre un enjambre de robots que se dedicaba a una maniobra de carácter militar.

Blockchain es una tecnología emergente y que ha ganado popularidad en los últimos años a raíz del auge de las criptomonedas. Sin embargo, sus aplicaciones trascienden la de la banca digital y puede aplicarse en una gran cantidad de escenarios. Aún así, es importante tener en cuenta las características de blockchain y de la aplicación que se quiera implementar para valorar si es más o menos adecuado su uso. Su uso aporta características de seguridad, almacenando de forma distribuida los datos en todos los nodos de la red, por lo que puede resultar interesante unir este concepto al de robótica de enjambre para asegurar este tipo de sistemas.

Por todo lo explicado, los objetivos que se han llevado a cabo para este trabajo han sido, en primer lugar, un estudio del estado del arte de la robótica de enjambre y el blockchain, reflexionando sobre la situación de la seguridad de estos sistemas. Por otra parte, se ha valorado la aplicabilidad de blockchain a tareas de enjambres de robots y se ha propuesto el diseño de un sistema que integre estas dos tecnologías de una forma correcta. Finalmente, se ha desplegado un escenario de prueba en miniatura que permite visualizar de forma práctica el diseño propuesto.

Mis más sinceros agradecimientos a:

Mi tutor, Paco,
por su colaboración,
ayuda, apoyo y orientación
a lo largo del desarrollo
de este proyecto.

Mi familia y
amigos cercanos,
por su presencia e
incondicional apoyo
todo este tiempo.

Índice general

Índice de figuras

1 Introducción

De entre todas las áreas que abarca la ingeniería, una de las más presentes y en constante auge desde la segunda mitad del siglo XX es la de la robótica. Esto es gracias al gran avance en la tecnología que se ha producido en los últimos años y a la continua introducción de nuevos usos y escenarios sobre los que se puede aplicar esta disciplina. Es por esto que, hoy en día, se ha convertido en algo más que presente y necesario para todo tipo de tareas que precisan de automatización para ser llevadas a cabo de una forma más rápida y eficiente.

Como ya se ha comentado, la robótica puede ser aplicada para una gran variedad de tareas, cuya dificultad para llevarlas a cabo puede ser mayor o menor. En función de este grado de complejidad, será necesario el uso de robots simples que tengan funcionalidades contadas o robots con mejores características y más complejos. Otra alternativa posible al uso de robots complejos y con funcionalidades más completas es la utilización de un conjunto de autómatas, con características más simples y menos capacidades que se coordinen entre sí para llevar a cabo tareas complejas que no podrían llevar a cabo individualmente.

Esta última opción que se ha descrito es conocida como robótica de enjambre, y es un área dentro de la robótica que ha sido estudiada para llevar a cabo numerosos tipos de tareas, como pueden ser ejercicios de reconocimiento, labores de rescate, prácticas militares, etcétera. Sin embargo, y como se puede intuir, un elemento fundamental dentro de sistemas de enjambres de robots es la comunicación entre sus miembros. Esto es debido a que, como más adelante se explicará con detalle, los robots deben tomar decisiones y actuar conjuntamente con el resto de los componentes del enjambre. Esta comunicación entre los componentes y la toma de decisiones conjunta introducen vectores de ataque desde el punto de vista de la seguridad que deben ser tratados con cuidado, ya que la explotación de una vulnerabilidad de un enjambre de robots puede tener consecuencias muy graves en función de la tarea que esté realizando.

Surge entonces, por este motivo, la necesidad de encontrar soluciones a esta problemática. Existen diversos enfoques aplicados para la subsanación de esta necesidad de seguridad, pero uno de los introducidos más recientemente es el uso de blockchain con esta finalidad.

Blockchain es una tecnología emergente en los últimos años que se ha aplicado sobre todo a las criptomonedas, desde que fuese popularizado con el surgimiento de Bitcoin. Pese a este factor, blockchain no es más que una estructura de datos combinada con algoritmos criptográficos que proporcionan seguridad en los datos almacenados de una forma distribuida, por lo que puede ser usado en muchas otras aplicaciones al margen de las criptomonedas. Estas características convierten a esta tecnología en una opción a tener en cuenta a la hora de asegurar la comunicación en enjambres de robots.

Teniendo en cuenta todo lo nombrado con anterioridad, surge la motivación para la realización del presente trabajo. En él, se propone un estudio y explicación de las diversas alternativas existentes respecto a la comunicación dentro de enjambres de robots, poniendo especial énfasis en la aplicación del blockchain. Finalmente, se tratará de desplegar un pequeño escenario de uso práctico para mostrar una aplicación real de la teoría propuesta.

2 Estado del arte

En el apartado actual se hará un repaso sobre todos los campos directamente relacionados con la orientación de este proyecto. En cada uno de ellos, se hará un breve repaso de su historia para conocer cómo han llegado a su estado actual y luego se profundizará sobre este estado que tienen hoy en día. Se comenzará por hacer un recorrido sobre la robótica en general y en el punto en el que está en la actualidad, siguiendo con un repaso de la robótica de enjambre y terminando con un análisis de la historia y el estado del blockchain.

2.1 Robótica

Los antecedentes más relacionados con la robótica que conocemos hoy en día surgieron a mediados del siglo XX [1], cuando se desarrolló un mecanismo conocido como 'telemanipulador'. Mediante este dispositivo, un operador era capaz de tratar con elementos radiactivos y peligrosos sin necesidad de mantener contacto y ponerse en peligro. Este tipo de artefactos empezó a ser utilizado en más industrias además de la nuclear, pero cuando surgió el primer concepto de robot fue cuando el operador humano que lo controlaba fue sustituido por un programa ejecutado en un ordenador.

Esta definición de robot es la que se ha mantenido hasta nuestros tiempos, una máquina la cual, con la correcta programación, puede realizar tareas tediosas, pesadas o repetitivas en las que sustituir a los seres humanos. Estas acciones que al principio eran limitadas a ser operaciones de repetición, evolucionaron con la introducción de sensores externos para proporcionar la capacidad de reaccionar a condiciones externas. La última evolución producida en la robótica ha sido el uso de inteligencia artificial, para dotar al robot de la capacidad de razonar lógicamente e introducir el aprendizaje.

Con el paso de los años, la complejidad de los robots ha ido aumentando, incorporando todo tipo de sensores y actuadores para llevar a cabo complejas tareas sin necesidad de supervisión humana. No obstante, este aumento en las funcionalidades de los robots implica un aumento a la par en costes de los robots, mantenimiento, etcétera. Este factor causa que, dependiendo de las características de la labor que se desee realizar, sea una mejor opción el uso de robots con cualidades más simples coordinados entre sí para desempeñar la función deseada. Esta es la idea principal de la robótica de enjambre.

2.2 Robótica de enjambre

Antes de proseguir con la realización de un estudio más detallado sobre las características y estrategias de seguridad dentro de la robótica de enjambre, algo esencial es tener una base

sobre en qué consiste para tener un marco sobre el cual trabajar. Por ello, en este apartado se hará una introducción a este concepto junto con sus propiedades, ventajas e inconvenientes que presenta, y escenarios en los que es aplicable. Como es normal, muchos aspectos más detallados de este área quedan fuera del alcance de este proyecto, ya que se desea poner foco sobre la seguridad por encima del resto de cuestiones.

2.2.1 Origen y definición

De la misma forma que pasa con otros conceptos tecnológicos que existen, la inspiración detrás de los enjambres de robots es tomada como imitación de un fenómeno natural. Seguramente, el primer ejemplo que se viene a la cabeza de este comportamiento es, por el nombre que ha adoptado, el de un enjambre de abejas. Se trata de una buena ejemplificación para ilustrar el caso, al igual que podrían serlo las hormigas, las termitas y demás casos de insectos sociales.

Figura 2.1: Enjambre de abejas trabajando conjuntamente

Tomando como referencia el ejemplo de las abejas, se puede observar que se tratan de individuos con unas capacidades limitadas que, si no fuese porque cooperan con muchos otros individuos, no podrían construir los complejos panales y almacenar la cantidad de miel que generan. Algo similar ocurre en el caso de las hormigas, cuyas integrantes de un hormiguero cooperan para crear los habitáculos y llevar a cabo tareas como el transporte de alimentos.

Una vez explicado el origen de esta idea y su símil dentro de la naturaleza, se puede pasar a ver sus inicios en la tecnología [2].

Los orígenes históricos de esta disciplina de la robótica se remontan al año 1988, donde Beni [3] y Fukuda [4] aplicaron por primera vez la palabra *enjambre* en un contexto relacionado con la robótica y elaboraron por primera vez definiciones para esta especialidad. En años posteriores se fueron haciendo nuevos avances e innovaciones dentro del área hasta llegar a la concepción que tenemos hoy en día, que de forma general se puede definir como se muestra a continuación.

La robótica de enjambre es un área dentro de la robótica que hace uso de la inteligencia distribuida para coordinar un conjunto de robots sencillos y de bajo coste que pueden efectuar complejas tareas gracias a la cooperación con el resto de integrantes, y que no serían capaces de realizar de forma individual. Dentro de los sistemas de este tipo, todos los agentes que lo componen deberán tener las mismas características y capacidad de decisión para que, de esa forma, las elecciones tengan que ser tomadas de forma grupal, sin que ningún integrante pueda tener un peso mayor en las determinaciones.

Por otro lado, la inteligencia distribuida [5], a la que se ha referido con anterioridad, se trata de un ámbito dentro de la inteligencia artificial, encargado del análisis del comportamiento colectivo de sistemas descentralizados y organizados de forma autónoma. Estos algoritmos son los encargados de determinar las normas y procedimientos necesarios para poder conseguir el correcto funcionamiento individual de los miembros dentro del enjambre y también de forma conjunta con el resto. Por nombrar algún ejemplo concreto de algoritmo de este tipo, se podría señalar el de optimización de colonia de hormigas, consistente en la optimización de problemas que tienen como objetivo hallar caminos hasta un fin, explorando el entorno conjuntamente y encontrando la ruta más óptima mediante diferentes iteraciones que van optimizando la solución alcanzada.

2.2.2 Características

Una vez se ha definido de una manera general y se tiene una idea inicial clara sobre el tema principal que se va a tratar, se puede pasar a presentar sus características. Las propiedades presentes dentro de un enjambre de robots pueden variar, pero los factores clave que deben estar presentes en la gran mayoría de los casos son los siguientes:

- **Flexibilidad**. La robótica de enjambre no está focalizada en realizar un tipo concreto de tareas, sino que pretende abarcar misiones de diferentes índoles y características. Por ello, el sistema debe ser flexible a la hora de ser capaz de llegar a distintas soluciones dependiendo de la tarea que se le ha cometido, además de poder actuar de una forma conjunta para reaccionar dinámicamente a los cambios de entorno.
- **Escalabilidad**. Esta propiedad se refiere a la capacidad del sistema para trabajar con tamaños de enjambre de diferentes tamaños, pudiendo aumentar el mismo para llevar a cabo tareas que sean más costosas en cuanto a recursos sin afectar al rendimiento.
- **Robustez**. Algo esencial dentro de los enjambres es la facultad de poder seguir trabajando pese a fallos en el sistema o problemas en el entorno. Además, debido a la simplicidad de cada uno de los elementos, si el enjambre perdiese alguno o un grupo pequeño de ellos debería seguir pudiendo trabajar correctamente y deberían de ser fácilmente reemplazables.
- **Autonomía**. El enjambre está compuesto por robots autónomos que pueden interactuar de forma física con el entorno y tomar decisiones por su cuenta.
- **Descentralización**. Uno de los objetivos principales que se desean con esta tecnología es la ejecución de tareas sin un líder centralizado, permitiendo así una menor vulnerabilidad ante ataques y fallos de un nodo central que controlase todo el enjambre.

Por lo tanto, este factor implica que los robots deberán de ser capaces de organizarse y coordinarse entre ellos sin ninguna clase de intervención humana o no humana que haga de entidad central.

- **Estigmergia**. Esta propiedad se refiere a la capacidad que deben tener los miembros de colaborar los unos con los otros a través del medio físico para efectuar las tareas colectivamente.

2.2.3 Ventajas e inconvenientes

Ahora que se han enumerado las distintas propiedades que caracterizan a los enjambres de robots, se puede proseguir haciendo una balanza entre los distintos beneficios y desventajas que tienen este tipo de sistemas. A continuación, se enumera una lista de ventajas que poseen:

- Los robots son capaces de lidiar con los cambios que se puedan producir en el entorno.
- Este tipo de sistemas poseen de una gran flexibilidad, lo que les permite poder encargarse de tareas de distinto tipo, y escalabilidad, por lo que puede cambiar su tamaño en función de las necesidades que tenga el trabajo.
- Cada uno de los agentes que componen el sistema disponen de una simplicidad mucho mayor que los utilizados fuera del modelo de enjambre, por lo que hace que cada uno de ellos sean bastante más baratos. Se podría llegar a pensar que el hecho de que se trate de una gran cantidad pueda hacerlos más caros que utilizar uno o unos pocos robots más complejos, pero en la mayoría de ocasiones sigue siendo menos costoso.
- El hecho de tener un sistema conformado por muchos integrantes hace que sea posible llevar a cabo diferentes acciones o tareas en paralelo, lo cual puede acortar el tiempo necesario para su realización, algo que sería imposible mediante el uso de un único autómata.

