UNIVERSIDAD
NACIONAL
DE COLOMBIA

Implementación de un módulo de software que dote a nodos potenciales de una red ad hoc con las capacidades para ingresar, permanecer, gestionar recursos y salir de un sistema distribuido que opera sobre esa red

John Edwar González O.

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Sistemas e Industrial
Bogotá, Colombia
2019

Implementación de un módulo de software que dote a nodos potenciales de una red ad hoc con las capacidades para ingresar, permanecer, gestionar recursos y salir de un sistema distribuido que opera sobre esa red

John Edwar González Ortiz

Tesis de grado presentada como requisito parcial para optar al título de:
Magister en Ingeniería - Telecomunicaciones

Director:
Ph.D. Jorge Eduardo Ortiz Triviño

Línea de Investigación:
Computación Aplicada
Grupo de Investigación:
TLON

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Sistemas e Industrial
Bogotá, Colombia
2019

Dedicatoria

A mis padres, a mis hermanos Didier y Giselle, a mis amigos Andrés y Annie, mi director de tesis Jorge Eduardo Ortiz y mi tutor Henry Zárate, y a mis compañeros del grupo de investigación TLÖN.

Resumen

El sistema TLÖN es un sistema de cómputo distribuido implementado sobre una red ad hoc y diseñado bajo un paradigma de social-inspiración que organiza sus recursos de forma análoga a como un país o estado gestiona sus recursos naturales e inmateriales. En esa analogía, el territorio corresponde a la red ad hoc, de forma que los recursos del sistema corresponden a la suma de los recursos que cada nodo aporta (CPU, disco duro, memoria, dispositivos de entrada y salida, etc.). Este informe presenta el análisis, diseño e implementación de una solución de software para afrontar ese reto. El módulo construido bajo los lineamientos de social-inspiración permite a cualquier nodo con capacidad de configuración en modo ad hoc unirse, permanecer, tomar decisiones, gestionar recursos locales para cooperar con la actividad del sistema general y finalmente salir del sistema distribuido. Para implementar la característica de autonomía en la gestión local de recursos, se utilizó el método multicriterio Scoring para la toma de decisiones. Para la negociación de la cantidad de recursos que aporta cada nodo, se empleó el método de concesiones sucesivas de Zeuthen. También fue necesario construir una interfaz de comunicación entre el módulo local y el sistema global, con el fin de establecer un protocolo adecuado para estas interacciones. Los resultaron mostraron, entre otras cosas, que un nodo prefiere realizar el proceso de unión con sistemas distribuidos ya formados en vez de crear sistemas con otros nodos solitarios.

Palabras clave: Red Ad Hoc, Sistema distribuido, Social-Inspiración, Toma de Decisiones.

Abstract

The TLÖN system is a distributed computing system implemented on an ad hoc network and designed under a social-inspiration paradigm, the one that has been adopted for this system organizes its resources analogously to how a country or state manages its natural and immaterial resources. In this analogy, the territory corresponds to the ad hoc network, so that the resources of the system correspond to the sum of the resources provided by each node (CPU, Hard Disk, Memory, input and output devices, etc.). This report presents the analysis, design and implementation of a software solution for this challenge. The module built under these social-inspiration guidelines allows that any node with capability of configuration in ad hoc mode joins, stays, makes decisions, and manages local resources to cooperate with the activity of the general system and finally exit the distributed system. To implement the autonomy feature in local resource management, the Scoring multicriteria method was used for decision-making. For the negotiation of the amount of resources provided by each node, Zeuthen's method of successive concessions was applied. It was also necessary to build a communication interface between the local module and the global system in order to establish a suitable protocol for these interactions. The results showed, among other things,

that a node prefers to carry out the union process with distributed systems already formed instead of creating systems with other solitary nodes.

Keywords: Ad Hoc Network, Decision-Making, Distributed System, Social-Inspiration.

Contenido

Resumen vii

1. Introducción 2

2. Marco Teórico 7
2.1. Tecnologías Inalámbricas . . . 7
2.2. Internet de las Cosas . . . 7
2.3. Red Ad Hoc . . . 9
2.4. Nube Móvil . . . 11
2.4.1. Tipos de Cooperación e Incentivos . . . 12
2.5. Sistema Distribuido . . . 13
2.6. Fog Computing . . . 14
2.7. Protocolos de Enrutamiento para redes Ad Hoc . . . 15
2.7.1. Protocolos Reactivos . . . 16
2.7.2. Protocolos Proactivos . . . 17
2.7.3. Protocolos Híbridos . . . 19
2.8. Protocolo BATMAN . . . 20
2.9. Sistema de Telecomunicaciones Social-Inspirado TLÖN . . . 22

3. Toma de Decisiones y Negociación 25
3.1. Función de Utilidad . . . 25
3.2. Toma de Decisiones . . . 26
3.2.1. Teoría de la Decisión bajo incertidumbre o riesgo . . . 27
3.2.2. Decisión Multicriterio . . . 29
3.2.3. Teoría de Juegos . . . 36
3.3. Negociación . . . 37
3.3.1. Modelo de Negociación de Stahl . . . 39
3.3.2. Modelo de Negociación de Rubinstein . . . 40

4. Modelo Aplicado al Sistema Distribuido 42
4.1. Unión de Nodo con Otro Nodo Solitario . . . 43
4.2. Unión de nodo con un sistema distribuido . . . 44
4.3. Método de decisión Multicriterio discreto Scoring . . . 44
4.4. Principio de Concesiones Alternativas de Zeuthen . . . 48

5. Resultados **52**
5.1. Módulo de Recepción y Envío de Solicitudes de Unión 53
5.2. Toma de Decisiones y Negociación 55
5.2.1. Tipos de Colaboraciones 57
5.2.2. Nodo con otro Nodo Solitario 61
5.2.3. Nodo Solitario con Sistema Distribuido 65
5.2.4. Autenticación de Integrantes del Sistema 71
5.3. Recepción de Instrucciones e Información 71
5.4. Salida del Nodo del Sistema 74

6. Conclusiones y recomendaciones **76**
6.1. Conclusiones 76
6.2. Recomendaciones 77

A. Anexo: Metodología SADT **79**

B. Anexo: Programas en Python **81**

C. Anexo: Resultados **82**

D. Anexo: Manual Programas **83**

Bibliografía **84**

1. Introducción

En las últimas décadas, la ciencia ha estado interesada en comprender los procesos biológicos que hacen posible la vida. Estos procesos están caracterizados por su complejidad y perfección, y algunos científicos han querido aplicarlos a la solución de problemas de la ingeniería [27]. El éxito de la adaptación de paradigmas bioinspirados se observa en redes neuronales, lógica difusa, o en inteligencia artificial. De igual manera, se ha tenido mucho interés en formular matemáticamente diferentes conceptos sociales. Por ejemplo, la justicia y la teoría de la elección social [26] se han usado como base para una equitativa distribución de recursos entre los miembros de sistemas distribuidos [56]. La social-inspiración, al igual que la bioinspiración, intenta aplicar este tipo de conceptos, y ha tomado fuerza a la hora de solucionar problemas de ingeniería, teniendo como punto de partida el éxito de toda clase de comunidades [10].

Se han tomado muchos ejemplos de la naturaleza, sobre todo de las comunidades de animales. Uno de estos se ha denominado inteligencia de enjambre, donde, como su nombre lo indica, se aplican patrones de comportamiento que tienen grupos de animales, tales como hormigas, abejas, aves, etc. para la auto-organización y descentralización de varios robots o agentes, lo que ofrece mayor tolerancia a fallas, mayor potencialidad en la escalabilidad de sistemas, entre otros. En [3] se analiza el modelado, análisis y arquitectura de hardware para este tipo de sistemas. Algunos ejemplos de esta aplicación se pueden observar en el caso de los enjambres de abejas, cuyos patrones de movimiento han servido de inspiración para controlar grupos de robots [66]. Por su parte, colonias de hormigas se han utilizado por agentes artificiales para encontrar la ruta más corta entre dos puntos [19]. También se han basado en colonias de hormigas para detectar los cambios entre redes y descubrir las topologías de comunicación asociadas a cada una de las configuraciones de red [2]. Al igual, los patrones de las hormigas en búsqueda de comida han servido para realizar minería de datos [3], ya que en esa búsqueda se hace un barrido espacial muy eficiente. En otros casos se utiliza la inteligencia de enjambre basándose en el algoritmo del murciélago para encontrar la localización de un objetivo dentro de ambientes 3D cerrados y desconocidos, pero estáticos [73].

En esta clase de comunidades, el hecho de estar acompañados por miembros de la misma especie, su disposición a la hora de cumplir las tareas que les corresponden dependiendo del rango y su deseo de colaboración ha determinado el éxito y su supervivencia en la naturaleza [45], donde a lo anterior se añade el concepto de estabilidad y eficiencia. En el caso de los

seres humanos, este hecho es similar, pero aquí el concepto tiene un sentido más amplio [48], ya que la sociedad se define como una asociación de personas que conforman una empresa cooperativa, se ayudan entre sí para obtener el bienestar de todos, y tienen como base la justicia y la democracia.

Las tecnologías inalámbricas y su rápido crecimiento han estado influenciados por la necesidad de comunicación que existe en nuestra sociedad [60]. Estas tecnologías no solo han permitido la comunicación entre personas, sino su interacción con el mundo que las rodea, esto a su vez ha creado la necesidad de llegar a zonas apartadas de la geografía terrestre donde existe poca infraestructura que soporte comunicación inalámbrica. Una solución muy importante son las redes ad hoc [22, 37, 43], las cuales no necesitan de ningún tipo de infraestructura física fija, son dinámicas y flexibles y además se pueden configurar sin ningún tipo de intervención por parte del usuario. La expansión de este tipo de redes ha creado la necesidad de la utilización de sistemas que sean capaces de manejar los recursos de manera eficiente y soporten funcionalidades complejas como la virtualización [40], que es esencial a la hora de proveer servicios y recursos a cada uno de los miembros de una red.

El grupo de investigación TLÖN de la Universidad Nacional de Colombia [1] fue creado en 2015 y está conformado por profesores y estudiantes de ingeniería a nivel de pregrado, maestría y doctorado. Que buscan resolver las necesidades planteadas por las nuevas tecnologías en cuanto a las problemáticas de integración e intercambio de información en ambientes heterogéneos. Por esto, uno de sus objetivos es la aplicación del paradigma de la social-inspiración [85] en sistemas distribuidos que se implementen sobre redes ad hoc con el fin de brindar recursos y aplicaciones de manera justa, con base en los modelos de justicia de Jhon Rawls [48], inmanencia de Baruch de Spinoza [71], paradigma de Thomas Kuhn [35] y estado de Thomas Hobbes [29]. Este proyecto en general busca ser un primer paso hacia la social-inspiración. Con este paradigma se busca, entre otras cosas, aplicar justicia a la red, donde los nodos con más recursos colaborarán con aquellos que menos recursos tienen. Por ser algo novedoso, no es un propósito que la red sea más rápida en este momento, pero las primeras pruebas pueden dar paso a una investigación que busque alcanzar esa característica. Este proyecto de tesis en específico es importante para el objetivo del grupo TLÖN, ya que en el camino hacia la social-inspiración es necesario tener en cuenta muchas características que poseen las comunidades, como la toma de decisiones, la negociación, el altruísmo y egoísmo, entre otros comportamientos que aquí se observan.

En este proyecto en específico, el primer paso que será aplicar la capacidad de toma de decisiones a los nodos potenciales de un sistema distribuido, de manera que ellos puedan decidir, con base en los recursos que les ofrece, si desean ingresar o no. Para implementar el proceso de toma de decisiones se debe crear previamente un submódulo para que el nodo reciba y envíe mensajes de saludo de manera que se dé a conocer ante otros nodos u otros

sistemas distribuidos que se encuentren en el área. Además es necesario hacer un intercambio de información de recursos del nodo y el sistema, para que luego el nodo decida si quiere ingresar o no al sistema. Luego se implementará un proceso de negociación que consistirá en establecer cuales son los recursos con los que va a colaborar el nodo con el sistema y cuáles tendrá derecho a utilizar una vez haga parte del mismo.

En todos los nodos del sistema distribuido debe haber un protocolo de comunicaciones que controle el intercambio de instrucciones e información de ida y vuelta entre el nodo y el sistema, de manera que se puedan recibir instrucciones de alto nivel propias del lenguaje TLÖN que luego serán traducidas a lenguaje de máquina. Hay que tener en cuenta que este protocolo también controlará la recepción de solicitudes de nodos potenciales del sistema, por lo que deberá ser capaz de procesarlas. Finalmente, si el nodo decide irse, deberá indicar cuáles son los recursos que se lleva consigo y entregar la información que tiene almacenada, luego, el sistema distribuido le dará su aprobación para poder salir del mismo.

El objetivo general de esta tesis es "Implementar un módulo de Software que dote a nodos potenciales de una red ad hoc con las capacidades para ingresar, permanecer, gestionar recursos y salir de un sistema distribuido que opera sobre esa red". Para esto, se plantean los siguientes objetivos específicos:

- Proponer los protocolos de comunicaciones entre el módulo nodo/Entidad y entre el Módulo nodo/Máquina virtual.
- Implementar un submódulo en el nodo, para que escuche y realice peticiones de unión a redes ad hoc.
- Implementar un modelo de toma de decisiones en los nodos, con el fin de establecer su ingreso a una red ad hoc.
- Implementar un submódulo que declare ante la Entidad cuáles son los recursos que va a compartir con el sistema distribuido.
- Implementar un submódulo para que el nodo se retire de la red ad hoc en un proceso de salida controlada o fortuita.

Para el desarrollo del proyecto de tesis, se realizaron los objetivos propuestos anteriormente de manera transversal. Se utilizó la técnica del método científico, así:

- **Revisión Bibliográfica**: Se validó la documentación relacionada con la social-inspiración; la toma de decisiones y negociación, su forma matemática y su aplicación; y sobre el protocolo de comunicaciones en capa 2 que se utilizó en la red ad hoc.

- **Realización de Pruebas**: Se realizaron prototipos programando dos familias de nodos, tres dispositivos Raspberry Pi y un computador Dell. Luego, se analizaron cada uno de los submódulos, se tomaron variables como el tiempo y finalmente se midieron la cantidad de nodos que ingresaron a la red luego de realizar el proceso de toma de decisiones y negociación.
- **Análisis de Resultados**: Con base en los resultados obtenidos, se realizaron análisis que indicaron si estos fueron satisfactorios y se decidió si se avanzaba hacia etapas posteriores del proyecto.
- **Modificación de Hipótesis**: Si los resultados no eran los esperados, se realizarían nuevamente los pasos previos, teniendo en cuenta nuevas técnicas, para así llegar al objetivo propuesto.
- **Aplicación y Validación**: Cuando todos los resultados fueron obtenidos y los estándares requeridos fueron alcanzados, se realizó la aplicación de todos los resultados en un prototipo funcional que marcó la finalización del proyecto.
- **Documentación**: Se realizó la documentación de todos los pasos previos según el formato existente en la Universidad.

Como resultado del proceso de investigación se crearon tres artículos científicos que fueron mostrando cada una de las etapas del proyecto.

- Revista Sistemas y Telemática de la Universidad Icesi de Cali, "Towards the design of a social-inspired module for the decision making of nodes in an ad hoc network", año 2016.
- Revista Ingeniería y Región de la Universidad Surcolombiana de Neiva, "Toma de Decisiones en Nodos del Sistema Distribuido TLÖN", año 2017.
- Simposio Internacional de Ingeniería Industrial y Administración de Operaciones (IEOM), "Sociability as a decision model for artificial agents", Bristol-UK, año 2017.

Este documento recopila todas las etapas de la investigación y está compuesto de las siguientes partes. En el primer capítulo se desarrolla el marco teórico, con algunos conceptos relacionados con las tecnologías inalámbricas, redes ad hoc y sistemas distribuidos. En el capítulo dos, se hace una revisión de diferentes modelos matemáticos que simulan la racionalidad y permiten la toma de decisiones. Se muestran procesos de toma de decisiones multicriterio discreto y definiciones sobre teoría de la negociación y algunos métodos. En el capítulo tres, se describe el método multicriterio de toma de decisiones escogido para la decisión inicial y luego el método de negociación que permitirá al nodo indicar cuáles son

los recursos con los que va aportar al sistema, en los dos principales escenarios ante los cuales se encontrará un nodo. En el cuarto capítulo se esbozan los esquemas de todos los programas que se realizaron y se explican por bloques de programas su funcionamiento y su razón de ser. Finalmente en el quinto capítulo se exponen los resultados de los experimentos realizados con cuatro nodos y en el capítulo sexto se presentan las principales conclusiones y recomendaciones.

2. Marco Teórico

2.1. Tecnologías Inalámbricas

Se prevé que en los próximos 10 años los cambios más significativos que tenga el Internet se deban a la alta proliferación que tienen los dispositivos inalámbricos [22, 37, 60]. Las facilidades que ofrece el uso del Internet en la interacción con el mundo físico que nos rodea y las relaciones con otras personas, está cambiando la forma en cómo vivimos y trabajamos. La manera en la cual los datos fluyen y se manejan, la necesidad de acceso a grandes cantidades de información, está creando retos en cuanto a la arquitectura de este tipo de tecnologías; esta tiene que evolucionar a un ritmo tal que soporte todas esas necesidades.

Un ejemplo de este rápido cambio es el protocolo TCP/IP el cual fue diseñado inicialmente para Routers cableados, nuevas versiones del protocolo tal como IP móvil son utilizados por servicios celulares móviles tales como enlaces de radio o enlaces desde dispositivos con estaciones base [60].

La tecnología inalámbrica más ampliamente usada es la relacionada con la red celular la cual provee servicios de telefonía y datos a dispositivos móviles [22], existen casi 5 mil millones de estos dispositivos alrededor del mundo y este número continúa en aumento día a día. En la Figura **2-1** se observa como estas redes han evolucionado desde sistemas análogos a sistemas digitales y CDMA, luego a sistemas de tercera generación 3G y finalmente los sistemas 4G que soportan velocidades del rango de 10-100 Mbps, como lo son LTE y WiMAX/IEEE 802.16.

2.2. Internet de las Cosas

Con la penetración de las tecnologías inalámbricas en la vida diaria, ha aparecido un término muy interesante que se relaciona a la interacción de las personas con cada dispositivo electrónico que lo rodea [38], ya sea sensores, actuadores, electrodomésticos, máquinas, etc. Teniendo en cuenta dispositivos con características mecánicas, eléctricas y electrónicas, hay más máquinas que gente en el planeta, y el crecimiento de los que están conectados a internet es exponencial, tan solo en el año 2010 ya habían 1.500 millones de dispositivos conectados

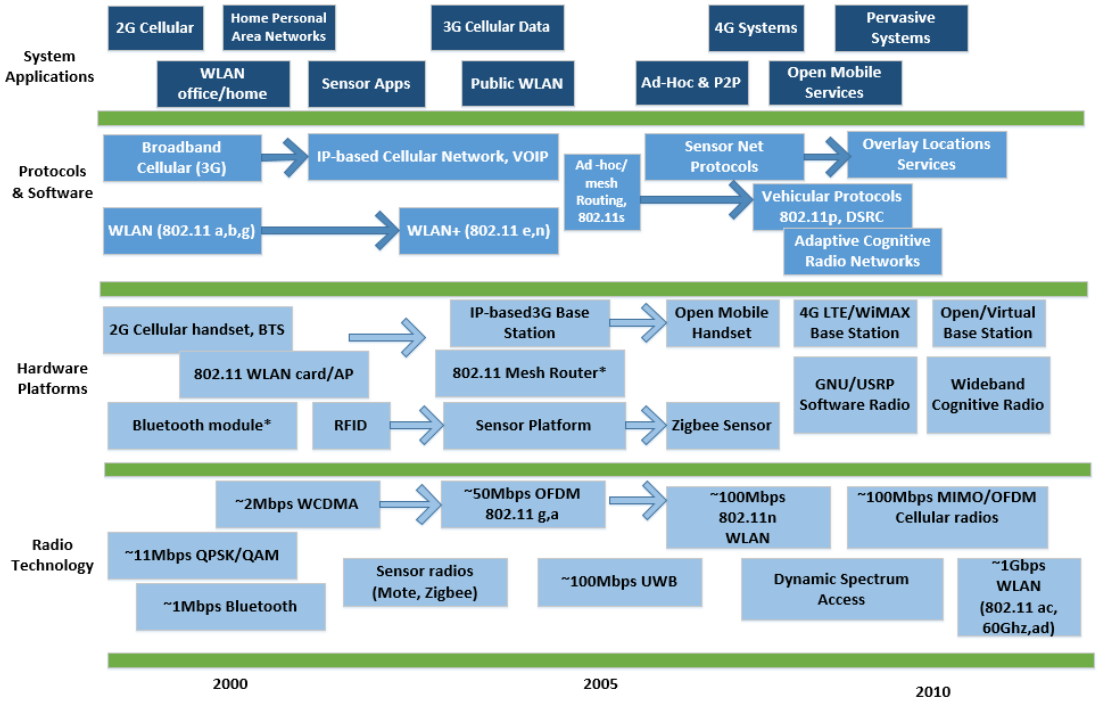

Figura 2-1.: Evolución de las redes inalámbricas, adaptado de [60]

a la red y se espera que en el año 2020, existan más de 20 mil millones [39].

El internet de las cosas se refiere a la comunicación de estos dispositivos con el Internet, esta tecnología ha venido encontrando aplicaciones en todo ámbito, desde la industria, sectores agrícolas, educativos hasta los hogares. Un ejemplo de su aplicación podría ser el de la nevera que verifica constantemente su contenido y la cual con la programación adecuada, podría establecer que no hay suficiente comida y automáticamente solicitaría al supermercado la cantidad necesaria. Una aplicación es la implementación del Internet de las cosas para monitorear las condiciones en el hogar [33]; mediante el diseño correcto de la arquitectura de red y su interconexión adecuada se encontró una alta confiabilidad de transmisión de la información. En [67] se aplica el paradigma de las redes definidas por Software (SDN por sus siglas en inglés) junto con el de Fog Computing para superar los retos del IoT, resaltar sus ventajas y exponer sus debilidades.

Este tipo de enfoque trajo un reto y fue la creación de un nuevo paradigma de enrutamiento ya que actualmente la mayoría de redes están basadas en el protocolo Ipv4 para direccionar cada dispositivo, Ipv4 permite un poco más de 4 mil millones de direcciones Ipv4; hoy se ha utiliza una técnica que consiste en dividir las redes en direccionamiento público y privado y además crear subredes mediante la técnica VLSM para tener la mayor cantidad de IP's disponibles, pero aún así la amenaza de que se agote todo el direccionamiento Ipv4

Figura 2-2.: Imagen de una red ad hoc, fuente propia

está latente. Por lo que se tuvo la necesidad de encontrar un método diferente [83]; Ipv6 apareció como una alternativa para poder aplicar el paradigma del internet de las cosas ya que con su direccionamiento adicional se pueden alcanzar hasta 340 sextillones de direcciones.

En Indonesia se empezó con la implementación de este método de direccionamiento de tal manera que se pudiera adelantarse a la problemática vista anteriormente, pero todavía presenta varios retos, por ejemplo el principal ISP del país se niega a realizar la transición a este tipo de tecnología, ya que indica que el contenido que se consume no está listo para ello [83], en Colombia por ejemplo [69], se analizan las consideraciones técnicas necesarias para la transmisión de televisión sobre Ipv6, debido a que muchos servicios como PPV (Pago por Ver) y Triple Play utilizan este protocolo.

2.3. Red Ad Hoc

Las redes ad hoc son una colección de dos o más dispositivos inalámbricos que poseen la capacidad de comunicarse entre ellos y tienen como principal característica la no utilización de infraestructura física fija ni administración centralizada [37, 22, 43], sino que basan su funcionamiento en la interacción de sus nodos quienes se encargan de compartir sus recursos para el uso de toda la red. Cada nodo puede ser un dispositivo con interfaz inalámbrica como un celular, un computador, una Tablet, entre otros; los cuales poseen características y recursos heterogéneos, como se puede ver en la Figura **2-2**.

Entre las principales ventajas de las redes ad hoc se tienen las siguientes [43, 37]:

- Facilidad de implementación y economía: Al no necesitar infraestructura física en su despliegue, se pueden utilizar los dispositivos móviles que estén disponibles para formarla. Una de las aplicaciones más importantes de las redes ad hoc es ante situaciones de catástrofe natural o guerras, en donde, las estaciones bases de comunicación, la fibra óptica y demás componentes basados en infraestructura, colapsan; una implementación rápida de una red ad hoc permite la comunicación entre entidades de emergencia y puede ser de gran ayuda a la hora de salvar vidas. En [13] se resalta la importancia de este tipo de redes, y de su modelado y simulación, aplicados a sistemas de telecomunicaciones de emergencia en las diferentes etapas de prevención, atención y recuperación.

- Adaptabilidad: Estas redes se caracterizan por poseer una gran movilidad, los nodos se desplazan de manera aleatoria sobre toda el área de cobertura de la red, entran y salen. Todos los nodos se reconfiguran y ajustan automáticamente a los cambios de topología. Entre sus múltiples usos, un ejemplo muy claro de su adaptabilidad es en el control de tránsito, específicamente en VANET (Vehicular Ad Hoc Networks). Como por ejemplo, para indicar a los vehículos a qué velocidad deben ir para disminuir al máximo las emisiones de CO2 [52]; o para aumentar la seguridad en las vías [51].

- Descentralizada: Cada nodo es independiente, y dependiendo del protocolo de enrutamiento, solo tiene conocimiento de la ubicación de sus vecinos más cercanos, la red no va a colapsar cuando uno de los nodos deje la red; lo cual la hace bastante tolerante a fallos.

- Cooperación: La comunicación entre dos nodos de una red ad hoc, puede pasar por varios nodos intermedios para llegar desde un origen a un destino determinado, cada nodo intermedio colabora con el paso del tráfico de un nodo a otro, es decir, comparte sus recursos para el funcionamiento de toda la red.

La red ad hoc posee ciertas limitaciones debido a las características de cada uno de sus componentes y del medio de transmisión [43, 37]:

- Limitado ancho de banda: Está relacionado con el ancho de banda de cada uno de los nodos que se encuentran en la ruta desde el origen hasta el destino de una conexión, y del hecho de que en la transmisión de paquetes se está consumiendo ancho de banda de los nodos intermedios. Se debe tener un control de acceso riguroso de nodos, dependiendo de su ancho de banda, ya que las comunicaciones pueden verse afectadas con el ingreso de una gran cantidad de nodos al sistema [74].

- Limitada capacidad de energía: Los dispositivos inalámbricos, en su gran mayoría funcionan a través de baterías, las cuales tienen cierta capacidad. En el proceso de realizar cualquier acción, enviar datos, pasarlos a otros nodos, en el cálculo de rutas, se genera

un gasto energético, el cual tiene que ser tenido en cuenta en la implementación de cualquier red ad hoc.

- Inconvenientes de seguridad: La comunicación, al propagarse por medio aéreo, hace que sea propensa a amenazas como acceso no autorizado a la información u otras encargadas de corromper los datos. Algunos autores están interesados en evaluar el impacto del nodo oculto en redes ad hoc, ataque que tiene que ver con el medio de transmisión [59].

- Utilización del espectro: Pueden existir problemas por la utilización del espectro de frecuencias debido a otros dispositivos ajenos a la red, lo que puede ocasionar ruidos y a la vez pérdida de paquetes en la transmisión. Por ejemplo se utiliza el modelamiento Cross-Layer para disminuir la pérdida de paquetes en un ambiente con mucho ruido [34].

Las redes ad hoc están dentro de una clasificación más general como se observa en [22], llamada nube móvil; esta incluye redes con infraestructura física fija llamadas redes Mesh, estas a la vez comprenden por ejemplo, redes de telefonía celular, las cuales se enlazan con estaciones base o redes inalámbricas que posean un BackBone con salida a Internet a través de Routers; se toma la información relevante a su comportamiento ya que el autor le imprime un enfoque social-inspirado el cual está relacionado al proyecto general que se maneja en el grupo de investigación TLÖN.

