UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES
DEPARTAMENTO DE INGENIRERÍAN CIVIL EN INFORMÁTICA

UNIVERSIDAD DEL BÍO-BÍO

“Implementación de mecanismo de difusión de mensajes basado en contexto con WANET mediante Wi-fi Direct”

Proyecto de título para optar al título profesional de
Ingeniería Civil en Informática

Autor
Felipe Hidalgo Alvarado

Profesor Guía
Patricio Galdames Sepúlveda

17 de marzo de 2017

RESUMEN

Este proyecto se presenta para dar conformidad a los requisitos exigidos por la Universidad del Bío-Bío en el proceso de titulación para la carrera de Ingeniería Civil en Informática. El proyecto, titulado “Implementación de mecanismo de difusión de mensajes en una red WANET mediante Wi-fi Direct”, tiene como resultado principal una aplicación para el sistema operativo Android que permite difundir mensajes desde un punto geográfico a otro utilizando solamente dispositivos móviles que cuenten con la aplicación utilizando la tecnología Wi-fi Direct, en la que los dispositivos pueden conectarse entre ellos de forma rápida y segura, formando una red de tipo WANET y permitiendo la difusión del mensaje sin necesidad de estar conectado a una red telefónica o wi-fi física.

La aplicación se desarrolla pensando en un ambiente donde no existan las formas tradicionales de comunicación a distancia con dispositivos móviles, ya sea por la falta de infraestructura o el colapso de estas por distintos factores. Entregando una alternativa a la necesidad de comunicarse y recibir información. Al realizar una conexión directa entre los dispositivos que se encuentren en el rango de alcance es capaz de ir transportando el mensaje a través de ellos seleccionando el dispositivo más óptimo, basándose en el contexto, utilizando la ubicación geográfica e información del comportamiento del usuario del dispositivo sobre los lugares que frecuenta, hasta llegar al lugar donde se seleccionó como meta del mensaje, donde la aplicación comenzará la difusión de este entre todos los dispositivos disponibles en el rango de alcance, para así asegurar que el mensaje llegará a destino en el menor tiempo posible. Se ideo e implemento un algoritmo de ruteo capaz de cumplir con este objetivo.

Dado que los dispositivos móviles son una herramienta y tecnología masiva, es posible ir trazando una vía de transporte para el mensaje, gracias a la gran densidad de dispositivos disponibles en la actualidad, permitiendo así una difusión más rápida del mensaje y posicionando este método como una alternativa factible para su implementación y uso en los casos para los que fue pensada.

ABSTRACT

The present proyect gives conformity to the requeriments of the Universidad del Bío-Bío for the profesional title of Civil Engineering in Computer Science. The proyect, entitled "Implementation of context-based message dissemination mechanism with WANET via Wi-fi Direct", has as main result an application for the Android operating system that allows to spread messages from one geographical point to another using only mobile devices that have the application using Wi-fi Direct technology, in which the devices can connect to each other quickly and safely, forming a WANET network, allowing the message to be transmitted without having to be connected to a telephone network or physical Wi-Fi.

The application is developed in an environment where there are no traditional forms of remote communication with mobile devices, either due to the lack of infrastructure or the collapse of these by different factors. Providing an alternative to the need to communicate and receive information. The application, by making a direct connection between the devices that are in range, is able to carry the message through them by selecting the most optimal device, based on the context, using the geographical location and behavioral information of the user about the places he frequents, in order to ensure that the message reaches its destination in the shortest possible time, until it reaches to the place where it was selected as the target of the message, the application will begin to spread the message among all available devices in range.

Since mobile devices are a massive technology, it is possible to trace a transport path for the message, thanks to the high density of devices available today, allowing a faster dissemination of the message and positioning this method as a feasible alternative for its implementation and use in the cases for which it was intended.

Índice General

Índice de Tablas

Índice de figuras

1 Introducción

Este documento se presenta para dar conformidad a los requisitos exigidos por la Universidad del Bío-Bío en el proceso de titulación para la carrera de Ingeniería Civil en Informática. En él se detalla el desarrollo de una aplicación móvil para el sistema operativo Android y las pruebas de su funcionamiento en laboratorio. Esta aplicación permite enviar mensajes de un punto geográfico a otro trazando un camino utilizando dispositivos móviles, combinando la capacidad P2P que estos poseen, denominada *Wi-fi Direct[3]*, e información del contexto obtenida a través de localización geográfica por medio de GPS e información del comportamiento del usuario del dispositivo respecto de que edificios frecuenta.

Hoy en día la cantidad de usuarios de Smartphone está aumentando rápidamente conforme avanza el tiempo, todo esto de la mano de que estos dispositivos permiten estar en todo momento comunicados y al tanto de lo que sucede en todo el mundo. La forma tradicional de mantener esta comunicación es a través de redes celulares ya sea *2g, 3g* o incluso la tecnología más actual *4g*, también existiendo la posibilidad de conectar los dispositivos móviles a redes domésticas mediante *Wi-Fi[2]*. Cuando se cuenta con todas estas redes comunicarse es rápido y relativamente fácil, pero se está dependiendo de la disponibilidad de estas redes quedando las comunicaciones detenidas cuando no se cuenta con ellas. En este ámbito la tecnología Wi-Fi Direct se presenta como una alternativa para dar solución a este problema ya que permite una conexión directa entre dispositivos inalámbricos mediante Wi-Fi P2P sin necesidad de ninguna clase de infraestructura externa, permitiendo velocidades y distancias mucho mayores a tecnologías de similares características como lo es *Bluetooth[24]*.

Con una cantidad de dispositivos móviles con esta tecnología en aumento y la problemática de depender de redes que pueden no estar disponibles en algunos sectores o por algún factor colapsar y dejar de estar disponibles, se ha propuesto implementar una red de tipo WANET. Con esta red se busca de trazar un camino de dispositivo en dispositivo de un lugar geográfico a otro utilizando la localización geográfica de los dispositivos a través de GPS y el comportamiento habitual del usuario acerca de los lugares que visita.

Los principales puntos a tratar durante el desarrollo de este proyecto son:

- **Limitantes y soluciones:** descubrir las limitantes propias del *framework Wi-Fi Direct* disponible para Android y proponer sus soluciones.
- **Algoritmo:** diseñar y construir un algoritmo que sea capaz de cumplir con el objetivo general del proyecto.
- **Aplicación:** Diseñar y construir una aplicación para el sistema operativo Android la cual en conjunto al algoritmo diseñado componen un programa completamente operativo.
- **Experimentos:** pruebas de laboratorio diseñadas para obtener y documentar información relevante respecto a capacidades de la tecnología utilizada y efectividad del algoritmo diseñado.

El algoritmo de ruteo consiste en que cada usuario que recibe un mensaje decide si él lleva el mensaje lo más cercano posible al destino o bien un vecino de este. Esta decisión se basa en la ubicación de los usuarios y en el conocimiento previo de los lugares más frecuentados por los usuarios.

Se realizó una encuesta en la Universidad del Bío-Bío para conocer el comportamiento de los usuarios respecto a los lugares que más frecuentan dentro de las dependencias de la universidad. Estableciendo un promedio de edificios frecuentados por los usuarios cargando esta información al software.

Se demuestra como con la tecnología Wi-fi Direct se puede emular una red WANET entre los dispositivos, permitiendo trazar una ruta de manera dinámica hasta llegar al destino seleccionado para el mensaje.

El documento cuenta de 10 capítulos, los cuales están constituidos de acuerdo a la documentación de proyectos de desarrollo de software. El contenido de cada capítulo se describe a continuación.

- **Capítulo 2 "DEFINICIÓN DEL PROYECTO".** Este capítulo incluye lo relativo al proyecto en sí, los objetivos generales y específicos, la metodología utilizada y la descripción de la técnica utilizada para cumplir tales objetivos.

- **Capítulo 3 "ESTADO DEL ARTE".** Se describe las tecnologías que se ven implicadas en el proyecto, tales como los dispositivos móviles, el sistema operativo Android, las tecnologías de comunicación de mensajes existentes y finalmente, la tecnología base utilizada en la aplicación, Wi-fi Direct.

- **Capítulo 4 "ESPECIFICACIÓN DE REQUERIMIENTOS DEL SOFTWARE."** Se detalla lo relativo a la aplicación, como sus alcances, objetivos generales y específicos, los requisitos para el correcto funcionamiento de la aplicación y la descripción global del software.

- **Capítulo 5 "ANÁLISIS".** En este capítulo se presenta el comportamiento y funcionamiento del software, utilizando DIA como herramienta de modelado.

- **Capítulo 6 "UBBmessenger".** En este capítulo se explica; el algoritmo desarrollado para dar solución a la problemática presentada, además los aspectos generales de la aplicación "UBBmessenger", tales como; información del contexto utilizada y como fue recopilada, algoritmo de búsqueda del camino más óptimo utilizado y la conexión de dispositivos mediante Wi-fi Direct.

- **Capítulo 7 "PRUEBAS".** En este capítulo se especifican las pruebas que se realizan al software para comprobar la correcta funcionalidad de la aplicación.

- **Capítulo 8 "EXPERIMENTOS".** En este capítulo se realizan mediciones considerando distintas condiciones para la difusión del mensaje.

- **Capítulo 9 "CONCLUSIONES".** En este capítulo se realiza el contraste entre los objetivos del proyecto, del sistema planteado y de lo alcanzado al final del proyecto. Se incluyen conclusiones generales del proyecto desde los puntos de vista académico y personal.

- **Capítulo 10 "BIBLIOGRAFÍA".** Se indican las fuentes de información utilizadas durante el desarrollo del software.

- **Capitulo 11 "Anexos".** Se adjuntarán elementos relevantes para el desarrollo de este proyecto.

2 DEFINICIÓN DEL PROYECTO

2.1 Objetivos del Proyecto

2.1.1 **Objetivo General**

Implementar y simular una red del tipo WANET la que permita difundir mensajes sin necesidad de una infraestructura física, aprovechando las capacidades de los dispositivos Android de comunicarse entre si utilizando Wi-fi Direct.

2.1.2 Objetivos Específicos

- Investigación de protocolos de ruteo en redes WANET.
- Proponer un protocolo de ruteo que se ajuste a las capacidades de la aplicación.
- Investigación de librería Wi-fi Direct de Android.
- Investigación del contexto en el que se implementará la red, permitiendo aportar datos estadísticos al desarrollo de la aplicación
- Investigación de los algoritmos de búsqueda del camino más corto para ser implementado en la aplicación.
- Representar el proyecto en un ambiente de simulación de redes de eventos discretos NS-3.

2.2 Ambiente de Ingeniería de Software

2.2.1 Metodología de desarrollo

La metodología utilizada se basará en metodologías ágiles, donde se definen tareas a realizar por periodos de tiempo de 1 a 2 semanas las cuales deben terminar antes de pasar a las nuevas tareas. La metodología en específico utilizada es KANBAN dada su flexibilidad a que el proyecto sea desarrollado por una sola persona.

KANBAN se centra en cinco prácticas fundamentales:

- Visualizar

Visualizar el flujo de trabajo y hacerlo visible es la base para comprender cómo avanza el trabajo. Sin comprender el flujo de trabajo, realizar los cambios adecuados es más difícil.

- Limitar el trabajo en curso

Limitar el trabajo en curso implica que un sistema de extracción se aplica en la totalidad o parte del flujo de trabajo. El sistema de extracción actúa como uno de los principales estímulos para los cambios continuos, incrementales y evolutivos en el sistema.

- Dirigir y gestionar el flujo

Se debe supervisar, medir y reportar el flujo de trabajo a través de cada estado. Al gestionar activamente el flujo, los cambios continuos, graduales y evolutivos del sistema pueden ser evaluados para tener efectos positivos o negativos.

- Hacer las Políticas de Proceso Explícitas

Configurar las reglas y directrices del trabajo, entender las necesidades y asegúrese de seguir las reglas.

- Utilizar modelos para reconocer oportunidades de mejora

Cuando los equipos tienen un entendimiento común de las teorías sobre el trabajo, el flujo de trabajo, el proceso y el riesgo, es más probable que sea capaz de construir una comprensión compartida de un problema y proponer acciones de mejora que puedan ser aprobadas por consenso.

2.2.2 Herramientas de apoyo al desarrollo de software

Para el desarrollo del proyecto se utilizarán las siguientes herramientas:

- **Android Studio 2.2.2:** Es un entorno de desarrollo integrado para la plataforma Android, fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android[1].
- **SDK Android (Android Developer Kit):** Provee las bibliotecas y herramientas de desarrollo necesarias para crear, probar y depurar aplicaciones Android.
- **Java Development Kit** (**JDK**): Es un software que provee herramientas de desarrollo para la creación de programas en Java que se integra a Android Studio.
- **Smartphone Samsung Galaxy S7:** Teléfono móvil utilizado para depurar y probar el correcto funcionamiento de la aplicación.
- **Smartphone Samsung Galaxy S5 neo:** Teléfonos móviles facilitados por la Universidad del Bío-Bío, utilizados para depurar y probar el correcto funcionamiento de la aplicación además de realizar las pruebas de laboratorio.
- **DIA :** Software utilizado para la creación de los distintos diagramas presentados en el informe.

2.3 Definiciones, Siglas y Abreviaciones

2.3.1 Definiciones

- **Peer:** Cliente que se conecta mediante Wi-fi Direct, al Group owner y es evaluado como posible siguiente portador del mensaje.
- **Group Owner:** Dispositivo que actúa como punto de acceso(AP), es el portador del mensaje, escanea los peers disponibles en el rango y los evalúa, como posible siguiente portador.
- **Portador:** Es quien lleva o trae algo de una parte a otra [6]. En el caso del presente proyecto lo que se está portando es un mensaje.
- **Smartphone:** Teléfono inteligente en español. Smartphone es un teléfono móvil con capacidades de almacenaje y procesamiento similares al de una computadora [11].
- **UBBmessenger:** Aplicación desarrollada en el presente proyecto.
- **Wi-Fi:** Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica. Los dispositivos habilitados con Wi-Fi pueden conectarse a internet a través de un punto de acceso de red inalámbrica [2].
- **Wi-Fi Direct:** Norma que permite que varios dispositivos WI-Fi se conecten entre sí sin necesidad de un punto de acceso intermedio [3].

2.3.2 Siglas y Abreviaciones

- **WANET:** *Wireless ad hoc network.* Red inalámbrica descentralizada ya que no requiere de una infraestructura física existente [4].
- **App:** Abreviación de Application; Aplicación en español. Tipo de programa informático diseñado como herramienta para permitir a un usuario realizar diversos tipos de trabajos [5].
- **P2P:** *Peer-to-Peer;* Par a Par en español. Es una red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos se comportan como iguales entre sí [7].
- **TCP:** *Transmission Control Protocol;* Protocolo de Control de Transmisión en español. Protocolo fundamental de Internet, que aporta transporte confiable y bidireccional de datos [8].

- **OS:** *Operating System;* Sistema Operativo en español. Conjunto de órdenes y programas que maneja los recursos de hardware y software de una sistema computacional [9].
- **AP:** *Access Point;* Punto de acceso en español. Un punto de acceso inalámbrico, es un dispositivo de red que interconecta equipos de comunicación inalámbricos [10].
- **API:** *Application Programing Interface;* Interfaz de programación de aplicaciones en español. Es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción [15].

2.4 Servicio de Mensajería

Es una forma de comunicación generalmente en tiempo real entre dos o más personas basada en un texto, este texto es enviado a través de dispositivos conectados a una red como Internet.

2.5 Trabajos relacionados

A la fecha no se ha encontrado ningún estudio que proponga un protocolo de ruteo aprovechando las capacidades de Wi-Fi Direct.

3 ESTADO DEL ARTE

En este punto se definen las tecnologías usadas para el desarrollo de la aplicación "UBBmessenger", los algoritmos de búsqueda estudiados, el ambiente de simulación de redes discretas, como también las tecnologías que existen actualmente para el envío de mensajes.

3.1 Teléfono inteligente

El teléfono Inteligente, desde ahora smartphone (contraparte en inglés) es un tipo de teléfono móvil, con mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora, con mayor conectividad que un teléfono móvil convencional, capacidad de navegación web y de ejecución de complejas aplicaciones [11].

Los fabricantes cada vez ofrecen smartphones con mayores y mejores capacidades tanto de procesamiento, conectividad y multimedia, con tal de ofrecer a los usuarios una mayor fluidez y prestaciones para mejorar la experiencia en su uso.

El uso de smartphones en el mundo se encuentra en un constante aumento. La compañía IHS Technology ha presentado los resultados de su estudio "Connected Device Market Monitor", en el que se recoge el número total de dispositivos conectados en el mundo dividido por categorías. En concreto, actualmente existen 8.100 millones, de los cuales 3.377 corresponden a smartphones, la mayor categoría de todas. Le siguen los PC, con 1.845 millones, el hardware de audio (1.429), tablets (733), los dispositivos conectados a la televisión (507) y las smart TV (289). Todo ello, según la empresa, equivale a una media de cuatro dispositivos por hogar [12]. Desplazando a los que por años fueron los dispositivos para acceso a la información predilectos las computadoras personales.

La mayoría de los smartphones funcionan bajo uno de los tres sistemas operativos que actualmente concentran la mayoría del mercado móvil; Android, iOS y Windows Phone. El mercado actual de SO para smarphones lo domina hasta el tercer trimestre de 2016 Android con un 87,8% de las ventas en el mercado mundial [13]. Es sobre este último sistema operativo sobre el cual se desarrollará la aplicación "UBBMessenger".

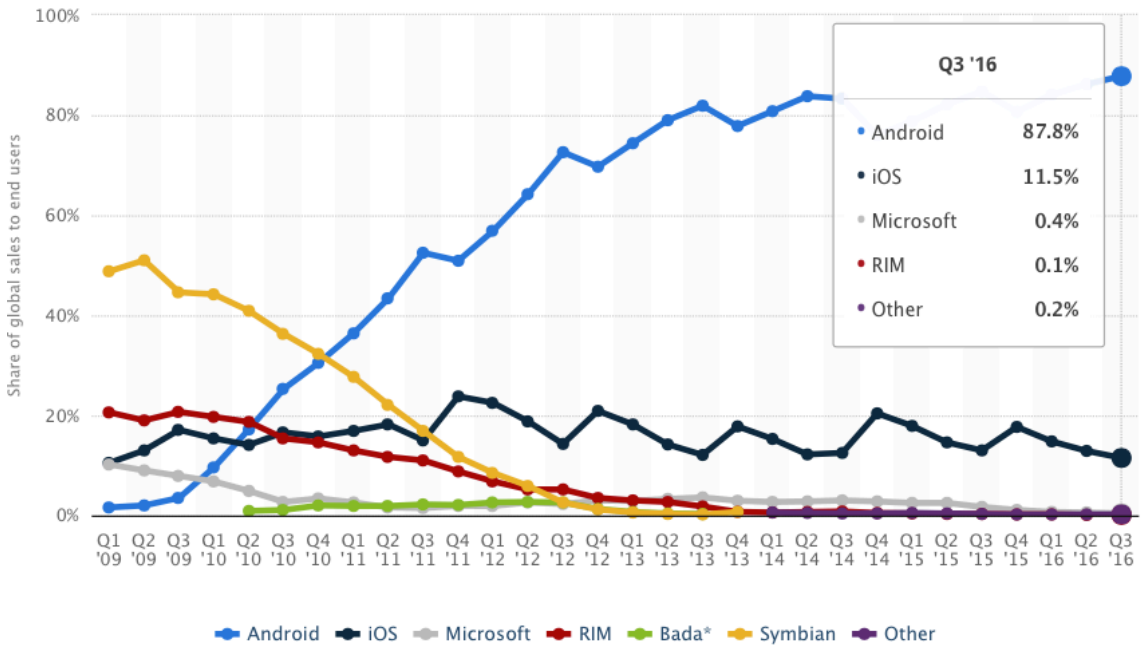

Figura 1: Distribución de mercado S.O móviles (ITuser)

3.2 Sistema Operativo Android

Android OS, o simplemente Android, es un sistema operativo móvil basado en el núcleo de Linux. Fue principalmente diseñado para dispositivos móviles con pantalla táctil, como smartphones y tablets. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente, luego en 2005, la adquirió. Android fue presentado en 2007 junto la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución hasta la versión 5.0, luego cambió al entorno Android Runtime (ART).

Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica, un framework OpenCore, una base de datos relacional SQLite, una Interfaz de

programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic [14].

Android ofrece nuevas posibilidades para aplicaciones móviles al proveer un entorno de desarrollo abierto, basado también en un kernel de código abierto Linux a diferencia de otros sistemas operativos que están construidos sobre otros sistemas propietarios que a menudo dan prioridad a las aplicaciones nativas por sobre las creadas por terceros y restringen la comunicación entre las aplicaciones y los datos nativos del teléfono, por ejemplo; Android permite usuarios avanzados sustituir aplicaciones nativas por una alternativa desarrollada por terceros. Todo esto sólo ha hecho crecer a Android en cantidad de usuarios y desarrolladores alrededor del mundo.

3.2.1 Arquitectura Android

Para poder desarrollar aplicaciones Android es importante saber cómo está estructurado el SO. Su Arquitectura está formada por varias capas que facilitan al desarrollador la creación de aplicaciones, además, esta distribución permite acceder a las capas de más bajo nivel gracias al uso de librerías, facilitando el trabajo del desarrollador, evitándole el trabajo de programar las funcionalidades necesarias para que la aplicación haga uso de los componentes de hardware del dispositivo móvil.

