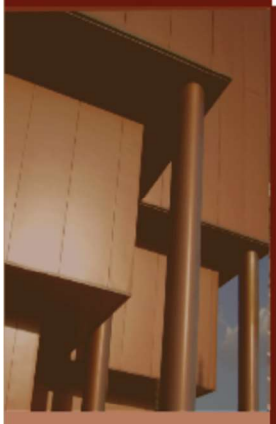
Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Diseño y desarrollo de una plataforma de comunicaciones descentralizada con WiFi Direct y protocolos M2M

Autor: Fausto Botello Jannone
Tutores: Luis Javier Reina Tosina
Alejandro Talaminos Barroso



Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Diseño y desarrollo de una plataforma de comunicaciones descentralizada con WiFi Direct y protocolos M2M

Autor:
Fausto Botello Jannone

Tutor:
Luis Javier Reina Tosina
Profesor titular

Alejandro Talaminos Barroso

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado: Diseño y desarrollo de una plataforma de comunicaciones descentralizada con WiFi Direct y protocolos M2M

Autor: Fausto Botello Jannone

Tutor: Luis Javier Reina Tosina

Alejandro Talaminos Barroso

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

Agradecimientos

A todos los que me han acompañado estos años.

Fausto Botello Jannone
Grado en Ingeniería de las Tecnologías de Telecomunicación
Sevilla, 2023

Resumen

El aumento del número de dispositivos IoT existentes hoy en día plantea enormes desafíos en los sistemas de comunicaciones. Para satisfacer las crecientes necesidades de los usuarios y utilizar eficientemente los escasos recursos disponibles, la comunicación Device-To-Device (D2D) se considera una importante tecnología emergente para el futuro de las redes.

En este trabajo se propone un algoritmo de consenso anticipado y recurrente para la elección del mejor punto de acceso en caso de caída del nodo principal, cambios en la red e itinerancia. Se presenta el diseño y desarrollo de una plataforma de comunicaciones descentralizada que permita una comunicación eficiente y segura entre dispositivos sin necesidad de un concentrador o enrutador central, utilizando la tecnología WiFi Direct.

Para este fin, se discuten las limitaciones actuales de los sistemas de comunicación centralizados tradicionales y los beneficios de los sistemas descentralizados, haciendo especial énfasis en los algoritmos de consenso. Más adelante, la investigación se centra en el diseño y la implementación de la plataforma propuesta, incluido el uso de WiFi Direct para el descubrimiento y la conexión de dispositivos, utilizando protocolos Machine-to-Machine (M2M) para una transferencia de datos eficiente, y se trata de solventar las limitaciones encontradas.

Cabe destacar el uso de diferentes herramientas de simulación como WiDiSi o MQTTX para la obtención de unos resultados que nos hagan reafirmar las hipótesis planteadas a lo largo de esta investigación.

Por último, analizando los resultados, dado que la mayoría de los estudios se basan en la aleatoriedad para la selección del líder, se propone el diseño de un algoritmo de consenso basado en el algoritmo Raft que permita establecer criterios más útiles para conseguir una mejora de calidad en torno a la elección, en el contexto de WiFi Direct finalizando con la presentación de los resultados que muestran como la flexibilidad inherente del algoritmo proporciona una solución adaptable a diversas aplicaciones y escenarios en sistemas distribuidos abriendo nuevas posibilidades para investigaciones futuras y desarrollos en el campo de las redes distribuidas.

Abstract

The increase in the number of existing IoT devices today poses enormous challenges in communication systems. To meet the growing needs of users and efficiently use scarce resources, Device-To-Device (D2D) communication is considered an important emerging technology for the future of networks.

In this work, an advanced and recurring consensus algorithm is proposed for selecting the best access point in case of main node failure, network changes, and roaming. The design and development of a decentralized communication platform is presented, enabling efficient and secure communication between devices without the need for a central hub or router, using WiFi Direct technology.

To this end, the current limitations of traditional centralized communication systems are discussed, along with the benefits of decentralized systems, with a special emphasis on consensus algorithms. Subsequently, the research focuses on the design and implementation of the proposed platform, including the use of WiFi Direct for device discovery and connection, using Machine-to-Machine (M2M) protocols for efficient data transfer, and addressing the encountered limitations.

It is worth noting the application of different simulation tools such as WiDiSi or MQTTX to obtain preliminary results that support the hypotheses raised throughout this research.

Finally, considering the results, given that the majority of studies rely on randomness for leader selection, the design of a consensus algorithm based on Raft algorithm is proposed in the context of WiFi Direct in order to establish more useful criteria for achieving an improvement in quality regarding the selection. This culminates in the presentation of results demonstrating how the inherent flexibility of the algorithm provides an adaptable solution for various applications and scenarios in distributed systems, thereby opening up new possibilities for future research and developments in the field of distributed networks.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

ACK	**Acknowledgement**
AES	**Advanced Encryption Standard**
AMQP	**Advanced Message Queuing Protocol**
API	**Application Programming Interface**
AP	**Access Point**
ASK	**Amplitude Phase Keying**
BAS	**Bonjour Access Service**
BPSK	**Binary Phase Shift Keying**
BSS	**Basic Service Set**
CBC	**Cipher Block Chaining**
CCK	**Complementary Code Keying**
CLB	**Cluster Building**
CoAP	**Constrained Application Protocol**
CPU	**Central Processing Unit**
CNG	**Cluster Neighbourhood Gathering**
COFDM	**Coded Orthogonal Frequency Division Multiplexing**
CRC	**Cyclic Redundancy Check**
D2D	**Device to Device**
DDS	**Data Distribution Service**
DHCP	**Dynamic Host Configuration Protocol**
DNE	**Dominant Node Election**
DNS	**Domain Name System**
DTLS	**Datagram Transport Layer Security**
DTN	**Delay Tolerant Networks**
DSSS	**Direct Sequence Spread Spectrum**
DS-UWB	**Direct Sequence Ultra-Wideband**
EGO	**Emergency Group Owner**
EMC	**Efficient Multi group formation and Communication**
ESS	**Extended Service Set**
FHSS	**Frequency Hopping Spread Spectrum**
FCI	**Final Cluster Interconnection**
GFSK	**Gaussian Frequency Shift Keying**
GM	**Group Member**
GOD	**Dominant Group Owner**
GO	**Group Owner**
IBFT	**Istanbul Byzantine Fault Tolerance**
IEEE	**Institute of Electrical and Electronics Engineers**
IOT	**Internet of Things**
IP	**Internet Protocol**
LAN	**Local Area Network**
LC	**Legacy Client**

M2M	**Machine-to-Machine**
MAC	**Media Access Control**
MB-OFDM	**Multiband Orthogonal Frequency Division Multiplexing**
MQTT	**Message Queuing Telemetry Transport**
NACK	**Negative Acknowledgement**
NBIP	**Name Based Interest Profiles**
NoA	**Notice of Absence**
NoSQL	**No Structured Query Language**
OPS	**Opportunistic Power Save**
O-QPSK	**Offset Quadrature Phase Shift Keying**
PBFT	**Practical Byzantine Fault Tolerance**
PoA	**Proof of Authority**
PoS	**Proof of Stake**
PoW	**Proof of Work**
P2P	**Peer-to-Peer**
PSK	**Phase Shift Keying**
QoS	**Quality of Service**
QPSK	**Quadrature Phase Shift Keying**
RSSI	**Received Signal Strength Indicator**
SPI	**Service Plugin Interface**
SSL	**Secure Socket Layer**
TCP	**Transmission Control Protocol**
TLS	**Transport Layer Security**
UDP	**User Datagram Protocol**
URI	**Uniform Resource Identifier**
WEP	**Wired Equivalent Privacy**
WLAN	**Wireless Local Area Network**
WPA	**WiFi Protected Access**
WPS	**WiFi Protected Setup**

1 INTRODUCCIÓN

Cada día sabemos más y entendemos menos.

- Albert Einstein -

En este capítulo se presentan el contexto y los objetivos del Trabajo. Se abordan conceptos preliminares y se presentan las hipótesis que guían la investigación. Al final, se proporciona una visión general de la estructura del Trabajo, delineando los contenidos de cada capítulo.

1.1 Motivación

El uso generalizado de smartphones con tecnologías de comunicaciones avanzadas permite nuevas oportunidades no solo para los usuarios sino también para proveedores, desarrolladores y fabricantes.
A día de hoy, el número de terminales y dispositivos alrededor de todo el mundo se puede estimar en billones [1], que forman entre sí un enorme sistema de red interconectado. Cada dispositivo está conectado a otros mediante una red cableada o un enlace inalámbrico. Las conexiones inalámbricas lideran la revolución tecnológica, donde la mayoría de los dispositivos se apoyan en puntos de acceso.

Los sistemas centralizados han sido la base sólida sobre la cual se desarrolla la actual estructura de Internet. Aun así, este tipo de jerarquía cuenta con una gran desventaja. Al igual que un castillo de naipes, todo se puede venir abajo en cuestión de milésimas de segundo. Cabe destacar que el constante mantenimiento de un punto de acceso y, por ende, de una infraestructura inalámbrica, supone un esfuerzo económico importante y no siempre es posible. Es ahí donde surgen los llamados sistemas descentralizados.

En los sistemas descentralizados, donde no existe una autoridad o control central, el uso del estándar IEEE 802.11 es importante, ya que define cómo se transmiten los datos y permite la comunicación inalámbrica entre dispositivos. La tecnología WiFi es una marca de certificación que indica que un dispositivo ha sido probado y certificado para cumplir con ciertos estándares 802.11 para la comunicación inalámbrica en redes de área local.

Más de dos décadas después de su diseño inicial, WiFi se ha convertido en una de las formas más comunes de acceder a Internet. Sin embargo, para continuar con su éxito, esta tecnología necesita evolucionar y abarcar un conjunto más amplio de casos de uso. Este estándar ya era capaz de ofrecer conectividad entre dispositivos en modo *ad hoc*, pero esto nunca tuvo buena acogida en el mercado puesto que presentaba numerosos inconvenientes cuando se enfrentaba a los requerimientos que le pedían los usuarios [2]. Era hora de dar un paso más allá.

En este escenario, hacia 2010 surgió WiFi Direct, una tecnología capaz de proporcionar comunicación entre dispositivos con la cobertura y velocidad adecuada, además de contar con propiedades heredadas de la propia infraestructura WiFi como es la calidad de servicio (QoS), el ahorro de batería o los mecanismos de seguridad.

Con estas premisas, en este Trabajo Fin de Grado se pretende utilizar la tecnología WiFi Direct para la implementación de una plataforma de comunicaciones descentralizada, aprovechando sus características de conexión directa y sin necesidad de un punto de acceso intermedio, explorando las posibilidades que ofrece en el contexto de la comunicación *Machine-to-Machine* (M2M).

Su uso en sistemas distribuidos facilita la comunicación entre nodos y habilita la tolerancia a fallos al permitir que los nodos se comuniquen directamente entre sí teniendo la capacidad de seguir funcionando correctamente en caso de fallo de uno o más componentes.

Uno de los retos de los sistemas distribuidos es alcanzar el mayor grado de fiabilidad en los sistemas distribuidos. Para ello, se necesitan protocolos o algoritmos que permitan al sistema actuar como un todo para así continuar funcionando a pesar de que ocurra algún fallo. Estos protocolos requieren la cooperación entre los distintos procesos para llegar a un consenso.

La Real Academia Española define la palabra consenso como "*Un acuerdo producido por consentimiento entre todos los miembros de un grupo o entre varios grupos*". En el campo de las TIC, el algoritmo de consenso se define como una metodología o esquema que se utiliza para alcanzar una opinión o acuerdo unánime respecto a una serie de datos en un proceso ampliamente distribuido.

Los algoritmos de consenso se pueden aplicar para llevar a cabo múltiples tareas. Pueden abarcar desde la toma de decisión a la hora de hacer un compromiso de un procesamiento de datos distribuido en una base de datos hasta la designación del líder de un conjunto de puntos nodales para la aceptación de un procesamiento de datos distribuidos. La propuesta de mejoras en los algoritmos de consenso es el objetivo principal del estudio que se presenta en este Trabajo.

1.2 Teoría de grafos

"Durante el Siglo XVIII, la ciudad de Königsberg estaba dividida en cuatro zonas por el río Pregel. Había siete puentes que comunicaban estas regiones. Los habitantes de la ciudad hacían paseos dominicales tratando de encontrar una forma de caminar por la ciudad, cruzando cada puente una sola vez, y regresando al lugar de partida."

Figura 1-1 Ilustración del problema de los puentes de Königsberg [3]

Así comenzaba el estudio de la teoría de grafos presentada en 1736 en un artículo de Leonhard Euler buscando una solución para resolver lo que se denominaba como el problema de los puentes de Königsberg. Para resolverlo, se representan las cuatro zonas con puntos y los puentes como arcos, tal y como se muestra en la Fig. 1-2.

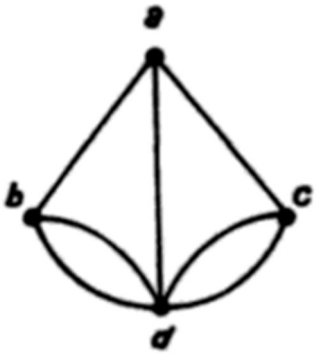


Figura 1-2 Esquema simplificado de nodos [3]

La base está en que tiene una representación gráfica muy intuitiva que le confiere un gran potencial a la hora de abordar problemas de muy distinta índole. El problema queda bastante esquematizado y simplificado si se consideran cuatro puntos (a los que denominaremos vértices) y siete arcos, que conectan algunos de los vértices. Esto dio pie a la propia definición de grafo.

Definición: Un grafo G es una pareja de conjuntos (V, E) [4], donde:

- V es distinto de vacío
- E es un conjunto de pares de elementos de V

De manera bastante natural se puede representar mediante esta teoría un sistema de comunicaciones y sus posibilidades de comunicación entre diferentes nodos. Lo más frecuente es que los puntos estén representados en el grafo mediante vértices y las posibilidades de comunicación mediante arcos. Puede darse el caso de que la comunicación esté restringida de manera que no sea posible el intercambio de información entre dos entidades cualesquiera.
Habitualmente, esta restricción viene dada por una red de comunicación que limita aquellos pares entre los que hay un canal explícito. A su vez, existen otros factores que limitan la comunicación, como el tener que compartir un espacio común, estar en cierta área de cobertura o tener conocimiento de la existencia de la otra entidad.

1.3 Sistemas descentralizados

Para comprender qué son los sistemas descentralizados y cómo funcionan, primero se debe definir lo que se conoce como centralización. En esencia, si algo está centralizado, significa que hay una sola entidad con autoridad completa sobre todo el sistema. Si se habla en el ámbito de los sistemas, se puede diferenciar entre un sistema centralizado o descentralizado.

En un sistema de distribución centralizada el nodo principal es el responsable de desglosar las tareas computacionales al igual que la distribución de la carga de datos a través de la red. En cambio, en un sistema de distribución descentralizada no hay nodo maestro, si no que cada nodo es autónomo para distribuir la carga computacional a través de la red. En la Tabla 1-1 se muestran algunas de las características principales que permiten comparar estos dos tipos de sistemas.

Características	*Sistemas Centralizados*	*Sistemas Descentralizados*
Control	Central	Distribuido
Diseño	Fácil	Difícil
Mantenimiento	Fácil	Difícil
Fallos	Único punto	Ningún punto único
Estabilidad	Baja	Alta
Vulnerabilidad	Sí	No
Comportamiento Malintencionado	Posible	Imposible
Escalabilidad	Baja	Alta

Tabla 1-1: Comparativa de características de los sistemas centralizados frente a los descentralizados [5]

En el contexto de este Trabajo interesa centrarse en los sistemas descentralizados, que surgen como alternativa y, a su vez, una manera más estable y segura de definir la jerarquía del diseño. No importa el tamaño del sistema, que puede ser minúsculo, interconectando solo unos pocos dispositivos, o tremendamente inmenso, ocupando países o incluso continentes. Ambos se enfrentan a los mismos retos: tolerancia al fallo, coste de mantenimiento y escalabilidad.

En los sistemas centralizados todos los usuarios están conectados a un servidor central que no solo gestiona las comunicaciones, sino también puede almacenar datos para que otros usuarios puedan acceder a su contenido. Estos sistemas son fáciles de implantar y se desarrollan rápidamente. Lamentablemente, esta ventaja también es un arma de doble filo, ya que, en este tipo de sistemas, al estar todos los usuarios conectados a un propietario de red central o servidor, si éste colisiona, el sistema colapsa, se cae y ya no es útil. Esta es la principal causa por la que el uso de sistemas centralizados no es la primera opción en muchas organizaciones.

Por otra parte, los sistemas descentralizados, como su propio nombre indica, no tienen un único punto centralizado donde se focaliza toda la información. Esta jerarquía cuenta con múltiples puntos propietarios. Así, cada uno de ellos almacena una copia de todos los recursos a los que puede acceder cada usuario. Este sistema puede ser igual de vulnerable para colapsar que uno centralizado, pero está diseñado para ser más tolerante a estos tipos de fallo. Esto es debido a que, cuando uno o más de los servidores principales falla, los otros siguen prestando servicio y la información se reencamina por la ruta más asequible. Los recursos seguirán siendo accesibles siempre y cuando al menos uno de los servidores centrales continúe dando servicio. Por lo general, esto ofrece al sistema la capacidad de resolver y reparar los problemas mientras se siga ofreciendo el servicio como de costumbre. El mejor ejemplo para entender este concepto es Internet en sí mismo.

Cuando se llega a una situación de colapso en este tipo de sistemas, el rendimiento puede verse afectado, pero en términos de tiempo de actividad general del sistema, sigue ofreciendo una gran mejora con respecto a los sistemas centralizados. Otra gran ventaja son las velocidades que alcanzan en el acceso a la información, ya que se pueden diseñar nodos especiales en zonas de alto tránsito para mejorar sus prestaciones.

Este tipo de diseños supone un aumento en el coste, ya que mantener un sistema así suele ser más costoso. Sin embargo, los sistemas descentralizados aún son bastante propensos a los mismos riesgos en términos de seguridad y privacidad que los sistemas centralizados.

Por último, se presentan los sistemas distribuidos. Son similares a los descentralizados, aunque se conciben siguiendo la evolución de la idea de que no existe punto de focalización, sino que desde cada punto se puede acceder a todo el sistema. Cada usuario tiene igual acceso que los demás, aunque se permite conceder o quitar privilegios cuando sea necesario. Para una mejor visualización véase la Fig. 1-3.

En los sistemas distribuidos se permite a los usuarios compartir la propiedad de los datos. Los recursos de hardware y software también se asignan entre los usuarios, lo que en algunos casos puede mejorar el rendimiento del sistema. Un sistema distribuido está a salvo de fallos producidos por uno de los componentes independientemente del resto, lo que puede mejorar considerablemente su tiempo de actividad.

Este tipo de jerarquías ha sido el resultado de la necesidad constante de evolución para resolver las limitaciones que presentaban los otros tipos de sistemas. Esto ha supuesto un incremento de la seguridad, el almacenamiento de datos o incluso el generar conciencia por temas de privacidad. Con estas características ha conseguido afianzarse como una de las principales opciones elegidas a día de hoy. Cabe destacar como las principales tecnologías están apostando por este tipo de sistemas. Un ejemplo notable lo tenemos con Blockchain, que está teniendo un impacto altamente significativo [6].

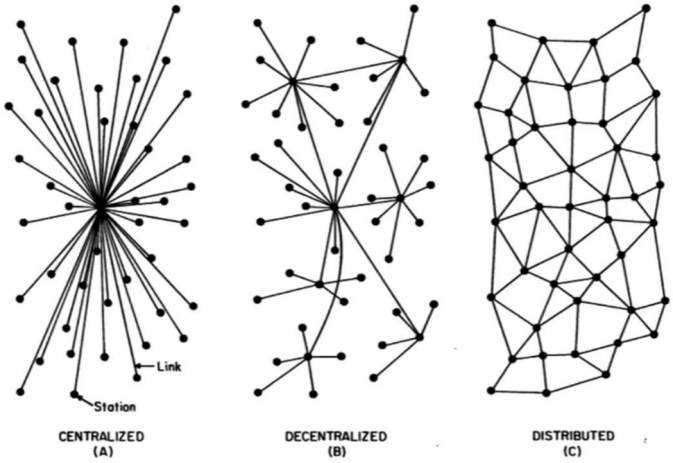


Figura 1-3 Sistema centralizado, descentralizado y distribuido [7]

Sin embargo, algunos de los inconvenientes principales que presentan este tipo de sistemas son la excesiva complejidad o un difícil mantenimiento [8]. Esta mayor complejidad puede complicar a los equipos las tareas organizativas, de gestión y mejora de los propios sistemas.

Es por este motivo que los sistemas descentralizados suelen ser la solución más flexible y a su vez robusta, puesto que no hay ninguna entidad que haga las funciones de supervisor. Un fallo de un nodo tiene efecto limitado dentro de un alcance, que dependerá en muchas ocasiones de la topología de la red.

Para lograr que una solución descentralizada funcione, debe haber algún método que garantice su corrección trabajando con lo que se denomina racionalidad acotada [9]. Esto supone la idea de que cada nodo perteneciente a la red solo dispone de su propia información y de la que le puedan proporcionar sus vecinos inmediatos.

En general, se parte de la premisa de que cada nodo tiene una visión parcial y no conoce a priori su estructura, su topología o su tamaño.

Así, se puede decir que, dado un conjunto de entidades autónomas, el problema de consenso puede plantearse mediante grafos para poder alcanzar un acuerdo en torno a una opinión, que se representará mediante un valor escalar, partiendo del conjunto inicial de opiniones de las entidades.

A este tipo de procesos se les conoce como procesos de difusión de información o de propagación de creencias. El algoritmo de consenso es uno de los métodos más utilizados para abordar estos problemas, ya que resulta esencial en los sistemas descentralizado y distribuidos para asegurar no solo un acuerdo entre todos los participantes sino también una coordinación para poder ser independiente de una autoridad centralizada.

1.4 Hipótesis y objetivos

Este Trabajo se sustenta sobre dos hipótesis fundamentales.

En primer lugar, que es posible mejorar el rendimiento de los algoritmos de consenso al establecer nuevos criterios para la selección del consenso y optimizar tanto la velocidad como la calidad de la toma de decisiones en procesos de múltiples participantes en un sistema distribuido.

La segunda hipótesis propone que el algoritmo de consenso Raft es la mejor opción para implementar en un sistema basado en la tecnología WiFi Direct. Raft es un algoritmo de consenso líder en la industria, destacando por su capacidad para proporcionar una toma de decisiones rápida y confiable en sistemas distribuidos. Además, Raft es conocido por ser fácil de entender, implementar y mantener, lo que lo hace adecuado para una amplia variedad de aplicaciones, incluyendo aquellas basadas en WiFi Direct.

La primera hipótesis se puede justificar como consecuencia de la madurez de las tecnologías de la computación, que permite la exploración y aplicación de nuevos criterios de selección del consenso en sistemas distribuidos. Este avance se ha desarrollado mediante herramientas y técnicas más sofisticadas que permiten la recolección, procesamiento y análisis de grandes volúmenes de datos en tiempo real, lo que significa que es posible considerar una amplia variedad de factores en el proceso de selección del consenso.

Además, la creciente complejidad de los sistemas distribuidos y la mayor participación de múltiples nodos en los procesos de consenso, han hecho que sea necesario implementar nuevos criterios de selección del consenso para garantizar la eficiencia y calidad en la toma de decisiones. Por lo tanto, se espera que la exploración y aplicación de nuevos criterios de selección del consenso puedan mejorar significativamente el rendimiento de los algoritmos de consenso en sistemas distribuidos y facilitar la adopción de estas tecnologías en un mayor número de aplicaciones.

La segunda hipótesis se apoya en estudios previos que han demostrado que Raft es altamente escalable y eficiente en términos de tolerancia a fallos y velocidad de transmisión en redes distribuidas. Además, la tecnología WiFi Direct ofrece una solución inalámbrica de bajo consumo y alta velocidad que se adapta perfectamente a las necesidades de transmisión de datos de un algoritmo de consenso como Raft.

El objetivo de este trabajo es diseñar y desarrollar una plataforma de comunicaciones descentralizada que permita una comunicación eficiente y segura entre dispositivos sin necesidad de un concentrador o enrutador central utilizando la tecnología WiFi Direct y protocolos M2M.

Para alcanzar este objetivo general se atenderá a los siguientes objetivos específicos:

- Identificar las limitaciones de los sistemas de comunicaciones centralizados tradicionales y los beneficios de los sistemas descentralizados, enfocándose en el estudio de los algoritmos de consenso.

- Investigar el uso de WiFi Direct para el descubrimiento y la conexión de dispositivos, así como la implementación de protocolos M2M para una transferencia de datos eficiente.

- Proponer un algoritmo de consenso anticipado y recurrente para la elección del mejor punto de acceso en situaciones de caída del nodo principal y cambios en la red e itinerancia.

- Implementar dicho algoritmo de consenso basado en Raft para la selección del líder en la plataforma propuesta y evaluar su rendimiento en comparación con la aleatoriedad en la elección del líder.

1.5 Estructura del proyecto

La memoria de este Trabajo está estructurada en capítulos, cuyos contenidos se resumen brevemente a continuación:

El capítulo I contiene una breve introducción al proyecto, incluyendo la motivación que ha llevado a desarrollar esta investigación junto a un análisis de cómo enfocar el problema empezando por la teoría de grafos y acabando en los algoritmos de consenso.

En el capítulo II se realiza una revisión de los principales algoritmos de consenso a la vez que trata de mostrar aquellos que se encuentran en el estado del arte.

En el capítulo III se estudia el uso de WiFi Direct para el descubrimiento y la conexión de dispositivos, así como la implementación de protocolos M2M para una transferencia de datos eficiente.

El capítulo IV expone los materiales utilizados durante el desarrollo del Trabajo.

En el capítulo V se presentan y analizan en detalle los resultados obtenidos con dos de los algoritmos expuestos en el capítulo II como son el algoritmo RedMesh basado en WiFi Direct y el algoritmo Raft. Para ello, se ejecutan simulaciones en entornos controlados como son WiDiSi y el IDE de Eclipse respectivamente.

El capítulo VI comienza con un análisis detallado de las propiedades fundamentales que se deben considerar para seleccionar un algoritmo de consenso adecuado. A continuación, se presenta la propuesta de un nuevo algoritmo innovador de consenso anticipado y recurrente que permite, seleccionar de manera eficiente el mejor punto de acceso en escenarios donde ocurren caídas del nodo principal y se producen cambios en la red. El capítulo finaliza con la implementación del algoritmo en grupos de WiFi Direct, utilizando protocolos M2M para la transferencia eficiente de los datos, discutiendo los resultados obtenidos en los diferentes modelos y escenarios implementados.

En el capítulo VII se presentan las conclusiones obtenidas a partir de la investigación realizada.

Por último, en el capítulo VIII se referencia toda la bibliografía utilizada para la realización de este Trabajo.

2 FUNDAMENTOS DE LOS ALGORITMOS DE CONSENSO

No puedes unir los puntos mirando hacia adelante; solo puedes unirlos mirando hacia atrás. Así que tienes que confiar en que los puntos de alguna manera se unirán en tu futuro

-Steve Jobs-

2.1 Introducción

Los sistemas centralizados ayudaron al crecimiento de las primeras redes de comunicaciones, ya que constituían la única opción. Hoy en día están casi obsoletos y los sistemas de comunicaciones se modifican continuamente para adaptarse a topologías distribuidas y al intercambio de datos entre usuarios de forma descentralizada.

Ante este escenario, surge la necesidad de poner de acuerdo a los diferentes nodos sobre la información que se comparte de manera ordenada y segura. El consenso es un concepto esencial en estos tipos de sistemas y existe una línea de investigación centrada en determinar qué algoritmo consigue llegar a un acuerdo entre los nodos de la manera más efectiva.

Los algoritmos de consenso funcionan bajo la hipótesis de la no disponibilidad de ciertos procesos y bajo la hipótesis de fallo de determinadas comunicaciones. Estos algoritmos juegan un papel notable para cuantificar la seguridad o validez de una red de datos que consta de múltiples nodos no confiables.

Se han propuesto muchos algoritmos a lo largo de los años y cada uno tiene su propio rendimiento y características. Comprender las diferentes características de cada algoritmo es crucial para poder diseñar e implementar sistemas efectivos. Es vital comparar técnicamente los algoritmos disponibles para resaltar sus fortalezas al igual que sus debilidades y sus casos de uso en la actualidad.

2.2 Definición de consenso y objetivos

Se puede definir un algoritmo de consenso como un mecanismo que permite a los nodos coordinarse en un entorno distribuido. Se debe garantizar que todos los agentes del sistema puedan ponerse de acuerdo respecto a una fuente única de verdad, incluso en el caso de que algunos de ellos fallen. O lo que es lo mismo, el sistema debe tener tolerancia a fallos [10] .

Si se habla en términos sencillos, solo es un método de toma de decisiones dentro de un grupo. Para apoyar la teoría se va a plantear un ejemplo. Supóngase que se tiene que tomar una decisión entre un grupo de diez personas sobre un proyecto que los beneficie a todos. Cada persona puede sugerir una idea, pero la mayoría estará a favor de la que más les beneficie. El resto tendrá que acatar la decisión más votada, sea de su gusto o no. Si se plantea el mismo escenario, pero escalándolo a miles de personas, supone un incremento en la dificultad de tomar una decisión. Los algoritmos de consenso no solo están de acuerdo con la mayoría de los votos, sino que también están de acuerdo con aquel que los beneficie a todos y, por lo tanto, siempre será una victoria para la red.

Los objetivos particulares de estos algoritmos son:

- Llegar a un acuerdo
- Colaboración
- Cooperación
- Igualdad de derechos
- Participación
- Actividad

Para un sistema de red distribuida, no es muy común que todos los nodos estén disponibles cada vez que se quiera llegar a un acuerdo, al igual que hay bastantes posibilidades de que se pierda información durante la transmisión de ésta. Los algoritmos de consenso resuelven uno de los mayores problemas que un sistema distribuido puede tener: asegura que el acuerdo se lleva a cabo con los mínimos recursos necesarios, al igual que asegura la integridad y transparencia de las decisiones que se toman.

Los algoritmos de consenso son vitales en los sistemas de gran escala ya que permiten que un conjunto de servidores distribuidos funcione como un grupo coherente. A su vez, llegan a un acuerdo, incluso en presencia de fallos o interrupciones. Para lograr esto, el algoritmo establece un umbral o número de máquinas miembros de un sistema que deben llegar a un consenso/acuerdo.

Dado un sistema distribuido en el que cada proceso comienza con un valor inicial, resolver el problema de consenso supone proponer un algoritmo distribuido válido que permita a cada proceso generar un valor del mismo tipo que los valores de entrada, de tal manera que se cumplan estas tres condiciones: llegar a un acuerdo, generar validez y terminar el acuerdo.

Los sistemas distribuidos son a menudo sistemas parcialmente síncronos, sujetos a fallos en procesos, en el canal o en la temporización y recuperación. En un sistema distribuido parcialmente síncrono, los procesos toman acciones dentro del tiempo estipulado y los mensajes se entregan dentro de un tiempo determinado. Sin embargo, estos límites de tiempo se establecen para el normal funcionamiento de la red sujeto a la inexistencia de fallos o lo que sería lo mismo, en un entorno ideal. Es por este motivo que los límites de tiempo mencionados anteriormente pueden ser traspasados ocasionalmente cuando se plantea un fallo puntual. Los procesos pueden detenerse y recuperarse; es posible mantener el estado de un proceso, o parte de él, en un almacenamiento estable para que el estado sobreviva al fallo. También se puede plantear la posibilidad de que los mensajes se puedan perder, duplicar o reordenar [11] .

Cualquier algoritmo de consenso práctico necesita considerar todos estos escenarios. Es más, las propiedades básicas de seguridad no deben verse afectadas por la ocurrencia de algún fallo. También se debe tener en cuenta el rendimiento del propio algoritmo, que debe ser bueno cuando no hay fallos, pase a no ser demasiado eficiente cuando se encuentra con ellos, pero que siga funcionando.

A continuación, se presentan algunos ejemplos de los algoritmos de consenso más populares.

2.3 Algoritmo de Paxos

El problema de la consistencia en los sistemas distribuidos ha sido objeto de estudio durante muchos años. El algoritmo de Paxos, propuesto por Lamport [12], cumple con los requisitos básicos para considerarse una solución al problema de cómo alcanzar un acuerdo sobre una resolución en un sistema distribuido. Este algoritmo posee buenas propiedades de tolerancia a fallos y, cuando el sistema es estable, combina estas propiedades con el rendimiento eficiente de un algoritmo. Por lo tanto, puede resultar útil en la práctica para resolver dicho problema.

Ejemplos representativos de su aplicación incluyen Google Chubby y Hadoop ZooKeeper, que son mecanismos de sincronización basados en la nube utilizados en arquitecturas de distribución [13].

En el artículo original, el algoritmo se describe como el resultado de descubrimientos de estudios arqueológicos de una antigua civilización griega. Dicho documento contiene también una prueba de corrección y una discusión del análisis de rendimiento.

El modelo considerado es un sistema distribuido parcialmente síncrono [14] donde cada proceso tiene un canal de comunicación directo entre sí. Los fallos permitidos por este algoritmo son bien de temporización, relacionados con mensajes, ya sean de pérdida, duplicado o reordenación de estos o fallos debidos a la detención de distintos procesos. Se permiten recuperaciones de dichos procesos siempre y cuando se disponga de ciertos recursos de almacenamiento estable.

Este algoritmo garantiza un funcionamiento seguro, lo cual implica que todo el proceso de llegar a un acuerdo y su posterior validación se realiza de manera confiable, independientemente de los fallos que puedan surgir en el proceso.

Cuando el sistema distribuido se estabiliza, es decir, no hay fallos ni procesos de recuperación y la mayoría de los procesos no se detienen por un período prolongado, se alcanza la terminación del acuerdo y podemos afirmar que el desempeño del algoritmo es satisfactorio.

El estilo utilizado para la descripción del algoritmo a menudo desvía la atención del lector. Es por esto que, a pesar de ser un algoritmo práctico y elegante, su formulación no es obvia. Aun así, es ampliamente utilizado y se puede ver en los numerosos artículos que se encuentran donde se aplica este algoritmo.

Véase la importancia de su uso en máquinas de estado replicadas con este algoritmo para la base de datos de un almacén de alto rendimiento [15] hasta sistemas de replicación de máquinas de estado resistente a las particiones de red parciales, un importante foco disruptivo de los últimos años [16].

2.3.1 Fundamentos de Paxos

El algoritmo define, por lo general, tres roles [17] (véase la Fig. 2-1):

- **Proponentes:** Encargados de proponer valores para llegar a un acuerdo entre los nodos.
- **Aceptores:** Aquellos que contribuyen a llegar a un acuerdo estipulado.
- **Aprendices:** Encargados simplemente de aprender el valor fijado llevado a consenso una vez validado.

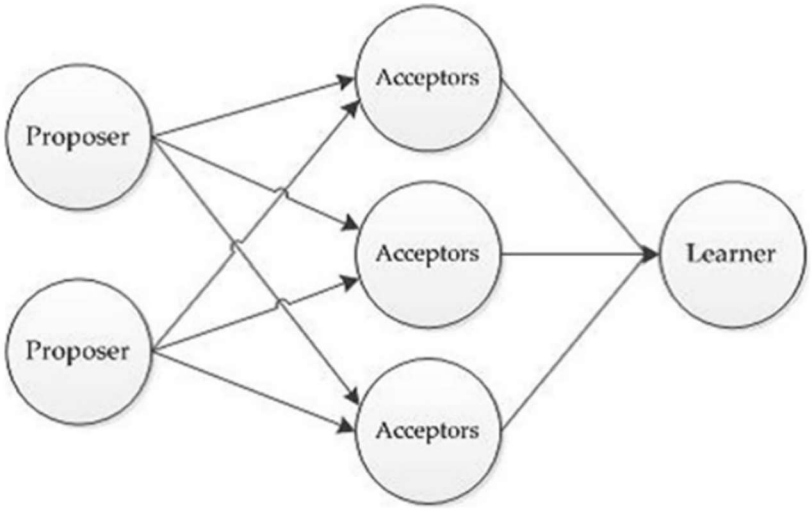


Figura 2-1 Esquema de roles que se consideran en algoritmo de Paxos [10]

En la práctica, los nodos pueden adoptar múltiples roles, incluidos todos ellos. Por lo general, nada impide que un nodo pueda enviar propuestas a otro nodo, luego pueda contribuir a la consecución de un acuerdo y a su vez aprender el valor final alcanzado por consenso.

Otra característica es que los nodos deben saber cuántos aceptores componen la mayoría. Esto es importante de destacar debido a la propiedad que establece que dos mayorías siempre se superpondrán en al menos un nodo. Este punto lo veremos con más detenimiento en los ejemplos propuestos más adelante.

Todos los nodos deben ser persistentes. En otras palabras, no pueden olvidarse de lo que han acatado previamente. Una iteración Paxos persigue la consecución de un consenso único. Esto quiere decir que solo se podrá llegar a un acuerdo por un único valor. Una vez que se ha alcanzado este consenso, no podrá progresar hacia otro consenso, ya que hará falta otra iteración.

2.3.2 Funcionamiento de Paxos

Para organizar este proceso, se evalúa un número distinto de propuestas en cada iteración. Las propiedades de seguridad del algoritmo no dependen de nada más que de que los números de la propuesta sean distintos, pero dado que los números más altos anulan los más bajos, para progresar necesitaremos que aumenten con el tiempo.

El mecanismo de votación se establece como sigue (véase la Fig. 2-2):

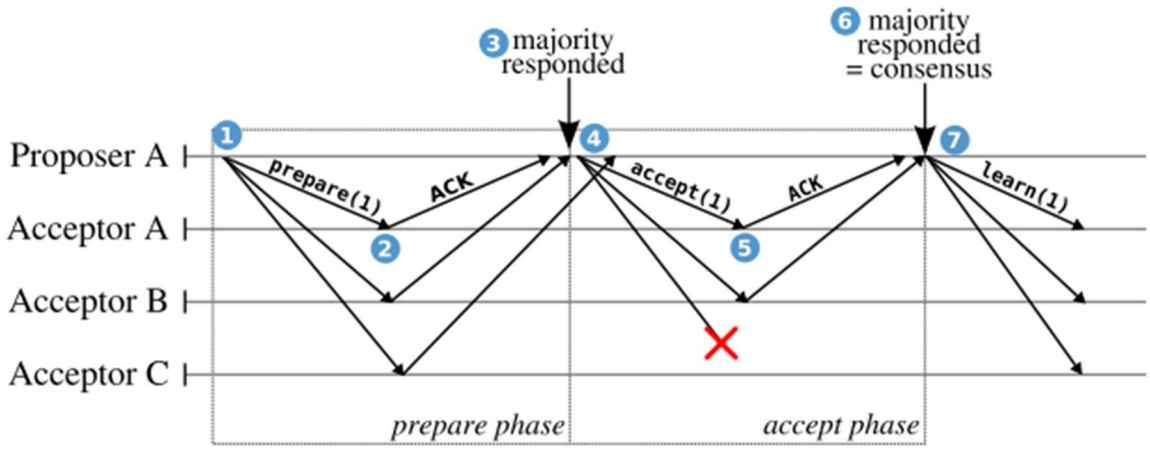


Figura 2-2 Esquema de funcionamiento del algoritmo de Paxos [18]

1. Antes de votar, el proponente manda un mensaje de preparación (n) a todos los aceptantes, donde n es el número de propuesta.
2. El aceptante compara n con el valor más alto de propuesta que tenga. Si n es superior, el aceptante responde a esto con la promesa de nunca aceptar ninguna propuesta con un número menor. Esto se hace para que las propuestas antiguas no sean ratificadas repentinamente.
3. El proponente espera a recibir la confirmación de la mayoría de los aceptantes. Si algún acuse de recibo contenía un valor, establece v en el valor más reciente (en el orden del número de propuesta) que recibió.
4. Luego, se envía un mensaje de aceptación - accept (n, v) - a todos los aceptantes, aunque solo es necesario que se lo envíe a la mayoría.
5. Al recibir el mensaje – accept (n, v) - un aceptante acepta v a menos que ya haya recibido - prepare (n') - para algún $n' > n$.
6. Si la mayoría de los aceptantes acepta el valor de una propuesta determinada, ese valor se convierte en el valor de decisión del protocolo.
7. Los aceptores envían mensaje a los aprendices para informar del valor que se ha llegado en consenso.

2.3.3 Variantes del algoritmo Paxos para sistemas distribuidos

Mientras que Paxos es un protocolo de consenso que acuerda un valor único, nos encontramos con una variante interesante como es el MultiPaxos. Es un protocolo de replicación de máquina de estado que acepta una secuencia o registro de valores y se utiliza para coordinar múltiples servidores en un sistema distribuido [19]. Este algoritmo es utilizado para asegurar que dichos servidores lleguen a un acuerdo en la selección de un valor o decisión común.

Los clientes envían comandos de máquina de estado a MultiPaxos. Éste coloca los comandos en un registro totalmente ordenado y las réplicas de máquinas de estado ejecutan los comandos en orden de registro, empezando por el estado inicial y ejecutando los mismos comandos en el mismo orden. Por este motivo, todas las réplicas de máquinas de estado se mantienen sincronizadas.