Por el contrario, estos sistemas también presentan algunos inconvenientes como los que ahora se nombrarán:

- La autonomía que presentan cada uno de los robots puede desencadenar que, en ciertas ocasiones, algunos integrantes tomen decisiones de forma esporádica y diferente al resto del grupo.
- El hecho de que se trate de un sistema descentralizado y todo lo que implica hace que no sea una buena opción en muchos casos.
- La poca complejidad de los robots hace que sea complicado el diseño de sistemas con aplicaciones reales que puedan tener una total garantía de éxito en sus tareas, lo que hace que en algunos casos sea necesario proporcionar un conocimiento global a los robots.

2.2.4 Ejemplos de uso

Como se ha podido ver, esta tecnología presenta diversos puntos a favor y en contra, lo que hace necesario tenerlos en cuenta a la hora de tomar la decisión de usar o no un sistema de este tipo. Para ilustrar mejor casos en los que sí sería una opción viable su uso, seguidamente se mostrarán una serie de ejemplos de tipos de tareas:

- **Coordinación de movimientos**. Es el caso en el cual los robots deben mantener una posición coordinada mientras realizan cierto tipo de movimiento, algo que se podría asemejar a un banco de peces o una bandada de aves. Puede tener finalidades como el transporte de objetos o propósitos de entretenimiento, como espectáculos que se realizan con drones creando figuras en el cielo.
- **Reconocimiento**. Se refiere a tareas en las cuales se debe identificar ciertos objetos o entidades dentro de un área concreta.
- **Transporte de objetos**. Consiste en la coordinación de los robots para trasladar objetos que no podrían ser cargados por uno solo.
- **Distribución espacial**. Existen tareas para las cuales es necesario el despliegue de robots repartidos dentro de un entorno para la monitorización de los alrededores.

Tareas con algunas de las características descritas se pueden encontrar en muchos campos, como puede ser el industrial, el militar, la agricultura, etcétera. Como tareas concretas en las que podría ser útil el uso de enjambres, se exponen a continuación algunos ejemplos:

- Tareas de agricultura, como la recogida o la replantación de semillas de un campo.
- Tareas militares de reconocimiento, defensa, ataque o desactivación de explosivos, por ejemplo.
- Tareas dentro de la medicina que pueden incluir el uso de nanorobots para exploración y actuación en el cuerpo humano.
- Tareas de rescate y reacción ante accidentes de forma automatizada.
- Tareas relacionadas con el espectáculo y un propósito de ocio meramente.

Figura 2.2: Espectáculo de drones cooperando para crear figuras

2.3 Blockchain

Ahora que ya se ha aportado una introducción básica sobre la robótica de enjambre, se puede pasar a hacer lo propio con el segundo elemento principal dentro de este trabajo, el blockchain. La estructura seguida será la misma que en el apartado anterior, proporcionando un contexto sobre los orígenes que iniciaron esta idea, pasando a dar una definición de la misma y su estado actual, siguiendo con las ventajas e inconvenientes y terminando con las aplicaciones que tiene hoy en día. Sus detalles más técnicos en materia de seguridad se analizarán en posteriores capítulos.

2.3.1 Origen y definición

La primera aproximación del blockchain de la forma en la que se conoce hoy en día ocurrió en el año 1990, cuando Stuart Haber y W. Scott Stornetta publicaron un trabajo [6] sobre la aplicación de criptografía a una cadena de bloques que no podía ser modificada con marcas de tiempo de los documentos. Sin embargo, no fue hasta 2008 cuando se produjo el gran auge de esta tecnología.

Este hecho fue producido por la publicación de Bitcoin por parte de una persona o grupo de personas de identidad desconocida con el nombre de Satoshi Nakamoto. En la publicación [7], se definían las características y el funcionamiento completo de este sistema monetario electrónico que se sostenía por medio de una cadena de bloques. Este hecho de que se popularizasen a la par el Bitcoin y el blockchain causa muchas veces en el público general una confusión y una creencia de que ambos conceptos son lo mismo, pero Bitcoin no es más que blockchain aplicado a un caso concreto de moneda digital.

Unos años más tarde, en 2013, se produjo otro avance que introdujo nuevas ideas al blockchain. Fue cuando Vitalik Buterin, creyente de que Bitcoin tenía limitaciones, publicó su propia moneda a la cual dio el nombre de Ethereum [8]. La principal novedad que introducía era el concepto de los *smart contracts*, que permitían llevar a cabo diversas funciones y que se explicarán más adelante con detalle. Desde entonces, una gran cantidad de monedas virtuales han aparecido mediante el uso del blockchain, pero también han surgido aplicaciones al margen de las transacciones monetarias donde aplicar esta tecnología con buenos resultados.

Habiendo repasado de forma superficial sus orígenes y su estado actual, a continuación se definirá de una manera general en qué consiste esta tecnología.

Blockchain es una base de datos distribuida y compartida entre todos los nodos que componen la red. Este registro tiene una peculiaridad a la hora del almacenamiento de los datos, y es que se almacenan divididos en bloques donde cada uno almacena un trozo de la información. Cuando a estos bloques se les completa el espacio que tienen disponible para almacenar datos, se crea uno nuevo y se encadena este último con el anterior con la ayuda de funciones criptográficas. Una forma simplificada sería verlo como un libro mayor dividido por hojas en las que se van apuntando transacciones, y cuando se comienza una nueva página se sabe con seguridad cual es la que le precede inmediatamente. Como ya se ha comentado y se verá posteriormente, no es necesario que los datos sean relativos a transacciones, como sí

ocurre en el caso de las criptomonedas.

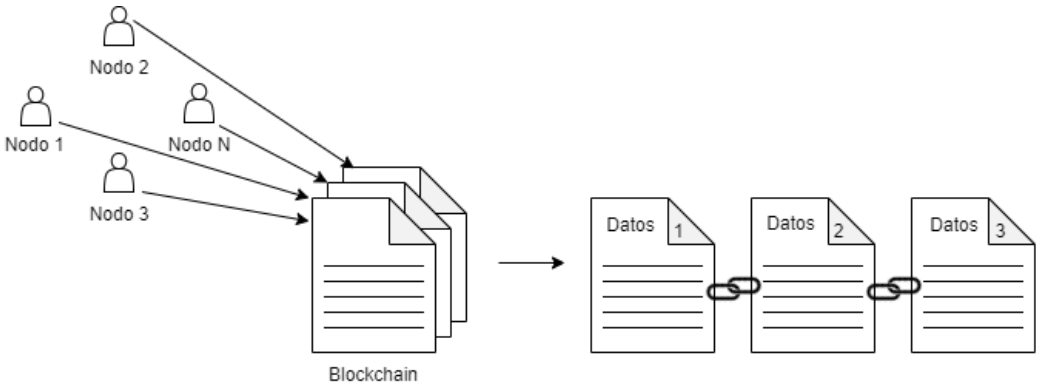

Figura 2.3: Diagrama representativo del sistema blockchain

2.3.2 Características

Se procederá ahora a listar una serie de propiedades que siempre estarán presentes mediante el uso de esta tecnología:

- **Inmutabilidad**. Esta característica es una de las principales, y garantiza que la cadena y, por lo tanto, la información almacenada en ella no puede ser alterada o modificada. Esto es posible gracias a que todos los nodos del sistema guardan una copia de la cadena y, además, cuando se quiere añadir un nuevo bloque debe de ser validado por todos ellos. Los detalles de estas operaciones se verán más adelante.

- **Descentralización**. Al igual que ocurría con la robótica de enjambre, una de las propiedades es la descentralización, y se debe a que, como ya se ha visto, todos los nodos guardan una copia de la información compartida. Este factor elimina la existencia de ninguna entidad central que se encargue de la gestión de la información.

- **Seguridad**. El hecho de que no haya una autoridad que controle el sistema implica un aumento de seguridad importante, al no tener un punto claro sobre el cual atacar. Además, a toda la información almacenada se le aplica funciones hash, lo cual garantiza una capa de seguridad adicional.

- **Consenso**. Que exista una descentralización tan grande hace necesaria la aparición de procedimientos y algoritmos de consenso para llegar a conclusiones conjuntas entre todos los nodos. En este caso, el consenso podría ser la verificación por parte de todos de que una transacción o un bloque es válido. Ejemplos de algoritmos de este tipo pueden ser los de Proof of Work (PoW), Proof of Stake (PoS) o Proof of Authority (PoA), que serán detallados con posterioridad.

2.3.3 Ventajas e inconvenientes

Aunque por las características que se han enumerado parezca que es una tecnología que es positiva en todos los aspectos, también presenta algunas desventajas. Primero se enumerarán algunas de los principales pros que tiene:

- Mayor tolerancia a fallos por la ausencia de gestión humana.
- Las operaciones son transparentes a todos los nodos de la red.
- Fidelidad y seguridad sin la necesidad de terceras partes autorizadoras.
- Mayor aguante ante ataques maliciosos al no existir un nodo central.
- No repudio de las operaciones, lo cual significa que una vez un nodo ha hecho algo se puede saber con certeza quien, cuando y qué se ha hecho.

Sin embargo, este sistema también tiene ciertos contras que habrá que tener en cuenta a la hora de considerar aplicar el blockchain a cierto caso concreto. Ahora se muestran algunas de estas desventajas:

- Algunas soluciones de blockchain tienen un alto consumo de energía por la complejidad computacional que tienen algunos algoritmos de PoW, por ejemplo.
- La inmutabilidad es una ventaja y una desventaja a la vez, ya que no se puede deshacer ninguna operación, aunque haya sido por accidente.
- A veces son ineficientes en aspectos como el almacenamiento de datos.
- No es completamente segura, ya que existen ataques como el del 51%, que se detallarán más adelante.
- Como es una tecnología novedosa existe una tendencia a intentar aplicarla a todo tipo de situaciones, pero solo es apropiada para casos con unas determinadas características. Para el caso en concreto de este trabajo se deberá valorar si es más o menos adecuada.

2.3.4 Ejemplos de uso

Blockchain es una tecnología que se puede aplicar a prácticamente todo. Sin embargo, como se ha comentado anteriormente, no siempre es la mejor opción y puede suponer una mala decisión usarla si el escenario no es el correcto. A continuación, se muestra una serie de casos de uso donde poder aplicar el blockchain, sin parar a evaluar si es más o menos adecuado su uso para cada una de las situaciones:

- El más utilizado en la actualidad, las criptomonedas o monedas digitales.
- Compartición de información de forma transparente entre organizaciones.
- Seguimiento veraz del historial de un vehículo de transporte hasta la entrega.
- Almacenamiento en la nube de forma descentralizada.
- Aplicación en dispositivos Internet of Things (IoT).

- Control de derechos de autor y lucha contra la piratería.
- Identificación digital como los DNI, pasaportes, etcétera.
- Historiales médicos seguros.

3 Objetivos

Una vez se ha introducido la idea principal del trabajo y se ha analizado el estado del arte y el marco teórico de los temas involucrados, se puede tener una idea más clara sobre los objetivos que se marcarán para este proyecto. Igual que ya se ha comentado brevemente con anterioridad, la idea principal de este trabajo es el estudio de la adaptación de la tecnología blockchain a los sistemas de enjambres de robots.

Sin embargo, antes de abordar la investigación de dicha propuesta, algo interesante sería realizar un análisis previo de otras alternativas en cuanto a infraestructura y comunicación de los enjambres. De esta forma, se podría tener una idea más general sobre otras posibilidades y poder ver sus ventajas y desventajas, así como características que incorporan en lo relativo a la seguridad del enjambre y la comunicación para poder compararlas entre sí y con la variante que incorpora el blockchain.

Ya se ha hecho una introducción sobre el blockchain, pero si se desea realizar un estudio sobre la posibilidad y viabilidad de introducir esta tecnología dentro de la robótica de enjambre, será necesario tener unos conocimientos más detallados sobre la misma y su funcionamiento interno. Por ello, se dedicará un bloque de la memoria a toda esta explicación de conceptos relativos.

Tras haber expuesto todos los conocimientos que se verán involucrados en la propuesta de solución final, se llevará a cabo un estudio con la finalidad de encontrar la forma más óptima de unir las dos tecnologías. Dicho análisis incluirá una propuesta de solución, que tendrá que ser valorada de forma posterior para obtener unas conclusiones sobre los beneficios que aporta y los inconvenientes que puede llegar a tener.

Finalmente, se tratará de desplegar un entorno de prueba para lanzar una puesta en marcha práctica de la solución que incorpore el blockchain, y poder tener así un caso de uso real, aunque sea a pequeña escala, para ilustrar la idea.

Los objetivos principales, enumerados de una forma breve, podrían ser los siguientes:

- Presentación de las distintas alternativas de comunicación e infraestructura utilizadas en enjambres de robots.
- Explicación en detalle de los conceptos teóricos de blockchain que serán utilizados para la solución.
- Propuesta de una solución que incorpore la robótica de enjambre junto con el blockchain.
- Despliegue de un pequeño caso de prueba para exponer la idea estudiada.

- Análisis del estudio realizado, elaboración de conclusiones obtenidas y búsqueda de mejoras y trabajo futuro.

4 Metodología

A continuación, se pasará a comentar la metodología y el proceso que se ha seguido hasta completar la elaboración de este proyecto.

El primer paso que se realizó fue establecer unas directrices generales del proyecto, para poder servir como punto de partida para comenzar la investigación e ir modelando la propuesta. En un primer momento, la idea que se tuvo fue la de llevar a cabo un análisis del estado del arte de la robótica de enjambre y los problemas presentes en cuanto a seguridad, además de hacer este análisis sobre la tecnología blockchain y los estudios existentes sobre la combinación con los enjambres de robots. Otro pensamiento que se tuvo en el inicio fue el de implementar un sistema de blockchain propio, desarrollando todos los componentes aunque fuese de una forma más simplificada.