2.4. Nube Móvil

El término nube móvil se define como una disposición cooperativa de nodos conectados dinámicamente que comparten recursos en tanto estos estén disponibles [22], El término Cloud definida por el autor es “la abstracción de un sistema que consiste de recursos distribuidos e interconectados”.
El autor en [22], señala las ventajas de este tipo de enfoque:

- Mejora en el rendimiento: El término rendimiento incluye calidad de servicio, cobertura, confiabilidad, seguridad, ancho de banda, entre otros; de las comunicaciones de todos los nodos de la red. En [61] y en [28] el autor utiliza un método enfocado en la reputación, para satisfacer las necesidades de seguridad para redes ad hoc que comparten recursos.

- Utilización eficiente y flexible de recursos: Permite la utilización de recursos de manera eficiente, ya que estos son limitados, costosos y escasos, Entre ellos están, por ejemplo,

la energía y utilización del espectro, en [76], el autor provee un método de optimización de la descarga de información que ahorra energía.

- Recursos distribuidos: Lo cual es bastante importante ya que permite aprovechar recursos que pertenecen a diferentes nodos en la red. El autor en [21] utiliza los nodos de la red para guardar la información de manera distribuida; en [31] el autor expone la idea de utilizar una nube móvil para almacenar datos localmente en vez de descargarlos de Internet; y en [49], el autor utiliza la tecnología de virtualización para alcanzar la disponibilidad de los servicios en una red, compartiendo los recursos de la misma.

2.4.1. Tipos de Cooperación e Incentivos

En una nube móvil existen diferentes tipos de cooperación e incentivos; éstas se presentan por la relación existente entre los nodos [22].

- Cooperación Forzada: Se presenta cuando, de una manera obligada o forzada, se solicita a un dispositivo inalámbrico servir a uno o más dispositivos diferentes. Si los dispositivos pertenecen al mismo propietario, en el caso de una red de sensores, esta clase de cooperación se convierte en auto-cooperación.
- Cooperación Altruista: Surge cuando un dispositivo móvil decide colaborar en una red sin esperar nada a cambio; por ejemplo al compartir la señal de Internet desde un celular, este está gastando más batería, pero no espera nada a cambio; esta cooperación se puede presentar en ambientes familiares o de amigos.
- Cooperación Egoísta: Se presenta cuando un dispositivo móvil colabora en una red si ve que puede obtener algún beneficio en hacerlo, por ejemplo en obtener algún incentivo o mejorar la reputación que pueda tener el nodo en la red. En algunos casos si muchos nodos se comportan de manera egoísta en una red ad hoc, puede ocasionar que esta colapse ya que no va a tener suficientes recursos para enviar paquetes, como se observa en [12]. En [77] se propone el protocolo SCNP (Protocolo de verificación y Negociación de Egoísmo por sus siglas en inglés) en donde permite negociar el grado de egoísmo el cual pueda tener un nodo con tal que colabore con más recursos en la red dando incentivos si lo hace; este proyecto en específico tiene mucho que ver con esto, ya que los nodos al realizar el proceso de ingreso tienen un índice de altruísmo-egoísmo el cual los hace más propensos a dar más o menos recursos cuando están negociando con la red, este comportamiento también regirá mientras hacen parte de la misma.
- Cooperación Social: La cooperación se presenta en un dispositivo por su impacto social, ya que el nodo puede mejorar su prestigio o porque este soporta aplicaciones que fueron diseñadas para cooperación entre múltiples nodos. El trabajo realizado en [53] muestra un ejemplo en el cual varios nodos colaboran de manera conjunta para mejorar el desempeño de la red formando clusters.

2.5. Sistema Distribuido

Definido por [42] un sistema distribuido es un conjunto de recursos informáticos que se pueden utilizar para hacer diferentes tipos de operaciones, en él hay un sistema operativo de alto nivel que se encarga de la administración de cada uno de los nodos del sistema, además para el usuario, los recursos pueden ser accesados con su nombre únicamente, sin importar su localización.

Para [22], un sistema distribuido se puede definir como una colección de nodos que están separados físicamente, pero que se unen entre sí por algún medio de telecomunicaciones; cada nodo tiene características propias, sus recursos son heterogéneos. Un sistema distribuido se puede implementar sobre cualquier red de comunicaciones y permite la comunicación entre múltiples nodos; en él, el usuario accede a cualquier componente de cualquier nodo, y lo percibe como si se tratara de un recurso local y en donde todos los dispositivos forman un solo equipo de cómputo.
Como se puede observar ambas definiciones no difieren mucho, definiciones similares se puede ver en [25] y [7]. Lo que es común entre las definiciones, es indicar las siguientes características que un sistema distribuido debe tener:

- Tolerancia a Fallos: Un fallo en alguno de los nodos de un sistema distribuido se puede presentar en cualquier momento, ya sea hardware o software, pero esto no debe importar ya que es un inconveniente local y el sistema distribuido debe tener la manera de tener un respaldo ante cualquier eventualidad.

- Heterogeneidad: La definición de sistema distribuido nos indica que está compuesta de múltiples nodos, cada uno de ellos pueden ser un computador, un celular, una tablet, etc. [25] Indica que esta característica no solo aplica al hardware sino al software, es decir, los sistemas operativos, los lenguajes de programación, etc.

- Escalabilidad: Esta característica, según se observa en [42], se refiere a su rendimiento confiable incluso con el aumento del número de nodos en la red.

- Concurrencia: Se refiere al hecho de que se puedan realizar múltiples procesos u operaciones de manera simultánea en la red. En [7] se observa como realizar este tipo de procesos, por hilos, semáforos, etc.

- Transparencia: Indica que para el usuario final o aplicaciones, el sistema se presente como un solo equipo de cómputo.

- Extensibilidad y Apertura: Las interfaces deben estar separadas y abiertas para permitir que dos procesos puedan comunicarse entre sí, además debe permitir que nuevos componentes sean adheridos al sistema distribuido.

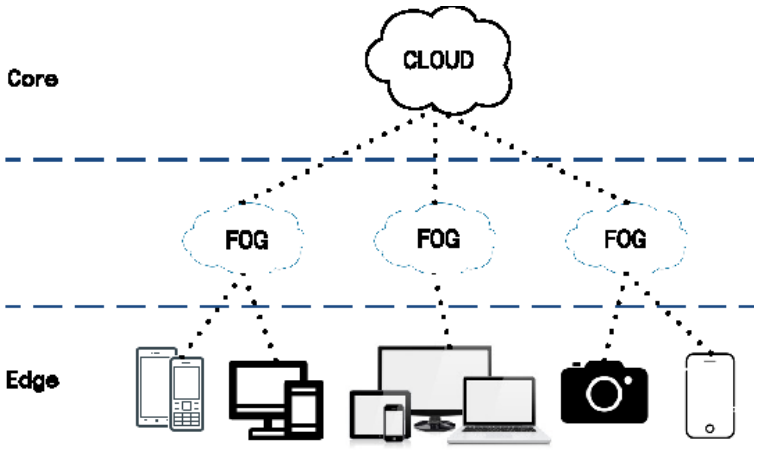

Figura 2-3.: Fog Computing, tomado de [4]

- Seguridad: La información que se maneja en todo sistema, debe estar protegida en contra de cualquier intento de acceso no autorizado, esta característica es muy importante en el sistema distribuido ya que se debe asegurar la integridad de la información.

2.6. Fog Computing

Este término tiene mucho que ver con Cloud Computing y se refiere a un tipo de arquitectura en la cual el procesamiento y almacenamiento de la información es realizada por los dispositivos que se encuentran al borde de la red e incluso por los mismos dispositivos que están conectados a la red Figura **2-3**, en vez de utilizar sistemas que se encuentren en la nube para este fin. Esta arquitectura también soporta redundancia lo que lo hace tolerante a fallos [11]. En [81] el autor la define como una "plataforma altamente virtualizada que provee procesamiento, almacenamiento, y servicios de red entre dispositivos finales y Datacenters tradicionales presentes en la nube". En [17] y [80] se observa la relación que tiene el concepto con el Internet de las Cosas ya que está más cerca del usuario final pues los dispositivos que se utilizan pueden ser computadores personales, celulares, tablets, etc. es por eso que es llamado Fog (Niebla) porque está más cerca al suelo que Cloud (Nube); además posee otras características que las hacen muy atractivas tales como [11]:

- Conocimiento de la Ubicación y Poca Latencia: Los primeros intentos para utilizar Fog Computing fueron en aplicaciones de video streaming y juegos los cuales requieren rapidez en la transmisión de datos, en [20] se demuestra como la arquitectura de Fog Computing junto a la tecnología de Machine Learning es usada para reducir la latencia y el consumo de energía en los dispositivos. .
- Gran Distribución Geográfica: Los servicios y aplicaciones que utilizan Fog Computing

requieren una gran distribución geográfica. Una aplicación del Fog computing podría ser el de los vehículos que van por una autopista y requieran información de las condiciones de la vía.

- Gran Cantidad de Nodos: Debido a la gran distribución geográfica es de esperar un gran número de nodos en este tipo de aplicaciones.
- Movilidad: Por el hecho de tratarse de redes que incluyen aplicaciones de redes de comunicación celular o Vanets, Fog Computing resuelve las necesidades que se puedan presentar ante las necesidades de movilidad de sus nodos.
- Heterogeneidad: Incluye muchos nodos de cualquier tipo y en cualquier ambiente de aplicaciones.
- Interoperabilidad: Por la gran cantidad de servicios que se pueden presentar en las redes soportadas por Fog Computing, es necesario que exista esta característica y permitir la unión con otros proveedores o servicios, por ejemplo todos los que se encuentren en la Nube (Cloud).

Los servicios que ofrece Fog Computing [81] son, la administración por medio de Software de redes, esto con el fin de evitar que miles de dispositivos tengan que ser administrados por un ente central lo que puede ser engorroso a la hora de encontrar y solucionar fallas sino que delega estas tareas a un software que solucione esto autónomamente; otro servicio es el de acercar la nube al usuario lo que descongestionaría la red e incluso permitiría que algunos dispositivos realizaran labores de cómputo y almacenamiento. Además se podría realizar administración distribuida donde un conjunto de usuarios podrían utilizar la red con sus propias aplicaciones las cuales no deban pasar por datacenters que estén centralizados esto impulsaría a los usuarios a tener mayor sentido de propiedad sobre su información y aplicativos.

Sus aplicaciones están orientadas a mejorar la experiencia del usuario aumentando la velocidad de las aplicaciones por la cercanía que presenta al usuario final, en [86] se hace uso de su arquitectura para mejorar el rendimiento de servicios Web, usando servidores intermedios. Además por su comportamiento es muy utilizado en redes vehiculares tipo ad hoc o VANET, en [36] se puede observar cómo se utiliza el Fog Computing para reemplazar el modelo de Cloud computing, para ello se hace uso de los carros que están más cerca para realizar procesamiento de información; basados en las observaciones se obtuvieron dos diferentes modelos de distribución de tareas de cómputo.

2.7. Protocolos de Enrutamiento para redes Ad Hoc

Con la aparición de las redes de emergencia de DARPA (Agencia de proyectos de investigación avanzados de defensa de los Estados Unidos por sus siglas en inglés) en los años 70,

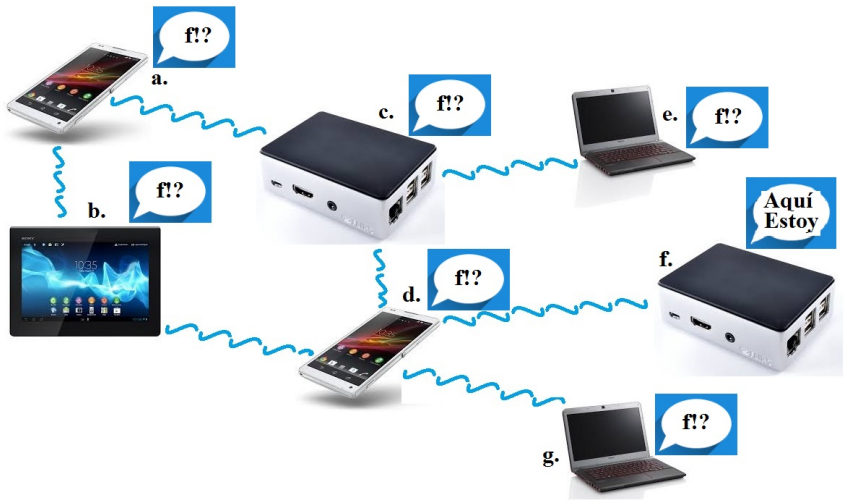

Figura 2-4.: Protocolos Reactivos

se desarrollaron numerosos protocolos de enrutamiento para redes ad hoc, estos protocolos, se dividen en tres clases principales [43, 86, 37], proactivos, reactivos (bajo demanda) y una tercera clase la cual se ha denominada como una clase híbrida. Se hace un repaso además, por los protocolos de enrutamiento más importantes que se encuentran en estas categorías.

2.7.1. Protocolos Reactivos

La principal característica de estos protocolos es que crean una ruta hasta su destino únicamente cuando se les es solicitado [8], esta ruta es mantenida en la tabla de enrutamiento por el tiempo en la cual la ruta sea utilizada o el destino ya sea inaccesible. Los nodos no tienen una tabla de topología de la red. Para el descubrimiento del nodo A, el requerimiento es enviado a todos los nodos vecinos los cuales envían mensajes por Broadcast hasta que el mensaje llegue al nodo destino (Figura **2-4**), cuando el nodo A recibe el mensaje responde el requerimiento para construir la ruta hasta el nodo origen. El proceso termina cuando todas las permutaciones de rutas son examinadas y la mejor ruta es escogida.

- AODV: Cuando el nodo A requiere alcanzar el nodo B [8], el Nodo A inunda la red con mensajes RREQ (requerimiento de ruta), a su paso por cada nodo, el protocolo crea tablas de enrutamiento temporales desde su ubicación hasta su origen. Cuando alcanza su destino, envía un mensaje RREP (Respuesta de ruta) sobre la misma ruta por la cual llegó el mensaje RREQ. La desventaja de este protocolo [43] es que se pueden presentar errores en los nodos intermedios por números de secuencias del nodo origen erróneas, es decir, que el mensaje RREP no está marcado con la secuencia correcta y esto da a paso a rutas que sean inconsistentes.

- DSR: Este protocolo funciona eliminando los mensajes periódicos de actualización de

tablas lo cual restringe el ancho de banda que consume el protocolo [64]. La característica principal que distingue este protocolo de los demás es que es basado en el concepto de Enrutamiento de Origen, es decir, los nodos deben mantener las rutas en una memoria caché que contienen los nodos originadores de los mensajes RREQ. Cada nodo originador de mensajes sabe la ruta salto a salto hasta su destino. El proceso de envío de un paquete hasta cierto destino tiene dos fases, descubrimiento de ruta y mantenimiento de ruta. El nodo primero consulta su memoria caché para verificar todas las rutas que conoce, si esa ruta no ha expirado será usada para enviar el paquete hasta su destino, si la ruta no existe, el nodo inicia un proceso de descubrimiento de ruta inundando sus vecinos con mensajes RREQ, cada nodo a su vez verifica su caché para verificar si conoce la ruta, sino realiza el mismo procedimiento. Cuando un nodo conoce la ruta hasta el destino o el destino es alcanzado el nodo envía un RREP a través de los nodos por donde recibió el mensaje. Las rutas en la memoria caché son mantenidas a través de mensajes originadores. La desventaja de este protocolo [43], es que sus cabeceras van creciendo a medida que pasa por un nodo hasta el destino.

- TORA: Es un protocolo de enrutamiento distribuido [43], el cual usa un algoritmo de inversión de enlace y provee una ruta hasta el destino libre de bucles. Cada nodo mantiene su información de la topología de nodos a un salto de distancia y tiene la capacidad de detectar particiones. Un característica clave de este protocolo es el envío de mensajes ante cualquier cambio de la topología de la red para mantener actualizada esa información en cada nodo. Para la creación de una ruta, primero se requiere de la configuración de una secuencia de enlaces desde el origen al destino de tal manera que la creación de rutas esencialmente corresponde a la asignación de direcciones a enlaces en una red. El proceso de mantenimiento de rutas reacciona ante cambios en la topología de tal manera que las rutas al destino son reestablecidas dentro de un tiempo definido. Esto lleva a la tercera función que es la eliminación de rutas, cuando se detectan nuevos conjuntos de nodos, todos los enlaces deben ser marcados como enlaces indirectos para eliminar rutas inválidas.

2.7.2. Protocolos Proactivos

Los protocolos proactivos [43] habilitan a los nodos para mantener la información de las rutas actualizadas. Estas tablas de enrutamiento son intercambiadas periódicamente con los otros nodos y cuando hay cambios en la topología de la red **2-5**. Esto significa que cuando un nodo necesita enviar un mensaje a un destino, éste ya sabe como llegar. La desventaja es que la mayoría de la información de las tablas de enrutamiento que se comparten son indeseadas, además tienen otra desventaja y es que al estar intercambiando la información de enrutamiento constantemente, consumen potencia y ancho de banda, ya que esto se da incluso cuando la red está en reposo.

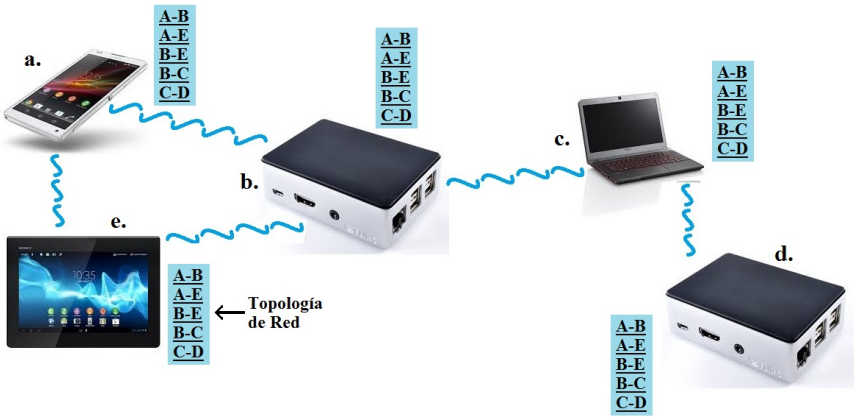

Figura 2-5.: Protocolos Proactivos

- BABEL: Babel es un protocolo de enrutamiento proactivo [79] que utiliza el algoritmo de Bellman-Ford. El protocolo detecta vecinos utilizando mensajes tipo "Hola", la frecuencia de recepción de dichos mensajes permite evaluar la calidad del enlace con otros nodos. Al recibir un "Hola", el nodo responde con un mensaje de "Hola recibido". El protocolo envía su tabla de periódicamente o cuando ocurre un evento en la red. Un problema que presenta este método es que por medio de las estimaciones de costos de los enlaces, un nodo puede agotar rutas factibles hacia un destino. Para solucionar esto, cuando un nodo no tiene suficientes rutas, envía un tipo de mensaje que estimula a los nodos vecinos a enviar nuevos mensajes para encontrar nuevas rutas.

- OLSR: En este protocolo [79], los nodos intercambian la información de enrutamiento de forma periódica para actualizar la información de la topología de la red, de esta forma los nodos determinan cuáles son sus vecinos dentro de su radio de transmisión. Después de este descubrimiento de vecinos, una cierta cantidad de nodos es escogido para actuar como MPR (Puntos Múltiples de Retransmisión por sus siglas en inglés) para enviar información de topología, estos nodos son un conjunto mínimo los cuales tienen comunicación directa con nodos más lejanos, o sea con nodos que estén a dos saltos de distancia, esto evita la inundación de la red con mensajes innecesarios a cada nodo de la red. Los nodos están enviando mensajes tipo Hola, los cuales indican a sus vecinos que están presentes en la red, el TTL es configurado en 1, de tal manera que esos mensajes no son reenviados. Una vez se obtenga la información de la topología de la red almacenada y actualizada, la ruta más corta de un origen a un destino se encuentra utilizando el algoritmo de Dijkstra. En [64] se observa cómo con el aumento de número de nodos en la red, el protocolo empieza a presentar inconvenientes, ya que no es bueno manejando la escalabilidad del sistema, OLSR mantiene en cada nodo una tabla de enrutamiento para todos los nodos de la red, el rendimiento del protocolo

disminuye con el ingreso de nuevos integrantes.

- DSDV: Es un método de enrutamiento que utiliza el algoritmo de Bellman-Ford [64], el cual encuentra la ruta más corta para definir un único camino a un destino. Cada tabla de enrutamiento lista todos los destinos posibles con un conteo de saltos y un número de secuencia, la información de enrutamiento es luego enviada por broadcast. Cada nodo transmite la información a sus vecinos. La información de enrutamiento es transmitida cada vez que se detecta un cambio en la topología. La gran desventaja de este protocolo [37] es que presenta gran cantidad de cabeceras por lo que no es recomendable para redes grandes, DSDV consume más ancho de banda que cualquier otro protocolo durante el envío de actualizaciones.

- WRP: Es un protocolo de enrutamiento proactivo [37] que utiliza una versión actualizada del algoritmo Bellman-Ford. Aquí cada nodo mantiene una tabla de distancia, una tabla de enrutamiento, una tabla de costo de enlace y una lista de mensajes de retransmisión (MRL). Los mensajes de actualización pueden ser enviados periódicamente o cuandoquiera que el estado de enlace cambie. La lista MRL contiene información acerca de cual vecino no ha recibido un mensaje de actualización. Al recibir un mensaje de actualización, el nodo modifica su tabla de distancia y busca mejores rutas. Aunque el protocolo ofrece varias rutas hasta un destino, debido a su mecanismo de sincronización basado en colas para evitar bucles, la cantidad de tablas e información necesaria para su funcionamiento incrementan su complejidad y bajan su rendimiento.

2.7.3. Protocolos Híbridos

Estos protocolos son proactivos y reactivos al mismo tiempo [37]. Fueron creados para aumentar la estabilidad permitiendo a los nodos más próximos trabajar juntos para formar una especie de backbone y reducir las grandes cabeceras en el descubrimiento de rutas. Aquí se utiliza un método proactivo para mantener las rutas de los nodos más cercanos y un método reactivo para descubrir las rutas de nodos más lejanos **2-6**. Estos protocolos son particionados en zonas y cada nodo pertenece a una de ellas.

- ZRP: Cada nodo mantiene actualizada la topología de cada zona, una zona es un conjunto de nodos que no están más lejanos que cierta cantidad de saltos de distancia [30]. Dentro de estas zonas, cada nodo puede ser encontrado mediante un método proactivo. Para las zonas exteriores el protocolo escoge cierta cantidad de nodos que se encuentren en las periferias de las zonas para que esos nodos envíen a la vez sus mensajes a sus nodos periféricos. En este protocolo las zonas se traslapan y un nodo puede ser miembro de múltiples zonas. La arquitectura de la red es jerárquica entonces cada nodo necesita almacenar esta información adicional lo que requiere más memoria.

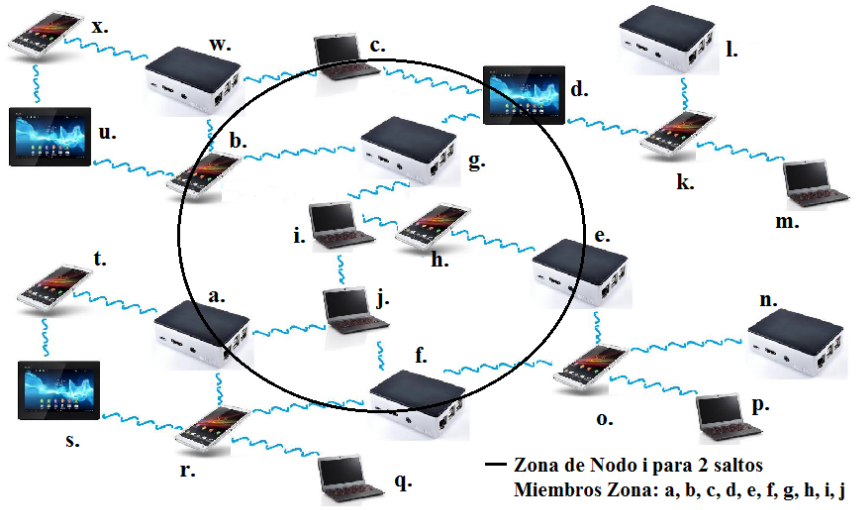

Figura 2-6.: Protocolos Híbridos

- ZHLS: Este es un protocolo de enrutamiento jerárquico que hace uso de la información de localización en un enfoque de enrutamiento peer-to-peer [30]. Aquí la red es dividida en zonas que no se traslapan. Cada nodo sabe su posición y por consiguiente el ID de su zona a través de GPS. Cuando la red es establecida, cada nodo sabe su nodo de menor nivel jerárquico y la zona de mayor nivel jerárquica. Si dos nodos están dentro del rango de comunicación se dice que existe un enlace físico, de lo contrario existirá un enlace virtual. La topología a nivel del nodo provee la información de cómo los nodos de diferentes zonas están conectados por enlaces virtuales. Un paquete es reenviado especificando el ID de la zona y el ID del nodo colocándolo en la cabecera del paquete.

2.8. Protocolo BATMAN

En el grupo de investigación se ha trabajado extensamente con el protocolo de enrutamiento B.A.T.M.A.N (Better Approach to Mobile Ad Hoc Networks) en donde se ha obtenido buenos resultados, el protocolo ha sido fácil de implementar, rápido, seguro y compatible con muchos dispositivos.

B.A.T.M.A.N [32, 41] es un protocolo de enrutamiento proactivo, que mantiene información de todos los nodos de la red, de tal manera que para cada posible destino se almacena la información del vecino más cercano; esto elimina la necesidad de que todos los nodos de la red conozcan la topología completa y evita el cálculo de rutas complejas debido a la permanente entrada y salida de nodos y a la movilidad de los mismos, lo cual resulta costoso en términos de procesamiento y ancho de banda.

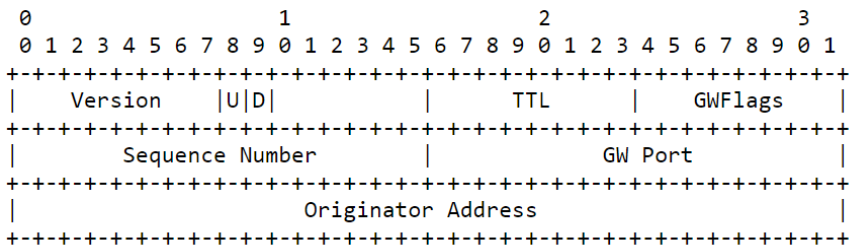

Figura 2-7.: Formato del mensaje OGM [41]

Para la comunicación, todos los nodos transmiten mensajes originadores OGM (Originator Message) a todos sus vecinos **2-7**, los cuales tiene dos principales objetivos, el primero es identificar los nodos vecinos, y el segundo es darse a conocer a toda la red; la información que le interesa a cada nodo con respecto a los demás es saber a través de qué vecino puede accederlo.

B.A.T.M.A.N es un protocolo de enrutamiento Capa 2, es decir, la información de enrutamiento es transportada utilizando tramas Ethernet. El protocolo encapsula y envía todo el tráfico hasta que alcance el destino emulando un gran switch virtual por lo cual no es afectado por el cambio de la topología de la red.

Para su implementación, se hace uso de dos módulos principales, Batman-adv la cual es la implementación del protocolo B.A.T.M.A.N en forma de un módulo en el Kernel de Linux que opera en capa 2. El hecho de estar en el Kernel elimina el inconveniente del procesamiento del paquete en el espacio de usuario, ya que al ir y volver limita el ancho de banda en dispositivos de gama baja. B.A.T.M.A.N ofrece el módulo Batctl el cual es una herramienta para configurar y depurar el módulo Batman-adv y ofrece una interfaz para todas las configuraciones posibles.