Las distintas capas están conformadas como se mencionan a continuación:

- **Aplicaciones**: Se incluyen todas las aplicaciones del dispositivo tanto las que poseen interfaz de usuario como las que no, las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros que vengan instaladas previamente en el dispositivo o que el usuario ha instalado. Todas las aplicaciones están escritas en lenguaje de programación Java.
- **Marco de trabajo de aplicaciones**: Los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso

de esas capacidades. Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

- **Bibliotecas:** Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library, bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.
- **Runtime de Android**: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik, la que ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecutaba hasta la versión 5.0 archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx". Desde la versión 5.0 utiliza el ART, que compila totalmente al momento de instalación de la aplicación.
- **Núcleo Linux:** Android depende de un kernel Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software [14].

Se puede observar que Android proporciona un entorno muy poderoso para el desarrollo de aplicaciones permitiéndonos tener acceso fácil he intuitivo a todos sus componentes.

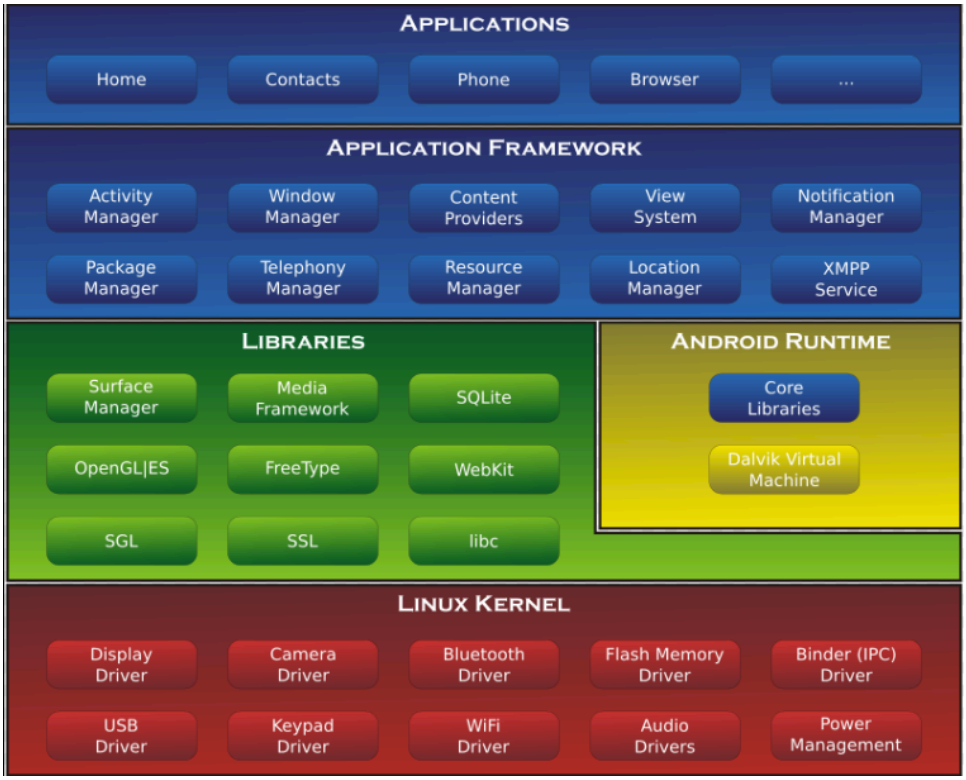

Figura 2: Arquitectura Android (Wikipedia)

3.3 Niveles API y versiones de Android

El nivel de API es un valor entero que identifica de forma unívoca el framework ofrecido por la versión de la plataforma Android. El framework consiste en: paquetes y clases, elementos para el fichero Manifest, elementos para los recursos, Intents y permisos.

Según el nivel de la API en la que se haya desarrollado la aplicación será necesario que el dispositivo tenga instalado una versión mínima del Sistema Operativo. Cundo se construye una aplicación para Android, en el fichero “Mainfest.xml” se deben declarar el nivel de API objetivo y el nivel de API mínimo para que la aplicación pueda ser ejecutada.

Esta configuración se utiliza para asegurar que la funcionalidad necesaria para ejecutar la aplicación está disponible en el dispositivo Android al momento de la instalación, este revisa los requisitos de API mínimos en el fichero “Mainfest.xml” y si el dispositivo cuenta con la versión de Android igual o superior al mínimo requerido la aplicación se

instalara de manera exitosa, de lo contrario se alerta al usuario que su dispositivo no es compatible con aquella aplicación.

Cada versión del SO Android se ha desarrollado bajo nombres inspirados en "postres" en inglés y cada nombre comienza por una letra distinta en orden alfabético, además de esto se le asigna un numero de versión y un nivel de api [14] y son diseñadas para tener retro compatibilidad con versiones anteriores del sistema operativo.

Cada nueva versión incorpora mejoras en rendimiento, nuevas funcionalidades y desecha otras que ya no son relevantes, generalmente por que han sido reemplazadas por una más versátil y fácil de utilizar.

Letra	**Nombre**	**Versión Android**	**Nivel API**
A	Apple Pie	1.0	1
B	Banana Bread	1.1	2
C	Cupcake	1.5	3
D	Donut	1.6	4
E	Éclair	2.0 - 2.1 - 2.1.X	5 - 6 - 7
F	Froyo	2.2	8
G	Gingerbread	2.3 - 2.3.4	9 - 10
H	Honeycomb	3.0 - 3.1 - 3.2	11 - 12 - 13
I	Ice Crean Sabdwich	4.0 - 4.0.4	14 -15
J	Jelly Bean	4.1 - 4.2 - 4.3	16 - 17 - 18
K	KitKat	4.4 - 4.4W	19 - 20
L	Lollipop	5.0 - 5.1	21 - 22
M	Marshmallow	6.0	23
N	Nougat	7.0	24

Tabla 1 Versiones Android

3.4 Tecnologías de envío de datos

Transmisión de datos o comunicación digital es la transferencia física de datos o bits por un canal de comunicación punto a punto o punto a multipunto.

Existen múltiples formas de enviar datos, la mayoría de estas formas requiere de una infraestructura instalada y disponible previamente; ya sea una red cableada o inalámbrica. Cada una con sus ventajas y desventajas, costos de implementación, rango de alcance y rendimiento [16].

Cuando se trata de dispositivos móviles son las tecnologías inalámbricas las que toman protagonismo.

3.4.1 Redes Móviles

Las redes móviles son un tipo de comunicación inalámbrica a través de ondas electromagnéticas. La red es distribuida sobre áreas de terreno llamadas celdas, cada una es provista por al menos un transceptor fijo, conocido como estación base. Cada base provee a la celda con cobertura de una red que permite la transmisión de voz, datos etc. Las redes para celulares funcionan de esta manera, dándole acceso a los smartphones no solo a llamadas de voz, sino que también a internet o envío de mensajes

Las redes celulares funcionan a distintas frecuencias y rangos de alcance, a menor frecuencia cuentan con un mayor rango de alcance, pero una menor capacidad de transmisión, las frecuencias más altas se ven limitadas en su rango de alcance incluso se pueden ver afectadas por una pared, pero poseen una mayor capacidad de transmisión [17].

De esto surgieron distintas tecnologías para permitir la conexión de datos para los dispositivos móviles.

- **2G:** Constituye el comienzo de la telefonía celular. Incluye los estándares GSM creado en 1991, 2.5G (GPRS) y 2.75G (EDGE). Se emplean señales digitales entre los dispositivos y las torres incrementando la capacidad. Permite la encriptación de las conversaciones y de los datos enviados

digitalmente, de forma tal que solo a quien se le envía puede recibirlos y leerlos. Introduce el uso de los servicios de datos como la navegación WAP, los SMS y MMS permite alcanzar velocidades de transmisión que van desde los 56 a los 114 kbps.

- **3G:** Se conoce como redes 3G las que permiten una transferencia de datos como mínimo de 200 kbps. Incluye las redes 3G, 3.5G (HSDPA) y 3.75G (HSUPA). Comenzó su uso comercial en el 2001, implementándose muy lentamente en la telefonía celular. Esta es la tecnología que posibilitó el auge y uso masivo de los smartphones.
- **4G:** Las redes 4G son estándares creados para perfeccionar los usados en 3G, introduciendo el estándar LTE que soporta conectividad a altas velocidades de movimiento, mejora los servicios y permite una mayor integración con los protocolos existentes. Puede alcanzar velocidades de descarga de hasta 300 mbps [18].

3.4.2 Wi-Fi (IEEE 802.11 WLAN)

Wi-Fi es una marca de la Alianza Wi-Fi, la organización comercial que adopta, prueba y certifica los equipos que cumplen con los estándares 802.11 relacionados a redes inalámbricas de área local (WLAN) [2].

Una WLAN es una red inalámbrica de dispositivos que se comunican con señales de radio frecuencia. Casi todos los dispositivos con capacidad WLAN basados en el estándar IEEE 802.11 son certificados Wi-Fi, lo cual es la razón por la que Wi-Fi se ha convertido en sinónimo de IEEE 802.111 WLAN.

IEEE 802.11 Es un conjunto de especificaciones de control de acceso a medios y capa física para implementar redes inalámbricas de área local. Trabaja en las frecuencias de bandas 900 Mhz, 2.4 , 3.5, 5 y 60 GHz. Fue creada por el Instituto de Electricidad e Ingenieros Electrónicos (IEEE) [19].

Todos los dispositivos que se pueden conectar de manera inalámbrica a una red son llamados estaciones. Estas están equipadas con interfaces controladoras de redes

inalámbricas. Las estaciones inalámbricas entran en dos categorías, puntos de acceso inalámbrico (AP) y clientes [20].

- **Punto de Acceso (AP):** Es un equipo de red que permite a dispositivos compatibles con Wi-Fi conectarse a una red alámbrica [21].
- **Cliente:** Suelen ser dispositivos portátiles como notebooks, smatphones, tablets o dispositivos no portátiles como computadores de escritorio o estaciones de trabajo que están equipados con una interfaz de red inalámbrica.

Las WLAN pueden operar en dos modos básicos; modo infraestructura y modo ad hoc. Sin embargo, la gran mayoría de redes WLAN se están usando en modo infraestructura.

3.4.2.1 Modo Infraestructura

El modo infraestructura es una red tipo cliente-servidor, que consiste en un dispositivo AP central y múltiples clientes conectados a él [22].

3.4.2.2 Modo Ad hoc

Ad hoc es un tipo de red descentralizada que consiste en solo clientes, donde cada uno se comunica directamente con los demás, sin necesidad de un punto de acceso AP.

Las configuraciones Ad hoc son comunicaciones punto a punto. Solo los dispositivos dentro de un rango definido pueden comunicarse entre sí para formar una red P2P [23].

La ventaja del modo Ad hoc es su flexibilidad y movilidad, a diferencia del modo infraestructura no está limitado por un punto físico dado que la ubicación no está limitada por el rango de alcance del AP. Las desventajas del modo Ad hoc son; un nivel de rendimiento en la tasa de transferencia mucho menor, y un consumo mayor de energía.

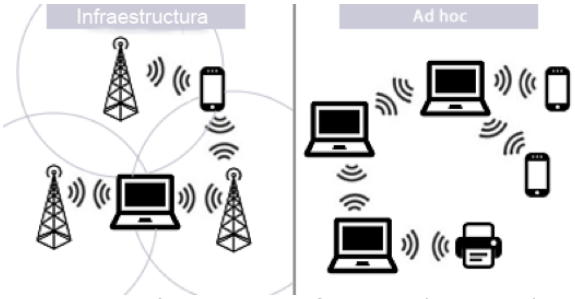

Figura 3: Infraestructura VS Ad Hoc (Wikipedia)

3.4.3 Bluetooth

Bluetooth es una tecnología inalámbrica de ondas de radio de corto alcance a una frecuencia de 2.4 GHz. Fue diseñada con el objetivo de simplificar las comunicaciones entre dispositivos informáticos, como computadores portátiles, smartphones, tablets y otros dispositivos, así como entre estos dispositivos e Internet.

La topología de Bluetooth es 1:N, donde un dispositivo que actúa como *Maestro* puede conectarse a uno o muchos dispositivos *Esclavos.* La red formada recibe el nombre de Piconet, donde el dispositivo *Maestro* es el responsable del control de la red.

Los equipos Bluetooth se clasifican por la clase y versión, existen clase 1, 2 y 3, donde su principal diferencia es el rango de alcance y versiones V1.0 a la V4.1 y su principal diferencia radica en una mejora en la tasa de datos que pueden transferir como la reducción en consumo de energía [24].

Versión Bluetooth	Transferencia máxima de datos
Bluetooth V1.0	768 Kbps
Bluetooth V1.2	1 Mbps
Bluetooth V2.0+EDR	3 Mbps
Bluetooth V3.0+HS	24 Mbps
Bluetooth V4.0	24 Mbps

Tabla 2: Versiones Bluetooth

3.4.4 Near Field Comunication

NFC, se trata de una tecnología inalámbrica de corto alcance (no más de un par de centímetros) que opera en la banda de los 13.56 Mhz, por lo tanto, no hace falta ninguna clase de licencia para su uso y deriva de las etiquetas RFID.

NFC es una plataforma abierta pensada desde su inicio para teléfonos y dispositivos móviles, dada su baja tasa de transferencia la que puede llegar a los 424 kbit/s, su enfoque más que para transmisión de grandes cantidades de datos es para comunicación instantánea, es decir identificación y validación de equipos/personas.

Normalmente esta tecnología es usada en conjunto con otras tecnologías de transmisión de datos como Bluetooth, donde NFC actúa como validador para establecer una conexión Bluetooth de manera más rápida y sencilla [25].

3.5 Wi-Fi Direct

Wi-Fi Direct es una marca de certificación dada por Wi-Fi Alliance a aquellos dispositivos que soportan la tecnología que habilita a dispositivos Wi-Fi conectarse directamente el uno con el otro pudiendo comunicarse para compartir, transmitir o sincronizar datos sin la necesidad de conectarse a un AP para poder comunicarse. Esta tecnología al estar certificada por Wi-Fi Alliance cuenta con los estándares para redes inalámbricas IEEE 802.11[26], por lo tanto, puede alcanzar velocidades de hasta 250 Mbps [27] y distancias de hasta 200m [28].

La conexión que se establece con Wi-Fi Direct es del tipo P2P donde se establece un grupo P2P donde un dispositivo actúa como Group owner P2P y el resto de los dispositivos actúan como Clientes P2P.

Su topología es, 1:N donde el Group Owner actúa similar a un AP, pudiendo conectarse a múltiples clientes. Esta topología es similar a la de redes WLAN, pero no permite la comunicación cliente-cliente a través del Group Owner, diferenciando esta tecnología del modo Ad hoc, aunque ambas sean soluciones de tipo P2P; dado su estructura son completamente diferentes.

Actualmente existen 11684 dispositivos electrónicos certificados por la Wi-Fi Alliance como productos con capacidad Wi-Fi direct [29].

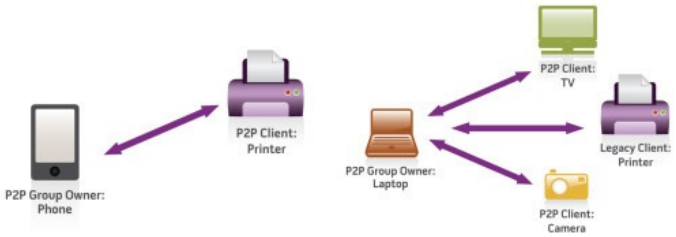

Figura 4: Wi-Fi Direct (omicrono)

3.5.1 API Wi-Fi Direct en Android

Wi-Fi Direct está disponible en Android desde Android 4.0 (API nivel 14) y versiones posteriores, permitiendo a dispositivos con el hardware adecuado conectarse entre ellos vía Wi-Fi sin el uso de un AP como intermediario. El framework Wi-Fi P2P de Android cumple con la certificación Wi-Fi Direct™ de Wi-Fi Alliance.

Usando esta API, se puede detectar y conectar a otros dispositivos que soporten Wi-Fi P2P, pudiendo comunicarse sobre una conexión rápida y a distancias mucho mayores que con una conexión Bluetooth.

La API Wi-Fi Direct consiste en las siguientes partes:

- Método que permite descubrir, pedir y conectar a peers que están definidos en la clase WifiP2pManager.
- Listeners que permiten notificar del éxito o fallo de los métodos de llamada de WifiP2pManager. Cuando se llama a los métodos WifiP2pManager, cada método puede recibir un listener específico pasado como parámetro.
- Intents son aquellos que notifican cuando eventos específicos son detectados por el framework Wi-Fi P2P, tales como, la conexión de un nuevo peer, inicio de búsqueda de peers o detectar actividad de nuevos peers disponibles [30].

3.5.2 Formación del Grupo

Existen muchas maneras en las que dos dispositivos pueden establecer un grupo P2P. Algunas técnicas de formación de grupo que son; Standard, Autónoma y Persistente.

El procedimiento de formación de grupo involucra dos fases:

- **Determinación del Group Owner P2P:** donde los dos dispositivos negocian el Group Owner basado en la necesidad/capacidad de ser el Group Owner P2P.
- **Proporcionando el grupo P2P:** donde se establece la sesión del grupo P2P usando las credenciales apropiadas y utilizando una configuración simple de Wi-Fi para intercambiar dichas credenciales.

3.5.2.1 Formación Estándar

En el caso de la formación estándar toma lugar cuando los dispositivos P2P establecen el primer descubrimiento el uno del otro y negocian quien tomara el rol de Group Owner. Este descubrimiento comienza con un escaneo activo tradicional IEEE 802.11, luego se ejecuta un nuevo algoritmo de búsqueda que alterna entre los estados de búsqueda y escucha. Una vez que los dispositivos P2P se han encontrado comienza la negociación del Group owner, implementada por un *handshake* de tres vías; Petición-Respuesta- Confirmación.

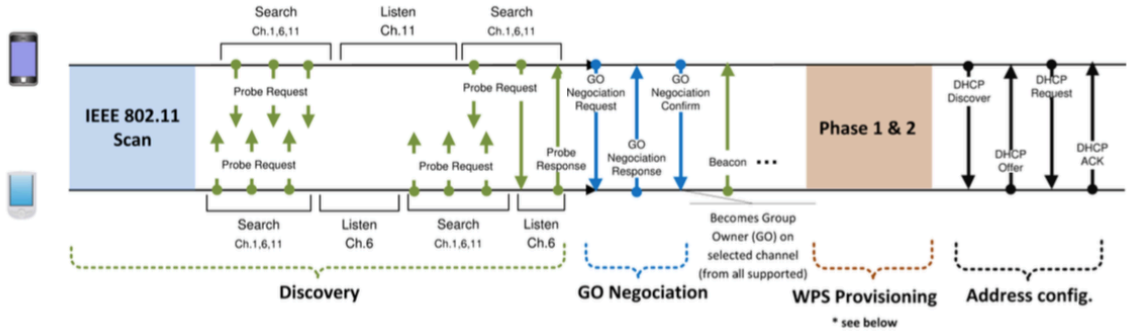

*Figura 5: Formación estándar (*communications with WiFi Direct: overview and experimentation*")*

3.5.2.2 Formación Persistente

Durante el proceso de formación del grupo P2P los dispositivos pueden declarar el grupo como *persistente* mediante el uso de una bandera. En este caso los dispositivos guardan las credenciales de la red y los roles tomados por cada uno, así uno de los dos dispositivos puede enviar la invitación a retomar la conexión con sus roles originales, siendo este método más rápido estableciendo la conexión ya que no hay una negociación de Group Owner y la fase de aprovisionamiento WPS es más ágil dado la reutilización de las credenciales.

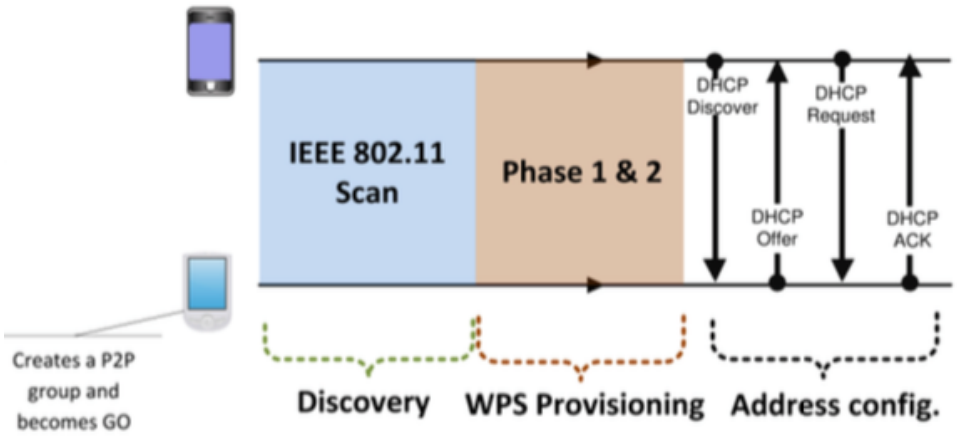

*Figura 6: Formación Persistente (*communications with Wi-Fi Direct: overview and experimentation*")*

3.5.2.3 Formación Autónoma

El dispositivo P2P puede de manera independiente decidir crear el grupo P2P, convirtiéndose inmediatamente en el Group Owner, estableciéndose en un canal y comenzando a transmitir. Otros dispositivos pueden descubrir el grupo con los mecanismos tradicionales de escaneo y solicitar unirse como clientes [31].