Sin embargo, tras hacer una investigación sobre esta última idea, se consideró más adecuado utilizar una implementación oficial de una blockchain ya existente. Esta decisión fue tomada por la dificultad que suponía el desarrollo de una cadena de bloques que implementase las funcionalidades deseadas, además de que se consideró más interesante la alternativa de usar una ya existente y montar un entorno de pruebas con la misma.

Hecha esta corrección con respecto al planteamiento inicial del proyecto, el siguiente paso fue efectuar un proceso de investigación. Este proceso incluyó todos los temas comentados anteriormente, extrayendo las ideas más importantes y relevantes para la temática del proyecto de forma que fuese posible reflejarlas en la memoria posteriormente. Por otro lado, el estudio fue de utilidad a la hora de conocer las distintas ventajas y desventajas que poseen cada una de las tecnologías, construyendo un conocimiento necesario para considerar la forma y la viabilidad de la integración de ambas.

Este conocimiento fue el utilizado en una fase posterior, que fue la del diseño del sistema. Se tomó como objetivo el diseño de un sistema general que fuese factible de ser empleado en un escenario real, sin tener en cuenta las características de cada tarea o escenario concreto. En él, se definieron todas las características que debía tener, así como el flujo de operaciones a realizar para desplegar una tarea con un enjambre.

Una vez diseñado el sistema, el último paso del desarrollo fue la búsqueda de una forma de instalar un entorno de pruebas, lo más parecido posible al diseño propuesto y en una escala menor. Para ello, se estudiaron las distintas alternativas de herramientas a utilizar en cuanto a lenguajes de programación, marcos de trabajo, blockchains, etcétera. Una vez escogidas, se procedió con el despliegue para comprobar el funcionamiento.

Finalmente, llevadas a cabo las pruebas, sacadas algunas conclusiones y visto el trabajo realizado con el sistema propuesto, se consideró que el alcance del proyecto era correcto y se

puso fin al mismo.

5 Desarrollo

Llegados a este punto y expuestos ya los objetivos que se han tomado como referencia, en este capítulo se procederá con las explicaciones y desarrollos de los temas necesarios para intentar cumplir con las metas propuestas.

5.1 Comunicación en enjambres de robots

Teniendo en cuenta las características que se han introducido anteriormente sobre la robótica de enjambre, se puede intuir que algo fundamental es la forma con la cual los individuos intercambian información para llegar a acuerdos dentro de una ejecución de una tarea. Por ende, otro aspecto esencial será que dicha comunicación sea realizada de la manera más segura posible, sin dar oportunidad a que la información sea perdida o alterada en su transcurso, lo cual podría conducir a situaciones catastróficas dependiendo de la tarea en concreto. Es por ello que en este apartado se hará un repaso sobre diferentes alternativas y características en lo relativo a la difusión de datos.

5.1.1 Forma de comunicación

Un aspecto interesante a resaltar antes de comentar opciones a la hora de la comunicación y sus características es que los enjambres de robots pueden clasificarse en dos tipos dependiendo de la homogeneidad de sus componentes. Estos tipos se dividen en:

1. **Enjambre homogéneo**. Es la opción más utilizada, y consiste en que todos los integrantes iguales a la hora de las acciones que pueden realizar y el peso que tienen sus decisiones.

2. **Enjambre jerárquico**. Puede darse el caso de que dentro de un enjambre haya diversos tipos de robots, y que algunos de ellos tengan un rol más importante dentro del sistema. Esta superioridad se puede traducir en mayor control sobre el resto del grupo, en mejores características, mayor capacidad, etcétera.

De la misma forma, otra característica importante es la forma mediante la cual se comunican los individuos, que en este caso se podría dividir en tres categorías:

1. **Comunicación directa**. Esta alternativa es sobre la que se pondrá la atención en el estudio, ya que se trata de intercambio de información realizado implícitamente entre los integrantes del enjambre. Este traspaso de datos se puede realizar de diferentes formas, ya sea mediante un canal WLAN, canales bluetooth, canales de radiofrecuencia, infrarrojos, audio, etcétera.

2. **Estigmergia**. Consiste en modificarse con el resto a través de información que se va dejando en el entorno.

3. **Detección local**. Las decisiones se toman en función del comportamiento de los robots adyacentes.

5.1.2 Amenazas

Como ya se ha comentado, el objetivo de este trabajo es la aplicación del blockchain dentro de este ámbito para estudiar los beneficios que aporta, sobre todo en cuanto a seguridad. Por ello, algo esencial es conocer los riesgos a los que se enfrentan sistemas de este tipo en escenarios normales [9].

En primer lugar, se deben conocer las distintas intenciones que puede tener posible atacante. Entre las más comunes pueden destacarse las de descubrir información confidencial, suplantar, corromper o introducir robots malignos dentro del enjambre, o tratar de eliminar robots o información del mismo. Estas diferentes finalidades que puede tener un atacante pueden ser realizadas por atacantes de diversas características, las cuales clasifican a continuación:

1. **Conocimiento**. Un atacante puede ser interno o externo en función del conocimiento que tienen del sistema.

   - Interno. En este caso se dispone de contraseñas, claves criptográficas u otros elementos que hacen posible la infiltración en el sistema de forma malintencionada. Este tipo de ataques hacen necesarios mecanismos de detección de intrusos.

   - Externo. El perpetrador procede de fuera del sistema, por lo cual no tiene acceso a los elementos anteriormente mencionados. Puede ser evitado mediante métodos de criptografía de clave pública y similares.

2. **Permisos**. El ataque puede ser de mayor o menor gravedad dependiendo de los permisos de los cuales disponga.

   - Pasivo. El atacante pasivo es capaz de interceptar información confidencial, la cual podría ser usada para posteriores ataques. De nuevo, se podría prevenir en gran medida estos ataques mediante criptografía.

   - Activo. En este caso, el atacante tiene permiso de modificar información o corromper el enjambre para introducir datos incorrectos.

3. **Alcance**. Una vulneración del sistema se podrá clasificar en local o global según la cantidad de robots que es capaz de afectar.

   - Local. El autor de un ataque local solo tiene acceso a los robots locales, es decir no tiene una visión de la información o lo que están haciendo el resto.

   - Global. Por otro lado, si se trata de un atacante global, puede saber cualquier tipo de comunicación ocurriendo en todo el enjambre.

Ahora que se conocen algunas de las diferentes propiedades que puede tener un ataque, seguidamente se mostrará un listado de posibles ataques a los que están expuestos los enjambres de robots. Específicamente se señalan ejemplos en lo relativo a la comunicación, ya que es donde se está centrando el análisis, y se pasan por altos ataques referentes al software, hardware u otros vectores de ataque.

- **Ataques de desautenticación**. Tienen como objetivo deshabilitar temporalmente a los robots de la red, lo cual afecta a la disponibilidad del sistema, pero también a la integridad de la información.
- **Análisis de tráfico**. Este tipo de ataques persigue el escaneo del flujo de información producido durante la comunicación para acceder a datos no cifrados, algo que afecta directamente a la privacidad y confidencialidad del sistema.
- **Inyección de datos**. Consiste en la intercepción de datos, modificarlos con un fin determinado y enviarlos al flujo de nuevo. Este ataque afecta de primera mano a la integridad de la información.
- **Denegación de servicio**. Este tipo de ataque y su versión distribuida (DDoS) permiten saturar el sistema y conseguir que no se pueda transmitir información durante un determinado periodo de tiempo. Como es normal, la disponibilidad del sistema se ve claramente afectada por estos ataques.
- **Suplantación**. En el caso concreto de la robótica de enjambre, este ataque podría consistir en la introducción de nuevos robots o suplantación de integrantes del enjambre, consiguiendo así tener más peso en las decisiones que se tomen de forma conjunta. La detección de robots malintencionados dentro del sistema es una de las mayores problemáticas dentro de la seguridad de esta tecnología.

5.2 Blockchain

Una vez se ha hecho una introducción sobre los diferentes problemas y amenazas a los cuales se enfrenta la comunicación dentro de los enjambres de robots, se pasará presentar los detalles de la tecnología propuesta para la integración, el blockchain.

Al igual que se ha introducido previamente, blockchain no es más que una forma de almacenar información de forma distribuida dentro de redes Peer-to-peer (P2P), o redes entre iguales. La peculiaridad que introduce esta tecnología es la forma que tiene de guardar los datos y proporcionarles garantías en cuanto a la seguridad mediante diferentes métodos.

5.2.1 Componentes

Para hacer una introducción sobre los aspectos fundamentales de blockchain, el primer paso será comentar todos los componentes que pueden entrar en juego dentro de una cadena de bloques, que se explicarán en esta sección.

5.2.1.1 Nodos

Los nodos son los integrantes que componen y participan en la blockchain. Cada uno de ellos dispone de una dirección, que no es más que una manera de identificar de forma única a cada uno de los nodos que la componen, de forma que los nodos puedan tener forma de referirse entre ellos y que se pueda saber cual de ellos ha realizado qué acción concreta. Esta dirección podrá estar correspondida con la clave pública de los integrantes, una clave necesaria para operaciones criptográficas relativas a la seguridad que se verán más tarde, o ser generada a partir de esta. Por otro lado, estos nodos poseen una clave privada que únicamente debe ser conocida por el nodo al que pertenece.

Cada uno de estos nodos guardará internamente el estado de la cadena de bloques, de forma que se pueda determinar de forma conjunta el estado de la cadena en cualquier momento. Además, no todos los nodos tendrán el mismo rol en la cadena, ya que hay algunos que pueden tomar la decisión de llevar a cabo el proceso de minería para validar las transacciones, concepto que se explicará posteriormente.

5.2.1.2 Bloques

Los bloques son la base de la blockchain, y son los encargados de almacenar toda la información o todas las transacciones. Estos bloques tendrán, dependiendo de la blockchain, un tamaño máximo determinado de los datos que puedan albergar. En el momento en el que este tamaño sea completado, se añadirá a la blockchain si se ha verificado que toda la información es correcta y otro nuevo bloque se creará, enlazándose con el anterior para poder seguir el orden de la cadena. El bloque primitivo o el primer bloque de una blockchain se conoce como bloque génesis.

Cada uno de estos bloques contendrá valores adicionales además de la propia información que se quiere almacenar. A continuación, se muestra una lista de los diferentes valores que pueden componer un bloque:

- **Marca de tiempo**. Valor que indica el momento en el que se ha generado el bloque.
- **Hash del bloque**. Valor de los datos del bloque completo una vez se le ha aplicado una función hash. Más adelante se explicará esto con detalle.
- **Hash del bloque anterior**. Este valor representará al bloque anterior, y lo enlazará de forma que se pueda saber con seguridad cual es dicho bloque y así sucesivamente.
- **Nonce**. Se trata de un valor determinado con unas características en concreto, que será el resultante de aplicar una función hash a la estructura del bloque completo. Más adelante también se comentarán más detalles sobre la utilidad de este valor.
- **Datos**. Los datos componen el cuerpo del bloque, y no son más que la información que se desea almacenar.

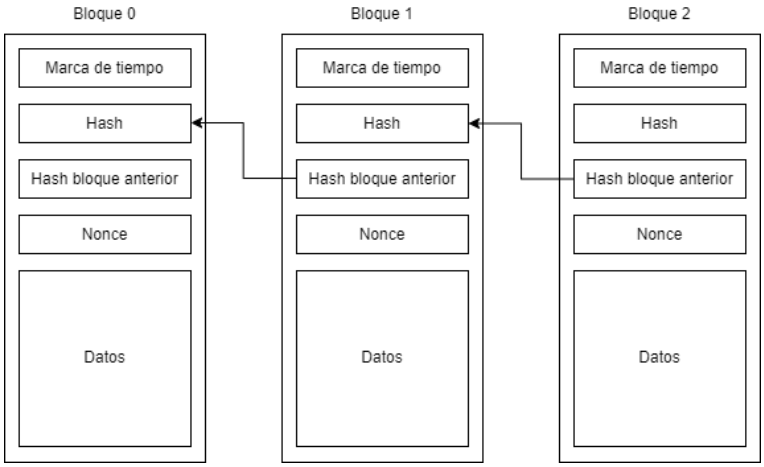

Figura 5.1: Ejemplo de estructura de los bloques

5.2.1.3 Cadena

Todos los bloques están conectados con su predecesor de forma inequívoca, como ya se ha comentado. Esta unión de todos los bloques que guardan la totalidad de la información representa la cadena. Esta estructura de datos es irrevocable, lo que quiere decir que cualquier escritura que se haya hecho sobre la misma permanecerá para siempre, a no ser que la cadena sea eliminada por completo.

5.2.2 Operaciones

Tras haber visto los diferentes componentes que integran la blockchain, se puede pasar a ver las diferentes operaciones que ocurren y que son necesarias para el mantenimiento del sistema.

5.2.2.1 Comprobación de la integridad

Previamente, se ha nombrado que cada uno de los bloques tienen un hash propio y uno del bloque anterior. Estos valores tienen la funcionalidad de asegurar la integridad, es decir, garantizan que los datos o las transacciones de los bloques sigan el orden correcto y no puedan ser modificados. Dichos valores son obtenidos mediante la aplicación de funciones hash o funciones resumen.

Las funciones resumen son un conjunto de operaciones que convierten una cadena de texto de longitud variable en otra cadena que tiene siempre el mismo número de caracteres, denominada hash o resumen. La propiedad que las hace especiales es que a partir del texto inicial es muy fácil y poco costoso computacionalmente calcular su resumen, mientras que a partir del resumen no se puede calcular directamente el valor inicial. Otra característica importante es que es muy difícil encontrar dos cadenas de entrada que resulten en el mismo hash, por lo que cada uno de los hashes de los bloques será único, actuando como si fuese una huella dactilar del mismo.