En [50] se realizan una serie de simulaciones sobre el protocolo de enrutamiento para determinar su eficacia y comparar las versiones de sus algoritmos, se determinó que la frecuencia de los mensajes originadores permiten un mayor conocimiento de la topología y una gran estabilidad en escenarios de poca movilidad, en [79] se hicieron algunos experimentos comparando B.A.T.M.A.N con otros protocolos de enrutamiento, y entre todos se resaltan como los mejores de lejos a Babel y B.A.T.M.A.N, el autor inicialmente resalta que no hay diferencias sustanciales en el desempeño de los dos, pero Babel funciona mejor en escenarios de mayor movilidad mientras que B.A.T.M.A.N funciona mejor en el escenario más estático, además se resalta que el protocolo es muy ligero y envía la información estrictamente necesaria para realizar la comunicación entre dos nodos por la mejor ruta.

En [32] se realiza una comparación de los protocolos B.A.T.M.A.N y OLSR, en un escenario

de pruebas de 49 nodos, se encontró que el protocolo OLSR presenta inconveniente de bucles debido a enlaces que no son simétricos mientras que para B.A.T.M.A.N este hecho fue una ventaja y funcionó más eficientemente, además encontraron que B.A.T.M.A.N tuvo mejor desempeño, menor retardo, un menor consumo de CPU del 17 % en promedio y menor longitud en las cabeceras de los mensajes.

2.9. Sistema de Telecomunicaciones Social-Inspirado TLÖN

Para abordar el tema de la tesis, inicialmente es necesario dar un repaso general sobre el proyecto que se tiene en el grupo de investigación TLÖN ya que esta tesis se desprende de ahí. Como se mencionó anteriormente uno de los objetivos en el grupo [1] es implementar un comportamiento social-inspirado en un sistema distribuido teniendo como base una red ad hoc, el sistema está basado en comunidades de agentes. La idea del sistema distribuido es que los dispositivos puedan compartir los recursos y servicios que se encuentren en la red ad hoc, de manera que los nodos que necesiten más recursos puedan acceder a ellos teniendo en cuenta el principio orientador de la justicia, de manera que a los nodos que más aportan respecto a los recursos que tienen disponibles, recibirán de igual manera una mayor cantidad de recursos y prioridad a la hora de requerirlos.

Actualmente existen muchos modelos que tratan de conseguir un comportamiento social entre los dispositivos [22]; pero todos ellos tienen el inconveniente de estar sujetos a la intervención del usuario, o tienen un esquema centralizado lo que los aleja de la característica ad hoc, es por eso que se hace necesario aplicar un paradigma social-inspirado y garantizar que la red se comporte realmente como una red ad hoc, es decir que sea lo suficientemente independiente para conformarse sin ninguna intervención, lo que sería útil en la conformación de redes de emergencia ya que garantizaría el acceso a recursos y servicios [13].

Existen además de la justicia cuya definición el grupo la tomó de [58], dos servicios orientadores que son el paradigma [35] que se define como el conjunto de creencias, valores y técnicas que comparten los miembros de una sociedad y la inmanencia [71] que es toda actividad relacionada a un sistema y en la cual su acción se ejecuta en su interior por ser esa su razón de ser. Estos principios orientadores tienen como objetivo principal la prestación de un buen servicio al usuario.

Existen cuatro elementos principales en la conformación del modelo del sistema distribuido TLÖN [1], el primero es el lenguaje de programación [85] con el cual se crearán los agentes que son a su vez los que van a interactuar y serán miembros de la sociedad; el segundo componente es un pseudo-estado el cual va a ser el encargado de regular el comportamiento de

los agentes y afectar la red de creencias; esto es necesario a la hora de crear las comunidades de agentes y buscar los objetivos de justicia e inmanencia en la red. El territorio es la red ad hoc con todos sus elementos como se ha mencionado anteriormente y además con sus controles y unas normas específicas que consoliden su constitución. El último componente es la máquina virtual que es la encargada de la comunicación entre el lenguaje y la red ad hoc, esta máquina virtual hará la traducción de lenguaje de alto nivel que es el lenguaje propio del sistema distribuido TLÖN y el lenguaje de los dispositivos que pueden ser órdenes en lenguaje python o en bash por ejemplo.

Los sistemas multi-agente hacen referencia a sistemas compuestos de múltiples componentes autónomos, los agentes pueden colaborar, competir o simplemente coexistir en un sistema multi-agente, los agentes son sistemas computacionales que actúan de manera independiente o autónoma para satisfacer sus propios requerimientos [68]; los agentes se ubican en un ambiente. La acción social se da cuando un agente es capaz de descubrir la existencia de otros agentes. El agente tiene parte de su control desde el pseudo-estado ya que este es el que define su comportamiento dependiendo de las funciones que haga y el lugar donde este viva o se mueva. El software es la esencia del agente [1] y es la que permite la comunicación entre diferentes agentes. Es necesario que el agente tenga una red de creencias [68] las cuales le permitan actuar con respecto a las señales o percepciones que el agente capte del medio.

En los lenguajes actuales y simuladores no existen las herramientas suficientes para la comprensión del funcionamiento de una red ad hoc, mediante el diseño de un lenguaje de dominio específico ayudará a la concepción de las redes ad hoc [68]; a la vez ayudará al control de la administración de recursos en la red ad hoc, a la integración de elementos de inteligencia artificial para ayudar a la gestión de la red, a la interoperabilidad entre los diferentes entornos de funcionamientos de los nodos, y al monitoreo del comportamiento de clusters de nodos.

A la hora de crear un lenguaje de este estilo se tienen problemas como es el del paradigma de programación y la aplicación del concepto de inmanencia para el modelo del comportamiento de las comunidades de agentes. Al finalizar el proceso de la creación de lenguaje [1], se espera obtener una herramienta que permita la creación de comunidades de agentes inteligentes los cuales permitan que cada nodo de la red pueda acceder de una manera justa a los recursos que están disponibles en la red ad hoc.

La capa de virtualización [1] que se observa en el modelo de la Figura **2-8** es la que se encarga de tomar los recursos de la red ad hoc que se encuentra en la parte inferior mediante un protocolo de comunicaciones el cual será específico del sistema TLÖN. La herramienta que realizará la virtualización será Docker; esta herramienta permite ejecutar aplicaciones sobre contenedores. Cada contenedor contiene todas las herramientas de Software necesarias para su ejecución, y operan sobre un cluster el cual es un conjunto de nodos interconectados

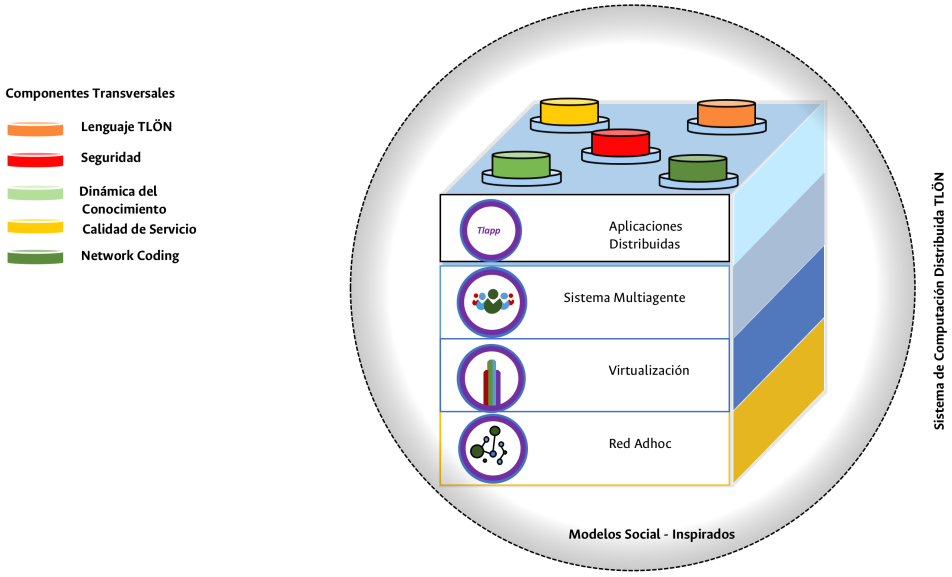

Figura 2-8.: Modelo Deseado

que ofrecen los recursos necesarios para poder formar un contenedor. En la primera capa de virtualización existe una serie de contenedores, los cuales representan todos los recursos que se van a disponer tales como, procesador, memoria, almacenamiento, dispositivos de entrada y salida; en un nivel superior a esta capa se encuentra el orquestador el cual se encarga de la administración de estos recursos de acuerdo a los principios sociales que se programen en él, además se encarga de la operación de los recursos para la siguiente capa que se observa en la Figura **2-8** que es el sistema operativo SOVORA (por sus siglas Sistema Operativo Virtualizado Orientado a Redes Ad Hoc), este sistema operativo soporta el resto del esquema que comprende la comunidad de agentes y las aplicaciones.

3. Toma de Decisiones y Negociación

El primer paso hacia la social-inspiración en sistemas distribuidos, es implementar la característica de racionalidad en los nodos que quieran hacer parte de los mismos, de manera que puedan decidir sobre ingresar o no, teniendo en cuenta los recursos que el sistema les ofrece, y además la capacidad de realizar la negociación sobre la cantidad de recursos que van a entregar al sistema. En este ámbito, se podría tener un escenario muy acorde al incluir la capacidad de toma de decisiones en los nodos, de manera que autónomamente eligieran la posibilidad de ingresar a un sistema distribuido, dependiendo de lo que éste les ofrezca. Esto haría que la red conserve la principal característica de las redes ad hoc que es la auto-organización, ya que el nodo sin necesidad de intervención del usuario evaluaría los recursos que le está ofreciendo el sistema distribuido y con base en los que tiene, elegiría si ingresa o no.

En este capítulo se observarán definiciones sobre toma de decisiones multicriterio y negociación, además se exponen diferentes métodos de los mismos los cuales podrían aplicarse a nodos potenciales del Sistema Distribuido TLÖN y la justificación de la elección del método multicriterio Scoring [62] para la toma de decisiones y el método de negociaciones sucesivas de Zeuthen [26, 16], como los que más se ajustan por los parámetros que se manejan. Los métodos escogidos se ven más a fondo en el capítulo número 4 ya que se observan en dos escenarios en los cuales se encontrarán los nodos.

Antes de empezar con estas definiciones se va a incluir un concepto muy importante en el estudio matemático de la racionalidad, y tiene que ver con la manera en que se expresan las preferencias de los decisores, la función de utilidad.

3.1. Función de Utilidad

No siempre las salidas tienen ganancias cuantificables, por ejemplo existe el prestigio, la calidad, cualidades sobresalientes, etc. Para reflejar esta valoración de los pagos, por el valor que tiene este para el decisor se usa la utilidad, la cual es una valoración personal de una cantidad. La utilidad se plasma en la función de utilidad [44, 26], la cual es una función que resume la importancia que un decisor asocia a diferentes cantidades. Es diferente dependiendo del decisor, ya que cada uno tendrá su escala de preferencias.

Para que una función, pueda ser llamada función de utilidad, esta debe cumplir con cierta axiomática, pero para llegar a ello, antes es necesario definir el concepto de lotería, ya que mediante esta se define una relación de preferencia que es lo que se utiliza para definir la función de utilidad. Una lotería se compone por unos valores de ganancia y unas probabilidades.

Existen dos tipos de loterías:
Loterías simples: La consecuencia es el resultado de una alternativa:

$$l = \begin{pmatrix} p_1 & \dots & p_n \\ x_1 & \dots & x_n \end{pmatrix} \text{ Con } p_i \in [0,1] \text{ y } \sum_{i=1}^{n} p_i = 1 \tag{3-1}$$

Loterías compuestas: Los premios son a su vez loterías:

$$L = \begin{pmatrix} q_1 & \dots & q_m \\ l_1 & \dots & l_m \end{pmatrix} \text{ Con } q_j \in [0,1] \text{ y } \sum_{j=1}^{m} q_j = 1 \tag{3-2}$$

Matemáticamente, la axiomática de Luce & Raiffa [44] define una función de utilidad y se expresa de la siguiente manera:

$$u : X \longrightarrow [0,1]$$

$$x_i \longrightarrow u(x_i) = u_i$$

La función u, definida sobre un conjunto de premios (X) que está en el intervalo $[0,1]$, de tal manera que a cada alternativa (x_i) se le asigna un número (u_i) y que tiene una estructura de preorden completo, es una función de utilidad. La función de utilidad es la función que especifica todos los pagos de la lotería, y la utilidad esperada de una lotería sería $\sum_j p_i u(r_i) = 1$. De tal manera que las alternativas de una decisión, son a la vez loterías.

Para estimar la función de utilidad, se propone ir comparando las loterías, de modo que en unas se obtenga un valor seguro y en otros ciertos valores con determinadas probabilidades. En [23] se propone automatizar la tarea del decisor, para ello incorpora las preferencias de un decisor humano a un sistema el cual evalúa la función de utilidad de manera continua. En el caso de [75], el decisor expresa sus preferencias, las cuales son recogidos por una función de base radial normalizada el cual inclina el proceso de selección hacia esa región que mejor se aproxime a las preferencias.

3.2. Toma de Decisiones

La toma de decisiones es el proceso mediante el cual, se realiza la elección de una opción entre varias, mediante un método que pueda resaltar algunas de las opciones de otras, de manera que se puedan organizar de manera jerárquica [65].

A la hora de abarcar la teoría de la toma de decisiones, el análisis se divide en tres grandes bloques [82], así:

- La teoría de decisión con incertidumbre o riesgo: En este caso, la toma de decisiones se realiza teniendo en cuenta que las consecuencias no pueden ser controladas, ya sea porque no se tiene suficiente información o porque las mismas están sujetas al azar.
- Decisión multicriterio: Aunque se sabe las consecuencias de escoger una opción, no se sabe qué es lo mejor entre todas ellas, algunos de sus objetivos pueden estar en conflicto.
- Teoría de Juegos: Las consecuencias de la decisión no dependen de la decisión adoptada por el decisor, sino también de la que elijan otros jugadores.

3.2.1. Teoría de la Decisión bajo incertidumbre o riesgo

El proceso de realizar una acción conlleva a cualquiera de un conjunto de salidas, si el caso es que se sabe cuál es la probabilidad de esas salidas, se dice que se toma la decisión bajo riesgo; pero si las probabilidades no son conocidas, el proceso de toma de decisión se realiza bajo incertidumbre [44, 57]; esto se puede dar ya sea porque las consecuencias de las decisiones están sujetas por la aleatoriedad o porque se desconozcan las consecuencias por falta de información. Los elementos que intervienen en una toma de decisión de estas características son:

$E = \{E_1, E_2, \ldots, E_m\}$; Conjunto de posibles escenarios
$A = \{A_1, A_2, \ldots, A_n\}$; Conjunto de posibles alternativas o decisiones
X_{ij}; Consecuencia de tomar la decisión A_i y se dé el escenario E_j
P_j; Probabilidad de que se dé el Estado E_j

Cuando existe una sola decisión que haya que tomarse en un momento dado y los conjuntos de estados y alternativas son finitos, se puede abordar el problema de toma de decisión mediante la Tabla **3-1**.

Tabla 3-1.: Matríz de Pagos.

	E_1	E_2	$\ldots$	E_m
	p_1	p_2	$\ldots$	p_m
A_1	X_{11}	X_{12}	$\ldots$	X_{1m}
A_2	X_{21}	X_{22}	$\ldots$	X_{2m}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
A_n	X_{n1}	X_{n2}	$\ldots$	X_{nm}

En este caso la mayor dificultad es la valoración de las alternativas para luego poder compararlas con otras [44, 57], se tienen distintos criterios de evaluación para ayudar a tomar una decisión, estos criterios se clasifican de acuerdo a si se utilizan las probabilidades de los estados o no:

Criterios de evaluación de alternativas conociendo las probabilidades de los estados [44, 57]

- Criterio del valor esperado: Como su nombre lo indica, este criterio elige la alternativa con el valor esperado más alto. Este criterio es util cuando se repite el proceso muchas veces, si se hace una sola vez no es el apropiado.

- Criterio de lo más probable: Supone elegir el mejor valor para el estado más probable. Este criterio es mejor utilizarlo cuando el proceso de decisión solo se hace una única vez.

- Criterio del escenario medio: Se escoge un escenario medio y se busca la alternativa óptima para ese escenario. No es aconsejable ya que en ocasiones los escenarios medios pueden distar de los escenarios reales.

Criterios de evaluación de alternativas sin utilizar las probabilidades de los estados [44, 57]

- Criterio de Wald: Este criterio se conoce como el criterio pesimista, ya que para cada alternativa se supone que va a pasar lo peor y se elige la alternativa que mayor valor tenga entre ellas. De este modo se va a asegurar que por lo menos se obtenga lo mejor posible en el peor de los casos.

- Criterio Optimista: Es un criterio que es precisamente opuesto que el criterio de Wald, para cada alternativa se supone que va a pasar lo mejor, y se elige el que dé el mejor valor, este criterio no se utiliza mucho ya que no tiene en cuenta los riesgos que conlleva elegir cualquiera de las alternativas.

- Criterio de Hurwics: Para este criterio se realiza una ponderación valorando cada alternativa entre lo mejor y lo peor posible, se hace multiplicando el valor más alto por α (índice de optimismo) y lo peor por $1 - \alpha$, para luego sumar ambas cantidades. Se elige la alternativa que mejor valor dé.

- Criterios de Savage: Este criterio considera una penalización por arrepentimiento al no prever el estado de la naturaleza. Es decir, si el decisor escoge una alternativa cuyo estado de la naturaleza no es el mayor, dejará de ganar la diferencia entre ese estado de la naturaleza contra el mayor, estas pérdidas se suman para obtener un ponderado. Con estos ponderados se crea una matriz en la cual se aplica uno de los métodos anteriores

que usualmente es el criterio de Wald lo cual se conoce como el criterio de minimizar al máximo el costo de arrepentimiento.

3.2.2. Decisión Multicriterio

Como se mencionó anteriormente, la toma de decisiones es el proceso mediante el cual se realiza la elección de una opción entre varias, mediante un método que pueda resaltar algunas de las opciones de otras, de manera que se puedan organizar de manera jerárquica. La elección consiste pues en elegir lo mejor entre lo posible.

Lo posible se refiere a la enumeración de todas las alternativas o establecer al menos una región factible [82, 15]. La región factible consiste en el conjunto de alternativas cuando estas no se pueden definir de manera precisa, este conjunto puede ser discreto o continuo; a su vez, la región factible puede ser limitada de manera estricta mediante restricciones o de manera flexible por niveles de aspiración.

En cuanto a lo mejor, se debe elegir uno o más criterios para poder resaltar esa alternativa de las demás [82]. Entre mayor número de criterios hayan, el problema de elección se vuelve un poco más complejo y se pueden resolver con métodos denominados de optimización multiobjetivo. Si lo posible viene definido por un conjunto discreto de alternativas existen métodos multicriterio discretos para resolver el problema.

Optimización Multiobjetivo

La optimización se refiere a encontrar las mejores soluciones a un problema dado, se puede por ejemplo decir que la meta es maximizar el rendimiento de un sistema con la mínima cantidad de recursos o en las situaciones diarias de la vida, se trata de maximizar las ganancias minimizando los costos [14, 57, 18].

Los problemas de optimización multiobjetivo se pueden definir como: “El problema de encontrar un vector de variables de decisión los cuales satisfacen restricciones y optimizan una función vector cuyos elementos representan las funciones objetivo. Esas funciones forman una descripción matemática de criterio de rendimiento las cuales están en conflicto con todas las otras. Por lo tanto, el término “optimizar” significa encontrar tal solución la cual dé los valores de todas las funciones objetivo aceptables para el decisor” [15].

Un problema de decisión multiobjetivo, requiere que el decisor haga una elección de varios valores. Los objetivos definen propiedades que una buena alternativa deben tener y los atributos el grado en que ese objetivo alcanza un grado específico. Los problemas multiobjetivo se refieren a problemas donde la meta es optimizar k funciones objetivo simultáneamente.

Esto puede involucrar la maximización de todas las funciones k, la minimización de todas las funciones k o una combinación de minimización y maximización. Usualmente los objetivos están en conflicto con otros, por ejemplo, al minimizar el coste en la elaboración de un producto, se puede sacrificar la calidad de este.

Formalizando [15], un problema de optimización multiobjetivo se utiliza para minimizar o maximizar $F(x) = (f_1(x), \ldots, f_k(x))$ sujeto a $g_i(x) \geq 0$, $i = 1, \ldots, m$, y $h_j(x) = 0$, $j = 1, \ldots, p$ $x \in \Omega$. Luego, una solución de un problema de optimización multiobjetivo minimiza o maximiza los componentes del vector $F(x)$, donde x es un vector de n dimensiones de variables de decisión $x = x_1, \ldots, x_n$ de algún universo Ω. Las restricciones $g_i(x)$ y $h_j(x)$ son restricciones que deben ser satisfechas y Ω contiene los valores de x que se usan para satisfacer una evaluación en $F(x)$.

Existe una definición que se va a utilizar sobre todo en la parte de negociación pero es muy utilizada en los procesos de optimización multiobjetivo y es el óptimo de Pareto [15], una solución $x \in \Omega$ es un óptimo de Pareto con respecto a Ω si y solo si no existe ningún $x' \in \Omega$ para el cual $v = F(x') = (f_1(x), \ldots, f_k(x'))$ domine $u = F(x) = (f_1(x), \ldots, f_k(x))$, es decir, que no existe ningún x que incremente algún criterio sin causar que el otro decremente (en el caso de la maximización).

En [84] se utiliza un algoritmo de optimización multiobjetivo basado en el enjambre de la partícula del caos para maximizar cuatro objetivos al mismo tiempo, minimizar la pérdida de potencia, minimizar la desviación de voltaje del nodo, mejorar la estabilidad del voltaje estático y minimizar el costo reactivo. En [24] se observan diferentes métodos de optimización basados en algoritmos genéticos, el resultado de ese trabajo fue un marco de articulación de preferencias unificado e independiente de escalas, el autor resalta la gran cantidad de aplicaciones que los algoritmos genéticos pueden ser utilizados para resolver problemas de decisión que incluyan variables numéricas o no numéricas.

Métodos de Decisión Multicriterio Discreto

Para tomar la decisión sobre el ingreso al sistema, el nodo toma cada uno de los recursos de sí mismo y los que tiene la red como los criterios para tomar la decisión. El nodo analiza cuales son los recursos que posee, de tal manera que asigna más peso a los recursos de los cuales necesita o carece. Por su bajo costo computacional, rapidez, facilidad de implementación, la forma en la cual vienen representados los recursos y además teniendo en cuenta que este proyecto se va a implementar en redes ad hoc donde los dispositivos son heterogéneos y muchos de ellos no cuentan con gran capacidad de procesamiento, se escogió esta clase de métodos de toma de decisiones para la implementación en el sistema, lo que sigue es un repaso de los principales métodos de decisión multicriterio discreto y las razones para la elección del méto-

do multicriterio Scoring [62] como el que más se ajusta en los escenario de ingreso al sistema.

Estos métodos se consideran cuando el conjunto de alternativas es discreto, y se pueden enumerar fácilmente [9]. Entre los métodos más importantes se pueden enumerar los siguientes:

Procesos Analíticos Jerarquizados (método AHP)

Es una técnica que permite la resolución de problemas multicriterio en donde, como una de las principales características se tiene la posibilidad de incorporar aspectos intangibles como el subjetivismo y la incertidumbre, aspectos inherentes a la toma de decisiones [47, 9]. Este método permite que el problema de toma de decisiones se pueda estructurar de manera visual mediante jerarquías, el método se divide en ocho etapas, de la siguiente manera:
Este método considera el problema dividido en 3 niveles o jerarquías [9].

1. Descomponer el problema en una jerarquía de elementos interrelacionados: En este paso es necesario identificar los tres niveles que permite estructurar el problema mediante una jerarquía de atributos.
   - Nivel 1, Propósito del problema: Este paso consiste en declarar qué es lo que se desea alcanzar, cuando se define la estructura del problema en todos los niveles, se pone al decisor a realizar juicios en cada uno de ellos.
   - Nivel 2, Criterios: Son los puntos de vista considerados importantes para la consecución del propósito del problema.
   - Nivel 3, Alternativas: Son las opciones que podrían ayudar a la consecución del objetivo principal del problema.

   Para cada uno de los "m" criterios, se debe repetir los pasos 2 al 5.

2. Realizar la Matriz de Comparación por pares de Alternativas (MCP), para ellos se establece una matriz en donde se comparan los criterios, de manera que se pueda realizar el siguiente rating (Tabla **3-2**):

Tabla 3-2.: Matriz de Comparación por Pares

Valor	Estimación
1	Igualmente Preferida
3	Moderadamente Preferida
5	Fuertemente Preferida
7	Muy Fuertemente Preferida
9	Extremadamente Preferida

Se realiza un rating de 1/9, 1/7, 1/5, o 1/3 cuando la segunda alternativa es preferida a la primera y se aplica un valor de 1 cuando la alternativa se compara consigo misma. Con ello se crea una matriz cuadrada a_{ij} que valora la importancia del criterio i respecto al j. Se tiene la matriz de comparaciones:

$$A = \begin{pmatrix} 1 & a_{12} & \dots & a_{1n} \\ a_{21} & 1 & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & 1 \end{pmatrix}$$

3. Desarrollar la Matriz Normalizada: Se suman los valores de cada columna, de la siguiente manera $V_1 = \sum_{i=1}^{n} a_{i1}$, $V_2 = \sum_{i=1}^{n} a_{i2}$, ..., $V_n = \sum_{i=1}^{n} a_{in}$, luego se divide cada elemento de la matriz entre el total de la columna

$$A_{normalizada} = \begin{pmatrix} \frac{1}{v1} & \frac{a_{12}}{v2} & \dots & \frac{a_{1n}}{vn} \\ \frac{2}{v1} & \frac{1}{v2} & \dots & \frac{a_{2n}}{vn} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{a_{n1}}{v1} & \frac{a_{n2}}{v2} & \dots & \frac{1}{vn} \end{pmatrix}$$

4. Luego se obtiene El Vector de Prioridad para el Criterio, para ello se calcula el siguiente vector columna calculando el promedio de las filas, así:

$$p = \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n} a_{1j} \\ \frac{1}{n}\sum_{i=1}^{n} a_{2j} \\ \vdots \\ \frac{1}{n}\sum_{i=1}^{n} a_{nj} \end{pmatrix}$$

O lo que es lo mismo:

$$p = \begin{pmatrix} p_{c11} \\ p_{c12} \\ \vdots \\ p_{c1n} \end{pmatrix}$$

5. El método permite calcular un grado de consistencia entre las opiniones que da el decisor, si este es aceptable se puede continuar con el proceso de toma de decisiones, sino se deben cambiar los juicios que se hacen a los criterios. Este se calcula por el Cociente de Consistencia (CR), para hallarlo, se hace una analogía de los valores suponiendo que la alternativa 1 vale w_1 y la alternativa 2 vale w_2, se tiene que $a_{12} = w_1/w_2$, ahora reemplazando estos valores en la matriz A se tiene que:

$$A = \begin{pmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots & \frac{w_n}{w_n} \end{pmatrix}$$

Si en el caso ideal se considera la línea i de la matriz de juicios y se multiplica por los elementos w_1, w_2, ... w_n; se tiene que:

$w_i/w_1 * w_1 = w_i$
$w_i/w_2 * w_2 = w_i \ldots$
$w_i/w_j * w_j = w_i \ldots$
$w_i/w_n * w_n = w_i$

Pero en el caso real, es otra cosa ya que estos valores no son exactos, de todas formas, si se hace lo mismo esto da como resultado una dispersión estadística del juicio dado sobre el valor w_i; por lo tanto con los valores que se obtengan al realizar esta operación en el caso real es la estimativa del valor y se halla de la siguiente manera:

$$w_i = \frac{1}{n} \sum_{j=1}^{n} a_{ij} * w_j$$

Luego, si se tiene una matriz A con los juicios ideales y una matriz A' con los desvíos producidos en un caso real, y si se desea verificar si el nivel de consistencia es admisible se implica que existe un vector columna w de valores $(j = 1, 2, \ldots, n)$, donde $w_i/w_j = a_{ij}$ y $A * w = n * w$. Si $\sum \lambda_i = \sum a_{ii} = n$, se genera una matriz A' y se cumple que:

$$A' * w' = \lambda_{Max} * w'$$

y,

$$\lambda_{Max} \geq n$$

Si hay consistencia:

$$\lambda_{Max} = n = \sum_{i=1}^{n} a_{nj} \lambda_i$$

Cuanto más parecido sea λ_{Max} al número de alternativas más consistente será el juicio de valor elaborado. El desvío de la consistencia se expresa por:

$$IC = \frac{\lambda_{Max} - n}{n - 1}$$

El método calcula la razón de consistencia definido como la relación entre IC de A y IA que es el índice aleatorio y se generan con comparaciones por pares generadas al azar y se muestran en la Tabla **3-3** para diferente número de elementos comparados.

$$RC = \frac{IC}{IA}$$

Tabla 3-3.: Índice de Consistencia Aleatoria.