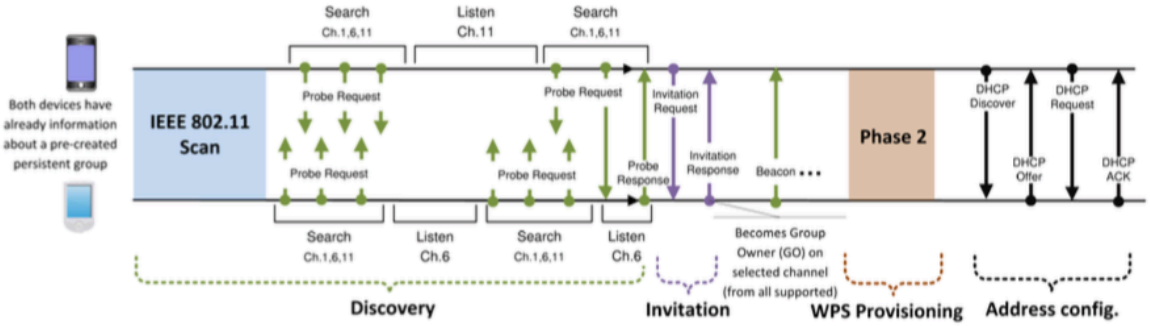

*Figura 7: Formación Autónoma (*communications with Wi-Fi Direct: overview and experimentation*")*

3.6 Redes WANET

Wireless ad hoc network; Red ad hoc inalámbrica en español. Es un tipo de red inalámbrica descentralizada. Esta red está formada por una colección de dos o más dispositivos equipados con capacidad de red inalámbrica llamados nodos. Estos dispositivos son capaces de comunicarse con otros nodos que están inmediatamente dentro de su rango de alcance o con uno que esta fuera de su rango de alcance, para lo anterior los nodos deben desplegar un nodo intermedio para ser el enrutador encargado de dirigir el paquete desde la fuente hasta el destino. Las redes Ad-hoc no tienen puerta de enlace ya que cada nodo puede actuar como puerta de enlace [32].

3.6.1 Protocolos de ruteo en redes Ad-hoc

Existen dos principales tipos de protocolos de ruteo; protocolo de enrutamiento de vectores de distancia y protocolo de ruteo bajo demanda

3.6.1.1 Protocolo de enrutamiento de vectores de distancia

El enrutamiento de vectores de distancia se basa en el cálculo de la dirección y la distancia a cualquier enlace en la red, donde dirección normalmente significa la siguiente dirección de salto y la interfaz de salida, donde distancia es una medida del costo para llegar a un determinado nodo. La ruta de menor costo entre dos nodos es la ruta con distancia mínima. Cada nodo mantiene un vector o tabla de distancia mínima a cada nodo. El costo de llegar a un destino se calcula utilizando varias métricas de ruta.

- **RIP:** Utiliza el recuento de saltos del destino.
- **IGRP:** Toma en cuenta otra información como el retardo del nodo y el ancho de banda disponible.

3.6.1.2 Protocolo de ruteo bajo demanda

Este tipo de protocolo encuentra la ruta bajo demanda inundando la red con paquetes de solicitud de ruta. Las principales desventajas de este protocolo son:

- Alto tiempo de latencia en la localización de la ruta.
- La inundación excesiva puede conducir a una obstrucción de la red [31]

3.6.2 Redes MANET

Movile Ad-hoc network; Red Ad-Hoc Móvil en español. Es un tipo de red WANET donde su principal característica es que sus nodos cuentan con la capacidad de cambiar su ubicación constantemente, la mayoría o todos estos nodos están ligados a una batería por lo tanto cuentan con una autonomía limitada, siendo la optimización de la energía un criterio fundamental para este tipo de redes.

El protocolo de ruteo más utilizado para estas redes dado su constante movimiento es el “Protocolo de enrutamiento por ubicación”.

3.6.2.1 Protocolo de enrutamiento por ubicación

Este protocolo utiliza información exacta de la ubicación de los nodos. La información es obtenida vía GPS, basado en la ubicación exacta un mejor camino hacia el nodo de destino puede ser determinado, este protocolo obtiene la ubicación de los nodos inundando la red con paquetes de solicitudes de ubicación y crea una tabla de ubicaciones.

3.7 Grafos

Existen situaciones que pueden ser representadas en forma binaria por medio de un diagrama de puntos y líneas; los puntos son llamados vértices o nodos y las líneas llamadas aristas, y el diagrama completo es llamado “grafo” [33].

Para que se entienda mejor, el grafo permite estudiar las interrelaciones entre unidades que interactúan unas con otras. Por ejemplo, una red de computadoras puede representarse y ser objeto de estudio mediante un grafo.

3.7.1 Algoritmos de búsqueda en grafos

Un grafo, representa un conjunto de nodos unidos en una red. Si dos nodos están unidos, al viajar de uno a otro se considerará sucesor el nodo al que nos movemos, y predecesor el nodo del que venimos, además, normalmente existirá un coste vinculado al desplazamiento entre nodos. Un algoritmo de búsqueda tratará de encontrar un camino optimo entre dos nodos como, por ejemplo, un camino que minimice el coste de desplazamiento, o el número de pasos a realizar.

Cuando se realiza la búsqueda en un espacio de estados se pueden hacer dos cosas;

- Buscar el estado final sin evaluar los estados sucesores.
- Buscar el estado final calculando mediante evaluación cuál es el mejor de los estados sucesores.

Cuando no se evalúan los estados se habla de “búsqueda no informada”, en caso contrario hablamos de “búsqueda informada” o “búsqueda heurística” [34].

3.7.1.1 Búsquedas no informadas

A continuación, se detallarán tipos de búsqueda en grafos no informadas, donde no se emplea ningún tipo de técnica especial, este tipo de búsqueda sólo expande el grafo con objetivo de encontrar el nodo meta, sin considerar factores como el costo de la solución o la cantidad de pasos necesarios para esta.

3.7.1.1.1 Búsqueda primero en anchura

Es una estrategia sencilla en la que se expande primero el nodo raíz, a continuación, se expanden todos los sucesores del nodo raíz, después sus sucesores, etc. En general, se expanden todos los nodos a una profundidad en el árbol de búsqueda antes de expandir cualquier nodo del próximo nivel.

Se puede implementar en la búsqueda de árboles con:

- Una frontera vacía
- Que sea una cola FIFO

Lo que significa que los nodos más superficiales se expanden antes que los nodos más profundos [34].

3.7.1.1.2 Búsqueda primero en profundidad

La búsqueda primero en profundidad siempre expande el nodo más profundo en la frontera actual del árbol de búsqueda. La búsqueda procede inmediatamente al nivel más profundo del árbol de búsqueda, donde los nodos no tienen ningún sucesor. Cuando esos nodos se expanden, son quitados de la frontera, así entonces la búsqueda retrocede al siguiente nodo más superficial que todavía tenga sucesores inexplorados.

Esta estrategia puede implementarse por la búsqueda de árboles con una cola último en entrar primero en salir (LIFO), también conocida como una pila [34].

3.7.1.2 Búsquedas informadas

En este tipo de búsqueda se emplea un mecanismo que permite evaluar los próximos estados el cual se denomina heurística o función de evaluación. En general, su uso no garantiza encontrar una solución óptima, pero sí una solución de buena calidad en

un tiempo razonable, se recomienda su utilización ya que en algunos casos los algoritmos no informados pueden alcanzar costos demasiado altos en tiempo y espacio.

En este tipo de búsqueda es muy importante diseñar adecuadamente la heurística, esta se debe diseñar con conocimientos específicos del problema, por lo tanto, es necesario que se conozca el problema con precisión.

3.7.1.2.1 Búsqueda Primero mejor avara

Esta búsqueda trata de expandir el nodo más cercano al objetivo, alegando que probablemente conduzca rápidamente a una solución. Así evalúa los nodos utilizando solamente la función heurística:

$$f(n) = h(n)$$

Donde $h(n)$ es el coste estimado del camino menos costoso desde el nodo " n " a un nodo objetivo.

La búsqueda primero mejor avara, no es completa ya que puede quedar atajada en un ciclo infinito y mantiene todos los nodos en memoria, por lo tanto, no es óptima.

3.7.1.2.2 Búsqueda Haz

Este algoritmo de búsqueda fue desarrollado para mejorar la búsqueda en anchura ya que esta garantiza encontrar la solución con mejor profundidad, pero resulta inviable usarla cuando el espacio de estados es muy grande debido a su consumo de memoria.

La búsqueda de Haz utiliza una función heurística y emplea una variable B, que especifica el número de estados que podrán almacenarse en cada nivel. Por lo tanto, el algoritmo almacena los B mejores estados en función de dicha heurística limitando el uso de memoria sólo a los nodos sucesores más prometedores para una posible solución.

3.7.1.2.3 Búsqueda A-estrella

El algoritmo A-estrella o A* permite encontrar soluciones óptimas para un problema. Es la forma de búsqueda primero el mejor más conocida y sirve para la búsqueda de caminos.

El algoritmo A*, no desarrolla un camino por interacción, si no que desarrolla varios caminos y elige los más prometedores. La idea de este algoritmo es evitar expandir caminos que ya son costosos.

Evalúa los nodos combinando $g(n)$, el coste para alcanzar el nodo y $h(n)$, el costo de ir al nodo objetivo.

$$f(n) = g(n) + h(n)$$

Ya que:

- $g(n)$ costo del camino desde el nodo inicio o estadio inicial al nodo *n* o estado actual, y $h(n)$ costo estimado del camino más barato o costo mínimo desde el nodo actual (*n*) al nodo objetivo.

Se tiene:

- $f(n)$ costo estimado de la mejor solución a través del estado actual (n).

En este caso, A* es óptima si $h(n)$ es una heurística admisible, es decir, con tal de que $h(n)$ nunca sobrestime el costo de alcanzar el objetivo. Las heurísticas admisibles son por naturaleza optimistas, porque piensan que el costo de resolver el problema es menor que el que es en realidad.

Ya que $g(n)$es el costo exacto para alcanzar n, tenemos como consecuencia inmediata que la $f\ (n)$ nunca sobrestima el costo verdadero de una solución a través de n.

Un ejemplo de una heurística admisible es la distancia en línea recta (h_{DLR}) que usamos para ir de un punto a otro. La distancia en línea recta es admisible porque el camino más corto entre dos puntos cualquiera es una línea recta, entonces la línea recta no puede ser una sobrestimación [34].

3.8 Topologías en redes

La topología de red se define como un mapa físico o lógico de una red para intercambiar los datos, quiere decir que es la forma en que esta está diseñada. Su concepto se puede definir como un conjunto de nodos interconectados los que actúan como emisores y receptores [35].

3.8.1 Topología en bus

Es un tipo de red en el cual cada nodo o dispositivo está conectado a un único canal de comunicaciones, el que se denomina bus, de esta manera todos los nodos comparten un mismo canal para comunicarse entre sí

3.8.2 Topología de anillo

Es un tipo de red en el cual cada nodo o dispositivo es conectado al otro nodo y el último es conectado al primero, formando una especie de anillo; para cada dispositivo existen exactamente dos vecinos.

3.8.3 Topología de estrella

Es un tipo de red en el cual cada nodo o dispositivo está conectado a un único nodo central, el cual está encargado de reenviar todas las transmisiones recibidas desde cualquier nodo periférico. Todas las comunicaciones entre nodos deben ser manejadas por este nodo central; no es posible que los nodos periféricos se comuniquen entre si directamente saltándose el uso del nodo central.

3.8.4 Topología en malla

Es un tipo de red de conexión punto a punto entre los nodos, donde todos están comunicados entre sí. Una red en malla tiene n(n-2)/2 canales físicos para conectar n dispositivos.

3.8.5 Topología tipo árbol

Es un tipo de red en que los nodos están colocados en forma de árbol. Se puede observar como una serie de redes estrella interconectadas, salvo que esta no posee un nodo central que se encargue de las comunicaciones.

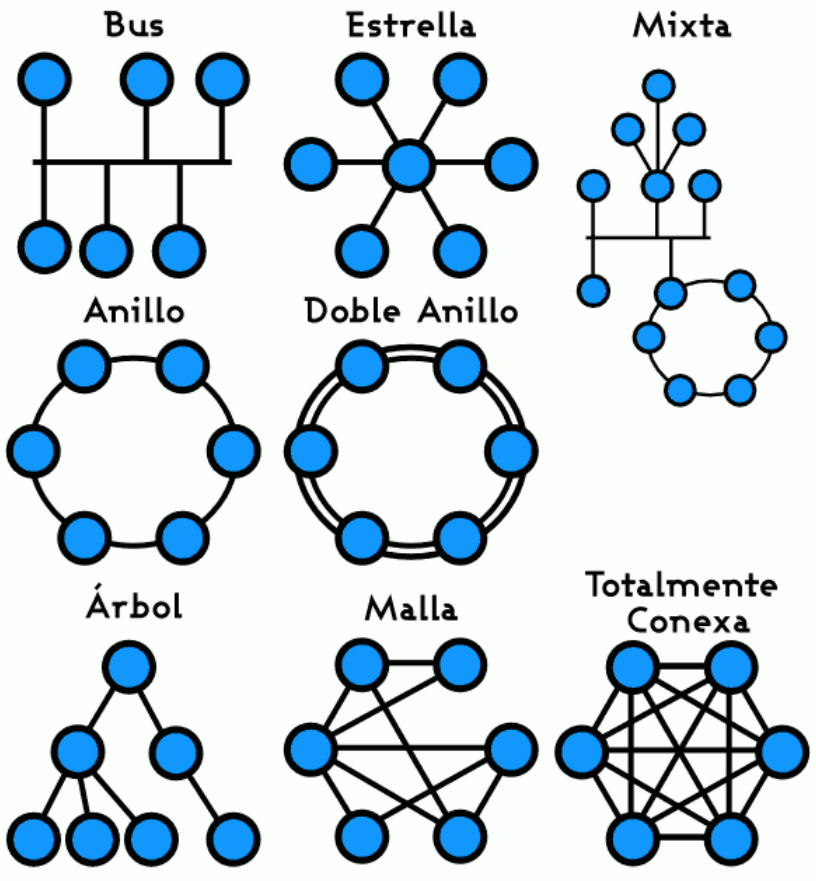

Figura 8: Topologías de redes (Wikipedia)

4 Especificación de requerimientos del software

4.1 Alcances

El proyecto a desarrollar tiene como objetivo crear una aplicación que permita enviar un mensaje a través de una red independiente. Aprovechando las capacidades P2P que permite la tecnología Wi-fi-Direct en dispositivos Android, además, proponer he implementar un protocolo de ruteo válido para que el mensaje sea entregado en su destino.

La aplicación no permite:

- La comunicación instantánea entre todos los dispositivos.
- El envío de más de un mensaje al mismo tiempo desde un mismo dispositivo.
- Que el dispositivo actúe como emisor y receptor de un mensaje a la vez, es decir no puede estar emitiendo un mensaje al mismo tiempo que recibe un mensaje.
- La comunicación entre usuarios específicos; la aplicación está diseñada para transportar y difundir el mensaje entre zonas geográficas, independientemente de que usuarios se encuentren en dicha zona.

4.2 Objetivos del software

4.2.1 Objetivo general

El correcto envío y difusión del mensaje entre dos zonas geográficas distintas creando una red independiente utilizando dispositivos Android mediante Wi-Fi P2P, aplicando el protocolo de ruteo propuesto.

4.2.2 Objetivos específicos

- La aplicación detecta dispositivos cercanos que cuenten con la tecnología Wi-Fi Direct.
- La aplicación permite establecer conexión entre el Group Owner y los clientes

- La aplicación hace uso de coordenadas geográficas de los edificios del área de estudio.
- La aplicación se apoya del uso de GPS para obtener la ubicación exacta de cada dispositivo.
- La aplicación utiliza correctamente el protocolo de ruteo propuesto.
- La aplicación permite al usuario enviar un mensaje introducido por teclado.
- La aplicación permite al usuario seleccionar el edificio donde quiere que su mensaje sea leído.
- La aplicación consigue un método de comunicación independiente, sin requerir de una infraestructura física previamente instalada.
- La aplicación despliega el mensaje cuando es recibido en la zona seleccionada como zona de difusión.

4.3 Requisitos mínimos del software

Para el correcto funcionamiento de la aplicación, se deben tener en cuenta ciertos requisitos mínimos que se detallarán a continuación.

4.3.1 Requisitos mínimos del software

Debido a que la aplicación se encuentra codificada en el lenguaje de programación Java y utiliza distintos framework y servicios específicos de la plataforma Android, funciona sobre la siguiente versión del SO Android y posteriores.

Nombre: Android
Desarrollador: Google
Versión: Android Kit-Kat 4.4 (API nivel 19) y posteriores.

4.3.2 Requisitos de software

Dado que SO Android provee todo lo necesario para la correcta ejecución de la aplicación, no se requiere de algún software en específico, para el funcionamiento la aplicación.

4.3.3 Requisitos de hardware

Es necesario un dispositivo móvil compatible con la versión Android descrita en el punto 4.3.2, además, el dispositivo debe estar certificado con la tecnología Wi-Fi direct y contar con GPS.

4.4 Descripción Global del Producto

4.4.1 Interfaz de hardware

La aplicación utiliza la conexión de sockets para la transferencia de archivos, haciendo uso del puerto 6464.

4.4.2 Interfaz Software

La aplicación hace uso de los servicios de google “Google Api Services” pre instalada en todos los dispositivos Android para recibir la ubicación a través del GPS por lo tanto se requiere que esta aplicación se encuentre actualizada desde la tienda de aplicaciones de google “Playstore”.

4.4.3 Interfaz de comunicación

En la interfaz de comunicación se menciona el protocolo con el cual la aplicación emplea la comunicación entre cliente-servidor.

Nombre: Transmisssion Control Protocol
Abreviación TCP

Este protocolo es uno de los protocolos fundamentales de la internet, es un protocolo de la capa de transporte y se usa para el envío de los segmentos de una manera confiable y libre de pérdidas.

4.5 Requerimientos específicos

4.5.1 Requerimientos funcionales del sistema

ID	Nombre	Descripción
1	Ingreso del mensaje	La aplicación debe permitir al usuario escribir un mensaje de manera libre, por medio de la entrada de texto estándar de Android.
2	Selección del lugar de destino	La aplicación debe permitir seleccionar el edificio del área de estudio donde quiere que se despliegue su mensaje.
3	Visualizar mensaje	La aplicación debe permitir al usuario visualizar el mensaje que está transportando en su dispositivo.
4	Desplegar mensaje	La aplicación debe desplegar automáticamente el mensaje si se encuentra en el área de difusión.
5	Búsqueda automática	La aplicación debe buscar automáticamente conexiones disponibles.
6	Conexión automática	La aplicación debe realizar automáticamente la petición de conexión a dispositivos disponibles.
7	Difusión automática	La aplicación al detectar por medio de GPS que se encuentra en un rango de 15 metros del lugar de destino, debe comenzar la difusión automática entre los clientes disponibles.
8	Selección automática	La aplicación debe seleccionar automáticamente el dispositivo disponible más óptimo para el transporte del mensaje.
9	Confirmación de envío de mensaje	La aplicación debe mostrar un mensaje de éxito cuando el mensaje sea transferido a otro dispositivo.

Tabla 3: Requerimientos funcionales

5 Análisis

5.1 Proceso de negocios Futuros

En esta sección se presenta el diagrama de Modelado de Procesos de negocio, o BPMN por sus siglas en inglés. Este diagrama se utiliza para representar los procesos que ocurren durante la ejecución de la aplicación y su interacción con las entidades u actores, que en este caso son el Group Owner y el cliente. El formato del diagrama BPMN se basa en el flujo de datos para una aportar una mayor comprensión al lector.

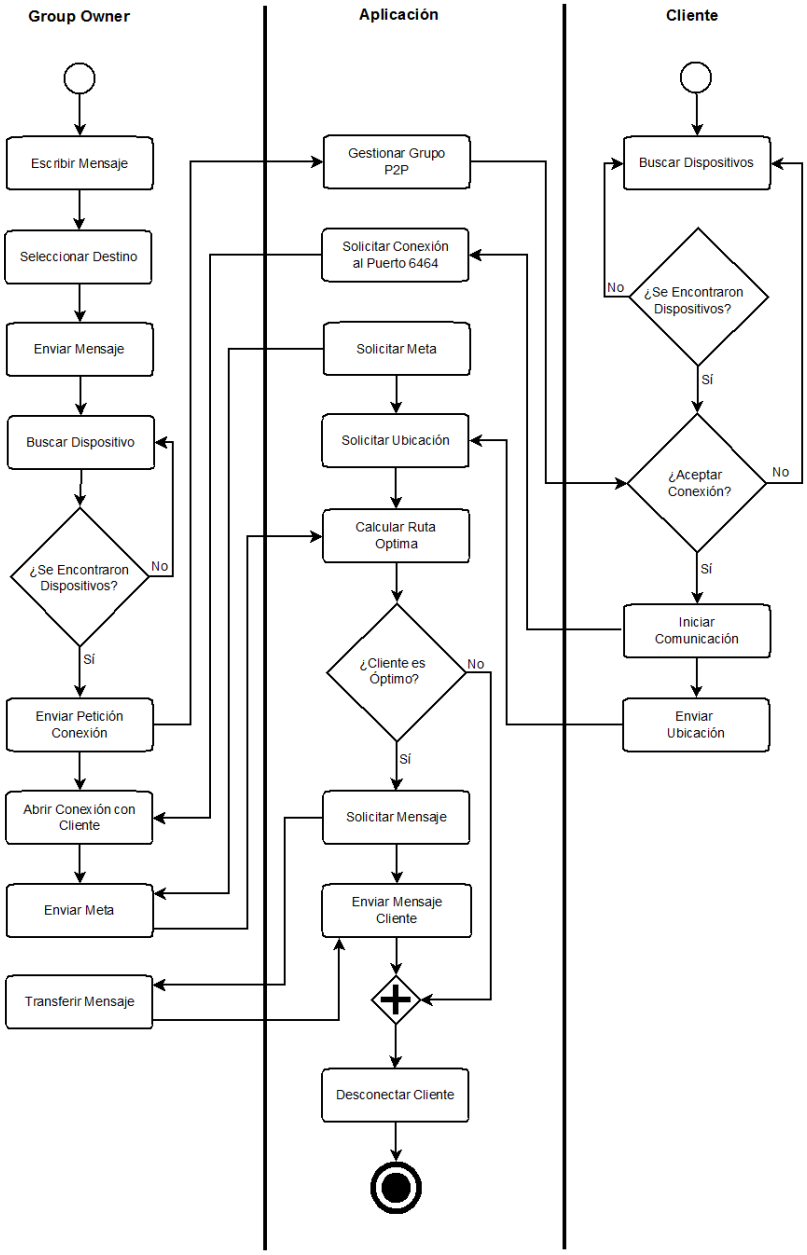

Figura 9: BPMN

5.2 Diagrama de flujo de datos

En esta sección se presentan los Diagramas de flujo de datos, estos muestran el flujo de la información que entra y sale del sistema, la interacción del sistema con las entidades externas y el flujo de datos interno.