A continuación, se muestra un ejemplo en el que se puede ver que, pese a que las dos cadenas de entrada difieren por un solo carácter, los dos resúmenes son totalmente diferentes. En el ejemplo se ha usado la función hash por simplicidad, aunque actualmente está obsoleta. Ejemplos de funciones resumen utilizadas en la actualidad podrían ser Secure Hash Algorithm 2 (SHA2) o SHA3.

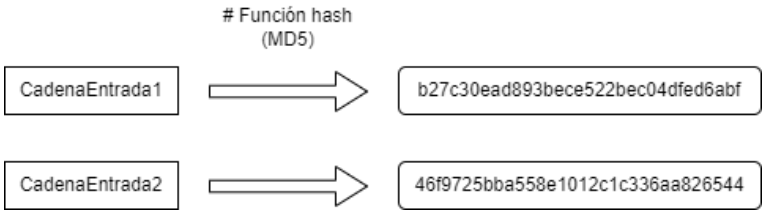

Figura 5.2: Ejemplo de uso de funciones resumen

De esta forma, si se quiere comprobar la integridad de la cadena completa, simplemente se debe ir recorriendo la misma desde el último bloque verificando que los resúmenes se corresponden de forma correcta, algo que se puede hacer de sencillamente gracias a la propiedad que tienen las funciones resumen de ser aplicadas con poco coste, como ya se ha comentado.

5.2.2.2 Autenticación e identidad digital

Ya se ha mencionado que una blockchain se trata de una red entre iguales, por lo que todos los nodos deben ser capaces de escribir datos o transacciones en la cadena. Para ello, cada una de esas peticiones de escrituras atraviesa un proceso de autenticación y autorización antes de integrarse definitivamente.

En primer lugar, es necesario autenticar a los usuarios y las escrituras que hacen en el sistema. El diseño de blockchain está pensado para funcionar sin una autoridad central, lo cual no cambia el hecho de que cada uno de los integrantes tenga que autenticar sus transacciones. Para ello, se hace uso de criptografía de clave pública, lo que implica que cada uno de los miembros tendrá una clave privada que sólo él mismo conocerá y una clave pública a la cual tendrá acceso cualquier nodo de la red.

La principal finalidad con la que se utilizan estas claves es para garantizar que el autor de una escritura es, efectivamente, quien dice ser. El uso de criptografía de clave pública con la motivación de identificar a un individuo es conocido como firma digital. En este proceso, para realizar la firma el usuario deberá aplicar una función de descifrado sobre el mensaje que se quiere enviar usando su clave privada. El individuo o individuos que tengan que verificar la identidad tendrán que aplicar la función de cifrado opuesta usando la clave pública del usuario emisor, conocida por todos. Este resultado se comparará con el mensaje original y, si ambos coinciden, se habrá confirmado la identidad digital del usuario.

Seguidamente se muestra un ejemplo del funcionamiento de la verificación mediante firma digital. Para el ejemplo se han comparado directamente los mensajes, pero también podría

hacerse uso de los resúmenes de estos.

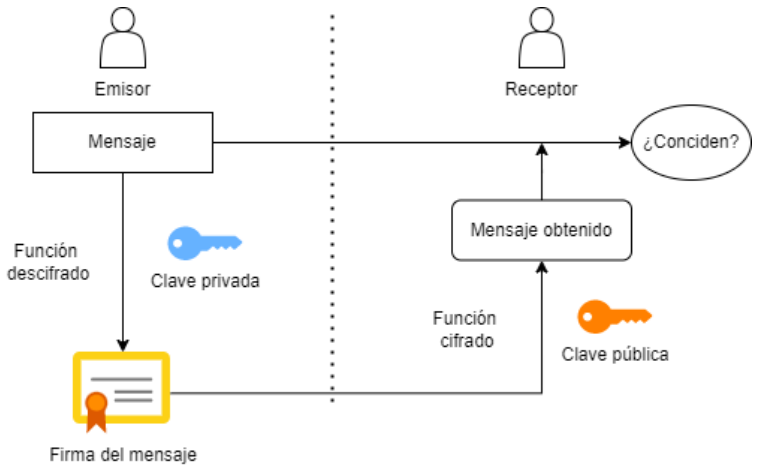

Figura 5.3: Funcionamiento de la firma digital

5.2.2.3 Mecanismos de consenso

Una vez una escritura o transacción ha sido acordada, tiene que ser aceptada y autorizada antes de que sea añadida a un bloque de la cadena. Con dicha finalidad, y pese a que no existe ninguna entidad central que tenga esta misión, existen los algoritmos de consenso. Estos algoritmos son procedimientos mediante los cuales los compañeros de la cadena llegan a un acuerdo sobre el estado actual de la cadena. En resumen, aseguran que cualquier nuevo bloque que se añada sea la única versión existente y acordada por todos los nodos que componen la red.

Existen varios algoritmos de consenso que pueden utilizarse dependiendo de la blockchain [10], y algunos de ellos son los siguientes:

- **Proof of Work**. Este algoritmo de consenso surgió con la aparición de Bitcoin y tiene relación directa con lo que se conoce como minería. Describe un sistema que requiere una cantidad de trabajo que disuade a atacantes de llevar a cabo acciones. En él, se debe realizar una tarea relativamente costosa computacionalmente, que puede ser encontrar un *nonce*, nombrado en la sección 5.2.1.2, el cual al aplicar una función hash al bloque completo su resultado comience por diez ceros, por ejemplo. De esta forma, el nodo que se dedicase a minar, debería ir probando muchos valores para este *nonce* con la finalidad de que se cumpla la condición nombrada. Una vez se encuentre el valor, el bloque sería autorizado y se añadiría en la cadena, siendo aceptado por todos los nodos de la red. En el caso de la red Bitcoin, el usuario que encontrase dicho valor recibiría una recompensa por su aportación.
- **Proof of Stake**. La prueba de participación surgió con la finalidad de solventar algunos problemas encontrados en PoW. Uno de ellos era que el proceso de minería resultaba en una alta latencia de producción de nuevos bloques, además del alto consumo energético por el coste computacional que supone. A su vez, PoW centraliza en cierto nivel la red

con los grandes grupos de mineros, mientras que PoS democratiza el acceso a las diferentes tareas. En resumidas cuentas, la finalidad que tiene este protocolo es incentivar a los participantes a poseer en todo momento una cantidad determinada de monedas, tomando como ejemplo su aplicación financiera. Este factor les hace ser elegibles en el proceso aleatorio de selección que asigna las tareas. Por ende, aquellos nodos que posean mas monedas, tendrán mayor peso en la red y unas oportunidades más grandes de ser seleccionados, de forma que podrán validar transacciones y crear nuevos bloques.

- **Proof of Authority**. Este protocolo tiene una gran diferencia respecto a los dos nombrados anteriormente, y es que los validadores disponen su identidad y reputación como una garantía de transparencia. Esto implica una selección de forma arbitraria de estos validadores confiables, existiendo un número limitado de estos. En primer lugar, se deberán elegir estos nodos de forma aleatoria de entre algunos que hayan sido votados por otros ya autorizados, de forma que se eviten nodos maliciosos. PoA no dispone de un sistema de minería, por lo que tiene algunos de los beneficios ya comentados para PoS. Como se ha comentado, la identidad y la reputación es valiosa en este protocolo, y el validador debe revelar de forma voluntaria quien es de forma que se puedan determinar los responsables del funcionamiento del sistema. De esta forma, si tienen comportamientos que conlleven un mal funcionamiento de la blockchain, podrá saberse y ser desautorizados. Sin embargo, presenta algunas desventajas como pueden ser una menor descentralización, al ser unos pocos los participantes en las tareas de la red, o el hecho de tener que exponer públicamente la identidad de los validadores y sacrificar su privacidad.

5.2.2.4 Smart contracts

Los *smart contracts* o contratos inteligentes fueron descritos por primera vez de una forma detallada por Nick Szabo [11] en el año 1997. No obstante, dada la tecnología existente en aquel momento no era posible ponerlo en práctica, algo que se hizo viable con la aparición de blockchain.

Pasando a la definición de un contrato inteligente, hay que recordar que un contrato es un acuerdo que ocurre entre dos o más partes donde se concreta lo que se debe o se puede hacer y qué pasa si no se hace. Los contratos siempre han sido verbales o escritos, sujetos a interpretación, leyes y demás aspectos burocráticos que implican costes de intermediarios. Sin embargo, un contrato inteligente puede hacerse cumplir y ejecutarse de forma autónoma sin necesidad de ningún intermediario.

En resumen, un *smart contract* es un trozo de código informático donde se describen los términos del contrato, el cual es visible por todos y no es alterable al estar soportado sobre la tecnología blockchain y distribuido en todos los nodos que forman la red. Por lo tanto, se trata de unos programas muy seguros y que solo son susceptibles de fallar cuando su programación no ha sido la correcta. Estos contratos se ejecutarán como reacción a diversos eventos o situaciones que puedan suceder, otorgando un gran abanico de posibilidades en cuanto a las aplicaciones que se les pueden dar.

Además, cabe destacar que estos contratos, al estar desplegados sobre la blockchain, son

inmutables. Esto implica que si un contrato ha sido programado de forma incorrecta accidentalmente y se quiere modificar, será necesario escribir en la blockchain la nueva versión del contrato y comenzar a interactuar con su dirección dentro de la cadena, dejando de usar la dirección del contrato antiguo. Cabe destacar que estos contratos pueden tener estado interno que almacene información, lo cual resulta muy útil en muchas ocasiones.

Los contratos están verificados por todos los nodos e integrados dentro de la blockchain, además de que poseen direcciones propias a través de las cuales pueden recibir transacciones por parte de los nodos de la red, incluso recibiendo parámetros, de forma que se ejecuten las funciones incluidas dentro de estos *smart contracts*. Algunos ejemplos de lenguajes de programación que son utilizados para escribir contratos inteligentes son Solidity, Vyper, Yul, Haskell o Rust.

5.2.3 Tipos

Una de las decisiones que deben tomarse a la hora de aplicar blockchain a un determinado caso es la elección del tipo que se desea utilizar. Dependiendo de las características de cada caso será mejor decantarse por uno o por otro. En esta sección se describirán las diferentes clases de blockchain que existen y sus propiedades.

Para empezar, la gran división que se puede hacer en este tipo de redes es en función de su acceso. Esto significa que existen blockchains las cuales son accesibles por cualquiera y no es necesario permiso, y otras que no están disponibles para todo el mundo y necesitan de una autorización simplemente para acceder a la red. Dentro de estas dos variantes se engloban los siguientes tipos de redes blockchain que se enumeran a continuación:

- **Públicas**. Como su propio nombre indica, este tipo pertenece al grupo de redes en las cuales no es necesario permiso para participar. En ellas, todos los nodos tienen los mismos derechos de acceso, creación de bloques y validación de los mismos. Principalmente se están usando en la actualidad para el uso de las criptomonedas, como el caso de la red Bitcoin o Ethereum, anteriormente nombradas.
- **Privadas**. Igual que el caso anterior, se puede entender por su nombre que se trata de una red cuyo acceso está controlado y se debe tener permisos para participar. La capacidad de llegar a ser un nodo de la red está determinada por una entidad central, que también puede otorgar derechos o funciones diferentes dependiendo del nodo. Este factor hace que se pierda la descentralización total, pero hay casos en los que es bueno sacrificar esta propiedad para ganar otras ventajas. Respecto a las públicas tienen la ventaja de un menor tiempo necesario para la validación de los datos, pero tienen la desventaja de que pueden llegar a ser más vulnerables a atacantes y nodos fraudulentos si consiguen tener acceso.
- **Consortium**. Este tipo surgió para tratar de solucionar algunas desventajas presentes en las blockchains privadas. También se tratan de redes con necesidad de autorización en el acceso, las cuales están gobernadas por un conjunto de entidades, en vez de una única como en el caso de las privadas, lo cual resulta en una menor centralización y mayores niveles de seguridad. Sin embargo, esto añade una dificultad adicional a la hora

de la cooperación entre las diferentes autoridades e introduce retos como el riesgo de desconfianza entre ellas.

- **Híbridas**. Están controladas por una única entidad, pero supervisadas por una blockchain pública que es necesaria para realizar ciertos tipos de validaciones en las transacciones.

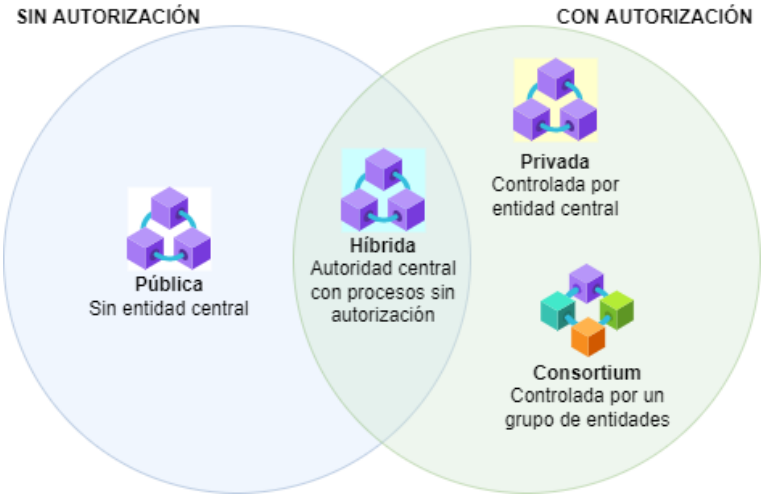

Figura 5.4: Esquema de los tipos de blockchain

5.2.4 Flujo de funcionamiento

Una vez se han conocido los componentes y las distintas tareas existentes dentro de una cadena de bloques, se procederá a describir un funcionamiento normal de este tipo de redes. Unos posibles pasos serían los siguientes, pudiendo interpretarse las transacciones como simples escrituras de datos en la blockchain:

1. El primer paso sería el registro de los nodos en el sistema blockchain, generando para cada uno de ellos un par de claves pública y privada, además de una dirección, obtenida a partir de la clave pública, que servirá para poder dirigirse entre ellos.
2. Un usuario crea una nueva transacción, firmándola con su clave privada para verificar la identidad.
3. El usuario transmite la transacción a la red para que los demás nodos puedan verla.
4. Los nodos de la red comprueban la validez de esta transacción en todos los sentidos.
5. Las transacciones se van añadiendo a bloques, que cuando completan su tamaño máximo son añadidos a la cadena.
6. Estos bloques se van añadiendo solo si se llega a un consenso con alguno de los mecanismos que se han comentado previamente.
7. Una vez la transacción se ha añadido a la cadena, se considera completa.