Número de elementos que se comparan	1	2	3	4	5	6	7	8	9	10
Índice de Consistencia Aleatoria (IA)	0	0	0,58	0,89	1,11	1,24	1,32	1.40	1,45	1,4

Se considera que la consistencia del decisor es aceptable cuando RC es menor al 10 %.

6. Luego los resultados del Vector de Prioridad para el Criterio son resumidos en la matriz Matriz de Prioridad (MP), listando las alternativas por fila y los criterios por columna.

$$\begin{array}{c} \\ Alternativa1 \\ Alternativa2 \\ \vdots \\ Alternativan \end{array} \begin{array}{c} \begin{array}{cccc} Criterio1 & Criterio2 & ... & Criterio3 \end{array} \\ \begin{bmatrix} p_{11} & p_{12} & p_{11} & p_{1m} \\ p_{21} & p_{22} & p_{11} & p_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nm} \end{bmatrix} \end{array}$$

7. Desarrollar una Matriz de Comparación por Criterios por pares como se hizo con las alternativas en los pasos (2), (3) y (4).

8. Se multiplican las matrices de Comparación por Criterios (7) por la matriz de prioridad de las alternativas (6); para obtener la matriz de prioridad global.

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nm} \end{pmatrix} \begin{pmatrix} p_{c11} \\ p_{c12} \\ \vdots \\ p_{c1n} \end{pmatrix} = \begin{pmatrix} p'_{11} \\ p'_{12} \\ \vdots \\ p'_{1n} \end{pmatrix}$$

El proceso se repite hasta terminar todas las comparaciones de los elementos del modelo, es decir, los criterios, subcriterios y alternativas. Luego se escoge la alternativa que obtenga un mayor puntaje en la matriz de prioridad global.

Este método de toma de decisiones es uno de los más utilizados actualmente, el inconveniente al compararlo con el que se eligió, el método multicriterio Scoring [62], es

que este conlleva un poco más de costo computacional al incrementar los criterios de decisión, ya que se deben realizar más matrices de comparaciones por pares para hallar la alternativa más adecuada.

Método Electre

El Método Electre se aplica a problemas que tienen un conjunto de alternativas discretas en el cual algunos de los criterios de evaluación no son cuantificables [15, 82], es decir, se encuentran en un rango o intervalo. Su principal objetivo es particionar el conjunto de soluciones entre alternativas más favorables y menos favorables.

El Método se apoya en tres conceptos claves, concordancia, discordancia, y valores de umbral. La concordancia entre dos alternativas i y j es una medida del peso del número de criterios por los cuales la alternativa i es preferida sobre la alternativa j ($i \succ j$) o por el cual la alternativa i es igual a la alternativa j ($i \sim j$). Hay que aclarar que esta preferencia no es transitiva, es decir, la alternativa i puede ser mejor que la alternativa j y esta a su vez mejor que la alternativa t, pero eso no significa que la alternativa i sea mejor que la alternativa t, ya que las razones de preferencia de la primera opción sobre la segunda, y las razones de preferencia de la segunda sobre la tercera pueden ser diferentes y no llevar a una preferencia de la tercera sobre la primera opción. La concordancia se denota como:

$$C(i,j) = \frac{\sum_{k \in A(i,j)} w(k)}{\sum_k w(k)}$$

Donde $w(k)$ es el peso del criterio k, $k = 1, \ldots, K$ y $A(i,j) = k \mid i \succeq j$, los pesos están dados por el decisor, y reflejan un conjunto de preferencias. La concordancia está entre 0 y 1.
La discordancia cuantifica hasta qué punto no existe ningún atributo en que una alternativa j sea mucho mejor que la alternativa i. Se requiere que se defina una escala de criterios común para todos ellos; esta escala se usa para comparar el malestar causado por el valor del peor y el mejor criterio para cada par de alternativas. Se denota como:

$$D(i,j) = \frac{\text{máx}_{k=1,K}(Z(j,k) - Z(i,k))}{R*}$$

Donde $Z(j,k)$ es la evaluación de la alternativa j con respecto al criterio k, y $R*$ es la más grande de las K escalas de criterios. La discordancia está entre 0 y 1.
El decisor debe definir un umbral p, q para los valores de concordancia y discordancia. Luego se crea una representación geométrica que define un grafo transitivo y completo para cada criterio, en el cual cada nodo es una alternativa y los arcos son dirigidos según la preferencia Figura **3-1**; en el caso de que las alternativas tengan igual preferencia, entonces un grafo irá de i a j y otro de j a i. El conjunto de arcos A del grafo el cual resume la relación entre la concordancia y discordancia se da por:

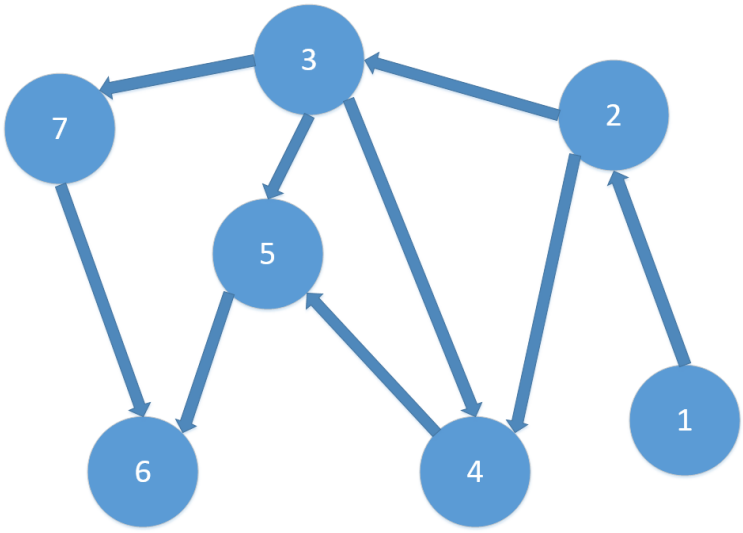

Figura 3-1.: Ejemplo de un grafo Electre

$$a(i,j) \in A \Leftrightarrow (C(i,j) > p) \cap (D(i,j) < q)$$

En este método puede darse que algunas elecciones del umbral eliminen todas las opciones, entonces tienen que reestablecerse estos valores. Este método tiene dos inconvenientes principales y es que no pueden ser utilizados para resolver problemas no convexos y las soluciones que se encuentran están en una cierta región de las soluciones Óptimas de Pareto.

3.2.3. Teoría de Juegos

Es un enfoque en el cual se estudia la manera en la cual se toman las mejores decisiones ante situaciones de conflicto. Las decisiones que tome un decisor debe tener en cuenta las decisiones que podrían tomar los otros decisores [70]. La teoría de juegos formaliza el estudio del conflicto y la cooperación, el juego se denomina como el modelo de una situación interactiva que incluye múltiples jugadores, los cuales basados en ciertas estructuras de incentivos, llevan a cabo toma decisiones que finalmente podrían afectar a otros jugadores [6]; en su ejemplo más fundamental que es el dilema del prisionero, muestra cómo en algunas situaciones, individuos deciden cooperar o no, logrando un beneficio incluso en la decisión de comportarse de manera egoísta. En [72] extienden el concepto del dilema del prisionero de dos individuos a un grupo mayor y en [63], postulan un algoritmo eficiente para la teoría del juego que satisfaga la condición de equilibrio de Nash, este consiste en una lista de estrategias, una para cada participante, en donde ninguno puede cambiar su estrategia unilateralmente para obtener una mejor utilidad.

Un juego se denomina formalmente de la siguiente manera $G = \{S_1, S_2, \ldots, S_n; u_1, u_2, \ldots, u_n\}$

y está formado por tres elementos:

- Los n jugadores que participan en el juego
- Las estrategias disponibles para cada jugador, que son: $S = S_1, S_2, \ldots, S_n$; donde $S_i = \{S_{i1}, S_{i2}, \ldots, S_{im}\}$ son las estrategias con las que cuenta cada jugador, a lo cual se conoce como perfil estratégico.
- El conjunto de funciones de pago $U = \{u_1, u_2, \ldots, u_n\}$, en donde la función de pagos de cada jugador es $u_i = u_i\{s_1, s_2, \ldots, s_n\}$, que es función de las estrategias elegidas por él y por los otros jugadores.

En una situación de conflicto entre varios jugadores, los mismos eligen sus estrategias en formas simultáneas, por lo cual cada jugador no conoce la estrategia de los demás jugadores. Cada jugador recibe un pago de $u_i = u_i\{s_1, s_2, \ldots, s_n\}$, dependiendo de las estrategias elegidas por los demás.

3.3. Negociación

Como se ha mencionado anteriormente, la parte definitiva sobre la decisión de ingreso consiste en una negociación sobre los recursos que el dispositivo y la red están dispuestos a compartir con el otro, al final debe haber un acuerdo entre la cantidad de los mismos y para ello se aplicó un método de negociación al sistema; por lo que ahora se van a observar algunas definiciones sobre el tema.

El problema en una negociación se refiere a la elección de una alternativa cuando hay varios actores que están en conflicto de intereses; pero todo se desarrolla en un marco de cooperación. Al final es un problema de consenso en donde la decisión final solo puede ser tomada cuando las partes en disputa están de acuerdo; existe la posibilidad de que haya una amenaza cuando ambas partes fracasan en alcanzar un acuerdo. Por lo cual las decisiones tienen que ser tomadas teniendo en cuenta el punto en disputa.

En la teoría de negociación lo más importante son las combinaciones de la utilidad de las partes en conflicto, en este caso, el nodo y el sistema, es decir las ganancias de cada uno de ellos sobre el punto de partida de la negociación o el punto llamado status quo. En la solución de negociación de Nash [26] y [54], se plantea un juego realizado entre dos jugadores y se modela lo que pasa cuando ambas partes actúan en forma racional, incluso cuando el resultado de la salida no es justo para uno de ellos; este juego consiste en negociar un bien cuando ambos jugadores están de acuerdo en realizarlo.

Formalizando, sea S un conjunto de vectores de utilidad, los puntos u_1^* y u_2^* las utilidades de los jugadores si estos no se ponen de acuerdo o punto de status quo, y la salida negociada como puntos $\bar{u}$, y $\bar{v}$. Entonces el objetivo sería encontrar una función la cual realice el mapeo de los posibles puntos de utilidad de no acuerdo y los puntos de vectores de utilidad hacia los puntos de la salida negociada $(\bar{u}, \bar{v}) = f(S, u^*, v^*)$; Para ello Nash definió una serie de axiomas que esa función debe cumplir para que limiten el comportamiento de la función deseada, que tiene la forma $(\bar{u}, \bar{v}) = max_{(u,v)\in S}(u - u^*)(v - v^*)$.

- Axioma 1. Invariancia con respecto a las transformaciones de la utilidad: es decir, si los actores de las negociaciones transforman sus escalas de utilidades de manera independiente, las coordenadas de la solución cambiarán en las mismas escalas de transformación de utilidad.
- Axioma 2. Eficiencia de Pareto: la salida negociada debe estar en la frontera de Pareto de la región de utilidades, es decir, debe ser al menos tan buena como el status quo, y no puede haber otro punto de negociación mejor que la salida negociada.
- Axioma 3. Simetría: Si hay dos jugadores cuyos roles son exactamente los mismos, es decir, no hay diferencia entre su status quo y sus posibles distribuciones de utilidad; la solución negociada tendrá que ser igual para ambos jugadores.
- Axioma 4. Independencia de las alternativas irrelevantes o consistencia de la contracción: La solución para T debe ser también la solución para S, si $T \subset S$. Es decir, si hay dos diferentes juegos de negociación y ambos tienen el mismo punto de status quo y si los puntos de negociación de un juego están incluidas en el otro, y además si el valor negociado del juego con mayor conjunto de alternativas es realmente un punto factible de negociación en el juego más pequeño, entonces debe ser la salida negociada para ese juego.

En el principio de concesiones alternativas de Zeuthen que se detalla en [26, 16]; Zeuthen propuso un procedimiento en el que las partes involucradas en la negociación hacen ofertas de forma alternada. Este método de negociación constituye una solución a la negociación de Nash e incluye elementos de la teoría de juegos no cooperativos, este método se verá con mayor detalle en el siguiente capítulo con un escenario aplicado al módulo que se está construyendo.

Luego, en [16], el autor en su solución de negociación, se propone el reemplazo del axioma 4 de independencia de Nash por un axioma de monotonicidad de manera que si hay una expansión en la región de utilidad y además si para el jugador 1 el nivel de utilidad permanece constante, el nivel de utilidad del jugador 2 debe aumentar ligeramente.

En lo que sigue, se van a observar algunos modelos de negociación.

3.3.1. Modelo de Negociación de Stahl

Es una de las versiones más simples de los métodos de negociación y se utiliza como base para realizar modelos más completos [55, 54], este método consiste en realizar ofertas alternativas, realizando descuentos sucesivos en un factor constante $0 < \delta_i < 1$ en cada periodo, pero hay un número finito de ofertas. Se utiliza para dividir un objeto el cual se normaliza a 1 para mayor simplicidad. Esta negociación se dá en un máximo de T periodos, t es el tiempo de una oferta de cada parte, si $t = T$ entonces la negociación termina y cada uno de los jugadores recibe la parte del objeto negociado. El proceso de negociación se puede definir en los siguientes pasos.

- Si t es un periodo impar el jugador 1 es el que ofrece, en este caso una repartición de $(x(t), 1 - x(t))$, donde $x(t)$ es la porción que el jugador 1 propone para sí mismo y $1 - x(t)$ es la porción que propone para el jugador 2.
- Si el jugador 2 acepta la oferta, el juego termina con un resultado de $(x, 1 - x)$.
- Si el jugador 2 rechaza la oferta, se pasa al siguiente periodo $t + 1$.
- Si t es par el jugador 2 ofrece una repartición de $(x(t), 1-x(t))$, donde $x(t)$ es la porción que el jugador 2 propone para sí mismo y $1 - x(t)$ es la porción que propone para el jugador 1.
- Si el jugador 1 acepta la propuesta, el resultado termina con un resultado de $(x, 1 - x)$
- Si el jugador 1 rechaza la oferta, se pasa al siguiente periodo $t + 1$.
- El juego continua con el jugador 1 realizando propuestas en los periodos impares y el jugador 2 realizando propuestas en los periodos pares, en cada periodo, se realiza un descuento δ, en un periodo de tiempo t se realiza un descuento $\delta^t - 1$, al término de la negociación las negociaciones se pueden resumir con la siguiente ecuación.

$$(\delta^{t-1}x(t), \delta^{t-1}(1 - x(t)))$$

Ahora se analizará los diferentes casos para diferentes valores de T.

- Si $T = 1$, se tiene un caso que se llama el Juego de Ultimatum, este caso es un juego de tómalo o déjalo, así que el jugador 2 acepta cualquier propuesta
- Para $T = 2$, en este caso cuando $t = 2$ todo el excedente irá al jugador 2 y en $t = 2$ el jugador propondría $(0, 1)$, hay que tener en cuenta el factor de descuento δ en esta oferta.
- Para $T = 3$, se realiza el mismo análisis de los periodos 2 y 3 es el mismo que se hizo con el modelo en $T = 2$, la propuesta óptima del jugador 1 es $(1 - \delta_2, \delta_2)$.

Al realizar inducción hacia atrás se observa que el jugador 2 preferirá quedarse con un valor $1-x_2 = 1-\delta$; en el periodo 1 la estrategia del jugador 2 será aceptar cualquier oferta de tal forma que $1-x_1 \geq \delta(1-\delta)$, en el primer periodo el jugador 1 propone entonces $x_1 = 1-\delta+\delta_2$.

Para 5 periodos el jugador 1 obtiene $1-\delta_2+\delta_1\delta_2(1-\delta_2)+\delta_1\delta_2$

Para $2k$ periodos el jugador 1 obtiene $1-\delta_2\left[\frac{1-(\delta_1\delta_2)^k}{1-(\delta_1\delta_2)}\right]$

Para $2k+1$ periodos el jugador 1 obtiene $1-\delta_2\left[\frac{1-(\delta_1\delta_2)^k}{1-(\delta_1\delta_2)}\right]+(\delta_1\delta_2)^k$

3.3.2. Modelo de Negociación de Rubinstein

En este modelo de negociación, se consideran dos jugadores realizando una negociación con ofertas alternativas durante un periodo de tiempo infinito [55, 54], es una extensión del modelo de Stahl, aquí el jugador 1 realiza una oferta en el primer periodo, si el jugador 2 la rechaza, el juego pasa al segundo periodo donde el jugador 2 realiza la oferta, si el jugador 1 rechaza la oferta, entonces el juego pasa al tercer periodo y así sucesivamente, el juego termina hasta que un jugador acepta la oferta del otro. La estrategia de un jugador en este modelo involucra. Cada jugador enfrente la misma estrategia en cada periodo, las ganancias son diferentes pero las preferencias son las mismas, es una extensión del método anterior para un número de periodos infinito. Se define entonces:

$$x_1^* = \frac{1-\delta_2}{1-\delta_1\delta_2}, \qquad x_2^* = \frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}$$
$$y_1^* = \frac{1-\delta_1}{1-\delta_1\delta_2}, \qquad y_2^* = \frac{\delta_1(1-\delta_2)}{1-\delta_1\delta_2}$$

El perfil de estrategia (s_1^*, s_2^*) es el siguiente:

- El jugador 1 propone x^* y aceptará y si y solo si $y_1 \geq y_1^*$.
- El jugador 2 propone y^* y aceptará x si y solo si $x_2 \geq x_2^*$.

Se puede observar que este juego tiene dos tipos de subjuegos, el primero movimiento del primer jugador es una oferta y el segundo en el cual el primer movimiento es una respuesta a una oferta.

- Para el primer tipo de estrategia se supone que el jugador 1 hace la primera oferta
  - El jugador 2, establece su estrategia como s_2^*
  - Si el jugador 2 acepta la propuesta de 1, el jugador 1 obtiene x_2^*
  - Si la oferta del jugador 1 es mayor que x_2^*, el jugador 2 acepta llevando a una ganancia más baja que x_1^* para el jugador 1.

  - Si la oferta del jugador 1 es menor que x_2^*, el jugador 2 la rechaza y ofrece y^*, luego el jugador 1 acepta, lo que conlleva a una ganancia de $\delta_1 y_1^*$. Y ya que $\delta_1 y_1^* < x_1$, es mejor para el jugador 1, utilizar la estrategia s_1^*

- Para el segundo tipo de estrategia se supone que el jugador 1 responde a la oferta del jugador 2
  - Se establece la estrategia del jugador 2 como s_2^*
  - La oferta con la cual el jugador 1 responde es (y_1, y_2)
  - Si el jugador 1 adopta la estrategia s_1^*, el aceptará la oferta si $y_1 > y_1^*$
  - Si el jugador 1 rechaza alguna oferta $y_1 \geq y_1^*$, el jugador 1 tendrá $\delta_1 x_1^* = y_1$, de tal manera que él no podrá incrementar su ganancia por desviación.

4. Modelo Aplicado al Sistema Distribuido

El primer paso hacia la social-inspiración en el sistema distribuido TLÖN [85] es brindarle al nodo la capacidad de racionalidad de manera que enfrente el dilema de pertenecer al sistema distribuido de manera similar a como una persona lo haría al tener que tomar la decisión sobre su ingreso a una organización social que cuenta con unos beneficios específicos. Este sistema contaría con justicia, el cual es un principio orientador que lo distinguiría de otros sistemas distribuidos, en donde a los nodos que más aportan recursos con relación a sus recursos disponibles, se les asignan más a la hora de necesitarlos, y viceversa; a los nodos que menos colaboren con recursos, de manera proporcional se les restringirá la ayuda a la hora que la necesiten. Esto sería un gran incentivo ya que los nodos además del hecho de poder acceder a los recursos heterogéneos con los que cuenta una red ad hoc, podrían mejorar su reputación; con lo cual podrán tener aún más posibilidades y prioridad a la hora de acceder a recursos del sistema.

Existen dos escenarios principales a saber en los cuales un nodo puede estar inmerso al tomar la decisión de acceder al sistema distribuido TLÖN:

- Unión de nodo con otro nodo solitario: En este caso, el nodo se encuentra solo y se comportaría como un nodo Dios, es decir, el nodo sería el encargado de formar el sistema distribuido sobre la red ad hoc; de ahí en adelante los demás nodos buscarían hacer parte de ese sistema.

- Nodo con un sistema distribuido ya constituido: Es el caso en donde el nodo se encuentra con un sistema el cual ya está constituido por mínimo dos nodos.

En el proceso de aceptación que tiene una sistema ante un nodo solitario, se debe tener cierta reserva a la hora de la admisión para evitar el hecho de que se acepte a un nodo egoísta, que vaya a aprovecharse de los recursos actuales del sistema y no aporte nada. Este efecto se puede observar en la teoría del nodo egoísta que se expone en [78].

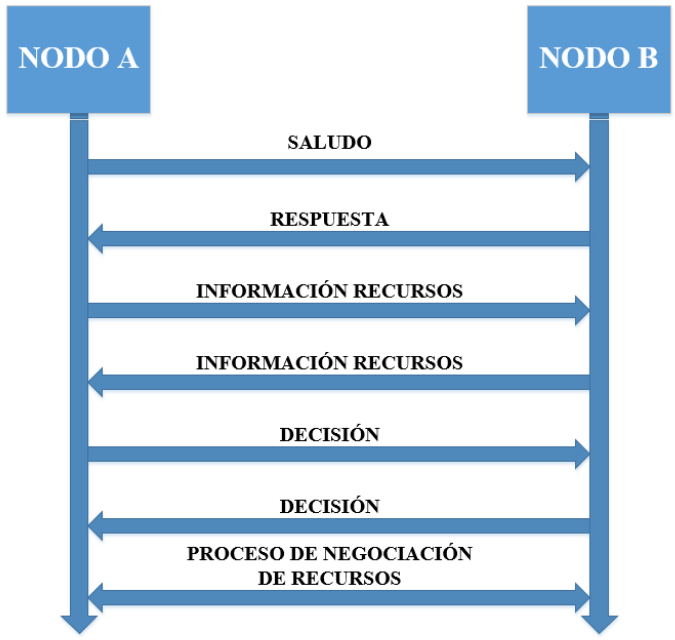

Figura 4-1.: Proceso de Formación del sistema distribuido, fuente propia

4.1. Unión de Nodo con Otro Nodo Solitario

En este escenario, una de las principales razones para la unión con otro nodo es ganar la reputación de haber sido uno de los nodos que creo el Sistema Distribuido sobre la red ad hoc, y por ende una de las principales características que posee el nodo es el altruismo desde una perspectiva sociológica como se define en [5], es decir, el hecho de dar de sí lo más posible, para aumentar su estatus e integración social. Para el ingreso se realiza el proceso que se observa en la Figura **4-1**.

Para ello, el nodo A estará escuchando mensajes de saludo de otros nodos buscando siempre el proceso de unión; en este caso, al encontrar un nodo, el primer paso que realiza es un saludo que va del nodo A hacia el nodo B indicando que es un nodo solitario, inmediatamente, el nodo B le envía una respuesta de la misma manera indicando su estado; una vez el nodo A recibe la respuesta, el otro nodo envía una solicitud de unión de tal manera que el nodo A se prepare para recibir la información de sus recursos, es decir, su capacidad de disco duro, Ram, procesamiento, disponibles y totales, además de sus periféricos como teclado, cámaras, micrófonos, etc. Y el nodo A, de manera similar enviará su propia información de recursos; en seguida, mediante un proceso de toma de decisiones multicriterio llamado Scoring [62] cada nodo decide si continuarán con el siguiente paso que consiste en la negociación de los recursos que cada uno va a aportar a la red; si hay un acuerdo, se creará el sistema distribuido y cada nodo se configurará de tal manera que estará escuchando solicitudes de unión

de otros nodos que quieran hacer parte del sistema.

4.2. Unión de nodo con un sistema distribuido

El segundo caso es cuando un nodo se encuentra a un sistema distribuido ya conformado; en este caso el nodo se comportaría como una persona al intentar ingresar a una organización o institución; y el sistema distribuido como el grupo que debe aceptar o no a un potencial integrante. El sistema debe tener en cuenta el comportamiento del nodo a la hora de dejarlo ingresar, ya que este se puede comportar de una manera egoísta y no aportar recursos suficientes, esto debe ser claro ya que este sistema debe velar por los recursos de cada uno de sus integrantes y compartirlos de una manera justa [56].

El proceso de ingreso es similar al de la Figura **4-1**, pero hay unas pequeñas variaciones. De igual manera el nodo estará escaneando el espectro, y recibirá un saludo por parte del nodo más próximo que esté dentro del sistema distribuido. Los nodos que hacen parte del sistema, están atentos a solicitudes de unión de tal manera que el nodo indicará su intención de realizar el proceso de negociación e inmediatamente enviará la información de sus recursos, el nodo anfitrión enviará la información de recursos disponibles en el sistema y cada parte realizará con base en el método multicriterio Scoring [62] la primera decisión de ingreso a la red, si ambas partes acceden a realizar la negociación, se utilizará el método de negociaciones de Zeuthen [26, 16] que se detalla posteriormente; si hay un acuerdo, el nodo ingresará a la red.

Mediante el método Scoring [62], la decisión se toma realizando una comparación de los recursos que tienen los dos nodos, si es de mayor utilidad para el nodo crear el sistema con su potencial vecino, lo hará. El método se muestra a continuación.

4.3. Método de decisión Multicriterio discreto Scoring

Este método es una manera rápida y sencilla para realizar una elección en un problema de decisión, El proceso para realizar una decisión multicriterio con este método es el siguiente [62]:

- Identificar el propósito del problema: Este paso consiste en identificar qué es lo que se desea alcanzar.

- Identificar las alternativas: Son las opciones que podrían ayudar a la consecución del propósito que se propuso en el paso 1.

- Listar los criterios los cuales van a ser tenidos en cuenta para elegir la mejor alternativa. En nuestro caso podrían ser cada uno de los recursos con los que cuenta el dispositivo y la red. Por ejemplo, Ram, disco duro, periféricos, sensores, etc.

- Asignar una ponderación para cada uno de los criterios, es decir, indicar qué tan importante es para el decisor el criterio en cuestión mediante la siguiente escala de 1 a 5 (Tabla **4-1**).

Tabla 4-1.: Asignación Importancia.

Ponderación	Importancia
1	Muy poco importante
2	Poco importante
3	Importancia media
4	Algo importante
5	Muy importante

- Indicar, para cada alternativa, en cuanto se satisface cada uno de los criterios.

- Calcular el Score para cada una de las alternativas, se puede hacer de la siguiente manera:

$$S_j = \sum_j w_i r_{ij}$$

Donde:

r_{ij}: Valor de la alternativa j para el criterio i
w_i: Ponderación de cada criterio
S_j: Score para cada alternativa j

Para el caso de toma de decisiones que se está manejando, se podría dejar de la siguiente manera.

1. El propósito del problema es decidir si ingresar o no al sistema distribuido, y obtener mejores recursos o reputación en la red, esto se hace dependiendo de los recursos que le ofrezca la red comparando la utilidad que estos le generen con la del escenario en el cual el nodo permanece solo.