En Paxos, cada operación de consenso es vista como una instancia separada del protocolo, lo que significa que se necesita un proceso de elección de líder y una ronda de proposición, aceptación y confirmación para cada operación individual. MultiPaxos introduce una optimización para este proceso, permitiendo a los sistemas distribuidos llevar a cabo múltiples operaciones de consenso en una secuencia rápida de fases de proposición, aceptación y confirmación, sin necesidad de volver a realizar la elección de líder en cada ronda. Esto se traduce en que se consigue con MultiPaxos una mayor eficiencia y rendimiento que con Paxos, especialmente en sistemas distribuidos que necesitan procesar una gran cantidad de operaciones de consenso. Además, MultiPaxos también es más fácil de implementar y mantener que Paxos, lo que lo hace una opción popular en sistemas distribuidos modernos.

El algoritmo MultiPaxos es utilizado en muchos sistemas distribuidos para garantizar la coherencia y la disponibilidad de datos en el sistema. Por ejemplo, se puede utilizar en sistemas de bases de datos distribuidas para asegurar que las actualizaciones a los datos se propaguen a través del sistema de manera coherente y confiable como es el caso de Apache Cassandra [20].

Apache Cassandra es un sistema de bases de datos distribuidas altamente escalable y disponible. Utiliza MultiPaxos como parte de su protocolo interno de replicación y consenso para garantizar la consistencia y la coherencia de los datos en todos los nodos del clúster.

Supóngase un escenario donde tenemos un clúster de nodos distribuidos en diferentes ubicaciones geográficas. Cada nodo en el clúster almacena y replica una porción de los datos de manera distribuida para lograr la redundancia y la alta disponibilidad. Cuando se realiza una actualización de datos en uno de los nodos, por ejemplo, la inserción de un nuevo registro o la modificación de un registro existente, el nodo coordinador responsable de esa operación utiliza el algoritmo MultiPaxos para llegar a un consenso con los demás nodos del clúster.

El algoritmo MultiPaxos permite que los nodos acuerden el orden en el que se aplicarán las actualizaciones en todos los nodos, asegurando así la coherencia y la consistencia de los datos. Esto significa que todas las réplicas de los datos se actualizarán de manera sincronizada y reflejarán los mismos cambios.

Gracias a MultiPaxos, Apache Cassandra puede garantizar que las actualizaciones de datos se propaguen de manera confiable y coherente en todo el sistema distribuido, incluso en presencia de fallos de nodos individuales o interrupciones de red. Esto permite a los usuarios obtener una visión consistente y confiable de los datos en todo momento, lo que es esencial en aplicaciones que dependen de una base de datos distribuida para operar correctamente.

Por otro lado, en el contexto de redes inalámbricas de baja potencia, donde los recursos son limitados surge una solución para el excesivo intercambio de mensajes que ocurre en Paxos. Estos intercambios se traducen en la necesidad de tener un gran ancho de banda para poder lograr un consenso.

Por esta razón, ha surgido un protocolo *many-to-many* como alternativa para aplicaciones que requieren una coordinación compleja entre sus miembros en redes inalámbricas de bajo consumo. Este protocolo es especialmente útil para tales aplicaciones, permitiendo una comunicación eficiente y una coordinación efectiva.

Se presenta así Wireless Paxos [21], una aplicación del algoritmo Paxos a redes inalámbricas donde la conectividad puede ser intermitente. En este contexto, los nodos pueden estar temporalmente desconectados, lo que puede provocar la falta de coordinación y de consenso en el sistema. Wireless Paxos se encarga de resolver este problema permitiendo que los nodos acuerden un orden de mensajes incluso si algunos nodos se encuentran temporalmente desconectados o si la conectividad de la red es pobre. De esta forma, el protocolo garantiza que se llegue a un consenso en el orden de los mensajes, lo que es fundamental en muchos sistemas distribuidos.

En concreto, el diseño de Wireless Paxos se basa en tres principios:

- Comunicación orientada a la transmisión, ya que se aprovechan las propiedades de transmisión de la tecnología del medio inalámbrico para difundir solicitudes y recopilar respuestas eficientemente desde todos los nodos.

- Transmisiones simultáneas, se construyen sobre transmisiones concurrentes para proporcionar baja latencia y alta confiabilidad.

- Computación local y agregación, distribuyendo la lógica de decisión del proponente a todos los nodos a través de una función de agregación para convertir Paxos a un esquema de *many-to-many*.

2.3.4 Verificación de Paxos: explorando sus aplicaciones y estudios relevantes

La verificación de Paxos ha sido objeto de numerosos estudios que buscan analizar su funcionamiento y aplicaciones en diversos contextos. Entre las técnicas utilizadas, destaca la verificación de modelos, la cual permite explorar automáticamente el espacio de estados de los sistemas.

Lamport escribió especificaciones en TLA+ para Paxos y sus variantes, como Fast Paxos [22], y las verificó utilizando el verificador de modelos TLA+ llamado TLC [23].

Yabandeh et al. [24]verificaron una implementación en C++ de Paxos utilizando CrystalBall, una herramienta construida sobre Mace [25], que incluye un verificador de modelos.

También se llegó a modelar Paxos en Promela, en este caso Delzanno et al. [26] y lo verificaron utilizando el verificador de modelos Spin [27]. Se realizó para reducir el espacio de estados, utilizando guardias de conteo para rastrear la mayoría, reinician las variables locales después de las operaciones de estado y utilizar un envío ordenado en lugar de envío FIFO.

Estos estudios han permitido identificar errores y optimizaciones en la implementación de Paxos, abriendo el camino hacia su aplicación en sistemas distribuidos de manera más confiable y eficiente. Tanta es la importancia de estos estudios que surge así el siguiente algoritmo a destacar.

El algoritmo Raft no es más que una alternativa a Paxos debido a su complejidad inherente. El estudio de Ongaro y Ousterhout [28] se propone diseñar un algoritmo de consenso que fuera más fácil de entender y de implementar sin comprometer la confiabilidad y la escalabilidad. Este algoritmo está basado en el liderazgo de un único líder y la replicación de registros de estado y se ha convertido en una alternativa popular a Paxos en muchos sistemas distribuidos debido a su naturaleza más intuitiva y su mayor legibilidad. A continuación, exploraremos en detalle el algoritmo Raft y su funcionamiento.

2.4 Algoritmo Raft

Propuesto por Diego Ongaro y John Ousterhout [28], Raft es un algoritmo de consenso que está diseñado para ser fácil de entender. Es equivalente a Paxos en tolerancia a fallos y rendimiento. La principal diferencia reside en que se descompone en subproblemas relativamente independientes.

Este algoritmo de consenso busca lograr consistencia en un grupo de nodos los cuales comparten información. Trabaja seleccionando a un nodo como líder, el cual realiza solicitudes y coordina el resto de nodos para implementarlas. El clúster de nodos de Raft, trabaja mientras exista una mayoría (51%) de nodos en línea.

2.4.1 Fundamentos de Raft

Este algoritmo, a diferencia de Paxos, establece consenso eligiendo primero a un líder distinguido y luego delegando toda la responsabilidad de administrar el registro al líder. El líder recopila entradas de registro de los clientes, las duplica en otros servidores y notifica a los servidores cuando las entradas de registro se pueden aplicar de manera segura a sus máquinas de estado [29].

Los nodos participantes pueden estar en tres estados (véase la Fig. 2-3):

- **Líder**: Todos los cambios que se realicen en el clúster pasan por él primero.
- **Candidato**: Cuando un líder muere, cualquier seguidor puede presentarse a una elección.
- **Seguidor**: Un seguidor simplemente recibe órdenes de los líderes y si alguien externo intenta comunicarse con él, el seguidor redirige la solicitud al último líder que conoce.

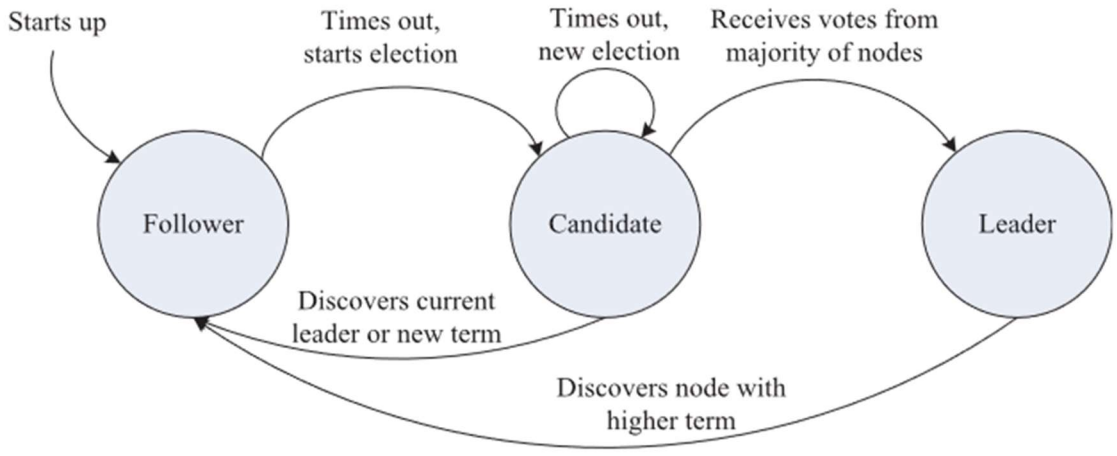


Figura 2-3 Modelo de máquina de estados para algoritmo Raft [30]

2.4 2 Funcionamiento de Raft

Un líder se elige a través de un procedimiento conocido como elección. Cada nodo tiene un tiempo de espera aleatorio después del cual, si un líder no se comunica con él, pasa a estado candidato y pide ser elegido como líder.

Este algoritmo realiza un seguimiento del paso del tiempo usando un contador llamado "término". Cuando se elige un nuevo líder, el recuento de términos aumenta y el término actual se envía con cada pulso. Cualquier seguidor, candidato o líder con plazos inferiores a los obtenidos en el pulso admite que se está ejecutando en un estado desactualizado e inmediatamente corrige su situación.

En el término actual, todos los seguidores pueden votar por un solo líder. Una vez que un seguidor ha votado en un término específico, solo puede votar por los candidatos que se presenten para los términos superiores. Para que un nodo sea elegido como líder es necesario que cuando pasa a estado candidato reciba la mayoría de votos de los seguidores en un término. En este punto se dice que hay quorum.

A continuación, se detallan los pasos que componen este algoritmo:

1. Como todavía no se ha designado ningún líder cuando comienza el clúster, no se podrán atender ninguna solicitud. En este punto, todos los temporizadores de los servidores comienzan a funcionar. El término inicial para todos los servidores se inicializa en 1, por lo que el término 1 no tiene líderes.

2. Cuando un seguidor llega al final de su término, se promociona a sí mismo como candidato para la elección del próximo mandato. Luego, el candidato comienza a solicitar votos y realiza un seguimiento del número de votos recibidos. Cuando los seguidores votan, extienden su mandato y reconocen que hay un nuevo líder.

3. Una vez que se elige a un líder, éste envía pulsos regularmente para demostrar que sigue ahí. Esto evita que los seguidores agoten el tiempo de espera y los relojes se reinician cada vez que un seguidor recibe un pulso de su líder.

4. Ahora que ya se tiene un líder, el clúster puede reanudar las operaciones normales y aceptar comandos del cliente. Cuando el líder recibe un comando de un cliente, primero agrega la entrada a su registro y luego envía las entradas a los seguidores.

Muchas veces, un seguidor o líder puede haber estado desconectado y no se ha podido mantener al día con las actualizaciones más recientes o simplemente ha ocurrido una partición de red. En general, Raft maneja estos escenarios de manera más efectiva que otros algoritmos de consenso distribuido, como Paxos, al reducir la cantidad de tiempo necesario para detectar y resolver particiones de red, y al garantizar que solo haya un líder activo en todo momento.

Un ejemplo de partición de red podría ser el siguiente:

Supóngase que se tiene un clúster de 5 nodos en una red y todos los nodos están en la misma red local comunicándose directamente entre sí. De repente, un problema en el enrutador de red divide la red en dos subredes, una con dos nodos y otra con tres.

En este escenario, se ha producido una partición de red, donde los nodos en cada subred no pueden comunicarse con los nodos en la otra subred. Cada subred tendrá su propio líder elegido por el algoritmo de elección de líder. Cuando se recupere la red original se llegaría a un conflicto de intereses ya que ambos líderes intentarán enviar comandos al mismo conjunto de seguidores, lo que comprometería la consistencia del sistema. ¿Cómo trata Raft este tipo de problema? [31].

Desde el punto de vista de los nodos que se quedan sin líder, si el líder del término actual falla, los relojes de los nodos seguirán funcionando hasta que se alcance un tiempo de espera de elección en cualquiera de los nodos.

Entonces, uno de los nodos se posicionará como candidato y comenzará a solicitar votos para convertirse en el nuevo líder, repitiendo así el algoritmo hasta que se elija un nuevo líder.

Una vez que la conexión se restablece, los nodos que se habían desconectado pueden unirse al clúster existente y sincronizarse con el líder actual, obteniendo un término superior. Si el antiguo líder envía un comando a cualquiera de los seguidores, éstos, al darse cuenta de que el término está desactualizado, lo alertarán de la presencia de un nuevo líder. En ese momento, el seguidor volverá instantáneamente al estado de espera de comandos del nuevo líder, quien continuará supervisando el clúster como de costumbre.

De esta manera, el algoritmo Raft asegura que solo haya un líder en cualquier momento y que el clúster pueda recuperarse de las particiones de red de manera segura y consistente.

2.4.3 Aplicaciones relevantes y estudios destacados del algoritmo Raft

La necesidad de garantizar la consistencia y la disponibilidad en sistemas distribuidos ha llevado al constante desarrollo y mejora de numerosos algoritmos de consenso. Raft es uno de los algoritmos más utilizados y estudiados. Tanto es así, que se ha convertido en una opción bastante popular en aplicaciones y sistemas distribuidos debido a su naturaleza intuitiva y su capacidad para garantizar la tolerancia a fallos [28].

Un ejemplo notable de su uso se encuentra en la base de datos MongoDB, debido a su utilización de sistemas basados en elección de líder y réplica de registros, que garantizan un registro consistente en todos los nodos y la replicación de los mismos datos en todas las instancias. Además, el envío de pulsos regulares permite mantener una comunicación constante y asegurarse de que todos los nodos estén activos.

MongoDB es una base de datos NoSQL flexible, escalable y de alto rendimiento que funciona como una base de datos no relacional que almacena datos en colecciones de objetos JSON llamados documentos. Estos documentos son muy flexibles y permiten que los datos no estén estructurados de una manera estricta como en las bases de datos relacionales [32].

Esta base de datos es especialmente adecuada para aplicaciones web, móviles y de IoT, y se utiliza en una variedad de industrias, desde finanzas y comercio electrónico hasta salud y medios de comunicación. Debido a su arquitectura distribuida y replicación de datos, MongoDB puede distribuir y replicar datos en múltiples servidores, lo que aumenta su disponibilidad, escalabilidad y tolerancia a fallos.

También ofrece una alta disponibilidad gracias a su capacidad para realizar consultas complejas y su soporte para consultas *ad hoc*, indexación completa, replicación automática y fragmentación horizontal.

En términos de consenso, MongoDB utiliza un protocolo de consenso derivado directamente de Raft, lo que permite que MongoDB proporcione capacidad de linealización y tolerancia a cualquier minoría de fallos. Este protocolo de consenso es una de las razones por las que MongoDB es una opción popular para proyectos de desarrollo de software de todo tipo.

El consenso distribuido es inherentemente complejo, y el algoritmo Raft busca abordar esta complejidad al ser un algoritmo de consenso comprensible. Durante los últimos años, se ha construido una comunidad muy activa alrededor de Raft con más de 20 implementaciones diferentes en lenguajes que van desde C++, Erlang y Java hasta Clojure, Ruby o Python [33].

Cabe destacar que una gran parte de la comunidad se ha centrado en buscar mejoras para enfocarlo a la tecnología Blockchain desde que Forbes anunciara en 2020 el creciente número de empresas que están adoptando esta tecnología en diversos campos, como software, energía, servicios financieros y atención médica [34], que están liderando la innovación en los procesos de negocio.

Véase como ejemplo el artículo de Fu et al. [35] para abordar el problema de la alta demanda de rendimiento y baja latencia característicos de las aplicaciones de Blockchain enfocado a partir del algoritmo Raft.

Este tipo de conexión entre el algoritmo Raft y la tecnología Blockchain resalta la relevancia del consenso distribuido en la seguridad y descentralización de este tipo de redes. A continuación, nos adentraremos en el contexto de esta tecnología y exploraremos sus implicaciones en los aspectos más fundamentales como la seguridad y la descentralización.

2.5 Algoritmos basados en Blockchain

En los últimos años, la tecnología Blockchain ha mostrado un gran potencial en los campos de la logística, la salud, las finanzas y el IoT [36]. Pero sobre todo ha tenido una gran popularidad en el contexto de las nuevas criptomonedas [37] donde las transacciones se registran después de ser verificadas. Estas transacciones son verificadas por muchos clientes o "validadores". Aunque hay diferentes algoritmos de consenso alternativos disponibles, los implementados más comúnmente son el de prueba de trabajo (PoW), el de prueba de participación (PoS) y el de prueba de autoridad (PoA) [38].

El consenso para Blockchain es un procedimiento en el que los pares de una red descentralizada llegan a un acuerdo sobre el estado actual de los datos. A través de esto, los algoritmos de consenso establecen seguridad y confianza en dicha red. Estos algoritmos funcionan como la columna vertebral de todas las cadenas de bloques de criptomonedas y son lo que las hacen seguras. Antes de profundizar en los diferentes mecanismos de consenso, primero se debe definir qué significa que las cadenas de bloques logren el consenso.

Una cadena de bloques o Blockchain es un libro de contabilidad digital descentralizado, distribuido y, a menudo, público, que se utiliza para registrar transacciones [39]. Cada una de estas transacciones se registra como un "bloque" de datos, que debe ser verificado de forma independiente por redes informáticas *Peer-to-Peer* (P2P) antes de que puedan agregarse a la cadena. Este sistema ayuda a proteger la cadena de bloques contra actividades fraudulentas.

Para garantizar que todos los participantes (nodos) en una red de cadena de bloques acuerden una única versión del historial, las redes de cadena de bloques como Bitcoin o Ethereum implementan algoritmos de consenso. Como se ha visto anteriormente, estos mecanismos tienen como objetivo hacer que el sistema sea tolerante a fallos.

Una de las principales características del Blockchain es la forma en que se registran los datos. En primer lugar, es inmutable, lo que significa que ningún usuario puede modificar los registros de transacciones. A su vez, es transparente, lo que implica que todos pueden ver y verificar las transacciones en una cadena de bloques a través de Internet. Por último, es descentralizado, por lo que ninguna entidad individual puede gobernar toda la red.

2.5.1 Algoritmo Blockchain I: Proof of Work (PoW)

En general, se considera que es el más confiable y seguro de todos los mecanismos de consenso, aunque abundan las preocupaciones sobre la escalabilidad [40]. El término "prueba de trabajo" se acuñó por primera vez a principios de la década de 1990. Fue el fundador de Bitcoin, Satoshi Nakamoto [41], quien aplicó por primera vez la tecnología en el contexto de las criptomonedas.

En PoW, los mineros compiten esencialmente entre sí para resolver acertijos computacionales extremadamente complejos utilizando ordenadores de alta potencia. El primero en encontrar el número hexadecimal de 64 dígitos ("hash") gana el derecho a formar el nuevo bloque y confirmar las transacciones. El minero exitoso también es recompensado con una cantidad predeterminada de criptomonedas, conocida como "recompensa en bloque".

Como requiere grandes cantidades de recursos computacionales y energía para generar nuevos bloques, los costos operativos detrás de PoW son notablemente altos. Esto actúa como una barrera de entrada para los nuevos mineros, lo que genera preocupaciones sobre las limitaciones de centralización y escalabilidad.

Y no son solo los costos los que son altos. La crítica más común de PoW es el impacto que tiene el consumo energético en el medio ambiente. Esto ha llevado a muchos a buscar protocolos de consenso más sostenibles y energéticamente eficientes, como la prueba de participación.

2.5.2 Algoritmo Blockchain II: Proof of Stake (PoS)

La prueba de participación es otro algoritmo de consenso que posee el mismo objetivo que la prueba de trabajo, aunque el proceso para validar transacciones en la red distribuida no es el mismo. Fue diseñado originalmente para resolver el problema del consumo energético [42].

Este sistema busca incentivar a los participantes para que posean, en todo momento, una determinada cantidad de monedas a base de bloquear una cantidad durante un tiempo determinado. La posesión de monedas, les permite ser elegidos por el proceso de selección aleatoria que se realiza para designar tareas. Bajo este esquema, mientras se tengan más reservas, se tiene mayor peso en la red y mayores oportunidades de ser elegidos. Una vez elegidos pueden validar transacciones y crear nuevos bloques dentro de la red, permitiendo obtener ganancias e incentivos por el trabajo realizado [40].

Las ventajas de este mecanismo son principalmente el ahorro de una gran cantidad de potencia de cómputo y energía, ya que los nodos no consumen potencia de cómputo adicional para la minería, y el ahorro de tiempo en la generación de bloques para alcanzar el consenso, mejorando así la eficiencia del consenso.

Por otro lado, cuenta con algunas desventajas como la complejidad a la hora de la implementación o que en este caso la minería tiene un costo bajo y es fácil que sea atacada, lo que se traduce en un problema de seguridad.

2.5.3 Algoritmo Blockchain III: Proof of Authority (PoA)

Llegar a un acuerdo en un entorno asíncrono es esencial para garantizar la consistencia en un sistema distribuido. Todos los protocolos asíncronos presentados atrás eran probabilistas o asumían un modo de fallo distinto al que se presenta a continuación.

El problema de los generales bizantinos [43] se usa para plantear la situación que se presenta entre un conjunto de sistemas informáticos que tienen un objetivo en común. Deben encontrar un plan de acción a partir de una estructura jerárquica, donde uno de los sistemas que tiene mayor rango proporciona una orden a partir de la cual el resto de sistemas tiene que fijar una decisión. Además, es posible que alguno de ellos no sea fiable y provea información falsa de forma intencionada. Se trata, además, de un problema clásico de las redes distribuidas.

La tolerancia a fallos bizantinos [44] es una propiedad de los sistemas que son capaces de resistir los tipos de fallos derivados del problema de los generales bizantinos de continuar operando incluso cuando algunos nodos no son capaces de comunicarse o cuando algunos actúan de forma maliciosa.

El algoritmo IBFT (Istanbul Byzantine Fault Tolerance) [45] es un protocolo de consenso de cadena de bloques que garantiza la finalidad inmediata y es utilizado en la red Ethereum. Mediante el análisis de Saltini [46] se muestra que el protocolo no garantiza la persistencia tolerante a fallos bizantinos cuando se opera en una red eventualmente síncrona. Aun así, es importante de destacar debido a su importancia por plantear un problema diferente de los vistos anteriormente.

El procedimiento del IBFT se basa en la elección de un proponente de bloques de forma rotatoria, que realiza una nueva propuesta de bloque y la transmite junto con el mensaje “pre-preparado”. Luego, los validadores reciben el mensaje del proponente y entran al estado “pre-preparado” para asegurarse de que todos los validadores estén trabajando en la misma secuencia y ronda. A medida que los validadores reciben $2F + 1$ mensajes, entran en el estado “preparado” y transmiten el mensaje de compromiso para informar a sus pares que aceptan el bloque propuesto y que van a insertarlo en la cadena. Finalmente, los validadores esperan $2F + 1$ mensajes de “comprometido” para entrar en el estado “comprometido” y luego insertar el bloque en la cadena (véase la Fig. 2-4).

Este algoritmo es capaz de tolerar como máximo F nodos defectuosos en una red de N nodos de validación, donde $N = 3F + 1$. Además, los bloques producidos por IBFT están fuertemente protegidos contra la manipulación a través de la recopilación de firmas del proponente y los validadores de votos, lo que proporciona garantías fuertes de la inmutabilidad de la cadena de bloques resultante.

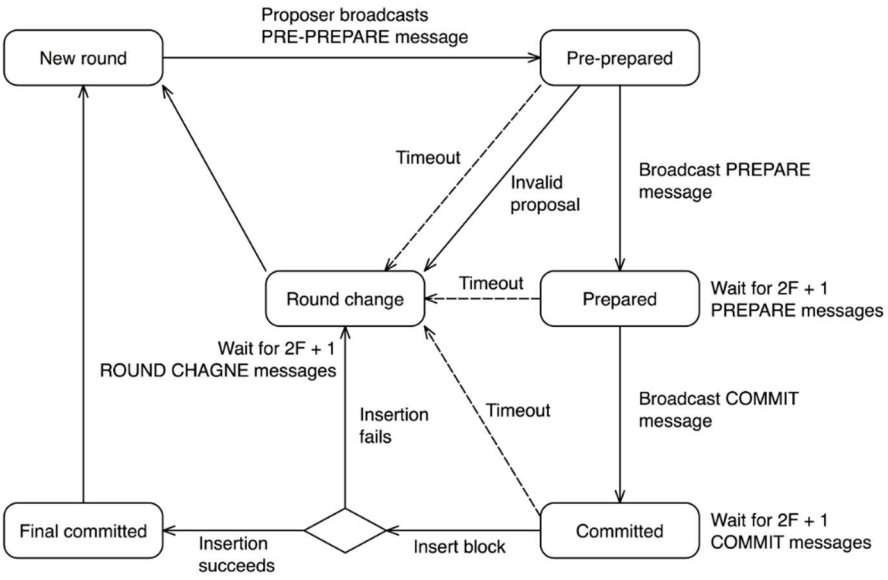


Figura 2-4 Modelo de máquina de estados para algoritmo de IBFT

2.5.4 Tendencias recientes de los mecanismos de consenso en Blockchain

En los últimos años, las aplicaciones en diversos campos que emplean mecanismos de consenso basados en Blockchain se han convertido en un tema de investigación destacado.

En términos financieros, Matzutt et al. [37] propusieron CoinPrune para abordar los problemas de escalabilidad de las criptomonedas sin cambiar las reglas de consenso de Bitcoin haciendo que los usuarios con mayor crédito, es decir, bancos y reguladores, sean considerados los nodos de consenso para la confirmación de transacciones.

Si se habla de aplicaciones de IoT, se puede encontrar Groupchain [47], donde se establece un grupo líder para alcanzar el consenso de manera efectiva. Los resultados experimentales demuestran que efectivamente se logra una optimización en el rendimiento de las transacciones y la latencia de confirmación que se discuten en Bitcoin.

También cabe destacar la aplicación a la Industria 4.0 donde los volúmenes de datos constantemente crecientes presentan problemas de seguridad, como la integridad de los datos y la escalabilidad del sistema. La tecnología Blockchain y en particular el algoritmo PoW se presenta como una solución prometedora para abordar estos problemas, ya que ofrece principios de diseño de sistemas distribuidos [48] que pueden adaptarse según la tasa de tráfico de comunicación entrante de dispositivos industriales de IoT.

En el ámbito del comercio de energía, Yang et al. [49] utiliza el concepto de "prosumidor" que define a los propietarios de recursos de energía distribuida que pueden producir y consumir energía. En este estudio se desarrolla el mecanismo de consenso de PoS para fomentar el comercio de energía de igual a igual entre los prosumidores, donde se utiliza la tecnología Blockchain en la red P2P gracias a su transparencia, seguridad y rapidez en la ejecución de transacciones.

Por poner un último ejemplo, en términos de redes vehiculares, Wang et al. [50] presentaron un mecanismo de consenso basado en tolerancia a fallos bizantinos y PoS (BFT-based PoS) para lograr un consenso eficiente y solucionar los posibles problemas de seguridad causados por un entorno no confiable y opaco. Para resolver los desafíos mencionados anteriormente, se propone una manera para compartir recursos de forma segura basado otra vez en Blockchain.

2.6 Algoritmo Chandra-Toueg

El diseño y verificación de aplicaciones distribuidas tolerantes a fallos es comúnmente visto como un reto bastante complejo. En los últimos años se han desarrollado distintos paradigmas que simplifican esta tarea.

En 1985, Fischer, Lynch y Paterson [51] demostraron que el problema de consenso no tenía solución en un sistema asíncrono sujeto a fallos de bloqueo en un solo proceso. Sin embargo, en 1991, Chandra y Toueg [52] demostraron que, al aumentar el modelo del sistema asíncrono con un detector de fallos, se puede llegar a resolver el problema de consenso.

Se propone un algoritmo para resolver este problema en un modelo asíncrono donde los procesos están totalmente interconectados a través de canales de comunicación punto a punto bidireccionales casi confiables, en los cuales la entrega de mensaje solo está garantizada si y solo si se consigue que ni el remitente ni el destinatario fallen.

Es por este motivo, que se siguen estas premisas:

- La hipótesis de que solo una minoría de procesos puede fallar.
- La presencia de un detector de fallos para proveer información en caso de que ocurra algún fallo.

Precisamente, este algoritmo sirve para resolver el consenso en una red de procesos poco confiables equipados con un detector de fallos eventualmente potente.

2.6.1 Detectores de fallos

Un detector de fallos es como un módulo o dispositivo abstracto establecido en cada proceso para que pueda proporcionar información relacionada con el estado operativo de los otros procesos del sistema.

Un estudio detallado de detectores de fallos realizado por Nie et al. [53] lleva a cabo una investigación minuciosa sobre más de 90 artículos clave en este aspecto, clasificándolos en ocho categorías según el procedimiento de prueba al que pertenecen. Para cada categoría, se detiene en encontrar su motivación, los temas clave, las soluciones y el estado actual de la investigación, revisando las contribuciones y presentando las tendencias en crecimiento en ese momento.

Para tener una imagen más visual se establece una estructura en capas del proceso (véase la Fig. 2-5).

- La capa de transporte proporciona a los procesos la capacidad de enviar y recibir mensajes a través del canal de comunicación.

- La capa del detector de fallos monitorea otros procesos y ofrece información posiblemente poco confiable sobre procesos defectuosos a la parte superior capa.

- La capa de consenso usa información de su módulo detector de fallos para lograr el acuerdo entre los procesos correctos.

- Finalmente, la capa de aplicación representa el conjunto de aplicaciones potenciales que podrían beneficiarse del servicio de consenso.

Los detectores de fallos se caracterizan por dos propiedades: integridad y precisión. La integridad se refiere al tiempo después del cual todos los procesos que fallan son permanentemente sospechosos por algún proceso correcto, mientras que la precisión se refiere al tiempo después del cual nunca se sospecha de algún proceso correcto por cualquier proceso correcto [54].

Aunque los detectores de fallos pueden cometer un número infinito de errores, en la práctica esto no siempre ocurre y estas dos propiedades no siempre se cumplen. En un problema de consenso es suficiente que el proceso se mantenga por un período de tiempo suficientemente largo para que el algoritmo logre su objetivo, es decir, para que los procesos correctos decidan.

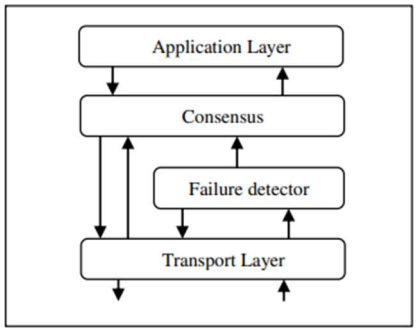


Figura 2-5 Capas en un proceso cuando se usa un detector de fallos para resolver el consenso [55]

Sin embargo, en un sistema asíncrono no es posible cuantificar cuánto tiempo es "suficientemente largo", ya que incluso un solo paso en el proceso puede tomar una cantidad de tiempo arbitrariamente larga [56]. Por lo tanto, la mayoría de las implementaciones de detectores de fallos se basan en mecanismos independientes del tiempo.

Estos detectores son capaces de distinguir entre procesos lentos y procesos muertos. Cada proceso tiene asociado un módulo de detector de fallas que emite continuamente una estimación de los procesos en el sistema que han fallado. La respuesta de salida no tiene que ser necesariamente correcta, de hecho, la principal contribución del artículo es caracterizar cuán falsa puede llegar a ser la salida de un detector de fallas, lo que puede ser de gran utilidad.

2.6.2 Funcionamiento del algoritmo Chandra-Toueg

Este algoritmo utiliza el paradigma del coordinador rotatorio y procede en "rondas" asíncronas. Cada proceso del algoritmo Chandra-Toueg realiza un seguimiento de su valor de decisión preferido al actual, que al principio sería su propia entrada, y una marca de tiempo que sería el número de rondas en las que actualizó por última vez su preferencia. Los procesos pasan por una secuencia de rondas asíncronas, cada una dividida en cuatro fases [57]:

1. Todos los procesos envían - ronda, preferencia, marca de tiempo - al coordinador de la ronda.

2. El coordinador espera a escuchar de la mayoría de los procesos. Luego, el coordinador establece como su preferencia un valor con la marca de tiempo más reciente entre todos los enviados y envía - ronda, preferencia - a todos los procesos.

3. Cada proceso espera la nueva propuesta del coordinador o que el propio detector de fallos sospeche del coordinador. Si recibe una nueva preferencia, la adopta como propia, establece una marca de tiempo para la ronda actual y envía - ronda, acuse de recibo - al coordinador. De lo contrario, envía - ronda, NACK - al coordinador.

4. El coordinador espera recibir ACK o NACK de la mayoría de procesos. Si recibe el apoyo de la mayoría, anuncia la preferencia actual como el valor de decisión del protocolo usando una transmisión fiable.

2.7 Algoritmo CoNICE

A medida que el mundo se vuelve cada vez más dependiente de la conectividad de una red, también es importante ser resilientes ante situaciones en las que la conectividad es intermitente [58]. Centrándonos en el caso de la conectividad a un punto de acceso (AP), si el AP está dañado, se puede decir que no está disponible, y se tiene que ser capaz de diseñar protocolos que permitan la difusión de información tolerando esta intermitencia en la conectividad.

Los algoritmos de consenso descritos anteriormente, como Paxos o Raft, son los más adecuados para entornos conectados con enlaces fiables, pero no siempre ocurre así. Surge entonces el problema de que la información deba ser difundida de forma intermitente en ciertos instantes, ya que la propia infraestructura de la red puede dejar de estar disponible en algunos momentos. Esto se puede lograr a través del ordenamiento causal y del consenso.

En este contexto se han realizado distintas aportaciones al diseño de algoritmos de consenso para entornos conectados intermitentemente, como el algoritmo OTR [59], una solución elegante para solventar el problema de consenso en redes donde la pérdida de mensajes puede ocurrir, aunque es necesario mejorar su eficiencia y finalización oportuna.

Para abordar este tipo de escenarios, las redes oportunistas, también conocidas como *ad hoc* toleran un grafo de topología desconectada con nodos móviles. Aprovechan los intercambios de mensajes de dispositivo a dispositivo en encuentros móviles entre nodos, al igual que en las redes tolerantes a retrasos (DTN) [60] sin depender de la infraestructura de red o de la disponibilidad de una ruta de extremo a extremo. Es ahí donde surge el algoritmo CoNICE [61].

2.7.1 Funcionamiento del algoritmo CoNICE

CoNICE proporciona un procedimiento de consenso con el objetivo de llegar a un acuerdo, de modo que los usuarios respeten el orden causal y que sea el mismo en todos los usuarios. La solución de consenso se basa en el algoritmo OTR extendiéndolo de varias maneras, principalmente con respecto a las invalidaciones de nombres y decisiones. La integración de nombre en CoNICE se asegura de que todos los usuarios interesados estén involucrados en cada sesión de consenso relevante para ellos. Esto reduce sistemáticamente los participantes del consenso a solo los usuarios interesados, ayudando a alcanzar más rápidamente las decisiones.

El esquema de consistencia multinivel de CoNICE se integra con un esquema de nomenclatura que sigue un marco de difusión basado en la jerarquía de temas, para mejorar la funcionalidad y el rendimiento. Surgen así los perfiles de intereses basados en nombres (NBIP).

Los NBIP capturan las suscripciones de un usuario localmente en sus dispositivos. Cada uno apunta a uno o más nodos (nombres) en un contenedor abstracto en el que un grupo de uno o más identificadores únicos pueden existir. A su vez, se incluyen los subespacios debajo en la jerarquía, enrutados a los nombres apuntados.

Cada sesión de consenso contiene las siguientes etapas:

1. Validez: Cualquier valor decidido ha sido el valor inicial de algún usuario.
2. Validez de actualización: Cualquier valor decidido es una actualización válida que fue creada.
3. Acuerdo: No hay dos usuarios que decidan diferente.
4. Terminación: Cada usuario correcto decide. Es decir, que no falla permanentemente ni se vuelve inalcanzable.
5. Integridad: Ningún usuario decide dos veces.
6.

Se consigue así la difusión constante de actualizaciones entre los usuarios en entornos de sistemas conectados de forma intermitente. CoNICE aprovecha la nomenclatura y la consistencia multinivel para el ordenamiento y consenso causal más selectivo y eficiente.

2.8 Algoritmo basado en WiFi Direct: RedMesh

En la actualidad, la interconexión de dispositivos electrónicos es esencial en nuestra vida cotidiana. La tecnología WiFi Direct surge como una alternativa a la infraestructura de red tradicional y se presenta como una forma de establecer conexiones inalámbricas de alta velocidad entre dispositivos móviles sin la necesidad de un punto de acceso intermedio. Esta tecnología permite la creación de una red inalámbrica *ad hoc* entre dos o más dispositivos, permitiendo la comunicación directa entre ellos, incluso sin la necesidad de una conexión a Internet.

En redes de comunicaciones, una red *ad hoc* se define como aquella en la que no se cuenta con un nodo central, sino que todos los dispositivos conectados están en igualdad de condiciones y se pueden comunicar unos con los otros [62]. Esto lo convierte en una solución costo-efectiva para la necesidad de armar una red independiente y fiable. Bluetooth, WiFi Direct y Zigbee son las tecnologías más conocidas usadas en redes *ad hoc* de conectividad inalámbrica, aunque WiFi Direct se presenta superior a las demás tecnologías en términos de área de cobertura, caudal y consumo de energía [63].

Una de las principales desventajas del estándar WiFi Direct es su limitación a la hora de afrontar comunicaciones intragrupo. Recientemente, han surgido en la literatura diferentes soluciones para permitir la comunicación entre dispositivos de diferentes grupos, generalmente utilizando un enfoque *multi-hop routing* [64]. Hasta donde se sabe, solo Baresi [65] y Casetti [66] propusieron algoritmos para crear redes de dispositivos WiFi Direct con el uso simultáneo de interfaces WiFi y WiFi Direct.

Sin embargo, estas propuestas no son eficientes ni efectivas para escenarios de gran escala, debido al uso de transmisiones, ya que ofrecen una conectividad limitada.

2.8.1 Tecnología WiFi Direct

Aunque en el siguiente capítulo se realizará una revisión más extensa de esta tecnología, aprovechamos esta sección para introducir brevemente algunos conceptos de WiFi Direct necesarios para entender cómo funciona el algoritmo RedMesh. WiFi Direct se basa en la especificación de los estándares 802.11 [67], lo que significa que es compatible con los dispositivos que ya poseen esta tecnología. Los equipos que utilizan WiFi Direct pueden conectarse a otros dispositivos con la misma capacidad, sin la necesidad de ningún otro módulo o *router* adicional, lo que lo hace especialmente útil en situaciones en las que no hay acceso a una red WiFi o cuando se desea establecer una conexión directa entre dispositivos móviles.

La tecnología WiFi Direct ofrece una variedad de aplicaciones en diferentes áreas, como la transmisión de archivos multimedia, juegos multijugador, aplicaciones de mensajería y aplicaciones empresariales, entre otros. Además, también se ha utilizado en aplicaciones de IoT para permitir la conexión entre dispositivos y sensores.

WiFi Direct soporta las conexiones directas entre usuarios sin necesidad de un punto de acceso físico central [68]. Dentro de un grupo de WiFi Direct se definen varios roles. En primer lugar, se tienen múltiples miembros (Group Member – GM´s) y un único dueño (Group Owner – GO) que actúa como punto de acceso para ofrecer un servicio básico (BSS). Los GO son responsables, por tanto, de la participación, avisos y comunicaciones entre los distintos GM´s.

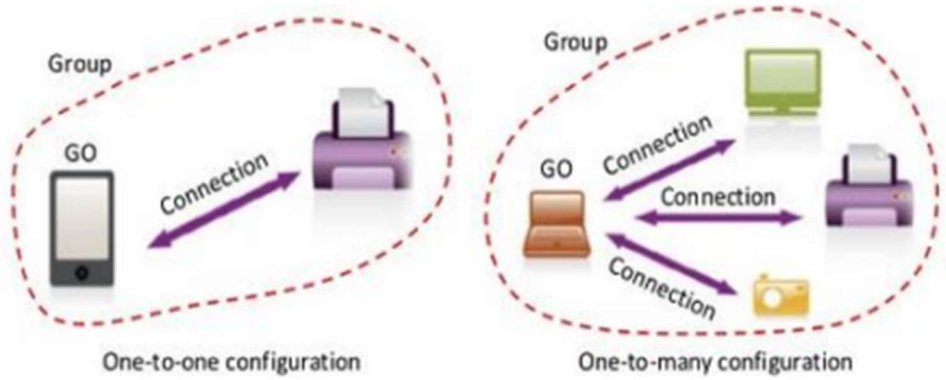


Figura 2-6 Comunicaciones dentro de un grupo [67]

2.8.2 Fundamentos del algoritmo RedMesh

RedMesh [69], es el primer algoritmo que construye redes malladas que interconectan dispositivos móviles listos para usar que están habilitados para WiFi Direct en escenarios a gran escala. Una vez establecidas, estas conexiones pueden usar solo comunicación unicast.