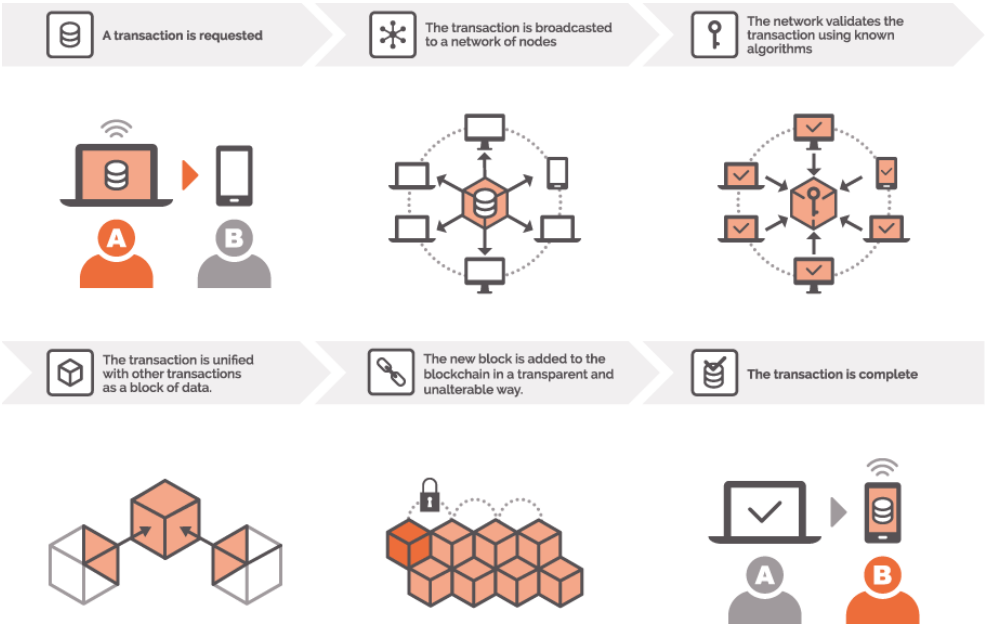

Figura 5.5: Diagrama del flujo de blockchain[1]

5.3 Diseño del sistema propuesto

Llegados a este punto, se han presentado y explicado, en mayor o menor medida, todos los elementos que formarán parte del sistema que se tiene como objetivo. Por ello, en este apartado se procederá a hacer una explicación del diseño propuesto para la integración de un sistema de enjambre de robots y la tecnología blockchain. Cabe destacar que se pondrá énfasis sobre todos los temas relativos a la seguridad, y que se pasarán por alto los detalles relativos al funcionamiento o la toma de decisiones de un enjambre de robots, ya que se considera fuera del alcance de este proyecto.

5.3.1 Estructura general

En primer lugar, es necesario realizar una definición de la estructura general que tendrá el sistema. Como ya es conocido, la intención es integrar un enjambre y una cadena de bloques, por lo que cada uno de estos robots que compone el enjambre se corresponderá con un nodo de los existentes dentro de un sistema blockchain, como se podría deducir por la similitud entre ambos sistemas. En consecuencia, cada uno de los robots deberá estar corriendo un proceso adicional para poder llevar a cabo la comunicación con los demás robots de la red mediante la blockchain, convirtiéndose de esta forma en un nodo de la red. A continuación, se muestra un esquema simple de la estructura general.

[1] https://kilroyblockchain.com/what-is-blockchain

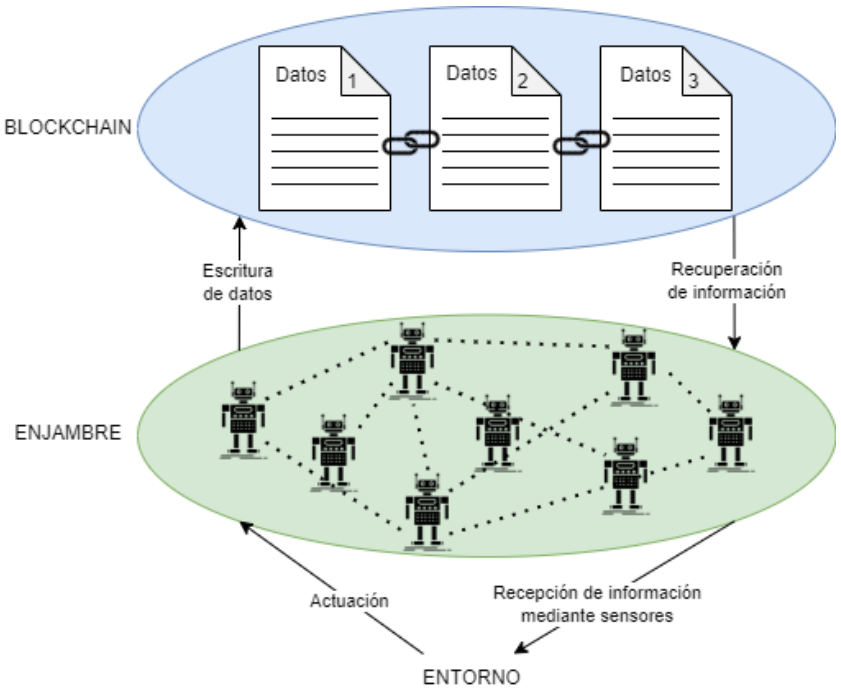

Figura 5.6: Estructura general del sistema

Como se puede observar, el esquema está muy simplificado y cada uno de los flujos que se muestran implican muchos procesos complejos de forma interna, pero se ha mostrado de esta forma para tener una idea inicial. Esencialmente, cada robot del sistema tendrá una copia de la cadena de bloques completa, e interaccionará con esta para el envío de datos (valores de sensores, votos sobre cierta decisión, etcétera) y la recuperación de la información almacenada (variables obtenidas de otros robots, acciones a realizar, etcétera). A su vez, el enjambre mantendrá una interacción con el entorno para recibir la información del mismo y utilizarla para actuar en consecuencia y en consenso con el resto de integrantes.

5.3.2 Tipo de blockchain

Una vez delimitada la estructura general, se puede proseguir explicando algunos detalles del sistema propuesto. En este caso, se comenzará por elegir el tipo de blockchain a utilizar en una hipotética implementación, teniendo en cuenta las distintas opciones que se han comentado con anterioridad en el apartado 5.2.3.

5.3.2.1 Elección del tipo

Primeramente, habría que tener bien delimitadas las características de la red que se quiere desplegar para saber qué tipo de blockchain aporta las mejores propiedades. En este caso, lo ideal sería tener una red de un cierto número de nodos, controlada de forma que solo puedan ingresar en ella aquellos robots que sean decididos por la persona o personas encargadas de desplegar el enjambre, con la intención de evitar robots maliciosos o intrusos, en la medida

de lo posible. Es por esto que queda descartada la opción de usar una blockchain pública, ya que cualquier atacante podría introducirse y comenzar a manipular el sistema mediante falsos valores, votaciones u otros factores.

Habiendo descartado las públicas, la decisión queda entre aquellas para las que es necesario permiso para ingresar, privadas y *consortium*. Ambas opciones presentan una mayor privacidad, eficiencia, estabilidad y latencia que las públicas. Esta elección tiene menos peso, ya que ambas podrían ser una buena opción, con algunas ventajas e inconvenientes diferentes. Al elegir una *consortium* blockchain, algunos problemas de las privadas se solventarían al tener una mayor descentralización y mejores niveles de seguridad. Sin embargo, habría que definir un mecanismo de cooperación entre las autoridades existentes y resolver la problemática de la confianza entre ellas. Por lo tanto, la elección sería determinada por los requerimientos y los niveles de complejidad que puede permitirse el proyecto en concreto, siendo las privadas más rápidas de implementar y las *consortium* más fiables en cuanto a seguridad.

5.3.2.2 Forma de ingresar

La red sobre la que se despliegue la blockchain deberá estar situada, idealmente, lo más próximo al escenario de actuación del enjambre para reducir la latencia lo máximo posible. En el caso de ser una tarea que abarcase una zona de grandes dimensiones, no habría otro remedio que ser accedida de forma remota, en vez de estar dispuesta en una red local, por ejemplo. Esencialmente, la red siempre deberá ser accesible por todos los nodos, pero inalcanzable por agentes externos que no estén autorizados.

Por ende, es necesario acordar la forma mediante la cual cada robot se conectará a la blockchain para iniciar su trabajo. La primera estrategia de seguridad en cuanto a este aspecto podría ser definir un modelo de *whitelisting* o lista blanca. Esta lista contendría la clave pública de cada uno de los nodos permitidos, así como la dirección obtenida a partir de esta clave pública. De esta forma, cuando cualquier nodo desee acceder a la blockchain, primero se verificará que su dirección esté presente dentro de la lista de nodos permitidos.

Para aumentar la seguridad de una simple lista blanca, otra estrategia de seguridad sería, una vez comprobado que la dirección está dentro de la lista, llevar a cabo una verificación de la identidad del nodo, haciendo uso de la clave pública almacenada y la firma digital explicada anteriormente. De esta manera, solo se podría acceder en el caso de que el nodo verifique su identidad de forma correcta con la clave privada que solo él posee. Llevados a cabo correctamente estos dos pasos, el nodo podría acceder al sistema y comenzar a realizar sus funciones.

A continuación, se enumeran los pasos del proceso completo para visualizarlos de forma clara:

1. Se crean las cuentas de los nodos dentro del sistema blockchain concreto, obteniendo sus pares de clave pública y privada, además de la dirección obtenida a través de la aplicación de una función hash a la clave pública.
2. Se crea la blockchain privada, obteniendo un identificador de la misma que será utilizado posteriormente para conectarse a ella.

3. Se rellena la lista con las claves públicas y las direcciones de los nodos que tendrán permiso de acceso.
4. Cuando un nodo intenta acceder introduciendo el identificador de la cadena, se comprueba que su dirección exista en la lista blanca.
5. Si la dirección es comprobada de forma exitosa, se lleva a cabo la comprobación de la identidad del nodo.
6. Cuando ambos procesos hayan sido finalizados satisfactoriamente, el nodo habrá accedido a la blockchain.

5.3.3 Consenso en el sistema

Una vez se ha definido la forma de acceder al sistema, otro aspecto de los aspectos restantes e importantes a tener en cuenta y diseñar es la forma que tendrán todos los nodos para ponerse de acuerdo entre ellos. Como se introdujo con anterioridad, este objetivo es conseguido mediante el uso de los algoritmos de consenso. Por lo tanto, al igual que se ha hecho con el tipo de blockchain, habrá que decidir cual de las diferentes alternativas se ajusta mejor para las características que presentan los enjambres de robots.

Como primer paso se ha decidido descartar la opción del algoritmo PoW, ya que este algoritmo está sobre todo pensado para cadenas de bloques públicas, además de que introduce un alto coste computacional y energético que no es permisible en un sistema como el que se desea implementar. Adicionalmente, este proceso de minería podría introducir una alta latencia y baja velocidad a la hora de creación de nuevos bloques, algo que tampoco beneficia al propósito que se tiene.

El algoritmo PoS se podría considerar una mejor opción que la anterior. Esto se debe a que se elimina el gran coste computacional y energético que acarrea el algoritmo PoW. Adicionalmente, la capacidad de validación de bloques se otorga en mayor medida a aquellos nodos con mayor peso o mayor cantidad de monedas. A esta característica se le podría sacar provecho recompensando a aquellos nodos que tomen mejores decisiones, de forma que tengan mayor peso a la hora de validar escrituras de datos y añadir nuevos bloques.

Finalmente, la última alternativa es el algoritmo PoA. Como ya se ha explicado, el número de validadores confiables es limitado en este caso, siendo elegidos de forma arbitraria los primeros. Posteriormente, se elegirán más validadores de forma aleatoria de entre nodos que hayan sido votados por los ya autorizados, evitando de esta forma integrantes maliciosos. La desventaja en cuanto al sacrificio de la identidad, teniendo que ser expuesta la de los validadores, no presenta una gran desventaja en este caso ya que no es preocupante que el resto de nodos conozcan la identidad, además de que ayuda a detectar y desautorizar a ciertos validadores en el caso de que estén tomando muchas acciones incorrectas. El sistema de minería tampoco está presente en este algoritmo, con los beneficios que esto supone y que ya se han comentado en el caso de PoS, además de tener una baja latencia de aceptación y tener unos intervalos de tiempo regular entre cada bloque añadido. Por todo lo que se ha señalado, se considera que puede ser la mejor opción a la hora de emplear una blockchain de las características que tiene este sistema, por lo que se hará uso este algoritmo.