2. Las alternativas son: ingresar a la red o no hacerlo (Si y No).

3. Criterios: Recurso 1, Recurso 2, Recurso 3, Recurso 4.

4. Asignación de la ponderación (Tabla **4-2**):

Tabla 4-2.: Asignación de la ponderación.

Criterio	Ponderación
Recurso 1	w_1
Recurso 2	w_2
Recurso 3	w_3
Recurso 4	w_4

5. Indicar en cuanto satisface la ponderación anterior a cada alternativa, (Tabla **4-3**).

Tabla 4-3.: Ponderación de cada alternativa.

Criterio	Ponderación (w_i)	Si (r_{i1})	No (r_{i2})
Recurso 1	w_1	r_{11}	r_{12}
Recurso 2	w_2	r_{21}	r_{22}
Recurso 3	w_3	r_{31}	r_{32}
Recurso 4	w_4	r_{41}	r_{42}

6. Calcular la ponderación para cada alternativa (Tabla **4-4**):

Tabla 4-4.: Cálculo Scoring para cada alternativa.

Criterio	Ponderación w_i	Si (r_{i1})	No (r_{i2})	Ponderación Si ($w_i * r_{i1}$)	Ponderación No ($w_i * r_{i2}$)
Recurso 1	(w_1)	r_{11}	r_{12}	$w_1 r_{11}$	$w_1 r_{12}$
Recurso 2	w_2	r_{21}	r_{22}	$w_2 r_{21}$	$w_2 r_{22}$
Recurso 3	w_3	r_{31}	r_{32}	$w_3 r_{31}$	$w_3 r_{32}$
Recurso 4	w_4	r_{41}	r_{42}	$w_4 r_{41}$	$w_4 r_{42}$
Score S_j				$S_1 = \sum_1 w_i r_{i1}$	$S_2 = \sum_2 w_i r_{i2}$

Para la decisión final, se decidió incluir un indicador de altruismo-egoísmo el cual influye en el comportamiento del nodo haciéndolo más egoísta o más altruista, esto por el hecho de que uno de los objetivos del grupo de investigación TLÖN es añadir el

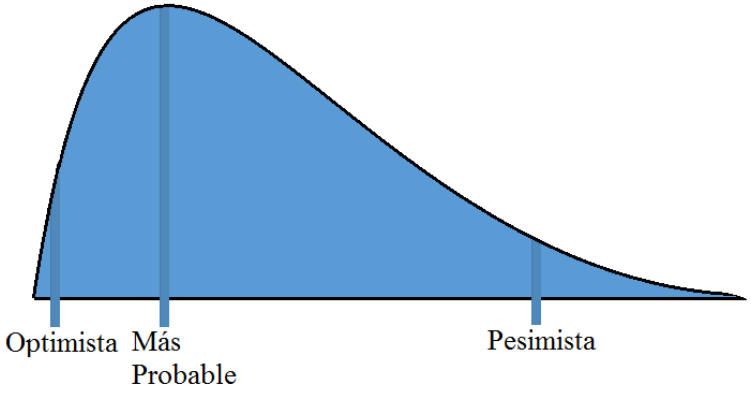

Figura 4-2.: Distribución Beta

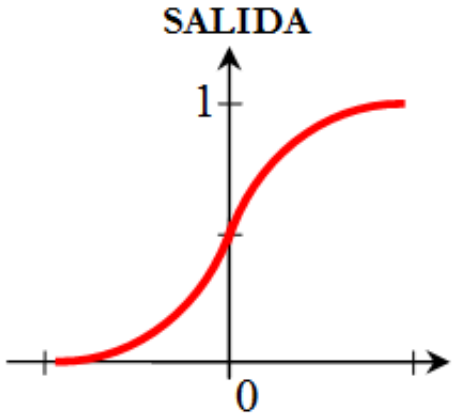

Figura 4-3.: Función de activación Sigmoidal

comportamiento social-inspirado a esta clase de sistemas, y esto influenciará al nodo, dependiendo del ambiente en el que se encuentre a ingresar y a aportar más o menos recursos, esto se basa en lo indicado por [22] y que se observa en la sección 2.4.1 que habla sobre el tipo de cooperación e incentivos en una Mobile Cloud. Esto se realizó mediante una distribución Beta que simula el deseo de colaborar del nodo y como se observa en la Figura **4-2**, hace que el nodo tenga más posibilidades de ser altruista, esto por tratarse de un ambiente de amigos o colegas de trabajo.

Finalmente para incluir este índice y los pesos de las decisiones se introducen estos valores en una función de activación sigmoidal **4-3** y dependiendo de la salida de la función, si es 1, el nodo ingresa, si es 0, el nodo no lo hace.

Para la función de activación se propuso la siguiente ecuación:

$$Out = 1/(1 + e^{(-(S1-S2*Ia))}) \tag{4-1}$$

Donde:
Out: Salida, si es 1 ingresa, si es 0 no lo hace
$S1$: Scoring Si
$S2$: Scoring No
Ia: Índice de Altruismo-Egoísmo

Después del proceso anterior, el nodo debe esperar la decisión que tome el otro nodo o la red, y si es afirmativa también, empiezan el proceso de negociación de recursos; es decir, deben ponerse de acuerdo en la cantidad de recursos que cada uno va a aportar al sistema distribuido.

Para la negociación de los recursos (Figura **4-4**), el nodo transmite una propuesta tentativa, esta elección la hace teniendo en cuenta la cantidad de recursos disponibles y su deseo de mejorar su reputación en la red. Luego, el sistema distribuido, con base en la teoría social de negociación, toma una decisión, si los recursos que está ofreciendo el nodo son suficientes la red le envía un mensaje de aceptación para que haga parte de la red, de lo contrario, envía una contrapropuesta indicándole al nodo que colabore con una cantidad adicional de recursos para que pueda ser aceptado, si el nodo está de acuerdo con esa contrapropuesta, envía un mensaje de aceptación e ingresa a la red, de lo contrario hará una contrapropuesta; si hay un acuerdo, el nodo se incorpora a la red, sino el nodo seguirá escaneando el espectro en busca de otros nodos y sistemas para realizar el mismo proceso. Cada nodo con los que no alcanza un acuerdo, se incluirá en una lista negra, esto con el fin de evitar que los nodos entren en un bucle de negociaciones sucesivas.

Para el método de negociación cooperativa que se plantea que realicen los nodos para ponerse de acuerdo sobre la cantidad de recursos que se aportarán al sistema, se analiza el método de concesiones alternativas de Zeuthen [26, 16].

4.4. Principio de Concesiones Alternativas de Zeuthen

El método de Zeuthen [26, 16], está basado en la negociación en el mercado laboral. Para la formalización, se puede suponer lo siguiente. Digamos que existen el jugador 1 y el jugador 2; los cuales quieren llegar a un acuerdo sobre la cantidad de un ítem que les corresponde a cada uno. El jugador 1, propone $x = (x_1, x_2)$ y el jugador 2, propone $y = (y_1, y_2)$, donde el primer componente se refiere a la propuesta para el jugador 1. Se propone un caso de no

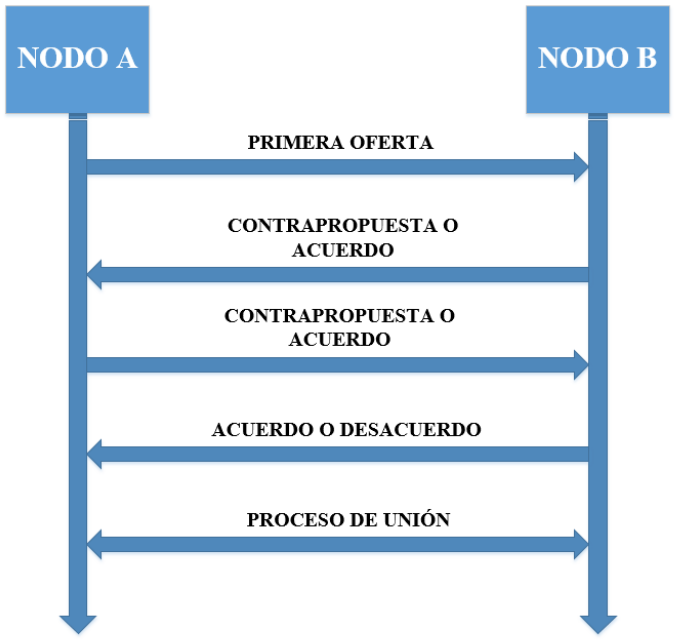

Figura 4-4.: Proceso de Negciación de recursos, fuente propia

acuerdo, en donde, en términos de utilidades, el jugador 1 recibiría $u_1(x_{01})$ y el jugador 2 $u_2(x_{02})$, donde $x_o = (x_{01}, x_{02})$ sería la situación de conflicto. Se asume que:

$u_1(x_{01}) < u_1(y_1) < u_1(x_1)$ y,
$u_2(x_{02}) < u_2(x_2) < u_2(y_2)$

Lo que quiere decir que el jugador prefiere su propuesta a la del otro jugador, pero prefiere estas dos situaciones a la situación de conflicto.

El procedimiento pues, para las situaciones a las que se enfrentan ambos jugadores, sería el siguiente. Digamos que x es la última propuesta del jugador 1 y y la última propuesta del jugador 2. Suponiendo que ambos jugadores son maximizadores bayesianos de la utilidad, entonces se puede suponer que p_{12} es una probabilidad subjetiva que el jugador 1 asigna a que el jugador 2 se apegue a su última oferta, es decir y. Y $(1-p_{12})$ la probabilidad subjetiva de que el jugador 2 aceptará la oferta hecha por el jugador 1.

Si el jugador 1 acepta la última oferta del oponente, el jugador 1 recibirá $u_1(y_1)$. Si el jugador 1 se apega a su última oferta personal, el jugador 1 obtendrá $u_1(x_1) > u_1(y_1)$ con una probabilidad de $(1-p_{12})$, pero también podría obtener la utilidad mínima $u_1(x_{01}) < u_1(y_1)$, con una probabilidad p_{12}. De manera que si se quiere maximizar la ganancia, el jugador 1, se apegaría a su última oferta x, si y solo si:

$(1 - p_{12}).u_1(x_1) + p_{12}.u_1(x_{01}) \geq u_1(y_1)$
Que es igual a:

$$p_{12} \leq \frac{(u_1(x_1) - u_1(y_1))}{(u_1(x_1) - u_1(x_{01}))} \tag{4-2}$$

Esta relación se llama límite de riesgo r_1, es el riesgo máximo que el jugador 1, está dispuesto a enfrentar para establecer un arreglo de acuerdo a su propia propuesta más que en los términos que le ofrece su oponente.

Para ello, el autor propone las siguientes reglas de decisión para saber cuál de los dos jugadores realizará la primera concesión, es decir, quién va a ceder un poco de su utilidad para poder lograr un acuerdo entre las dos partes.

- Si $r_1 > r2$, entonces el jugador 2 tiene que hacer la siguiente concesión.
- Si $r_1 < r2$, entonces el jugador 1 tiene que hacer la siguiente concesión.
- Si $r_1 = r2$, entonces ambos jugadores tienen que ceder un poco.

Digamos que el jugador 1 propone $x = (x_1, x_2)$, mientras que el jugador 2 propone $y = (y_1, y_2)$, entonces:

$$r_1 = \frac{(u_1(x_1) - u_1(y_1))}{(u_1(x_1) - u_1(x_{01}))} \leq \frac{(u_2(y_2) - u_2(x_2))}{(u_2(y_2) - u_2(x_{02}))} = r_2 \tag{4-3}$$

O,
$(u_1(x_1) - u_1(x_{01})).(u_2(x_2) - u_2(x_{02})) \leq (u_1(y_1) - u_1(x_{01})).(u_2(y_2) - u_2(x_{02}))$
De acuerdo a las reglas de Zeuthen, r_1 tendría que hacer la concesión, si se especifica que la nueva concesión es $x' = (x'_1, x'_2)$, de tal manera que:

$$r'_1 = \frac{(u_1(x'_1) - u_1(y_1))}{(u_1(x'_1) - u_1(x_{01}))} \leq \frac{(u_2(y_2) - u_2(x'_2))}{(u_2(y_2) - u_2(x_{02}))} = r'_2 \tag{4-4}$$

O,
$(u_1(y_1) - u_1(x_{01})).(u_2(y_2) - u_2(x_{02})) \leq (u_1(x'_1) - u_1(x_{01})).(u_2(x'_2) - u_2(x_{02}))$

Como se puede observar, en la primera productoria de Nash del jugador 1, es menor a la productoria del jugador 2, por lo cual el jugador 1 realiza la primera concesión, luego, en la siguiente ronda, la productoria de Nash del jugador 2, es menor que la del jugador 1, por

lo cual, éste debe hacer la siguiente oferta. En cada ronda, se producirá una productoria de Nash más alta, así sucesivamente hasta que en algún momento, algún jugador haga una oferta con el valor más alto posible del producto de Nash. Este punto sería el que aceptarían los dos jugadores y en el cual se obtendría un acuerdo.

5. Resultados

Para validar los resultados, se hizo la implementación del experimento mediante Python, en dos familias de dispositivos: un computador portátil y tres dispositivos embebidos Raspberry Pi. Cuyos requerimientos mínimos de software son:

- Versión 2.6.9 de Python o superior.
- Se usó la versión 2.7.0 de Raspbian, que es el sistema operativo de la Raspberry, con Kernel de Linux 4.9.
- Batman-Adv y Batctl, versiones 2016.4.

Las pruebas se realizaron con los siguientes dispositivos:

- Raspberry 1
  ARMv6-compatible processor rev 7 (v6l)
  8GB de disco duro
  500 MB de RAM
  Adaptador Wi-Fi, GPIO HDMI, 4 puertos USB, teclado y mouse
- Raspberry 2
  ARMv7 Processor rev 5 (v7l)
  16GB de disco duro
  1GB de RAM
  Adaptador Wi-Fi, GPIO HDMI, 4 puertos USB
- Raspberry 3
  ARMv6-compatible processor rev 7 (v6l)
  8GB de disco duro
  500 MB de RAM
  Adaptador Wi-Fi, GPIO HDMI, 4 puertos USB
- Computador marca Dell
  Procesador Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz
  51GB de disco duro disponibles
  4GB de RAM

Teclado, mouse, pantalla, puertos USB, adaptador Wi-Fi y demás periféricos de un computador portátil

En el escenario de pruebas no se tuvieron en cuenta algunos parámetros de eficiencia de la red como los tiempos, la distancia entre los dispositivos ni la escalabilidad. Los objetivos estaban orientados principalmente a que el módulo funcionara.

El objetivo general de la tesis fue dividido en cinco grandes submódulos ampliamente relacionados entre sí. Por ejemplo, gran parte de ellos utilizan el programa de comunicaciones en capa 3, que hace uso del protocolo IPV6 para el envío de instrucciones e información entre nodos y todos son ejecutables recibiendo instrucciones de nivel superior, como se fijó en el primer objetivo de la tesis. Las pruebas fueron exitosas teniendo en cuenta que lo importante era la funcionalidad mas no el tiempo de ejecución, el cual es un poco elevado por sincronización en algunas partes del programa. Se mostrarán los diagramas que esbozan el funcionamiento general de cada submódulo y pruebas basadas en el eje central de la tesis que es la toma de decisiones y la negociación.

5.1. Módulo de Recepción y Envío de Solicitudes de Unión

Para el segundo objetivo de la tesis se creó este submódulo que es el encargado de enviar y recibir mensajes UDP a los nodos vecinos, principalmente anunciando su existencia, pero también indicando si el nodo es solitario o hace parte de un sistema distribuido. La Figura **5-1** muestra los diferentes programas que permiten al nodo cumplir esa función.

Inicialmente, se tiene un bloque llamado TLON_Adhoc, que es el demonio que se ejecuta con el encendido del dispositivo. Esto garantiza que el nodo pueda crear la red ad hoc y empiece a enviar los mensajes sin ninguna intervención por parte del usuario. En el bloque de creación de la red ad hoc (CreateAdhoc.py), el nodo tiene en cuenta la MAC del dispositivo para crear un nombre de red, que es el que irradia cada dispositivo cuando se configura en modo ad hoc, es decir, el ESSID. Este nombre será utilizado en la creación del nombre del sistema distribuido y tendrá en cuenta el nombre del nodo dios para construirlo. El nodo dios es el encargado de crear el sistema distribuido y de la unión del primer nodo, además actúa como servidor y recibe información de recursos y otros datos del proceso de negociación, lo cual se detallará más adelante. El programa CreateAdhoc.py crea una red ad hoc que funciona mediante el protocolo B.A.T.M.A.N en la capa dos del modelo OSI y establece comunicación con los otros nodos para el intercambio de información e instrucciones sobre el protocolo IPV6 en capa 3. Pero lo desarrollado en este proyecto tiene que ver con la capa 4 ya que los mensajes de saludo son enviados en datagramas UDP ya que no importan si son recibidos o no por los nodos que se encuentren en el área. Mientras que la comunicación que

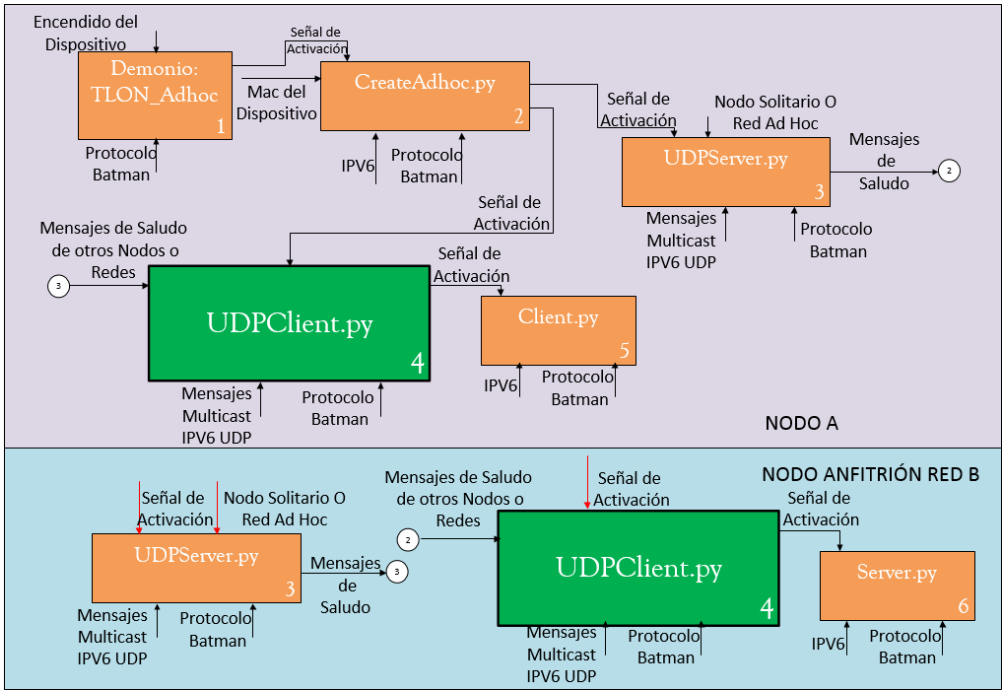

Figura 5-1.: Envío y Recepción de Mensajes de Unión

se realiza posteriormente para el intercambio de información específica para los procesos de toma de decisiones y negociación tienen que asegurar la llegada a su destino por lo que se utilizaron datagramas TCP.

Una vez se ha establecido el modo ad hoc en los nodos, CreateAdhoc.py envía mensajes de activación a los programas encargados de enviar y recibir mensajes de saludos hacia y desde otros nodos. El programa servidor de mensajes UDP (UDPServer.py) se encarga de consultar el estado del nodo para enviar mensajes indicando si es un nodo solitario o pertenece a un sistema distribuido. El programa cliente de mensajes UDP (UDPClient.py) escucha los mensajes que envía el servidor UDP desde otros nodos para el procesamiento de los saludos. Además este programa se encarga de consultar si se ha atendido anteriormente o no, pues no tiene sentido que el nodo se involucre en un bucle y resuelva las solicitudes del mismo nodo todo el tiempo. Por esto consulta su base de datos y descarta los nodos que ya fueron procesados.

Cuando se recibe un mensaje de saludo desde un nodo, se verifica si este hace parte de un sistema distribuido. Si es así, el nodo se configura como cliente de una conexión cliente-servidor y se le envía un saludo específico para establecer una conexión directa, realizar un intercambio de información y que el nodo proceda a realizar la toma de decisiones. Si el mensaje que recibe el nodo proviene de un nodo solitario, ambos se ponen de acuerdo sobre cuál va a ser el servidor y cuál va a ser el cliente, y el servidor tomará el rol de nodo dios,

que ante todos los nodos será el que creó el sistema distribuido.

En las pruebas realizadas, el proceso de elección de un nodo dios fue exitoso. Este se realizó teniendo en cuenta los dos últimos octetos de la dirección IPV6 del nodo. Dado que la asignación de esta dirección es aleatoria por el protocolo APIPA, se aprovechó este hecho ya que no era justo que el nodo dios fuera el que tuviera más recursos y resultaba mejor un proceso de elección de este tipo.

5.2. Toma de Decisiones y Negociación

Para la siguiente etapa de la unión del nodo al sistema distribuido que se relaciona con los objetivos tres y cuatro de la tesis, se creó el submódulo que se observa en la Figura **5-2**. Aquí el nodo se encontrará con dos escenarios; en el primero el nodo se une a un sistema distribuido ya formado y en el otro el nodo crea el sistema distribuido con otro nodo.

El programa principal crea una base de datos que será la encargada de almacenar toda la información concerniente al sistema distribuido, como recursos del nodo, información de recursos de otros nodos, información de la toma de decisiones, registro de eventos que se presenten en la ejecución del programa, entre otros. El formato de almacenamiento de la información es JSON, lo que la hace fácil de acceder y muy ligera. Además, es fácil de transferir entre nodos, por ejemplo en el proceso de toma de decisiones y negociación.

Antes de iniciar el proceso, el nodo debe reunir la información de los recursos que tiene disponible, la cual se compone de datos sobre la cantidad de disco duro, RAM, procesador, dispositivos de entrada y salida y acceso a internet. Esta información se guarda en un archivo independiente de la base de datos principal, para un mejor intercambio a la hora de realizar el proceso de ingreso.

Como se observa en la Figura **5-2** el programa cliente del nodo (Client.py) se encarga de administrar todas las labores del ingreso del nodo a la red. Cuando se activa el proceso de envío de saludos, este pasa la instruccción al programa de comunicaciones (Ipv6Communication.py), el cual se encarga del envío y recepción de todos los datos que salen hacia otros nodos. En este caso, envía un mensaje al nodo dios o nodo anfitrión del sistema indicando que desea realizar el proceso para crear o ingresar al sistema distribuido.

Al hacer el requerimiento de ingreso, el nodo que está actuando como servidor, recibe la solicitud del cliente mediante el programa de comunicaciones (Ipv6Communication.py) y envía la instrucción al programa servidor del nodo (Server.py), que se encarga de atender todas las solicitudes de los clientes. En este punto, los nodos estarán listos para empezar a hacer el proceso de ingreso, para ello intercambiarán la información sobre la cantidad de RAM,

CPU, disco duro, periféricos y demás recursos que tiene disponible. De igual manera, el sistema hará un escaneo de todos los recursos de los nodos que lo integran y se los entregará al nodo. Luego de intercambiar la información de los recursos, se envía al programa que realiza la toma de decisiones. Este programa hace uso de un método de toma de decisiones multicriterio llamado Scoring (DecisionScoring.py), el cual, basado en los recursos del nodo y del sistema, evalúa la conveniencia del ingreso. Cuando ambas partes toman una decisión, la intercambian. Si es afirmativa en ambos casos, se pasa al proceso de negociación sobre la cantidad de recursos que el nodo va a aportar al sistema y sobre la cantidad de recursos del sistema a los que el nodo podrá acceder.

El método escogido para realizar la negociación entre ambas partes se llama principio de concesiones alternativas de Zeuthen (NegotiationClient.py y NegotiationServer.py), donde como su nombre lo indica, cada parte realiza una oferta de cierta cantidad de recursos que el nodo propone para el sistema, y con base en ello el otro jugador decide si aceptarla o rechazarla. Cada jugador hace una propuesta y en cada ronda de negociación cede un poco hasta que haya un acuerdo. Hay que tener en cuenta que el sistema también debe indicar la cantidad de recursos a la que el nodo tendrá derecho a acceder. Es probable que el sistema sea atractivo a un nodo por el hecho de que este tiene un sensor en específico, salida a internet, etc. Si en el proceso de negociación no se tiene un acuerdo sobre el acceso a este, el nodo no encontrará una razón para unirse al sistema.

Si hay un acuerdo sobre la cantidad de recursos que cada parte proporciona, el nodo ingresa y el sistema lo incluye en la base de datos que representa a los nodos miembros. Este realizará las funciones correspondientes para hacer efectivo ese ingreso. Si no hay un acuerdo, ambos nodos procederán a incluirse mutuamente en una clase de lista negra, la cual indicará que la negociación ya fue hecha para que los nodos no ingresen en un bucle de negociaciones sucesivas.

El nodo que acaba de ser aceptado creará una carpeta en donde almacenará toda la información relacionada con el sistema distribuido. En esta carpeta también se almacenará toda la información de la virtualización, que es el corazón del sistema distribuido, es decir, los contenedores que crea el programa Docker para virtualizar cada uno de los recursos y construir el equipo de cómputo donde se colocará el sistema operativo y el resto de elementos que componen el sistema distribuido TLÖN. Aunque esto último no es del alcance de la tesis como se mencionó anteriormente, se realizan interacciones con este sistema.

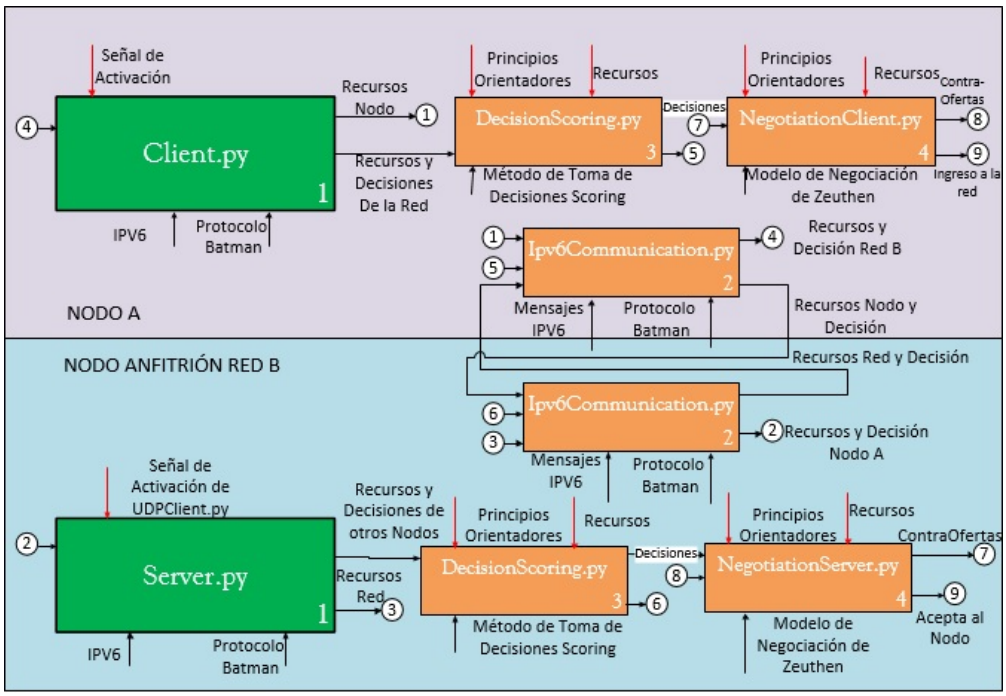

Figura 5-2.: Toma de Decisiones y Negociación

5.2.1. Tipos de Colaboraciones

Para controlar el comportamiento del nodo se añadió un índice de altruismo-egoísmo, es decir, un parámetro que influencia al nodo para que se comporte de cierta manera según el ambiente donde se encuentre. Este índice va de cero a uno: si el índice tiende a cero, el comportamiento del nodo es más altruista y cuando tiende a uno, es más egoísta. Para esto, se considera el comportamiento del individuo en la comunidad y su disposición a colaborar de manera altruista como un incentivo para mejorar su reputación o estatus en la sociedad y de igual forma ser tenido en cuenta por el sistema a la hora de la asignación de recursos, cuando sea requerido por el dispositivo. Este índice también ayudó a simular los escenarios a los cuales puede pertenecer un nodo dependiendo de su localización física. Según lo explica Fitzek en [22], dependiendo de la relación que se tenga entre usuarios o el ambiente en el que se encuentre el dispositivo, se va a esperar cierto tipo de colaboración. En el caso del sistema, se simularon cinco tipos de colaboración.