Este algoritmo ha demostrado ser muy efectivo, logrando conectividad total en el 97,28% de los 1.250 escenarios probados con hasta 250 nodos, en un total de 187.500 nodos [69].

WiFi Direct organiza los dispositivos en grupos, en los que un dispositivo, el GO, desempeña el papel de coordinador del grupo, proporcionando detección de nodos, formación de grupos, autenticación, DHCP y servicios de reenvío de datos para el resto de los miembros del grupo.

2.8.3 Funcionamiento del algoritmo RedMesh

Con el objetivo de maximizar la conectividad de la red, el algoritmo utiliza la siguiente secuencia de etapas:

1. Elección de nodo dominante (DNE)

El algoritmo comienza con la etapa DNE en la que cada nodo u comienza por descubrir sus vecinos $N(u)$, y los vecinos de esos vecinos $N(N(u))$, que llamaremos vecindarios de segundo nivel. Para ello, cada nodo inicializa su interfaz WiFi Direct y se anuncia a sí mismo emitiendo su identificador a través del Servicio de Publicidad Bonjour (BAS). Una vez que los nodos descubren a sus vecinos directos, transmiten esa información a esos vecinos, lo que permite el cálculo de la vecindad de segundo nivel. De su lista de vecinos directos, cada nodo establece si es un *Dominant Group Owner* (GOD), es decir, si tiene la identificación más alta en su vecindario. Los GODs entonces se convierten en GOs.

En la Fig. 2-7 se muestran 9 GODs (cuadrados amarillos), que son los nodos que tienen el ID más alto en su vecindario.

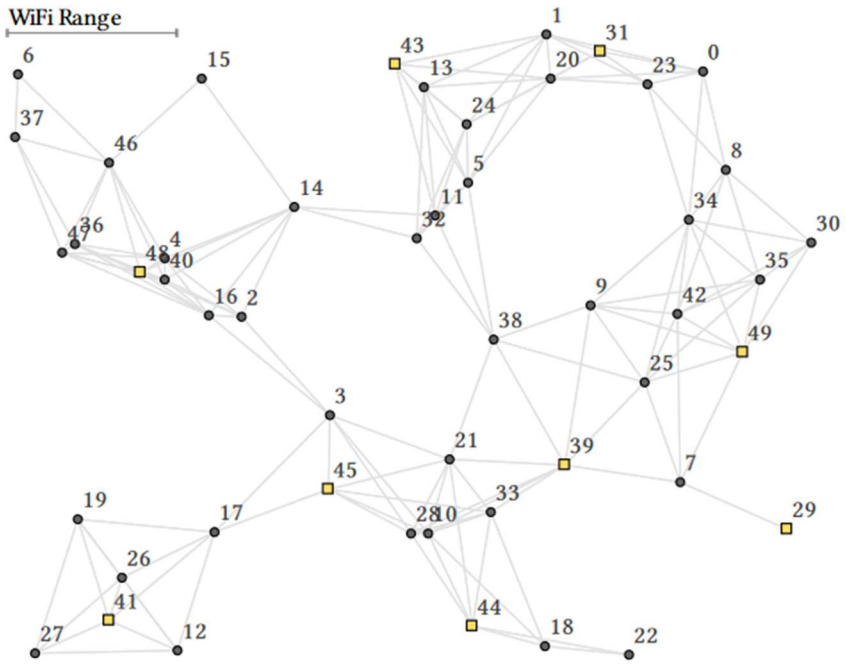


Figura 2-7 Resultado de simulación de este proceso para un escenario de 50 nodos [69]

2. Construcción de clúster (CLB)

Los GODs activan la etapa CLB al tratar de esclavizar a sus vecinos. Una vez esclavizado, un nodo se convierte en un GO e intenta esclavizar a sus propios vecinos inferiores (aquellos con ID más bajos), lo que hace que el proceso de esclavización fluya de los nodos de ID más altos a los más bajos. Cada GO (incluidos los GOD) garantizará que todos los vecinos inferiores estén esclavizados por algún nodo. Con esto se asegura la conectividad, es decir, todos los vecinos inferiores estarán conectados a un nodo. Por lo tanto, al final de esta etapa, todos los nodos se agrupan en clústeres dominados por un GOD (véase la Fig. 2-8).

A su vez se intenta esclavizar hasta L[1] (número máximo de esclavos permitidos por GO) esclavos, de manera que se maximice el número de esclavos y se reduzca el número de GO.

Al tener menos GO, se requieren menos transmisiones de radio, lo que contribuye a un enrutamiento más rápido (ya que hay menos grupos para cruzar), se reducen las interferencias de radio y los costos de mantenimiento de la red. En consecuencia, se forman mejores redes y más eficientes energéticamente.

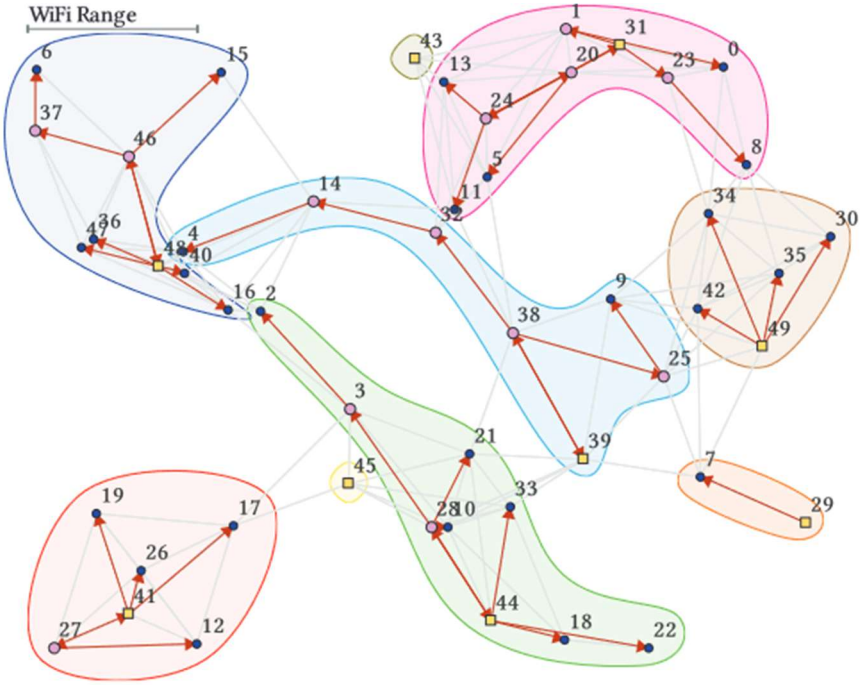


Figura 2-8 Formación de clústeres [69]

3. Reunión de vecindario de clúster (CNG)

En esta etapa se preparan los clústeres para la interconexión. Desde la perspectiva de un clúster, sus nodos son locales, mientras que todos los demás se consideran nodos remotos. Los nodos locales con vecindarios que incluyan nodos remotos se pueden usar para la interconexión de clústeres y se denominan nodos candidatos de puerta de enlace.

El objetivo de la etapa CNG es triple. Por un lado, enviar información de vecindad a nivel local y remoto (desde los GO, hasta los GOD), e identificar los nodos candidatos de puerta de enlace. Por otra parte, se promueve a algunos de estos candidatos a GO y, por último, enviar información de vecindario a nivel de clúster a clústeres vecinos de mayor rango.

[1] L no puede exceder el número máximo de esclavos del dispositivo, normalmente 8, ni puede ser inferior al número mínimo de vecinos que pueden cubrir la vecindad de un nodo, que es 5 [70], de lo contrario no se garantizará la conectividad. Entonces, $L \in [5, 8]$.

4. Interconexión de clúster principal (MCI)

Constituye la primera etapa de interconexión de clústeres, y es donde la mayoría de estas interconexiones ocurren. La etapa es desencadenada por los GODs más altos entre los grupos vecinos y desciende a los grupos más bajos. Los GODs comienzan esperando las notificaciones de finalización de MCI de todos sus clústeres vecinos superiores, luego intentan conectarse a los clústeres vecinos inferiores y concluyen enviando notificaciones de finalización MCI a todos sus clústeres vecinos inferiores. Cada notificación de finalización de MCI contiene los clústeres que se conectaron correctamente de forma directa, los que se conectaron indirectamente a través de clústeres intermedios y también los que se cree que no están conectados (véase la Fig. 2-9).

Los clústeres no conectados aún pueden estar conectados a través de alguna otra ruta de red que los clústeres en cuestión no conocen. Los GODs actualizan su tabla de visibilidad cuando reciben notificaciones de finalización de MCI y también cuando se interconectan con otros clústeres. Almacenan las identificaciones de los GODs de todos los clústeres vecinos de rango inferior en una lista llamada "cltsToConnect", ordenados en orden descendente, y luego intentan conectarse a cada clúster en la lista utilizando el procedimiento de interconexión de clústeres.

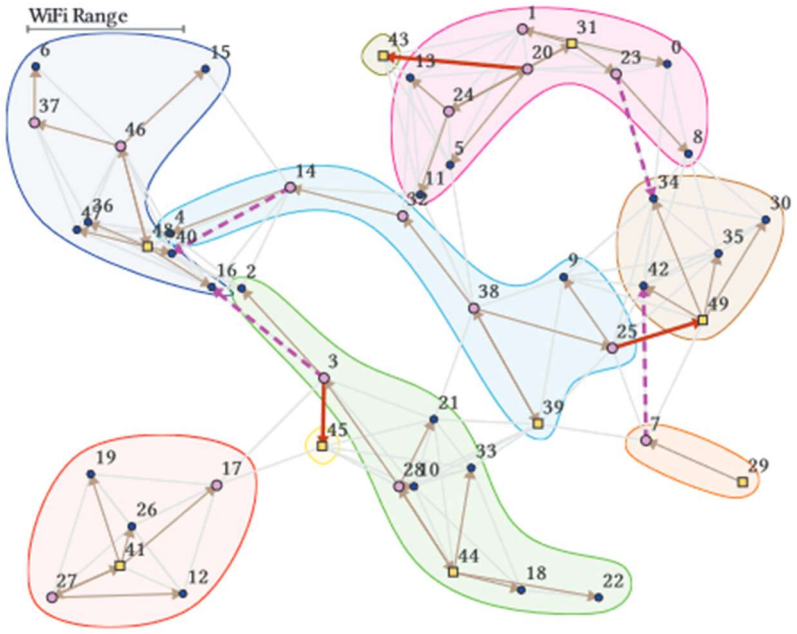


Figura 2-9 Escenario después de la etapa MCI [69]

5. Interconexión de clúster final (FCI)

Esta etapa es una ronda de interconexión de clústeres complementaria, en la que los clústeres intentan conectarse con vecinos más altos que no están conectados. Comienza cuando los GODs de ID más bajos de cada vecindario de clúster reciben notificaciones de finalización de MCI de todos sus clústeres vecinos y luego fluye hacia arriba. Los GODs almacenan todos los clústeres vecinos no conectados superiores en la lista "cltsToConnect", ahora ordenados en orden ascendente. El procedimiento de interconexión de clústeres se usa luego para conectarse a estos clústeres y el procedimiento de fin de interconexiones para enviar notificaciones de fin de FCI a clústeres vecinos superiores (véase la Fig. 2-10).

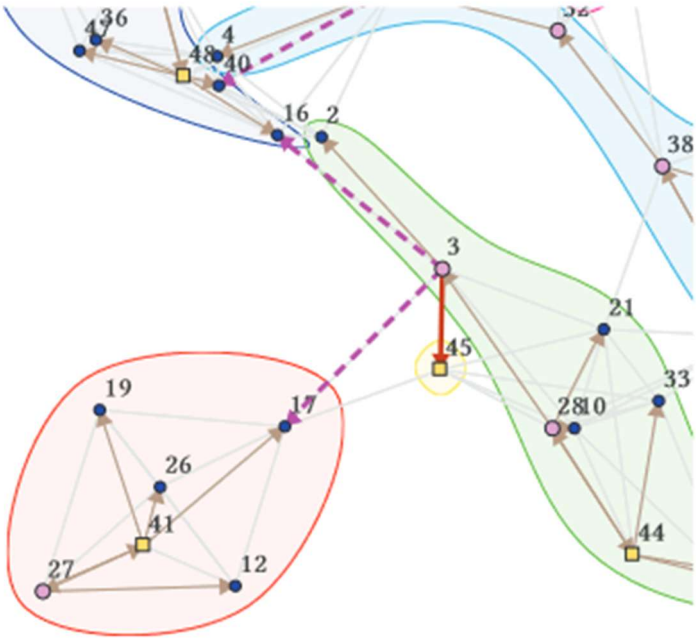


Figura 2-10 Escenario después de la etapa FCI [69]

2.9 Comparativa entre algoritmos de consenso

Los algoritmos de consenso constituyen un área de investigación propia de los sistemas descentralizados. Muchas de las propuestas analizadas han sido diseñadas para mantener una base de datos, un registro o una máquina de estado de forma distribuida para un conjunto de servidores completamente interconectado. Una vez revisados distintos ejemplos de algoritmos de consenso es importante hacer una comparativa entre ellos para analizar su utilidad aplicada a nuestro entorno particular.

El algoritmo Raft se propuso para abordar los problemas de comprensión que tiene asociado el algoritmo Paxos. El propio documento de Raft afirma que es significativamente más comprensible que Paxos e igual de eficiente. Se ha comprobado en múltiples trabajos que la sencillez de Raft viene de su abstracción y su excelente presentación [71]. Al describir el algoritmo Paxos simplificado utilizando el mismo enfoque que Raft, encontramos que los dos algoritmos difieren solo en su enfoque para la elección del líder. De hecho, este proceso en Raft es bastante más intuitivo que en Paxos.

Raft utiliza un proceso de votación para elegir un líder, entre un conjunto de nodos, y ese es el único nodo que permite realizar cualquier actualización en la base de datos. Las actualizaciones son aplicadas posteriormente por el resto de los nodos siguiendo la decisión del líder. Aunque este método permite que el sistema alcance alta consistencia, puede sufrir problemas de escala, ya que tiende a ser intenso en términos de transmisión de mensajes.

En Paxos, se dividen los términos entre los servidores, mientras que en Raft se permite que un seguidor se convierta en candidato en cualquiera de los términos, siempre teniendo en cuenta que los seguidores votarán por un solo candidato por período. Si un líder tiene entradas de registro no comprometidas de un plazo anterior, Paxos los replicará en el término actual mientras que Raft los replicará en su término original. Otro aspecto interesante es que los seguidores de Paxos votarán por cualquier candidato, mientras que en Raft, los seguidores solo votarán por un candidato si la entrada del candidato está, al menos, igual de actualizada.

Los algoritmos basados en Blockchain, sin embargo, son diferentes de las otras propuestas de consenso, ya que funcionan con un número desconocido de participantes sin tratar en ningún momento de averiguarlo. En cambio, sus transiciones son más seguras ya que las medidas introducidas para evitar que los participantes maliciosos alteren las decisiones se asume pagando un mayor costo. Este concepto difiere de la idea de elección de líder, que será necesario para resolver el problema propuesto.
Otro aspecto a considerar es que, cuando se introducen fallos bizantinos, se observa que la principal diferencia con Raft, o con cualquier otro algoritmo de tolerancia a fallos, es que los seguidores ya no confían ciegamente en su líder. Cada bloque requiere múltiples rondas de votación por parte del conjunto de validadores para llegar a un acuerdo mutuo, que se registra como una recogida de firmas sobre el contenido del bloque. Un validador nunca asume que el "líder" o el "proponente del bloque" es correcto u honesto. En su lugar, verifica el bloque propuesto al igual que otros motores de consenso que operan en un entorno no confiable.

Los algoritmos suelen seguir un patrón común estructurando su ejecución en rondas. En cada ronda, un nodo llamado líder trata de imponer una decisión. Una ronda puede no tener éxito debido a fallos o incertidumbres. Los distintos algoritmos difieren en la forma en que eligen un líder para la siguiente ronda: los procesos en el algoritmo Chandra-Toueg rotan el rol de líder entre todos los procesos, mientras que los procesos en Paxos o Raft eligen líderes directamente de manera aleatoria. Estos dos enfoques, a menudo, se denominan paradigma del coordinador rotatorio y basado en el líder, respectivamente.

El algoritmo Chandra-Toueg también se centra en el efecto de la latencia de la solicitud de lectura/escritura en el rendimiento general, donde se intenta encontrar la mejor solución para el sistema de base de datos. Surge debido a la necesidad de escalabilidad y accesibilidad en el sistema de bases de datos distribuidas. Los enfoques anteriores como maestro-esclavo y replicaciones tripartitas no son adecuados en el contexto de clústeres con una gran cantidad de nodos.

Cuando se propone CoNICE, se sigue una estructura similar al modelo publicador/suscriptor para asegurar una difusión consistente de actualizaciones entre los usuarios en entornos de conexión intermitente.

Tanto es así que se demuestra que CoNICE destaca por lograr un grado considerablemente mayor de efectividad en el acuerdo, debido a que aprovecha la nomenclatura y la consistencia multinivel para el ordenamiento y consenso causal más selectivo y eficiente, lo que justifica la aplicabilidad de CoNICE en escenarios prácticos con conexiones intermitentes.

Por último, el algoritmo RedMesh es capaz de crear redes de malla de dispositivos listos para ser habilitados para WiFi Direct. Con esta tecnología se pueden establecer conexiones de comunicación directa entre dispositivos en la red sin necesidad de un punto de acceso centralizado. Esto permite que los dispositivos se comuniquen directamente entre sí, creando una red autónoma y descentralizada, lo que genera ciertas ventajas como una mejora en la resiliencia y la flexibilidad en situaciones en las que la conectividad es limitada o intermitente. Además, este algoritmo es muy escalable y se puede ampliar fácilmente al agregar más nodos a la red. La idea de no depender de un punto de acceso centralizado para enrutar los mensajes a través de la red es bastante poderosa ya que los propios dispositivos pueden utilizar WiFi Direct para establecer conexiones P2P y transmitir mensajes directamente entre ellos.

A pesar de las ventajas consideradas, es importante tener en cuenta que, aunque los algoritmos de consenso mencionados anteriormente han demostrado ser eficientes y aplicables en diversos escenarios, también presentan algunas posibles limitaciones.

En primer lugar, algunos algoritmos, como Raft, pueden mostrar desafíos en términos de escalabilidad cuando se trata de grandes conjuntos de nodos, ya que el proceso de votación y la transmisión de mensajes pueden volverse intensivos. Además, la selección aleatoria de líderes en algoritmos como Paxos y Raft puede resultar en cierta ineficiencia en la toma de decisiones, especialmente en entornos con alta variabilidad de latencia de red.

Por su parte, los algoritmos basados en Blockchain pueden tener limitaciones en cuanto a la velocidad de procesamiento y al costo asociado con las medidas de seguridad implementadas para evitar la manipulación por parte de participantes maliciosos. Asimismo, en los sistemas distribuidos sujetos a fallos bizantinos, la desconfianza en los líderes puede generar un aumento en la complejidad y la latencia de los algoritmos de consenso.

Finalmente, el algoritmo RedMesh, aunque presenta ventajas notables en términos de flexibilidad y resiliencia en redes con conectividad limitada o intermitente, puede presentar limitaciones en cuanto a la cobertura y el alcance de la red, especialmente en entornos de gran escala.

En general, es importante comprender las diferentes características de cada algoritmo de consenso en relación con los requisitos y las características del entorno en el que se implementarán. La elección de un algoritmo de consenso inadecuado puede tener graves consecuencias para la seguridad y el rendimiento del sistema.

3 WiFi Direct para la conexión entre dispositivos

Los científicos estudian el mundo tal como es; los ingenieros crean el mundo que nunca ha sido.

- Theodore von Karman

Este capítulo tiene el propósito de profundizar en uno de los objetivos de este Trabajo y analizar detalladamente el uso de WiFi Direct como una solución altamente eficiente para el descubrimiento y la conexión de dispositivos, teniendo en cuenta la implementación precisa de protocolos Machine-to-Machine (M2M) con el fin de lograr una transferencia de datos óptima y eficaz.

3.1 Fundamentos de WiFi Direct

Esta tecnología también es conocida como WiFi P2P, un estándar lanzado por la WiFi Alliance en octubre de 2010 [72]. Hereda de WiFi todos los mecanismos de infraestructura como la seguridad, el ahorro de batería o la calidad de servicio (QoS) entre otros. Debido a dicha relación de herencia, WiFi Direct permite a los dispositivos mantener la conexión con otros dispositivos WiFi Direct. Comparando esta tecnología con sus análogas *ad hoc* (véase la Tabla 3-1), es bastante obvio como destaca por encima de las demás y es mucho más atractiva a la hora de intercambiar mensajes de texto o voz, al igual que el intercambio de archivos multimedia.

WiFi Direct introduce nuevas oportunidades para implementar redes oportunistas reales a través de los dispositivos móviles de los usuarios. Sin embargo, su especificación original no tiene en cuenta todos los parámetros que pueden surgir de un escenario de red oportunista [73], no solo en términos de requisitos técnicos (por ejemplo, recursos y conectividad disponibles) sino también en términos de características y perfiles de los usuarios, que pueden influir significativamente en el rendimiento del sistema y las interacciones entre dispositivos.

Uno de los principales problemas que sufre la tecnología WiFi Direct es que la comunicación entre pares puede llevarse a cabo solo con los usuarios de un mismo grupo [74]. No es posible comunicarse con miembros de otro grupo diferente que no sea de nuestra red, aunque esté dentro de la zona de cobertura. Esto limita bastante la propagación de mensajes en la red. Otro problema a destacar es la completa dependencia de los GO´s, ya que son el centro del grupo y la red no es capaz de funcionar después de su desconexión.

En general, un grupo de WiFi Direct contiene múltiples GM´s que están interconectados a un GO formando una topología $1: n$. Cada cliente en un grupo puede ser cliente de WiFi Direct o *Legacy Client* (LC). La principal diferencia es que el LC es un cliente WiFi certificado, pero no P2P (no soporta una conexión WiFi Direct). La red que se forma se subdivide en tres fases: descubrimiento de dispositivos, descubrimiento de servicios y formación de grupos.

Standard	*Bluetooth*	*UWB*	*ZigBee*	*WiFi*
IEEE spec.	802.15.1	802.15.3a	802.15.4	802.11a/b/g
Frequency band	2.4 GHz	3.1-10.6 GHz	868/915 MHz; 2.4 GHz	2.4 GHz; 5 GHz
Max. signal rate	1 Mb/s	110 Mb/s	250 Kb/s	54 Mb/s
Nominal range	10 m	10 m	10-100 m	100 m
Nominal TX power	0-10 dBm	-41.3 dBm/MHz	(-25) - 0 dBm	15 - 20 dBm
Number of RF channels	79	(1 - 15)	1/10; 16	14 (2.4 GHz)
Channel bandwidth	1 MHz	500 MHz – 7.5 GHz	0.3/0.6 MHz; 2 MHz	22 MHz
Modulation	GFSK	BPSK, QPSK	BPSK (+ASK), O-QPSK	BPSK, QPSK, COFDM, CCK, M-QAM
Spreading	FHSS	DS-UWB, MB-OFDM	DSSS	DSSS, CCK, OFDM
Coexistence mechanism	Adaptative freq. hopping	Adaptative freq. hopping	Dynamic freq. selection	Dynamic freq. selection, transmit power control (802.11h)
Basic cell	Piconet	Piconet	Star	BSS
Extension of the basic cell	Scatternet	Peer-to-peer	Cluster tree, Mesh	ESS
Max. number of cell nodes	8	8	> 65000	2007
Encryption	E0 stream cipher	AES block cipher (CTR, counter mode)	AES block cipher (CTR, counter mode)	RC4 stream cipher (WEP), AES block cipher
Authentication	Shared secret	CBC-MAC (CCM)	CBC-MAC (ext. of CCM)	WPA2 (802.11i)
Data protection	16-bit CRC	32-bit CRC	16-bit CRC	32-bit CRC

Tabla 3-1 Comparación entre protocolos Bluetooth, UWB, ZigBee y WiFi [63]

3.1.1 Descubrimiento de dispositivos

Los dispositivos WiFi Direct se comunican estableciendo grupos P2P que son funcionalmente equivalentes a las redes de infraestructura WiFi tradicionales. Para ello, la primera fase del proceso de establecimiento de grupo es responsable de encontrar y determinar rápidamente otros dispositivos P2P para que nuestro dispositivo pueda intentar establecer una conexión.

Para poder ser descubiertos, los pares deberán estar en el modo escucha donde el canal elegido ha de estar disponible hasta que se complete la conexión.

Esta fase se subdivide en dos tareas principales (véase la Fig. 3-1):

Por un lado, el escaneo, que consiste en descubrir todos los canales disponibles que se encuentren a su alrededor y acumular toda la información posible sobre cuál será el más adecuado para completar una conexión; por otro lado, la tarea de descubrimiento, que se utiliza para proporcionar un canal común que permita la comunicación P2P de búsqueda simultánea.

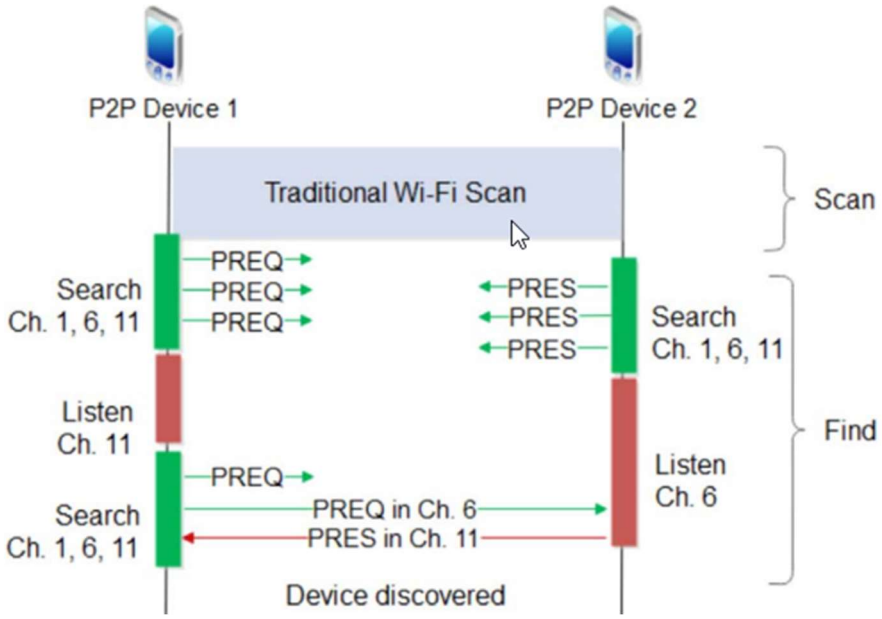


Figura 3-1 Procedimiento de descubrimiento de dispositivos en WiFi Direct [75]

Cuando un dispositivo P2P quiere conectarse a un grupo, éste manda una solicitud al GO para indicar su voluntad de unirse al grupo. En cambio, cuando una búsqueda de un dispositivo P2P descubre un *peer* que ya está conectado a un grupo, envía una solicitud al GO indicando el ID del dispositivo de destino para que esté disponible para intercambiar información de descubrimiento o para comenzar la formación de un grupo.

Sin embargo, este caso no ocurre cuando el dispositivo de búsqueda es un LC, ya que solo puede descubrir GO´s. Mientras tanto, el GO espera que otros dispositivos, ya sean heredados o P2P, lo descubran.

3.1.2 Descubrimiento de servicios

Es un proceso opcional que se ejecuta justo después de la fase de descubrimiento de dispositivos y antes de la formación de grupo. Su propósito es determinar la compatibilidad de servicios ofrecidos por el *peer* descubierto a través del intercambio de información genérica. Con este proceso, se consigue una lista de todos los servicios junto con todo lo que ofrece un dispositivo P2P y determinar si están actualizados.

3.1.3 Formación de grupo

Esta fase consiste en que el dispositivo P2P decide ser parte de un grupo, ya sea creando el suyo propio y convirtiéndose así en GO o a otro grupo junto con los otros dispositivos que han sido ya descubiertos. En este segundo caso, el procedimiento se basa en intercambiar credenciales de los candidatos con el GO y determinar el tipo de formación.

Por lo general, nos encontramos tres tipos; estándar, autónomo y persistente (véase la Fig. 3-2).

En la formación estándar, los dispositivos primero se descubren unos a otros y luego negocian cuál de ellos actuará como GO. Durante el proceso de descubrimiento, los dispositivos escuchan por los canales 1, 6 u 11 en la banda de 2.4 GHz. Justo después, para enlazarse, usan lo que se denomina el "three-way handshake", que se corresponde con un asentimiento en tres partes (request/response/confirmation). Una vez que se han establecidos los roles dentro del grupo, sigue un proceso de comunicación segura usando WiFi Protected Setup (WPS), seguido del protocolo DHCP para asignar direcciones IP. En contraposición, en la formación autónoma, el dispositivo crea el grupo y se asigna a sí mismo como GO.

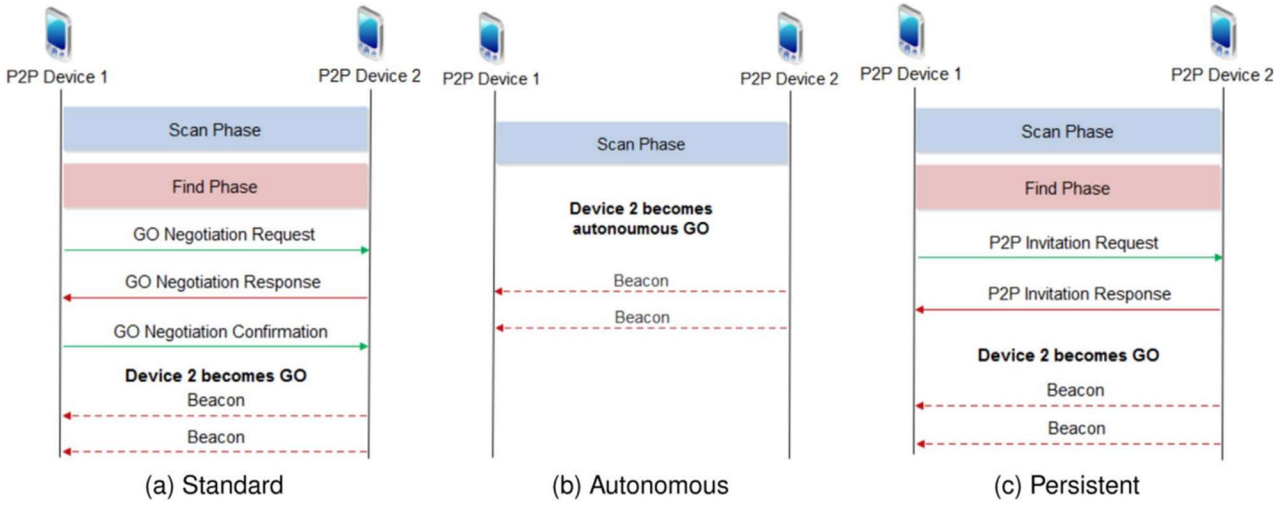


Figura 3-2 Métodos de formación de grupos en WiFi Direct [75]

Por último, en las formaciones persistentes, se permite el almacenamiento de todas las credenciales intercambiadas en la fase inicial. Esta característica, permite reemplazar las negociaciones para decidir el GO con el procedimiento de invitación "two-way handshake". Esto mejora el WPS para un reinicio de grupo más rápido.

3.2 Consideraciones sobre el rendimiento y la eficiencia en la conexión de dispositivos

En este apartado se pretende analizar tres aspectos clave para el correcto funcionamiento de la tecnología WiFi Direct: la seguridad, el ahorro de batería y la calidad de servicio.

La seguridad es un aspecto fundamental para proteger la información transferida entre dispositivos y garantizar la privacidad de los usuarios. En un mundo cada vez más conectado, donde la transferencia de datos se realiza a través de redes inalámbricas y conexiones a Internet, la seguridad se convierte en un desafío importante. Es necesario implementar medidas de seguridad robustas, como encriptación de datos, autenticación de usuarios y protección contra ataques cibernéticos, para evitar la interceptación no autorizada o el acceso no deseado a la información sensible.

Además de la seguridad, el ahorro de batería es un factor crucial en la experiencia del usuario, especialmente en dispositivos móviles que a menudo están limitados por la duración de la batería. La eficiencia energética es esencial para maximizar el tiempo de uso de los dispositivos sin comprometer su rendimiento. Esto implica optimizar el consumo de energía en todas las áreas, como la transmisión de datos, el procesamiento de aplicaciones y el uso de recursos del sistema. Las tecnologías de ahorro de energía, como el modo de suspensión, la gestión inteligente de energía y la optimización del uso del procesador, desempeñan un papel vital en la prolongación de la vida útil de la batería y, por lo tanto, en la satisfacción del usuario.

Por otro lado, la calidad de servicio también es un factor esencial para asegurar una transferencia de datos fluida y sin interrupciones. La experiencia del usuario depende de la capacidad de los dispositivos y las redes para proporcionar una conectividad estable y un rendimiento óptimo. La calidad de servicio abarca aspectos como la velocidad de conexión, la latencia, la disponibilidad de red y la capacidad de manejar cargas de datos pesadas. Para garantizar una experiencia satisfactoria, es necesario contar con infraestructuras de red confiables, protocolos de comunicación eficientes y algoritmos de enrutamiento inteligentes que minimicen las interrupciones y optimicen el rendimiento.

3.2.1 Seguridad

Como ya se ha mencionado anteriormente, la tecnología WiFi Direct permite la conexión directa entre dispositivos compatibles sin necesidad de un punto de acceso tradicional. Esta conexión directa plantea desafíos en términos de seguridad [76], ya que los dispositivos se conectan entre sí sin la protección adicional de una red de área local (LAN) o de un enrutador.

Los mecanismos de seguridad para comunicaciones seguras en redes inalámbricas 802.11 se han desarrollado en el siguiente orden cronológico [77]:

- **Wired Equivalent Privacy (WEP):** Fue el primer mecanismo de seguridad implementado en redes inalámbricas 802.11. Sin embargo, WEP demostró ser vulnerable a ataques y su nivel de seguridad fue considerado insuficiente [78] [79].

- **WiFi Protected Access (WPA):** Desarrollado como una mejora de seguridad sobre WEP [80]. Introdujo mejoras en la autenticación y en el cifrado de datos para ofrecer una mayor protección. WPA fue una solución temporal mientras se desarrollaba el estándar más sólido, WPA2.

- **802.11i (WPA2):** Este estándar de seguridad para redes inalámbricas 802.11 introdujo un cifrado de datos más seguro, utilizando el protocolo de cifrado AES (Advanced Encryption Standard), a la vez que mejoró la autenticación y la gestión de claves [81]. Sin embargo, a medida que se avanza hacia una mayor seguridad en las redes inalámbricas, surgen nuevas generaciones de protocolos de seguridad como es el caso de WPA3.

- **WPA3:** Es la convención de seguridad más reciente con los estándares más altos. Anunciado por la WiFi Alliance en 2018, utiliza el protocolo de Autenticación Simultánea de Iguales, lo cual hace que muchos ataques sean imposibles, protegiendo así la red de potenciales ataques que pudieran comprometer la seguridad con la configuración de WPA2 [82]. WPA3 contempla un cifrado de datos individualizado, de tal forma que, aun cuando un atacante averigüe la contraseña, no podrá desencriptar el tráfico que se haya realizado antes de la intrusión.

Estos avances en los mecanismos de seguridad reflejan la evolución y el esfuerzo continuo por mejorar la seguridad en las redes inalámbricas 802.11 y proteger la privacidad de los usuarios.

En el caso de WiFi Direct, la provisión de seguridad comienza después de que se haya realizado el descubrimiento y, si es necesario, se hayan negociado los roles respectivos.

Para garantizar la seguridad, se utilizan protocolos de seguridad como WPA2-PSK (WiFi Protected Access 2 - Pre-Shared Key) para cifrar la comunicación y protegerla de posibles ataques. El uso de una clave de seguridad compartida entre los dispositivos asegura que solo los usuarios autorizados puedan acceder a la información transferida.

3.2.2 Ahorro de batería

Constituye otro aspecto importante a considerar en WiFi Direct, especialmente en dispositivos con baterías limitadas, como smartphones o tablets. Como se dijo en apartados anteriores, los dispositivos pueden actuar tanto de GO (soft-AP) como de cliente P2P, y se espera que ambos dispositivos ahorren energía tanto como sea posible.

Los que se conviertan en GO tendrán que realizar ciertas funciones (como la emisión de señales de baliza y el reenvío de datos) que provocará un mayor consumo de energía en comparación con un cliente P2P, que puede beneficiarse de los protocolos de ahorro de energía ya definidos en IEEE 802.11.

Con el fin de abordar el desequilibrio en el consumo de energía entre un GO y un cliente P2P, y permitir que los dispositivos con batería operen de manera eficiente, la especificación de WiFi Direct define dos nuevos protocolos que pueden ser utilizados por un GO [83]:

Protocolo de Ahorro de Energía Oportunista (OPS): Permite al GO coordinar el ahorro de energía de los dispositivos miembros del grupo, sincronizando sus períodos de actividad y descanso para reducir el consumo de energía global. Esto permite que los dispositivos con batería operen de manera más eficiente y maximicen la duración de la misma.

Protocolo de Aviso de Ausencia (NoA): Permite al GO notificar a los dispositivos vecinos sobre su período de inactividad planificado. Esto permite que los dispositivos vecinos ajusten su funcionamiento y ahorren energía durante ese tiempo, evitando la necesidad de estar constantemente escuchando y consumiendo energía innecesariamente.

Estos dos protocolos introducidos en la especificación de WiFi Direct ayudan a equilibrar el consumo de energía entre los GO y los clientes P2P, permitiendo un funcionamiento eficiente y prolongando la duración de la batería en dispositivos con recursos limitados.

3.2.3 Calidad de servicio

La vida útil de las aplicaciones que utilizan WiFi Direct depende de la selección adecuada del GO. Si un propietario de grupo se vuelve disfuncional, la calidad de servicio (QoS) de estas aplicaciones se verá comprometida debido a la pérdida de información, el tiempo desperdiciado en reparar el grupo y el agotamiento de recursos.

La calidad de servicio constituye aspecto crítico en la comunicación inalámbrica y en particular en WiFi Direct, ya que afecta directamente a la eficiencia y eficacia de las transmisiones de datos. La implementación de algoritmos de consenso anticipado y recurrente en este contexto es esencial para garantizar una buena QoS, ya que permiten la identificación y resolución temprana de conflictos en la comunicación entre dispositivos.

En este contexto, distintos trabajos han llevado a cabo una amplia investigación sobre cómo elegir propietarios de grupo para mejorar la calidad del software y las aplicaciones [84] [85] [86].

Las distintas aportaciones proporcionan una metodología de software para mejorar la QoS en términos de confiabilidad y, por extensión, la fiabilidad y la capacidad de reparación del software sobre la tecnología WiFi Direct, así como la optimización de los recursos utilizados para realizar los servicios.

A continuación, se presentan limitaciones técnicas del protocolo WiFi Direct que son consideradas en el diseño de nuestro modelo y se muestran cómo estas limitaciones afectan a los diferentes atributos de QoS [87] .

Selección aleatoria: La selección del GO es un proceso aleatorio porque se basa en un valor llamado "valor de intención" establecido también de manera aleatoria para cada dispositivo durante la fase de negociación. Esta forma de selección puede llevar a elegir un GO débil, lo que resulta en un rendimiento deficiente para el grupo creado y en el establecimiento de un grupo poco confiable [88].

Falta de mecanismo de delegación: Hasta ahora, en el estándar de WiFi Direct no se ha definido ningún mecanismo de delegación o transformación fluida al papel de GO dentro del grupo creado. Si el propietario del grupo desea abandonar o finalizar la sesión del grupo por cualquier motivo, el grupo se romperá y, por lo tanto, los clientes perderán su conexión. Como resultado, se perderá la información y el servicio no se completará de manera adecuada.

Falta de reformación sin problemas: Cuando el grupo se termina, el grupo debe ser reestablecido desde cero y todos los procesos de formación del grupo deben repetirse; sin embargo, no hay garantía de que el grupo recién creado incluya a todos los miembros anteriores del grupo. Existe una alta probabilidad de crear múltiples grupos pequeños a partir del grupo roto. Este proceso de reformación de manera convencional consume mucho tiempo y prolonga la reparación del grupo, lo que reduce la confiabilidad y la capacidad de reparación del grupo.

Los esfuerzos de investigación en esta línea se centran en mejorar la selección del GO para que puedan prolongar la vida útil del grupo y así garantizar una alta disponibilidad para el grupo creado, o en la introducción de un nodo de respaldo para el GO.

Por poner varios ejemplos, algunos investigadores evalúan el Indicador de Intensidad de Señal Recibida (RSSI) para seleccionar el GO o el dispositivo de respaldo. Un valor alto de RSSI resulta en un bajo consumo de energía para transferir datos entre dispositivos, por lo tanto, seleccionar el GO o el respaldo con el RSSI más alto prolongará la vida útil del grupo. Zhang et al. [84] utilizan el valor de RSSI del par para mejorar el mecanismo de selección del GO. Si hay dos dispositivos A y B que desean comunicarse, el dispositivo A recibirá una solicitud de B y luego calculará el RSSI del dispositivo B y actualizará la tabla de RSSI con el promedio de RSSI, este proceso se realizará para cada par. Luego se calculará el valor de intención basado en el valor de RSSI más grande. Cuando dos pares tienen el mismo valor de RSSI, se considerará otro factor como la dirección MAC.