5.3.4 Smart contracts

Ya se ha definido el concepto y la utilidad de los contratos inteligentes con anterioridad. Estos contratos jugarán un papel fundamental dentro del sistema diseñado. Como ya se ha dicho, estos contratos son validados y publicados en la blockchain, por lo que cada nodo tendrá una copia, y los nodos tienen que realizar transacciones a su dirección para llevar a cabo las acciones que sean necesarias dentro de la cadena. En el momento en que un contrato es añadido a la cadena, se quedará en ese lugar para siempre, por lo que si se desean realizar cambios en el contrato habrá que añadir otro nuevo incluyendo los cambios y dejar de usar el antiguo.

Cada uno de estos contratos puede definir múltiples funciones, por lo que no es necesario publicar un contrato para cada funcionalidad que se desee integrar en la cadena. Para cada nueva blockchain que se cree para una tarea concreta del enjambre, se deberá publicar e incluir un contrato en la cadena que contenga las funciones necesarias. En este caso, para todas las cadenas nuevas creadas será necesario incluir en el contrato una función que sea la encargada de registrar al robot, de forma que el contrato tenga conocimiento del nodo para poder tener comunicación con el mismo posteriormente. Además, el estado interno que tienen los contratos dentro de la cadena puede ser de gran utilidad para el almacenamiento de datos.

Adicionalmente, estos contratos deberán incluir funciones para llevar a cabo las tareas concretas que tengan que realizar los robots, y para cada escenario serán diferentes. Ejemplo de funciones que se podrían incluir serían funciones que sirvan para aportar el voto o la opinión sobre cierta decisión, o una función que amplíe el conocimiento general del enjambre, indicando información recopilada sobre cierto punto del entorno, por ejemplo. También podrían haber otras funciones que ejecuten la decisión conjunta o la estrategia una vez que haya sido elegida, o que consulten cierta información que han recolectado otros miembros del enjambre.

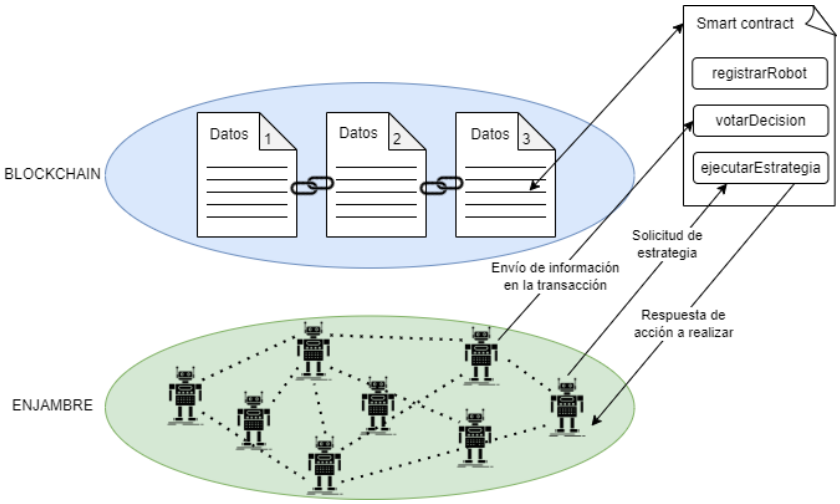

Figura 5.7: Ejecución de acciones mediante contratos inteligentes

En el anterior esquema se muestra un ejemplo muy simplificado de cómo podría funcionar la comunicación con los contratos. Se supone un sistema en el que los nodos tienen que votar ciertas decisiones en función del entorno que les rodea y actuar ejecutando una estrategia decidida por los votos de todos los nodos.

En primer lugar, un nodo enviaría una transacción (o llamada) a la función `votarDecision` del contrato para mandar su voto. Los contratos pueden tener un estado interno que puede ser modificado, y sus funciones pueden devolver o no valores. En este caso, el estado del contrato sería modificado, teniendo en cuenta el voto que ha realizado el nodo. Una vez fuesen recopilados los votos, o de forma periódica, los nodos harían una llamada a la función que les indica la próxima acción a realizar, que en este caso es la de `ejecutarEstrategia`. Esta función consultaría el estado del contrato con todos los votos y devolvería una respuesta, pudiendo ser diferente para cada uno de los nodos.

5.3.5 Elementos criptográficos

Diseñar e implementar una blockchain desde cero no es un trabajo sencillo. Por ello, una buena opción sería utilizar alguna implementación ya existente para dar soporte al sistema. No obstante, si se quisiese hacer este desarrollo desde el principio para tener un sistema completamente controlado en cuanto a características y propiedad, a continuación se nombrarán algunos elementos y algoritmos criptográficos que podrían usarse de forma ideal.

Dentro de la tecnología blockchain se hace uso de criptografía de dos tipos: las funciones hash (o funciones resumen) y la criptografía de clave asimétrica (o de clave pública).

Como se ha comentado, las funciones resumen son utilizadas, principalmente, para garantizar la integridad de la cadena y de los datos, calculando el hash de los bloques para comprobar que no haya habido la más mínima modificación. Por ello, algo esencial es el uso de una función hash cuyo correcto funcionamiento esté probado en la actualidad y que tenga las mejores propiedades posibles en cuanto a uniformidad, eficiencia y mínima colisión. Algunos ejemplos de funciones que ya no cumplen estos requerimientos y que han sido considerados obsoletos son Message-Digest Algorithm 5 (MD5) o Secure Hash Algorithm 1 (SHA1), ya que se han conseguido encontrar métodos de atacarlos de forma satisfactoria. Sin embargo, versiones posteriores a SHA1, sí son consideradas robustas en la actualidad.

Secure Hash Algorithm 3 (SHA3) se trata de la tercera y última versión de esta familia de funciones hash. Esta versión introduce mejoras respecto a su predecesora en cuanto a resistencia a ataques y rendimiento, además de introducir un diseño interno completamente diferente. Dispone de cuatro longitudes de bits diferentes: 224, 256, 384 y 512, siendo usada con más frecuencia la de 256 bits. Por ser un algoritmo actual y cuya seguridad y fiabilidad ha sido comprobada, se usará este algoritmo para el cálculo de los hashes dentro del sistema propuesto.

En cuanto a la criptografía de clave pública, esta es empleada dentro de la blockchain con la función de autorización y verificación de identidad. Para este tipo de criptografía también existen diversos algoritmos entre los que elegir, pero en este caso se ha elegido utilizar el

algoritmo Elliptic Curve Digital Signature Algorithm (ECDSA). Pese a que el algoritmo de curvas elípticas sea menos adoptado que otros como , tiene ventajas como que necesita menos longitud de clave para proveer el mismo nivel de seguridad o que tiene un mejor rendimiento. Como curiosidad, es el algoritmo utilizado para la blockchain de Ethereum.

5.3.6 Límites de seguridad

Se ha propuesto un diseño de sistema que se considera válido para la integración de blockchain y robótica de enjambre. Sin embargo, es importante ser conscientes de las posibles vulnerabilidades o los vectores de ataque que puede tener un perpetrador a la hora de intentar arremeter contra el sistema. Por ello, en este apartado se nombrarán algunas de estas sensibilidades que se podrían encontrar en cuanto a seguridad.

Debido a que el sistema está completamente montado sobre blockchain, las ventajas y los inconvenientes en cuanto a seguridad presentes serán los que conlleven consigo el uso de esta tecnología, suponiendo un uso y administración seguros de los elementos y funciones criptográficos. Por lo tanto, ahora se comentarán algunas de las principales consideraciones del blockchain en cuanto a su seguridad, valorando la importancia y la gravedad que tienen en este caso.

- **Ataques del 51%**. Este tipo de ataques se producen cuando un atacante consigue más de la mitad de la capacidad computacional a la hora de añadir nuevos bloques dentro de una blockchain. Este ataque no es preocupante en este caso, ya que es una vulnerabilidad derivada del uso del algoritmo PoW, que no es utilizado en este sistema.
- **Entidad central de autorización**. Si se usase una blockchain privada, existiría una única entidad central encargada de aceptar las peticiones de ingreso a la blockchain por parte de los nodos. Esta centralización podría llegar a desencadenar que un atacante consiguiese autorizar a nodos externos maliciosos, conocidos en este caso como robots bizantinos. Existen estudios dedicados específicamente al tratamiento de este tipo de riesgo [12].
- **Vulnerabilidades en los contratos**. Los *smart contracts* no dejan de ser trozos de código y funciones escritos en ciertos lenguajes de programación. Por lo tanto, como cualquier código, pueden estar programados de forma incorrecta o insegura de forma que puedan ser explotados por un atacante, por lo que debe tratarse con mucha atención este código. Un ejemplo de esto fue lo ocurrido en el caso de DAO[2].
- **Ataques DDoS**. La red o los nodos que la componen podrían ser víctimas de un ataque de denegación de servicio distribuido, de forma que se congestione la comunicación inundando la red o los nodos con peticiones. Por ello, será necesario implementar medidas que puedan detectarlo y evitarlo.

[2]https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/

6 Entorno de prueba

El principal objetivo de este proyecto era realizar un estudio sobre la viabilidad de integrar las dos tecnologías presentadas. De todas formas, se ha considerado que sería interesante hacer un pequeño despliegue en un entorno de prueba para ilustrar de forma práctica el concepto presentado.

6.1 Esquema del escenario

Debido a que se trata de un escenario de prueba y a que los recursos son limitados, se desplegará un ejemplo en miniatura con una tarea muy simple, pero el mismo sistema sería escalable a labores más complejas. A continuación, se muestra un esquema visual del entorno que se montará para tener una idea inicial.

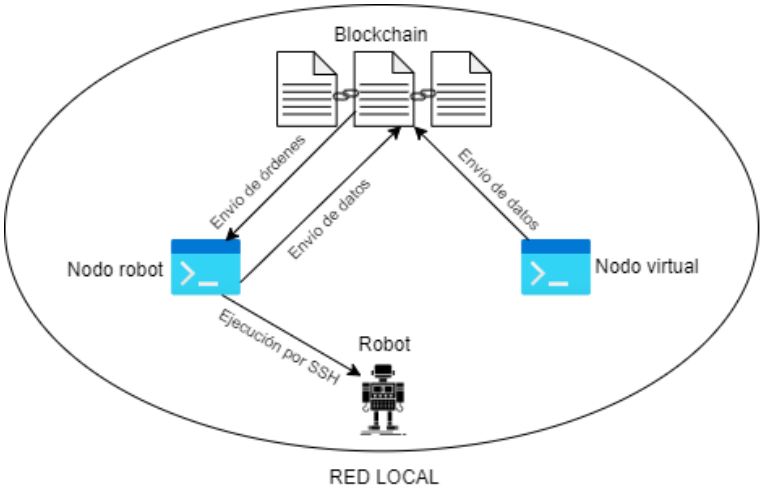

Figura 6.1: Esquema del escenario de prueba

Para comenzar, todos los elementos se encontrarán en una misma red local para que puedan ser accesibles entre sí. La blockchain contendrá el contrato inteligente con las funciones necesarias para llevar a cabo la tarea, y a ella accederán dos nodos diferentes. Un nodo, que en el esquema se le ha llamado nodo virtual, simplemente enviará datos simulando lo que haría un robot al detectar características del entorno. El segundo nodo, el nodo robot, también enviará datos, pero además se encargará de recibir las órdenes sobre decisiones que hayan sido tomadas en el *smart contract* y tengan que ser ejecutadas en el robot.

De forma ideal, el nodo robot estaría corriendo directamente en el robot físico pero, debido a las limitaciones que tiene el modelo de robot que se dispone (64MB de RAM), no es capaz de ejecutar los procesos necesarios para interactuar con la cadena y realizar las operaciones por falta de memoria. Para solventar esta problemática, la idea que se ha pensado es tener este nodo corriendo en una máquina que sea la encargada de recibir las órdenes del contrato. De esta forma, cada vez el nodo reciba estas órdenes se encargará de iniciar una sesión SSH con el robot, ejecutando un programa cuyos parámetros serán las instrucciones recibidas, de forma que el robot actúe en consecuencia.

6.2 Herramientas utilizadas

En este punto se llevará a cabo una introducción de todas las herramientas de las cuales se ha hecho uso para montar la prueba de ejecución.

6.2.1 Ethereum

El elemento con más peso dentro del sistema es la blockchain sobre la cual se llevará a cabo todo el flujo de información. El diseño y desarrollo completo de un sistema de cadena de bloques es una tarea muy compleja si se desea hacer de una forma correcta, por lo que se ha decidido trabajar sobre uno ya existente. En este caso, la plataforma que se ha seleccionado ha sido Ethereum.

Ethereum fue creada en 2015 por parte de un programador llamado Vitalik Buterin, con la finalidad de confeccionar un instrumento en el cual desplegar aplicaciones descentralizadas y colaborativas. Pese a que su aplicación más famosa en la actualidad es la de las criptomonedas, puede ser usada para todo tipo de propósitos. Además, sigue un modelo de desarrollo de código abierto, lo que resulta ideal para la elaboración de proyectos personalizados y, por lo tanto, para esta prueba.