Colaboración Muy Altruista

En este caso, se estableció un escenario en donde los nodos tendrían un índice de altruismo-egoísmo muy bajo. Este escenario se simuló con una distribución beta, cuyos parámetros fueron α=1 y β=5, de manera que los índices tendieran al altruismo, como se observa en la Figura **5-3**. Este escenario se da cuando los dispositivos pertenecen al usuario, y pueden confiar totalmente entre ellos.

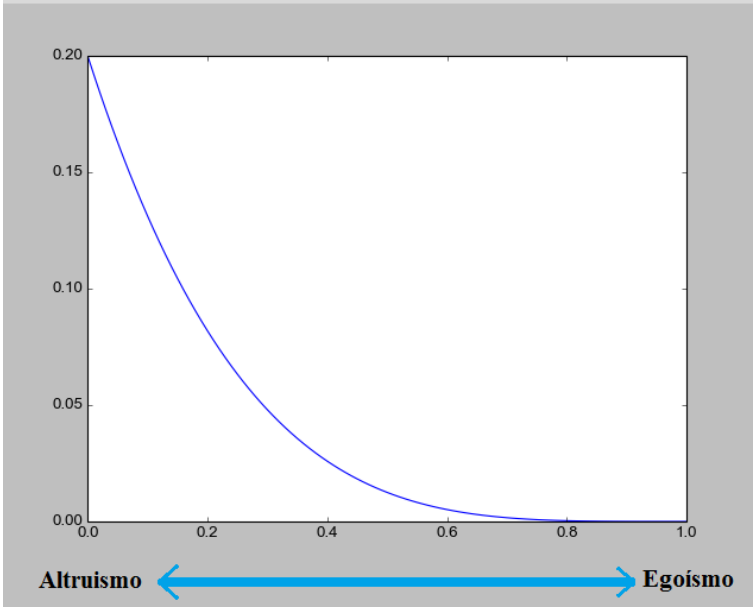

Figura 5-3.: Comportamiento Muy-Altruista

Colaboración Altruista

El índice de altruismo-egoísmo se encuentra un poco hacia el centro, como se observa en la Figura **5-4**, y es un escenario común en ambientes familiares, donde cada nodo colabora sin esperar nada a cambio. Utiliza la distribución beta con parámetros α=2 y β=5.

Colaboración Neutral

El índice de altruismo-egoísmo se encuentra distribuido justo en el centro, los parámetros de la función beta son α=5 y β=5. Hay igual probabilidad de que hayan comportamientos altruistas y egoístas, como se observa en la Figura **5-5**. Es un escenario que se puede presentar en una empresa, entre colegas o compañeros.

Colaboración Egoísta

En este caso la gráfica tiende a la derecha, como se observa en la Figura **5-6**, y se acerca al egoísmo. Este tipo de escenarios se da en espacios públicos donde no hay control sobre qué clase de dispositivos pueden encontrarse en el área y qué clase de amenazas representan. Los parámetros de la función son α=5 y β=2.

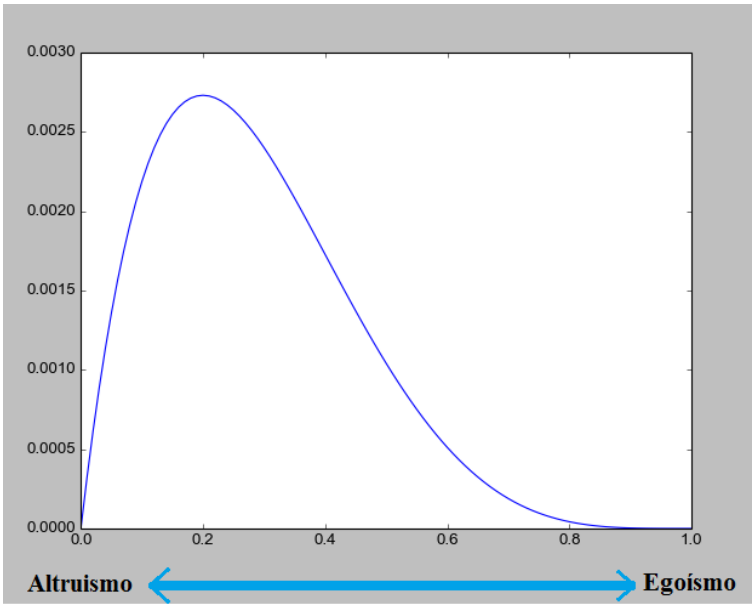

Figura 5-4.: Comportamiento Altruista

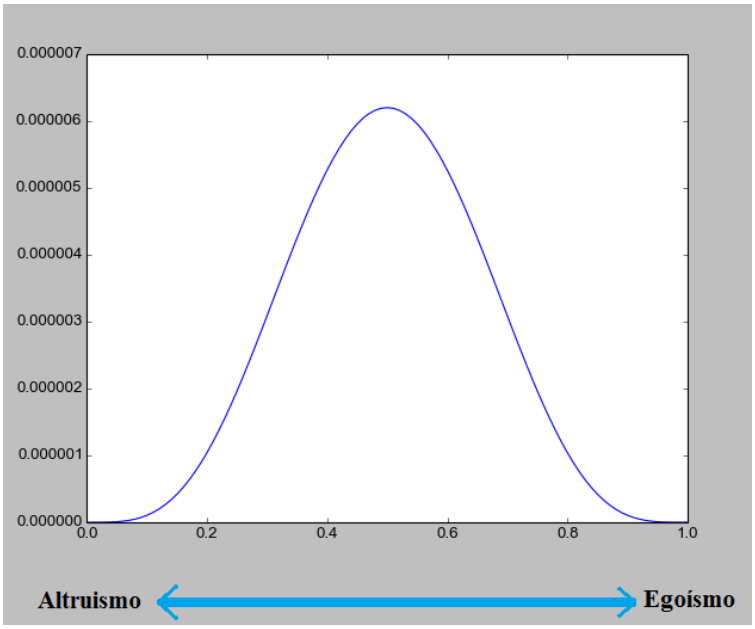

Figura 5-5.: Comportamiento Neutral

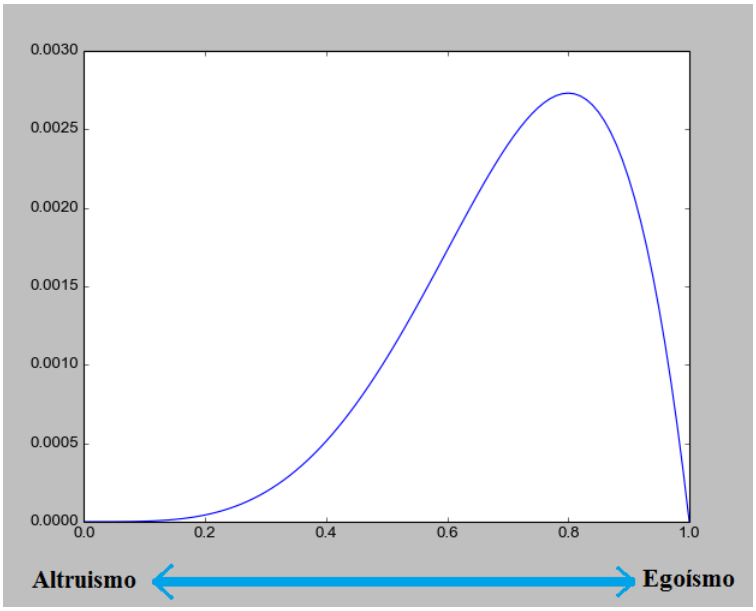

Figura 5-6.: Comportamiento Egoísta

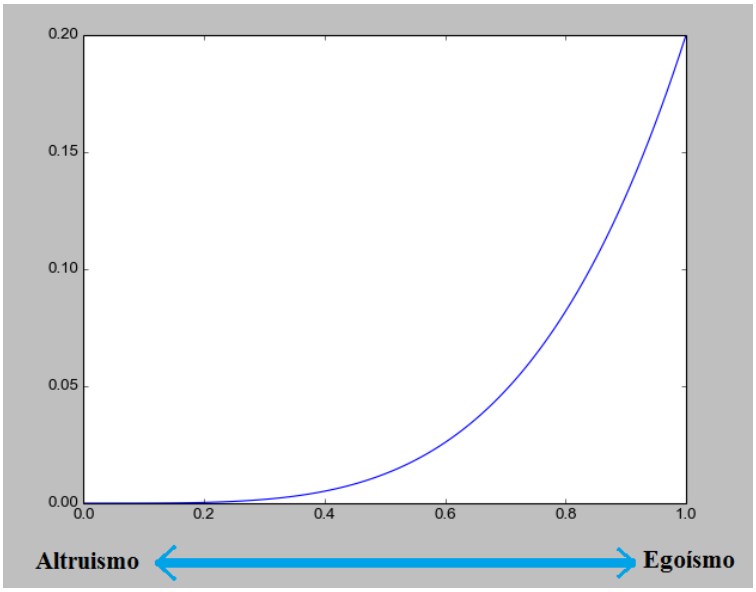

Figura 5-7.: Comportamiento Muy Egoísta

Colaboración Muy Egoísta

Este tipo de colaboración no se da realmente en ningún escenario, ya que es un comportamiento en el que el nodo no colabora en ningún momento, por lo tanto no creará o no se unirá al sistema. Sin embargo, se tuvo en cuenta por la simetría en las pruebas y para realizar todo el espectro de altruismo-egoísmo. Los parámetros de la función beta son α=5 y β=1 y su posición está en el extremo derecho, como se observa en la Figura **5-7**.

5.2.2. Nodo con otro Nodo Solitario

Para el primer paso, en el escenario de dos nodos se hicieron todas las permutaciones con los dispositivos disponibles y todos los escenarios mostrados anteriormente. Para todos los casos que se documentan en las tablas que siguen, se hicieron 100 pruebas en cada uno de los escenarios. Esos datos se obtuvieron con el programa Python y representan el promedio de cada uno de ellos. Para la obtención de los resultados, se colocó a uno de los nodos como cliente y el otro como servidor, se utilizó el programa para que hiciera 100 procesos de toma de decisión y 100 procesos de negociación. Luego, cada una de las decisiones se iban colocando en un archivo de texto con la ayuda del programa Python. Este archivo luego se sacaba y se procesaba en Excel para una mejor organización. Los resultados en python y en Excel se adjuntan en el anexo C al igual a los programas específicos que se utilizaron para sacar los resultados.

Toma de Decisiones

Tabla 5-1.: Porcentaje de acuerdo de las partes para realizar la negociación de recursos(%)

Escenarios	Muy Altruista		Altruista		Neutro		Egoísta	
	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ
Raspberry 1-Computador	91	8,1	80	11,5	32	9,6	2	2,6
Computador- Raspberry 2	92	7,9	76	8,2	36	8,1	19	2,2
Raspberry 1 - Raspberry 3	97	8,7	73	5,0	42	7,6	14	4,1
Raspberry 2 - Raspberry 3	94	7,7	76	8,8	37	8,5	14	8,2
Raspberry 1- Raspberry 2	90	8,4	75	9,4	40	6,2	12	5,7
Computador - Raspberry 3	94	5,7	78	9,2	35	9,4	6	5,5

El comportamiento en cada uno de los escenarios de nodo solitario con nodo solitario se observa en la Tabla **5-1**, donde los dispositivos Raspberry 1 (R1), Raspberry 2 (R2), Raspberry 3 (R3) y el Computador portátil (C). Realizan negociaciones entre ellos. La cantidad

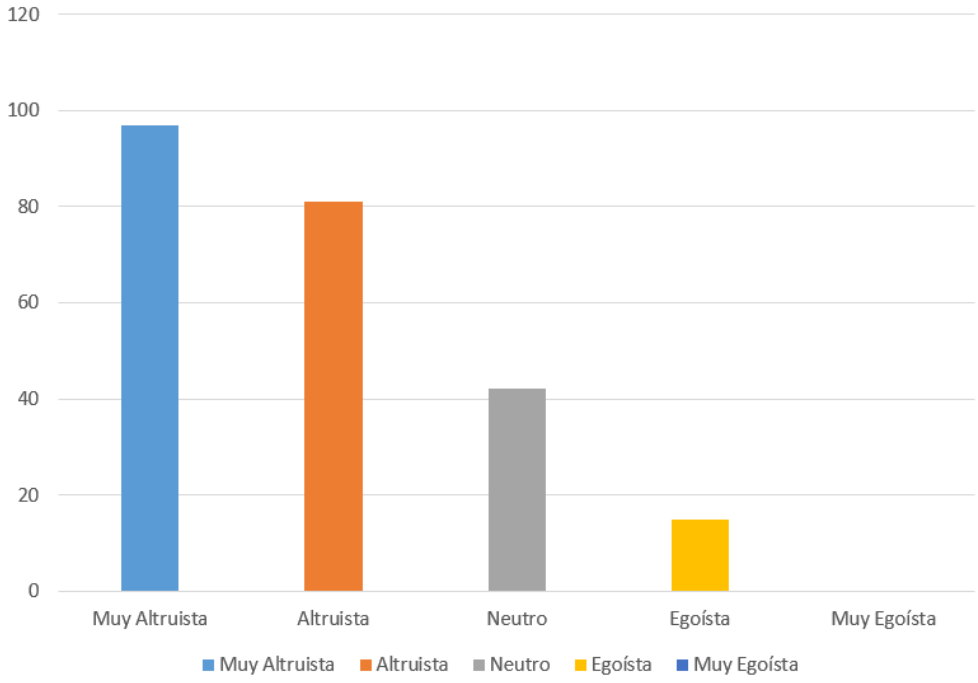

Figura 5-8.: Porcentajes de Ingreso

de nodos que toman una decisión sobre realizar el proceso de creación del sistema distribuido disminuyen mientras se pasa del rango altruista al rango egoísta (gráfica **5-8**). Si la cantidad de recursos que tienen los nodos es similar en el escenario egoísta, estos tienen una posibilidad ligeramente superior de realizar el siguiente paso de la negociación; mientras que si la cantidad de recursos es muy diferente, como en el caso del computador y cualquiera de las raspberry, sobre todo la 1 y la 3, las probabilidades bajan ligeramente. Cabe aclarar que este escenario ocurrirá únicamente cuando se crea el sistema, luego de ello, todos los nodos preferirán hacer la negociación con la red ya constituida, como se observará en pruebas posteriores.

En esta parte se omitió el escenario muy egoísta, ya que ningún nodo aceptó continuar con el proceso de ingreso, sino que descartó por completo crear el sistema distribuido, y en el proceso de análisis no añadía ningún valor. El valor que aparece en la primera columna en cada escenario de comportamiento corresponde al promedio y el segundo dato corresponde a la dispersión. Entre las pruebas realizadas con cada par de dispositivos, el porcentaje de ingreso varía con respecto al comportamiento; pero es muy similar cuando se compara con cualquier otro par de dispositivos. Por lo general el ingreso es más exitoso cuando se comparan dos dispositivos que tienen recursos similares como por ejemplo en los escenarios entre todas las Raspberrys, o el del computador y la Raspberry 2 ya que esta última es la que más tiene recursos.

Negociación

Tabla 5-2.: Porcentaje de recursos entregados en cada escenario (%)

Escenarios	Escenario 1				Escenario 2				Escenario 3			
	R1	σ	C	σ	C	σ	R2	σ	R1	σ	R3	σ
Muy Altruista	83,6	17,1	81,5	17.6	82,1	19,1	83,1	22,3	83,4	21,5	84,4	19,1
Altruista	71	16,3	69,7	15,2	73,3	17,3	75,4	16,8	73,9	18,6	74,3	18,9
Neutro	51,1	15,8	50,1	11,8	51,3	10,9	53,3	15,9	52,3	17,5	51,2	15,6
Egoísta	46,8	11,5	43	9,8	46,8	8,8	47,1	10	44,4	11,4	45,7	14,3
Muy Egoísta	46,5	10,5	44,1	9,9	47,8	5,8	48,1	8,5	44,9	8,8	44,8	9,9

Tabla 5-3.: Porcentaje de recursos entregados en cada escenario (%)

Escenarios	Escenario 4				Escenario 5				Escenario 6			
	R1	σ	R2	σ	R2	σ	R3	σ	C	σ	R3	σ
Muy Altruista	83,9	18,4	83,1	17,9	85,9	15,3	84,6	18,2	84,8	21,1	85,8	24,1
Altruista	70,7	17,6	71	17,6	70,4	16,8	70,5	14,3	71,8	18,5	72,8	18,9
Neutro	49,2	13,5	50	15,4	52,1	12,5	54	12,2	50	14,3	48,9	11,4
Egoísta	43,3	11,8	42,8	13,4	45	11,5	44,9	11,2	46,9	11,9	46,1	12,2
Muy Egoísta	43,9	6,7	44,1	9,6	46,5	8,4	46,2	12,1	47,7	7,1	47,2	8,1

En el proceso de negociación, se tuvo en cuenta el escenario muy egoísta; no solo por la simetría de las pruebas, sino porque aporta datos interesantes para el análisis. Suponiendo que el nodo acepta seguir con el proceso de negociación, se encontrará ante una situación en la que debería indicar qué porcentaje de los recursos que tiene disponible va a aportar al sistema distribuido. La negociación se realiza mediante el método de concesiones sucesivas de Zeuthen, los resultados obtenidos se presentan en las Tablas **5-2** y **5-3**; las cuales corresponden al mismo proceso de negociación que se dividió en dos por la cantidad de información.

En cada bloque de escenario de comportamiento se ven cuatro casillas. La primera corresponde al promedio de recursos entregados por el primer dispositivo (Raspberry 1 (R1), Raspberry 2 (R2), Raspberry 3 (R3) y Computador portátil (C)) y la segunda a la desviación estándar de esa prueba. Las dos últimas casillas corresponden al promedio de los recursos entregados por el dispositivo con el cual se negoció y su desviación, respectivamente. En el primer caso del escenario entre la Raspberry 1 y el computador, en un ambiente muy altruista la Raspberry entregó un total de 83,6 % de sus recursos disponibles; mientras que el computador

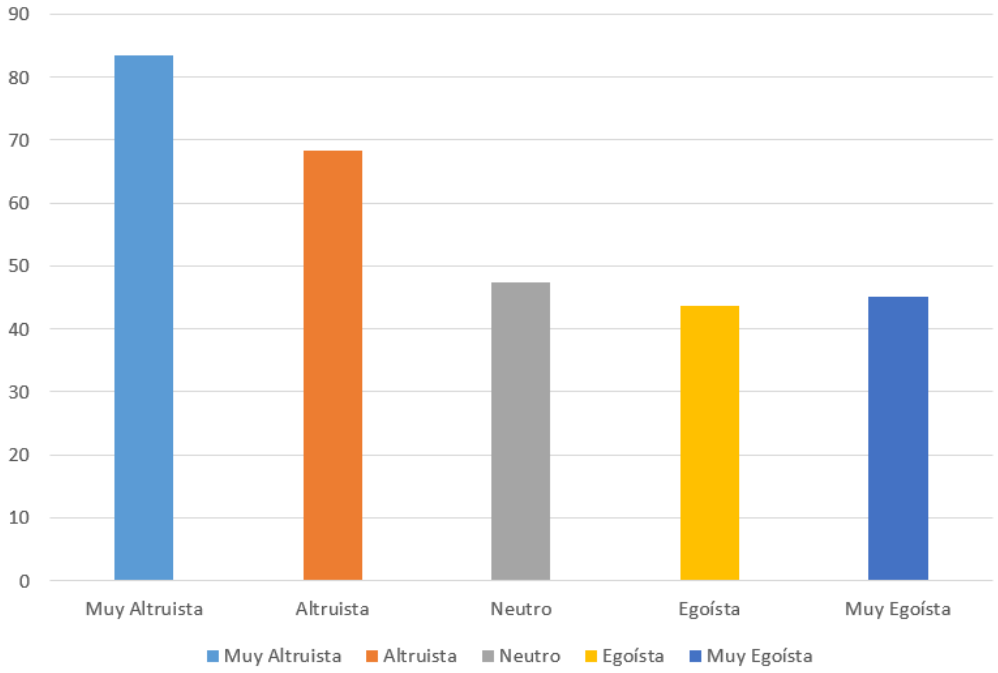

Figura 5-9.: Porcentajes de Ingreso

entregó un total de 81,5 %. Se observa que el equipo que tiene menos recursos siempre da un poco más, con el fin de ser aceptado por la otra parte. En el escenario muy egoísta, la cantidad de recursos entregados por los dispositivos sube ligeramente, ya que la otra parte se comporta de la misma forma y exige más cantidad de recursos. Se observa que los recursos entregados son muy similares en los escenarios al compararse por pareja de dispositivos; lo que quiere decir que no afecta mucho la diferencia que exista entre la cantidad de recursos de los dispositivos que están negociando y la cantidad de recursos que entregan, aunque el que menos tiene siempre aporta un poco más. En la Figura **5-9** se observa cómo varía la cantidad de recursos entregados al sistema con la variación del índice de egoísmo; entre más egoísta es el dispositivo, menos recursos entrega.

La desviación estándar de esta prueba se elevó bastante por el hecho de que aunque el índice de egoísmo estaba entre los rangos de las gráficas de comportamiento mostradas más arriba, este era susceptible de variar entre esos límites, por lo que aunque el nodo estuviera en un rango muy altruista, en algún momento podría llegar a ser neutro.

Creación del Sistema

Debido al método de negociación y a los parámetros que se utilizan en el sistema distribuido, los nodos llegan a un acuerdo incluso en una ronda de negociaciones de hasta 20 propuestas y contrapropuestas. En este caso, se limitaron únicamente a diez ya que entre mayor número

Tabla 5-4.: Porcentaje de nodos que crean el sistema después de aceptar realizar la negociación (%)

Escenarios	Muy Altruista		Altruista		Neutro		Egoísta		Muy Egoísta	
	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ
Raspberry 1 - Computador	100	0,0	100	0,0	99	1,2	75	4,2	47	9,5
Computador - Raspberry 2	100	0,0	100	0,0	98	2,3	79	5,6	43	7,8
Raspberry 1 - Raspberry 3	100	0,0	100	0,0	96	3,2	70	6,7	39	9,8
Raspberry 2 - Raspberry 3	100	0,0	100	0,0	100	0,0	73	4,5	48	10,1
Raspberry 1 - Raspberry 2	100	0,0	100	0,0	92	3,4	73	6,7	47	8,7
Computador - Raspberry 3	100	0,0	100	0,0	100	0,0	69	7,8	48	9,4

de negociaciones, más recursos físicos se consumen y lo que se espera es que la red ad hoc se comporte de una manera rápida. Además, dependiendo del tipo de cooperación, el nodo solicita una mínima cantidad de recursos entregados por el otro nodo para poder crear el sistema distribuido. De acuerdo con lo anterior, la Tabla **5-4** muestra el porcentaje de nodos que aceptaron realizar la negociación, lograron llegar a un acuerdo y crearon el sistema. Se observa que en los tres primeros escenarios de comportamiento, una vez el nodo toma la decisión de realizar la negociación de los recursos, es muy probable que llegue a un acuerdo con su contraparte. En estos casos, las rondas de negociación siempre eran menos de cinco; mientras que en el escenario muy egoísta, en múltiples ocasiones se observaron rondas de negociación que superaban las veinte y muy rara vez se extendían indefinidamente.

5.2.3. Nodo Solitario con Sistema Distribuido

En este escenario, se hizo que el nodo a realizara el proceso de ingreso con un sistema distribuido ya conformado. Se formó el sistema con 3 dispositivos, y se configuró el cuarto dispositivo para realizar la negociación con el sistema.

Toma de Decisiones

Se utilizó cada uno de los nodos para que realizaran el proceso de ingreso con el sistema. En el primer paso, que es la toma de decisiones, se obtuvieron los resultados que se presentan en la Tabla **5-5**. Se observa que a diferencia del caso nodo-nodo, hubo mayor porcentaje de acuerdo sobre pasar a la siguiente etapa de la negociación. Esto se debe a que los nodos encuentran en un sistema ya constituido, un lugar en el que pueden confiar más a la hora de encontrar recursos y sentirse parte de una comunidad.
Se observa también que la Raspberry 1 tuvo una alta inclinación a hacer parte del sistema distribuido en casi todos los ambientes, incluso en el muy egoísta, aunque sus recursos eran

Tabla 5-5.: Porcentaje de acuerdo de las partes para realizar la negociación de recursos (%)

Escenarios	Muy Altruista		Altruista		Neutro		Egoísta		Muy Egoísta	
	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ
Computador - Red	100	0,0	99	1,2	74	7,6	21	13,1	9	2,6
Raspberry 1 - Red	100	0,0	100	0,0	100	0	98	5,5	77	16,8
Raspberry 2 - Red	100	0,0	100	0,0	93	4,6	40	8,4	18	8,2
Raspberry 3 - Red	100	0,0	100	0,0	95	5,8	42	9,6	23	5,2

muy similares a los de la Raspberry 3. En el caso del computador, este presentó menos inclinación a hacer parte del sistema distribuido, ya que la suma de los recursos que presentan las Raspberrys combinadas no se comparan con los recursos que tiene el computador y aquí el deseo de hacer parte del sistema se debió más a un sentido de altruismo que de necesidad de recursos. La dispersión en el caso del escenario del comportamiento muy altruista y altruista fue cero, ya que todos los dispositivos aceptaron la unión sin excepción.

Negociación

Al igual que en el escenario nodo-nodo, se realizó la misma prueba de negociación del nodo con el sistema y se observó que los porcentajes de recursos entregados al sistema bajaron un poco con respecto al anterior escenario, como se observa en las Tablas **5-6** y **5-7**. En este caso, el comportamiento de la red variaba, al igual que el del nodo.

Tabla 5-6.: Promedio de porcentaje de recursos entregados en cada escenario (%)

Escenarios	Computador - Red				Raspberry 1 -Red			
	C	σ	Red	σ	R1	σ	Red	σ
Muy Altruista	84,9	19,2	40,2	19,2	82,5	22	38,3	11,2
Altruista	72,11	15,2	35,4	12,3	69,3	11,1	33,2	9,2
Neutro	51,7	17,1	32,2	13,4	50,1	12,4	29,3	8,2
Egoísta	42,8	12,2	24,5	9,4	38,9	9,3	23,1	6,4
Muy Egoísta	38,1	11,1	26,1	7,3	33,9	6,9	24,5	5,4

En este caso, se puede observar que los recursos que ofrece la red no superan el 50 % de los recursos que tiene disponible. Esto se programó de esta manera, ya que en un escenario altruista, nuevos nodos podrían acceder hasta a un 80 % de los recursos disponibles de la red. Esto sería desventajoso para nuevos nodos,aunque aseguraría cierta cantidad de recursos para nodos nuevos, más los recursos que aporta el nodo que es aceptado. Al igual que en

Tabla 5-7.: Promedio de porcentaje de recursos entregados en cada escenario (%)

Escenarios	Raspberry 2 - Red				Raspberry 3 -Red			
	R2	σ	Red	σ	R3	σ	Red	σ
Muy Altruista	81,9	26,2	39,2	8,4	84,4	18,2	39,1	7,5
Altruista	72,3	17,4	34,2	11,2	69,3	13,2	31,8	8,9
Neutro	52	14,1	30,1	8,3	50	8,3	28,8	11,2
Egoísta	39,3	12,9	24,6	5,3	41,1	8,1	23,8	7,9
Muy Egoísta	36	9,3	23,9	4,6	33,9	4,5	24,1	8

el caso nodo-nodo, el porcentaje de recursos entregados disminuye a medida que el nodo se vuelve más egoísta.