Otros, sin embargo, como Shahin et al. [89] introducen el protocolo de Formación y Comunicación Eficiente de Múltiples Grupos (EMC) para grupos WiFi Direct, permitiendo así que múltiples grupos se comuniquen entre sí. Utilizan la potencia de la batería restante del par para seleccionar el mejor nodo que se ajuste al rol de GO. Se utiliza una fase de descubrimiento de servicios opcional para permitir que los pares descubiertos compartan el porcentaje de batería.

Por último, Chaki et al. [90] introducen una nueva metodología para restablecer el grupo roto utilizando la fase de negociación. Inyectan una bandera de emergencia GO (EGO) de un solo bit durante la fase de negociación. EGO es la bandera que muestra la disposición del par para reemplazar al GO actual en caso de fallo o salida del GO. Los autores consideran diferentes parámetros como la potencia de batería residual, la velocidad de la CPU y el rango de transmisión para calcular el valor de EGO.
El GO recopila los valores de EGO de todos los pares o el propio par comparte su valor de EGO con el GO, de modo que el GO puede marcar al par con el valor de EGO más alto para restablecer el grupo en caso de falla o salida del GO. Como resultado, reducen el tiempo requerido para reformar el grupo desde cero.

3.3 Protocolos M2M para la transferencia eficiente de datos

Los protocolos M2M representan una parte importante del uso de WiFi Direct para la transferencia eficiente de datos en redes distribuidas. A través de ellos se realiza la comunicación entre dispositivos sin intervención humana, lo que mejora la eficiencia de la transferencia de datos y reduce los errores.

En general, una de las principales diferencias entre los protocolos de comunicación radica en su modelo de interacción. Por un lado, existe el modelo de petición/respuesta, y, por otro lado, el modelo de publicador/suscriptor.

El modelo de petición/respuesta es uno de los paradigmas de comunicación más básicos. Representa un patrón de intercambio de mensajes común en las arquitecturas cliente/servidor. Permite que un cliente solicite información a un servidor, que recibe la solicitud, la procesa y envía una respuesta. Este tipo de intercambio de información suele ser gestionado de manera centralizada.

Por su parte, el modelo de publicador/suscriptor es de particular interés, ya que los protocolos basados en este modelo surgen como una alternativa al modelo tradicional. Aquí, el “cliente” con el rol de suscriptor no tiene que solicitar información al servidor. En lugar de eso, el suscriptor interesado en recibir mensajes se suscribe a eventos específicos (temas) dentro del sistema. El cliente se suscribe a un intermediario, que actúa como el punto central en esta arquitectura y se encarga de filtrar los mensajes entrantes y enrutarlos adecuadamente entre los publicadores y los suscriptores.

Por estas razones, este modelo de interacción es interesante para las aplicaciones de IoT debido a su capacidad para proporcionar escalabilidad y simplificar las interconexiones entre diferentes dispositivos, permitiendo una comunicación dinámica, de muchos a muchos y asíncrona.

A continuación, se revisan algunos ejemplos de protocolos M2M [91] que se adaptan bien a la tecnología WiFi Direct.

3.3.1 MQTT - Message Queuing Telemetry Transport

MQTT [92] es un protocolo de mensajería diseñado especialmente para dispositivos con restricciones de recursos. Fue desarrollado originalmente por IBM y ahora es un estándar abierto. Utiliza el modelo de comunicación publicador/suscriptor, en el cual los propios clientes no requieren actualizaciones, lo que reduce el uso de recursos. Esto hace que este modelo sea óptimo para su implementación en entornos con limitaciones de ancho de banda.

Se caracteriza por utilizar TCP/IP como protocolo de transporte, el cual, está orientado a la conexión y es altamente confiable. Además, soporta el intercambio de mensajes bidireccionales. Fue diseñado para la comunicación asíncrona, donde las suscripciones o publicaciones entre diferentes entidades ocurren de forma paralela. Además, el protocolo es capaz de proporcionar transferencias confiables [93].

El publicador (cliente) publica mensajes en el intermediario (*broker*) y asigna un tema (*topic*) a cada mensaje. Un suscriptor (servidor) se suscribe a un tema específico. Cuando el intermediario recibe una publicación, reenvía el mensaje a los servidores suscritos al *topic* del mensaje. Un sistema MQTT típico consta de muchos dispositivos pequeños distribuidos, como sensores, que funcionan como publicadores. Sus datos son agregados por un intermediario central y consumidos por un suscriptor (véase la Fig. 3-3).

Una de sus principales ventajas es su consumo eficiente de energía y ancho de banda [95], lo que lo hace una opción económica para su implementación. Además, MQTT cuenta con alta escalabilidad y capacidad de integración con otras tecnologías. Sin embargo, una desventaja importante es que no ofrece autenticación integrada, lo que puede ser un riesgo para aplicaciones críticas que requieren seguridad adicional [91]. Se utiliza comúnmente en aplicaciones de IoT para la monitorización y control de dispositivos conectados, como los sensores.

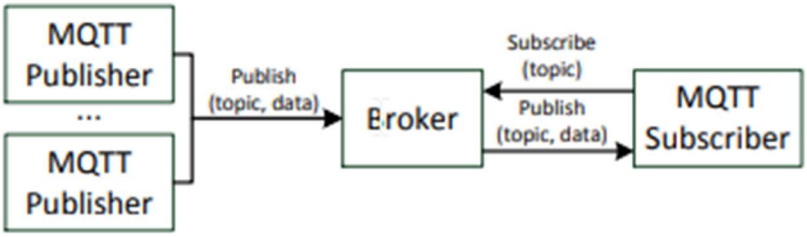


Figura 3-3 Comunicación MQTT [94]

3.3.2 CoAP - Constrained Application Protocol

CoAP [96] es un protocolo de aplicación web ligero y eficiente diseñado para dispositivos con restricciones de recursos, que utiliza el modelo solicitud-respuesta para la comunicación entre dispositivos. A diferencia de MQTT, CoAP utiliza Identificador de Recursos Universal (URI) en lugar de temas.

Este protocolo soporta la transferencia de datos en formato JSON y XML, lo que permite una fácil integración con otros sistemas. En particular, éste utiliza el protocolo de transporte UDP para la comunicación y admite el cifrado de seguridad DTLS (Datagram Transport Layer Security) de esta manera, los clientes y servidores se comunican a través de una conexión [97].

Una de sus principales ventajas es su bajo consumo de energía y ancho de banda, lo que lo hace una excelente opción para dispositivos IoT. Asimismo, su facilidad de implementación y configuración, y la compatibilidad con múltiples plataformas y sistemas operativos, lo convierten en una opción popular en la industria.

No obstante, CoAP también cuenta con algunas limitaciones, como su limitado soporte para seguridad y autenticación, y que no es adecuado para aplicaciones que requieren alta disponibilidad o tiempos de respuesta muy cortos.

Aun así, CoAP ofrece más funcionalidades que MQTT, como el soporte para la negociación de contenido para expresar una representación preferida de un recurso. Esto permite que el cliente y el servidor evolucionen de forma independiente, añadiendo nuevas representaciones sin afectarse mutuamente.

Se utiliza comúnmente en aplicaciones IoT para la monitorización y control de dispositivos, así como para la recopilación de datos en tiempo real.

3.3.3 AMQP - Advanced Message Queuing Protocol

El protocolo AMQP, surgido de la industria financiera, se basa en un protocolo de transporte confiable subyacente como TCP y permite la comunicación asincrónica de publicador/suscriptor con mensajería.

Su principal ventaja radica en su capacidad de almacenamiento y reenvío, asegurando la confiabilidad incluso tras interrupciones en la red [98].

AMQP facilita la comunicación efectiva y eficiente entre dispositivos, con soporte para transferencia de datos en múltiples formatos. Al definir un protocolo a nivel de cable, las implementaciones de AMQP pueden interoperar entre sí [99].

Además, AMQP destaca por ofrecer seguridad y autenticación avanzadas, convirtiéndolo en una opción adecuada para aplicaciones empresariales de alta disponibilidad. Sin embargo, es importante tener en cuenta que su implementación puede requerir una mayor inversión en hardware y software, y no es adecuado para aplicaciones con baja potencia y ancho de banda limitado.

3.3.4 DDS - Data Distribution Service

Finalmente, el protocolo DDS fue creado como un *middleware* de red para evitar las desventajas de la arquitectura centralizada de publicador/suscriptor. Es un protocolo comúnmente basado en TCP que cuenta con nodos descentralizados de clientes en todo el sistema, permitiendo que estos nodos se identifiquen como suscriptores o publicadores a través de un servidor de localización.

Es una tecnología de comunicaciones en tiempo real que se utiliza en aplicaciones críticas donde la velocidad y la precisión de los datos son fundamentales. El uso de este sistema evita la necesidad de que los usuarios identifiquen la ubicación de otros nodos potenciales o los temas en los que están interesados, ya que los nodos de DDS se auto descubren en una red y envían/reciben telemetría de forma anónima basada únicamente en los temas (véase la Fig. 3-4).

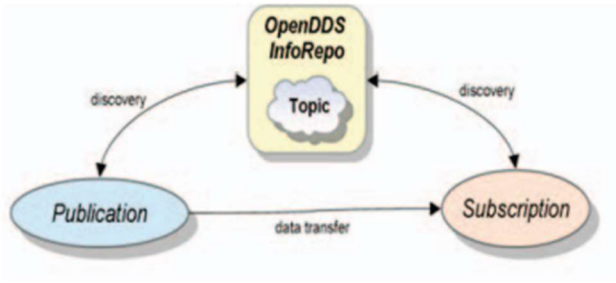


Figura 3-4 Arquitectura de comunicaciones DDS [100]

En DDS, los publicadores y suscriptores pueden comunicarse como iguales a través del bus de datos, lo que permite un intercambio de datos asincrónico basado en sus intereses. El hecho de que no haya intermediario disminuye la probabilidad de fallos del sistema, ya que no hay un único punto de fallo para todo el sistema, lo que lo hace más confiable. Ambos lados de la comunicación están desacoplados entre sí, y un publicador puede publicar datos, aunque no haya suscriptores interesados.

Introduce un concepto denominado Global Data Space [64], un espacio virtual donde las aplicaciones pueden compartir la información de una forma asíncrona leyendo y escribiendo en contenedores estructurados y lógicos de datos denominados *topics*.

Una de las características más destacadas del protocolo DDS es su escalabilidad, que proviene de su soporte para el descubrimiento dinámico. El proceso de descubrimiento, logrado a través del protocolo de descubrimiento incorporado de DDS, permite a los suscriptores averiguar qué publicadores están presentes y especificar la información en la que están interesados con la calidad de servicio deseada definida, y a los publicadores publicar sus datos.

Una de las ventajas de usar DDS es un amplio conjunto de políticas de QoS ofrecidas (más de 20 QoS según lo definido por el estándar). Para el mecanismo de seguridad [101], DDS implementa diversas soluciones. Basado en un protocolo de transporte de elección, TLS puede usarse en caso de que TCP sea el protocolo de transporte, o el protocolo DTLS en caso de que se use UDP. De manera similar, para TLS también se han propuesto mecanismos mejorados en caso de entornos limitados, debido al exceso de sobrecarga que supone DTLS. Con este fin, la Especificación de Seguridad de DDS define un extenso modelo de seguridad y una arquitectura de Interfaz de Plugin de Servicio (SPI) diseñados para implementaciones de DDS adecuadas en sistemas IoT [102].

3.3.5 Comparativa entre protocolos M2M

Para entender el alcance que puede llegar a tener esta línea de estudio se pueden destacar algunos artículos como el de Chen et al. [100], que tiene como objetivo proporcionar una comprensión completa del rendimiento real de varios candidatos de protocolos IoT en una red de baja calidad con baja confiabilidad, alta latencia y ancho de banda estrecho.

A continuación, se muestran los protocolos M2M mencionados anteriormente en función de sus características principales (véase la Tabla 3-2).

Protocol	*Req.-Rep.*	*Pub.-Sub*	*Transport*	*QoS*	*Security*
MQTT		✓	TCP	3 levels	TLS/SSL
CoAP	✓	✓	UDP	Limited	DTLS
AMQP	✓	✓	TCP	3 levels	TLS/SSL
DDS		✓	TCP/UDP	Extensive	TLS/DTLS/DDS sec

Tabla 3-2 Comparativa protocolos M2M [91]

A través de la variación de cada una de las variables independientes, como el ancho de banda de la red, la tasa promedio de pérdida de paquetes de la red y la latencia de la red, en dicho artículo se realizaron pruebas para evaluar su impacto en el ancho de banda consumido, la pérdida real de paquetes y la latencia experimentada de la telemetría; estas métricas se analizan como indicadores de varios aspectos del rendimiento de todos los protocolos. Además, se consideran diversos escenarios con una combinación de ancho de banda estrecho, alta latencia y alta pérdida de paquetes para permitir una mejor evaluación del rendimiento del protocolo en una red de baja calidad realista.

En cuanto al rendimiento del protocolo, este estudio ha revelado que tanto DDS como MQTT, que son protocolos basados en TCP, no experimentan pérdida de paquetes en condiciones degradadas de la red, con una tasa promedio de pérdida de paquetes de la red de hasta 25% y una latencia del sistema de hasta 400 ms. La tecnología DDS y su alta confiabilidad garantiza la entrega correcta de un gran número de muestras, incluso con altas frecuencias de muestreo [103].

La Tabla 3-3 muestra un resumen de las fortalezas y debilidades de cada protocolo expuesto en este capítulo.

Protocolo	*Fortalezas*	*Debilidades*
MQTT	Simple Bajo uso de memoria y de CPU Baja latencia en tasas de muestreo bajas Comunidad activa de desarrolladores	No proporciona descubrimiento automático No tiene suficiente seguridad a nivel de protocolo Alta latencia en tasas de muestreo altas No es adecuado para aplicaciones en tiempo real con altas tasas de muestreo
CoAP	Simple Bajo uso de memoria y de CPU Bajo consumo de ancho de banda Descubrimiento de recursos	Alta latencia QoS deficiente en entrega de mensajes No apto para aplicaciones en tiempo real (SSL/TLS no disponible) Falta de soporte para herramientas y bibliotecas

AMQP	Muy extendido Alta interoperabilidad entre diferentes proveedores Diseñado para admitir aplicaciones críticas en la industria financiera	No proporciona descubrimiento automático No garantiza la interoperabilidad No apto para aplicaciones en tiempo real Falta de bibliotecas de código abierto
DDS	Descubrimiento automático Alta fiabilidad Alta escalabilidad Interoperabilidad Tolerancia a fallos Amplio soporte de calidad de servicio Adecuado aplicaciones de tiempo real No hay intermediarios que actúen como cuellos de botella	Cantidad de memoria utilizada Complejidad de desarrollo y configuración Falta de bibliotecas de código abierto Concebido originalmente para LAN aisladas

Tabla 3-3 Fortalezas y debilidades protocolos M2M [104]

Llegados a este punto, tras analizar y comparar los protocolos expuestos, se identifica una opción prometedora para la implementación de aplicaciones M2M: el protocolo DDS. Esta alternativa resulta altamente interesante debido a su arquitectura descentralizada y altamente escalable, que elimina la necesidad de elementos intermediarios y ofrece ventajas significativas para entornos distribuidos.

A pesar de sus beneficios, es importante tener en cuenta que DDS aún no ha sido ampliamente adoptado por la industria [108]. Sin embargo, es importante destacar que esta tendencia podría cambiar en el futuro, especialmente con el auge de implementaciones de código abierto que están actualmente alcanzando cierta madurez.

No obstante, en este Trabajo particular, se toma la decisión de decantarse por el protocolo MQTT debido a su capacidad demostrada para operar de manera confiable en condiciones desafiantes de red, así como su eficiencia y simplicidad de uso. Estas características lo convierten en una opción adecuada para aplicaciones M2M, especialmente en dispositivos con limitaciones de memoria y potencia de procesamiento.

Aunque MQTT presenta ciertas debilidades, como la falta de descubrimiento automático y seguridad a nivel de protocolo, estas limitaciones pueden ser mitigadas mediante la implementación de medidas adicionales y la adaptación a los requisitos específicos de la aplicación en cuestión. Además, la comunidad activa de desarrolladores de MQTT ha contribuido a su continua mejora y expansión en el ámbito de IoT, lo que refuerza su posición como una elección sólida para este estudio.

4 MATERIALES

Nunca te fíes de un ordenador que no puedas tirar por la ventana
-Steve Wozniak-

La tecnología WiFi Direct introduce nuevas oportunidades para desplegar redes oportunistas reales a través de los dispositivos móviles de los usuarios. Se están desarrollando múltiples trabajos de investigación [73] acerca de la viabilidad de crear redes oportunistas sobre el marco de WiFi Direct mediante el análisis del rendimiento de protocolos en escenarios reales con un número variable de dispositivos móviles. Este análisis experimental a través de pruebas reales permite analizar las ventajas y limitaciones de usar distintos protocolos para implementar este tipo de redes a gran escala, al mostrar los tiempos requeridos para formar un grupo de tamaño variable y las mejores configuraciones para soportar operaciones de redes oportunistas y aplicaciones de capa superior.

Sin embargo, esto representa solo el primer paso en esta dirección, ya que las redes oportunistas se caracterizan por presentar distintos parámetros dinámicos que deben tenerse en cuenta y que tienen un impacto significativo en el rendimiento de este protocolo, por ejemplo, recursos y conectividades disponibles, al igual que características y perfiles de los usuarios, que pueden influir notablemente en el rendimiento del sistema y las interacciones entre dispositivos.

Aunque WiFi Direct está cada vez más implementado e integrado en numerosos dispositivos Android y es frecuente que los investigadores utilicen prototipos en dispositivos Android para evaluar sus resultados, este enfoque experimental no siempre resulta adecuado para la evaluación de escenarios complejos. Alternativamente, se recurre al uso de simuladores para evaluar el rendimiento de la red en escenarios complejos con un gran número de nodos antes de proceder al prototipado.

En este capítulo se presenta primero WiDiSi [109], una herramienta de simulación de redes WiFi Direct que funciona como una extensión de PeerSim [110], un software de código abierto ampliamente utilizado como marco de simulación para redes P2P a gran escala. En este Trabajo se aplicará para simular y evaluar el rendimiento de redes WiFi Direct en diferentes escenarios y configuraciones. En particular, en el siguiente capítulo se utilizará para confirmar la primera hipótesis de que es posible mejorar el rendimiento de un algoritmo de consenso, centrándonos en el número de nodos que conforman la red. A través de pruebas realizadas con la herramienta de simulación WiDiSi sobre el algoritmo de consenso RedMesh, se obtendrán resultados que permitirán analizar distintas características que podrían mejorar la efectividad de ejecución del algoritmo de consenso. Estos resultados proporcionarán información valiosa para comprender el impacto del tamaño de la red en el rendimiento del algoritmo y podrían conducir a futuras mejoras y optimizaciones.

Posteriormente se presentará una implementación del algoritmo Raft donde se evaluará el desempeño del algoritmo y su utilidad enfocándolo a un entorno de redes WiFi Direct, proporcionando así información relevante para validar la segunda hipótesis y analizar la eficacia de este algoritmo en entornos de redes inalámbricas de cara a poder enfocar el desarrollo y propuesta del nuevo algoritmo de consenso.

Por último, para la transferencia de datos dentro del algoritmo propuesto se realizará mediante protocolos M2M, donde se desarrollará una última simulación utilizando las herramientas Mosquitto y MQTTX.

En los apartados que siguen se presentan las herramientas utilizadas para la realización de los desarrollos y estudios que se presentan en los siguientes capítulos.

4.1 WiDiSi para la evaluación de redes WiFi Direct en escenarios complejos

Antes de presentar esta herramienta de simulación, se debe conocer la plataforma Peersim, un entorno de simulación de redes de pares P2P de código abierto y gratuito, que se utiliza comúnmente en la investigación y la enseñanza de sistemas distribuidos. Es una herramienta basada en eventos discretos que permite a los usuarios definir y modelar redes de pares, establecer protocolos y analizar la dinámica del sistema en función de diversas medidas de rendimiento.

PeerSim fue diseñado para fomentar la programación modular basada en objetos llamados bloques de construcción. Cada bloque es fácilmente reemplazable por otro componente que implementa la misma interfaz. Es altamente configurable y extensible, lo que significa que los usuarios pueden ajustar la simulación a sus necesidades específicas. También ofrece una amplia gama de protocolos de red y modelos de comportamiento que se pueden utilizar para simular diversas aplicaciones y sistemas, incluyendo algoritmos de consenso para redes de pares.

A su vez es compatible con una variedad de sistemas operativos y lenguajes de programación, lo que lo hace fácilmente accesible para usuarios de diferentes plataformas y niveles de experiencia. Además, cuenta con una gran comunidad de desarrolladores y usuarios, lo que garantiza el soporte continuo y el desarrollo de nuevas características y funcionalidades.

Juntos, Peersim y WiDiSi se pueden utilizar para simular y evaluar diferentes algoritmos de consenso distribuido en un entorno controlado y reproducible.

El flujo de trabajo principal de un modelo de simulación es el siguiente:

1. Elegir un tamaño de red (número de nodos).

2. Elegir uno o más protocolos para experimentar e inicializarlos.

3. Elegir uno o más objetos de control para monitorizar las propiedades que interesan y modificar algunos parámetros durante la simulación.

4. Ejecutar su simulación invocando la clase Simulator con un archivo de configuración, que contiene la información anterior.

Todos los objetos creados durante la simulación son instancias de clases que implementan una o más interfaces. Las principales se presentan en la Tabla 4-1.

Node	La red P2P está compuesta por una serie de nodos, los cuales son contenedores de protocolos. La interfaz del nodo proporciona acceso a los protocolos que contiene, al igual que a un ID fijo del propio nodo.
Protocol	Las clases que implementan la interfaz Protocol se instalan en cada nodo. Luego, definen los comportamientos que los nodos deben realizar cada vez que se ejecute un ciclo de simulación o cuando se desencadena por un evento.
Linkeable	Normalmente esta interfaz esta implementada por protocolos. Proporciona un servicio a otros protocolos para acceder a un conjunto de nodos vecinos.
Control	Las clases que implementan esta interfaz se pueden programar para su ejecución en ciertos puntos durante la simulación. Esta interfaz se utiliza para monitorear y administrar el estado de la red observando o modificando la simulación para su control.

Tabla 4-1 Interfaces principales de Peersim

Con WiFi Direct se consigue una comunicación P2P destinada a proporcionar una conexión directa entre dispositivos WiFi sin ningún punto de acceso fijo. Para simular este tipo de red en Peersim se considerará que consta de varios dispositivos/nodos que se ejecutan dentro de un contenedor de nodos, responsable de realizar un seguimiento de los nodos que se encuentran dentro de la simulación en todo momento. Cada nodo ejecuta un conjunto de aplicaciones que representan los comportamientos que el nodo debería tener en un dispositivo real. Cada nodo también incluye la interfaz WiFi Direct que implementa el protocolo WiFi P2P dentro del propio simulador.

A través de la API WiFi Direct en WiDiSi es posible detectar otros dispositivos y conectarse a ellos. Esto permite que una aplicación descubra los pares disponibles, configure la conexión con los pares y consulte la lista de pares. Cuando se establece una conexión P2P a través de WiFi, el dispositivo continúa manteniendo la conexión de enlace ascendente a través del móvil o cualquier otra red disponible para la conectividad a Internet en el dispositivo. Esta API cuenta con métodos que permiten detectar pares, solicitarlos y conectarse a ellos y se definen dentro de la clase WifiP2pManager (véase la Tabla 4-2).

Método	Descripción
initialize()	Registra la aplicación con el marco de trabajo WiFi. Debe llamarse a este método antes de llamar a cualquier otro método P2P WiFi.
connect()	Inicia una conexión entre pares con un dispositivo que tiene la configuración especificada.
requestConnectInfo()	Solicita la información de conexión de un dispositivo.
createGroup()	Crea un grupo entre pares con el dispositivo actual como propietario del grupo.
removeGroup()	Elimina el grupo entre pares actual.
discoverPeers()	Inicia la detección de pares.
requestPeers()	Solicita la lista actual de pares detectados.

Tabla 4-2 Métodos principales de la clase WifiP2pManager

4.2 Visual Paradigm para diagramas UML

Visual Paradigm 17.1 es una herramienta de modelado visual que permite a los profesionales de la planificación y diseño de redes WiFi crear diagramas y modelos visuales que representan diferentes aspectos de la red [111]. Proporciona una interfaz intuitiva (véase la Fig. 4-1) y una amplia gama de opciones de modelado para ayudar a analizar y evaluar las propiedades de la red inalámbrica.

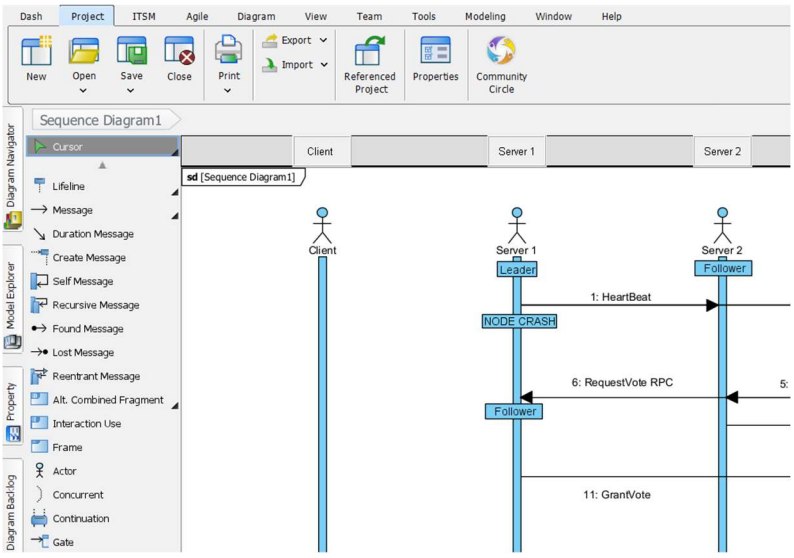


Figura 4-1 Entorno de modelado dentro de Visual Paradigm

Una de las principales características de Visual Paradigm 17.1 es su capacidad para representar visualmente la topología de la red, incluyendo los puntos de acceso, los dispositivos clientes y las interconexiones entre ellos. Esto permite a los usuarios tener una visión clara de cómo se estructura la red y cómo se comunican los diferentes componentes.

4.3 Eclipse para desarrollo de código en lenguaje Java

Eclipse (Fig. 4-2) es un entorno de desarrollo integrado (IDE) ampliamente utilizado para el desarrollo de aplicaciones en el lenguaje de programación Java [112]. Esta poderosa herramienta proporciona un conjunto completo de características y funcionalidades que facilitan la escritura, edición, depuración y pruebas de código Java de manera eficiente y productiva.

Figura 4-2 Inicio IDE de Eclipse

4.4 Explorando el canal inalámbrico mediante Acrylic WiFi Analyzer

Este analizador de WiFi recopila datos de varios canales de red y puntos de acceso, a la vez que presenta una visión general con paneles de control e informes visuales [113]. Mediante la detección y visualización de los canales utilizados por las redes cercanas, así como la identificación de puntos de interferencia, esta herramienta proporciona una visión clara de la situación actual de la red (Véase la Fig. 4-3).

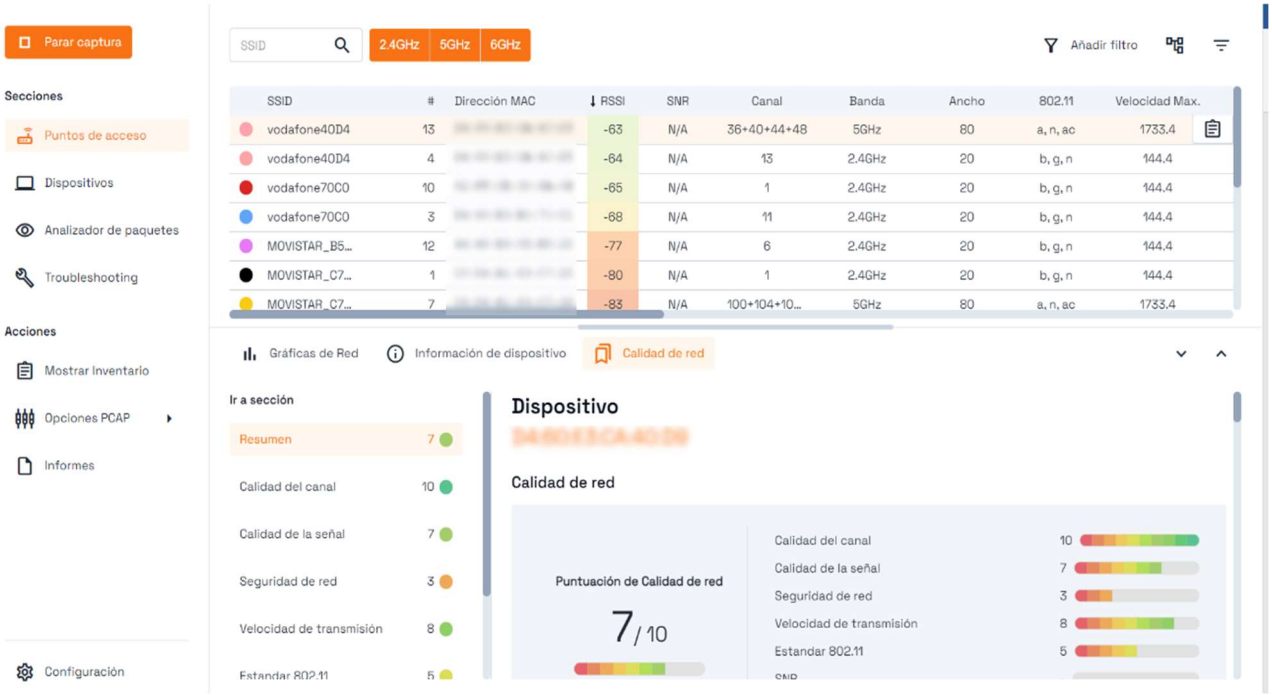


Figura 4-3 Entorno de análisis de redes de Acrylic

Mediante el análisis de los canales y la detección de interferencias, esta herramienta permite identificar las causas de problemas como baja señal, desconexiones frecuentes u otros inconvenientes relacionados con la conectividad inalámbrica. Esto facilita la toma de acciones correctivas y la optimización de la configuración de la red, mejorando así la velocidad y estabilidad de la misma.

La función principal de un analizador de red es examinar la conexión, recopilar datos y detectar los problemas que causan una señal de WiFi deficiente. El escáner WiFi de Acrylic utiliza un algoritmo avanzado para escanear y analizar el espectro de frecuencias de las redes inalámbricas cercanas. Aunque este algoritmo no es accesible a nivel de usuario, nos da una idea de qué parámetros son los principales a la hora de analizar o puntuar en este caso una conexión.

Esto proporciona una validación en la búsqueda de qué propiedades se podrían mejorar para diseñar una propuesta de algoritmo. Acrylic WiFi Analyzer es capaz de identificar los puntos débiles de una red y proporcionar información valiosa para esta optimización.

4.5 Mosquitto y MQTTX para la transferencia eficiente de datos

Mosquitto es un conocido bróker de mensajes MQTT de código abierto. Como se vio en el capítulo anterior, MQTT es un protocolo de mensajería ligero diseñado para la comunicación entre dispositivos de IoT que requieren una transmisión eficiente y confiable de datos. Mosquitto proporciona un entorno de servidor MQTT de alto rendimiento y escalable, permitiendo la conexión y comunicación fluida entre dispositivos IoT y aplicaciones.

Dentro de sus funcionalidades clave encontramos:

Bróker MQTT: Mosquitto actúa como intermediario entre los dispositivos conectados, permitiendo la publicación y suscripción de mensajes MQTT.

Protocolo MQTT: Implementa el protocolo MQTT para la transmisión eficiente y confiable de mensajes entre dispositivos.

Soporte de QoS: Ofrece diferentes niveles de calidad de servicio para garantizar la entrega de mensajes en entornos de red con conectividad intermitente o limitada.

Autenticación y autorización: Permite configurar políticas de seguridad para autenticar y autorizar a los clientes y controlar el acceso a los mensajes.

Integración con TLS/SSL: Proporciona soporte para cifrado de extremo a extremo mediante el uso de protocolos de seguridad como TLS/SSL.

Retención de mensajes: Admite la retención de mensajes para que los nuevos suscriptores puedan recibir mensajes previamente publicados.

Escalabilidad: Diseñado para ser escalable y capaz de manejar grandes volúmenes de mensajes y conexiones simultáneas.

En la parte de implementación, encontramos útil el uso de Mosquitto para establecer la comunicación y transferencia de datos de forma eficiente y confiable en entornos de WiFi Direct. Con Mosquitto, podemos crear aplicaciones que aprovechen el protocolo MQTT para intercambiar mensajes y datos entre dispositivos conectados directamente a través de WiFi Direct. Esto nos permite construir soluciones de IoT como aplicaciones colaborativas que se beneficien de la comunicación directa y de bajo consumo de energía proporcionada por WiFi Direct.

Por otro lado, MQTTX es una herramienta de código abierto y multiplataforma que proporciona una interfaz gráfica (véase la Fig. 4-4) para interactuar con bróker MQTT. Esta aplicación es ampliamente utilizada por desarrolladores para simplificar el proceso de prueba y depuración de sus aplicaciones basadas en MQTT.

Sus principales características son:

Conexión a brókeres MQTT: Los usuarios pueden configurar los detalles de conexión de forma sencilla, como la dirección del bróker, el puerto, las credenciales de autenticación y el protocolo de seguridad.

Suscripción y publicación de temas: Los usuarios pueden suscribirse a uno o varios temas MQTT y recibir mensajes en tiempo real. Además, se puede publicar mensajes en los temas correspondientes para simular la interacción entre dispositivos MQTT.

Exploración de mensajes: La herramienta proporciona una interfaz intuitiva para explorar los mensajes recibidos. Los usuarios pueden ver el contenido completo de los mensajes lo que facilita la comprensión y análisis de los datos intercambiados a través de MQTT.

En el contexto de este Trabajo, se utilizará como una herramienta para simular y plantear un primer boceto del algoritmo. Con MQTTX, se podrán establecer conexiones con el bróker Mosquitto y simular la comunicación entre los dispositivos que implementen dicho algoritmo ya que esta herramienta permitirá la suscripción a los temas relevantes y recepción en tiempo real de los mensajes enviados por los nodos simulados.

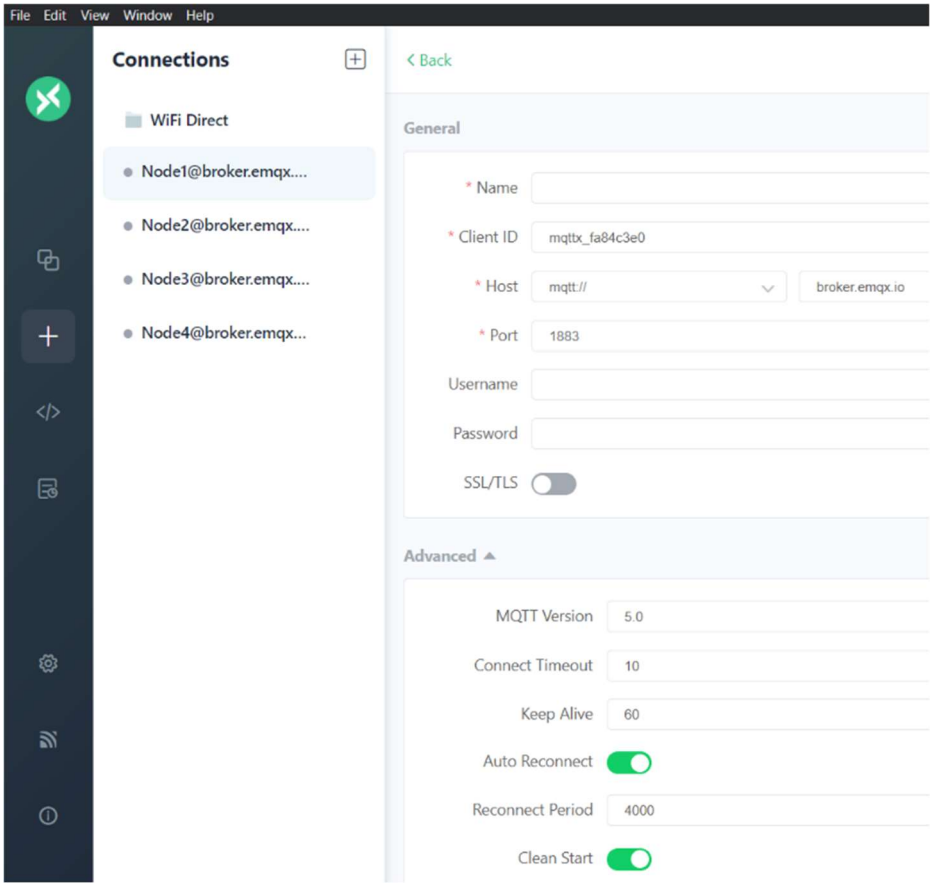


Figura 4-4 Entorno de trabajo en MQTTX

La combinación de Mosquitto como bróker MQTT y MQTTX como herramienta de simulación proporcionará una solución completa y eficiente para probar y validar el algoritmo propuesto. Esto permitirá obtener resultados significativos y tomar decisiones informadas sobre posibles mejoras y optimizaciones en el diseño del algoritmo, en aras de mejorar el rendimiento y la eficiencia de la red inalámbrica en estudio.

5 IMPLEMENTACIÓN EN DIFERENTES ENTORNOS DE SIMULACIÓN

La simplicidad es la máxima sofisticación.
-Leonardo da Vinci-

En este capítulo, se profundiza en las dos hipótesis clave de la investigación. La primera se centra en mejorar el rendimiento de un algoritmo de consenso; específicamente, se evalúa su desempeño en relación con el número de nodos en la red. Se utilizan pruebas y resultados de simulaciones mediante la herramienta WiDiSi, aplicados al algoritmo de consenso RedMesh. A partir de estos resultados, se analizan diversas características que podrían potenciar la ejecución del algoritmo.

Por otro lado, la segunda hipótesis sostiene que el algoritmo de consenso más adecuado para la tecnología WiFi Direct es Raft. Estudios recientes respaldan esta afirmación, demostrando que Raft supera a otros algoritmos en términos de rendimiento. Por ejemplo en un estudio en el que se probó Raft en la base de datos Cassandra [56] utilizando las herramientas Opscenter y Stress para monitorear e imponer una alta carga al sistema, respectivamente. Los resultados mostraron un equilibrio de carga aceptable y, en consecuencia, una mejora en la eficiencia del sistema mediante la aplicación del algoritmo Raft.

Esta parte de la implementación se llevará a cabo mediante un repositorio de GitHub [114] del algoritmo de consenso Raft para ver cómo se puede llegar a combinar con la tecnología WiFi Direct.

5.1 Prueba en entorno de simulación WiDiSi

En esta primera parte de la implementación se ejecutarán pruebas de un algoritmo que utiliza la tecnología WiFi Direct como es RedMesh con el simulador de alto nivel WiDiSi, ya que este simulador es muy útil a la hora de evaluar un nuevo protocolo en un entorno real, especialmente en sus primeras etapas de desarrollo. Como ya se ha expuesto, entre otras ventajas, consigue tener en cuenta una escalabilidad extrema del sistema al igual que su dinamismo.

Para poder evaluarlo se podrán crear y modificar parámetros de red a través de un archivo de texto de configuración antes de que comience su simulación.

En primer lugar, es necesario verificar el comportamiento de WiDiSi usando una serie de reglas y monitores. Para un correcto funcionamiento se debe comportar como un dispositivo WiFi Direct real en una red WiFi Direct. Para ello, se definen las siguientes reglas, las cuales están divididas en dos categorías: reglas generales impuestas por la radiopropagación en el espacio y reglas de WiFi Direct definidas por la especificación P2P de WiFi. A continuación, se señalan las más importantes:

Reglas generales:

- Un par no puede conectarse a otro par fuera de su rango de proximidad.
- Un grupo no puede constar de más de un número predefinido de pares.
- Un par no puede descubrir más de un número predefinido de dispositivos y servicios.
- Un par no puede descubrir dispositivos y servicios fuera de su rango de radio.

Reglas de WiFi Direct:

- Un cliente no puede conectarse a otro cliente directamente.
- Un par no puede descubrir otros pares o servicios si no ha iniciado el descubrimiento.
- La formación de un grupo no puede tardar más de 15 segundos.
- Un par no puede tener acceso a los datos del grupo si abandonó el grupo o si el grupo ya no está disponible.
- El propietario del grupo siempre es detectable.
- Cuando dos dispositivos P2P negocian para decidir el rol GO, el dispositivo con mayor intención debe convertirse en el propietario del grupo.

- Si el propietario del grupo falla, el grupo debe cancelarse.
- Los roles de GO no se pueden transferir dentro de un grupo antes de cancelar el grupo.
- Un cliente no puede comunicarse con otro cliente en el mismo grupo directamente (el mensaje debe pasarse a través del propietario del grupo; se aplicará la demora del canal)
- El descubrimiento permanece activo hasta que se inicia una conexión o se forma un grupo P2P (para clientes)

Por otro lado, la API que se utiliza de WiFi Direct en WiDiSi es muy similar a la WiFi P2P en Android. Sin embargo, hay algunas diferencias. La limitación más importante es que WiDiSi es un simulador de un solo hilo; esto significa que todos sus componentes internos se sincronizan alrededor de los ciclos de Peersim. Las aplicaciones de Android, en cambio, pueden ser multihilo. Para resolver esta discrepancia, se decidió utilizar PeerSim en un modo híbrido, es decir, en el que se utilizan tanto sus motores impulsados por ciclos como impulsados por eventos.