Otra de las razones por las cuales se ha escogido es porque una de sus bases fundamentales es la introducción de los *smart contracts*, un elemento importante para el diseño de sistema que se ha propuesto con anterioridad para la integración con enjambres de robots. Un concepto relacionado con este tema es la Ethereum Virtual Machine (EVM), un programa que tiene como finalidad servir de capa de abstracción al ejecutar el código almacenado en la cadena, de forma que un atacante no pueda atentar contra la red o los nodos que la componen. Con este objetivo, lo que hace la Máquina Virtual de Ethereum es controlar el acceso a los recursos de las máquinas y limitar sus acciones en una máquina virtual. Otro propósito que tiene es facilitar el desarrollo de las llamadas DApps o aplicaciones descentralizadas. En definitiva, EVM hace posible funcionar los contratos inteligentes y las DApps a través del uso de Solidity.

Por estos motivos se ha decidido desplegar el sistema sobre una red Ethereum, pero para ello será necesaria una implementación que se pueda utilizar. Por ello, se ha hecho uso de Geth.

6.2.2 Geth

Geth es una de las tres implementaciones originales del protocolo Ethereum, junto a las de C++ y Python. Está escrita en el lenguaje de programación Go y es completamente de código abierto. Una vez que un ordenador comience a ejecutar Geth, dicha máquina se convertirá en un nodo de Ethereum.

Por otro lado, gracias a Geth también es posible el despliegue de redes privadas, de forma que solo las cuentas autorizadas puedan tener conexión con la cadena. Toda la información sobre la creación de cuentas y configuración de redes privadas con sus nodos será expuesta con posterioridad.

También cabe comentar que Geth dispone de dos algoritmos de consenso disponibles para su elección. En primer lugar, existe un algoritmo llamado Ethash que implementa *proof of work* y que permite el acceso al proceso de minería a cualquier nodo que tenga la intención de participar y prestar sus recursos computacionales para esta tarea. Se trata de un algoritmo correcto para su uso en una red pública donde puede acceder cualquiera, y la dificultad de minería se ajusta de forma automática, de forma que se haga más complicado crear un nuevo bloque a medida que aumenten los recursos existentes en la red.

El otro algoritmo de consenso posible tiene el nombre Clique, e implementa *proof of authority*, que es un algoritmo elegido durante el diseño del sistema como válido para las características que se desean. De hecho, es ideal para redes privadas como la que se quiere desplegar, y los nodos que firman las transacciones y los bloques son autorizados y desautorizados mediante un mecanismo de votación. Además, se evita el consumo de recursos que supone el proceso de minería, siendo mucho más ligera su ejecución. Es por todo esto que se elegirá este algoritmo a la hora de montar la red y sus características, como se verá más tarde.

6.2.3 Go

Go, o también conocido como Golang, es un lenguaje compilado de código abierto y de tipado estático introducido en 2009 y diseñado por Rob Pike, Robert Griesemer y Ken Thompson con la intención de crear un lenguaje que mejorase la productividad en la era de las redes de computadores y multiprocesamiento. Algunas características que destacan de este lenguaje de programación y por las cuales se ha seleccionado para la implementación son:

- Una librería estándar muy potente con un gran abanico de funcionalidades y herramientas implementadas por defecto, como funciones criptográficas de todo tipo, por ejemplo.
- Recolector de basura, por lo que se evitan problemas de memoria y desbordamiento de búfer.
- Facilidad de uso y simplicidad.
- Gran escalabilidad.

- Compilación y ejecución muy rápidas.

Dentro del despliegue de prueba, este lenguaje se usará para escribir los programas que correrán cada uno de los nodos de la red, encargados de las siguientes funciones:

- Comunicarse con el entorno y recoger los valores necesarios.
- Interactuar con los contratos inteligentes desplegados en la blockchain para enviar los datos obtenidos y recibir las órdenes a ejecutar.
- Poner en marcha las instrucciones recibidas y actuar en consecuencia a ellas.

6.2.4 Solidity

Anteriormente se ha comentado que la Máquina Virtual de Ethereum hace uso de Solidity para posibilitar el funcionamiento de los *smart contracts* y las DApps, pero no se ha explicado lo que es en sí.

Solidity es un lenguaje de alto nivel creado en 2014 por distintos colaboradores de Ethereum, orientado a la escritura de contratos y dirigido en específico a la máquina virtual de Ethereum. Su sintaxis se podría asimilar a la de JavaScript pero, a diferencia de este, presenta un tipado fuerte cuando hay que declarar tipos de argumentos y variables.

Los contratos escritos mediante este lenguaje son programados siempre en un ordenador, pero para su ejecución se debe hacer uso de la EVM, siendo desplegados en una red Ethereum de una forma descentralizada y distribuidos entre todos los nodos. Una herramienta utilizada para ayudar en este proceso de despliegue y publicación de los contratos es la que se explica en el siguiente punto.

6.2.5 Truffle

Truffle es un entorno de desarrollo y un conjunto de herramientas introducidas por Tim Coulter en 2015, orientadas a contratos inteligentes y la elaboración de aplicaciones sobre una blockchain, haciendo uso de la Máquina Virtual de Ethereum. Este marco de trabajo proporciona un entorno en el que desarrolladores de contratos puedan escribir, implementar y probar código destinado a aplicaciones distribuidas basadas en la blockchain de Ethereum.

En el marco de este proyecto, Truffle se ha utilizado para varias tareas. Una de ellas es la compilación de los contratos escritos en Solidity, produciendo como salida un código en Go que sirve como interfaz para interactuar con ellos mediante el propio lenguaje Go. La otra funcionalidad para la que se ha requerido el uso de esta herramienta ha sido la de publicar los contratos en la red privada. Para ello, Truffle se encarga de crear una transacción que tiene como fin publicar en la blockchain el contrato, devolviendo la dirección en la que se ha publicado para poder interactuar con el mismo posteriormente. Más adelante se mostrará un ejemplo de este funcionamiento.

6.2.6 LEGO Mindstorms EV3 y ev3dev

Como caso de uso concreto para el despliegue de prueba se ha decidido hacer uso del robot educativo Mindstorms EV3 de LEGO. Está compuesto de un dispositivo central, llamado ladrillo, que será el encargado de coordinar todas las tareas del robot. Al ladrillo se le podrán conectar hasta un máximo de cuatro motores y hasta un máximo de cuatro sensores, que serán los encargados de realizar el movimiento del robot y recibir la información del entorno.

Figura 6.2: Ladrillo de LEGO Mindstorms EV3

Mindstorms EV3, por defecto, permite ejecutar programas creados o añadidos mediante las herramientas que proporciona LEGO, pero en este caso será necesario poder ejecutar programas personalizados y creados de forma externa al programa de LEGO. Con este objetivo, se hará uso de ev3dev, un sistema operativo para este modelo de robots basado en Debian y que proporciona soporte para distintos lenguajes de programación para poder trabajar con el robot de una forma más personalizada. Para utilizarlo, será necesario tener la imagen del sistema operativo en una tarjeta microSD e introducirla en el robot antes de iniciarlo.

6.3 Proceso de despliegue

Ahora que ya se conocen todos los elementos que formarán parte del entorno de prueba, se procederá a comentar todos los componentes y los pasos que se han seguido para el despliegue del escenario.

6.3.1 Creación de las cuentas

El paso fundamental y que se ha llevado a cabo en primer lugar es la creación y configuración de la red blockchain que sustentará al sistema. Para ello, lo primero será instalar Geth en la máquina que va a hacer como nodo de lanzamiento, es decir, el nodo que ayudará

a inicializar y descubrir al resto dentro de la red P2P. En este caso se hará la instalación de Geth para Linux, ya que se desplegará en máquinas con sistema operativo Linux.

Para comenzar, se creará la cuenta asociada al robot, de forma que pueda interactuar con la blockchain. Para ello, se creará un directorio `RobotNode` para posteriormente usarlo para guardar toda la información necesaria, inicializando la cuenta de la siguiente forma:

```
$ geth --datadir RobotNode account new
```

Este comando pedirá una *passphrase* que servirá para asegurar la cuenta y, una vez sea introducida, devolverá la dirección del nuevo nodo, indicando el archivo donde se guardará la información necesaria para obtener su clave privada. La dirección tendrá un formato como el siguiente: `0xf92384d8573fac89b8c76e1c2be9da89c6419b01`.

Habrá que hacer lo mismo para el resto de nodos que se deseen crear, pero en un escenario real como el diseñado, todo este proceso tendría que estar automatizado mediante código.

6.3.2 Configuración de la blockchain

Una vez se hayan creado todas las cuentas necesarias, el siguiente paso será crear el archivo de génesis, que servirá para inicializar el nodo génesis de la blockchain. Para ello, en Geth existe una herramienta llamada `puppeth` que sirve para este objetivo. El programa pedirá información acerca de la red que se desea crear, pidiendo datos como:

- El nombre de la nueva red, a la que se le ha dado el nombre `swarmnet`.
- Qué se desea realizar, que en este caso será configurar un nuevo archivo génesis.
- El algoritmo de consenso que se quiere emplear, que será Clique como ya se comentó anteriormente.
- Las direcciones de los nodos que tienen permiso de autorizar transacciones y bloques desde un principio.
- El identificador de la red, necesario para poder acceder a ella posteriormente, que se ha establecido en 1515.

Una vez se haya configurado todo habrá que exportar el archivo, que resultará en un JSON con toda la información necesaria para inicializar la red.

6.3.3 Inicialización de los nodos

Llegado a este punto será necesario inicializar los nodos con el archivo de génesis que se acaba de crear. Esto se hará con el comando que se muestra a continuación, que habrá que ejecutarlo para todos los nodos:

```
$ geth geth --datadir RobotNode init swarmnet.json
```

Además, habrá que crear un nodo de arranque que servirá para que los nodos se descubran entre sí. Para ello, será necesario crear un identificador de este nodo, que se hará de la siguiente forma:

```
$ bootnode -genkey bootnode.key
```

6.3.4 Ejecución de los nodos

Ahora que ya se tienen los nodos inicializados y el nodo de arranque disponible, ya se puede probar a ejecutar los nodos para ver si funcionan correctamente. Para ello, lo primero será lanzar el `bootnode`, como se puede ver a continuación:

```
$ bootnode -nodekey bootnode.key -verbosity 9 -addr :30310
```

De esta forma, el servicio del nodo de arranque ya estará corriendo y se podrá proceder a lanzar el resto de nodos, lo cual se hará de la siguiente manera:

```
$ geth --datadir RobotNode --syncmode 'full' --port 30311 --http
↪  --http.addr 'localhost' --http.port 3334 --http.api
↪  'personal,db,eth,net,web3,txpool,miner' --bootnodes
↪  'enode://507c93ed35a3cc0d9526d504940781698c0b810@127.0.0.1:0?discport=30310'
↪  --networkid 1515 -unlock '0xfbdaa5dcf6ea0323359a1abb20659868351e86a6'
↪  --password RobotNode/password.txt --mine --allow-insecure-unlock
↪  --rpc.gascap 0 --miner.gasprice 0
```

Se trata de un comando con un gran número de argumentos. Algunos de ellos se comentan a continuación:

- `datadir`: directorio para las bases de datos y las claves del usuario.
- `http`: habilita el servidor HTTP-RPC. En un escenario real, si la comunicación entre los nodos se realizase mediante HTTP, sería necesario el uso de Transport Layer Security (TLS) para asegurar el envío de datos.
- `http.api`: APIs ofrecidas en el servidor.
- `bootnodes`: URL del nodo de arranque para poder hacer el descubrimiento de nodos.
- `mine`: habilita la validación de bloques en el nodo.
- `miner.gasprice`: establece el precio a pagar por minar una transacción. Se establece a 0 porque no es necesario en este caso.

6.3.5 Desarrollo de contratos inteligentes

Una vez se ha comprobado que se pueden ejecutar todos los nodos correctamente, es hora de escribir los contratos que servirán de base para la ejecución de funcionalidades del sistema. Para ello, se crea una carpeta llamada `Truffle` y se inicializa el proyecto.

Esto creará una estructura de directorios y archivos de configuración, donde habrá que escribir los contratos en la carpeta `contracts`. En este caso, el contracto escrito para el

ejemplo es muy simple. Únicamente tendrá una variable en el estado que será un total que podrá ser modificado mediante llamadas a la función de incremento. Si cuando el robot solicite la orden a realizar este valor es par, la orden será girar el robot a la derecha. En cambio, si el valor total es impar, el robot girará a la izquierda. El contrato de ejemplo empleado se puede encontrar aquí[1].

6.3.6 Despliegue de los contratos

Escrito el contrato, el siguiente paso será desplegarlo en la blockchain. Tras configurar Truffle con los valores de la blockchain desplegada, se puede ejecutar la migración para introducir el contrato en la cadena y que sea validada la transacción. Una vez se añade a la blockchain, se muestran datos como el hash de la transacción o la dirección del contrato, necesario para interactuar con este posteriormente.

Figura 6.3: Introducción del contrato en la cadena

6.3.7 Interacción con los contratos

En primer lugar, para la interacción con los contratos será necesario generar una interfaz para poder trabajar con ellos desde el código Go. Para ello, se hace uso de la herramienta `Application Binary Interface (ABI)`.

```
$ abigen --sol=RobotSwarm.sol --pkg=main --out=robotSwarm.go
```

Los programas que ejecutarán cada uno de los nodos irán incrementando el valor total de forma aleatoria y el robot irá solicitando las órdenes para ejecutar. Hay que tener en cuenta la frecuencia de escritura de bloque que hay en la blockchain, algo que puede hacer que no se incremente el valor a tiempo para solicitar la orden porque aún no haya podido ser escrita la transacción. El código completo utilizado para el despliegue se puede encontrar en el repositorio de GitHub [13]. De nuevo, esto no es más que un ejemplo muy pequeño de lo

[1]https://github.com/magp314/BlockchainSwarmTest/blob/main/Truffle/contracts/RobotSwarm.sol

que se podría hacer con este tipo de sistemas, pero las posibilidades que se podrían llegar a conseguir son mucho mayores y complejas.