Para presentar ampliamente el diagrama de flujos de datos, se presentara en 3 niveles; desde el más amplio al más detallado de cada proceso, los diagramas que se presentan son; Diagrama de Flujo de Datos de Contexto o nivel 0, este es el diagrama más general presentando una visión a grandes rasgos del sistema y su interacción con las entidades externas, el siguiente es el nivel 1 o Diagrama de Flujo de datos Superior, donde se detallan los procesos internos del sistema que describe el proceso principal y por ultimo tenemos el diagrama de nivel 2 o Diagrama de Flujo de datos de Detalle, en este diagrama se especifica cada uno de los procesos detallados en el Diagrama de Flujo de Datos Superior.

5.2.1 Diagrama de Flujo de Datos de Contexto: Nivel 0

En este primer nivel se presenta sólo un proceso, el cual es identificado con un número 0, llamado “Gestionar mensaje”. Se muestra el flujo de datos entre las entidades “Cliente” y “ Group Owner” con el sistema, omitiendo detalles de sus procesos internos de este.

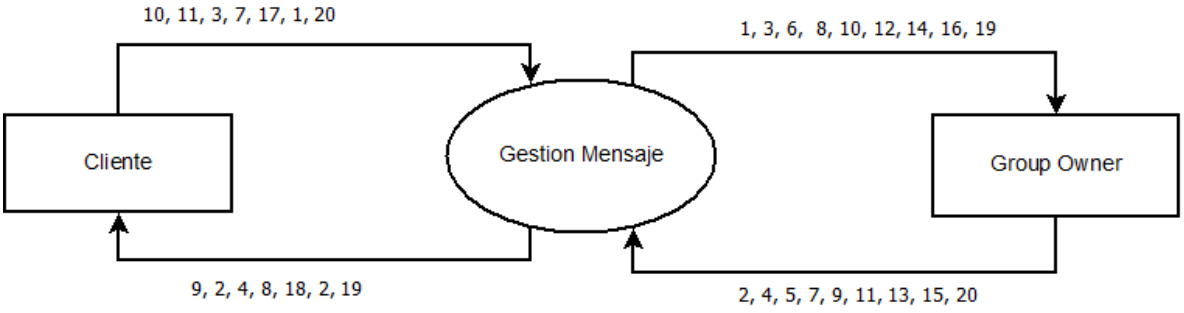

Figura 10: DFD contexto

Donde:

1. Solicitud mensaje.
2. Respuesta solicitud mensaje.

3. Solicitud meta.
4. Respuesta solicitud meta.
5. Solicitud lista clientes.
6. Respuesta solicitud lista clientes.
7. Solicitud mi peso ruta mínima.
8. Respuesta mi peso ruta mínima.
9. Solicitud conexión cliente (Wi-Fi P2P).
10. Respuesta conexión cliente.
11. Solicitud inicio servidor puerto 6464.
12. Respuesta solicitud inicio servidor.
13. Solicitud comparación peso cliente.
14. Respuesta comparación.
15. Sol envío mensaje.
16. Respuesta solicitud envío mensaje.
17. Solicitud conexión servidor puerto 6464.
18. Respuesta solicitud conexión servidor.
19. Solicitud ubicación.
20. Respuesta solicitud ubicación.

5.2.2 Diagrama de Flujo de datos Superior: Nivel 1

- Gestionar grupo P2P: Se encarga de escanear y detectar dispositivos disponibles con la aplicación, generar la lista de dispositivos para el Group Owner y gestionar las solicitudes de conexión.
- Gestionar envío de mensaje: Se encarga evaluar cada cliente que se conecta y determinar si es un posible portador mejor que el actual, de ser así, se le traspasa el mensaje al nuevo portador y se libera el portador actual.
- Gestionar mi peso: Se encarga de solicitar la meta del mensaje y calcular el costo de transporte del dispositivo evaluado según las distancias de coordenadas geográficas, trazando el camino más corto posible que podría recorrer el mensaje de seleccionar este dispositivo como portador y su costo asociado.

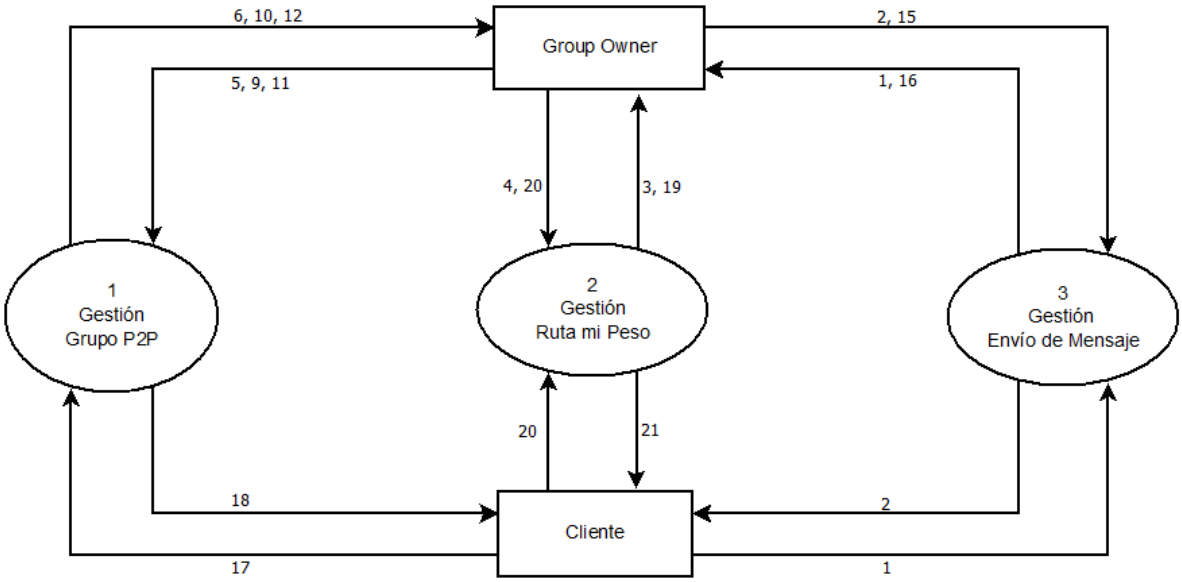

Figura 11: DFD Superior

Donde:

1. Solicitud mensaje.
2. Respuesta solicitud mensaje.
3. Solicitud meta.
4. Respuesta solicitud meta.
5. Solicitud lista clientes.
6. Respuesta solicitud lista clientes.
7. Solicitud mi peso ruta mínima.
8. Respuesta mi peso ruta mínima.
9. Solicitud conexión cliente (Wi-Fi P2P).
10. Respuesta conexión cliente.
11. Solicitud inicio servidor puerto 6464.
12. Respuesta solicitud inicio servidor.
13. Solicitud comparación peso cliente.
14. Respuesta comparación.
15. Sol envío mensaje.
16. Respuesta solicitud envío mensaje.
17. Solicitud conexión servidor puerto 6464.
18. Respuesta solicitud conexión servidor.
19. Solicitud ubicación.
20. Respuesta solicitud ubicación.

5.2.3 Diagrama de Flujo de datos de Detalle: Nivel 2

En el Diagrama de detalle, cada proceso descrito en el Diagrama de nivel Superior se divide en subprocesos con el fin de detallar aún más los procesos del sistema.

Proceso 1

- Gestionar Cliente P2P: Este proceso permite establecer la conexión con el cliente P2P dado que se ha solicitado la conexión con este, otorgándole su rol como cliente en la conexión al Group Owner.
- Gestionar Group Owner: Dado que se ha gestionado la conexión con el cliente se deriva en la creación de un Grupo P2P donde el dispositivo que solicita la conexión actúa como Group Owner.

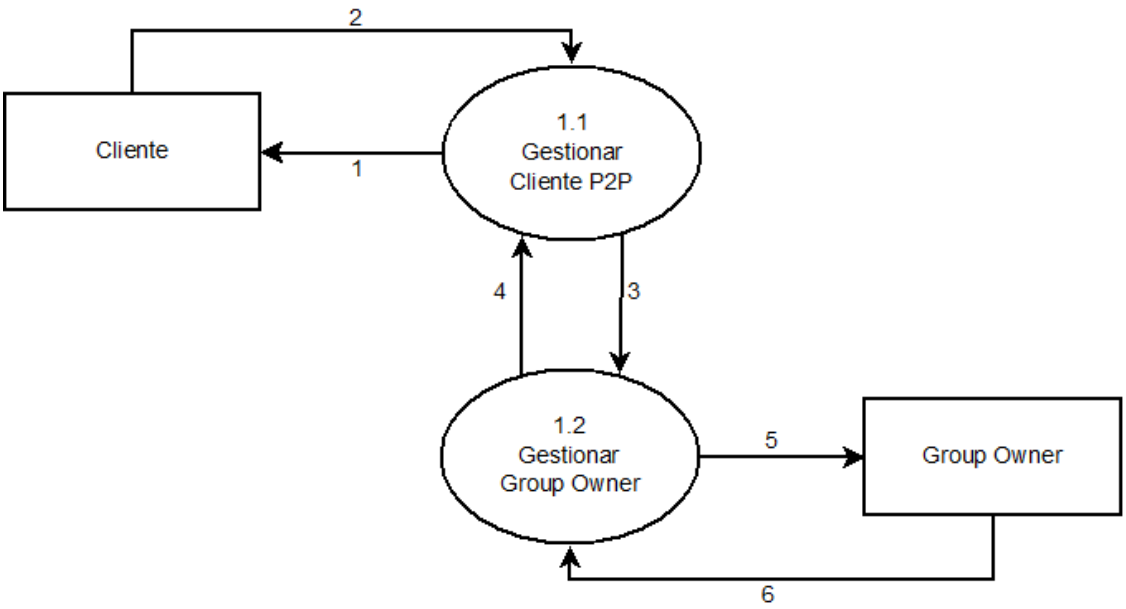

Figura 12: DFD Detalle 01

Donde:

1. Solicitar conexión Wi-Fi P2P a Cliente.
2. Respuesta Solicitud Conexión Wi-Fi P2P desde cliente.
3. Group Owner intent.
4. Confirmación entrada a grupo P2P.
5. Solicitud conexión a grupo Wi-Fi P2P desde cliente.
6. Respuesta solicitud conexión a grupo Wi-Fi P2P de cliente.

Proceso 2

- Calcular el edificio más cercano: Este proceso solicita al almacén de datos "Ubicación Dispositivo" las ultimas coordenadas del dispositivo almacenadas, al almacén "Preferencias Edificios" los edificios seleccionados por el usuario como los más visitados y al almacén "Coordenadas Edificios GPS" las coordenadas de cada uno de estos edificios, para luego calcular a cuál de estos se encuentra a una distancia menor.
- Calcular peso ruta más optima: Este proceso recibe el edificio más cercano a las preferencias del usuario y el edificio meta del mensaje, consultadas al almacén "Coordenadas edificios GPS" y al almacén "Edificios cercanos". Por medio del algoritmo A* traza la ruta más óptima tomando como punto de partida el edificio más cercano actual al cliente obteniendo el peso o costo de esta ruta tanto para el Group Owner como para el cliente.

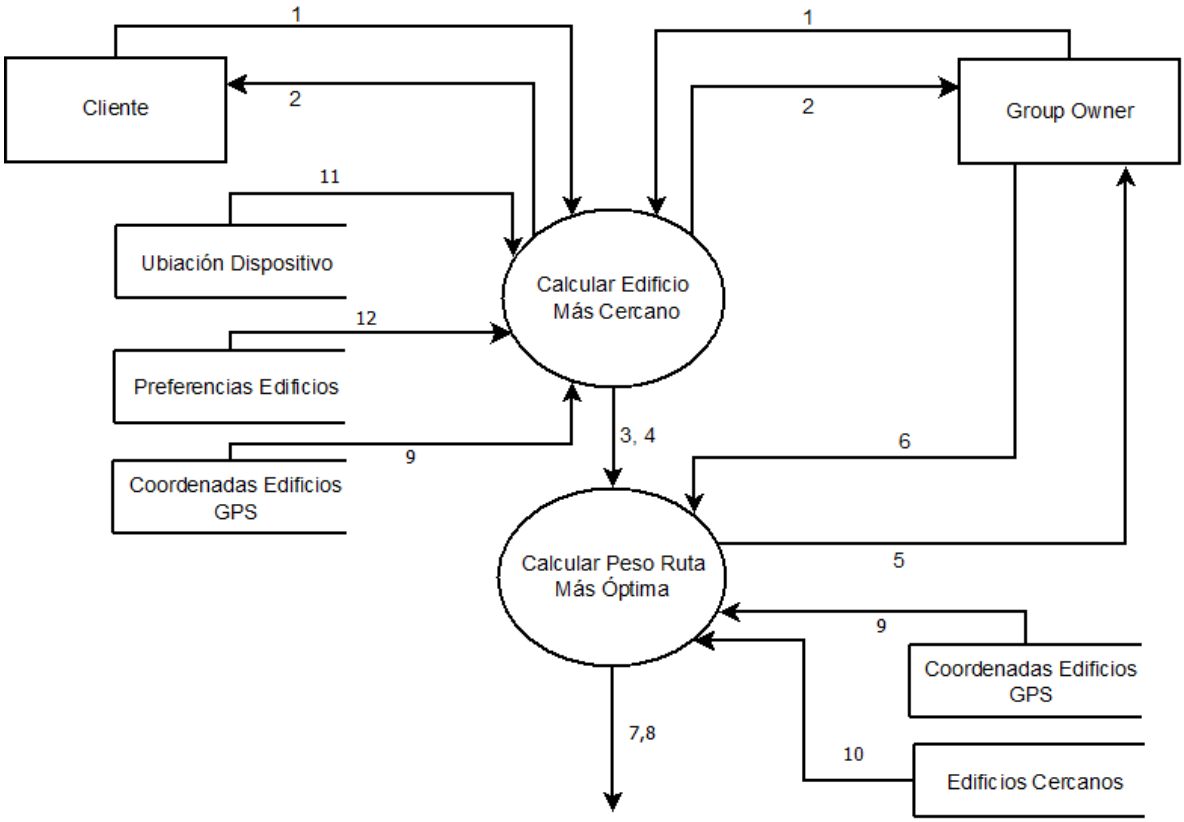

Figura 13: DFD Detalle 02

Donde:

1. Solicitud edificio más cercano.
2. Respuesta solicitud edificio más cercano.
3. Edificio más cercano cliente.
4. Edificio más cercano Group Owner.
5. Solicitud meta mensaje.
6. Respuesta solicitud meta mensaje.
7. Peso ruta cliente.
8. Peso ruta Group Owner.
9. Coordenadas edificio.
10. Vecinos
11. Coordenadas Dispositivo.
12. Preferencias Edificio.

Proceso 3

- Compara pesos ruta: Este proceso obtiene los pesos o costos de la ruta calculados para el Cliente y el Group Owner, los compara y obtiene el portador óptimo para el mensaje.
- Gestión portador mensaje: Este proceso obtiene el portador óptimo para el mensaje y responde tanto al Group Owner como al Cliente sus solicitudes para portar el mensaje y envía la orden de mantener o transferir el mensaje.
- Gestión transferencia mensaje: Este proceso gestiona los estados tanto del Group Owner como del Cliente como portadores del mensaje transfiriendo el mensaje al Cliente, cambiando su estado a portador transformándolo en Group Owner o manteniendo el Group Owner actual y descartando al Cliente.

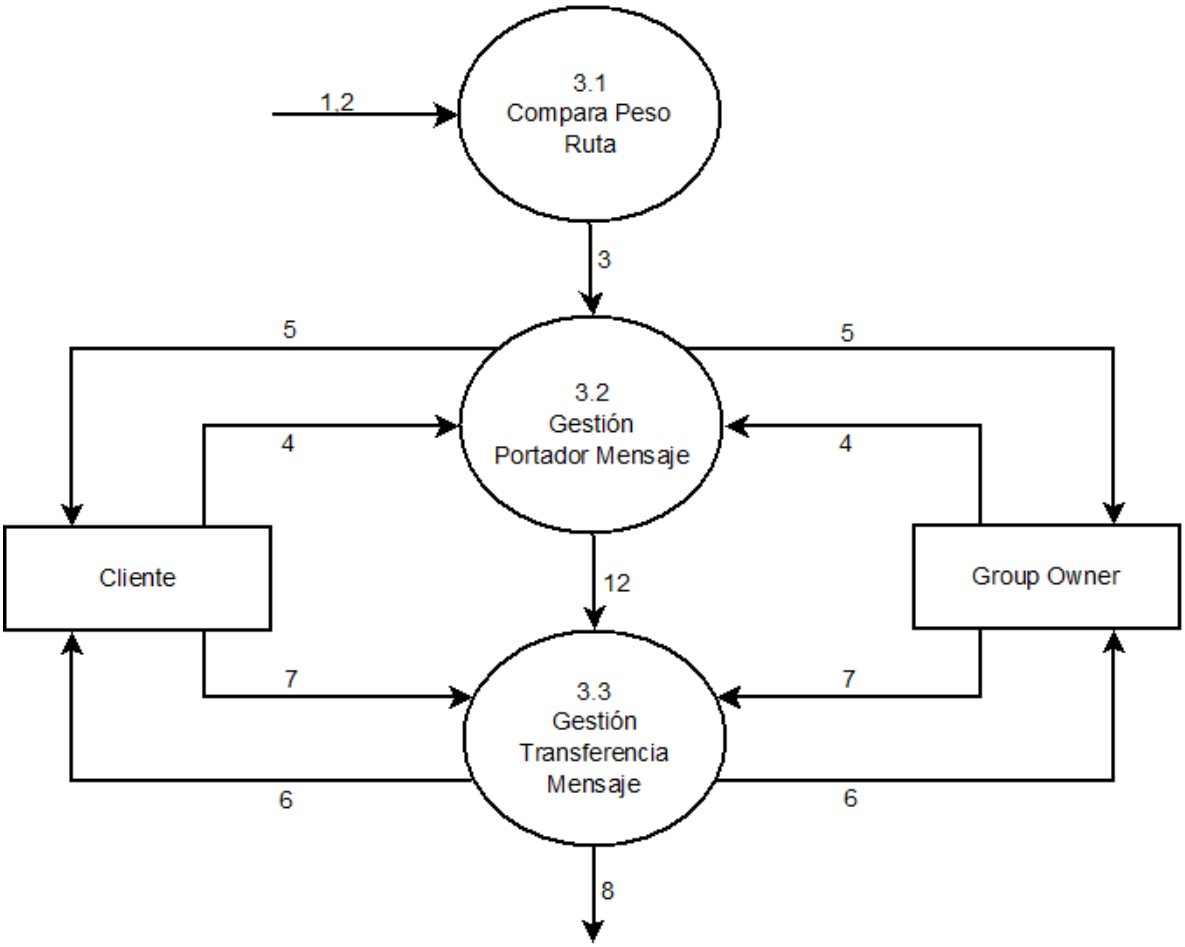

Figura 14: DFD Detalle 03

Donde:

1. Peso ruta cliente.
2. Peso ruta Group Owner.
3. Portador óptimo.
4. Solicitud portador.
5. Respuesta portador
6. Solicitud estado portador.
7. Respuesta estado portador.
8. Mensaje a portador.
9. Solicitud peso Group Owner.
10. Respuesta peso Group Owner.
11. Respuesta difusión.
12. Solicitud transferencia mensaje.

6 UBBMessenger

En este punto se especifican los aspectos de la aplicación “UBBMessenger”, la cual fue desarrollada para estudiar las capacidades de la API de Wi-Fi Direct disponible en el S.O Android para crear redes libres de infraestructura física y poder transportar datos de un lugar a otro.

6.1 Tipos de usuarios

Usuario	Características
General	Dispositivo que no ha asumido ningún rol aún, se encuentra en etapa de escaneo y descubrimiento de otros dispositivos Wi-FI Direct disponibles.
Group Owner	Dispositivo que actúa como emisor del mensaje, solicitando conexiones a los dispositivos cercanos disponibles y solicitando información para evaluarlo como potencial portador del mensaje, mientras asume este rol el dispositivo no acepta conexiones de otro Group Owner.
Cliente	Dispositivo que recibe la solicitud de conexión del Group Owner y es evaluado por el mismo como potencial portador del mensaje. No puede conectarse a otro usuario Cliente. No puede recibir la conexión de más de un Group Owner a la vez.

Tabla 4: Tipos de usuarios

6.2 Estados de la Aplicación

La aplicación UBBMessenger se pueden identificar los siguientes estados y es capaz de pasar de un estado a otro de manera autónoma.

Estado	Características
Descubriendo Peers	Este estado comienza junto a la aplicación, la cual comienza a escuchar a través del servicio “broadcast receiver” todos los dispositivos disponibles para la conexión a través de Wi-Fi Direct.