Por otra parte, WiDiSi soporta importantes funcionalidades de WiFi Direct, tales como la formación de grupos estándar y autónomos, el descubrimiento de dispositivos y de servicios, la terminación de grupos y la entrega de sockets simples.

A continuación se muestra un extracto del código [115] donde se ejecutan las modificaciones.

```
############ Simulation parameters ########################
random.seed             1234567890
network.size            200

###Simulation.endtime could be the number of cycles for controls
simulation.endtime      620
simulation.logtime      100
simulation.experiments 1

#Network Dynamic
MIN_NET_SIZE                    200
MAX_NET_SIZE            200
NET_ADD_STEP            1
SIN_PERIOD              1000
NET_ADD_CYCLE_START     1
NET_ADD_CYCLE_END       simulation.endtime

############More code###################################

############ Test application ##############################
protocol.test1 newApplication
{
        p2pmanager              p2pmanager
        wifimanager             wifimanager
        p2pinfo                 nodinf
        step                    1
}
```

El escenario que se plantea para esta prueba es un entorno de gran escala (máximo de 200 nodos). La aplicación busca continuamente compañeros e intenta conectarse a los que se encuentran dentro del rango. Cuando dos o más nodos estén conectados entre sí, formarán un grupo e intercambiarán sus valores.

Para mantener los parámetros constantes durante toda la etapa de testeo, se utiliza el mismo número de semilla aleatoria en cada simulación. Se considera el algoritmo predefinido, al igual que se mantiene constante el tiempo de simulación, que se fija en 5 minutos. En cuanto al rango de proximidad, se establece en un radio de 200 metros y lo que se ajusta entonces es el tamaño de la red.

En la Figura 5-1 se muestra un ejemplo de una simulación realizada.

Se puede observar cómo los nodos son representados como vértices y las aristas que los unen se utilizan para mostrar las conexiones entre ellos. En verde se representan los GO y su tamaño variará respecto al número de clientes a los que dan servicio, que se representan en color azul. Además de visualizar el gráfico, también se muestran otros datos útiles como el tiempo, ya sea real o de simulación, el número de nodos dentro de la red, el número de nodos conectados o incluso el porcentaje de conectividad.

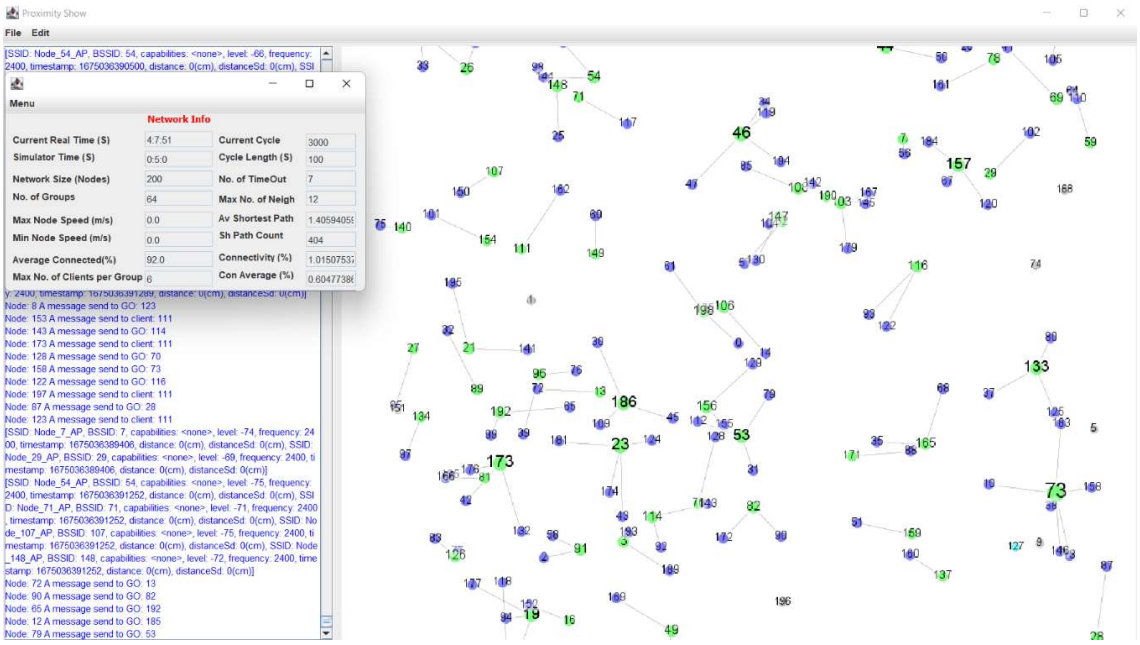


Figura 5-1 Ejecución del simulador para un tamaño de red de 200 nodos

Esta herramienta es de bastante utilidad si se quieren extraer algunas conclusiones a base de simular varias veces la herramienta y comprobar la primera de las hipótesis, por lo que ha considerado un total de cuatro simulaciones variando el número de nodos. A continuación, se presentan y discuten las siguientes gráficas como resultados de dichas simulaciones:

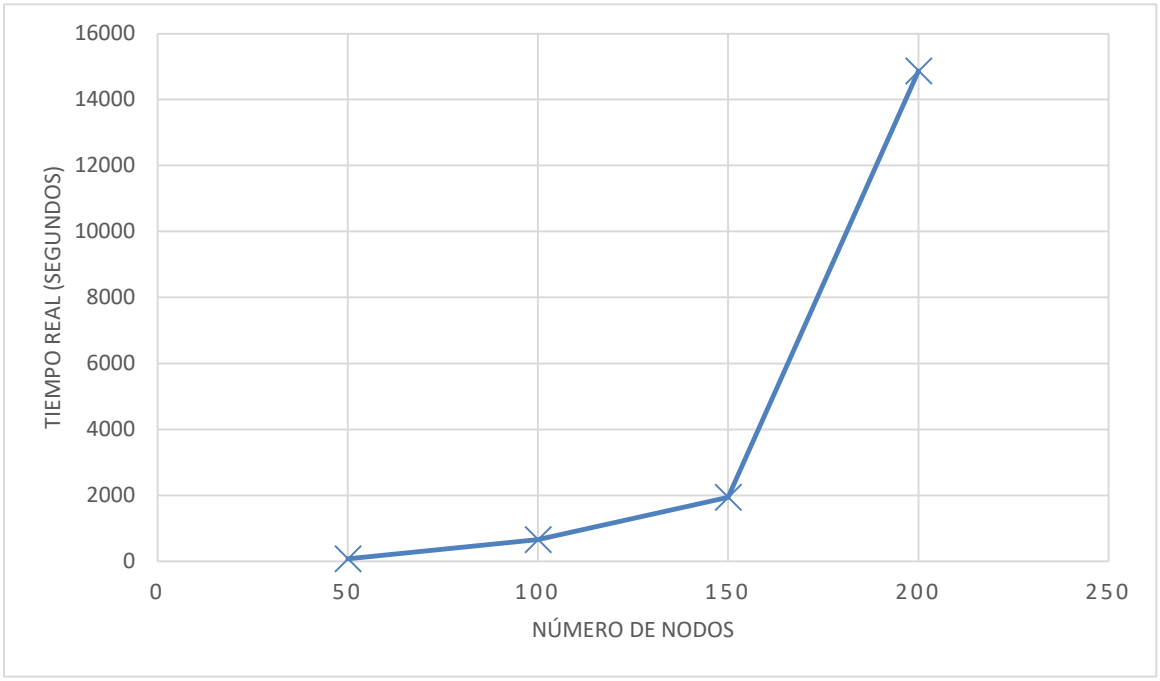


Figura 5-2 Tiempo real de ejecución dependiendo del número de nodos

A partir de los resultados obtenidos, se puede observar como en la Figura 5-2 se aprecia un aumento en el tiempo de ejecución del algoritmo de consenso a medida que aumenta el número de nodos en la red. En particular, se puede notar que el tiempo de ejecución se incrementa de manera significativa a medida que se pasa de 50 a 100 nodos, lo que indica que la escalabilidad del algoritmo puede ser un problema en redes de gran tamaño.

Sin embargo, también se observa que el aumento del tiempo de ejecución no es lineal, sino que se incrementa de manera exponencial a medida que aumenta el tamaño de la red. Por ejemplo, el tiempo de ejecución para 150 nodos es casi tres veces mayor que para 100 nodos, mientras que el tiempo de ejecución para 200 nodos es aproximadamente 22 veces mayor que para 100 nodos.

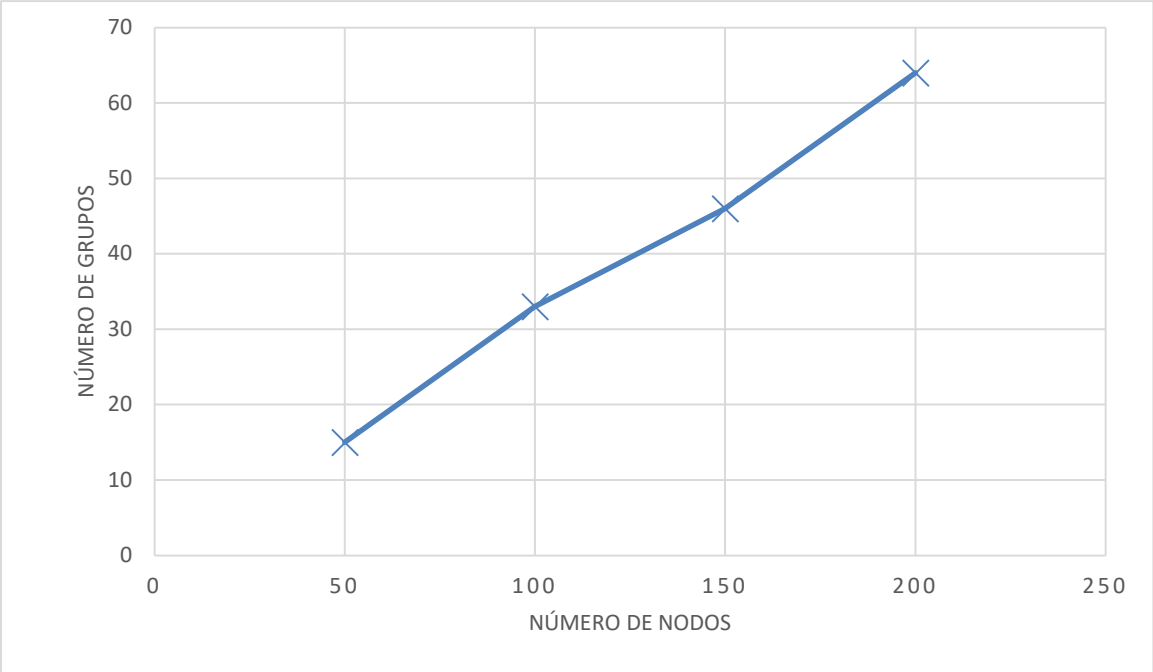


Figura 5-3 Número de grupos formados dependiendo del número de nodos

Por otro lado, en la Figura 5-3 se observa como el número de grupos formados aumenta a medida que aumenta el número de nodos en la red. Esto se debe a que hay más nodos disponibles para conectarse y formar grupos. Mientras que en la Figura 5-4 se muestra un aumento en el porcentaje de nodos conectados a medida que aumenta el número de nodos en la red. Es decir, a medida que la red crece en tamaño, más nodos están conectados entre sí.

En general, estos resultados sugieren que el rendimiento del algoritmo de consenso depende en gran medida del número de nodos que conforman la red, y que un aumento en el número de nodos puede tener un impacto significativo en el tiempo de ejecución y la efectividad del algoritmo. También sugiere que el número de grupos formados puede ser una métrica útil para medir la efectividad de la red en términos de conectividad y colaboración ya que se ve como con un mayor número de nodos, se logra una mayor conectividad en la red.

Por lo tanto, se puede concluir que el tamaño de la red tiene un impacto significativo en el tiempo de ejecución del algoritmo de consenso, y que este factor debe ser considerado cuidadosamente al diseñar y optimizar el algoritmo lo que valida la primera hipótesis de este Trabajo y dará paso a un análisis posterior sobre qué más características se deben centrar en mejorar para incrementar la eficiencia del algoritmo de consenso.

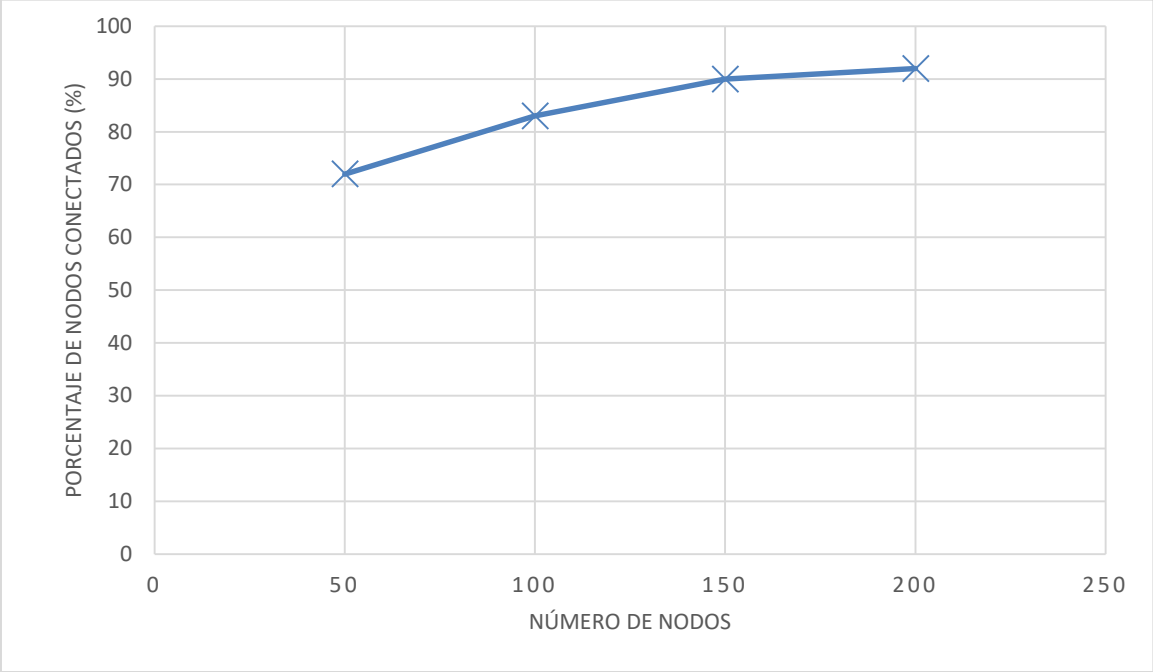


Figura 5-4 Porcentaje de nodos conectados dependiendo del número de nodos

5.2 Implementación del algoritmo de consenso Raft en Java

Para comenzar a diseñar un algoritmo, es necesario tener claro cómo se va a evaluar y en función de qué criterios. La respuesta más coherente es en función de si realmente consigue un objetivo autoimpuesto previamente, que suele ser que cumple su funcionalidad con eficiencia y exactitud.

Aunque puede no ser obvio a priori, es importante destacar que un aspecto fundamental para evaluar un algoritmo es su simplicidad y facilidad de comprensión. Los algoritmos deben ser fáciles de entender, ya que en general, es difícil obtener beneficios de un algoritmo a menos que se pueda implementar correctamente.

En los últimos 30 años, el algoritmo Paxos ha sido el estándar por excelencia en el consenso, y prácticamente todas las implementaciones de consenso desde entonces se han basado en él. El problema es que Paxos es bastante difícil de entender. Fue así como surgió la necesidad de buscar una alternativa de algoritmo de consenso simple, y se implementó Raft.

Es por esto que en esta parte del Trabajo se ha decidido usar un repositorio de GitHub [114] que implementa el algoritmo Raft en lenguaje Java. A través del análisis del código, se intentará buscar una mejora aplicable a la tecnología WiFi Direct.

5.2.1 Justificación de Java como lenguaje de programación elegido

En la implementación del algoritmo de consenso Raft, se seleccionó Java como el lenguaje de programación principal por varias razones:

1. **Amplia adopción y disponibilidad:** Java es uno de los lenguajes de programación más populares y ampliamente utilizados en el desarrollo de software. Cuenta con una gran base de usuarios y una amplia comunidad de desarrolladores, lo que facilita el acceso a recursos, bibliotecas y documentación.

2. **Orientación a objetos:** Java es un lenguaje orientado a objetos, lo que facilita la creación de una estructura modular y jerárquica para la implementación del algoritmo de consenso Raft.

3. **Portabilidad:** Java es conocido por su portabilidad, lo que significa que el código Java puede ejecutarse en diferentes sistemas operativos y arquitecturas sin necesidad de modificaciones significativas. Esto es especialmente importante para aplicaciones distribuidas, donde los nodos pueden tener diferentes configuraciones y entornos.

4. **Amplio conjunto de bibliotecas y herramientas:** La comunidad de desarrollo de Java ha creado una amplia gama de bibliotecas y herramientas que facilitan la implementación de sistemas distribuidos y algoritmos de consenso. Éstas proporcionan soluciones existentes para tareas comunes, como la comunicación en red, la persistencia de datos y la concurrencia, lo que acelera el desarrollo y reduce la complejidad.

5. **Madurez y confiabilidad:** Java es un lenguaje maduro y ha sido utilizado en numerosos proyectos empresariales a lo largo de los años. Su robustez y confiabilidad son bien conocidas, lo que lo convierte en una elección sólida para implementaciones críticas y sistemas distribuidos que requieren una alta disponibilidad y resistencia a fallos.

5.2.2 Diseño de la implementación en Java

Este código tiene dos secciones fundamentales como son la parte del cliente y la del servidor.

En primer lugar, se analizará la clase Client que representa un nodo cliente en el algoritmo Raft y es responsable de la comunicación con otros nodos y del mantenimiento del estado relacionado con el proceso de consenso. En esta clase, se utilizan paquetes de comunicación unicast y multicast para facilitar la comunicación entre los nodos.

La clase Client contiene una serie de variables importantes, como “clientId” y “receivedValue”. Estas variables almacenan información relevante para el proceso de consenso y son utilizadas para tomar decisiones en el algoritmo Raft.

Además, esta clase gestiona una lista de comandos, representada por la variable “command”. Esta lista almacena los comandos enviados por otros nodos y permite al cliente mantener un registro de las operaciones realizadas durante el proceso de consenso.

Para manejar los mensajes de comunicación entre los nodos, la implementación utiliza objetos de la clase “Message”. Estos objetos contienen la información necesaria para la comunicación, como el contenido del mensaje y los identificadores de los nodos involucrados.

La gestión del estado y la persistencia de los registros se lleva a cabo utilizando variables y métodos en la clase Client. Por ejemplo, la variable "localValue" representa el valor local del nodo cliente, que se actualiza en función de los comandos recibidos. La persistencia de los registros se logra mediante el uso de variables y estructuras de datos adecuadas para mantener el estado del cliente y garantizar la coherencia entre los nodos.

Por otro lado, tenemos la clase Server que representa el nodo servidor en el sistema distribuido. Contiene variables para almacenar información sobre el nodo, como el ID, el término actual, el rol actual (líder, candidato o seguidor) o el valor de la máquina de estado. También cuenta con una lista de registros que se utiliza para almacenar las operaciones realizadas en el sistema.

La clase Server tiene varios métodos para configurar y obtener las variables de estado, así como para enviar y recibir mensajes, pero principalmente contiene la lógica central del algoritmo Raft.

Un método fundamental es run() de ServerThread, una clase que extiende la clase Thread. Se inicializa una interfaz gráfica de usuario para el servidor y espera a que otros servidores se unan al clúster enviando un mensaje de unidifusión con su propio número de puerto a un grupo de multidifusión y escuchando las respuestas. Después de unirse al clúster, el servidor entra en un bucle y comienza a comportarse de acuerdo con su rol actual, que puede ser seguidor, candidato o líder. El bucle se ejecuta indefinidamente hasta que se detiene el programa.

Se puede observar que, dependiendo del rol del nodo, se ejecutan diferentes conjuntos de funciones.

- Si el nodo del servidor es un candidato, envía mensajes RequestVote a otros nodos y espera sus respuestas. Si obtiene la mayoría de los votos, se convierte en el líder. Si no recibe suficientes votos, vuelve al rol de seguidor.

- Si el nodo del servidor es un líder, envía mensajes AppendEntry a otros nodos y espera sus respuestas. También maneja las solicitudes de los clientes agregándolas a su registro y enviando mensajes a otros nodos para replicar las entradas. Como líder, el servidor envía periódicamente mensajes de latidos a todos los demás servidores para informarles que todavía ejerce su función.

- Si el nodo del servidor es un seguidor, espera mensajes del líder o candidato. Si recibe un mensaje AppendEntry, procesa el mensaje y envía un reconocimiento al líder. Si recibe un mensaje RequestVote, procesa el mensaje y envía una respuesta de voto al candidato. Si no recibe ningún mensaje dentro del período de tiempo de espera de la elección, cambia a un rol de candidato y comienza una nueva elección.

El código también escucha los mensajes de los clientes y otros servidores y responde en consecuencia. Por ejemplo, cuando el servidor recibe un mensaje de entrada de anexo de un líder, verifica si el mensaje es válido y actualiza su registro en consecuencia. Si el servidor recibe un mensaje de solicitud de voto de un candidato, verifica si el mandato del candidato es mayor que el suyo y vota por el candidato si lo es. Si el servidor recibe un mensaje de respuesta de solicitud de voto de otro servidor, comprueba si se concede la respuesta y cuenta el número de respuestas concedidas que recibe.

En general, este código implementa la lógica central del algoritmo de consenso de Raft y proporciona una base para construir un sistema distribuido que use el algoritmo. Sin embargo, al código le faltan algunos detalles importantes como el manejo de errores, y no está claro cómo se comportaría en varios escenarios de fallo.

5.2.3 Implementación práctica

A continuación, ejecutaremos una demostración gráfica seleccionando el número de servidores que se quieran implementar. El algoritmo Raft es un protocolo de consenso distribuido diseñado para mantener la coherencia y disponibilidad en un sistema distribuido compuesto por múltiples nodos. En este caso el código está programado para trabajar con un solo cliente y cinco servidores.

En la Fig. 5-5 se puede observar como se abre la Interfaz Gráfica de Usuario (GUI) mostrándonos por pantalla tanto nuestro cliente como los cinco servidores creados.

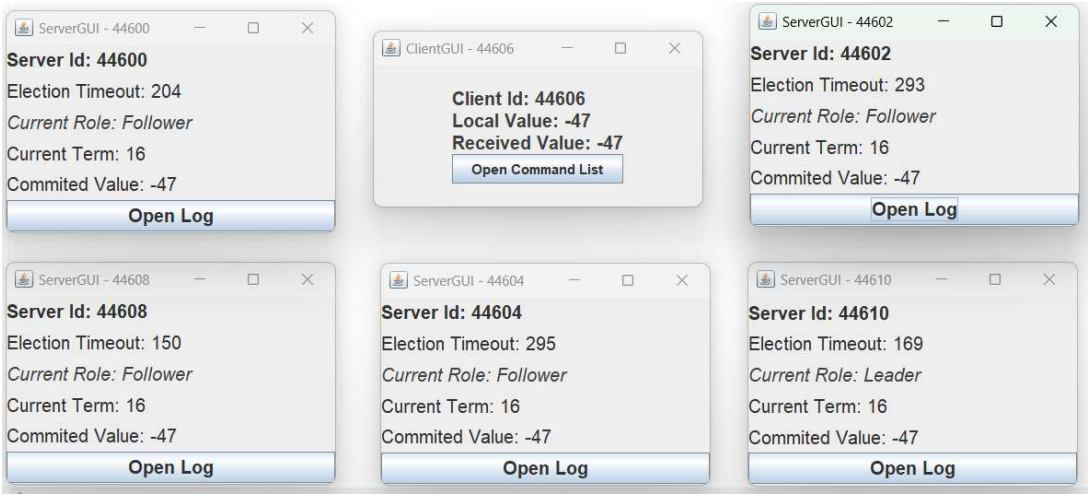


Figura 5-5 GUI con Cliente y Servidores

En primer lugar, analizaremos a nuestro cliente, el cual, nos proporciona datos internos de su registro como su valor ID asociado, el valor que recibe por parte del servidor encargado de transmitir el mensaje y el valor que se guarda en el registro local.

Si analizamos ahora a los servidores vemos como también nos muestran sus datos en el registro. En primer lugar, tal y como vimos en el cliente, también tiene asociado un valor ID único que le servirá como identificación. El concepto de continuidad viene definido en Raft, que divide el tiempo en términos que se muestra en los servidores. Vemos como tienen un término común y va avanzando según se ejecuta el programa.

A lo largo de esta ejecución se plantean varios casos que es interesante analizar:

1. Elección del líder

Inicialmente, todos los nodos del sistema son seguidores. Cuando un nodo se inicia, comienza en este estado y espera recibir mensajes del líder. Si un seguidor no recibe mensajes del líder durante un tiempo de espera (timeout), comienza una elección para convertirse en líder y se convierte en candidato.

Durante una elección, un seguidor envía una solicitud de voto a todos los demás nodos (véase la Fig. 5-6). Si un nodo no ha votado en el término actual y recibe una solicitud de voto, vota por el nodo solicitante. Un nodo se convierte en líder si obtiene votos de la mayoría de los nodos.

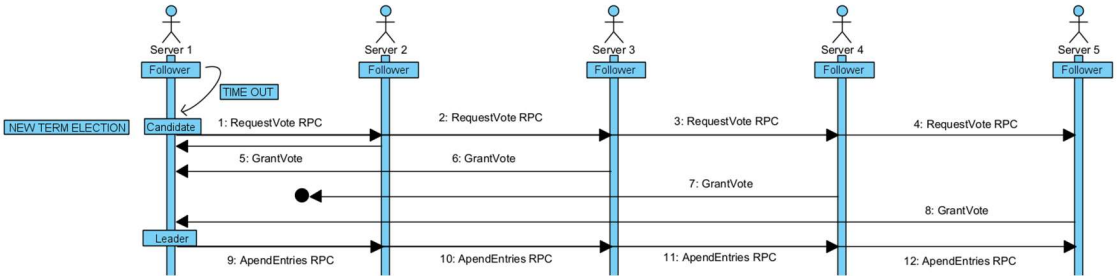


Figura 5-6 Proceso de elección del líder

2. Replicación y compromiso de registros:

Una vez que se ha elegido un líder, comienza la replicación de registros en los seguidores (véase la Fig. 5-7). Los clientes envían solicitudes de escritura al líder. El líder agrega las solicitudes a su registro y las envía a los seguidores para su replicación. Los seguidores confirman la replicación de los registros y envían una confirmación al líder. El líder espera recibir confirmaciones de la mayoría de los seguidores antes de considerar que un registro está confirmado.

Una vez que un registro ha sido replicado en la mayoría de los nodos, incluido el líder, se considera
"comprometido". El líder notifica a los seguidores sobre el compromiso del registro y les indica que lo apliquen a sus estados. Los seguidores aplican el registro a su estado y responden con una confirmación de aplicado. Una vez que el líder recibe confirmaciones de la mayoría de los seguidores, considera que el registro está aplicado

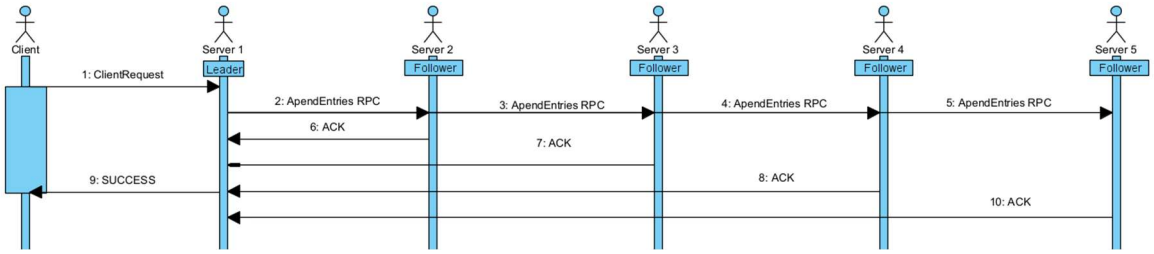


Figura 5-7 Proceso de replicación y compromiso de registros

3. Reelección:

La reelección ocurre cuando el líder actual se vuelve inaccesible o falla. En este caso, los seguidores detectan la falta de actividad del líder y comienzan un nuevo proceso de elección para elegir un nuevo líder. Para eso cada seguidor tiene un mecanismo aleatorio de timeout para detectar la falta de mensajes del líder (véase la Fig. 5-8). Si un seguidor no recibe mensajes del líder dentro de un tiempo de espera determinado, asume que el líder ha fallado o está inaccesible.

El seguidor que inicia la elección se convierte en candidato y envía una solicitud de voto a los demás nodos.

Convertido ahora en candidato envía una solicitud de voto a todos los demás nodos del sistema. La solicitud de voto contiene información sobre el término actual y el ID del candidato. Cada nodo vota por el candidato si aún no ha votado en el término actual. Al votar por un candidato, un nodo también actualiza su término actual al del candidato.

El candidato espera recibir votos de la mayoría de los nodos para convertirse en líder. Si un candidato recibe votos de la mayoría, se convierte en líder para el término actual. La mayoría se determina en función del número total de nodos en el sistema.

Una particularidad durante la reelección, se obtiene cuando múltiples seguidores detectan la falta de actividad del líder y comienzan a ejecutarse elecciones simultáneas. Aquí destaca el uso de la aleatoriedad en los tiempos de espera, ya que cada candidato, aparte de incluir su ID en las solicitudes de voto, espera un tiempo aleatorio antes de iniciar la elección. El tiempo aleatorio ayuda a reducir la probabilidad de conflictos entre elecciones simultáneas. Un nodo solo vota por un candidato si no ha votado previamente en el término actual y el ID del candidato es el más reciente que ha visto. Una vez que se elige un nuevo líder, el proceso de replicación de registros y compromiso continúa desde ese punto.

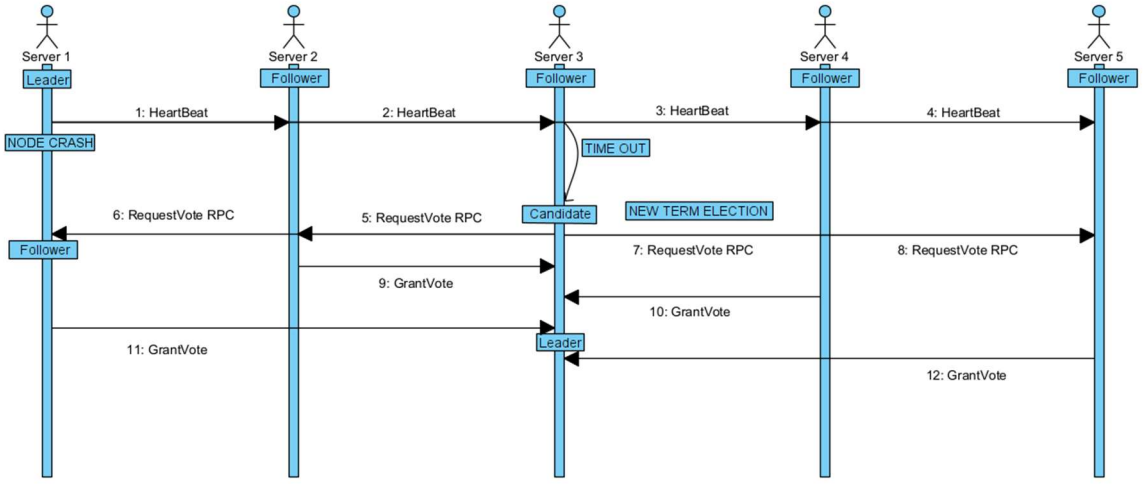


Figura 5-8 Proceso de Reelección

Existen diferentes enfoques y opiniones sobre el uso de la aleatoriedad en algoritmos de consenso como Raft. Algunos consideran que la aleatoriedad puede proporcionar beneficios en términos de equidad y prevención de bloqueos en situaciones específicas. Sin embargo, también es válido cuestionar la dependencia de la aleatoriedad y explorar alternativas más deterministas para mejorar el rendimiento del algoritmo.

Habiendo analizado las diversas perspectivas sobre el uso de la aleatoriedad en algoritmos de consenso, resulta evidente la importancia de buscar constantemente mejoras y alternativas para optimizar el rendimiento de estos algoritmos. Es en este contexto que surge la necesidad de proponer un nuevo enfoque para el algoritmo de consenso, uno que se base en criterios más deterministas y predictivos, en lugar de depender únicamente de la aleatoriedad. Al explorar criterios basados en el rendimiento del sistema, la disponibilidad de recursos y otras características específicas de los nodos, podemos construir un algoritmo más sólido y eficiente, capaz de brindar estabilidad, confiabilidad y un consenso rápido y consistente. En el siguiente capítulo, se presentará en detalle esta propuesta, examinando cómo estos nuevos criterios pueden contribuir a mejorar el algoritmo de consenso y abordando posibles desafíos y beneficios asociados a este enfoque innovador.

6 PROPUESTA DE ALGORITMO DE CONSENSO

Apunta a las estrellas y llegarás a la luna
-Leopoldo Fernández Pujals-

Mientras que la mayoría de los estudios de algoritmos de consenso se basan en la aleatoriedad para la selección del líder, algunos autores proponen criterios más útiles para conseguir una mejora de calidad en torno a la elección. Esta observación ha generado un creciente interés en desarrollar algoritmos de consenso más sofisticados y efectivos que permitan una toma de decisiones más precisa y confiable en entornos distribuidos.

En este capítulo se presentan las consideraciones a tener en cuenta para la definición de un nuevo algoritmo de consenso que mejore sus prestaciones, detallando métricas y criterios para evaluar el rendimiento y la eficiencia del algoritmo en diferentes contextos y configuraciones de red.

Posteriormente, se introduce una propuesta formal del algoritmo y se definen algunos escenarios de aplicación. En estos escenarios se muestran los resultados de aplicar el algoritmo, todo ello en el contexto de WiFi Direct. Se resaltan las características distintivas y se evalúa su rendimiento en comparación con otros enfoques existentes. Finalmente, se describe la implementación, primero utilizando Java siguiendo el enfoque de Raft, y luego se aborda la implementación de las comunicaciones con el protocolo MQTT.

La propuesta que se presenta tratará de cumplir uno de los objetivos principales de este Trabajo, que era obtener un algoritmo de consenso anticipado y recurrente para la elección del mejor punto de acceso diseñado para superar las limitaciones de los enfoques aleatorios y proporcionar criterios más sólidos para la elección del líder.

El algoritmo de consenso propuesto presenta una estructura bien definida que permite calcular una puntuación para cada nodo en función de sus características y prioridades. La utilización de sesgos para cada criterio brinda flexibilidad al sistema, permitiendo ajustar la importancia relativa de cada característica según las necesidades específicas del entorno.

La fórmula para calcular dicha puntuación se basa en el denominado “Valor de Importancia” de cada nodo que será una suma ponderada de dichos criterios, lo que permite evaluar de manera adecuada la influencia de cada característica en la elección del próximo nodo GO. Además, la inclusión de la prioridad del nodo en dicho cálculo permite forzar la selección de un nodo específico como GO, si así se requiere.

Por último, la normalización de las puntuaciones asegura que todas estén en un rango comparativo y sean evaluadas de manera justa, facilitando la elección del nodo con la puntuación más alta.

En cuanto a la adaptabilidad, cabe destacar que el algoritmo también se plantea para distintos escenarios mediante la definición de perfiles con diferentes sesgos. Estos perfiles representan casos de uso específicos que permiten evaluar y ajustar los pesos y el rendimiento del algoritmo en diferentes contextos que serán desgranados más adelante con algunos ejemplos, asegurando una mayor eficiencia y optimización del sistema.

La estructura del algoritmo propuesto sigue una línea similar a la del algoritmo Raft en términos de elección del líder (o GO en este caso) a través de un proceso de consenso.

Una de las principales adiciones del algoritmo propuesto es la consideración de características específicas para la transferencia de datos. Por ejemplo, el algoritmo toma en cuenta el número de nodos en la zona de cobertura, la intensidad de señal recibida en el nodo, el nivel de batería y el ancho de banda. Estos criterios son especialmente relevantes en un entorno MQTT, donde los nodos necesitan comunicarse entre sí y compartir datos de manera eficiente.

El algoritmo se adapta bien al contexto de WiFi Direct. Esto implica tener en cuenta la capacidad de cada nodo para servir como GO, ya que una vez elegido, el GO actuará como Soft-AP, gestionando la comunicación entre todos los dispositivos cliente P2P asociados. La elección del GO en WiFi Direct es esencial para garantizar una comunicación estable y eficiente entre los dispositivos, y el algoritmo propuesto busca optimizar esta selección.

Para llegar hasta este punto, se ha tenido que investigar los algoritmos de consenso establecidos en capítulos anteriores, ya que tienen ciertas limitaciones cuando se aplican a la tecnología WiFi Direct debido a las características específicas de esta tecnología de comunicación inalámbrica.

A continuación, se resumen algunas de las limitaciones más importantes a tener en cuenta en el diseño de un algoritmo de consenso para WiFi Direct:

Escalabilidad: Cuando se trata de un gran número de dispositivos conectados a través de WiFi Direct la gestión y coordinación de un gran grupo de dispositivos pueden generar una sobrecarga significativa en el proceso de consenso.

Latencia: En WiFi Direct, la latencia puede ser una preocupación, especialmente en entornos con una alta densidad de dispositivos o en redes inestables.

Ancho de banda limitado: Los algoritmos de consenso que requieren una gran cantidad de datos para ser transferidos entre los nodos pueden implicar problemas de congestión y baja velocidad de transmisión.

Consumo de energía: Los algoritmos de consenso pueden requerir una cantidad significativa de recursos computacionales y de energía para su ejecución. En dispositivos móviles o con batería limitada, esto puede ser un factor crítico que restrinja la viabilidad de ciertos algoritmos.

Disponibilidad de canales de comunicación: En WiFi Direct, la comunicación directa entre dispositivos se realiza en canales específicos. La disponibilidad y la interferencia en estos canales pueden afectar la comunicación y, por lo tanto, el proceso de consenso.

Tolerancia a fallos: En un entorno WiFi Direct, los dispositivos pueden ser más propensos a desconectarse o cambiar de estado debido a la movilidad y otras condiciones cambiantes. Los algoritmos de consenso deben ser capaces de lidiar con la posible pérdida de conexiones de manera efectiva.

Seguridad: La seguridad en la comunicación es fundamental para cualquier algoritmo de consenso. WiFi Direct puede ser susceptible a ciertos ataques de seguridad, y los algoritmos de consenso deben estar diseñados para abordar estas vulnerabilidades.

Al considerar el uso de algoritmos de consenso en este entorno, es fundamental evaluar cómo cada algoritmo aborda y supera estas limitaciones para garantizar su efectividad y rendimiento adecuado en una red WiFi Direct.

6.1 Propiedades a evaluar por el algoritmo

En anteriores capítulos se ha tratado de poner en contexto todo el marco teórico acerca de los algoritmos de consenso con algunos ejemplos de muy distinta índole y se ha demostrado que para cada algoritmo se prima una característica diferente. Esta amplia visión hace posible ver los distintos enfoques por lo que se han optado en el pasado y deja ver una interesante línea de desarrollo a investigar.

Comprender las diferentes características de los algoritmos de consenso es fundamental para poder diseñar e implementar sistemas efectivos. En este análisis, se examinarán las diversas características que deben tenerse en cuenta al seleccionar un algoritmo, al igual que se establecerán métricas para poder evaluarlas. Al comprender dichas características, se podrá evaluar y comparar mejor las diferentes opciones disponibles y elegir la mejor para un sistema específico. Es importante tener en cuenta que ningún algoritmo es perfecto y es útil sopesar los pros y los contras antes de tomar una decisión. Por este motivo en este apartado se remarcan una serie de características que tratan de destacar su importancia a la hora del análisis de rendimiento para con un algoritmo.

6.1.1 Número de nodos en la zona de cobertura

En un grupo de WiFi Direct, la cobertura de red desempeña un papel crucial en la descentralización y el proceso de consenso. Una zona de cobertura más amplia permite la participación de un mayor número de dispositivos, lo que fomenta la distribución de poder y control entre los nodos.

Una mayor zona de cobertura presenta ventajas en términos de escalabilidad y resiliencia, al permitir la conexión de un mayor número de dispositivos y aumentar la capacidad del sistema para manejar transacciones sin afectar negativamente su rendimiento. Además, al tener más nodos en la red, se mejora la resiliencia, ya que el sistema puede seguir funcionando incluso si algunos dispositivos fallan o se desconectan.

Cuando se planea conectar un gran número de dispositivos en un grupo P2P, es importante seleccionar como GO aquel dispositivo que tenga el mayor número de dispositivos en su rango para conectar como clientes. Sin embargo, también es necesario establecer un límite máximo en el número de nodos, ya que un aumento en la cantidad de nodos implica un mayor consumo de recursos de comunicación y computación para lograr el consenso. Este aumento de recursos puede afectar la eficiencia del algoritmo, especialmente en entornos con limitaciones de recursos.