En la gráfica **5-10** se muestran los recursos entregados de acuerdo con la variación del índice de egoísmo. La gráfica muestra una línea recta que va desde cuando el nodo es 100 % altruista y entrega la totalidad de recursos libres al sistema y termina cuando el nodo tiene un índice de egoísmo del 0.9, en cuyo caso no hubo acuerdo de los recursos a entregar. Esta gráfica se hizo con 100 negociaciones entre el nodo y el sistema, el cual variaba su comportamiento aleatoriamente, igual que el nodo, desde el altruismo hasta el egoísmo.

También se pudo constatar la variación de los recursos entregados por el nodo, de acuerdo con el comportamiento de la otra parte, ya sea el nodo o el sistema. Se obtuvieron los resultados de las Figuras **5-11** y **5-12**. En la Figura **5-11**, se puede observar cómo aumentan los recursos que el nodo entrega al sistema cuando el nodo es más altruista, casi sin ser afectado por las variaciones del comportamiento de la otra parte, mientras esta sea altruista también.

En la Figura **5-12**, se observa desde diferentes ángulos cómo la cantidad de recursos que se entrega se ve afectada cuando ambas partes son egoístas, ya que hace que el nodo tenga que aportar más. En este caso, es cuando la negociación se presenta en su máxima expresión, ya que ninguno quiere dar ninguna ventaja al otro, por lo cual seden lo mínimo posible y es cuando se presentan más rondas de negociación. Usualmente, en un ambiente altruista, las rondas son de una o dos propuestas, mientras que cuando ambos son egoístas, estas se componen de hasta 10 rondas. En ocasiones, no se llega a ningún acuerdo, es decir, las rondas de negociación siguen indefinidamente.

Cuando la otra parte es altruista y el nodo es egoísta, el nodo aporta menor cantidad de recursos, ya que la otra parte va a aceptar cualquier oferta que el nodo haga, por desventajosa que sea. Se observa también en estos casos, que aunque la teoría dice que el primero que

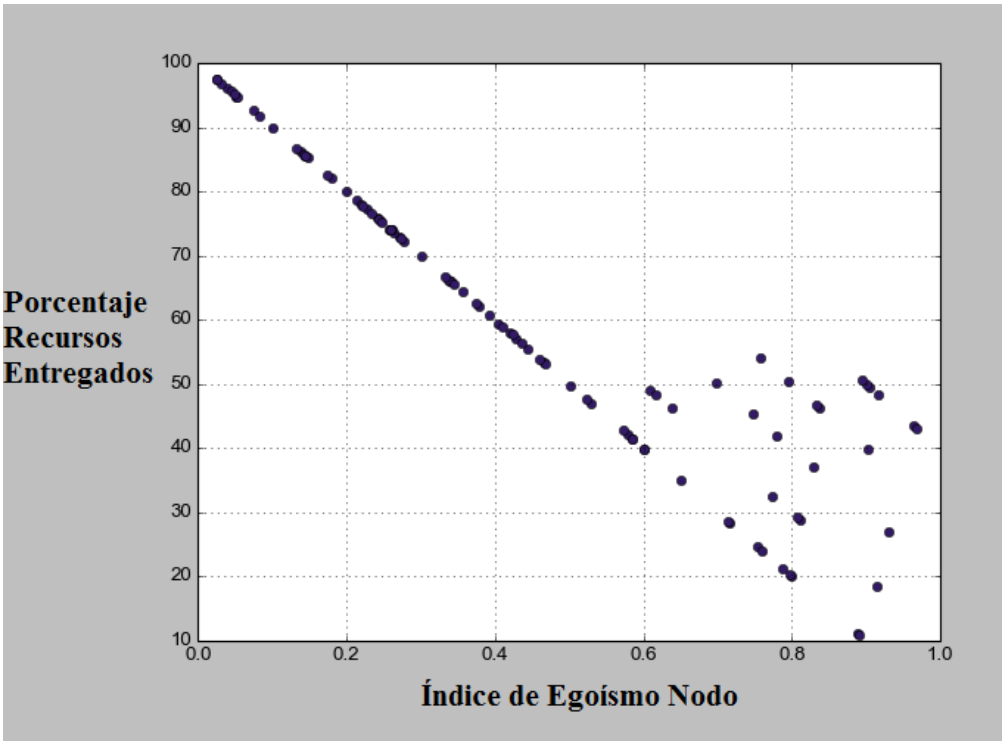

Figura 5-10.: Porcentajes de Recursos Entregado, de acuerdo al índice de egoísmo del Nodo

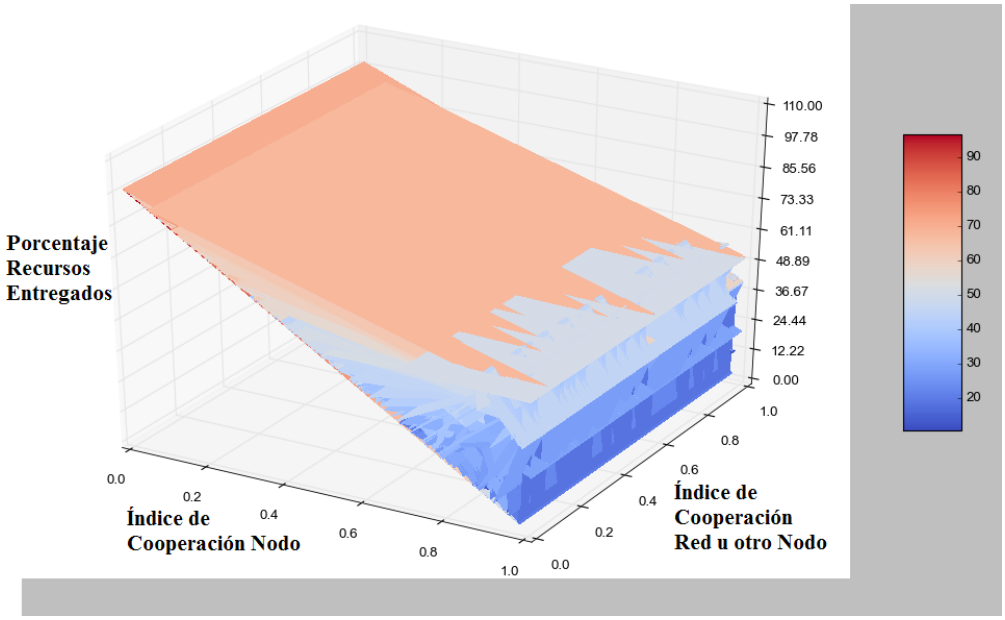

Figura 5-11.: Porcentajes de recursos entregado, de acuerdo a la cooperación del nodo y la red

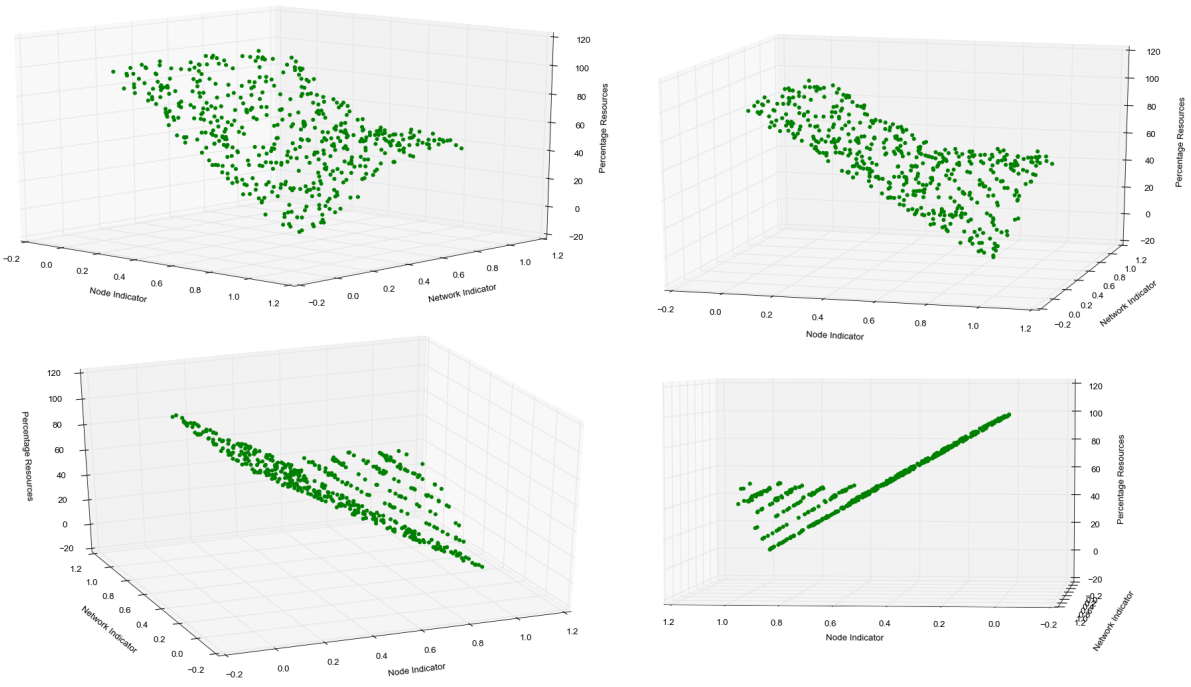

Figura 5-12.: Porcentaje de recursos entregados, de acuerdo con la cooperación del nodo y la red

realiza la oferta es siempre el que tiene la ventaja a la hora de realizar la negociación, es interesante el hecho de que eso no ocurre aquí. Por lo que también se debe tener en cuenta el comportamiento del individuo, pues si este es demasiado altruista, va a dar una mayor cantidad de recursos en la primera oferta.

Ingreso al Sistema

Tabla 5-8.: Porcentaje de nodos que ingresan al sistema después de aceptar realizar la negociación (%)

Escenarios	Muy Altruista		Altruista		Neutro		Egoísta		Muy Egoísta	
	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ	$\overline{X}$	σ
Computador - Red	100	0	100	0	100	0	83	5,8	44	14,5
Raspberry 1 - Red	100	0	100	0	100	0	86	6,5	46	13,2
Raspberry 2 - Red	100	0	100	0	100	0	88	4,6	42	12,7
Raspberry 3 - Red	100	0	100	0	98	2,3	81	7,1	48	15,6

Los porcentajes de ingreso en estos escenarios son muy parecidos a los que se vieron en el caso nodo-nodo (Tabla **5-8**). Los resultados se explican por el hecho de que los nodos que

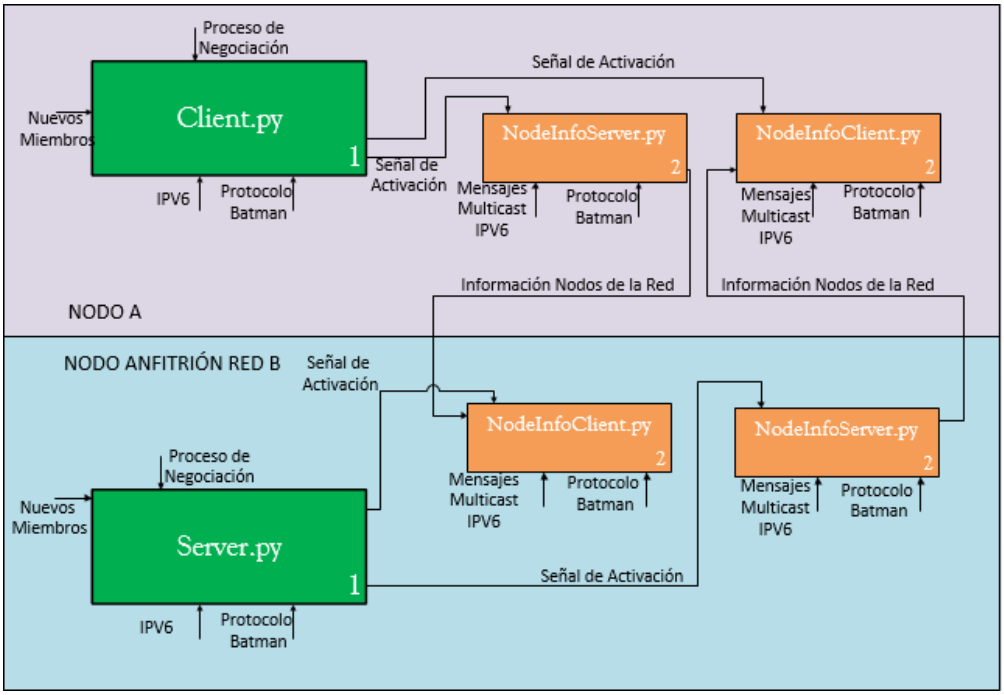

Figura 5-13.: Información de los miembros de la Red

menos recursos tienen, ven en la red la oportunidad de acceder a más recursos y poder aumentar las posibilidades de satisfacción de sus necesidades, mientras que en los dispositivos que más tienen recursos, como los computadores, uno de sus principales incentivos para el ingreso sería el ganar reputación en el sistema. Una vez el nodo toma la decisión de pasar a la siguiente etapa, es muy probable que lleguen a un acuerdo en los tres primeros escenarios de comportamiento, aunque en el escenario egoísta, en un poco menos de la mitad de las veces se llegó a un acuerdo.

Entre las pruebas que se realizaron a la hora de formar el sistema distribuido, el principal inconveniente que se encontró fueron los tiempos de su formación. Entre más nodos quieren ingresar al sistema distribuido, los tiempos fueron más altos. Este fenómeno es producto de la sincronización que requieren los programas de recepción y envío de archivos necesarios para tomar la decisión inicial de ingreso y luego en la negociación del nodo con el sistema distribuido sobre la cantidad de recursos que va a aportar, ya que se presenta un proceso de oferta y contraoferta de recursos que conlleva a un intercambio de mensajes entre el nodo y la red.

5.2.4. Autenticación de Integrantes del Sistema

Aunque no es uno de los objetivos de la tesis tratar con temas complejos de seguridad en el sistema distribuido, se debe asegurar por lo menos que los nodos que son ajenos al sistema o nodos que no han hecho el proceso de unión accedan a información o recursos del sistema distribuido, por lo que se ha realizado un método muy sencillo, pero confiable para tratar este asunto.

Una vez el nodo es admitido en el sistema distribuido, el nodo anfitrión debe notificar a los demás integrantes del sistema para que reciban solicitudes únicamente de estos nodos, ya que pueden existir otros sistemas distribuidos en el área u otros nodos que no han sido aceptados y probablemente quieran acceder a los recursos sin haber sido autorizados a ingresar. En la Figura **5-13** se observa el proceso. Inicialmente, se observan los bloques cliente (Client.py) y servidor (Server.py) del nodo A y el nodo anfitrión de la red B, respectivamente. Como se mencionó anteriormente, desde aquí se administra el proceso de unión, ya que son ellos los que llaman a todos los programas encargados de la toma de decisiones y negociación. Luego, una vez el proceso de negociación haya finalizado y se haya llegado un acuerdo sobre los recursos a entregar, se activan los programas NodeInfoServer.py y NodeInfoClient.py. El submódulo NodeInfoServer.py consulta la base de datos del sistema distribuido buscando información de los nodos miembros del sistema, y envía un mensaje UDP con la información del listado actualizado. Por su parte, el submódulo NodeInfoClient.py del nodo A escucha los mensajes UDP con la información de todos los integrantes del sistema que le envía el nodo anfitrión y la guarda en la carpeta del sistema distribuido.

Los nodos que ya son miembros del sistema escuchan todo el tiempo esta información, de tal manera que cuando hay un integrante nuevo en el sistema, actualizan su base de datos. Cuando los nodos reciben la información, primero revisan su base de datos de tal manera que solo reciben la información de actualización de nodos que ya reconozcan como miembros del sistema, así se evita que un nodo que no hace parte del sistema distribuido pueda suplantar a un miembro legítimo.

5.3. Recepción de Instrucciones e Información

Para el primer objetivo de la tesis se creó un método que permite la recepción de mensajes de nivel superior provenientes del sistema distribuido. Este objetivo estuvo encaminado al desarrollo de una plataforma sobre la cual todas las instrucciones pueden ser recibidas, mas no a la creación de todas las instrucciones como tal. Esto por la innumerable cantidad de estas y por ser un objetivo del lenguaje.

No obstante, se realizaron algunas instrucciones para probar el funcionamiento del protoco-

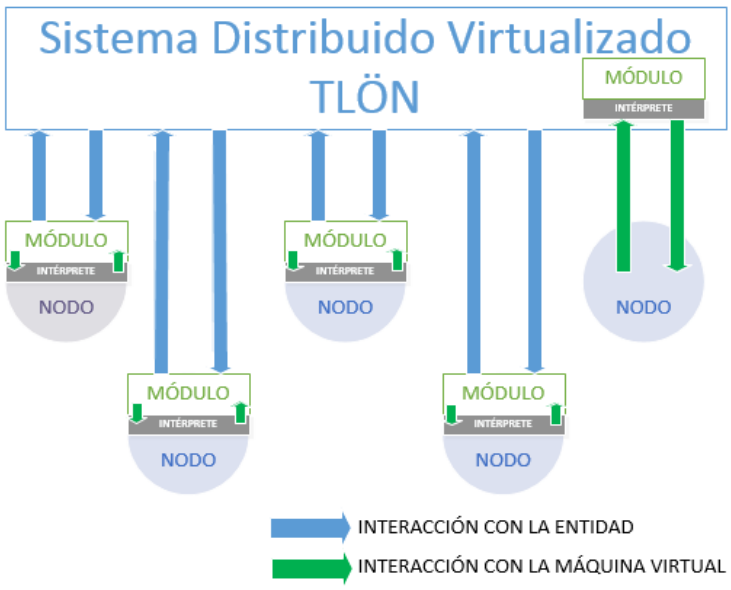

Figura 5-14.: Interacción del Sistema Distribuido con el módulo

lo. Estas se componen de instrucciones básicas relacionadas con el procesamiento, almacenamiento, manejo de dispositivos de entrada y salida, para la creación de la red ad hoc, toma de decisiones, negociación, y para realizar operaciones como suma, multiplicación, guardar información, entre otros. En la práctica, aparece como una clase de Command Prompt (CMD) o símbolo del sistema, el cual permite la simulación de instrucciones que normalmente vienen desde el sistema distribuido. Las instrucciones se reciben en un lenguaje propio del sistema distribuido TLÖN y son ejecutadas a nivel de máquina en Python y en Bash.

Como se puede observar en la Figura **5-14**, las instrucciones salen desde el sistema distribuido y son recibidas por el módulo en el nodo. Luego, son manejadas internamente por el protocolo de comunicaciones el cual analizará la instrucción y, dependiendo de la misma, la procesará y la ejecutará. Cuando se tiene una respuesta, esta se envía de nuevo al sistema distribuido. Las pruebas realizadas con el sistema fueron exitosas. Se hicieron algunos experimentos simulando la recepción de instrucciones desde el sistema distribuido relacionadas con el procesamiento de información, tales como requerimientos de operaciones como sumas, multiplicaciones, u otros como listar archivos o los miembros del sistema. Algunas instrucciones más complejas como la creación de la red ad hoc, la toma de decisiones o la negociación, se demoraron un poco más, debido a la cantidad de interacciones que conllevan.

Por ejemplo, una instrucción simple como la suma de dos números suponía inicialmente el envío de la instrucción al sistema distribuido, este analizaba el comando “suma” y los parámetros “a” y “b”. El requerimiento podría ser enviado a un nodo en específico o ser

ejecutado en el sistema. Si era enviado a un nodo, este recibía la instrucción, la analizaba y luego la ejecutaba internamente y devolvía la respuesta enviadándola al sistema distribuido para que la pasara al nodo que la solicitó. Esta respuesta se mostraba en el CMD del nodo.

Se hicieron pruebas con dispositivos de entrada y salida utilizando los pines de propósito general que tiene la raspberry. Aquí se conectaron sensores, los cuales fueron leídos y su información fue compartida en el sistema. Además, se conectaron actuadores tales como leds o zumbadores eléctricos que fueron activados con instrucciones desde la consola de comandos.

Además, se hicieron algunas pruebas con los pines de propósito general de la Raspberry. Para ello, se conectaron sensores. Uno de ellos fue un codificador de rotación, el cual funciona con un selector que al girar envía los datos de su posición a los pines. Esta información fue capturada por la Raspberry y mostró en el CMD. Lo interesante fue la opción de acceder a este sensor desde otra Raspberry o desde el computador, de manera que el nodo envia la instrucción al sistema distribuido para el acceso a ese recurso. Si en el proceso de negociación, ese recurso fue otorgado, el sistema distribuido envía la instrucción al nodo que lo tenía y este activa el sensor para que la información pueda ser enviada al origen del requerimiento. Entre los sensores que se colocaron hay sensores de temperatura, switches, y sensores de luz.

Para instrucciones de dispositivos de salida, esta opción fue similar. Se conectaron leds, zumbadores e incluso pantallas para mostrar datos. Para las pantallas, el sistema distribuido podría requerir la visualización de una imagen de manera distribuida entre los dispositivos que contaran con una pantalla; para ello, se cortaba la imagen en partes iguales y se enviaba cada parte a cada uno de los dispositivos para que ellos la mostraran. Además, si un nodo requiere la activación de un led que está en otro nodo, este consulta al sistema distribuido, el cual verifica si este nodo tiene acceso al led. Si fuese así, la instrucción se envía al dispositivo, el cual acciona el led.

En la Figura **5-14** se puede observar cómo uno de los módulos se encuentra dentro del sistema distribuido. Se programó este caso ya que en algunas ocasiones es necesario crear un nodo que sea completamente virtual o hay dispositivos que no tienen suficientes recursos físicos para poder realizar la toma de decisiones y la negociación por sí mismos. En estos casos, estos nodos deben recurrir a la ayuda del sistema distribuido para que tomen la decisión por ellos. El trabajo de virtualizar los recursos del nodo corresponde al sistema distribuido, pero se diseñó el método para que el sistema interactuara con el nodo y se hiciera el proceso de toma de decisiones y negociación. Si un nodo que quiere hacer el proceso de ingreso tiene recursos físicos, como mínimo debe tener la capacidad de configurarse en modo ad hoc para que se pueda comunicar con el sistema. En el caso de que el nodo sea completamente virtual, el sistema distribuido le asignará una Ip virtual para que sirva como identificador y los otros nodos puedan comunicarse con él.

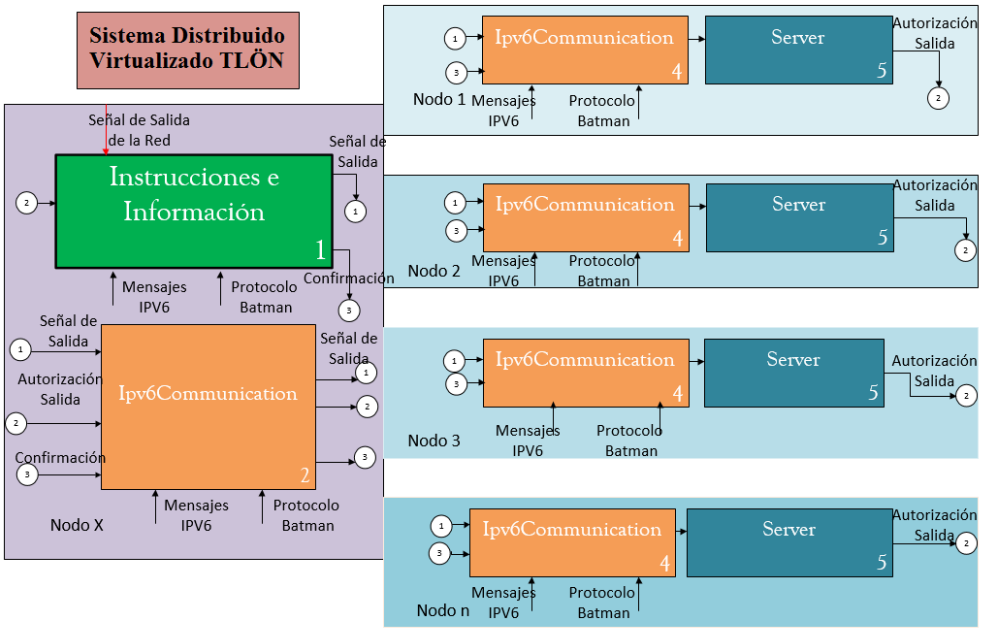

Figura 5-15.: Salida del Nodo del Sistema

5.4. Salida del Nodo del Sistema

Para el objetivo cinco de la tesis que se trata sobre la salida del nodo del sistema distribuido, se realizó el siguiente procedimiento; Cuando el nodo A recibe una señal de salida proveniente de un nivel superior, sigue los pasos que se muestran en la Figura **5-15**. La instrucción es procesada por el módulo de instrucciones que tiene el nodo A, la cual identifica la orden de salida y luego envía la instrucción al sistema distribuido el cual tiene que consultar todos los procesos que el nodo está llevando a cabo con cada uno de los nodos pertenecientes al mismo; si alguno de los recursos del nodo A está siendo utilizado, el nodo y el sistema se pondrán de acuerdo para esperar un tiempo el cual sea suficiente para que el nodo que está utilizando el recurso pueda llevar a cabo su tarea. Después de esto, el nodo intentará nuevamente realizar el proceso de salida, si el sistema distribuido verifica que el nodo está libre, le dará la autorización para que salga.

Antes de que el nodo complete su proceso de salida, debe verificar su base de datos de tal manera que devuelva toda la información que tenga almacenada, ya que el sistema puede haberle solicitado el almacenamiento de alguna parte de un archivo, y para que no haya ningún inconveniente con pérdida de información, el nodo la reúne y la envía al sistema para que este la vuelva a repartir entre los nodos miembros, de manera que haya redundancia de la misma.

Si el nodo sale del área de cobertura de la señal del sistema distribuido de una manera

fortuita, lo anterior no sería posible, ya que el nodo no tendría el tiempo de informar al sistema sobre su salida, en este caso, el nodo estaría intentando de nuevo la conexión al mismo. Dependiendo del tipo de sistema para el cual el nodo sea programado, es decir, si está planeado que el sistema tenga bastante movilidad o si va a ser un sistema fijo, el nodo después de un tiempo acorde, al no recibir ninguna señal por parte del sistema, elimina la información del mismo. Esto, por el hecho de que el sistema distribuido es dinámico, y si por ejemplo el nodo deja de tener contacto durante cierto tiempo, es probable que cuando regrese se encuentre con una red totalmente diferente, con nuevos miembros y nuevos recursos que le puedan interesar o no, por lo cual es necesario que realice el proceso de ingreso de nuevo.

Una vez el nodo sale del sistema distribuido, vuelve de nuevo a su estado inicial en donde estará buscando otros sistemas distribuidos que estén en el área y realizará todo el proceso de nuevo, si quiere ingresar a alguno de los que están a su alcance. Hay que tener en cuenta que el nodo al salir del sistema distribuido, esperará un tiempo prudencial para volver a realizar el ingreso a ese sistema en específico si es que así lo requiere, ya que no tiene sentido que el nodo esté solicitando la salida del sistema para ingresar inmediatamente.

6. Conclusiones y recomendaciones

6.1. Conclusiones

El proceso que se realizó para que el nodo se diera a conocer y escuchara las solicitudes de unión de otros nodos fue exitoso; se utilizaron mensajes UDP para llevar los mensajes de saludo sobre el protocolo IPV6, La Ip fue utilizada como identificación de los nodos y esta información fue almacenada en una base de datos que era consultada cada vez que los miembros del sistema necesitaban realizar alguna validación. Las pruebas que se hicieron en casos en donde habían varios nodos solitarios dieron buenos resultados en cuanto a la elección de un nodo dios el cual fuera el encargado de realizar las gestiones para crear el sistema distribuido y adherir a cada uno de los integrantes al mismo.