6.4 Prueba de ejecución

Una vez explicado el proceso necesario para montar el entorno, se va a proceder a mostrar un resultado de su ejecución.

6.4.1 Despliegue final

En primer lugar, se lanzan todos los nodos de Geth en una máquina virtual o en varias, que simulan los diferentes robots del sistema. Además, será necesario guardar en el robot el ejecutable que se encargará de rotar el robot en función de la orden recibida. Para ello, indispensable compilarlo teniendo en cuenta que el sistema operativo del robot EV3 tiene una arquitectura ARMv5.

```
$ GOOS=linux GOARCH=arm GOARM=5 go build ./robot.go
```

Adicionalmente, habrá que ejecutar los dos programas de Go que se encargarán de ir haciendo el envío de datos y, en el caso del nodo robot, recoger las instrucciones para enviarlas al robot mediante SSH y que este las ejecute.

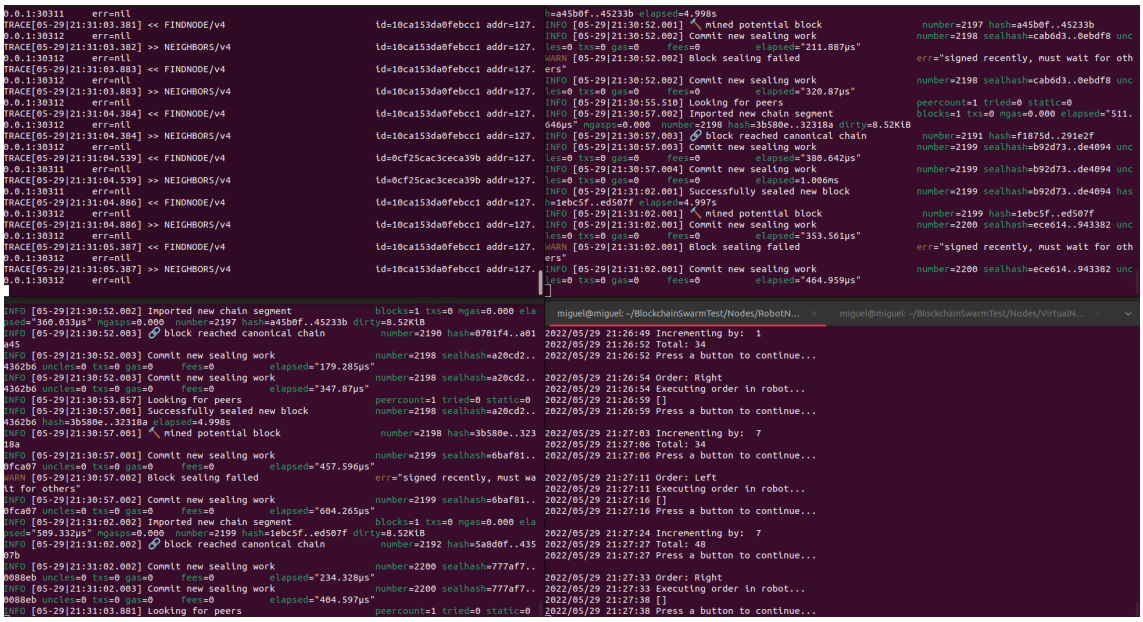

Figura 6.4: Captura de los procesos desplegados

En la anterior captura se puede observar el escenario montado. Los elementos que se pueden ver en la imagen son:

- Arriba a la izquierda, el nodo de arranque encargado de hacer que los demás nodos puedan descubrirse entre sí.
- Arriba a la derecha, el nodo de Ethereum asociado al robot.
- Abajo a la izquierda, el nodo de Ethereum encargado únicamente de enviar datos.
- Abajo a la derecha, el programa en Go encargado de interactuar con el contrato, enviando datos y recibiendo órdenes, además de ejecutarlas en el robot.

Algo a destacar de la captura es el problema de la latencia comentado anteriormente. En la esquina inferior izquierda se puede ir viendo el incremento y el valor total que tiene el estado del contrato, y posteriormente recibir la orden que tiene que realizar y que se ejecuta en el robot.

En un punto se puede ver cómo el valor se tendría que incrementar en 7, pero al obtener el total del contrato sigue devolviendo 34. Sin embargo, posteriormente la orden es girar a la izquierda, es decir, el total es impar a la hora de solicitar la orden. Esto es debido al factor de la latencia comentado con anterioridad, y es que los datos no son escritos de forma instantánea, sino que tienen que ser validados por los nodos para poder incluirse en la cadena.

6.4.2 Vídeo demostrativo

Adicionalmente, para tener una demostración en vivo del funcionamiento del test realizado, se puede acceder al vídeo[2] adjuntado a pie de página.

En el vídeo se puede ver el mismo despliegue comentado anteriormente, con los diferentes nodos y el programa Go. En primer lugar, se lanza el nodo de arranque y posteriormente se ejecutan el nodo robot y el nodo virtual, tal y como se presenta en el esquema de la sección 6.1. Hecho esto, se puede ejecutar el programa encargado de comunicarse tanto con la blockchain como con el robot, ubicado en la esquina inferior derecha.

Una vez todo está lanzado, el programa incrementará el total almacenado y lo mostrará por pantalla. Al recibir una entrada del teclado, el nodo envía una transacción solicitando una orden al contrato. Esta transacción se puede ver reflejada en la esquina superior derecha cada vez que se realiza, mostrando el texto `Submitted transaction`. Una vez se recibe la instrucción que tiene que realizar el robot, inmediatamente se establece una sesión SSH que ejecuta el movimiento que ha recibido, como se puede comprobar en la cámara colocada abajo a la izquierda.

[2] https://youtu.be/3Cx$_h$bXtC − I

7 Conclusiones

Finalmente, una vez completado el desarrollo del proyecto y viendo el resultado y el alcance al que se ha llegado, se van a tratar de proponer algunas conclusiones que se han obtenido y posibles o futuras líneas de investigación y ampliación del tema.

El propósito principal que se había puesto para este proyecto era el estudio sobre la seguridad y los problemas en el caso concreto de la comunicación dentro de enjambres de robots. Como medio y solución para tratar esta problemática se propuso el estudio de la posibilidad de la unión con una de las tecnologías más populares de la actualidad como es el blockchain. Para ello, se había planteado llevar a cabo un estado del arte de ambas para tener un conocimiento previo necesario para poder abordar su integración. Posteriormente, habría que analizar los desafíos en cuanto a la seguridad de las comunicaciones en los enjambres y ver de qué forma blockchain puede ayudar a solventarlos. Una vez hecho esto, el siguiente paso sería el diseño de un sistema que fuese capaz de integrar las dos tecnologías de la mejor manera y de forma que sea lo más viable posible. Finalmente, para ilustrar todos los conceptos teóricos presentados, la idea era hacer el montaje de un entorno de prueba donde ver de forma práctica el funcionamiento general de una forma sencilla.

Una vez se ha llegado a este punto final del proyecto, podría decirse que se han cumplido todos los objetivos que se propusieron en un principio. En primer lugar, se ha hecho una introducción sobre los enjambres de robots y el blockchain que sirve como base para entender temas posteriores. A su vez, se han presentado todos sus elementos y funcionamiento, así como sus virtudes y sus desventajas, de forma que se pueda establecer de forma crítica un diseño del sistema, realizado con posterioridad. Por último, se ha conseguido solventar las dificultades encontradas a la hora de lanzar el entorno de prueba, de forma que se ha podido comprobar de forma empírica, aunque muy simplificada, el funcionamiento que tendría realmente.

Pese a que se hayan cumplido todos los objetivos propuestos para el alcance de este proyecto, este tipo de sistemas tienen una complejidad que no puede ser tratada en profundidad en un solo trabajo. En la actualidad existen muchos estudios sobre la aplicación de blockchain a muchos ámbitos, pero en concreto a la robótica de enjambre también existen con diversas líneas de investigación y desarrollo. Es un área que permite un amplio abanico donde indagar, desde el estudio de las formas de construir el conocimiento del enjambre hasta la reflexión sobre el tema de la latencia y cómo aminorarla lo máximo posible. En lo estrictamente relativo a la seguridad, también existe exploración sobre problemáticas como la introducción de robots maliciosos dentro de enjambres y la tolerancia a fallo que hay ante estos mediante el uso del blockchain.

En cuanto a conclusiones obtenidas durante todo el trascurso y el desarrollo del trabajo,

podrían nombrarse unas cuantas. Para empezar, se podría decir que la comunicación dentro de los enjambres de robots es un aspecto muy importante y que cabe tratar con sumo cuidado, ya que una vulnerabilidad aprovechada en redes de este tipo podría suponer graves consecuencias, en función de la misión en concreto. Por otro lado, el blockchain es una tecnología muy novedosa y aclamada en la actualidad, razón por la cual puede llegar a ser usada de forma incorrecta, ya que no es apta para usarse en todo tipo de escenarios.

Sin embargo, dada la naturaleza distribuida de la robótica de enjambre y los beneficios que aporta blockchain en este tipo de redes entre iguales, se ha visto que la integración de ambos elementos puede ser una opción más que viable. Como se ha visto, gracias a esta unión se solventan muchos de los problemas existentes, haciendo uso de las propiedades que aporta blockchain.

No obstante, aunque sea una buena opción siempre existen desventajas y puntos débiles. Siempre es necesario estudiar el caso concreto que se quiera desplegar, ya que no en todos los casos hacer uso de blockchain va a ser la mejor opción, debido a que desplegar una red de este tipo puede no ser lo más barato y en casos en los que los requerimientos de seguridad no sean muy altos puede haber soluciones más sencillas. Por todo esto, uno de los mayores retos de este trabajo ha sido encontrar una forma lógica mediante la cual se pueda hacer uso de una cadena para mantener un sistema de robótica de enjambre y las operaciones que supone, manteniendo unos niveles de seguridad alto gracias a las propiedades del blockchain.

Otra de las incertidumbres o contras que puede tener su uso es un tema que ya ha sido comentado, y es el de la latencia. Debido a la necesidad de aprobación de todas las operaciones por parte de los nodos de la red, la velocidad con la que llega la información a los componentes puede llegar a ser más lenta de lo deseado, en el caso de que se trate de una tarea donde la capacidad de reacción y la velocidad en la toma de decisiones sean esenciales.

Por todo lo anterior, se deberían poner en una balanza varios elementos para valorar cómo de necesario o de factible es la aplicación de un sistema de este tipo. En resumidas cuentas, algunos de los elementos que se podrían valorar a la hora de la decantación podrían ser el nivel de seguridad requerido por el sistema, el coste que se puede permitir en el despliegue o las necesidades en cuanto a velocidad de comunicación. De esta forma, se podría terminar concluyendo que se trata de una solución que resuelve muchos de los problemas en cuanto a seguridad en redes entre iguales como son los enjambres de robots, pero siempre será necesaria una evaluación de cada caso concreto.

Bibliografía

F.M. Sánchez-Martín, P. Jiménez Schlegl, F. Millán Rodríguez, J. Salvador-Bayarri, V. Monllau Font, J. Palou Redorta, and H. Villavicencio Mavrich. Historia de la robótica: de Arquitas de Tarento al Robot da Vinci (Parte II). *Actas Urológicas España*, 31:185 – 196, 03 2007.

Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In *Robots and biological systems: towards a new bionics?*, pages 703–712. Springer, 1993.

Toshio Fukuda and Seiya Nakagawa. Approach to the dynamically reconfigurable robotic system. *Journal of Intelligent and Robotic Systems*, 1(1):55–72, 1988.

Lynne E Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. In *AAAI fall symposium: regarding the intelligence in distributed intelligent systems*, pages 1–6, 2007.

Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. In *Conference on the Theory and Application of Cryptography*, pages 437–455. Springer, 1990.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 1:22–23, 2013.

Fiona Higgins, Allan Tomlinson, and Keith M Martin. Survey on security challenges for swarm robotics. In *2009 Fifth International Conference on Autonomic and Autonomous Systems*, pages 307–312. IEEE, 2009.

Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, 154:113385, 2020.

Nick Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.

Volker Strobel, Eduardo Castelló Ferrer, and Marco Dorigo. Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. 2018.

Miguel Ángel García. Blockchain swarm test. `https://github.com/magp314/BlockchainSwarmTest`, 2022.

Heiko Hamann. Swarm robotics: A formal approach. 2018.

Ahmad Reza Cheraghi, Sahdia Shahzad, and Kalman Graffi. Past, present, and future of swarm robotics. In *Proceedings of SAI Intelligent Systems Conference*, pages 190–233. Springer, 2021.

Liqun Chen and Siaw-Lynn Ng. Securing emergent behaviour in swarm robotics. *Journal of Information Security and Applications*, 64:103047, 2022.

Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

Jean-Paul A Yaacoub, Hassan N Noura, Ola Salman, and Ali Chehab. Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations. *International Journal of Information Security*, pages 1–44, 2021.

Lista de Acrónimos y Abreviaturas

ABI	Application Binary Interface.
ECDSA	Elliptic Curve Digital Signature Algorithm.
IoT	Internet of Things.
MD5	Message-Digest Algorithm 5.
P2P	Peer-to-peer.
PoA	Proof of Authority.
PoS	Proof of Stake.
PoW	Proof of Work.
SHA1	Secure Hash Algorithm 1.
SHA2	Secure Hash Algorithm 2.
SHA3	Secure Hash Algorithm 3.
TLS	Transport Layer Security.