Para evaluar la importancia del número de nodos en la zona de cobertura, se deben considerar métricas como el grado del nodo, que indica la cantidad de dispositivos vecinos con los que un nodo puede establecer comunicación directa. Esta métrica proporciona información sobre la conectividad del dispositivo y su influencia en el proceso de consenso.

Existen diversas formas de obtener el grado del nodo en un grupo de WiFi Direct. Una opción es utilizar mensajes de descubrimiento para identificar y contar los dispositivos vecinos con los que se puede establecer conexión directa. Estos mensajes pueden intercambiarse entre los nodos para construir una lista de vecinos directos. Otra opción es utilizar comandos o API proporcionados por el sistema operativo o bibliotecas de desarrollo de WiFi Direct para obtener información sobre los dispositivos vecinos y sus capacidades de comunicación.

Estas herramientas suelen ofrecer funciones para descubrir y enumerar los dispositivos cercanos, lo que facilita la obtención del grado del nodo.

El grado del nodo se puede expresar en términos de número absoluto, indicando la cantidad de dispositivos vecinos, o como un porcentaje en relación con el número total de dispositivos en la zona de cobertura. La unidad utilizada para medir el grado del nodo es simplemente el número de dispositivos vecinos.

Es importante tener en cuenta que el grado del nodo puede cambiar dinámicamente a medida que los dispositivos se unen o abandonan la red. Por lo tanto, es recomendable realizar un monitoreo continuo de la topología de la red y actualizar el grado del nodo en consecuencia.

6.1.2 Nivel de intensidad de señal

El nivel de intensidad de la señal emitida en cada nodo es un factor diferencial y de gran importancia en un grupo de WiFi Direct. La intensidad de la señal puede influir en la capacidad de los nodos para comunicarse entre sí y alcanzar un consenso efectivo. Por tanto, es fundamental comprender las métricas, unidades y el rango de valores asociados a esta intensidad de señal.

Una ventaja de tener una intensidad de señal fuerte dentro de un sistema es que aumenta la confiabilidad de la comunicación. Las señales más fuertes tienen menos probabilidades de ser interrumpidas o distorsionadas por interferencias o ruido, lo que facilita la comunicación y coordinación entre los nodos. Esto es especialmente relevante en los algoritmos de consenso, donde todos los nodos deben tener una vista coherente de los datos para llegar a un acuerdo.

Es crucial encontrar un equilibrio en la intensidad de la señal, evitando tanto una señal demasiado fuerte que cause interferencias con otros nodos y sistemas, comprometiendo la confiabilidad de la comunicación y el rendimiento, como una señal demasiado débil que obligue a los nodos a transmitir a niveles de potencia más altos, agotando rápidamente la vida útil de la batería.

Establecer niveles adecuados de intensidad de señal es fundamental para garantizar una comunicación confiable sin interferencias, al tiempo que se maximiza la eficiencia energética del sistema, especialmente en entornos donde el consumo de energía es una preocupación primordial.

En este contexto, para medir la intensidad de la señal se utiliza el Indicador de Fuerza de Señal Recibida (RSSI). Este parámetro indica la calidad de la conexión entre dos dispositivos y es esencial en un grupo de WiFi Direct ya que un RSSI bajo puede afectar negativamente la tasa de transferencia de datos del enlace inalámbrico (véase la Tabla 6-1).

Nivel RSSI	*Min. Pr (dBm)*	*Máx. Pr (dBm)*	*Calidad*
0	-∞	-81	Deficiente
1	-80	-78	Mala
2	-77	-73	Estándar
3	-72	-65	Buena
4	-64	0	Excelente

Tabla 6-1 Mapeo de potencia recibida (Pr) a niveles de RSSI cuantizados

En términos de unidades y rango de valores, el RSSI se mide en decibelios (dBm) y generalmente varía en un rango negativo. Cuanto mayor sea el valor absoluto del RSSI, más fuerte será la señal. Sin embargo, los rangos específicos de valores de RSSI pueden variar según el hardware y el entorno de la red. Es importante consultar las especificaciones del dispositivo y considerar el contexto del entorno de WiFi Direct para evaluar la intensidad de señal óptima.

6.1.3 Tasa de transmisión de datos

La tasa de transmisión de datos, medida en bit/s, es un factor clave que influye en la eficiencia y confiabilidad del sistema de comunicación. En WiFi Direct, el rango de valores típico varía desde unos pocos Mbps hasta varios cientos de Mbit/s. Sin embargo, es importante tener en cuenta que el rendimiento real puede verse afectado por varios factores, como la calidad de la señal, la distancia entre los dispositivos y la interferencia del entorno, siempre dependiendo de los estándares y las capacidades de los dispositivos.

Una mayor tasa de bit proporciona una velocidad de comunicación más rápida, lo que facilita el intercambio de información y el consenso entre los nodos de la red. Esto es especialmente valioso en sistemas distribuidos con restricciones de tiempo, como aplicaciones que requieren toma de decisiones en tiempo real.

Sin embargo, es importante tener en cuenta que el uso de tasas de bit más altas implica un mayor consumo de ancho de banda y recursos de transmisión, lo que puede ser un desafío en entornos P2P con recursos limitados. Además, velocidades de bit más altas aumentan el riesgo de errores y corrupción de datos, lo que puede afectar la precisión y confiabilidad de la comunicación, especialmente en algoritmos de consenso que dependen de una vista coherente de los datos entre los dispositivos.

La latencia, es decir, el tiempo que tarda un mensaje en viajar desde el emisor hasta el receptor, también es un aspecto crítico en sistemas de comunicación en tiempo real. Aunque una alta tasa de bits puede agilizar la transmisión, no garantiza una baja latencia si existen demoras en la propagación del mensaje. En aplicaciones que requieren respuestas rápidas y coordinación precisa entre los nodos, una baja latencia es fundamental para evitar retrasos y asegurar la sincronización efectiva.

En el contexto de algoritmos de consenso, una latencia reducida es especialmente relevante, ya que la rapidez con la que los nodos llegan a un acuerdo sobre una decisión afecta la eficiencia del sistema y su capacidad para mantener la coherencia en la red. Por lo tanto, el algoritmo de selección del GO debe considerar tanto la tasa de bit como la latencia para garantizar una comunicación eficiente y oportuna entre los dispositivos P2P.

6.1.4 Diferentes estándares de comunicación WiFi: IEEE 802.11g/ac/ax

Existen varios estándares diferentes para la comunicación WiFi, entre los que se destacan IEEE 802.11g, 802.11ac y 802.11ax [116]. Estos estándares especifican diferentes características técnicas y capacidades de la comunicación WiFi, como las bandas de frecuencia utilizadas, las velocidades máximas de datos y el nivel de compatibilidad con otros dispositivos.

802.11g: La contribución más importante de esta norma es lograr una mayor velocidad que estándares anteriores capaz de alcanzar velocidades de hasta 54 Mbps en la banda de 2,4 GHz, además de incorporar OFDM, lo que la hace más eficiente que el resto de los estándares 802.11 en esta banda. El principal problema es que este estándar operará en una banda donde existen muchas fuentes de interferencia y cada vez está más poblada.

802.11ac: Este estándar también se conoce como WiFi 5 o Gigabit. Como elemento fundamental, 802.11ac proporciona mejores tasas de hasta 433 Mbps por flujo de datos utilizando sistemas MIMO. Funciona en la banda de 5 GHz, por lo tanto, un rango que todavía no está tan saturado como el de 2.4 GHz y que además le ofrece un mayor ancho a cada canal. También cabe destacar que el estándar 802.11ac presenta compatibilidad total con 802.11a/b/g/n sin degradar las características de la red en presencia de implementaciones híbridas.

802.11ax: Conocido como WiFi 6, está diseñado para funcionar tanto en la banda espectral de 2,4 GHz como en la de 5 GHz. Soporta una velocidad máxima de 11 Gbps además de usar MIMO y MU-MIMO. Por último, destacar que en este estándar se introduce OFDMA para mejorar la eficiencia espectral general.

El uso de estándares más nuevos en WiFi Direct, como IEEE 802.11ac y 802.11ax, brinda ventajas en términos de velocidades de datos más altas y una comunicación más eficiente. Esto permite un intercambio de información más rápido entre los dispositivos conectados mediante WiFi Direct. Además, estos estándares suelen ser compatibles con una amplia gama de dispositivos, lo que facilita la creación de redes de comunicación P2P con múltiples nodos.

Sin embargo, es importante tener en cuenta que los dispositivos diseñados para soportar los estándares más nuevos de WiFi Direct pueden ser más costosos. Además, es posible que los dispositivos más antiguos no sean compatibles con estos estándares más recientes, lo que limitaría la interoperabilidad en una red WiFi Direct.

6.1.5 Prestaciones computacionales

En cuanto a las prestaciones computacionales en el contexto de los algoritmos de consenso, es importante considerar diversas métricas para evaluar el rendimiento y la capacidad del algoritmo en diferentes condiciones y configuraciones de red. Algunas de las métricas clave que se pueden tener en cuenta son:

Tiempo de procesamiento: Se refiere al tiempo necesario para que el algoritmo de consenso realice sus cálculos y llegue a un acuerdo entre los nodos. Un tiempo de procesamiento más corto indica una mayor eficiencia y capacidad de respuesta del algoritmo.

Uso de memoria: Referido a la cantidad de memoria requerida por el algoritmo de consenso para almacenar y procesar datos. Un uso eficiente de la memoria garantiza un mejor rendimiento y evita posibles cuellos de botella en el sistema.

Ancho de banda de la red: Se refiere a la cantidad de datos que se pueden transmitir a través de la red en un período de tiempo determinado. Un mayor ancho de banda permite una comunicación más rápida y fluida entre los nodos, lo que resulta en un mejor rendimiento del algoritmo.

Eficiencia energética: Es un aspecto crucial en aplicaciones móviles y dispositivos con recursos limitados, como dispositivos móviles y sensores IoT. Al diseñar algoritmos de consenso para WiFi Direct, se debe tener en cuenta el consumo de energía de los dispositivos involucrados. Esto implica optimizar los cálculos y las comunicaciones para reducir la carga de energía y prolongar la vida útil de la batería de los dispositivos.

Para medir y registrar estas métricas se utilizan herramientas y técnicas de monitoreo. Por ejemplo, se pueden emplear herramientas de perfilado de código para analizar el tiempo de ejecución y el uso de memoria de un algoritmo de consenso. Por otro lado, para medir el ancho de banda de la red se pueden utilizar herramientas de monitoreo de tráfico de red, como capturadores de paquetes.

Estas métricas de rendimiento pueden ayudar a identificar posibles limitaciones en el algoritmo de consenso, así como a optimizar su diseño e implementación. Además, es esencial considerar los requisitos de hardware y software del algoritmo y asegurarse de que se alineen adecuadamente con las capacidades de los equipos en los que se está ejecutando. Esto garantiza un rendimiento óptimo y evita problemas de compatibilidad o falta de recursos.

En el caso específico de WiFi Direct, la capacidad de procesamiento del dispositivo que se convierte en el GO es un factor importante a considerar. Para grupos P2P con un gran número de nodos y aplicaciones multimedia, se requiere un dispositivo GO con suficiente potencia de procesamiento y memoria para atender adecuadamente a los clientes conectados. Sin embargo, en aplicaciones más simples con menos nodos, como la transferencia de datos entre dispositivos móviles, los requisitos de procesamiento son menos significativos.

6.1.6 Nivel de batería

Los algoritmos de consenso, como Proof of Work (PoW) y Proof of Stake (PoS), requieren una cantidad significativa de potencia computacional, lo que a su vez consume una cantidad considerable de energía de la batería de los nodos participantes. El nivel de batería de un nodo puede tener un impacto importante en el rendimiento y la eficiencia del algoritmo.

Cuando el nivel de batería es bajo, el nodo puede experimentar retrasos en el proceso de consenso debido a la falta de energía para realizar los cálculos necesarios. Además, un nivel de batería bajo puede provocar que el nodo se apague, lo que resulta en una pérdida de participación en el proceso de consenso. Esto afecta negativamente al rendimiento y a la eficiencia del algoritmo al reducir la cantidad de nodos participantes y aumentar el tiempo necesario para llegar a un consenso.

En el contexto de WiFi Direct, se utilizan técnicas como TIM (Traffic Indication Map) y DTIM (Delivery Traffic Indication Message) para administrar eficientemente la energía. Estas técnicas asignan ranuras de tiempo en el intervalo de sincronización beacon, permitiendo a los dispositivos clientes saber cuándo deben activarse para recibir información de tráfico pendiente. Al controlar la activación/desactivación de los nodos y su participación en la comunicación de red contribuye a una mejora en la eficiencia energética y una mayor vida útil de la batería.

Además, al elegir un dispositivo para ser el GO en un grupo P2P de WiFi Direct, es importante considerar la vida útil de la batería del dispositivo. Si se elige un dispositivo con una vida útil de la batería baja como GO, es probable que se agote rápidamente, lo que resultaría en la disrupción del grupo P2P. Por lo tanto, es necesario seleccionar un dispositivo con una vida útil de la batería adecuada para mantener la estabilidad y continuidad del grupo P2P durante un período prolongado.

6.1.7 Número de interfaces WiFi del nodo

La cantidad de interfaces WiFi en un nodo impacta directamente en su capacidad de comunicación y, por ende, en el proceso de consenso. Un nodo con múltiples interfaces WiFi tiene la capacidad de establecer conexiones simultáneas con un mayor número de nodos, lo que resulta en un proceso de consenso más rápido y eficiente. Esto se debe a que puede iniciar más conexiones y enviar más mensajes a otros nodos, reduciendo el tiempo necesario para llegar a un consenso.

Por otro lado, un nodo con un número limitado de interfaces WiFi puede enfrentar dificultades para comunicarse con suficientes nodos, lo que resulta en un proceso de consenso más lento o incluso la incapacidad de alcanzar un consenso. Esto tiene un impacto negativo en el rendimiento general de la red.

Es importante tener en cuenta que la cantidad de interfaces WiFi puede variar según el dispositivo y las capacidades del hardware. Algunos dispositivos pueden tener una sola interfaz WiFi, mientras que otros pueden tener múltiples interfaces integradas o admitir la conexión de interfaces externas. El rango de valores de interfaces WiFi puede variar desde 1 hasta varios, dependiendo de las características del dispositivo y los requisitos de la aplicación.

No hay un número específico de interfaces WiFi que garantice automáticamente un rendimiento óptimo, ya que esto también depende de otros factores, como la capacidad de procesamiento y la capacidad de memoria del nodo. Es importante considerar el equilibrio entre la cantidad de interfaces WiFi y los recursos disponibles en el nodo para lograr un rendimiento eficiente en el proceso de consenso.

6.1.8 Opciones de seguridad

La seguridad garantiza la integridad, la confidencialidad y la autenticidad de la red. Para lograr una seguridad robusta, se utilizan protocolos como WPA3 y WPA2, que proporcionan mecanismos de protección eficaces contra amenazas.

WPA3 y WPA2 son protocolos de seguridad estándar ampliamente utilizados en redes WiFi. Utilizan algoritmos de cifrado avanzados, como AES (Advanced Encryption Standard) y TKIP (Temporal Key Integrity Protocol), para asegurar la confidencialidad de los datos transmitidos. El rango de valores para la seguridad de las redes WiFi puede variar, pero se recomienda utilizar WPA3 o, como mínimo, WPA2 con AES para una seguridad óptima.

Estos protocolos ofrecen autenticación sólida para garantizar que solo los dispositivos autorizados puedan unirse a la red. Además, implementan mecanismos de gestión de claves para proteger la integridad de los datos y prevenir ataques de suplantación de identidad.

Es esencial destacar que la seguridad en WiFi Direct también aborda la protección de la privacidad de los participantes. La información confidencial, como las opciones de votación en los algoritmos de consenso, debe ser transmitida de manera segura para evitar la interceptación y el uso malicioso de los datos.

6.1.9 Recuperación de errores durante caídas del sistema

La recuperación de errores es fundamental en WiFi Direct para mantener la integridad y consistencia del sistema. Sin una adecuada recuperación de errores, existe el riesgo de corrupción o pérdida de datos, lo que puede afectar negativamente el rendimiento y la confiabilidad de la red.

Los algoritmos de consenso se basan en mecanismos de replicación y votación para mantener la coherencia frente a fallos. Sin embargo, en caso de fallo del sistema sin una recuperación de errores adecuada, estos mecanismos pueden no funcionar correctamente, lo que resulta en inconsistencias en la red.

Una recuperación de errores adecuada permite que el sistema continúe procesando transacciones incluso después de una caída. Esto es esencial para mantener la disponibilidad del sistema y garantizar la continuidad de los servicios para los usuarios. Además de la recuperación de errores, es importante contar con mecanismos de detección temprana de fallos en WiFi Direct. Esto implica una supervisión constante de la conectividad de los nodos y la detección de comportamientos anómalos. Estos mecanismos permiten identificar y abordar problemas potenciales antes de que se conviertan en fallos mayores, contribuyendo a la estabilidad y confiabilidad del sistema.

Es importante implementar métricas y valores específicos para medir la eficacia de la recuperación de errores, como el tiempo de recuperación, la tasa de éxito de la recuperación y la cantidad de datos recuperados. Estas métricas ayudan a evaluar el rendimiento de los mecanismos de recuperación y a identificar posibles mejoras en el sistema.

6.2 Optimización de la selección del dispositivo GO en redes P2P

En el entorno de las redes P2P, la elección adecuada del dispositivo GO es esencial para garantizar un servicio óptimo para todos los clientes del grupo. Aunque un dispositivo con una vida útil de batería más larga puede prolongar la duración del grupo, existen otros factores a considerar, como una conexión a Internet débil o incluso capacidades de procesamiento limitadas.

En este contexto, se propone un nuevo algoritmo de consenso basado en un mecanismo de puntuación que utilizará un polinomio ponderado cuyo objetivo será abordar las limitaciones de los enfoques tradicionales y proporcionar un mecanismo flexible para la toma de decisiones consensuadas. Este algoritmo utiliza la importancia asignada a cada nodo y aplica sesgos estratégicos basados en su relevancia relativa, permitiendo una mayor influencia de los nodos más importantes en el resultado final del consenso.

6.2.1 Propuesta del Algoritmo

a. Inicialización:

La puntuación final asignada a cada nodo representa su nivel de influencia relativa dentro del sistema y se expresa como un valor numérico. Esta puntuación puede ser estática, si se mantiene constante a lo largo del tiempo, o dinámica, si se ajusta en función de cambios en las condiciones del sistema.

Dicha puntuación tendrá asignado un Valor de Importancia (*VI*) en cada nodo, que se calcula mediante la combinación de diferentes criterios (*Ci*) asociados a ese nodo. Cada criterio tiene un sesgo o importancia relativa (*Si*) que determina su peso en la puntuación final del nodo. El sesgo refleja la influencia relativa de cada criterio en el sistema.

Al calcular el Valor de Importancia, se realiza una suma ponderada de los criterios considerados. Cada criterio es multiplicado por su respectivo sesgo y luego se realiza la suma de todos los términos ponderados. La fórmula general para el cálculo del Valor de Importancia sería:

$$VI = \left[\sum_{i=1}^{n} Ci * Si\right] * (1 - P) + P$$

Donde:

VI: Valor de Importancia del nodo.
Ci: Valor del criterio i del nodo en consideración.
Si: Sesgo del criterio i.
P: Prioridad del nodo.

Se incluirá también la prioridad en el cálculo del Valor de Importancia. Será un valor numérico que por defecto será 0 y se pondrá a 1 si se quiere forzar a ese nodo para ser el próximo GO.

b. Cálculo de puntuación:

El Valor de Importancia muestra la disposición de un dispositivo P2P para convertirse en un GO dentro de un grupo. El dispositivo que tenga el Valor de Importancia más alto se convertirá en el próximo GO.

El Valor de Importancia es un parámetro que debe elegirse cuidadosamente. El enfoque más útil para elegir el Valor de Importancia se basa en las capacidades del dispositivo para servir como GO, ya que una vez que un dispositivo se convierte en GO, debe servir a todos los dispositivos cliente P2P asociados para la comunicación.

Todos los datos destinados a los clientes P2P en un grupo dado deben ser enrutados a través del GO. De manera similar, los dispositivos en el mismo grupo P2P también se comunican entre sí a través del GO. Por lo tanto, el GO será responsable de todo el reenvío de datos y funcionará como un Soft-AP.

En esta formulación, se considerará que el valor de n corresponde al número total de nodos en la red y que el valor de i representa los cuatro criterios específicos que se utilizarán para evaluar cada nodo expuestos a continuación:

1. **Mayor número de nodos en la zona de cobertura:** Para extraer esta información, se pueden utilizar técnicas como la exploración de la red para identificar los dispositivos vecinos y estimar el número de nodos en la zona de cobertura.

2. **Nivel de intensidad de señal recibida en el nodo:** Esta información se puede extraer midiendo la potencia de la señal recibida (RSSI) o utilizando métricas de calidad de la conexión.

3. **Nivel de batería del nodo:** Para extraer esta información, se puede acceder al estado de la batería del dispositivo a través de la API o el sistema operativo correspondiente.

4. **Ancho de banda:** Utilizando métodos o herramientas disponibles para realizar pruebas de ancho de banda en cada nodo se puede obtener esta información. Esto implica enviar y recibir datos de prueba para medir la tasa de transferencia de datos y determinar el ancho de banda.

Por otro lado, se deben asignar sesgos a cada una de las características que se quieren considerar en la ecuación polinómica. Esto se hará en función de la importancia relativa de cada característica. Para ello se utilizarán las conclusiones obtenidas en capítulos anteriores para considerar las características de cada nodo que pueden influir en la elección del mejor punto de acceso.

Para esto es recomendable establecer perfiles que representen casos de uso comunes o situaciones particulares relevantes para la selección del GO. Estos perfiles ayudarán a evaluar y ajustar los pesos y el rendimiento del algoritmo en diferentes contextos, permitiendo una mayor adaptabilidad y optimización del sistema.

A continuación, se presentan varios ejemplos de perfiles que podrían considerarse al implementar el algoritmo de selección del GO:

Perfil de cobertura amplia: Tiene como objetivo asegurar que el dispositivo seleccionado como GO tenga la capacidad de atender a la mayor cantidad de nodos posible, priorizando la expansión de la red. Se consigue maximizando la cobertura de la red, permitiendo que un mayor número de dispositivos se conecten y se beneficien de la comunicación P2P.

Sesgo 1 = Alto (0.55)
Sesgo 2 = Moderado (0.20)
Sesgo 3 = Bajo (0.10)
Sesgo 4 = Bajo (0.15)

Justificación: En un perfil que busca maximizar la cobertura de la red, el número de nodos en la zona de cobertura es el criterio más importante, ya que cuantos más dispositivos pueda atender el GO, mayor será la extensión de la red. La intensidad de la señal y el ancho de banda también son factores relevantes, ya que una señal fuerte y un alto ancho de banda aseguran una comunicación efectiva entre los nodos. La batería del nodo tiene un peso menor en este perfil, ya que, aunque es relevante, no es tan crítico como los otros criterios para garantizar una amplia cobertura.

Perfil de estabilidad de la señal: Busca garantizar una comunicación confiable y estable, priorizando la calidad de la señal recibida en cada nodo. Se busca equilibrar la cantidad de nodos en la zona de cobertura con la calidad de la señal y se considera la disponibilidad de energía suficiente en los nodos seleccionados. Esto garantizará que el GO seleccionado tenga una buena calidad de señal, evitando problemas de conectividad y asegurando una comunicación confiable.

Sesgo 1 = Moderado (0.20)
Sesgo 2 = Alto (0.50)
Sesgo 3 = Moderado (0.20)
Sesgo 4 = Bajo (0.10)

Justificación: En este perfil, la estabilidad de la señal es el criterio primordial, por lo que se le asigna el mayor peso. Una señal fuerte y estable asegura una comunicación confiable y de alta calidad entre los nodos. El número de nodos en la zona de cobertura sigue siendo relevante, ya que una buena cobertura permite una comunicación fluida en toda el área. La batería del nodo también tiene un peso significativo, ya que una batería suficientemente cargada garantiza que el GO pueda mantener su función de manera estable.

Perfil de conservación de energía: En este perfil, se considera el nivel de batería del nodo como un factor determinante en la elección del GO. Los dispositivos con niveles más altos de batería tendrán una mayor probabilidad de ser seleccionados, ya que se espera que puedan mantenerse activos durante más tiempo y brindar una conexión estable a los demás nodos. Además, se busca equilibrar la conservación de energía con otros criterios, como la cobertura y la calidad de la señal. Se evalúa la eficiencia energética de los nodos y se selecciona aquellos que puedan ofrecer un equilibrio adecuado entre conservación de energía y rendimiento de la red.

Sesgo 1 = Bajo (0.10)
Sesgo 2 = Moderado (0.20)
Sesgo 3 = Alto (0.60)
Sesgo 4 = Bajo (0.10)

Justificación: En este perfil, la conservación de energía es el factor clave, por lo que el nivel de batería del nodo recibe el mayor peso. Se busca maximizar el tiempo de actividad del GO seleccionado y, por lo tanto, se prefiere que tenga una batería con mayor capacidad.

Perfil de máxima transferencia de datos: Se enfoca en obtener la máxima capacidad de transferencia de información en la red. Garantizando una amplia cobertura, una señal estable, un consumo de energía reducido y un alto ancho de banda se obtendrá una comunicación eficiente y rápida entre los nodos de la red.

Sesgo 1 = Bajo (0.10)
Sesgo 2 = Moderado (0.20)
Sesgo 3 = Bajo (0.10)
Sesgo 4 = Alto (0.60)

Justificación: En este perfil, el objetivo es maximizar la capacidad de transferencia de información, por lo que el ancho de banda y la intensidad de señal son criterios clave. Un ancho de banda alto y una señal fuerte permiten una comunicación eficiente y rápida entre los nodos.

Perfil equilibrado: Si se desea tener un equilibrio entre la cobertura, la estabilidad de la señal, la conservación de energía y la máxima transferencia de datos se asignarán valores idénticos a todas las características.
Sesgo 1 = Moderado (0.25)
Sesgo 2 = Moderado (0.25)

Sesgo 3 = Moderado (0.25)
Sesgo 4 = Moderado (0.25)

Una vez obtenidos los valores de los criterios (Ci) para cada nodo y definidos los sesgos adecuados, se procede a calcular el Valor de Importancia (VI) de cada nodo utilizando la fórmula mencionada anteriormente (véase Algoritmo-1).

Algoritmo-1 – Cálculo Valor de Importancia nodo i

Inputs: PERFIL, C1, C2, C3, C4, P
Outputs: VI

```
Función CALC_VI (PERFIL, C1, C2, C3, C4, P):
VI = null
        switch (PERFIL)
            case 1:
            S1 = 0.55
            S2 = 0.20
            S3 = 0.10
            S4 = 0.15
            VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) +(C4 * S4)] *(1-P) + P
            break

            case 2:
            S1 = 0.20
            S2 = 0.50
            S3 = 0.20
            S4 = 0.10
            VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) +(C4 * S4)] *(1-P) + P
            break

            case 3:
            S1 = 0.10
            S2 = 0.20
            S3 = 0.60
            S4 = 0.10
            VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) +(C4 * S4)] *(1-P) + P
            break

            case 4:
            S1 = 0.10
            S2 = 0.20
            S3 = 0.10
            S4 = 0.60
            VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) +(C4 * S4)] *(1-P) + P
            break

            case 5:
            S1 = 0.25
            S2 = 0.25
            S3 = 0.25
            S4 = 0.25
            VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) +(C4 * S4)] *(1-P) + P
            break
            default
            VI = 0.0
```

c. Proceso de consenso:

Una vez que se han calculado todas las puntuaciones de los nodos que conforman el grupo WiFi Direct, se procede al proceso de consenso para seleccionar el nodo con la puntuación más alta como la opción preferida. Este proceso implica comparar las puntuaciones de todos los nodos y aplicar una función para determinar el nodo seleccionado.

Para realizar el proceso de consenso, todos los miembros del grupo envían sus métricas al nodo GO, donde se recopilan todas y se ejecuta el algoritmo de puntuaciones de los nodos miembros del grupo WiFi Direct.

El primer paso a la hora de llegar a un consenso consiste en comprobar si algún nodo tiene asignada una prioridad. Si algún nodo tiene una prioridad asignada, se le designará como Backup GO.

Lo próximo será normalizar las puntuaciones ya que es necesario para asegurar que las puntuaciones estén en un rango comparativo y puedan ser evaluadas de manera justa.

Para normalizar la puntuación se siguen los siguientes pasos:

1. Obtener los Valores de Importancia de todos los nodos.
2. Encontrar el valor mínimo y el valor máximo entre todas las puntuaciones.
3. Para cada nodo aplicar la siguiente fórmula para normalizar su puntuación:

$$\text{vi\_norm} = (\text{vi} - \min) / (\max - \min)$$

Esta fórmula ajusta los valores de puntuación al rango de 0 a 1, donde el valor mínimo tiene una puntuación de 0 y el valor máximo tiene una puntuación de 1. Los valores intermedios se escalan proporcionalmente en función de su relación con los valores mínimo y máximo.

Una vez que se han calculado todas las puntuaciones normalizadas, el nodo GO compara las puntuaciones y determina cuál será el sucesor en caso de fallos. El resultado se envía de vuelta a los nodos.

En caso de que varios dispositivos obtengan la misma puntuación final, se utilizará como criterio de desempate la dirección MAC más pequeña en orden lexicográfico entre todos los dispositivos que tengan esa misma puntuación.

Este algoritmo está pensado para su ejecución cíclica y preventiva para evitar la disolución del grupo en caso de caída del nodo que ejerce el papel de GO.

6.2.2 Escenarios para aplicación de perfiles

Gracias a la versatilidad y flexibilidad del algoritmo se pueden adoptar los perfiles mencionados anteriormente para aplicarlos a una amplia gama de escenarios. Por poner algunos ejemplos:

Perfil de cobertura amplia: En el escenario de zona de grandes catástrofes, el perfil de cobertura amplia se traduce en maximizar la cobertura de la red P2P basada en WiFi Direct. El objetivo es garantizar que el dispositivo seleccionado como GO pueda atender a la mayor cantidad de nodos posible en áreas afectadas por desastres naturales. Esto implica establecer conexiones entre dispositivos móviles *ad hoc* para compartir información vital, incluso cuando las infraestructuras de comunicación tradicionales están dañadas. Priorizar la expansión de la red y permitir que un mayor número de dispositivos se conecten facilita la comunicación y la coordinación de esfuerzos de rescate y ayuda.

Perfil de estabilidad de la señal: En el contexto de streaming en tiempo real, el perfil de estabilidad de la señal es fundamental. Al establecer una conexión P2P utilizando WiFi Direct, se prioriza la calidad de la señal recibida en cada nodo. Esto asegura una comunicación confiable y estable entre los dispositivos, evitando problemas de conectividad y pérdida de datos. Además, se considera la disponibilidad de energía suficiente en los nodos seleccionados, lo que garantiza que el dispositivo designado como GO tenga una buena calidad de señal y pueda mantener una transmisión fluida de contenido multimedia en tiempo real.

Perfil de conservación de energía: En el escenario de comunicaciones en un hospital, la conservación de energía es clave para asegurar una conexión estable y duradera entre dispositivos médicos. El perfil de conservación de energía prioriza los dispositivos con niveles más altos de batería, ya que se espera que puedan mantenerse activos durante más tiempo y brindar una comunicación confiable a los demás nodos. Al establecer conexiones directas y seguras mediante WiFi Direct, se reduce la dependencia de la red WiFi convencional y se optimiza el consumo de energía de los dispositivos médicos, garantizando así una comunicación eficiente y segura en un entorno crítico como un hospital.

Perfil de máxima transferencia de datos: En el contexto de la monitorización en tiempo real, el perfil de máxima transferencia de datos busca obtener una comunicación eficiente y rápida entre los nodos de la red. Al utilizar WiFi Direct, se garantiza una amplia cobertura, una señal estable y un consumo de energía reducido. Estos aspectos contribuyen a una mayor capacidad de transferencia de información en tiempo real desde los sensores distribuidos en diferentes dispositivos hasta un dispositivo central de monitoreo. La comunicación eficiente y rápida permite a los usuarios controlar y supervisar diversos aspectos, como la temperatura, la seguridad y el consumo de energía, en aplicaciones de domótica.

A continuación, se muestran varios ejemplos prácticos de aplicación del algoritmo frente a un escenario concreto.

Escenario de grandes catástrofes.

Especificaciones: Se presenta una situación de emergencia de un terremoto devastador en una zona urbana densamente poblada. La infraestructura de comunicaciones convencionales se ve severamente afectada, lo que dificulta la comunicación entre las personas y los equipos de rescate. En este escenario, se implementa una red de comunicación de emergencia utilizando nodos móviles para facilitar la comunicación y coordinación.

Para este caso específico se definen 5 nodos y el perfil de cobertura amplia. Se asignarán valores aleatorios para cada criterio y se calculará el Valor de Importancia de cada nodo. Dado que no existe ningún nodo con prioridad se establecerá en todos los nodos el valor de prioridad predeterminado de 0.

Perfil de cobertura amplia:

S1 = Alto (0.55)
S2 = Moderado (0.20)
S3 = Bajo (0.10)
S4 = Bajo (0.15)

Nodo 1:

C1 = Número de nodos en la zona de cobertura: 1
C2 = Intensidad de señal recibida: -70 dBm → Nivel RSSI 3
C3 = Nivel de batería: 10%
C4 = Ancho de banda: 10 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(1 * 0.55) + (3 * 0.20) + (10 * 0.10) + (10 * 0.15)] * (1 - 0) + 0
= (0.55 + 0.60 + 1.00 + 1.50) * 1 + 0
= 3.65

Nodo 2:

C1 = Número de nodos en la zona de cobertura: 2
C2 = Intensidad de señal recibida: -60 dBm → Nivel RSSI 4
C3 = Nivel de batería: 50%
C4 = Ancho de banda: 2 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(2 * 0.55) + (4 * 0.20) + (50 * 0.10) + (2 * 0.15)] * (1 - 0) + 0
= (1.10 + 0.80 + 5.00 + 0.30) * 1 + 0
= 7.20

Nodo 3:

C1 = Número de nodos en la zona de cobertura: 1
C2 = Intensidad de señal recibida: -80 dBm → Nivel RSSI 1
C3 = Nivel de batería: 60%
C4 = Ancho de banda: 8 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(1 * 0.55) + (1 * 0.20) + (60 * 0.10) + (8 * 0.15)] * (1 - 0) + 0
= (0.55 + 0.20 + 6.00 + 1.20) * 1 + 0
= 7.95

Nodo 4 (Mayor número de nodos en la zona de cobertura):

C1 = Número de nodos en la zona de cobertura: 5
C2 = Intensidad de señal recibida: -65 dBm → Nivel RSSI 3
C3 = Nivel de batería: 50%
C4 = Ancho de banda: 8 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(5 * 0.55) + (3 * 0.20) + (50 * 0.10) + (8 * 0.15)] * (1 - 0) + 0
= (2.75 + 0.60 + 5.00 + 1.20) * 1 + 0
= 9.55

Nodo 5:

C1 = Número de nodos en la zona de cobertura: 1
C2 = Intensidad de señal recibida: -75 dBm → Nivel RSSI 2
C3 = Nivel de batería: 75%
C4 = Ancho de banda: 6 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(1 * 0.55) + (2 * 0.20) + (75 * 0.10) + (6 * 0.15)] * (1 - 0) + 0
= (0.55 + 0.40 + 7.50 + 0.90) * 1 + 0
= 9.35

Para normalizar los valores, utilizaremos la fórmula vi_norm = (vi - min) / (max - min), donde el valor máximo es 9.55 y el valor mínimo es 3.35.

Para el Nodo 1:
vi_norm = (3.65 – 3.65) / (9.55 – 3.65) = 0 / 5.90 = 0.00

Para el Nodo 2:
vi_norm = (7.20 – 3.65) / (9.55 – 3.65) = 3.55 / 5.90 = 0.60

Para el Nodo 3:
vi_norm = (7.95 – 3.65) / (9.55 – 3.65) = 4.30 / 5.90 = 0.73

Para el Nodo 4:
vi_norm = (9.55 – 3.65) / (9.55 – 3.65) = 5.90 / 5.90 = 1.00

Para el Nodo 5:
vi_norm = (9.35 – 3.65) / (9.55 – 3.65) = 5.70 / 5.90 = 0.97

Basado en los valores normalizados, podemos hacer la siguiente conclusión:

El Nodo 4 tiene el mayor valor de importancia normalizado, con un valor de 1. Esto indica que, según los criterios dados y el perfil de cobertura amplia, el Nodo 4 es el más importante en términos de satisfacer las necesidades de cobertura de la red.

Escenario de monitoreo de recursos naturales en áreas remotas

Especificaciones: Se está llevando a cabo un proyecto de monitoreo ambiental en áreas remotas, como bosques o regiones montañosas, donde no hay acceso a redes eléctricas convencionales. Los dispositivos utilizados para recopilar datos ambientales, como sensores de temperatura y humedad, están alimentados por baterías recargables o energía solar limitada. La comunicación entre los dispositivos es crucial para recopilar datos en tiempo real y tomar decisiones basadas en la información recopilada.

En este escenario, el perfil de conservación de energía sería fundamental para asegurar que los nodos seleccionados como GO tengan baterías con niveles suficientemente altos para mantener una operación estable durante un período prolongado. Los nodos con niveles de batería más altos tendrían una mayor probabilidad de ser seleccionados como GO, ya que se espera que puedan mantenerse activos y proporcionar una conexión confiable a otros nodos.

Para este caso específico se definen los mismos 5 nodos del ejemplo anterior y el perfil de conservación de energía. Dado que no existe ningún nodo con prioridad se establecerá en todos los nodos el valor de prioridad predeterminado de 0.

Perfil de conservación de energía:

S1 = Bajo (0.10)
S2 = Moderado (0.20)
S3 = Alto (0.60)
S4 = Bajo (0.10)

Nodo 1:

C1 = Número de nodos en la zona de cobertura: 1
C2 = Intensidad de señal recibida: -70 dBm → Nivel RSSI 3
C3 = Nivel de batería: 10%
C4 = Ancho de banda: 10 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(1 * 0.10) + (3 * 0.20) + (10 * 0.60) + (10 * 0.10)] * (1 - 0) + 0
= (0.10 + 0.60 + 6.00 + 1.00) * 1 + 0
= 7.70

Nodo 2:

C1 = Número de nodos en la zona de cobertura: 2
C2 = Intensidad de señal recibida: -60 dBm → Nivel RSSI 4
C3 = Nivel de batería: 50%
C4 = Ancho de banda: 2 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(2 * 0.10) + (4 * 0.20) + (50 * 0.60) + (2 * 0.10)] * (1 - 0) + 0
= (0.20 + 0.80 + 30.00 + 0.20) * 1 + 0
= 31.20

Nodo 3:

C1 = Número de nodos en la zona de cobertura: 1
C2 = Intensidad de señal recibida: -80 dBm → Nivel RSSI 1
C3 = Nivel de batería: 60%
C4 = Ancho de banda: 8 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(1 * 0.10) + (1 * 0.20) + (60 * 0.60) + (8 * 0.10)] * (1 - 0) + 0
= (0.10 + 0.20 + 36.00 + 0.80) * 1 + 0
= 37.10

Nodo 4:

C1 = Número de nodos en la zona de cobertura: 5
C2 = Intensidad de señal recibida: -65 dBm → Nivel RSSI 3
C3 = Nivel de batería: 50%
C4 = Ancho de banda: 8 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(5 * 0.10) + (3 * 0.20) + (50 * 0.60) + (8 * 0.10)] * (1 - 0) + 0
= (0.50 + 0.60 + 30.00 + 0.80) * 1 + 0
= 31.90

Nodo 5 (Mayor porcentaje de batería):

C1 = Número de nodos en la zona de cobertura: 1
C2 = Intensidad de señal recibida: -75 dBm → Nivel RSSI 2
C3 = Nivel de batería: 75%:
C4 = Ancho de banda: 6 Mbps

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(1 * 0.10) + (2 * 0.20) + (75 * 0.60) + (6 * 0.10)] * (1 - 0) + 0
= (0.10 + 0.40 + 45.00 + 0.60) * 1 + 0
= 46.10

Para normalizar los valores, utilizaremos la fórmula vi_norm = (vi - min) / (max - min), donde el valor máximo es 46.10 y el valor mínimo es 7.70.

Para el Nodo 1:
vi_norm = (7.70 – 7.70) / (46.10 – 7.70) = 0 / 38.40 = 0.00

Para el Nodo 2:
vi_norm = (31.20 – 7.70) / (46.10 – 7.70) = 23.50 / 38.40 = 0.61

Para el Nodo 3:
vi_norm = (37.10 – 7.70) / (46.10 – 7.70) = 29.40 / 38.40 = 0.77

Para el Nodo 4:
vi_norm = (31.90 – 7.70) / (46.10 – 7.70) = 24.20 / 38.40 = 0.63

Para el Nodo 5:
vi_norm = (46.10 – 7.70) / (46.10 – 7.70) = 38.40 / 38.40 = 1.00

Basado en los valores normalizados, podemos hacer la siguiente conclusión:

En este caso, el Nodo 5 es el que tiene mejor porcentaje de batería. La elección de este perfil permite a los dispositivos con niveles más altos de batería ser seleccionados preferentemente como GO, lo que garantiza que puedan proporcionar una conexión estable a otros nodos durante un período prolongado. Al equilibrar la conservación de energía con otros criterios, como la calidad de la señal y la cobertura, el algoritmo logra maximizar la vida útil de la batería sin comprometer la confiabilidad de la comunicación.