	Este estado sólo se detiene cuando existe una solicitud de conexión en progreso.
Conectar a Peer	Este estado comienza cuando un dispositivo asume el rol de Group Owner queriendo transmitir un mensaje, comenzando intentos de conexión con los dispositivos descubiertos en el estado anterior permitiendo el intercambio de datos entre ambos dispositivos.
Evaluando Peer	Este estado comienza una vez la conexión es establecida con éxito y ambos dispositivos intercambian información la cual servirá para establecer el siguiente portador del mensaje
Transfiriendo mensaje	Este estado comienza una vez que la aplicación detecta un mejor portador para el mensaje, transfiriendo la tarea de dispositivo y liberando al dispositivo anterior.
Difundiendo mensaje	Este estado comienza cuando el dispositivo que actúa como Group Owner detecta que se encuentra en la meta seleccionada para el mensaje y envía el mensaje a cada uno de los dispositivos disponibles detectados en el estado “Descubriendo Peers”.

Tabla 5: Estados de la aplicación

6.3 Conexión de dispositivos

La aplicación permite conectar a varios dispositivos a través de la tecnología Wi-Fi Direct formando una red P2P, permitiendo el traspaso de datos de un dispositivo a otro, como está explicado en el punto *3.5 Wi-Fi Direct*.

Dado el enfoque que toma la aplicación de formar una red de difusión sin infraestructura física y el estudio de las capacidades de la API Wi-Fi Direct en Android, la tarea de búsqueda y conexión de dispositivos cercanos es completamente autónoma por parte de la aplicación, siendo esta la que comienza el escaneo de dispositivos cercanos al momento de iniciar Figura 15) y los intentos de conexión luego de que el usuario decide enviar un mensaje (*Figura 16*). Permitiéndole al usuario sólo decidir la frecuencia con la que la aplicación intentara conectarse a dispositivos cercanos y la cantidad de dispositivos que quiere que reciban y difundan su mensaje *(Figura 17).*

Figura 15: UbbMessengrer: "Inicio"

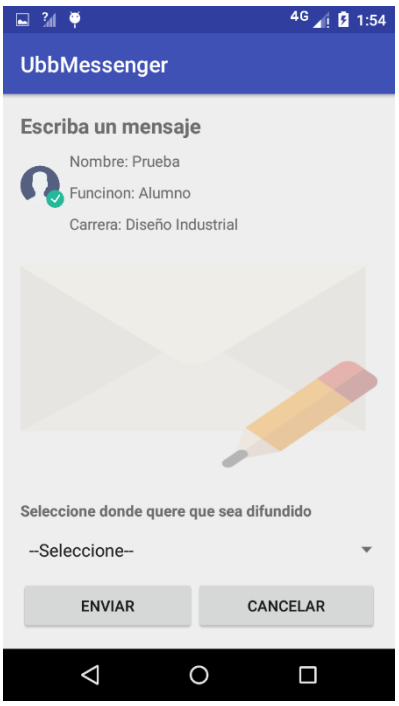

Figura 16: UbbMessengrer: "Escribir mensaje"

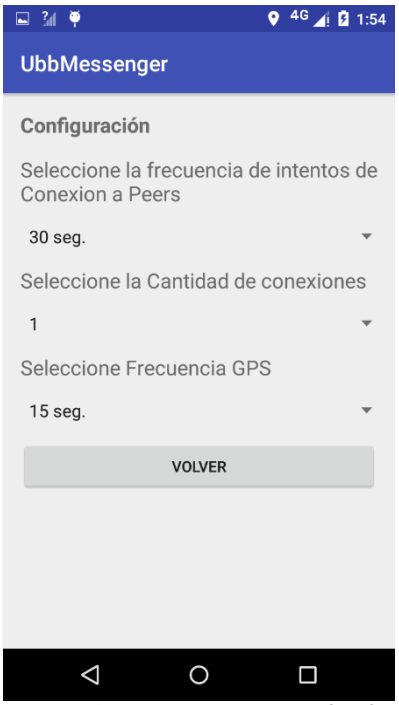

Figura 17: UbbMessengrer: "Configuración"

Luego, cuando el dispositivo que intenta difundir el mensaje el cual está actuando como Group Owner dispone de una lista de dispositivos disponibles, comienza con los intentos de conexión *(Figura 18)* dándole un tiempo límite a cada intento de 15 segundos, los cuales transcurridos sin éxito en la conexión detiene el proceso con el Peer actual pasando al siguiente.

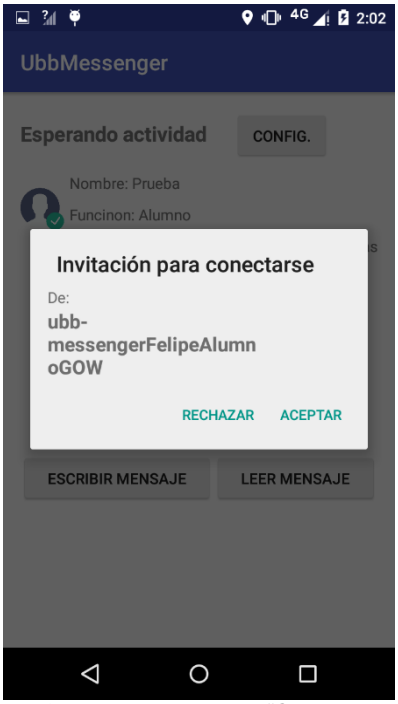

Figura 18: UbbMessengrer: "Conexión entrante"

6.4 Envío de mensajes

En la UI principal *(Figura 15)* y solo si el dispositivo no se encuentra ya portando e intentando de distribuir un mensaje, el botón "Escribir mensaje" se encontrará disponible para el usuario. Al ser presionado la UI cambiara mostrando al usuario una sección para escribir su mensaje, desplegando una lista de los edificios disponibles como destino para el mensaje y los botones "Enviar" el cual comenzará con el proceso de conexión a dispositivos *(Figura 18)* en búsqueda de encontrar al mejor candidato y el botón "Cancelar" el cual descartara el mensaje.

La aplicación comienza un estado de búsqueda y conexión de dispositivos disponibles hasta que encuentre un mejor candidato para portar el mensaje o se encuentre en el destino y comience la difusión. No permitiéndole al usuario realizar el envío de otro mensaje hasta que el actual salga del dispositivo *(Figura 19).*

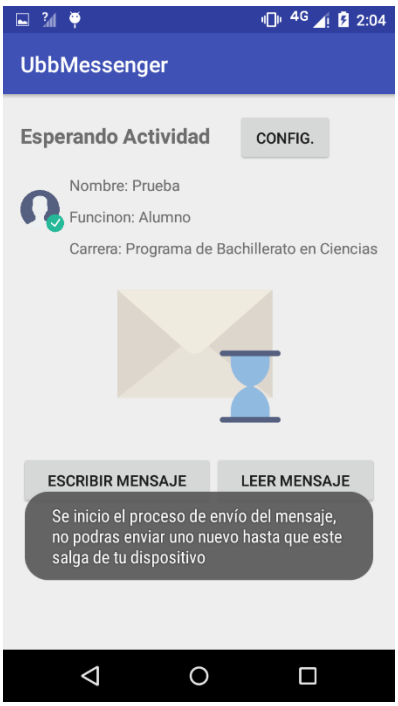

Figura 19: UbbMessengrer: "Proceso envío Mensajes"

6.5 Técnica propuesta: Mecanismo de difusión basado contexto con WANET mediante Wi-Fi Direct

En este punto se detalla la técnica de ruteo propuesta para el mecanismo de difusión basándonos en las redes WANET y las capacidades disponibles en la API Wi-Fi Direct en Android, luego de haber estudiado y entendido las capacidades y limitantes de este Framework, la técnica propuesta la llamaremos *Ruteo bajo demanda del camino probable más óptimo*.

La tecnología Wi-Fi Direct disponible en Android sólo permite crear redes con una topología tipo estrella cuyo funcionamiento fue especificado en la sección ***3.8.1 Topología de estrella***, no permitiendo crear una red WANET tradicional que pueda cubrir una amplia zona utilizando cada dispositivo como un nodo que permite la comunicación con sus vecinos.

6.5.1 Ruteo bajo demanda del camino probable más óptimo.

Esta técnica propone que, utilizando información de comportamiento otorgada por el usuario, apoyado de sensores de ubicación disponibles en Android como lo es GPS y el algoritmo de búsqueda de camino más corto A*, se puede calcular y estimar el usuario o dispositivo más óptimo para acercar o transportar el mensaje a su lugar de destino, para que este sea difundido.

En el primer inicio de la aplicación el usuario debe ingresar los 3 edificios que más visita de la Universidad, en base a estos edificios y su ubicación actual se asume que el usuario se dirige o acercará a alguno de estos edificios, en este caso el más cercano dadas las coordenadas.

Al realizar el envío de un mensaje por parte de un usuario, el algoritmo consulta en la base de datos la tabla que representa el grafo de la universidad a partir de sus edificios y sus vecinos directos en línea recta, además consulta la tabla con las coordenadas de todos los edificios y calcula las distancias en línea recta que existen entre estos, trazando el camino más óptimo a partir del edificio más cercano al usuario y la meta que posee el mensaje.

Esta operación se realiza con el usuario que actualmente posee el mensaje, los dispositivos disponibles con los que ha logrado conexión y enviado la solicitud para que realicen el cálculo de su ruta optima, comparando la distancia total que tomaría al mensaje recorrer dependiendo de cada dispositivo la meta seleccionada y las preferencias de comportamiento ingresadas por el usuario.

Se asume que, si bien el usuario puede no dirigirse al edificio seleccionado, en el camino puede encontrarse con otros usuarios con preferencias similares o mejores haciendo al mensaje saltar de un usuario a otro conforme las rutas optimas que se vallan calculando. También se asume que los edificios son puntos donde se concentran grandes cantidades de usuarios, los cuales tienen distintas preferencias en sus edificios más visitados, apoyando la tesis de que trazar la ruta a partir de la ubicación de los edificios es una opción completamente viable.

Figura 20: Grafo Universidad del Bío-Bío

Por ejemplo, se tiene un usuario en la Facultad de Ciencias Empresariales que desea enviar un mensaje a los usuarios presentes en la Facultad de Ingeniería.

Su información en la aplicación es la siguiente:

Dispositivo 0:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima calculada**	**Distancia de la ruta**
-Facultad de Ciencias Empresariales -Biblioteca -Aulas AC	Facultad de ciencias Empresariales	Facultad de Ingeniería Edificio Gantes	-Facultad de ciencias Empresariales -Biblioteca -Departamento de Ing. Mecánica -Salas multipropósitos 3 -Salas de estudio Industrial -Escuela de Trabajo social -Facultad de Ing. E. Gantes	359m.

Tabla 6: Dispositivo 0 ejemplo 1

El dispositivo inicial es capaz de encontrar a otros 2 dispositivos e intenta conectarse a ellos informándoles de la meta del mensaje, cuya información es la siguiente:

Dispositivo 1:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima calculada**	**Distancia de la ruta**
-Biblioteca -Aulas AA -Departamento de Ing. Mecánica	-Biblioteca	Facultad de Ingeniería Edificio Gantes	-Biblioteca -Departamento de Ing. Mecánica	301m

			-Salas multipropósitos 3 -Salas de estudio Industrial -Escuela de Trabajo social -Facultad de Ing. E. Gantes	

Tabla 7: Dispositivo 1 ejemplo 1

Dispositivo 2:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima Calculada**	**Distancia de la ruta**
-Escuela de Arquitectura -Salas de estudio Multipropósito -Casino	Escuela de Arquitectura	Facultad de Ingeniería	-Escuela de Arquitectura -Facultad de Ciencias -Facultad de Ing. E. Gantes	136m

Tabla 8: Dispositivo 2 ejemplo 1

Dado la naturaleza de la conexión donde los dispositivos no se encuentran necesariamente estáticos y la conexión podría perderse por motivos de rango de alcance del Wi-Fi realizar la evaluación de todos los dispositivos disponibles y seleccionar al más óptimo de la lista se vuelve una tarea menos viable, ya que, este podría dejar de estar disponible una vez realizada la comparación del mejor, por lo tanto la aplicación al detectar un portador más óptimo al actual hace un intercambio inmediato del mensaje aun siendo el dispositivo 2 mejor candidato que el dispositivo 1, ambos son mejores alternativas que el dispositivo 0 y el siguiente portador será determinado por cuál de los dispositivos logra primero una conexión través de Wi-Fi Direct con el que actúa como emisor, comenzando nuevamente el ciclo de búsqueda, conexión, comparación y transmisión del mensaje hasta

que el dispositivo detecte que se encuentra en un rango de 25 metros de la meta donde se comenzara la difusión del mensaje a todos los dispositivos disponibles en ese rango.

Ejemplo 2: Ahora suponiendo que el mismo emisor encontró dispositivos disponibles, pero con otras preferencias y rutas posibles, pero desea enviar el mensaje al mismo destino. La aplicación se expone a evaluar el siguiente caso:

Dispositivo 0:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima calculada**	**Distancia de la ruta**
-Facultad de Ciencias Empresariales -Biblioteca -Aulas AC	Facultad de Ciencias Empresariales	Facultad de Ingeniería Edificio Gantes	-Facultad de Ciencias Empresariales -Biblioteca -Departamento de Ing. Mecánica -Salas multipropósitos 3 -Salas de estudio Industrial -Escuela de Trabajo Social -Facultad de Ing. E. Gantes	359m.

Tabla 9: Dispositivo 0 ejemplo 2

Dispositivo 1:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima calculada**	**Distancia de la ruta**
-Aulas AB -Aulas AC -Departamento Ing. Industrial	-Aulas AC	Facultad de Ingeniería Edificio Gantes	-Aulas AC -Aulas AB -Edificio Metodológico	319m.

			-Aulas AA -Departamento Ing. Industrial -Aula Magna -Salas de estudio Trabajo Social -Facultad de Ing. E. Gantes	

Tabla 10: Dispositivo 1 ejemplo 2

Dispositivo 2:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima Calculada**	**Distancia de la ruta**
-Escuela de Diseño Industrial -Centro de Estudio de Polímeros Avanzados -Casino	Casino	Facultad de Ingeniería	-Casino -Escuela de diseño Industrial -Centro de Biomateriales y Nanotecnología -Departamento de Ing. En Maderas -Departamento de Ing. Industrial -Aula Magna -Sala de estudio Trabajo Social	413m

			-Facultad de Ing. E. Gantes	

Tabla 11: Dispositivo 2 ejemplo 2

En este caso la aplicación se expone a decidir entre dos opciones, una más óptima que la actual (Dispositivo 1) y otra que se presenta como una opción menos optima (Dispositivo 2), en el caso de lograr conectarse en primera instancia con el Dispositivo 2 este será evaluado y descartado por poseer una distancia mayor a la actual. Al conectarse al Dispositivo 1 este será evaluado siendo seleccionado como una opción más óptima a la actual, transformándolo en el nuevo portador del mensaje.

Ejemplo 3: Ahora suponiendo el caso de que el mismo dispositivo inicial detecta otros dos dispositivos, pero en este caso ninguno es mejor al actual

Dispositivo 0:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima calculada**	**Distancia de la ruta**
-Facultad de Ciencias Empresariales -Biblioteca -Aulas AC	Facultad de Ciencias Empresariales	Facultad de Ingeniería Edificio Gantes	-Facultad de Ciencias Empresariales -Biblioteca -Departamento de Ing. Mecánica -Salas multipropósitos 3 -Salas de estudio Industrial -Escuela de Trabajo Social -Facultad de Ing. E. Gantes	359m.

Tabla 12: Dispositivo 1 ejemplo 3

Dispositivo 1:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima calculada**	**Distancia de la ruta**
-Laboratorio CIM -Casino -Edificio de sistema territorial	-Casino	Facultad de Ingeniería Edificio Gantes	-Casino -Escuela de Diseño Industrial -Centro de Biomateriales y Nanotecnología -Departamento de Ing. En Maderas -Departamento de Ing. Industrial -Aula Magna -Sala de estudio Trabajo Social -Facultad de Ing. E. Gantes	413m.

Tabla 13: Dispositivo 1 ejemplo 3

Dispositivo 2:

Preferencia Edificios	**Edificio más cercano según ubicación actual**	**Meta seleccionada**	**Ruta óptima Calculada**	**Distancia de la ruta**
-Escuela de Diseño Industrial	Casino	Facultad de Ingeniería	-Casino	413m

-Centro de Estudio de Polímeros Avanzados -Casino			-Escuela de Diseño Industrial -Centro de Biomateriales y Nanotecnología -Departamento de Ing. En Maderas -Departamento de Ing. Industrial -Aula Magna -Sala de estudio Trabajo Social -Facultad de Ing. E. Gantes	

Tabla 14: Dispositivo 2 ejemplo 3

En este caso ninguno de los dispositivos disponibles es mejor candidato al actual por lo tanto la aplicación se conectará y evaluará cada dispositivo, descartándolos y repitiendo el ciclo de búsqueda, conexión y evaluación de dispositivos disponibles hasta encontrar un mejor candidato que el actual o hasta encontrarse en un rango de 25 metros del punto geográfico almacenado correspondiente al edificio meta.

7 Pruebas

En este punto se llevarán a cabo pruebas sobre la aplicación desarrollada durante este proyecto. El desarrollo de estas pruebas se realiza usando 5 dispositivos en total posicionándolos en topología estrella, explicada en la sección *3.8.3 Topología estrella* a una distancia de 2 metros del nodo central el cual actuará como Group Owner.

7.1 Elementos de prueba

- **Difusión del mensaje:** En este elemento de prueba se verifica que al encontrarse en un edificio meta, el mensaje sea difundido en todos los dispositivos disponibles.
- **Conexión a clientes:** En este elemento de prueba se verifica que la aplicación solicite la conexión a los dispositivos disponibles.
- **Selección del más óptimo:** En este elemento de prueba se verifica que la aplicación sea capaz de seleccionar el dispositivo disponible más óptimo para el destino del mensaje.

7.2 Especificación de pruebas

ID	Características a probar	Objetivo de la prueba	Enfoque para la definición de casos de prueba	Actividades de prueba	Criterios de cumplimiento
01	Funcionalidad	Verificar que la aplicación solicite la conexión automáticamente a los dispositivos disponibles.	Caja negra	-Enviar solicitud de conexión. -Conectarse	-Interfaz gráfica cliente despliega mensaje de conexión entrante.
02	Validación	Verificar que la aplicación no permita al usuario enviar un mensaje	Caja negra	-Mientras se es portador de un mensaje	-Interfaz gráfica despliega aviso

		nuevo mientras es portador de otro.		presionar el botón. "Escribir mensaje"	informando que en este momento. está portando un mensaje y no puede enviar otro.
03	Funcionalidad	Verificar que la aplicación seleccione correctamente el portador más óptimo.	Caja negra	-Conectarse a cliente -Solicitar peso ruta según meta -Evaluar pesos	Si el cliente es más óptimo que el Group Owner transferirá el mensaje desplegando por interfaz que fue enviado. De no ser más óptimo el cliente el Group Owner lo descartará.
04	Funcionalidad	Verificar que la aplicación sea capaz de detectar que se encuentra en el edificio meta	Caja negra	-Solicitar ubicación actual GPS -Consultar ubicación GPS de la meta -Calcular distancia a la meta	La aplicación al detectar que se encuentra a una distancia menor a 25m de la meta pasara a estado de difusión.

05	Funcionalidad	Difundir el mensaje a todos los dispositivos disponibles.	Caja negra	-Conectar clientes al Group owner -Transferir mensaje	La app mostrara en la interfaz de todos los clientes disponibles el mensaje.
06	Interfaz	Mostrar el mensaje portado	Caja negra	-presionar el botón “Leer Mensaje”	Si es portador un mensaje la aplicación mostrara por interfaz la información del mensaje. Si no es portador la aplicación mostrara aviso de que no es portador.

Tabla 15: Especificación de pruebas

7.2.1 Resultados de las pruebas

ID	Datos de entrada	Salida esperada	Salida obtenida	Éxito/Fracaso
01	Mensaje enviado desde dispositivo	-Interfaz gráfica cliente despliega mensaje de conexión entrante.	-El dispositivo cliente recibió la solicitud de conexión.	Éxito
02	Mensaje	Notificación que informe al usuario que ya es portador de un mensaje.	-Notificación "Usted ya es portador de un mensaje, no puede enviar otro hasta que el actual salga del dispositivo".	Éxito
03	Peso cliente	Cliente muy pesado cancelando conexión.	Cliente muy pesado cancelando conexión	Éxito
04	Ubicación actual GPS	Distancia menta menor a 25m difundiendo.	"Es meta difundiendo".	Éxito
05	Lista de dispositivos disponibles	Difundiendo mensaje	Mensaje difundido	Éxito
06	Presionar "Leer mensaje"	Alerta de que no es portador de ningún mensaje.	"Usted no porta ningún mensaje"	Éxito

Tabla 16: Resultados de pruebas

8 Experimentos

En esta sección se realizarán y documentaran distintos experimentos para evaluar el rendimiento de la técnica aplicada. Se enviarán mensajes con distintos escenarios de meta y preferencias de clientes disponibles.

Los distintos datos que se evaluarán en el experimento son:

- Tiempo total de conexión: Tiempo que demora el Group Owner conectarse y evaluar cada cliente disponible.
- Tiempo de transferencia: Tiempo que demora el mensaje llegar al cliente más óptimo disponible.
- Tiempo total de difusión: Tiempo que demora el Group Owner difundir el mensaje a todos los clientes disponibles.
- Consumo de batería: Niveles de batería consumidos por la aplicación en estado de difusión, y evaluación de clientes.