El método de toma de decisiones escogido, estuvo acorde a las necesidades del proyecto, ya que era de los menos costosos computacionalmente hablando, lo que es perfecto para las redes ad hoc. Además de ser un método sencillo es muy completo porque permite al nodo dar una valoración de cada uno de los criterios, exaltando aquellos que realmente necesita, y además de poder valorar qué tanto ese criterio aplica a las opciones disponibles.

El método de negociación de concesiones sucesivas de Zeuthen permitió al nodo indicar al sistema con qué cantidad de recursos iba a colaborar. El método es muy completo ya que permite hacer propuestas y contrapropuestas durante gran cantidad de rondas, este método resultó ser atractivo por su facilidad de implementación, además cada jugador sigue realizando propuestas llegando siempre a un acuerdo en un número finito de etapas, por supuesto el número de estas iteraciones se limitaron a tres por tratarse de una red ad hoc.

Se encontró que ante el comportamiento altruista de un nodo, este no es afectado de manera considerable por los posibles comportamientos egoístas de sus vecinos; mientras que en un escenario egoísta, el nodo se ve obligados a aportar más recursos ya que la otra parte así lo exige; incluso llegando a un desacuerdo total de las propuestas, por lo tanto en esos casos no se formó el sistema distribuido.

Los nodos prefieren realizar procesos de unión con sistemas ya constituidos en vez de realizarlos con nodos solitarios, esto se debe a que los nodos encuentran una organización que les puede ofrecer de manera justa acceso a recursos que el nodo quizás no posea o un lugar

en donde convivir y en el cual puedan ganar reputación.

Para el primer objetivo de la tesis se diseñó un submódulo mediante el cual el nodo recibe instrucciones de nivel superior y las procesa a bajo nivel, las pruebas realizadas fueron exitosas y permitieron la formalización de la ejecución de las instrucciones y la recepción de las respuestas. Se simuló la ejecución de instrucciones relacionadas a procesamiento de información en los nodos como sumas y multiplicaciones, manejo de algunos dispositivos de entrada y salida conectados a puertos de propósito general en los dispositivos raspberry pi, almacenamiento de información en los nodos, instrucciones como la creación de la red ad hoc, entre otros; todos fueron exitosos, en el caso de instrucciones más complejas se tuvieron algunos tiempos altos, pero fueron ocasionados por tiempos añadidos para permitir una mejor sincronización entre las raspberries.

Cuando el nodo realizó el proceso de salida del sistema distribuido, en todo momento realizó la devolución de toda la información almacenada lo que permitió la integridad de toda la información del sistema; en ocasiones el nodo tuvo que esperar a que ciertos nodos terminaran los procesos que estaban realizando con los recursos que el nodo saliente tenía; pero al final el sistema distribuido permitió la salida de cada uno de ellos. En el caso de la salida fortuita, cada nodo eliminó la información que tenía de cada sistema; el tiempo para realizarlo depende de cada situación o ambiente específico en donde se configure.

Una aplicación en la industria podría ser aquella en donde una persona vaya conduciendo y esté llegando a una ciudad. Aquí, el automóvil intentaría hacer parte de una VANET y estaría tratando de recolectar información para el usuario. El nodo escanearía los sistemas que estuvieran en el área e inicialmente obtendría la información de los recursos que esa red posee, es decir, información sobre el clima, sobre el estado de la carretera, sobre el tráfico, etc. La cual le interesaría al usuario. Luego, el nodo para poder acceder a esa información realiza la negociación con el sistema el cual le pediría en cambio de esa información algunos recursos. Si el sistema no tiene alguna información que el nodo necesita, buscaría un nuevo sistema para unirse.

6.2. Recomendaciones

Aunque las pruebas de todas las etapas del proyecto fueron exitosas, no era un objetivo de la tesis que estas fueran óptimas en cuanto a tiempo. Debido a un proceso de sincronización entre nodos, se tuvo que aumentar un poco los tiempos de comunicación lo que hizo que algunos procesos como el intercambio de recursos y de propuestas entre los nodos fueran un poco elevados, un proyecto encaminado a optimizar estos tiempos sería muy importante a la hora de usarlo en aplicaciones con gran cantidad de nodos.

Con la optimización de los tiempos en el programa, en el futuro podría utilizarse el mismo esquema pero utilizando un método más complejo para tomar la decisión, bajo necesidades específicas de algunos sistemas, se podrían utilizar métodos de optimización multiobjetivo para la toma de decisiones, en los cuales se requieran maximizar o minimizar algunos objetivos; en el caso de la negociación podría añadirse más rondas de negociación de tal manera que los nodos lleguen a un acuerdo un mayor número de rondas de propuestas.

Proyectos posteriores podrían dirigirse a la aplicación del proyecto en grandes escenarios con la optimización anteriormente señalada, hay muchas aplicaciones en las cuales se puede utilizar pero se considera que el proyecto se acomoda a un escenario en donde no haya ninguna intervención por parte del usuario más que leer información mostrada por el sistema como una red de sensores; ya que el nodo sería autónomo de elegir a qué sistema unirse de acuerdo a sus necesidades mientras que en un escenario con usuarios que tomen la decisión sobre la unión del nodo al sistema, no se le encuentra mayor utilidad.

A. Anexo: Metodología SADT

Para definir los submódulos que fueron programados y las relaciones que hay entre ellos se seleccionó una metodología de diseño para enlazar cada bloque de programación dentro de la solución propuesta en este trabajo, para ello se usó una técnica jerárquica para describir con mayor detalle los componentes y elementos claves del sistema.

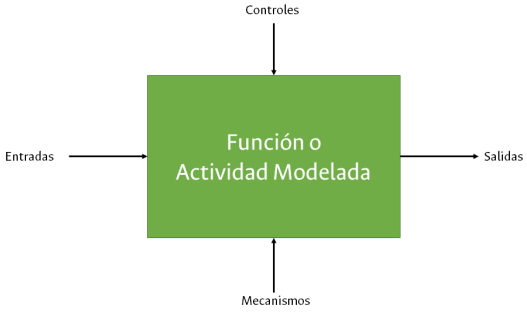

Figura A-1.: Diagramas SADT.

La Técnicas de Diseño y Análisis Estructurado - SADT (por sus siglas en inglés) [46] es un lenguaje gráfico para describir sistemas basados en relaciones jerárquicas, como finalidad un sistema modelado con SADT se enfoca en identificar las fase de análisis y diseño de sistemas, utiliza un enfoque de arriba hacia abajo para describir sistemas, posteriormente evolucionó y es conocido dentro de los estándares de IDEF, con la identificación IDEF0 continuando con las bases de la técnica SADT como herramienta en la ingeniería de software.

Esta técnica cuenta con cuatro elementos para definir las funciones o actividades a modelar, como se ve en la figura **A-1**, la descripción de cada uno de ellos es la siguiente:

- **Controles:** Son las elementos que agregan restricciones a las actividades
- **Entradas:** Son los elementos que son transformados por las actividades
- **Salidas:** Son las entradas transformadas

- **Mecanismos:** Son los elementos que llevan a cabo la actividad

Se presentan los diagramas detallados de cada uno de los objetivos de esta tesis usando SADT como herramienta descriptiva de los submódulos desarrollados. Para el desarrollo de este trabajo se crearon las formas descrita sobre la figura **A-2**.

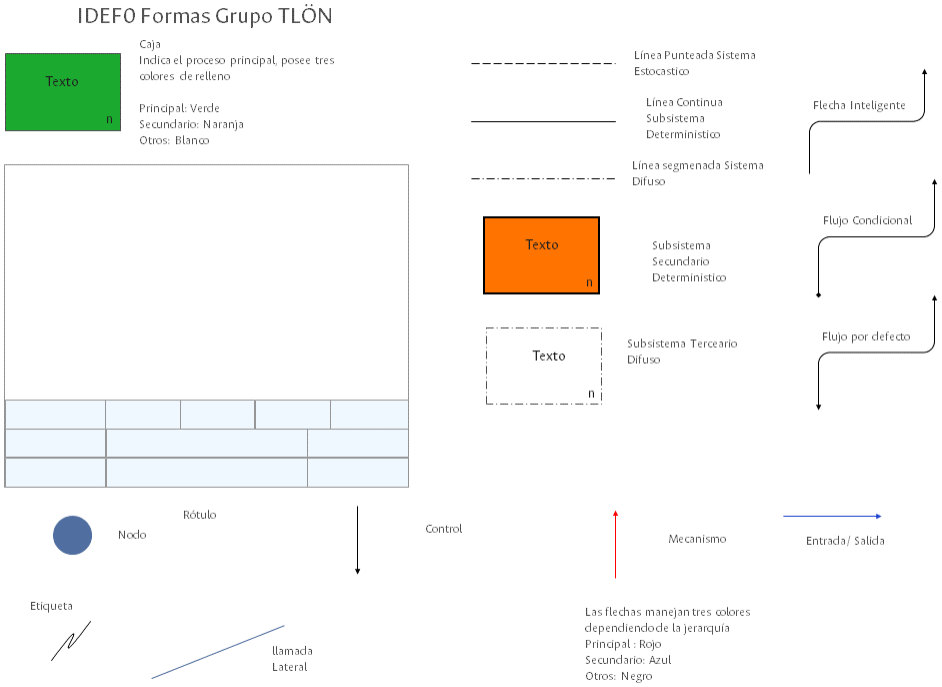

Figura A-2.: Formas SADT TLÖN.

B. Anexo: Programas en Python

Se adjuntan en CD los pogramas realizado en python los cuales permiten realizar todos los procesos que llevaron a cumplir los objetivos de esta tesis.

C. Anexo: Resultados

Para los resultados obtenidos se dispuso de dos nodos unidos mediante una conexión TCP. Se colocaron dos programas que controlaban la toma de decisiones y luego la negociación. Estos programas son pruebastesisclient.py y pruebastesisserver.py, cada uno en el nodo cliente y servidor respectivamente. pruebastesisserver.py llamaba a su vez a los programas DecisionScoringServer.py y NegotiationServertest.py y pruebastesisclient.py llamaba a DecisionScoringclient.py y NegotiationClienttest.py. Con la activación de los programas en cada uno de los nodos inicialmente se llegaba a un loop que hacía que los programas se ejecutaran 100 veces. Luego el programa realizaba la toma de decisiones y ese dato se imprimía en un archivo .txt para su procesamiento en Excel. Luego se pasaba a la toma de decisiones, esto se imprimió en un archivo .txt, y se aprovechó el programa en python para realizar el promedio de cada escenario el cual se encuentran al final de los archivos para un procesamiento más sencillo de la información. Al final los resultados fueron puestos en Excel. Se aprovechó también el uso de python para las gráficas que se encuentran en el cuerpo de la tesis, específicamente las figuras **5-10**, **5-11** y **5-12**.

D. Anexo: Manual Programas

Se adjuntan el manual de instalación de todos los programas, esto va desde la instalación del sistema operativo en la Raspberry hasta la puesta en marcha del programa en cada nodo.

Bibliografía

[1] Grupo de investigación TLÖN.: *TLÖN - Grupo de Investigación en Redes de Telecomunicaciones Dinámicas y Lenguajes de Programación Distribuidos*. `http://tlon.unal.edu.co/`. 2015. – [Accesado 6-Abril-2017]

[2] Agharazi, Hanieh ; Kolacinski, Richard M. ; Theeranaew, Wanchat ; Loparo, Kenneth A.: A swarm intelligence-based approach to anomaly detection of dynamic systems. En: *Swarm and Evolutionary Computation* (2018)

[3] Ali, Hasnain ; Kar, Arpan K.: Discriminant Analysis using Ant Colony Optimization–An Intra-Algorithm Exploration. En: *Procedia Computer Science* 132 (2018), p. 880–889

[4] Alonso-Monsalve, Saúl ; García-Carballeira, Félix ; Calderón, Alejandro: Fog computing through public-resource computing and storage. En: *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on* IEEE, 2017, p. 81–87

[5] Arboleda, Juan Pablo S.: *Altruismo y solidaridad en el Estado de Bienestar*, Universitat Autónoma de Barcelona, Tesis de Grado, 2008

[6] Aumann, Robert J.: Game theory. En: *Game Theory*. Springer, 1989, p. 1–53

[7] BAL, HENRI E. ; Grune, Dick ; Jacobs, Ceril J. ; Langendoen, Koen G.: *Concepts and notations for concurrent programming*. Vol. 21. ACM Computing Surveys, 1989

[8] Barbeau, Michael ; Kranakis, Evangelos: *Principles of Ad Hoc Networking*. Vol. 1. Carleton University, Canada : Wiley, 2007

[9] Berumen, S. A. ; Llamazares Redondo, F.: La utilidad de los métodos de decisión multicriterio (como el AHP) en un entorno de competitividad creciente. En: *Cuadernos de Administración* 20 (2007), Nr. 34, p. 65–87

[10] Bonabeau, Eric ; Dorigo, Marco ; Theraulaz, Guy: Inspiration for optimization from social insect behaviour. En: *Nature* 406 (2000), Nr. 6791, p. 39

[11] Bonomi, Flavio ; Milito, Rodolfo ; Zhu, Jiang ; Addepalli, Sateesh: Fog computing and its role in the internet of things. En: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* ACM, 2012, p. 13–16

[12] Buchegger, Sonja ; Le Boudec, J-Y: Self-policing mobile ad hoc networks by reputation systems. En: *IEEE Communications Magazine* 43 (2005), Nr. 7, p. 101–107

[13] Ceballos, Henry Z. ; Ortiz, Jorge E.: Servicios en comunicaciones de emergencia. En: *Ingenio Magno* 4 (2014), Nr. 1

[14] Chiong, Raymond: *Nature-inspired algorithms for optimisation*. Vol. 193. Springer, 2009

[15] Coello, C. a C. ; Lamont, G. B. ; Veldhuizen, D. a V.: *Evolutionary Algorithms for Solving Multi-Objective Problems Second Edition*. California : Springer Science & Business Media, 2007

[16] Crawford, V.: A note on the Zeuthen-Harsanyi theory of bargaining. En: *Journal of Conflict Resolution* 24 (1980), Nr. 3, p. 525–535

[17] Dastjerdi, Amir V. ; Buyya, Rajkumar: Fog computing: Helping the Internet of Things realize its potential. En: *Computer* 49 (2016), Nr. 8, p. 112–116

[18] Deb, K.: *Multi-Objective Optimization Using Evolutionary Algorithms*. Vol. 16. California : John Wiley & Sons, 2001

[19] Dorigo, Marco ; Birattari, Mauro: Ant colony optimization. (2011), p. 36–39

[20] Du La, Quang ; V.Ngo, Mao ; Quang Dinh, Thinh ; Quek, Tony Q. ; Shin, Hyundong: Enabling Intelligence in Fog Computing to Achieve Energy and Latency Reduction. En: *Digital Communications and Networks* (2018)

[21] Dudkowski, D. ; Marrón, P. J. ; Rothermel, K.: An efficient resilience mechanism for data centric storage in mobile ad hoc networks. En: *7th International Conference on Mobile Data Management (MDM'06)* IEEE, 2006, p. 7–7

[22] Fitzek, F. ; Katz, M.: *MOBILE CLOUDS: Exploiting Distributed Resources in Wireless,Mobile and Social Networks*. The Atrium : John Wiley & Sons, 2013

[23] Fonseca, C. M. ; Fleming, P. J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. En: *ICGA* Vol. 93 Citeseer, 1993, p. 416–423

[24] Fonseca, Carlos Manuel Mira d.: *Multiobjective genetic algorithms with application to control engineering problems.*, University of Sheffield, Tesis de Grado, 1995

[25] G., Coulouris ; J., Dollimore ; T., Kindberg: *Sistemas Distribuidos. Conceptos y Diseño, 3Â^a edición*. Vol. 2. Pearson Educación, 2001

[26] Gaertner, W.: *A primer in social choice theory: Revised edition.* Oxford : Oxford University Press, 2009

[27] Gómez, N. A. ; Maldonado, C. E.: Sistemas bio-inspirados: Un marco teórico para la ingeniería de sistemas complejos. En: *Bogotá: Editorial Universidad del Rosario* (2011)

[28] Hatzivasilis, G. ; Manifavas, C.: Building trust in ad hoc distributed resource-sharing networks using reputation-based systems. En: *Informatics (PCI), 2012 16th Panhellenic Conference on* IEEE, 2012, p. 416–421

[29] Hobbes, Thomas: *Leviathan: with selected variants from the Latin edition of 1668.* Vol. 2. Hackett Publishing Company, 1994

[30] Ilyas, Mohammad: *The Handbook of Ad Hoc Wireless Networks.* Vol. 1. Florida Atlantic University, Boca Raton Florida : CRC Press, 2003

[31] Javad, M. ; Khalaj, B. ; Katz, M. ; Fazelnia, G. ; Karimi, Parishad ; Del Ser, J.: Mobile clouds: How to find opportunities, 2012, p. 170–172

[32] Johnson, D. ; Aichele, C. ; Ntlatlapa, N.: A simple pragmatic approach to mesh routing using BATMAN. En: *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries* 1 (2008), p. 10

[33] Kelly, Sean Dieter T. ; Suryadevara, Nagender K. ; Mukhopadhyay, Subhas C.: Towards the implementation of IoT for environmental condition monitoring in homes. En: *IEEE Sensors Journal* 13 (2013), Nr. 10, p. 3846–3853

[34] Khayyat, Khalid M. ; Gebali, Fayez: Cross-layer modeling of wireless ad hoc networks in the presence of channel noise. En: *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE* IEEE, 2009, p. 1–6

[35] Kuhn, Thomas S.: *La estructura de las revoluciones científicas.* Vol. 3. Fondo de cultura económica México DF, 2013

[36] Kui, Xiaoyan ; Sun, Yue ; Zhang, Shigeng ; Li, Yong: Characterizing the Capability of Vehicular Fog Computing in Large-scale Urban Environment. En: *Mobile Networks and Applications* (2018), p. 1–18

[37] Kumar, S. ; Basavaraju, T. ; Puttamadappa, C.: *Ad Hoc Mobile Wireless Network: Principles, Protocols and Applications.* Boca Raton : CRC Press, 2007

[38] Kurose, James F. ; Ross, Keith W.: *Computer networking: a top-down approach.* Vol. 4. Addison Wesley Boston, USA, 2009

[39] Lawton, George: Machine-to-machine technology gears up for growth. En: *Computer* 37 (2004), Nr. 9, p. 12–15

[40] Liang, Chengchao ; Yu, F R.: Wireless network virtualization: A survey, some research issues and challenges. En: *IEEE Communications Surveys & Tutorials* 17 (2015), Nr. 1, p. 358–380

[41] Lindner, M. ; Wunderlich, S. ; S., Eckelmann.: *B.A.T.M.A.N (Better Approach to Mobile Ad-Hoc Network)*. `http://www.open-mesh.org/`. 2011. – [Accesado 9-Feb-2016]

[42] Liu, Ming T.: Distributed loop computer networks. En: *Advances in computers* Vol. 17. Elsevier, 1978, p. 163–221

[43] Loo, J. ; Lloret, J. ; Hamilton, J.: *Mobile Ad Hoc Networks: Current Status and Future Trends*. Boca Raton : CRC Press, 2012

[44] Luce, R. D. ; Raiffa, H.: *Games and Decisions: Introduction and Critical Survey*. New York : John Wiley & Sons, 1957

[45] MacArthur, Robert: Fluctuations of animal populations and a measure of community stability. En: *ecology* 36 (1955), Nr. 3, p. 533–536

[46] Marca, David A. ; McGowan, Clement L.: *SADT: structured analysis and design technique*. McGraw-Hill, Inc., 1987

[47] Moreno JimÃ©nez, J.: El Proceso Analítico Jerárquico (Ahp). Fundamentos, Metodología Y Aplicaciones. En: *Recta* 1 (2002), p. 21–53

[48] Nagel, T.: Rawls on justice. En: *The Philosophical Review* (1973), p. 220–234

[49] Niswar, M. ; Sabri, A. A. ; Warni, E. ; N., Musa M.: Memory sharing management on virtual private server. En: *ICT for Smart Society (ICISS), 2013 International Conference on* IEEE, 2013, p. 1–4

[50] Oehlmann, Fabian: *Simulation of the B̈etter Approach to Mobile Adhoc Networking"Protocol*, Technische Universität München, Tesis de Grado, 2011

[51] Orozco, Ana M. ; Llano, Gonzalo ; Michoud, Roger: Redes vehiculares Ad-hoc: aplicaciones basadas en simulación. En: *Ingenium* 6 (2012), Nr. 12, p. 11–22

[52] Orozco, Oscar A. ; Llano, Gonzalo: OSA: A Vanet application focused on fuel saving and reduction of CO2 emissions. En: *Sistemas & Telemática* 12 (2014), Nr. 29, p. 25–47

[53] Ospina-López, Juan P. ; Ortiz-Triviño, Jorge E.: Caracterización de un clúster y sus recursos en una red Ad Hoc a partir de la distribución geométrica truncada. En: *Iteckne* 12 (2015), Nr. 1, p. 68–75

[54] Ozdaglar, A.: *Existence of a Nash Equilibrium, Extensive Form Games, Nash Bargaining Solution.* https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-254-game-theory-with-engineering-applications-spring-2010/lecture-notes/. 2010. – [Accesado 22-Sep-2016]

[55] Parra, José B ; Santiago, Evelinda ; Murillo, Misael ; Atonal, Candy: Estrategias para negociaciones exitosas. En: *e-Gnosis* 8 (2010)

[56] Pitt, Jeremy ; Busquets, Dídac ; Riveret, Régis: The pursuit of computational justice in open systems. En: *AI & SOCIETY* 30 (2015), Nr. 3, p. 359–378

[57] Pliego, Odette A.: *Programación Lineal Multiobjetivo: Análisis, Técnicas y Casos de Aplicación*, Universidad Nacional Autónoma de México, Tesis de Grado, 2012

[58] Rawls, John: *Teoría de la justicia*. Fondo de cultura económica, 2012

[59] Ray, Saikat ; Carruthers, Jeffrey B. ; Starobinski, David: Evaluation of the masked node problem in ad hoc wireless lans. En: *IEEE Transactions on mobile computing* 4 (2005), Nr. 5, p. 430–442

[60] Raychaudhuri, Dipankar ; Gerla, Mario: *Emerging Wireless Technologies and the Future Mobile Internet*. Cambridge : Cambridge University Press, 2011

[61] Rezaei, B. A. ; Sarshar, N. ; V.P., Roychowdhury: Distributed Resource Sharing in Low-Latency Wireless Ad Hoc Networks. En: *IEEE/ACM Transactions on Networking* 18 (2010), Nr. 1, p. 190–201

[62] Roche, Hugo ; Vejo, Constantino: Análisis multicriterio en la toma de decisiones. En: *Métodos Cuantitativos aplicados a la administración. Analisis multicritério–AHP. 2004. Material apoyo AHP, 11 f* (2005)

[63] Roughgarden, T.: Algorithmic game theory. En: *Communications of the ACM* 53 (2010), Nr. 7, p. 78–86

[64] Royer, Elizabeth M. ; Toh, Chai-Keong: A review of current routing protocols for ad hoc mobile wireless networks. En: *IEEE personal communications* 6 (1999), Nr. 2, p. 46–55

[65] Saaty, T.: How to make a decision: The analytic hierarchy process. En: *European Journal of Operational Research* 48 (1990), Nr. 1, p. 9–26

[66] Şahin, Erol: Swarm robotics: From sources of inspiration to domains of application. (2004), p. 10–20

[67] Salman, Ola ; Elhajji, Imad ; Chehab, Ali ; Kayssi, Ayman: IoT Survey: An SDN and Fog Computing Perspective. En: *Computer Networks* (2018)

[68] Sánchez, Joaquín F ; Ospina, Juan P. ; Gonzalez, John E. ; Ortiz, Jorge E.: Sociability as a decision model for artificial agents. En: *Proceedings of the 2017 International Symposium on Industrial Engineering and Operations Management (IEOM)* IEOM Society, 2017

[69] Sarmiento, Danilo Alfonso L. ; Torres, José I ; Ramírez, Ramón E: Estado del arte de IPTV y consideraciones técnicas para su migración a IPv6 en Colombia. En: *Redes de Ingeniería* 2 (2011), Nr. 1, p. 45–64

[70] Soto A., M. R.: Teoría de los juegos: Vigencia y limitaciones. En: *Revista de Ciencias y Sociales* 11 (2005), Nr. 3, p. 497–506

[71] Spinoza, B: *Ética demostrada según el orden geométrico*. Vol. 4. N2kt, 1984

[72] Stewart, A. J. ; Plotkin, J. B.: From extortion to generosity, evolution in the Iterated Prisoner's Dilemma. En: *Proceedings of the National Academy of Sciences* 110 (2013), Nr. 38, p. 15348–15353

[73] Suárez, Patricia ; Iglesias, Andrés ; Gálvez, Akemi: Make robots be bats: specializing robotic swarms to the Bat algorithm. En: *Swarm and Evolutionary Computation* (2018)

[74] Támara, L. J. ; Alzate, M. A.: Control of admission for AD HOC mobile network based on estimates available bandwidth. En: *Tecnura* 15 (2011), Nr. 29, p. 24–34

[75] Tanaka, M. ; Watanabe, H. ; Furukawa, Y. ; Tanino, T.: GA-based decision support system for multicriteria optimization. Systems, Man and Cybernetics. En: *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference* Vol. 2 IEEE, 1995, p. 1556–1561

[76] Thanapal, P. ; Saleem Durai, M. A.: A Survey of Mobile Cloud Computing for Extending Energy of Mobile Devices. 573 (2014), p. 549–555

[77] Toh, CK ; Kim, Dongkyun ; Oh, Sutaek ; Yoo, Hongseok: The controversy of selfish nodes in ad hoc networks. En: *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on* Vol. 2 IEEE, 2010, p. 1087–1092

[78] Toh, CK ; Kim, Dongkyun ; Oh, Sutaek ; Yoo, Hongseok: The controversy of selfish nodes in ad hoc networks. En: *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on* Vol. 2 IEEE, 2010, p. 1087–1092

[79] Troyano, Albert B.: Protocolos de encaminamiento en redes inalámbricas mesh: un estudio teórico y experimental. En: *Sevilla, España: Universidad de Sevilla* (2011)

[80] Upadhyay, Nitin: Fogology: What is (not) Fog Computing? En: *Procedia computer science* 139 (2018), p. 199–203

[81] Vaquero, Luis M. ; Rodero-Merino, Luis: Finding your way in the fog: Towards a comprehensive definition of fog computing. En: *ACM SIGCOMM Computer Communication Review* 44 (2014), Nr. 5, p. 27–32

[82] Vitoriano, B.: Teoría de la decisión: decisión con incertidumbre, decisión multicriterio y teoría de juegos. En: *Universidad Complutense de Madrid* (2007), p. 3–104

[83] Wibowo, Satriyo: Measuring IPv6 readiness on the most accessed web-based content provider in Indonesia. En: *ICT For Smart Society (ICISS), 2017 International Conference on* IEEE, 2017, p. 1–5

[84] Xiao, He ; Xia, Pang ; Da-rui, Zhu ; Chong-xin, Liu: Multi-objective reactive power optimization based on chaos particle swarm optimization algorithm. En: *Instrumentation and Measurement, Sensor Network and Automation (IMSNA), 2013 2nd International Symposium on* IEEE, 2013, p. 1014–1017

[85] Zárate, H. ; Sánchez, J. F. ; Ospina, J. P. ; Ortiz, J. E.: Sistema de telecomunicaciones social-inspirado mediante Comunidades de Agentes. En: *Congreso Internacional de Computación Colombia-México,*, 2015, p. 56–63

[86] Zhu, Jiang ; Chan, Douglas S. ; Prabhu, Mythili S. ; Natarajan, Preethi ; Hu, Hao ; Bonomi, Flavio: Improving web sites performance using edge servers in fog computing architecture. En: *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on* IEEE, 2013, p. 320–323