Escenario para aplicar prioridad

Especificaciones: Se presenta la organización de un evento en el que los asistentes necesitan conectarse a través de WiFi Direct para compartir archivos y participar en actividades interactivas. Se tienen varios dispositivos disponibles, como smartphones, tablets y laptops, que actuarán como nodos en la red de Wi-Fi Direct.

Se podría asignar la máxima prioridad a un nodo que actúe como fuente de contenido. Este dispositivo sería responsable de transmitir la información clave a los demás dispositivos. Al asignarle la máxima prioridad, aseguras una transferencia de datos fluida y sin interrupciones durante la presentación.

Para este caso específico se definen 3 nodos y el perfil equilibrado. Se asignarán valores aleatorios para cada criterio y se calculará el Valor de Importancia de cada nodo. Dado que en este caso si existe prioridad, se establecerá en todos los nodos el valor de prioridad predeterminado de 0 a excepción del nodo 1 que contará con prioridad 1.

Perfil equilibrado:

Sesgo 1 = Moderado (0.25)
Sesgo 2 = Moderado (0.25)
Sesgo 3 = Moderado (0.25)
Sesgo 4 = Moderado (0.25)

Nodo 1 (Fuente de contenido prioritaria):

C1 = Número de nodos en la zona de cobertura: 3
C2 = Intensidad de señal recibida: -60 dBm → Nivel RSSI 4
C3 = Nivel de batería: 80%
C4 = Ancho de banda: 20 Mbps
Prioridad: 1 (máxima prioridad)

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(3 * 0.25) + (4 * 0.25) + (80 * 0.25) + (20 * 0.25)] * (1 - 1) + 1
= (0.75 + 1.00 + 20.00 + 5.00) * 0 + 1
= 0 + 1
= 1.00

Nodo 2:

C1 = Número de nodos en la zona de cobertura: 1
C2 = Intensidad de señal recibida: -65 dBm → Nivel RSSI 3
C3 = Nivel de batería: 70%
C4 = Ancho de banda: 15 Mbps
Prioridad: 0

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(1 * 0.25) + (3 * 0.25) + (70 * 0.25) + (15 * 0.25)] * (1 - 0) + 0
= (0.25 + 0.75 + 17.50 + 3.75) * 1 + 0
= 22.25 * 1 + 0
= 22.25

Nodo 3:

C1 = Número de nodos en la zona de cobertura: 1
C2 = Intensidad de señal recibida: -70 dBm → Nivel RSSI 3
C3 = Nivel de batería: 90%
C4 = Ancho de banda: 12 Mbps
Prioridad: 0

VI = [(C1 * S1) + (C2 * S2) + (C3 * S3) + (C4 * S4)] * (1 - P) + P
= [(1 * 0.25) + (3 * 0.25) + (90 * 0.25) + (12 * 0.25)] * (1 - 0) + 0
= (0.25 + 0.75 + 22.50 + 3.00) * 1 + 0
= 26.50 * 1 + 0
= 26.50

En este caso, el Nodo 1 al actuar como la fuente de contenido, tiene asignada prioridad 1 y obtiene un Valor de Importancia de 1. Esto se debe a que cumple con criterios favorables en cuanto al número de nodos en la zona de cobertura, intensidad de señal recibida, nivel de batería y ancho de banda por lo que será designado como Backup y será el encargado de transmitir la información clave durante el evento.

6.2.3 Desarrollo del algoritmo con transferencia de valores simulados

El algoritmo propuesto, cuyo enfoque y objetivos se describen previamente, se traducirá ahora a código Java para su ejecución en un entorno de desarrollo. Sin embargo, en lugar de utilizar datos reales, se simularán valores que representen diferentes situaciones y escenarios que podrían encontrarse en la práctica.

Esta simulación nos permitirá comprender de manera más visual y práctica cómo el algoritmo procesa y maneja estos valores, y cómo influyen en los resultados obtenidos. Al utilizar valores simulados, también se simplifica la demostración del algoritmo, evitando la necesidad de datos reales o fuentes de información externas.

A lo largo de este apartado, se explicará paso a paso la implementación del algoritmo en Java. Se proporcionarán ejemplos de salida para facilitar la comprensión de cada paso y sus implicaciones en el funcionamiento del algoritmo con un posterior análisis de los resultados obtenidos con los valores simulados.

Por último, a través de esta simulación se podrán extraer conclusiones valiosas sobre el comportamiento y la efectividad del algoritmo, así como identificar posibles mejoras o ajustes necesarios.

Primera simulación:

Cuando se ejecuta esta implementación, se solicita al usuario ingresar el número de nodos que se desea simular, así como el perfil común para todos los nodos. Se verifica que estos valores estén dentro del rango válido, que en este caso son los cinco perfiles definidos al inicio del algoritmo. Además, se le solicita al usuario seleccionar un nodo para darle prioridad en caso de ser necesario.

En los demás casos, se ha optado por generar valores numéricos aleatorios dentro de rangos específicos para obtener el resto de las métricas. Estos valores se ajustan para reflejar condiciones realistas de una red inalámbrica y permiten al algoritmo analizar y tomar decisiones en función de ellos, utilizando la clase ‘MetricGenerator’.

En la clase ‘Main’, se crean los hilos correspondientes a cada nodo para su posterior ejecución. Por cada nodo, se muestra por pantalla el número de nodo correspondiente y el perfil acordado, lo cual proporciona información básica sobre el nodo que se está simulando. Si el nodo tiene asignada una prioridad, también se muestra esta información.

A continuación, se generan valores aleatorios para las métricas del nodo. Utilizando estas métricas, se calcula el *Valor de Importancia* utilizando el método calc_vi() de la clase ‘AlgorithmSimulator’. Luego, se redondea el valor y se muestra por pantalla junto con las métricas correspondientes.

Una vez que todos los hilos han finalizado su ejecución, se realiza el proceso de consenso y selección del nodo de respaldo mediante una llamada al método ‘processScores()’ del objeto ‘ConsensusManager’. Esto implica recopilar y procesar todas las puntuaciones de los nodos para determinar el nodo de respaldo.

A continuación, se muestran algunos ejemplos de ejecución:

```
Ingrese el número de nodos a simular: 5
Ingrese el perfil común para todos los nodos (1-5): 2
Si desea darle prioridad a algún nodo, selecciónelo (0 para ninguno): 0
Nodo: 1
Perfil: Estabilidad de la señal
Número de nodos: 5
RSSI: 2
Batería: 45.0 %
Ancho de Banda: 7 Mbps
Puntuación: 11.7
--------------------
Nodo: 2
Perfil: Estabilidad de la señal
Número de nodos: 5
RSSI: 4
Batería: 76.0 %
Ancho de Banda: 92 Mbps
Puntuación: 27.4
--------------------
Nodo: 3
Perfil: Estabilidad de la señal
Número de nodos: 2
RSSI: 4
Batería: 9.0 %
Ancho de Banda: 48 Mbps
Puntuación: 9.0
--------------------
Nodo: 4
Perfil: Estabilidad de la señal
Número de nodos: 5
RSSI: 0
Batería: 83.0 %
Ancho de Banda: 63 Mbps
Puntuación: 23.9
--------------------
Nodo: 5
Perfil: Estabilidad de la señal
Número de nodos: 3
RSSI: 4
Batería: 13.0 %
Ancho de Banda: 23 Mbps
Puntuación: 7.5
--------------------
Puntuaciones normalizadas:
Nodo 1: 0.21
Nodo 2: 1.0
Nodo 3: 0.08
Nodo 4: 0.82
Nodo 5: 0.0
Backup GO: Nodo 2
```

Figura 6-1 Ejecución perfil "Estabilidad de la señal" sin prioridad

Se puede observar que, aunque todos los nodos tienen el mismo perfil, cada uno difiere en las métricas individuales. En base a estas métricas se establecen unas puntuaciones individuales que refleja su rendimiento. Estas puntuaciones se utilizan para comparar y seleccionar el nodo de respaldo. En este caso es el Nodo 2 cuenta con mejores prestaciones haciendo énfasis en el nivel RSSI de señal ya que el objetivo es garantizar un rendimiento óptimo y confiable dentro de la red inalámbrica en caso de caída del nodo principal.

```
Ingrese el número de nodos a simular: 3
Ingrese el perfil común para todos los nodos (1-5): 3
Si desea darle prioridad a algún nodo, selecciónelo (0 para ninguno): 1
Nodo: 1
Perfil: Conservación de energía
Prioridad: 1
Número de nodos: 3
RSSI: 4
Batería: 28.0 %
Ancho de Banda: 34 Mbps
Puntuación: 1.0
--------------------
Nodo: 2
Perfil: Conservación de energía
Número de nodos: 3
RSSI: 0
Batería: 97.0 %
Ancho de Banda: 21 Mbps
Puntuación: 60.6
--------------------
Nodo: 3
Perfil: Conservación de energía
Número de nodos: 1
RSSI: 2
Batería: 36.0 %
Ancho de Banda: 50 Mbps
Puntuación: 27.1
--------------------
Backup GO: Nodo 1
```

Figura 6-2 Ejecución perfil "Conservación de energía" con prioridad

Esta vez contamos con una prioridad en uno de los nodos. Esto hace que no se realice la comparativa entre puntuaciones ya que se considera que uno de los nodos será el más indicado para tomar el control en caso de caída del nodo principal. Se observa que, aunque este nodo obtiene la puntuación más baja ya que tiene peores características se le designa como backup GO.

Pueden existir varios motivos para tomar esta decisión. Como la confiabilidad del nodo ya que puede ser considerado más confiable o robusto en comparación con los otros nodos, incluso si obtiene una puntuación más baja. Esto podría deberse a factores como su historial de rendimiento, estabilidad de conexión o capacidad de recuperación ante fallas.

Otro motivo podría ser que dicho nodo puede tener capacidades o funcionalidades especiales que lo hacen más adecuado para asumir el rol de backup GO. Por ejemplo, puede tener recursos adicionales, mayor capacidad de procesamiento o una configuración específica que lo habilite para desempeñar mejor esta función en caso de fallos.

Lo que se ha logrado hasta ahora es la ejecución de la simulación del algoritmo utilizando valores aleatorios, lo cual nos ha permitido obtener conclusiones significativas sobre la selección del nodo de respaldo en una red inalámbrica. Ahora, es el momento de llevar a cabo la aplicación práctica de estas conclusiones en la transmisión de datos reales. Siguiendo los hallazgos y recomendaciones obtenidos en capítulos anteriores, procederemos a implementar el protocolo MQTT para la transmisión de datos.

Este protocolo, conocido por su eficiencia, confiabilidad y capacidad de tolerancia a fallos, nos permitirá establecer una comunicación efectiva y segura entre los nodos de la red. Al utilizar MQTT, podremos asegurar una transmisión de datos eficiente, con garantía de entrega y capacidad de respaldo en caso de fallos en el nodo principal. De esta manera, la combinación de la simulación del algoritmo y la implementación del protocolo MQTT nos proporcionará una solución completa y robusta para la transmisión de datos en entornos de redes inalámbricas.

6.3 Transferencia de las métricas de cada nodo con MQTT

En un entorno de comunicación basado en WiFi Direct, MQTT se destaca como un protocolo eficiente y ligero para la transferencia de métricas entre los nodos que participan en la red tal y como vimos en capítulos anteriores.

La implementación de MQTT en un entorno de WiFi Direct implica una serie de pasos clave para garantizar una transferencia eficiente de las métricas de cada nodo:

1. **Configuración del entorno MQTT:** La arquitectura de comunicación se basa en el paradigma publicador/suscriptor. En esta arquitectura, el cliente generalmente mantiene una conexión persistente con el bróker. Los clientes pueden actuar como publicadores, suscriptores o ambos al mismo tiempo. El publicador por lo general no envía directamente los datos a los suscriptores, ya que suele desconocer la ubicación de estos últimos. En cambio, son los suscriptores quienes se suscriben para recibir datos de su interés.

2. **Definición del *topic*:** El nexo de unión entre publicadores y suscriptores es el *topic*, que se define mediante un nombre y un formato de datos. En este contexto, representa las métricas que se desean transferir entre los nodos. Cada nodo es responsable de publicar sus métricas en un *topic* específico, y los demás nodos pueden suscribirse a este *topic* para recibir actualizaciones correspondientes.

   Los *topics* en MQTT siguen una estructura jerárquica, lo que permite una gran flexibilidad en la suscripción a uno o varios *topics* mediante el uso de comodines (# y +).

   El patrón publicador/suscriptor es ideal cuando la infraestructura de red es desconocida y cambia con frecuencia. El desacoplamiento entre publicadores y suscriptores permite que la incorporación o desconexión de una entidad de comunicación no afecte a las demás. La conectividad intermitente y itinerante se alinea con las ventajas de este patrón.

3. **Publicación de las métricas**: Cada nodo recopila sus métricas locales, como el número de nodos en la zona de cobertura (C1), la intensidad de señal recibida (C2), el nivel de batería (C3) y el ancho de banda (C4). Estas métricas se publican en el *topic* con su identificador correspondiente y fecha, lo que facilita la identificación y el seguimiento de las métricas a lo largo del tiempo.

4. **Suscripción y recepción de las métricas:** Los demás nodos interesados en las métricas de un nodo específico se suscriben a ese *topic*. Al hacerlo, estos nodos esperan las actualizaciones de las métricas que el nodo publicador envía a ese *topic*. Cada vez que se publique una nueva métrica en el topic, los nodos suscritos recibirán la información actualizada.

5. **Procesamiento y análisis de las métricas:** Una vez que los nodos receptores reciben las métricas, pueden procesar y analizar los datos para tomar decisiones o realizar acciones basadas en esa información. Por ejemplo, el GO utilizará esta información para calcular la puntuación del Valor de Importancia de cada nodo mediante una fórmula predefinida que tenga en cuenta las métricas recibidas. Estos cálculos permiten evaluar el estado de los nodos y tomar decisiones inteligentes basadas en los resultados.

   Véase a continuación (Fig. 6-3), un ejemplo de posible escenario en el que se tienen tres nodos conectados a un bróker MQTT.

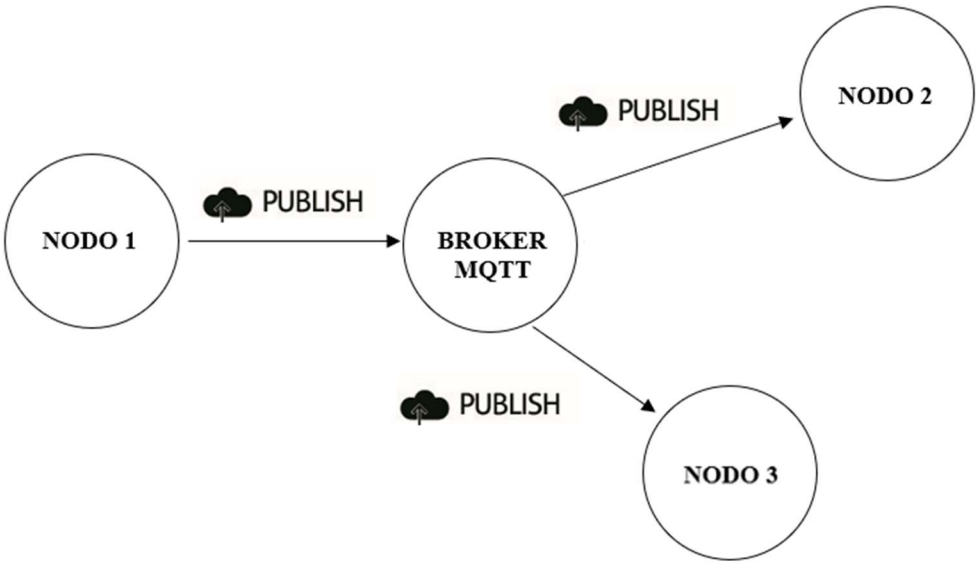


Figura 6-3: Esquema transferencia de datos con protocolo MQTT

El nodo 1 es el que actúa como publicador y envía mensajes a un *topic* específico en el bróker MQTT para que otros nodos lo reciban.

Los nodos 2 y 3 actúan como suscriptores. Se suscriben al mismo tema en el que el Nodo 1 está publicando mensajes. Cuando se publica un mensaje en ese tema, estos nodos lo recibirán y podrán procesarlo.

A continuación, se muestra cómo funciona esta comunicación en este escenario simple:

A) Configuración Inicial:

- Nodo 1: Publica mensajes en el tema "metrics".
- Nodo 2 y Nodo 3: Se suscriben al tema "metrics".

B) Acción:

- Nodo 1 procesa internamente información y decide enviarla a través de MQTT.
- Nodo 1 publica un mensaje en el tema "metrics" con sus correspondientes métricas.

C) Resultado:

- El bróker MQTT recibe el mensaje de Nodo 1 y verifica qué nodos están suscritos al tema "metrics".
- El bróker envía una copia del mensaje a Nodo 2 y otra copia a Nodo 3, ya que ambos están suscritos a ese tema.
- Nodo 2 y Nodo 3 reciben el mensaje con el valor de las métricas.

Cada uno de los nodos suscriptores puede procesar la información según sea necesario.

D) Conclusión:

- Los nodos 2 y 3 han recibido y procesado con éxito la información publicada por Nodo 1 a través del bróker MQTT.

Esto permite una comunicación eficiente y escalable entre nodos en un sistema distribuido.

La combinación de MQTT y WiFi Direct ofrece una solución efectiva para la transferencia de métricas en entornos donde se requiere una comunicación directa entre dispositivos y se busca minimizar el consumo de recursos de red. Al utilizar MQTT, los nodos pueden intercambiar información de manera eficiente y confiable, lo que facilita el análisis y la toma de decisiones basadas en datos en tiempo real.

6.4 Explorando la implementación en la comunicación entre dispositivos

Esta sección detalla el uso de las herramientas como Mosquitto y MQTTX para comprender y experimentar con los conceptos fundamentales de MQTT, como la publicación y suscripción a *topics*, la estructura de mensajes y cómo se establece la comunicación bidireccional entre dispositivos.

Segunda simulación:

Antes de comenzar a interactuar con los mensajes MQTT, se debe establecer una conexión con el bróker MQTT. En este caso, iniciando la aplicación de Mosquitto, nos proporcionará la funcionalidad directa de un bróker MQTT. Su configuración inicial y puesta en marcha brindan una base esencial para el funcionamiento de la red de comunicación basada en MQTT.

Lo siguiente que se debe hacer es crear un grupo. En este caso, se ha definido como "WiFi Direct" (Véase la Fig. 6-4), y este grupo simulará la creación del grupo WiFi Direct con sus respectivos nodos. En términos prácticos, este grupo representa una red de dispositivos conectados a través de WiFi Direct que participan en la transferencia de métricas. La creación de grupos permite organizar los dispositivos de manera lógica y eficiente, lo que simplifica la administración de la comunicación entre nodos y mejora la escalabilidad del sistema.

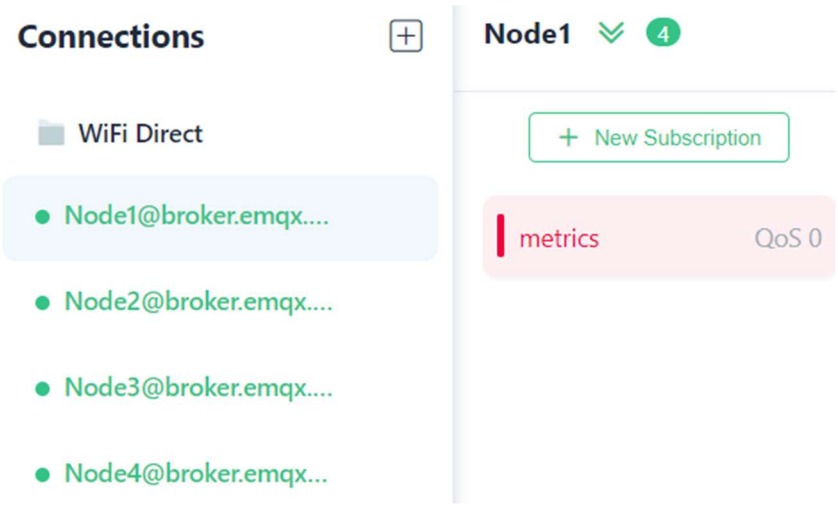


Figura 6-4: Establecimiento de conexiones y suscripción del topic principal

Además, se pueden añadir tantas suscripciones como sean necesarias. Una suscripción define qué *topics* específicos está interesado en seguir un nodo en particular. Por ejemplo, si un nodo está interesado en recibir información sobre métricas de otros nodos, puede suscribirse al *topic* de métricas para estar al tanto de las actualizaciones. Esta capacidad de selección de suscripciones brinda flexibilidad y adaptabilidad al sistema, ya que cada nodo puede personalizar los datos que desea recibir según sus necesidades y funciones.

Cada vez que un nodo publica sus métricas en el *topic* (véase las Fig. 6-5 y 6-6), los nodos suscritos a ese *topic* recibirán la información actualizada. Esto refleja una situación típica en la transferencia de datos, donde múltiples dispositivos colaboran para compartir información relevante y mantenerse al tanto del estado de la red y los nodos individuales.

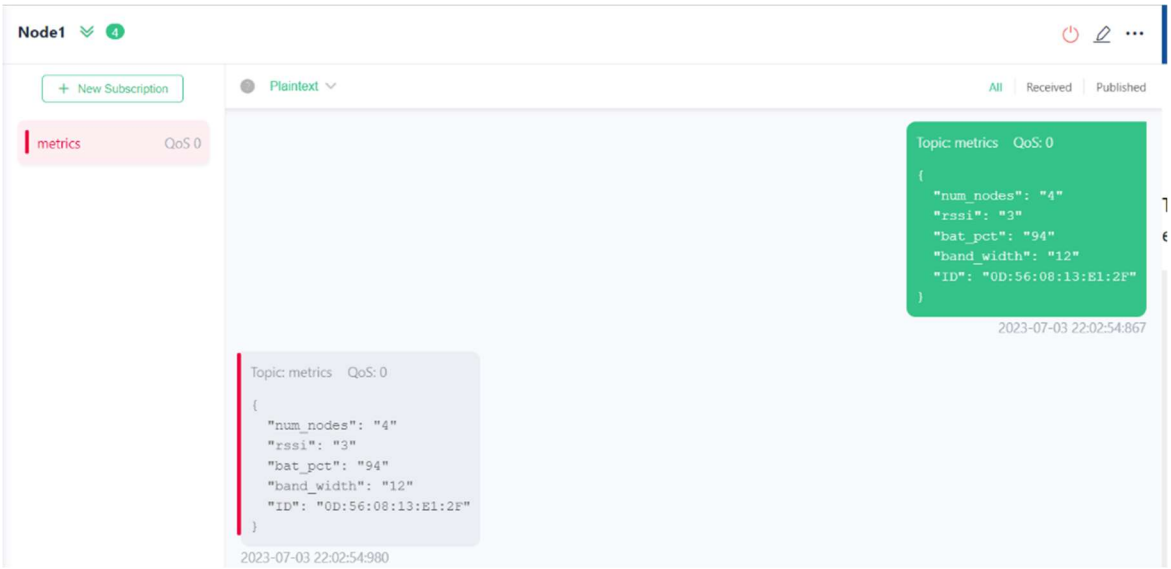


Figura 6-5: Publicación de valores por parte del Nodo 1

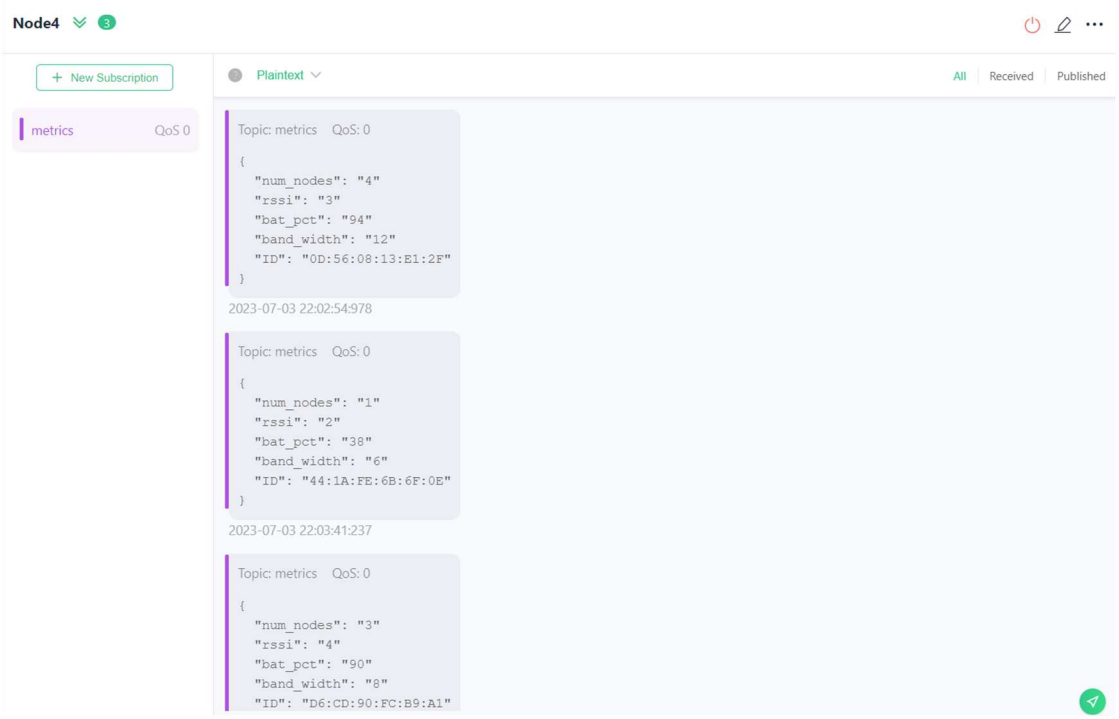


Figura 6-6: Suscripción que muestra valores publicados por otros nodos en el Nodo 4

Por razones de simplicidad, los datos han sido enviados directamente en formato JSON. Sin embargo, en una perspectiva futura, resulta interesante explorar la posibilidad de transmitir los datos en el *topic* en forma de texto ordinario, incorporando caracteres alfanuméricos para ser posteriormente analizados y convertidos en formato JSON antes de su publicación en el broker.

El objeto JSON que ha sido estructurado incluye los siguientes campos:
- num_nodes: Representa la cantidad de nodos en el grupo.
- rssi: Indica el nivel de intensidad de señal.
- bat_pct: Refleja el porcentaje de batería disponible en el dispositivo.
- band_width: Cuantifica el ancho de banda medido en Mbps.
- ID: Hace referencia a la dirección física del terminal que emite los datos.

Como una característica adicional de la aplicación, cada mensaje es acompañado por un sello temporal que refleja el momento preciso en que los datos son enviados o recibidos. Mediante la continua experimentación con estas herramientas, se logra apreciar la capacidad intrínseca que ofrecen y cómo posibilitan el establecimiento de una comunicación eficaz, bidireccional y sumamente adaptable entre dispositivos dispersos en una diversidad de contextos.

Al igual que cada nodo contribuye con sus propias métricas, las cuales son compartidas con otros nodos suscritos, el GO actual, respaldado por esta información, asume la responsabilidad de ejecutar internamente el algoritmo de consenso anticipado y recurrente. Este procedimiento culmina en la publicación del nodo con la puntuación más alta en el *topic* adecuado, otorgándole el título de "Backup GO". Esta designación se materializa en situaciones donde el nodo principal falla o se producen cambios en la red y su itinerancia.

Por último, el reto final consistirá en fusionar ambas implementaciones para alcanzar el ansiado algoritmo de consenso anticipado, enraizado en el concepto de selección de líder, tal como se aplica en el algoritmo Raft. Esta combinación se fusionará con la tecnología WiFi Direct, la cual se respalda en protocolos M2M para garantizar una transferencia de datos efectiva y eficiente. Esta suma de conceptos y tecnologías culmina en un enfoque integral que promete resolver el desafío principal de este Trabajo.

6.5 Implementación del algoritmo con la transmisión de mensajes utilizando MQTT

El presente apartado se centra en una posible implementación del algoritmo de consenso basado en elección de líder como sucedía con Raft combinado con la transmisión de mensajes utilizando el protocolo MQTT para posteriormente ser utilizado en un sistema de comunicaciones descentralizado como pudiera ser un grupo de WiFi Direct.

Para comprender plenamente esta implementación, es esencial recordar el objetivo fundamental de este algoritmo de consenso y su trascendencia en el contexto de la comunicación en redes de nodos descentralizadas.

Este algoritmo tiene como objetivo principal lograr un novedoso enfoque de consenso anticipado y periódico, diseñado para facilitar la selección eficaz del punto de acceso óptimo en situaciones donde se presenten desconexiones del nodo principal y se experimenten cambios en la estructura de la red.

Este enfoque culminaría con su implementación en entornos de grupos de dispositivos WiFi Direct, donde se logra una transferencia de datos altamente eficiente mediante el empleo de protocolos M2M

6.5.1 Diseño de la implementación MQTT

Elección de Biblioteca MQTT para Java:

Para la implementación MQTT en Java, se ha optado por utilizar la biblioteca Eclipse Paho, que proporciona una serie de herramientas para trabajar con MQTT. Esta elección se basa en la robustez y la amplia comunidad de desarrollo que respalda esta biblioteca. Eclipse Paho ofrece una implementación Java de cliente MQTT, lo que facilita la conexión y la comunicación con el bróker MQTT.

Eclipse Paho es ampliamente reconocido por su confiabilidad y flexibilidad. La elección de una biblioteca estable y bien mantenida es fundamental en cualquier implementación de MQTT, ya que garantiza un funcionamiento eficiente y confiable del sistema. Además, la comunidad activa de desarrollo de Eclipse Paho brinda acceso a actualizaciones y mejoras continuas, lo que es esencial para mantener la robustez de la comunicación MQTT en el tiempo.

Configuración del Bróker MQTT:

Para este sistema de comunicaciones, se configura un bróker MQTT que actúa como servidor centralizado para la comunicación entre nodos. El bróker se encarga de enrutar los mensajes de los publicadores a los suscriptores adecuados. La elección del bróker MQTT depende de los requisitos específicos del proyecto, y en este caso, se utiliza Mosquitto, configurado un bróker local en el puerto 1883 para facilitar la comunicación dentro de la red de nodos.

La elección de Mosquitto como bróker MQTT se basa en su simplicidad y versatilidad. Mosquitto es una implementación de código abierto ampliamente utilizada que ofrece un alto rendimiento y es adecuada para aplicaciones tanto pequeñas como grandes. Al configurar un bróker local, se asegura un control total sobre la infraestructura de comunicación, lo que puede ser crucial para aplicaciones críticas.

Tercera simulación

El código simula un proceso de consenso entre varios nodos y selecciona un nodo de respaldo. A continuación, se presenta una descripción general de la estructura y el flujo del programa:

1. **Entrada de usuario:**

   El programa comienza solicitando al usuario la entrada de información clave:
   - El número de nodos a simular.
   - El perfil común para todos los nodos.
   - Si se desea dar prioridad a algún nodo.

2. **Creación y ejecución de hilos:**

   Se crean hilos para cada nodo de acuerdo con el número especificado por el usuario. Cada nodo se configura con su perfil, prioridad y otros datos de interés. Finalmente, los nodos se agregan a la lista de nodos para ser procesados más adelante.

3. **Configuración MQTT**

   Primero se crea una instancia del objeto MqttSubscriber. Este objeto se utiliza para suscribirse a un tema MQTT y recibir mensajes publicados en ese tema.

   Se llama al método connect(). Este método establece una conexión MQTT con un broker, responsable de enrutar los mensajes entre los publicadores y los suscriptores.

```
public void connect() {
    try {
        // Generar un identificador único para la conexión
        String clientId = "client-" + UUID.randomUUID().toString();

        // Crear conexión MQTT
        client = new MqttClient("tcp://localhost:1883", clientId, new MemoryPersistence());
        MqttConnectOptions connOpts = new MqttConnectOptions();
        connOpts.setCleanSession(true);
        client.connect(connOpts);

        System.out.println("Mqtt Connected");

    } catch (MqttException me) {
        // Manejo de excepciones al conectar
        System.err.println("Error al conectar al servidor MQTT: " + me.getMessage());
        me.printStackTrace();
    }
}
```

Luego, se llama al método de suscripción. Este método indica al suscriptor que está interesado en recibir mensajes publicados en el tema que le interesa.

```
public void subscribe(String topic) {
    try {
        // Suscripción al tema MQTT
        client.setCallback(this);
        client.subscribe(topic);
        System.out.println("Suscrito a " + "'" + topic + "'");
    } catch (MqttException me) {
        // Manejo de excepciones al suscribirse
        System.err.println("Error al suscribirse al tema MQTT: " + me.getMessage());
        me.printStackTrace();
    }
}
```

En esta implementación, se ha adoptado una estructura de doble topic (Véase Fig. 6-7). Esta elección se basa en la necesidad de lidiar con varios tipos de mensajes de manera eficiente y organizada. Al tener dos temas distintos, uno para las métricas generales y otro para la puntuación de los nodos, logramos una separación clara de las funciones y datos transmitidos. El tema de las métricas, denominado "metrics," se utiliza para la recopilación y difusión de información relacionada con el estado y el rendimiento de los nodos. Por otro lado, el tema de la puntuación, llamado "backupGO," se emplea exclusivamente para la comunicación de las puntuaciones calculadas por cada nodo junto con datos identificativos.

Esta estructura de doble topic proporciona una organización eficaz de los mensajes MQTT. Al separar claramente las métricas y la puntuación en dos temas distintos, se simplifica el proceso de suscripción y publicación para los nodos. Esto significa que los nodos pueden suscribirse a los temas que les interesen y recibir solo los mensajes relevantes, lo que reduce la carga de procesamiento y mejora la eficiencia general del sistema.

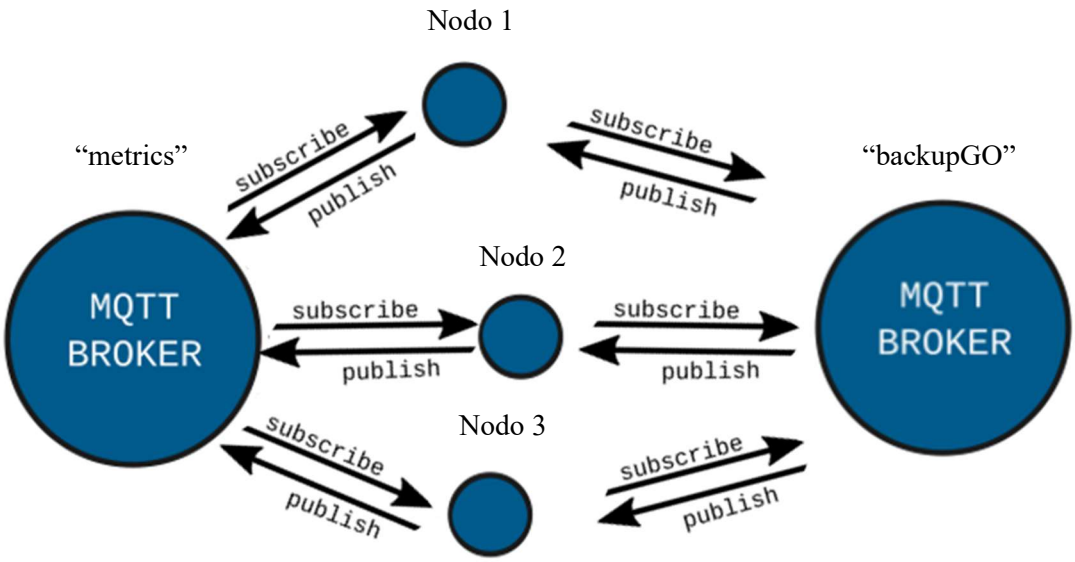


Figura 6-7: Estructura del doble topic

4. **Generación de valores aleatorios**

Después de la configuración MQTT, se generan valores aleatorios para las diversas métricas.

A continuación, se crea un objeto JSON llamado metricsInfo utilizando la biblioteca Gson. Este objeto JSON contiene las métricas generadas y otros datos, como el tipo de mensaje (msgType), la prioridad (priority), el número de nodo (nodeNumber), la dirección MAC (macAddress) y la fecha (date).

```
// Crear objeto JSON con las métricas
JsonObject metricsInfo = new JsonObject();
metricsInfo.addProperty("msgType", 1);
metricsInfo.addProperty("priority", priority);
metricsInfo.addProperty("nodeNumber", nodeNumber);
metricsInfo.addProperty("numNodes", numNodes);
metricsInfo.addProperty("rssi", rssi);
metricsInfo.addProperty("batPCT", batPCT);
metricsInfo.addProperty("bandwidth", bandwidth);
metricsInfo.addProperty("macAddress", mac);
metricsInfo.addProperty("date", (new Date().toString()));
```

5. **Publicación de métricas**

Se llama al método publishMetrics(). Este método se utiliza para publicar el objeto JSON metricsInfo en el *topic* MQTT "metrics". Se consigue así enviar las métricas generadas al servidor MQTT, donde otros nodos que estén suscritos a este tema podrán recibir y procesar estas métricas.

```
public class MqttTransfer {

    public static void publishMetrics(String broker, String clientId, String topic, JsonObject jsonMetrics) throws MqttException {
        try {
            MqttClient client = new MqttClient(broker, clientId);
            client.connect();

            MqttMessage message = new MqttMessage(jsonMetrics.toString().getBytes());
            client.publish(topic, message);

            client.disconnect();
        } catch (MqttException e) {
            throw e;
        }
    }
```

Véase un ejemplo de ejecución de cómo se reciben las métricas:

```
Mensaje recibido - Tipo: 1
Tema: 'metrics'
Mensaje:
{"msgType":1,"priority":0,"nodeNumber":1,"numNodes":3,"rssi":4,"batPCT":
62.0,"bandwidth":66,"macAddress":"74:0a:b7:50:4b:cc","date":"Tue  Sep  05
22:02:48 CEST 2023"}
Nodo 1 - Métricas publicadas en el topic 'metrics'
```

6. **Cálculo de puntuación**

El cálculo de puntuación es un paso crucial en el proceso de selección del nodo de respaldo. Aquí, se utiliza el Valor de Interés, calculado por el algoritmo, para determinar la puntuación de un nodo en función de varios parámetros, como el número de nodos, la señal RSSI, el nivel de batería y el ancho de banda.

Una vez calculada la puntuación, se redondea para mayor claridad y se empaqueta en un mensaje JSON. Este mensaje contiene a su vez información importante, como el número de nodo, la puntuación y la dirección MAC. Luego, se publica en el tema "backupGO" a través de MQTT para que otros nodos puedan acceder a él.

```
Mensaje recibido - Tipo: 2
Tema: 'backupGO'
Mensaje:
{"msgType":2,"nodeNumber":1,"score":21.6,"macAddress":"74:0a:b7:50:4b:cc
","date":"Tue Sep 05 22:02:48 CEST 2023"}
```

Este proceso de cálculo y publicación es fundamental para el proceso de selección del nodo de respaldo, ya que proporciona a los nodos la información necesaria para tomar decisiones informadas en caso de que el nodo principal falle.

7. **Proceso de consenso**

Una vez que todos los nodos han recopilado y publicado sus métricas, se realiza el proceso de consenso utilizando el objeto ConsensusManager. Este objeto se encarga de administrar y evaluar la información de los nodos procesados, calcular puntuaciones y determinar el nodo de respaldo. Esta acción se realizará periódicamente para garantizar la disponibilidad continua de un nodo de respaldo en caso de fallo del nodo principal.

8. **Resultados:**

Por último, se imprimen las puntuaciones normalizadas de los nodos simulando qué nodo sería el seleccionado en caso de caída del nodo principal. Esta información proporciona una visión clara de cómo se tomarían decisiones de respaldo en función de las puntuaciones calculadas.

```
Puntuaciones normalizadas:
Nodo 1: 0.78
Nodo 2: 0.55
Nodo 3: 0.0
Nodo 4: 0.6
Nodo 5: 0.92
Nodo 6: 0.28
Nodo 7: 1.0
Nodo 8: 0.38
Nodo 9: 0.05
Nodo 10: 0.13
Backup GO: Nodo 7
```

Diagramas de Flujo y UML:

A fin de comprender mejor la estructura de esta implementación, se ha elaborado una serie de diagramas.

En primer lugar, se presenta un diagrama de flujo (véase Fig. 6-8) de la ejecución donde se sigue la siguiente dinámica:

En el inicio del programa, se inicializan variables como el número de nodos, el perfil y si algún de ellos tiene prioridad. A continuación, se establece una conexión MQTT para la comunicación entre nodos, suscribiéndose a los temas "metrics" y "backupGO." Posteriormente, se generan valores aleatorios que representan los valores de dichas métricas. Estos datos se utilizan para crear un objeto JSON que se publica en el tema "metrics" a través de MQTT, permitiendo que otros nodos reciban y procesen estas métricas.

Luego, se realiza un cálculo de la puntuación basado en el perfil, las métricas y la prioridad, y se crea un objeto JSON con la puntuación, que se publica en el tema "backupGO." Finalmente, con los datos de la puntuación si hubiera una caída del nodo principal, tendríamos determinado cual es el nodo que según las características del entorno cumpliría mejor la función de GO.