Para la realización de los experimentos, se utilizarán cinco equipos provistos por el laboratorio de transferencia tecnológica, actuando uno como emisor y los otros cuatro como receptores.

Se realizarán los experimentos en el laboratorio de transferencia tecnológica, este será el punto desde donde se calculan las distancias y las rutas optimas a los distintos destinos.

8.1 Gráfico Tiempo de Difusión vs número de clientes

Estos gráficos muestran los resultados de los tiempos de difusión de un mensaje al total de clientes disponibles en distintas distribuciones de los dispositivos y distancias.

- **Distribución 1:** muestra los tiempos de difusión en segundos en una topología tipo estrella a una distancia de 40 cm. cada dispositivo del que actúa como emisor.
- **Distribución 2:** muestra los tiempos de difusión en segundos en una topología tipo árbol a una distancia de 2 m. cada dispositivo del que actúa como emisor.
- **Distribución 3:** muestra los tiempos de difusión en segundos en una distribución tipo fila posicionando cada dispositivo a una distancia de 1 metro, 2 metros, 3 metros y 4 metros del dispositivo emisor.

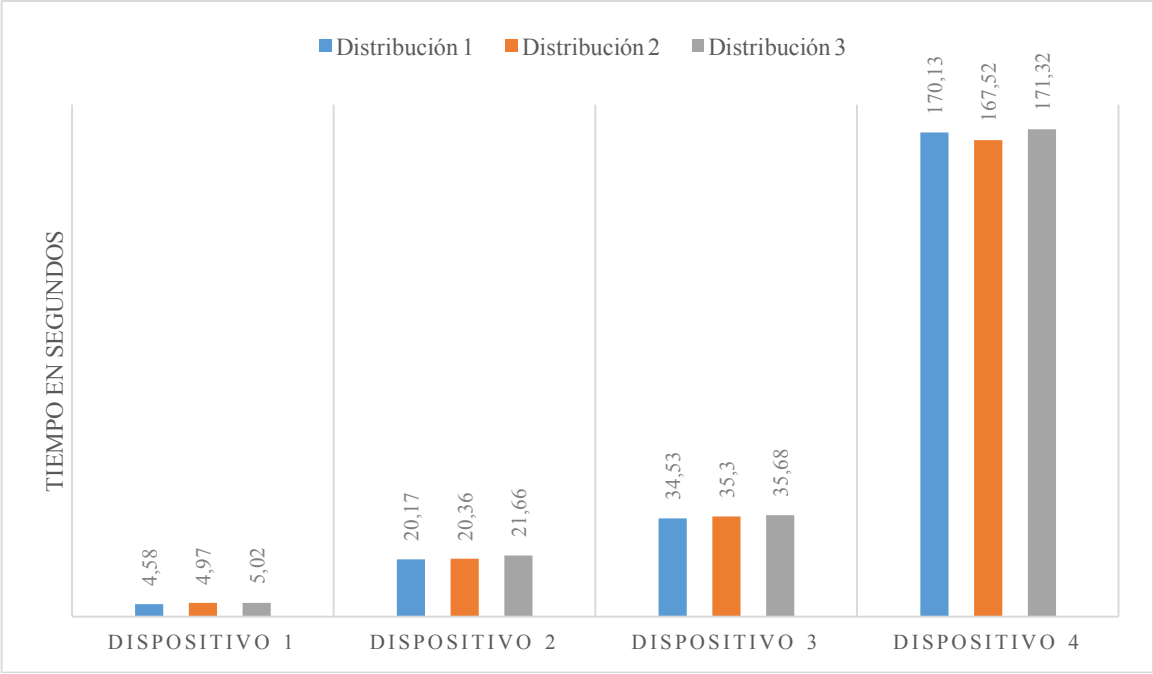

Figura 21 Gráfico difusión

Como se puede observar la distancia a la que se ubiquen los dispositivos no tienen una mayor repercusión en los tiempos de difusión del mensaje mientras se mantengan dentro del rango de alcance del Wi-Fi. Tampoco tiene una incidencia la distribución que se utilice

ya que esto no condiciona a que dispositivo le llegara primero la solicitud de conexión desde el emisor ya que, este comienza las solicitudes de conexión en el orden que detecto a sus vecinos y genero la lista con la información de cada uno, pudiendo ser el más lejano o más cercano el primero en recibir la conexión siendo esto totalmente al azar.

También se observó que los tiempos entre cada dispositivo que recibió el mensaje son de aproximadamente 15 segundos. Tiempo que se le dio internamente a la aplicación para que mantuviera la conexión e hiciera el traspaso de datos con éxito. Con el ultimo vecino agregado a la lista interna ocurre un error del tipo “hardware ocupado” tomándole un mayor tiempo poder recibir el mensaje.

8.2 Grafico tiempo de evaluación cliente óptimo vs número de clientes

Este grafico muestra los resultados de los tiempos que demora la aplicación en evaluar los clientes disponibles con un distinto número de clientes hasta encontrar un cliente más óptimo.

Se tomó el tiempo en segundos que demora la aplicación en evaluar 1 cliente menos optimo, luego 2 clientes menos óptimos, luego 3 clientes menos óptimos y finalmente 3 clientes menos óptimos y 1 cliente óptimo.

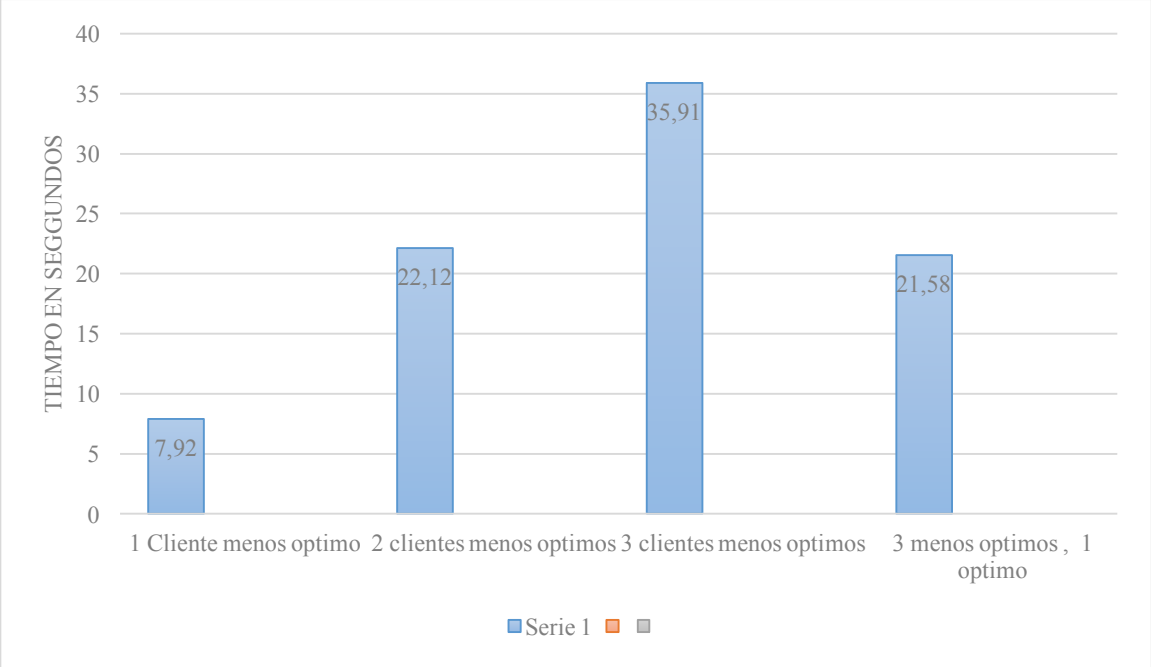

Figura 22: Gráfico evaluación Cliente Optimo

Como se pudo observar durante este experimento a mayor cantidad de clientes el tiempo que tarda aumenta bastante proporcionalmente siendo lo que más demora el proceso establecer la conexión. Una vez establecida la conexión que el cliente entregue su ruta posible más óptima tarda una cantidad de tiempo casi imperceptible por el usuario.

También se pudo observar nuevamente que la conexión a los clientes es completamente al azar ya que cuando se presentaron cuatro clientes siendo uno de ellos el óptimo fue a este el segundo que intentó establecer conexión. Tardando en transferir un mensaje un tiempo muy similar al que demoro evaluar dos clientes menos óptimos.

8.3 Gráfico consumo batería vs número de clientes difundido el mensaje

Este grafico muestra los resultados de consumo de batería para el dispositivo que actúa como nodo central según la cantidad de clientes que se difunde el mensaje.

Para este experimento se estableció al dispositivo central con su pantalla encendida al 100% de brillo. Se desconectó del cargador con un 99% de batería cargada. Además, se agregaron 2 dispositivos personales extra al experimento.

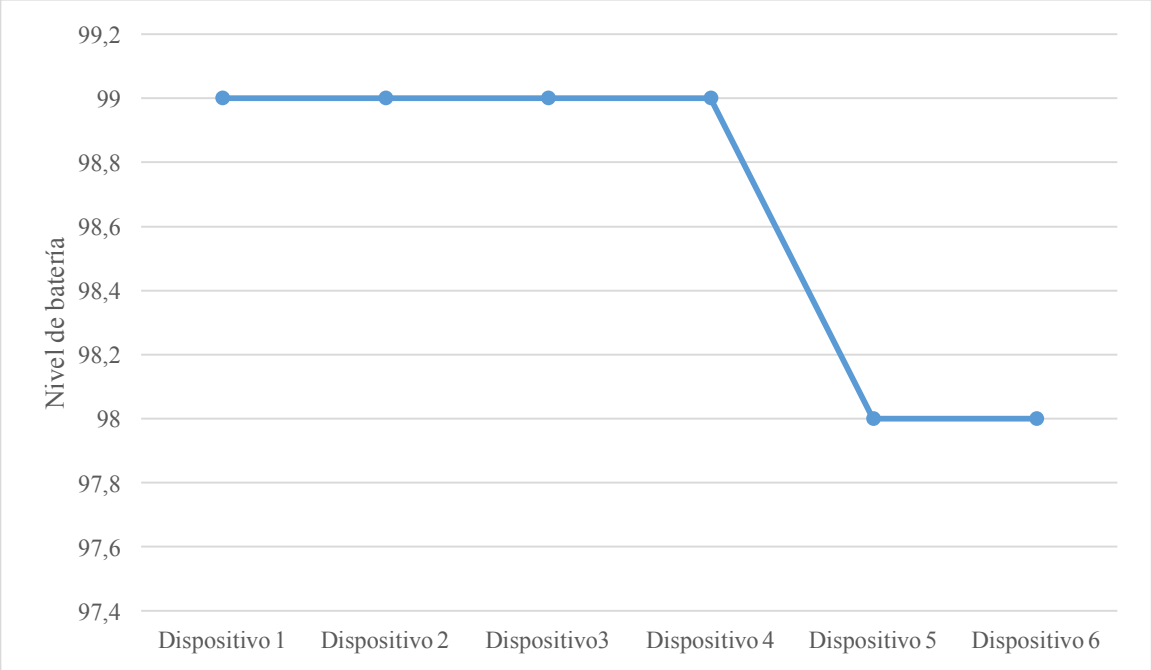

Figura 23: Grafico consumo batería difusión del mensaje

Como se puede observar bajo las condiciones descritas anteriormente el nivel de batería bajó un 1% luego de entregar el mensaje a tan sólo 4 dispositivos en un lapso aproximado de 55 segundos. Mostrando que el consumo de batería de la aplicación puede ser un punto en contra de esta propuesta ya que hace uso intensivo tanto del GPS como del Wi-Fi.

8.4 Gráfico Consumo de batería reposo vs Buscando Peers

En este gráfico se muestra el consumo de la batería en función del tiempo en estado de escaneo en búsqueda de *peers* y en estado de reposo.

Para realizar este experimento que busca mostrar cuan amplio es el impacto en el consumo de la batería por la aplicación desarrollada, se estableció el brillo de la pantalla en 50% con pantalla activada y se tomó el nivel de batería cada cinco minutos.

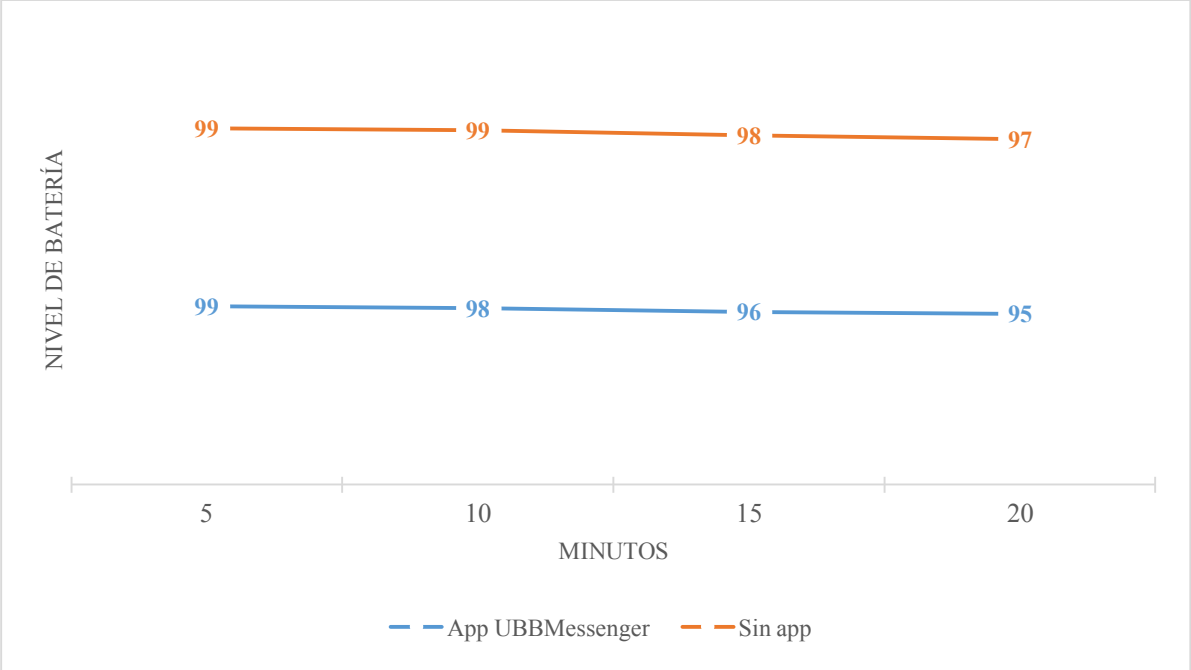

Figura 24: Gráfico consumo batería APP vs Reposo

Como se puede observar el uso de la aplicación representa un mayor consumo de batería en este y en el experimento anterior donde se midió este punto se puede inferir que la duración de la batería mientras se ejecuta la aplicación, podría ser el único punto en contra de esta tecnología.

También se propone para trabajos futuros considerar el nivel de batería en el algoritmo de ruteo, para así evitar equipos que puedan agotar su batería antes de entregar el mensaje.

8.5 Experimento de transporte de mensaje

El siguiente experimento a realizarse en el laboratorio de transferencia tecnológica se utilizarán cuatro dispositivos dispuestos por la universidad y se dispondrán como lo indica la figura

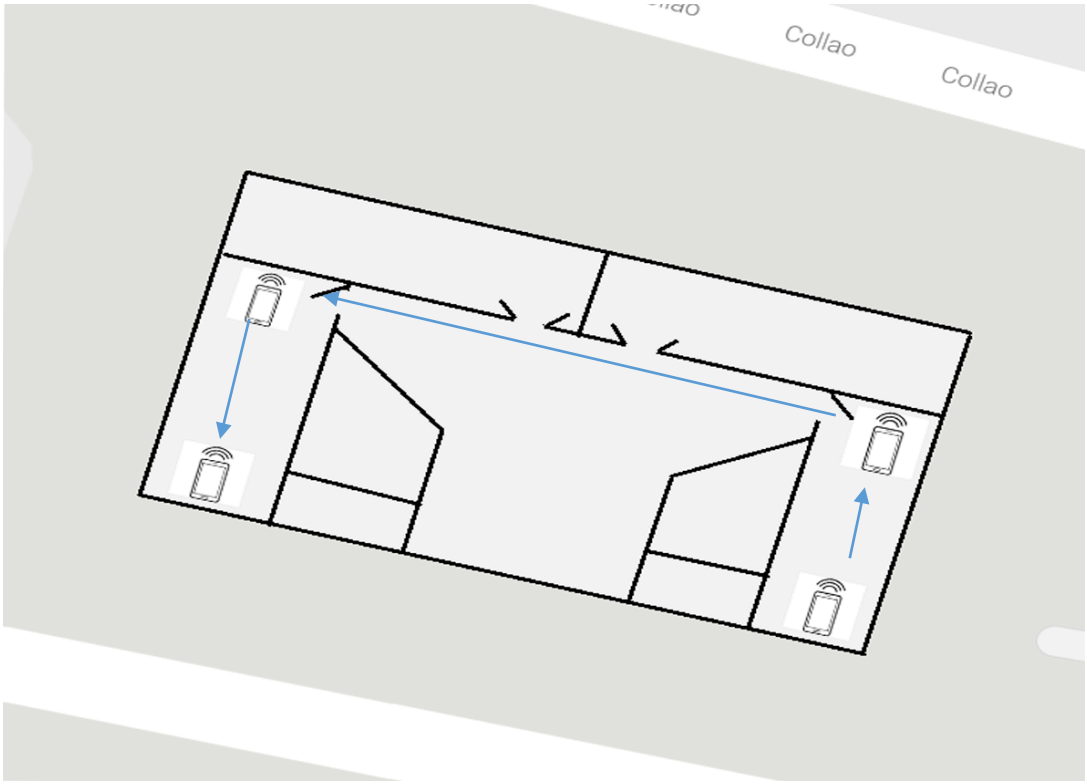

Figura 25: transporte de mensaje

El objetivo de este experimento es emitir un mensaje desde el laboratorio 1 hasta el laboratorio 2 siguiendo el sentido de las flechas. Se realizaron 3 experimentos de transporte donde los resultados son los siguientes.

Prueba 1	Disp. 1	Disp. 2	Disp.3
Salto 1	11s	-	-
Salto 2	-	35 s	-
Salto 3	-	-	68 s

Tabla 17: prueba transporte 1

Prueba 2	Disp. 1	Disp. 2	Disp.3
Salto 1	12s	-	-
Salto 2	-	-	-
Salto 3	-	-	68 s

Tabla 18: prueba transporte 2

Prueba 3	*Disp. 1*	*Disp. 2*	*Disp.3*
Salto 1	*10s*	*-*	*-*
Salto 2	*-*	*35 s*	*-*
Salto 3	*-*	*-*	*32 s*

Tabla 19: prueba transporte 3

Se puede observar los tiempos que tardaron los mensajes en salir de cada dispositivo en el caso de la prueba 1 el mensaje del dispositivo 2 tardo mas ya que la primera conexión lograda fue con el dispositivo anterior el cual fue descartado por ser menos optimo que el actual y la siguiente conexión fue con el dispositivo más óptimo, por esto el tiempo de transporte se duplico.

En la prueba 2 el dispositivo 1 detecto directamente al dispositivo 3 saltando al dispositivo 2 re afirmando la observación de que las conexiones se producen de manera aleatoria, también se observa que el tiempo fue mayor ya que primero se conectó al dispositivo origen el cual descartó por ser menos óptimo saltando inmediatamente al dispositivo 3.

En la prueba 3 se pueden observar los tiempos de transporte del mensaje en condiciones óptimas donde el primer dispositivo conectado era el inmediatamente mas óptimo.

9 Conclusiones

Los objetivos presentados en la propuesta del proyecto fueron alcanzados con éxito, si bien no se realizó una simulación en el entorno de eventos discretos NS-3 propuesta inicialmente. Se optó por el desarrollo de una aplicación completamente funcional, más allá de la propuesta de un principio. Dado a que en el transcurso de la investigación de la tecnología *Wi-Fi Direct* en *Android* se observó que de esta manera se obtendrían resultados mejores y más reales. Esta decisión rindió frutos permitiendo prescindir de un ambiente simulado, realizándose las pruebas pensadas inicialmente para el simulador en un ambiente físico de laboratorio. Dado esta decisión también la planificación inicial sufrió cambios y debió ser extendida a dos semestres para poder realizar una investigación a cabalidad.

La investigación de la tecnología *Wi-Fi Direct* descrita por la IEEE y su implementación disponible como *framework* en Android reveló que; El *framework* disponible en *Android* no es una versión completa tal y como es descrita por el organismo internacional. *Wi-Fi Direct* es descrita por la IEEE como una tecnología que permitiría la formación de redes inalámbricas P2P, en las cuales cada nodo es capaz de transmitir información entre sus vecinos. Esto no es posible de implementar en Android donde sólo el nodo central es capaz de comunicarse con los demás nodos. Pudiendo ser representada como una topología del tipo estrella. Otra limitante encontrada fue que este *framework* fue pensado para; escanear dispositivos cercanos, realizar una conexión con estos, transferir datos y detener la conexión. Tareas que al realizarse de forma repetitiva por las características dinámicas del algoritmo causan un bloqueo entre los procesos que requieren acceder al hardware del dispositivo, necesitando un control exhaustivo y minucioso de estos accesos, además de requerir tiempos mínimos para ejecutar cada tarea correctamente. Otra limitante encontrada fue que dado motivos de seguridad cada solicitud de conexión entrante en el dispositivo debe ser confirmada por el usuario.

Se propuso y desarrolló un algoritmo capaz de compensar la mayoría de las limitantes del *framework* Wi-Fi Direct provisto para Android. El cual permite transportar un mensaje estimando todas las rutas disponibles y seleccionando la mejor opción a través de las múltiples conexiones y solicitudes de rutas con su costo por cada vecino detectado.