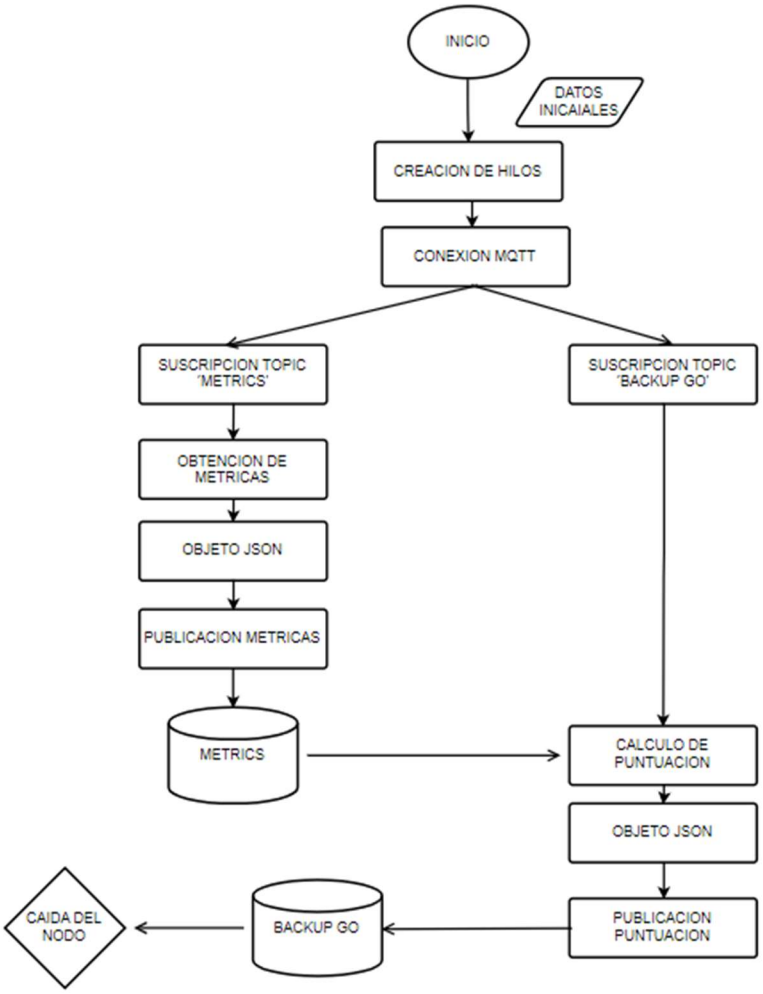


Figura 6-8: Diagrama de flujo de la implementación

Por otro lado, tenemos un diagrama de UML (véase Fig- 6-9) donde se puede ver como funciona el tratamiento de mensajes MQTT en la ejecución de dicha implementación.

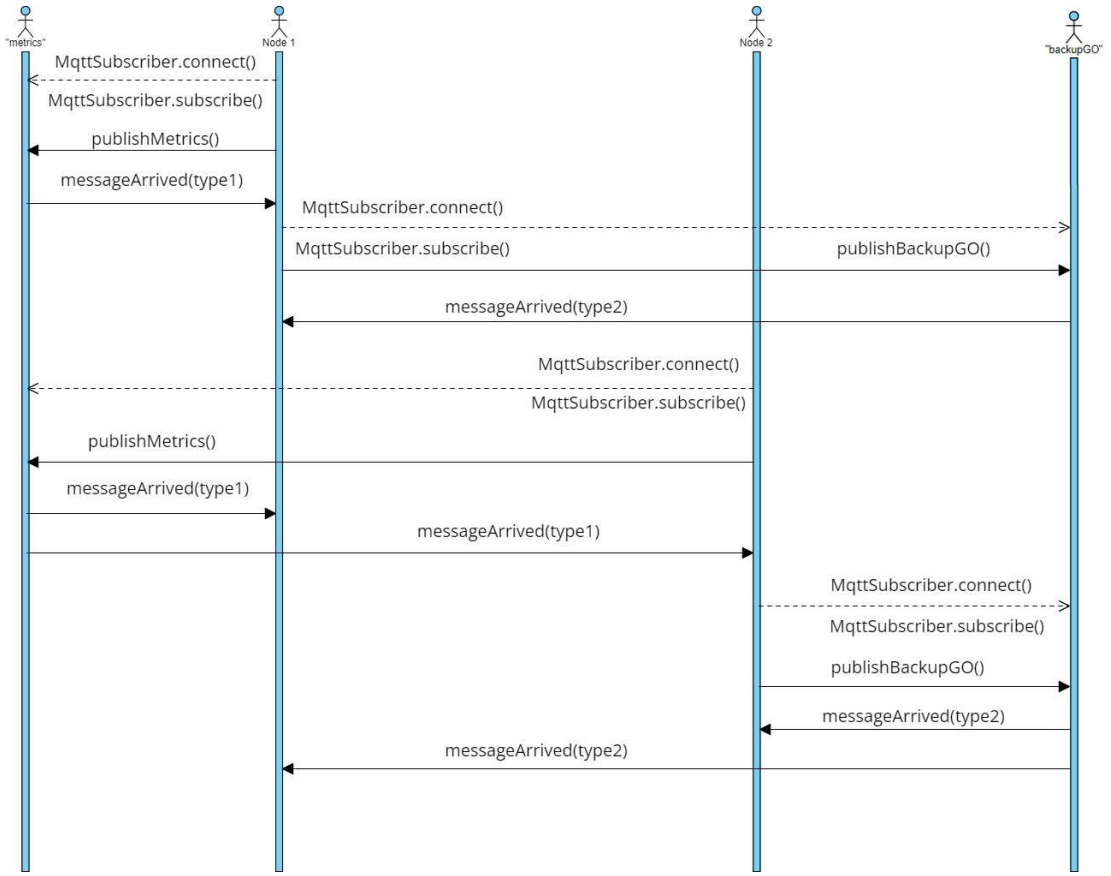


Fig 6-9: Transmisión de mensajes MQTT

Estos diagramas ayudan a visualizar cómo los nodos publican y suscriben métricas y puntuaciones en los temas MQTT, así como cómo se gestionan y procesan estos datos en el contexto del algoritmo de consenso.

6.6 Discusión

Personalmente, el autor del presente Trabajo tenía cierta reticencia sobre el uso de la aleatoriedad como parte fundamental de un algoritmo de consenso. Aunque la aleatoriedad puede ser útil en ciertos contextos, se consideró que sería más interesante incluir otros criterios que pudieran mejorar el rendimiento del algoritmo Raft y alcanzar un consenso más eficiente.

En lugar de depender de factores aleatorios, se ha optado por examinar criterios basados en el rendimiento del sistema, la disponibilidad de recursos o incluso características específicas de los nodos. Al centrarse en criterios más deterministas y predictivos, se ha considerado que se podría lograr una mayor estabilidad y confiabilidad en el algoritmo, lo que a su vez contribuiría a una mejor tolerancia a fallos y un consenso más rápido y consistente.

Al utilizar un polinomio ponderado y considerar diferentes propiedades con pesos asignados, se logra una mayor influencia de los nodos más importantes en el resultado final del consenso. Esto permite adaptarse a diversos escenarios y necesidades, mejorando la eficiencia y la precisión de las decisiones tomadas en el sistema.

Por otra parte, la capacidad de asignar pesos a diferentes propiedades y considerar la influencia de nodos específicos proporciona una versatilidad excepcional en el algoritmo de consenso. Los pesos pueden ajustarse y sintonizarse según las necesidades y requisitos particulares de cada aplicación.

Por ejemplo, en un entorno donde ciertos nodos poseen roles críticos o características especiales, es posible otorgarles una ponderación mayor para asegurar que su voz tenga un impacto significativo en el consenso. Esta personalización se traduce en una adaptabilidad sin igual a las dinámicas cambiantes de diversas aplicaciones.

Además, la versatilidad del algoritmo no se limita únicamente a la asignación de pesos. También puede ajustarse para considerar una variedad de propiedades y métricas específicas de cada entorno. Dentro de las posibilidades estudiadas para seguir mejorando el algoritmo resulta importante destacar las siguientes:

1. Pérdida de paquetes: Suele deberse a la congestión en la red. Al evaluar y controlar esta pérdida puede garantizar la integridad de la comunicación y la entrega de datos sin errores.

2. La superposición de canales: Ocurre cuando múltiples dispositivos utilizan las mismas frecuencias de radio o canales para la comunicación. Esto puede generar interferencias y degradar la calidad de la señal. Controlar este aspecto puede evitar conflictos de comunicación y garantizar un rendimiento óptimo.

3. Latencia: En el contexto de Wi-Fi Direct, la latencia sigue siendo una métrica crítica, especialmente cuando se trata de aplicaciones en tiempo real, como la transmisión de voz o video. Una latencia alta puede dar lugar a retrasos perceptibles, lo que afectaría negativamente la experiencia del usuario

4. Jitter: Se refiere a la variación en el retardo entre la llegada de paquetes de datos. Un jitter alto puede afectar la calidad de la comunicación en tiempo real y causar problemas de sincronización.

5. SNR (Relación Señal-Ruido): Aquí se compara la potencia de la señal deseada con la potencia del ruido en un canal de comunicación. Una alta SNR indica una señal fuerte y una calidad de comunicación superior, mientras que una baja SNR puede resultar en errores de transmisión.

6. Latitud y longitud: La ubicación geográfica de un dispositivo puede ser relevante en aplicaciones de seguimiento o geolocalización, donde es importante conocer la ubicación precisa de los nodos.

7. Movilidad: Evaluar la movilidad es fundamental en sistemas que involucran dispositivos móviles, como vehículos autónomos o dispositivos IoT portátiles. Donde también entraría la velocidad máxima.

8. Predictibilidad: Este criterio se refiere a la capacidad del sistema para mantener un rendimiento constante y predecible a lo largo del tiempo. Tener analizado esto puede garantizar un funcionamiento confiable y consistente del sistema durante un tiempo prolongado.

Todas estas métricas son interesantes al evaluar una mejora en el algoritmo. Algunas de estas métricas pueden ser más relevantes dependiendo de las aplicaciones específicas y los escenarios de uso. La capacidad de adaptar el algoritmo para considerar estas métricas demuestra su versatilidad y su capacidad para abordar una amplia variedad de situaciones.

Por ejemplo, en una aplicación de IoT en el contexto de la salud (Internet of Medical Things, IoMT), es posible priorizar métricas relacionadas con la precisión de los datos recopilados, mientras que, en una red de sensores industriales, se pueden enfocar en la eficiencia energética. Esta capacidad de personalización garantiza que el algoritmo pueda adaptarse a las particularidades de cada aplicación, lo que lo convierte en una solución escalable y versátil.

Cabe destacar que asignar prioridad a un nodo basándose únicamente en su puntuación más baja puede parecer contradictorio desde una perspectiva puramente objetiva. Sin embargo, en situaciones reales, los algoritmos y decisiones de asignación de prioridades pueden tener en cuenta diversos factores y consideraciones para garantizar un rendimiento óptimo y una adecuada resiliencia de la red en casos de fallos o situaciones adversas.

En cuanto a la implementación del algoritmo siguiendo el enfoque de Raft, los resultados obtenidos muestran una mejora sustancial en términos de eficiencia. Este logro es notable, ya que demuestra que la adaptación y optimización del algoritmo son altamente beneficiosas en entornos específicos como WiFi Direct.

Por su parte, la implementación de las comunicaciones bajo el paradigma MQTT destaca por su capacidad para gestionar eficazmente la comunicación entre dispositivos en un entorno inalámbrico, superando las limitaciones de los enfoques convencionales. Si bien MQTT es una opción ampliamente utilizada en sistemas distribuidos y ofrece ventajas en términos de simplicidad y eficiencia en entornos de red estándar, existen otros protocolos como DDS que superan significativamente a MQTT en ciertos aspectos críticos. DDS, por ejemplo, ha demostrado un rendimiento sobresaliente en lo que respecta a la latencia de telemetría, especialmente en condiciones de red deficientes, como alta latencia del sistema, alta tasa de pérdida de paquetes y ancho de banda limitado. DDS destaca particularmente en términos de latencia, lo que lo convierte en una alternativa confiable para aplicaciones en tiempo real. Sin embargo, es importante tener en cuenta que DDS puede tener un mayor consumo de memoria y CPU en comparación con otros protocolos, lo que podría ser relevante en dispositivos altamente limitados utilizados en el Internet de las cosas.

Además, es importante destacar que DDS puede tener un mayor consumo de ancho de banda en comparación con tecnologías centralizadas cuando los publicadores aumentan su frecuencia de muestreo. Sin embargo, este aumento en el consumo de ancho de banda se compensa con un rendimiento superior en términos de latencia y confiabilidad de los datos. Estas características hacen que DDS sea una opción incluso más atractiva.

Una posible limitación del algoritmo es que el sistema no pueda extraer el valor requerido o no reciba el valor esperado. Esto puede ocurrir debido a varios factores, como problemas de conectividad, errores en la fuente de datos o interrupciones en el flujo de información. La falta de un sólido manejo de excepciones y errores en el código podría resultar en un comportamiento impredecible o en la interrupción de procesos críticos.

7 CONCLUSIONES Y LÍNEAS FUTURAS

No sabía cómo hacerlo, así que lo intenté, fallé, fallé, fallé y luego lo hice bien

-Elon Musk-

En este Trabajo se han realizado las siguientes aportaciones:

1. Se han definido distintas métricas que pueden aportar una toma de decisiones más eficaz en entornos con múltiples participantes.

2. Se ha propuesto un algoritmo de consenso novedoso basado en Raft que podría abordar desafíos específicos en entornos dinámicos y mejorar la toma de decisiones en tiempo real.

3. Se ha demostrado la viabilidad de implementar el algoritmo propuesto en un sistema WiFi Direct.

Aunque los resultados que se han presentado son simulados, la implementación mediante MQTT de las comunicaciones necesarias para la transmisión de mensajes, permite que la implementación del algoritmo en un entorno real sea prácticamente directa, y podría abrir una línea de futuro avance de este Trabajo, encaminada a desarrollar nuevos algoritmos que mejoren significativamente el rendimiento de sistemas descentralizados.

El algoritmo propuesto proporciona un enfoque flexible y adaptable para la toma de decisiones consensuadas en un sistema distribuido. La flexibilidad inherente al algoritmo propuesto se erige como uno de sus aspectos más destacados y valiosos. Esta característica permite que el algoritmo se adapte eficazmente a una amplia gama de aplicaciones y escenarios dentro de sistemas distribuidos. La capacidad de personalizar el algoritmo para considerar métricas específicas proporciona una ventaja competitiva sobre enfoques genéricos.

Por otra parte, la integración de las comunicaciones con protocolo MQTT constituye un avance significativo en la optimización de la selección de líderes. La combinación de Raft y MQTT proporciona un marco sólido para abordar los desafíos específicos de WiFi Direct.

Se destaca también la importancia de implementar mecanismos de manejo de errores robustos para garantizar que el sistema pueda recuperarse o tomar medidas adecuadas en caso de fallos inesperados en la obtención o recepción de datos requeridos. Esto contribuirá a una mayor robustez y confiabilidad de tu sistema, incluso en condiciones adversas.

Líneas futuras

La implementación exitosa del algoritmo de consenso en un entorno real no solo representa un hito importante en la investigación actual, sino que también abre un abanico de posibilidades para futuros trabajos de investigación y desarrollo en el campo de las redes distribuidas.

En este contexto, la Raspberry Pi emerge como una herramienta de gran utilidad. Este dispositivo de bajo costo y tamaño reducido se ha convertido en una opción popular para proyectos de electrónica y programación experimental. Cuando se trata de redes WiFi Direct, la Raspberry Pi se presenta como un recurso valioso que permite simular nodos y crear redes distribuidas para poner a prueba y evaluar el rendimiento de algoritmos de consenso en un entorno real.

La flexibilidad que ofrece la Raspberry Pi es clave en esta implementación. Cada nodo simulado puede operar de manera independiente, lo que posibilita la evaluación del comportamiento del algoritmo en diversos escenarios. Además, la Raspberry Pi permite la configuración de parámetros de red personalizables, como la latencia, el ancho de banda y la pérdida de paquetes. Esto facilita la creación de escenarios de red más complejos y la evaluación exhaustiva del rendimiento de los algoritmos de consenso en situaciones realistas.

Sin embargo, es fundamental tener en cuenta que las redes WiFi Direct no están exentas de vulnerabilidades de seguridad. Por lo tanto, otro interesante camino de investigación y desarrollo podría centrarse en la implementación y evaluación de mecanismos de seguridad específicos. Esto incluiría la autenticación de nodos, el cifrado de datos y la detección y prevención de posibles ataques, contribuyendo así a hacer que estas redes sean más seguras y confiables.

En cuanto a otras posibles mejoras, sería esencial explorar la obtención de métricas precisas y sustituir los valores simulados por datos reales siempre que sea posible, considerando la posibilidad de no poder extraer ciertas métricas. Además, el manejo de grupos WiFi Direct podría optimizar aún más el algoritmo, incluyendo la creación automática de grupos cuando se elige un nuevo líder y el despliegue automático del bróker de publicación/suscripción en dicho líder.

8 Bibliografía

[1] «Mobile device subscriptions charted country by country | World Economic Forum». https://www.weforum.org/agenda/2022/10/mobile-device-subscription-rise-technology/

[2] S. Trifunovic, B. Distl, D. Schatzmann, y F. Legendre, «WiFi-Opp: ad-hoc-less opportunistic networking», en Proceedings of the 6th ACM workshop on Challenged networks, en CHANTS '11. New York, NY, USA: Association for Computing Machinery, sep. 2011, pp. 37-42.

[3] C. Berge, The Theory of Graphs. Courier Corporation, 2001.

[4] K. Thulasiraman y M. N. S. Swamy, Graphs: Theory and Algorithms. John Wiley & Sons, 2011.

[5] M. A. Khan, F. Algarni, y M. T. Quasim, «Decentralised Internet of Things», en Decentralised Internet of Things: A Blockchain Perspective, M. A. Khan, M. T. Quasim, F. Algarni, y A. Alharthi, Eds., en Studies in Big Data. Cham: Springer International Publishing, 2020, pp. 3-20.

[6] S. M. Hosseini Bamakan, A. Motavali, y A. Babaei Bondarti, «A survey of blockchain consensus algorithms performance evaluation criteria», Expert Syst. Appl., vol. 154, p. 113385, abr. 2020.

[7] P. Baran, «Memorandum on distributed communications: I. Introduction to distributed communications networks». Rand Coorporation, 1964.

[8] J. Wu, Distributed System Design. CRC Press, 1998.

[9] P. Baran, «On Distributed Communications Networks», IEEE Trans. Commun. Syst., vol. 12, n.o 1, pp. 1-9, mar. 1964.

[10] R. Bhardwaj y D. Datta, «Consensus Algorithm», en Decentralised Internet of Things: A Blockchain Perspective, M. A. Khan, M. T. Quasim, F. Algarni, y A. Alharthi, Eds., en Studies in Big Data. Cham: Springer International Publishing, 2020, pp. 91-107.

[11] L. Lamport, «Time, clocks, and the ordering of events in a distributed system», en Concurrency: the Works of Leslie Lamport, New York, NY, USA: Association for Computing Machinery, 2019, pp. 179-196.

[12] L. Lamport, «The part-time parliament», en Concurrency: the Works of Leslie Lamport, New York, NY, USA: Association for Computing Machinery, 2019, pp. 277-317.

[13] W.-C. Shi y J.-P. Li, «Research on consistency of distributed system based on Paxos algorithm», en 2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP), dic. 2012, pp. 257-259.

[14] R. Van Renesse y D. Altinbuken, «Paxos Made Moderately Complex», ACM Comput. Surv., vol. 47, n.o 3, pp. 1-36, abr. 2015.

[15] W. J. Bolosky, D. P. Bradshaw, R. B. Haagens, N. P. Kusters, y P. Li, «Paxos replicated state machines as the basis of a high-performance data store», mar. 2011.

[16] H. Ng, S. Haridi, y P. Carbone, «Omni-Paxos: Breaking the Barriers of Partial Connectivity», en Proceedings of the Eighteenth European Conference on Computer Systems, Rome Italy: ACM, may 2023, pp. 314-330.

[17] J. Aspnes, «Notes on Theory of Distributed Systems», Yale Univ. Dep. Comput. Sci., pp. 90-96, 2022.

[18] F. Hupfeld et al., «FaTLease: scalable fault-tolerant lease negotiation with Paxos», Clust. Comput., vol. 12, n.o 2, pp. 175-188, jun. 2009.

[19] R. Van Renesse y D. Altinbuken, «Paxos Made Moderately Complex», ACM Comput. Surv., vol. 47, n.o 3, p. 42:1-42:36, feb. 2015.

[20] A. Lakshman y P. Malik, «Cassandra: structured storage system on a P2P network», en Proceedings of the 28th ACM symposium on Principles of distributed computing, en PODC '09. New York, NY, USA: Association for Computing Machinery, ago. 2009, p. 5.

[21] V. Poirot, B. Al Nahas, y O. Landsiedel, «Paxos Made Wireless: Consensus in the Air», en Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks, en EWSN '19. USA: Junction Publishing, mar. 2019, pp. 1-12.

[22] L. Lamport, «Fast Paxos», Distrib. Comput., vol. 19, pp. 79-103, oct. 2006.

[23] M. A. Kuppe, L. Lamport, y D. Ricketts, «The TLA+ Toolbox», Electron. Proc. Theor. Comput. Sci., vol. 310, pp. 50-62, dic. 2019.

[24] M. Yabandeh, N. Knežević, D. Kostić, y V. Kuncak, «Predicting and preventing inconsistencies in deployed distributed systems», ACM Trans. Comput. Syst., vol. 28, n.o 1, p. 2:1-2:49, ago. 2010.

[25] C. E. Killian, J. W. Anderson, R. Braud, R. Jhala, y A. M. Vahdat, «Mace: language support for building distributed systems», ACM SIGPLAN Not., vol. 42, n.o 6, pp. 179-188, jun. 2007.

[26] G. Delzanno, M. Tatarek, y R. Traverso, «Model Checking Paxos in Spin», Electron. Proc. Theor. Comput. Sci., vol. 161, pp. 131-146, ago. 2014.

[27] K. Havelund, J. Penix, y W. Visser, SPIN Model Checking and Software Verification: 7th International SPIN Workshop Stanford, CA, USA, August 30 - September 1, 2000 Proceedings. Springer, 2006.

[28] D. Ongaro y J. Ousterhout, «In Search of an Understandable Consensus Algorithm», en 2014 USENIX Annual Technical Conference (USENIX ATC 14), 2014, pp. 305-319.

[29] J. Hu y K. Liu, «Raft consensus mechanism and the applications», J. Phys. Conf. Ser., vol. 1544, p. 012079, may 2020.

[30] D. Huang, X. Ma, y S. Zhang, «Performance Analysis of the Raft Consensus Algorithm for Private Blockchains», IEEE Trans. Syst. Man Cybern. Syst., vol. 50, n.o 1, pp. 172-181, ene. 2020.

[31] H. Howard y R. Mortier, «Paxos vs Raft: Have we reached consensus on distributed consensus?», en Proceedings of the 7thWorkshop on Principles and Practice of Consistency for Distributed Data, abr. 2020, pp. 1-9.

[32] S. Zhou y S. Mu, «Fault-Tolerant Replication with Pull-Based Consensus in MongoDB», Proc. 18th USENIX Symp. Networked Syst. Des. Implement. April 12–14 2021.

[33] H. Howard, «ARC: Analysis of Raft Consensus», University of Cambridge, Computer Laboratory, UCAM-CL-TR-857, 2014.

[34] S. McNew, «Council Post: A 2020 And Post-Pandemic Outlook For Cryptocurrency And Blockchain Industries», Forbes. https://www.forbes.com/sites/forbesbusinessdevelopmentcouncil/2020/05/21/a-2020-and-post-pandemic-outlook-for-cryptocurrency-and-blockchain-industries/

[35] W. Fu, X. Wei, y S. Tong, «An Improved Blockchain Consensus Algorithm Based on Raft», Arab. J. Sci. Eng., vol. 46, n.o 9, pp. 8137-8149, sep. 2021.

[36] T. Liu, Y. Yuan, y Z. Yu, «The service architecture of Internet of things terminal connection based on blockchain technology», J. Supercomput., vol. 77, n.o 11, pp. 12690-12710, nov. 2021.

[37] R. Matzutt, B. Kalde, J. Pennekamp, A. Drichel, M. Henze, y K. Wehrle, «CoinPrune: Shrinking Bitcoin's Blockchain Retrospectively», IEEE Trans. Netw. Serv. Manag., vol. 18, n.o 3, pp. 3064-3078, sep. 2021.

[38] H. Sheikh, R. M. Azmathullah, y F. Rizwan, «Proof-of-Work Vs Proof-of-Stake: A Comparative Analysis and an Approach to Blockchain Consensus Mechanism», Int. J. Res. Appl. Sci. Eng. Technol. IJRASET, vol. 6, 2018.

[39] C. Cachin y M. Vukolic, «Blockchain consensus protocols in the wild», presentado en International Symposium on Distributed Computing, Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, oct. 2017.

[40] M. Xie, J. Liu, S. Chen, y M. Lin, «A survey on blockchain consensus mechanism: research overview, current advances and future directions», Int. J. Intell. Comput. Cybern., vol. 16, n.o 2, pp. 314-340, ene. 2022.

[41] S. Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System», Cryptogr. Mail. List Httpsmetzdowdcom, mar. 2009.

[42] Y. Xu, X. Yang, J. Zhang, J. Zhu, M. Sun, y B. Chen, «Proof of Engagement: A Flexible Blockchain Consensus Mechanism», Wirel. Commun. Mob. Comput., vol. 2021, p. e6185910, ago. 2021.

[43] L. Lamport, «The Weak Byzantine Generals Problem», J. ACM, vol. 30, n.o 3, pp. 668-676, jul. 1983.

[44] C. Attiya, D. Dolev, y J. Gil, «Asynchronous Byzantine consensus», en Proceedings of the third annual ACM symposium on Principles of distributed computing, en PODC '84. New York, NY, USA: Association for Computing Machinery, ago. 1984, pp. 119-133.

[45] H. Moniz, The Istanbul BFT Consensus Algorithm. 2020.

[46] R. Saltini y D. Hyland-Wood, «Correctness Analysis of IBFT», ene. 2019.

[47] K. Lei, M. Du, J. Huang, y T. Jin, «Groupchain: Towards a Scalable Public Blockchain in Fog Computing of IoT Services Computing», IEEE Trans. Serv. Comput., vol. 13, n.o 2, pp. 252-262, mar. 2020.

[48] U. Javaid y B. Sikdar, «A Checkpoint Enabled Scalable Blockchain Architecture for Industrial Internet of Things», IEEE Trans. Ind. Inform., vol. 17, n.o 11, pp. 7679-7687, nov. 2021.

[49] J. Yang, A. Paudel, y H. B. Gooi, «Compensation for Power Loss by a Proof-of-Stake Consortium Blockchain Microgrid», IEEE Trans. Ind. Inform., vol. 17, n.o 5, pp. 3253-3262, may 2021.

[50] S. Wang, D. Ye, X. Huang, R. Yu, Y. Wang, y Y. Zhang, «Consortium Blockchain for Secure Resource Sharing in Vehicular Edge Computing: A Contract-Based Approach», IEEE Trans. Netw. Sci. Eng., vol. 8, n.o 2, pp. 1189-1201, abr. 2021.

[51] M. J. Fischer, N. A. Lynch, y M. S. Paterson, «Impossibility of distributed consensus with one faulty process», J. ACM, vol. 32, n.o 2, pp. 374-382, abr. 1985.

[52] T. D. Chandra y S. Toueg, «Unreliable failure detectors for reliable distributed systems», J. ACM, vol. 43, n.o 2, pp. 225-267, mar. 1996.

[53] C. Nie y H. Leung, «A survey of combinatorial testing», ACM Comput. Surv., vol. 43, n.o 2, p. 11:1-11:29, feb. 2011.

[54] B. Chaurasia y A. Verma, «A Comprehensive Study on Failure Detectors of Distributed Systems», J. Sci. Res., vol. 64, pp. 250-260, jul. 2020.

[55] R. Cortiñas Rodríguez, Failure detectors and communication efficiency in the crash and general omisión failure models. Servicio Editorial de la Universidad del País Vasco/Euskal Herriko Unibertsitatearen Argitalpen Zerbitzua, 2011.

[56] M. Fazlali, S. Eftekhar, M. M. Dehshibi, H. Tabatabaee Malazi, y M. Nosrati, Raft Consensus Algorithm: an Effective Substitute for Paxos in High Throughput P2P-based Systems. 2019.

[57] N. Hayashibara, P. Urban, A. Schiper, T. Katayama, y F. Communications, «Performance Comparison Between the Paxos and Chandra-Toueg Consensus Algorithms», oct. 2002.

[58] K. Fall, «A delay-tolerant network architecture for challenged internets», en Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, en SIGCOMM '03. New York, NY, USA: Association for Computing Machinery, ago. 2003, pp. 27-34.

[59] A. Benchi, P. Launay, y F. Guidec, «Solving Consensus in Opportunistic Networks», en Proceedings of the 2015 International Conference on Distributed Computing and Networking, Goa India: ACM, ene. 2015, pp. 1-10.

[60] K. K. Ahmed, M. H. Omar, y S. Hassan, «Survey and Comparison of Operating Concept for Routing Protocols in DTN», J. Comput. Sci., vol. 12, n.o 3, pp. 141-152, abr. 2016.

[61] M. Jahanian y K. K. Ramakrishnan, «CoNICE: Consensus in Intermittently-Connected Environments by Exploiting Naming with Application to Emergency Response», en 2020 IEEE 28th International Conference on Network Protocols (ICNP), oct. 2020, pp. 1-12.

[62] R. Ramanathan y J. Redi, «A brief overview of ad hoc networks: challenges and directions», IEEE Commun. Mag., vol. 40, n.o 5, pp. 20-22, may 2002.

[63] J.-S. Lee, Y.-W. Su, y C.-C. Shen, «A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi», en IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society, nov. 2007, pp. 46-51.

[64] A. Talaminos Barroso, J. Reina Tosina, y L. M. Roa Romero, «Sistema IoMT para la comunicación distribuida y descentralizada de sensores en redes de área personal», XXXIV Simposium Nacional de la Unión Científica Internacional de Radio. Sevilla, España., 2019.

[65] L. Baresi, N. Derakhshan, S. Guinea, y F. Arenella, «MAGNET: A middleware for the proximal interaction of devices based on Wi-Fi direct», en 2017 IEEE International Conference on Communications (ICC), may 2017, pp. 1-7.

[66] C. E. Casetti, C. F. Chiasserini, Y. Duan, P. Giaccone, y A. Perez Manriquez, «Data Connectivity and Smart Group Formation in Wi-Fi Direct Multi-Group Networks», IEEE Trans. Netw. Serv. Manag., vol. 15, n.o 1, pp. 245-259, mar. 2018.

[67] J. H. Lee, M.-S. Park, y S. C. Shah, «Wi-Fi direct based mobile ad hoc network», en 2017 2nd International Conference on Computer and Communication Systems (ICCCS), jul. 2017, pp. 116-120.

[68] R. Alnashwan y H. Mokhtar, «Wi-Fi Direct Issues and Challenges», en Advances in Security, Networks, and Internet of Things, K. Daimi, H. R. Arabnia, L. Deligiannidis, M.-S. Hwang, y F. G. Tinetti, Eds., en Transactions on Computational Science and Computational Intelligence. Cham: Springer International Publishing, 2021, pp. 525-539.

[69] A. Teófilo, J. M. Lourenço, y H. Paulino, «RedMesh: A WiFi-Direct Network Formation Algorithm for Large-Scale Scenarios», en MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Darmstadt Germany: ACM, dic. 2020, pp. 147-157.

[70] C. Petrioli y S. Basagni, «Degree-constrained multihop scatternet formation for Bluetooth networks», dic. 2002, pp. 222-226 vol.1.

[71] H. Howard, M. Schwarzkopf, A. Madhavapeddy, y J. Crowcroft, «Raft Refloated: Do We Have Consensus?», ACM SIGOPS Oper. Syst. Rev., vol. 49, n.o 1, pp. 12-21, ene. 2015.

[72] WiFi Alliance, «Wi-Fi Display Technical Specification Version 2.1». 2017.

[73] M. Conti, F. Delmastro, G. Minutiello, y R. Paris, «Experimenting opportunistic networks with WiFi Direct», en 2013 IFIP Wireless Days (WD), nov. 2013, pp. 1-6.

[74] C. Funai, C. Tapparello, y W. Heinzelman, «Enabling multi-hop ad hoc networks through WiFi Direct multi-group networking», en 2017 International Conference on Computing, Networking and Communications (ICNC), ene. 2017, pp. 491-497.

[75] M. A. Khan, W. Cherif, F. Filali, y R. Hamila, «Wi-Fi Direct Research - Current Status and Future Perspectives», J. Netw. Comput. Appl., vol. 93, pp. 245-258, sep. 2017.

[76] V. Kumkar, A. Tiwari, P. Tiwari, A. Gupta, y S. Shrawne, «Vulnerabilities of Wireless Security protocols (WEP and WPA2)», Int. J. Adv. Res. Comput. Eng. Technol., vol. 1, ene. 2012.

[77] D. Santhadevi, V. Kareer, y S. Tokas, «A Review- Issues and Challenges in Wireless Network Security», Int. J. Radio Freq. Des., vol. 2, n.o 2, pp. 8-13, 2016.

[78] A. Stubblefield, J. Ioannidis, y A. Rubin, «Using the Fluhrer, Mantin, and Shamir Attack to Break WEP», presentado en Network and Distributed System Security Symposium, 2002.

[79] E. Tews, R.-P. Weinmann, y A. Pyshkin, «Breaking 104 Bit WEP in Less Than 60 Seconds», en Information Security Applications, S. Kim, M. Yung, y H.-W. Lee, Eds., en Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 188-202.

[80] «Wi-Fi Protected Setup | Wi-Fi Alliance». https://www.wi-fi.org/discover-wi-fi/wi-fi-protected-setup

[81] A. H. Lashkari, M. M. S. Danesh, y B. Samadi, «A survey on wireless security protocols (WEP, WPA and WPA2/802.11i)», en 2009 2nd IEEE International Conference on Computer Science and Information Technology, ago. 2009, pp. 48-52.

[82] E. Baray y N. Kumar Ojha, «'WLAN Security Protocols and WPA3 Security Approach Measurement Through Aircrack-ng Technique'», en 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), abr. 2021, pp. 23-30.

[83] D. Camps-Mur, A. Garcia-Saavedra, y P. Serrano, «Device-to-device communications with Wi-Fi Direct: overview and experimentation», IEEE Wirel. Commun., vol. 20, n.o 3, pp. 96-104, jun. 2013.

[84] H. Zhang, Y. Wang, y C. C. Tan, «WD2: an improved wifi-direct group formation protocol», en Proceedings of the 9th ACM MobiCom workshop on Challenged networks, en CHANTS '14. New York, NY, USA: Association for Computing Machinery, sep. 2014, pp. 55-60.

[85] K. Jahed, O. Farhat, G. Al-Jurdi, y S. Sharafeddine, «Optimized group owner selection in WiFi direct networks», en 2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), sep. 2016, pp. 1-5.

[86] M. A. Khan, W. Cherif, y F. Filali, «Group owner election in Wi-Fi direct», en 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), oct. 2016, pp. 1-9.

[87] E. Yasser, A. Salah, y A. Kamel, «Enhancing the quality of service of mobile-based software over Wi-Fi direct», en 2021 The 4th International Conference on Software Engineering and Information Management, en ICSIM 2021. New York, NY, USA: Association for Computing Machinery, jul. 2021, pp. 1-9.

[88] R. M. Mbala, J. M. Nlong, y J. R. K. Kamdjoug, «Android Wi-Fi Direct Architecture: From Protocol Implementation to Formal Specification», Int. J. Eng. Res. Technol., vol. 9, n.o 2, feb. 2020.

[89] A. A. Shahin y M. Younis, «Efficient multi-group formation and communication protocol for Wi-Fi Direct», oct. 2015.

[90] P. Chaki, M. Yasuda, y N. Fujita, «Seamless Group Reformation in WiFi Peer to Peer network using dormant backend links», en 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), ene. 2015, pp. 773-778.

[91] J. Dizdarević, F. Carpio, A. Jukan, y X. Masip-Bruin, «A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration», ACM Comput. Surv., vol. 51, n.o 6, p. 116:1-116:29, ene. 2019.

[92] «MQTT - The Standard for IoT Messaging». https://mqtt.org/

[93] D. Dinculeană y X. Cheng, «Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices», Appl. Sci., vol. 9, n.o 5, Art. n.o 5, ene. 2019.

[94] L. Durkop, B. Czybik, y J. Jasperneite, «Performance evaluation of M2M protocols over cellular networks in a lab environment», en 2015 18th International Conference on Intelligence in Next Generation Networks, feb. 2015, pp. 70-75.

[95] S. Wagle, «Semantic data extraction over MQTT for IoTcentric wireless sensor networks», en 2016 International Conference on Internet of Things and Applications (IOTA), ene. 2016, pp. 227-232.

[96] C. Bormann, A. P. Castellani, y Z. Shelby, «CoAP: An Application Protocol for Billions of Tiny Internet Nodes», IEEE Internet Comput., vol. 16, n.o 2, pp. 62-67, mar. 2012.

[97] N. Naik, «Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP», en 2017 IEEE International Systems Engineering Symposium (ISSE), oct. 2017, pp. 1-7.

[98] F. T. Johnsen, T. H. Bloebaum, M. Avlesen, S. Spjelkavik, y B. Vik, «Evaluation of transport protocols for web services», en 2013 Military Communications and Information Systems Conference, oct. 2013, pp. 1-6.

[99] T. M. Tukade y R. Banakar, «Data transfer protocols in IoT-an overview», Int. J. Pure Appl. Math., vol. 118, pp. 121-138, ene. 2018.

[100] Y. Chen y T. Kunz, «Performance evaluation of IoT protocols under a constrained wireless access network», en 2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT), abr. 2016, pp. 1-7.

[101] T. White, M. Johnstone, y M. Peacock, «An investigation into some security issues in the DDS messaging protocol», Aust. Inf. Secur. Manag. Conf., ene. 2017.

[102] J. Du, C. Gao, y T. Feng, «Formal Safety Assessment and Improvement of DDS Protocol for Industrial Data Distribution Service», Future Internet, vol. 15, n.o 1, Art. n.o 1, ene. 2023.

[103] J. Hoffert, D. C. Schmidt, y A. Gokhale, «Evaluating Transport Protocols for Real-Time Event Stream Processing Middleware and Applications», en On the Move to Meaningful Internet Systems: OTM 2009, R. Meersman, T. Dillon, y P. Herrero, Eds., en Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 614-633.

[104] A. Talaminos-Barroso, M. A. Estudillo-Valderrama, L. M. Roa, J. Reina-Tosina, y F. Ortega-Ruiz, «A Machine-to-Machine protocol benchmark for eHealth applications – Use case: Respiratory rehabilitation», Comput. Methods Programs Biomed., vol. 129, pp. 1-11, jun. 2016.

[105] A. Foster, «A Comparison Between DDS, AMQP, MQTT, JMS, REST, CoAP, and XMPP». 2017.

[106] K. An, A. Gokhale, D. Schmidt, S. Tambe, P. Pazandak, y G. Pardo-Castellote, «Content-based filtering discovery protocol (CFDP): scalable and efficient OMG DDS discovery protocol», en Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, en DEBS '14. New York, NY, USA: Association for Computing Machinery, may 2014, pp. 130-141.

[107] W. Yuan y K. Nahrstedt, «Energy-efficient soft real-time CPU scheduling for mobile multimedia systems», en Proceedings of the nineteenth ACM symposium on Operating systems principles, en SOSP '03. New York, NY, USA: Association for Computing Machinery, oct. 2003, pp. 149-163.

[108] C. Bayılmış, M. A. Ebleme, Ü. Çavuşoğlu, K. Küçük, y A. Sevin, «A survey on communication protocols and performance evaluations for Internet of Things», Digit. Commun. Netw., vol. 8, n.o 6, pp. 1094-1104, dic. 2022.

[109] L. Baresi, N. Derakhshan, y S. Guinea, «WiDiSi: A Wi-Fi direct simulator», en 2016 IEEE Wireless Communications and Networking Conference, abr. 2016, pp. 1-7.

[110] A. Montresor y M. Jelasity, «PeerSim: A scalable P2P simulator», en 2009 IEEE Ninth International Conference on Peer-to-Peer Computing, sep. 2009, pp. 99-100.

[111] «Ideal Modeling & Diagramming Tool for Agile Team Collaboration». https://www.visual-paradigm.com/ (accedido 12 de julio de 2023).

[112] C. Guindon, «Eclipse Desktop & Web IDEs | The Eclipse Foundation», Eclipse Foundation. https://www.eclipse.org/ide/ (accedido 12 de julio de 2023).

[113] «Diseño, site survey, heatmaps y análisis- Acrylic WI-FI», Acrylic WiFi, 20 de junio de 2023. https://www.acrylicwifi.com/ (accedido 12 de julio de 2023).

[114] A. Aragão, «Raft: Understandable Distributed Consensus». 12 de octubre de 2022.

[115] N. Derakhshan, «WiDiSi». 8 de julio de 2022. Accedido: 18 de abril de 2023. [En línea]. Disponible en: https://github.com/nasser1941/WiDiSi

[116] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, y B. Walke, «The IEEE 802.11 universe», IEEE Commun. Mag., vol. 48, n.o 1, pp. 62-70, ene. 2010.