A este algoritmo se le llamo "*Ruteo bajo demanda del camino probable más corto"* descrito en la sección 6.5.1.

El algoritmo alimentado por variables como la distancia entre edificios, preferencias de comportamiento del usuario y coordenadas actuales de este provista por sensores GPS, supero las expectativas. Transformándose en un protocolo de ruteo válido que entrego todos los mensajes según lo esperado en las pruebas realizadas.

Dada las limitaciones anteriormente nombradas, no es posible implementar la tecnología Wi-Fi Direct disponible en Android en una red *WANET* tradicional, impidiendo, por ejemplo; implementarla como una red de mensajería instantánea.

La aplicación final desarrollada surge como una alternativa completamente viable para el transporte y difusión de mensajes cuando ninguna de las redes tradicionales se encuentre disponible.

Esta tecnología e implementación cuenta con un amplio campo de estudio por desarrollar, por ejemplo; el envío de mensajes a usuarios específicos mediante criptografía asimétrica, conseguir rutas más rápidas integrando nuevos parámetros respecto al contexto en que se utilizará la aplicación, mejorando la técnica de ruteo considerando estado de la batería del dispositivo candidato, ajustando tiempos de conexión, etc.

Finalmente, desde el punto de vista académico este proyecto de título me permitió ahondar en una tecnología que se encuentra todavía poco explorada y demostrar que existe un gran campo de nuevos usos aún por descubrir. También me permitió crear y proponer una solución viable a las limitantes de esta tecnología que se fueron presentando durante la investigación, logrando el objetivo principal propuesto para este proyecto de crear una red de difusión móvil.

10 Bibliografía

[1] “Android Studio”. **Fuente:** https://en.wikipedia.org/wiki/Android_Studio

[2] “Wi-Fi” **Fuente:** https://es.wikipedia.org/wiki/Wifi

[3] “Wi-Fi direct” **Fuente:** https://es.wikipedia.org/wiki/Wi-Fi_Direct

[4] “Wireless ad hoc network”

Fuente: https://en.wikipedia.org/wiki/Wireless_ad_hoc_network

[5] “APP” **Fuente:** https://es.wikipedia.org/wiki/APP

[6] “Portador, ra” **Fuente:** http://dle.rae.es/?id=TjF68AB

[7] “Peer-to-Peer” **Fuente:** https://es.wikipedia.org/wiki/Peer-to-peer

[8] “Transmission Control Protocol”

Fuente: https://es.wikipedia.org/wiki/Transmission_Control_Protocol

[9] “Operating System”

Fuente: https://en.wikipedia.org/wiki/Operating_system

[10] “Punto de acceso inalámbrico”

Fuente: https://es.wikipedia.org/wiki/Punto_de_acceso_inalámbrico

[11] “ Teléfono Inteligente” **Fuente:** https://es.wikipedia.org/wiki/Teléfono_inteligente

[12] “Los dispositivos conectados en el mundo ya superan los 8000 millones”

Fuente: http://www.ituser.es/en-cifras/2016/06/los-dispositivos-conectados-en-el-mundo-ya-superan-los-8000-millones

[13]” Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 3rd quarter 2016“

Fuente: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[14] “Android” **Fuente:** https://es.wikipedia.org/wiki/Android

[15] “Interfaz de programación de aplicaciones”

Fuente: https://es.wikipedia.org/wiki/Interfaz_de_programación_de_aplicaciones

[16] “Transmisión de datos”

Fuente: https://es.wikipedia.org/wiki/Transmisión_de_datos

[17] “Cellular network” **Fuente:** https://en.wikipedia.org/wiki/Cellular_network

[18] “Las redes de transmisión de datos usadas en los teléfonos celulares”

Fuente:https://norfipc.com/celulares/redes-transmision-datos-usadas-telefonos-celulares.php

[19] “IEEE 802.11” **Fuente:** https://en.wikipedia.org/wiki/IEEE_802.11

[20] " Wireless LAN" **Fuente:** https://en.wikipedia.org/wiki/Wireless_LAN

[21] "Wireless acces point" **Fuente:** https://en.wikipedia.org/wiki/Wireless_access_point

[22] "Modo infraestructura" **Fuente:** https://meshias.wordpress.com/2010/10/16/redes-inalambricas-en-modo-infraestructura/

[23] "Red ad hoc inalámbrica"

Fuente:https://es.wikipedia.org/wiki/Red_ad_hoc_inalámbrica

[24] "Bluetooth" **Fuente:** https://www.bluetooth.com

[25] "Near field communication"

Fuente: https://es.wikipedia.org/wiki/Near_field_communication

[26] Wi-Fi Alliance, "Discover Wi-Fi, Wi-Fi Direct"

Fuente: http://www.wi-fi.org/discover-wi-fi/wi-fi-direct

[27] Wi-Fi Alliance, "How fast is Wi-Fi Direct?"

Fuente: http://www.wi-fi.org/knowledge-center/faq/how-fast-is-wi-fi-direct

[28]Wi-Fi Alliance, "How far does a Wi-Fi Direct connection travel?"

Fuente:http://www.wi-fi.org/knowledge-center/faq/how-far-does-a-wi-fi-direct-connection-travel

[29] Wi-Fi Alliance, "Wifi-Direct-certified products"

Fuente: http://www.wi-fi.org/wi-fi-direct-products

[30] Android Developers "Wi-Fi Peer-to-Peer"

Fuente: https://developer.android.com/guide/topics/connectivity/wifip2p.html

[31] Daniel Camps-Mur, Andres Garcia-Saavedra and Pablo Serrano, *"Device to device communications with WiFi Direct: overview and experimentation*"

Fuente: http://agsaaved.github.io/papers/2012_camps_wircommag.pdf

[32] Lu Han, "*Wireless Ad-hoc Networks".* 2004

Fuente:https://pdfs.semanticscholar.org/cbd2/9938dd4cbeb8aec3c21ddb60f0f20a4356cb.pdf

[33] Robin J. Wilson "Graph theory"

Fuente: http://www.maths.ed.ac.uk/~aar/papers/wilsongraph.pdf

[34] Tatiana Gutierrez "*Inteligencia artificial, métodos de búsqueda*". 2015

[35] "Types of Network Topology"

Fuente: http://www.studytonight.com/computer-networks/network-topology-types

[36] Colin Funai, Cristiano Tapparello, Wendi Heinzelman "Supporting Multi-hop Device-to-Device Networks Through WiFi Direct Multi-Group Networking". 2015

Fuente: https://arxiv.org/pdf/1601.00028.pdf

11 Anexos

11.1 Árbol de descomposición funcional

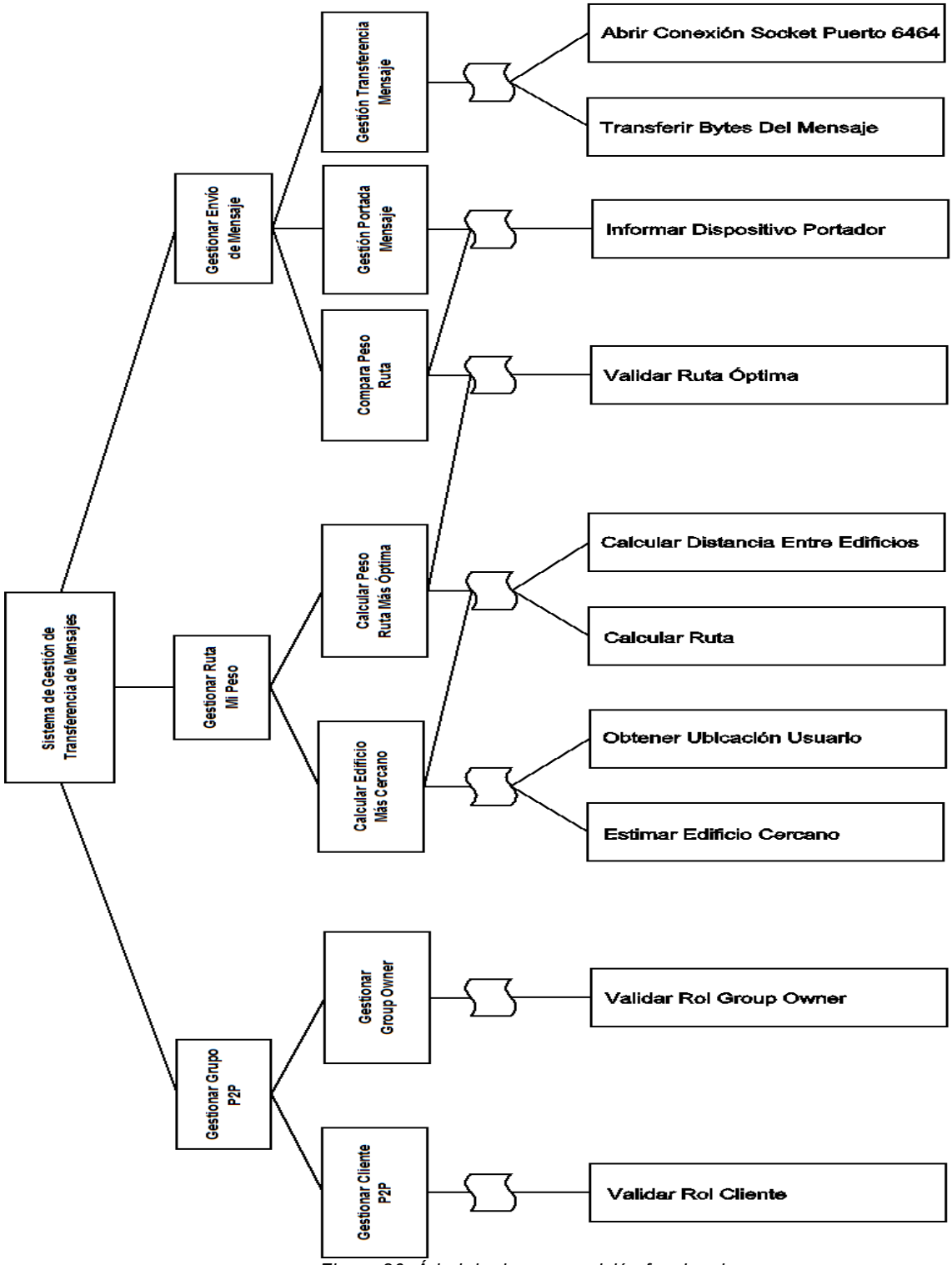

Figura 26: Árbol de descomposición funcional

11.2 Diseño de interfaz y navegación de la aplicación

En este apartado se presenta el diseño estándar que se usó en la aplicación, su jerarquía de menú y navegación.

11.2.1 Jerarquía de menú

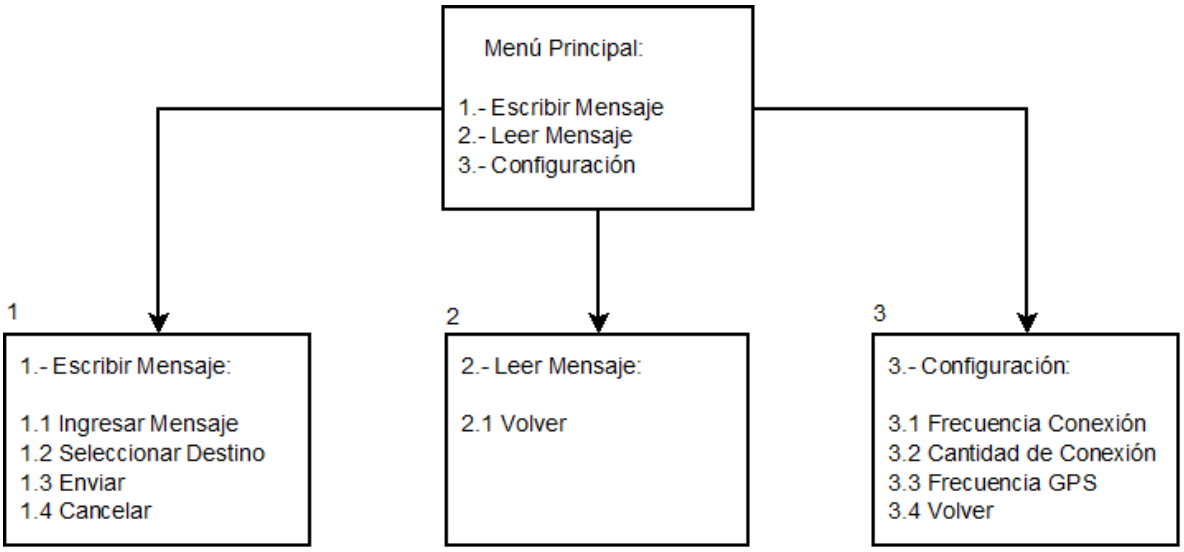

Figura 27: Jerarquía de menú

11.2.2 Navegación de menú

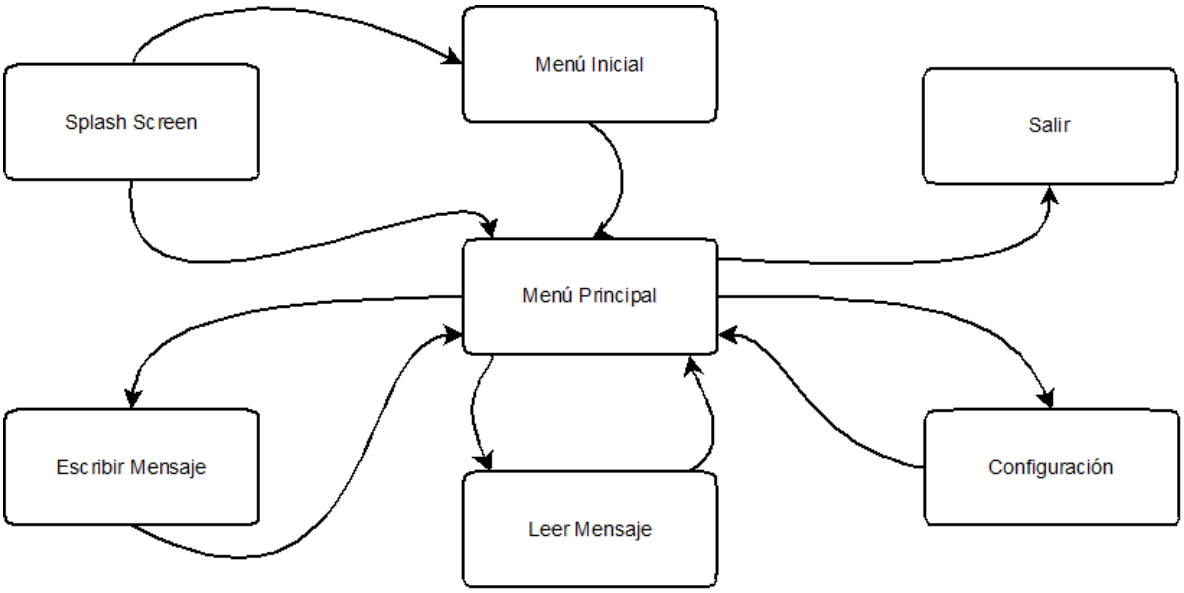

Figura 28: Navegación de menú

11.2.3 Diseño de la interfaz

En este punto se mostrará el diseño general, para el usuario utilizado para desarrollar la aplicación y el diseño final de la aplicación.

11.2.3.1 Diseño interfaz gráfica usuario primer inicio

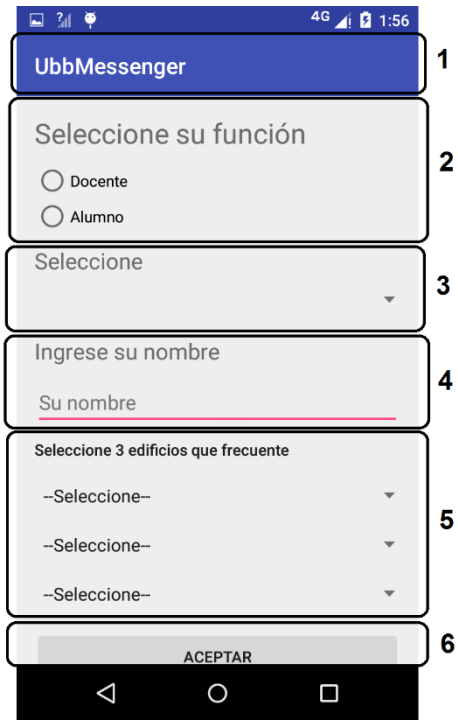

Figura 29: UbbMessengrer: "Diseño interfaz primer inicio"

Área 1: Encabezado de la aplicación. Incluye el nombre de la aplicación.

Área 2: Incluye título del área y opciones seleccionables.

Área 3: Incluye título y opción seleccionable según preferencia anterior.

Área 4: Incluye lugar donde el usuario ingresa su nombre.

Área 5: incluye 3 ítems donde el usuario selecciona edificios.

Área 6: Opción botón('Aceptar').

11.2.3.2 Diseño interfaz gráfica usuario general

Figura 30: UbbMessengrer: "Diseño interfaz Usuario general"

Área 1: Encabezado de la aplicación. Incluye el nombre de la aplicación.

Área 2: Incluye el título de la ventana en que se encuentra y el botón de configuración.

Área 3: Incluye la información del usuario actual del dispositivo.

Área 4: Incluye imagen acorde al estado que se encuentra la aplicación.

Área 5: Opción de botones ('Escribir mensaje' y 'Leer mensaje').

11.2.3.3 Diseño interfaz gráfica usuario que envía mensaje

Figura 31: UbbMessengrer: "Diseño interfaz Envía mensaje"

Área 1: Encabezado de la aplicación. Incluye el nombre de la aplicación.

Área 2: Incluye el título de la ventana en que se encuentra.

Área 3: Incluye la información del usuario actual del dispositivo.

Área 4: Incluye espacio donde permite al usuario escribir su mensaje, además, posee una imagen de fondo acorde al estado.

Área 5: Incluye una lista desplegable con los edificios para seleccionar el destino.

Área 6: Opción de botones ('Enviar' y 'Cancelar').

11.2.3.4 Diseño interfaz gráfica usuario que recibe mensaje

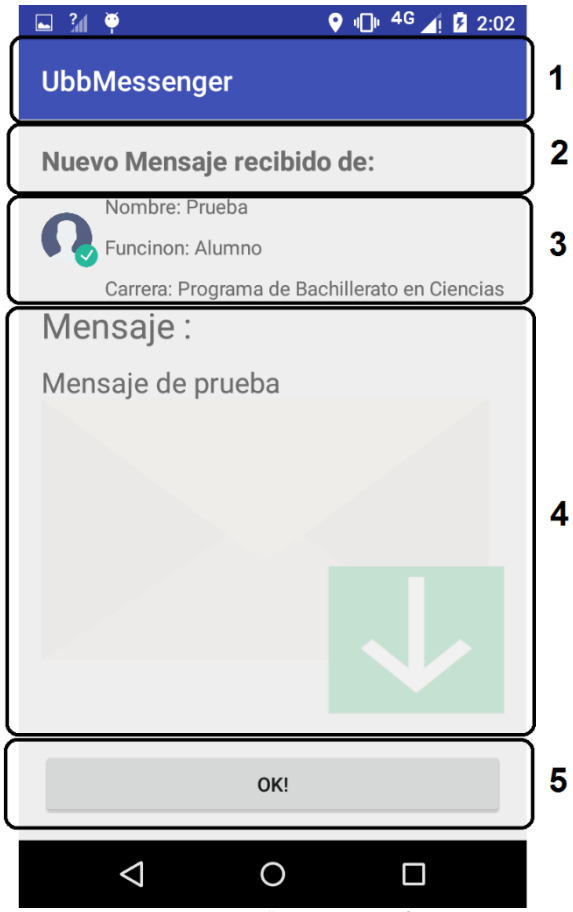

Figura 32: UbbMessengrer: "Diseño Interfaz Recibe mensaje"

Área 1: Encabezado de la aplicación. Incluye el nombre de la aplicación.

Área 2: Incluye el título de la ventana en que se encuentra.

Área 3: Incluye la información del usuario que envió el mensaje.

Área 4: Incluye mensaje recibido, posee imagen acorde al estado de la aplicación.

Área 5: Opción de botón ('Ok!).

11.3 Especificación de módulos

N° Módulo: 1		**Nombre Módulo**: Abrir conexión socket a puerto 6464	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: Puerto al cual conectarse	**Tipo de dato:** int	**Nombre:** Socket para transferencia de datos	**Tipo de dato:** socket

Tabla 20: módulo 1

N° Módulo: 2		**Nombre Módulo**: Transferir Bytes del Mensaje	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: Buffer mensaje	**Tipo de dato:** String	**Nombre:** Mensaje	**Tipo de dato:** string

Tabla 21: módulo 2

N° Módulo: 3		**Nombre Módulo**: Informar dispositivo Portador	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: Buffer decisión portador	**Tipo de dato:** string	**Nombre:** Decisión	**Tipo de dato:** string

Tabla 22: módulo 3

N° Módulo: 4		**Nombre Módulo**: Validar ruta más óptima	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: -Peso ruta cliente -Peso ruta Group Owner	**Tipo de dato:** -float - float	**Nombre:** Ruta más corta	**Tipo de dato:** float

Tabla 23 módulo 4

N° Módulo: 5		**Nombre Módulo**: Calcular distancia entre edificios	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: -Nombre edificio A -Nombre edificio B -Coordenadas edificio A -Coordenadas edificio B	**Tipo de dato:** -String -String -Array de float -Array de Float	**Nombre:** Distancia	**Tipo de dato:** float

Tabla 24: módulo 5

N° Módulo: 6		**Nombre Módulo**: Calcular ruta	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: -Edificio Inicio -Edificio Meta	**Tipo de dato:** -string -string	**Nombre:** Ruta	**Tipo de dato:** array de string

Tabla 25: módulo 6

N° Módulo: 7		**Nombre Módulo**: Obtener ubicación usuario	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: Almacenar ubicación	**Tipo de dato:** Boolean	**Nombre:** Ubicación usuario	**Tipo de dato:** array de float

Tabla 26: módulo 7

N° Módulo: 8		**Nombre Módulo**: Estimar edificio cercano	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: -Ubicación usuario -Preferencia edificios	**Tipo de dato:** -array de float -array de string	**Nombre:** Edificio más cercano	**Tipo de dato:** string

Tabla 27: módulo 8

N° Módulo: 9		**Nombre Módulo**: Validar rol Group Owner	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: Conexión Group Owner	**Tipo de dato:** String	**Nombre:** Resultado validación	**Tipo de dato:** int

Tabla 28: módulo9

N° Módulo: 10		**Nombre Módulo**: Validar rol Cliente	
Parámetros de entrada		**Parámetros de Salida**	
Nombre: Conexión cliente	**Tipo de dato:** String	**Nombre:** Resultado validación	**Tipo de dato:** int

Tabla 29: módulo10

12 Anexo: Código Fuente

En esta parte del anexo se presenta la codificación de las clases y métodos más importantes de la aplicación.

Clase ***aEstrella;*** esta clase es la encargada de recibir el nombre del edificio inicial y el nombre del edificio meta para calcular su ruta más óptima, calculando las distancias entre cada edificio dada las coordenadas geográficas y el grafo almacenados en una base de datos sqlite.

Sus principales métodos son:

Método **vecinos**; método que consulta a la base de datos y dado el nombre de un edificio devuelve un arraylist con el nombre de todos sus vecinos directos.

```
private ArrayList<Edificio> vecinos(Edificio padre, SQLiteDatabase
DataBase){
        ArrayList<Edificio> vecinos= new ArrayList<Edificio>();
        vecinos.clear();
        ArrayList<Integer> vecinosLista=new ArrayList<Integer>();
        String nombrePadre=padre.nombre;
        int indicePadre=0;


        Cursor cIndicePadre = null;
        try {
             cIndicePadre=DataBase.rawQuery("SELECT indice FROM
Coordenadas Where lugar="+"'"+nombrePadre+"'"+"",null);

            if(cIndicePadre.moveToFirst()){
                indicePadre=cIndicePadre.getInt(0);
            }
        } finally {
            if(cIndicePadre != null)
                cIndicePadre.close();
        }

        Cursor cVecinos = null;
        String aux;
        try {
            cVecinos=DataBase.rawQuery("Select "+"\""+indicePadre+"\""+"
FROM edificiosCercanos ",null);
            if(cVecinos.moveToFirst()){
                do{
                    aux=cVecinos.getString(0);

                    if(!aux.isEmpty()) {
```

```
                    vecinosLista.add(cVecinos.getInt(0));
                }

            }while (cVecinos.moveToNext());

        }
    } finally {

        if(cVecinos != null)
            cVecinos.close();
    }

    for(int i=0; vecinosLista.size()>i;i++){
        String auxNombre="";
        int indice=vecinosLista.get(i).intValue();
                Cursor cNombreVecino=null;
        try {
            cNombreVecino=DataBase.rawQuery("SELECT lugar FROM
Coordenadas WHERE indice="+indice+"",null);
            if(cNombreVecino.moveToFirst()){
                auxNombre=cNombreVecino.getString(0);
            }
        } finally {

            if(cNombreVecino != null)
                cNombreVecino.close();
        }
        float distanciaMeta=getDistance(auxNombre,eFin.nombre);
        float distanciaPadre=getDistance(auxNombre,nombrePadre);
        vecinos.add(new
Edificio(auxNombre,padre,distanciaMeta,distanciaPadre));
    }


    return vecinos;

}
```

Método **getDistance**; método que a partir del nombre de dos edificios calcula la distancia en metros que existe entre ellos, consultando sus coordenadas almacenadas en la base de datos.

```
private float getDistance(String nombre, String meta){
    Location locActual= new Location("");
    Location locMeta = new Location("");
    float latitudActual=0;
    float longitudActual=0;
    float latitudMeta=0;
    float longitudMeta=0;
```

```
        Cursor c = null;
        Cursor cMeta= null;
        try {
            c=db.rawQuery("Select latitud, longitud FROM Coordenadas
WHERE lugar=+"+"'"+nombre+"'"+"", null);
            if (c.moveToFirst()) {
                do {
                    latitudActual = (float) c.getDouble(0);
                    longitudActual =(float) c.getDouble(1);

                } while(c.moveToNext());
            }

        } finally {

            if(c!= null)
                c.close();
        }

        try {
            cMeta=db.rawQuery("Select latitud, longitud FROM Coordenadas
WHERE lugar=+"+"'"+meta+"'"+"", null);
            if (cMeta.moveToFirst()) {
                do {
                    latitudMeta = (float) cMeta.getDouble(0);
                    longitudMeta =(float) cMeta.getDouble(1);

                } while(cMeta.moveToNext());
            }
        } finally {
            if(cMeta != null)
                cMeta.close();
        }

        locActual.setLatitude(latitudActual);
        locActual.setLongitude(longitudActual);

        locMeta.setLatitude(latitudMeta);
        locMeta.setLongitude(longitudMeta);

        float distancia=locActual.distanceTo(locMeta);

        return distancia;
    }
```

Método **aEstrella2**; método que por medio del algoritmo conocido como A* calcula la ruta más óptima para un edificio de inicio y un edificio final.

```
private float aEstrella2( Edificio inicio , Edificio fin){

        boolean exit=false;
        abierta.add(inicio);
        cerrada.clear();
```

```
        Edificio actual;
        int indice=0;

        while (!abierta.isEmpty() && !exit){
            indice=getMinPeso(abierta);
            actual=abierta.get(indice);
            abierta.remove(indice);

            if(actual.nombre.equals(fin.nombre)){
                Log.d(TAG, "aEstrella2: se encontro la meta");
                exit=true;
                return ruta(actual);

            }
            cerrada.add(actual);
            sucesores.clear();
            sucesores.addAll(vecinos(actual,db));
            auxSuc.clear();
            auxSuc.addAll(sucesores);

            for(int i=0; sucesores.size()>i;i++){
                for(int j=0; abierta.size()>j;j++){

if(sucesores.get(i).nombre.equals(abierta.get(j).nombre)){
                        if(sucesores.get(i).g<abierta.get(j).g){
                            abierta.get(j).g=sucesores.get(i).g;
                            abierta.get(j).padre=sucesores.get(i).padre;
                            auxSuc.set(i,null);
                        }
                    }
                }
            }
            while(auxSuc.contains(null)){
                for(int i=0; auxSuc.size()>i; i++){
                    if(auxSuc.get(i)==null){
                        auxSuc.remove(i);
                        break;
                    }
                }
            }
            sucesores.clear();
            sucesores.addAll(auxSuc);

            for(int i=0; sucesores.size()>i;i++){
                for(int j=0; cerrada.size()>j;j++){

if(sucesores.get(i).nombre.equals(cerrada.get(j).nombre)){
                        auxSuc.set(i,null);
                    }
                }
            }
            while(auxSuc.contains(null)){
                for(int i=0; auxSuc.size()>i; i++){
                    if(auxSuc.get(i)==null){
                        auxSuc.remove(i);
```

```
                    break;
                }
            }
        }
        abierta.addAll(auxSuc);

    }

    return 0;

}
```

Clase **ComunicationThread**; esta clase se ejecuta en el lado del servidor en un hilo y es la encargada de una vez establecida la comunicación con el cliente, darle las instrucciones a este para que le responda con la información solicitada, instruirlo para que se prepare para ser portador del mensaje o no, o difundir el mensaje a este cliente.

```
public class CommunicationThread extends Main2Activity implements
Runnable{

    private String TAG ="Comunication Thread";
    private Socket clientSocket;
    private String meta;
    private String mensaje;
    private BufferedReader input;
    private PrintWriter out;
    private String read;
    private Context context;
    private float pesoCliente;
    private float miPeso;
    private ServerThread mServer;
    private Main2Activity mActivity;
    private float distanciaMeta=0;


    public CommunicationThread(Socket clientSocket, String mensj,String
edMeta,float peso ,Context context,ServerThread miServer, Main2Activity
activity) {

        this.mensaje=mensj;
        this.clientSocket = clientSocket;
        this.context=context;
        this.meta=edMeta;
        this.miPeso=peso;
        this.mServer=miServer;
        this.mActivity=activity;

    }

    @Override
    public void run() {
```

```
        synchronized (this) {
            Location E1= new Location("");
            E1=mActivity.coordenadasEdificio(meta);
            Location loc1= new Location("");
            loc1.setLatitude(mActivity.usr_coordenadas[2].getLatitud());

loc1.setLongitude(mActivity.usr_coordenadas[2].getLongitud());
            distanciaMeta=loc1.distanceTo(E1);
            Log.d(TAG, "run comunication thread: distancia de la meta:
"+distanciaMeta);
            if (distanciaMeta <= 25) {
                Log.d(TAG, "run: Me encuentro en un radio de 20 metros
del destino Comenzando difuncion del mensaje");
                try {
                    out = new PrintWriter(new BufferedWriter(
                            new
OutputStreamWriter(clientSocket.getOutputStream())),
                            true);
                    out.println("META");

                } catch (IOException e) {
                    e.printStackTrace();
                }
                out.flush();
                Log.e(TAG, "el mensaje es: " + mensaje);
                try {
                    out = new PrintWriter(new BufferedWriter(
                            new
OutputStreamWriter(clientSocket.getOutputStream())),
                            true);
                    out.println(mensaje);
                    mActivity.setDifundido(true);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } else{

                Log.e(TAG, "Meta del mensaje: " + meta);
            try {
                out = new PrintWriter(new BufferedWriter(
                        new
OutputStreamWriter(clientSocket.getOutputStream())),
                        true);
                out.println(meta);

                input = new BufferedReader(new
InputStreamReader(this.clientSocket.getInputStream()));
                read = input.readLine();
                pesoCliente = Float.parseFloat(read);

            } catch (IOException e) {
                e.printStackTrace();
            }
```

```
            if (miPeso > pesoCliente) {
                out.flush();
                Log.e(TAG, "el mensaje es: " + mensaje);
                try {
                    out = new PrintWriter(new BufferedWriter(
                            new
OutputStreamWriter(clientSocket.getOutputStream())),
                            true);
                    out.println(mensaje);
                } catch (IOException e) {
                    e.printStackTrace();
                }


                mActivity.runOnUiThread(new Runnable() {
                    public void run() {
                        mActivity.hideALL();
                        mActivity.showEnviado();

                    }
                });
                try {
                    sleep(5000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                mActivity.runOnUiThread(new Runnable() {
                    public void run() {

                        mActivity.hideEnviado();
                        mActivity.showEsperando();
                    }
                });
                mServer.restCantidadMensaje();
                if(mServer.cantidadConexiones<=0){
                    mActivity.setDeviceName();
                    mActivity.setPotador(false);
              }
            }
            if (miPeso < pesoCliente) {
                            out.flush();
                Log.e(TAG, "ComunicationThread : Cliente muy pesado
cancelando conexion  ");
                try {
                    out = new PrintWriter(new BufferedWriter(
                            new
OutputStreamWriter(clientSocket.getOutputStream())),
                            true);
                    out.println("NO");
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
          }
        }
```

Clase **ClientThread**; esta clase se ejecuta en un hilo por el lado del cliente, se encarga de establecer la conexión al socket 6464 y recibir las instrucciones del servidor para responderle con los datos requeridos y competir por ser el nuevo portador del mensaje.

```
public class ClientThread  implements Runnable  {
    private String TAG= "Client Thread ";
    private int puerto;
    private String ip="";
    private Socket socketCliente;
    private PrintWriter out;
    private BufferedReader input;
    private String meta;
    private Context context;
    private String read;
    private float miPesoCliente=0;
    private Coordenadas [] cord= new Coordenadas[3];
    private Main2Activity mActivity;

    public ClientThread(String GOW_IP, Context context, Coordenadas[]
miscord, Main2Activity activity){
        super();
        this.ip=GOW_IP;
        this.context=context;
        this.cord=miscord;
        this.mActivity=activity;

    }
    @Override
    public void run() {

        if (Looper.myLooper() == null) {
            Looper.prepare();
        }
        Log.e(TAG, "Thread Cliente Ejecutandose");
        try {
           this.socketCliente = new Socket(ip, 6464);
           try {
```

```
                this.input = new BufferedReader(new
InputStreamReader(this.socketCliente.getInputStream()));
                read = input.readLine();
                Log.e(TAG, "Edificio Meta: " + read);

                meta = read;

                if (meta.equals("META")) {

                    this.input = new BufferedReader(new
InputStreamReader(this.socketCliente.getInputStream()));
                    read = input.readLine();

                    mActivity.runOnUiThread(new Runnable() {
                        public void run() {
                            mActivity.hideEsperando();
                            mActivity.setNewInMessage(read);
                            mActivity.showNewMensaje();
                            mActivity.setDeviceNameRM();
                            try {
                                socketCliente.close();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                         }
                    });


                } else{

                miPesoCliente=mActivity.miPeso(meta,cord);

                Log.e(TAG, "run: Mi Peso cliente es: " + miPesoCliente,
null);

                String str = String.valueOf(miPesoCliente);
                out = new PrintWriter(new BufferedWriter(
```

```
                        new
OutputStreamWriter(socketCliente.getOutputStream())),
                        true);
                out.println(str);

                this.input = new BufferedReader(new
InputStreamReader(this.socketCliente.getInputStream()));
                read = input.readLine();

                if (read.equals("NO")) {
                    this.socketCliente.close();
                    mActivity.midiscoverPeers();

                } else {

                    mActivity.setDeviceNameGOW();
                    mActivity.setPotador(true);
                    mActivity.setMeta(meta);
                    mActivity.captureMensajeFinal(read);
                    this.socketCliente.close();
                }
            }
            } catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
}
```

Clase **WiFiDirectBroadcastReceiver**; esta clase es fundamental para establecer conexiones por medio de Wi-Fi Direct ya que permite saber, si esta función se encuentra activada en el dispositivo, obtener la lista de dispositivos disponibles para conexión, permite saber el estado en que se encuentra el wi-fi (conectado, desconectado, ocupado y buscando dispositivos) y permite utilizar banderas de control sobre este para evitar colisiones en las múltiples llamadas a los servicios Wi-Fi el cual es uno de los principales problemas en el uso de este Framework.

```
public class WiFiDirectBroadcastReceiver extends BroadcastReceiver {


    private WifiP2pManager mManager; // servicio del sistema
    private Channel mChannel;          // canal por donde se realizara la
conexion p2p
    private Main2Activity mActivity;   //actividad asociada al receptor
    WifiP2pInfo info;

    PeerListListener myPeerListListener;


    public WiFiDirectBroadcastReceiver(WifiP2pManager manager, Channel
channel,
                                       Main2Activity activity) {
        super();
        this.mManager = manager;
        this.mChannel = channel;
        this.mActivity = activity;
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action))
{
```

```
            int state =
intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);

                mActivity.setIsWifiP2pEnabled(true);

            } else {
                mActivity.setIsWifiP2pEnabled(false);
                Toast.makeText(mActivity, "habilite el wifi",
                        Toast.LENGTH_SHORT).show();
            }
        }
         else if
(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {

            if (mManager != null) {


                    mManager.requestPeers(mChannel,
mActivity.peerListListener);
            }
        } else if
(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {

            if (mManager == null) {
                return;
            }
            NetworkInfo networkInfo = (NetworkInfo) intent

.getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);

            if (networkInfo.isConnected()) {
                Log.e(mActivity.TAG, "onReceive: SE CONECTO", null);
                mActivity.setConectado(true);

                mManager.requestConnectionInfo(mChannel, mActivity);
            } else {
                Log.e(mActivity.TAG, "onReceive: SE DESCONECTO", null);
```

```
                mActivity.setConectado(false);

                if(mActivity.getPeersConectados()>0) {

                    mActivity.restPeerConectado();
                }

            }
        } else if
(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action)) {

            Toast.makeText(mActivity, "Cambio en el estado del wifi",
                    Toast.LENGTH_SHORT).show();
        }else if
(WifiP2pManager.WIFI_P2P_DISCOVERY_CHANGED_ACTION.equals(action)) {
            if(mManager==null){
                return;
            }
            int
estado=(int)intent.getIntExtra(WifiP2pManager.EXTRA_DISCOVERY_STATE,0);
            if(estado==1){
            mActivity.setBuscando(false);
            }else if(estado==2){
                mActivity.setBuscando(true);

            }
        }
    }
}
```

Clase **Main2Activity**; esta es la clase principal de la aplicación la cual controla toda la interfaz y posee los timertask que permiten a la aplicación operar de manera automática.

Sus métodos más importantes son:

Método **miPeso**; este método permite dada las coordenadas del usuario obtener cuál de los edificios almacenados en sus preferencias se encuentra más cercano y a partir de este edificio solicitar el cálculo de la distancia en metros a la clase *aEstrella* de una ruta optima hasta la meta seleccionada.

```
public float miPeso(String eMeta, Coordenadas[] user_cordenadas) {
        aEstrella caminoMasCorto;
        float miPeso = 0;
        float distancia1=0;
        float distancia2=0;
        float distancia3=0;
        Location loc1= new Location("");
        Location loc2= new Location("");
        Location loc3= new Location("");
        loc1.setLatitude(user_cordenadas[0].getLatitud());
        loc1.setLongitude(user_cordenadas[0].getLongitud());
        loc2.setLatitude(user_cordenadas[1].getLatitud());
        loc2.setLongitude(user_cordenadas[1].getLongitud());
        loc3.setLatitude(user_cordenadas[2].getLatitud());
        loc3.setLongitude(user_cordenadas[2].getLongitud());

        // primero saber cuales son los 3 edificios frecuentes desde las
sharedPreferences y calcular de cual esta más cerca
        SharedPreferences prefs =
                getSharedPreferences("MisPreferencias",
Context.MODE_PRIVATE);

        String edificio1 = prefs.getString("edificio1", "null");
        String edificio2 = prefs.getString("edificio2", "null");
        String edificio3 = prefs.getString("edificio3", "null");

        Location E1= new Location("");
        Location E2= new Location("");
        Location E3= new Location("");

        E1=coordenadasEdificio(edificio1);
        E2=coordenadasEdificio(edificio2);
        E3=coordenadasEdificio(edificio3);


distancia1=loc1.distanceTo(E1)+loc2.distanceTo(E1)+loc3.distanceTo(E1);

distancia2=loc1.distanceTo(E2)+loc2.distanceTo(E2)+loc3.distanceTo(E2);
```

```
distancia3=loc1.distanceTo(E3)+loc2.distanceTo(E3)+loc3.distanceTo(E3);



        if(distancia1<distancia2&& distancia1<distancia3){// calcular
peso edificio 1 + la distancia que se encuentre de este edificio
            caminoMasCorto= new aEstrella(edificio1,eMeta,mContext);
            miPeso=loc3.distanceTo(E1)+caminoMasCorto.costo;
        }
        if (distancia2<distancia1&& distancia2<distancia3){// calcular
peso edificio2 + la distancia que se encuentre de este edificio
            caminoMasCorto= new aEstrella(edificio2,eMeta,mContext);
            miPeso=loc3.distanceTo(E2)+caminoMasCorto.costo;
        }
        if (distancia3<distancia1&& distancia3<distancia2){// calcular
peso edificio 3 + la distancia que se encuentre de este edificio
            caminoMasCorto= new aEstrella(edificio3,eMeta,mContext);
            miPeso=loc3.distanceTo(E3)+caminoMasCorto.costo;
        }

        return miPeso;
    }
```

Método **onConectionInfoAvaliable**; este método es provisto por la API WiFi Direct y permite revisar si existe alguna conexión establecida y realizar operaciones en base a la información obtenida de esta conexión.

```
@Override
    public void onConnectionInfoAvailable(final WifiP2pInfo info) {
            this.info = info;

        group_ownerIP=
info.groupOwnerAddress.getHostAddress().toString(); // obtengo  la IP del
group owner

        esGPOW=info.isGroupOwner;

        Log.d(TAG,"La IP del group owner es: "+group_ownerIP);

        /**Despues de la negociacion de grupo se asigna al group_owner
como el servidor*/

        if (info.groupFormed && info.isGroupOwner) {
           Log.e(TAG,"Es group owner");
           if(serverthread!=null){
                Log.e(TAG,"GROUP OWNER: Ya se encuentra ejecutando el
thread servidor");

            }else {
```

```
                serverthread = new ServerThread(mensajeFinal,
seleccionEdificio, miPeso, mContext,configCantidadConexiones, this);
                new Thread(serverthread).start();
            }

        } else if (info.groupFormed) {
            Log.e(TAG,"Es Cliente");
            ClientThread clientThread = new ClientThread(group_ownerIP,
mContext,usr_coordenadas,this);
            new Thread(clientThread).start();

        }
    }
```

Método **conectarPeersDisponibles**; este método es ejecutado dentro de un timertask y comienza con los intentos de conexión a la lista de dispositivos disponibles obtenida por el método *WiFiDirectBroadcastReceiver* para poder iniciar comunicación con estos.

```
  private void conectarPeersDisponibles() {

        miPeso = miPeso(seleccionEdificio, usr_coordenadas); // calcula
el peso actual antes de conectar a los peers

        List peersAux = new ArrayList();
        peersAux.addAll(peers);
        tamaño = peersAux.size();
        peersConectados = peersAux.size();
        setPeersConectados(peersAux.size());
        Log.d(TAG, "conectarPeersDisponibles: Mi peso es: " + miPeso);

        breakGroup();
        deletePersistentGroups();  // metodo que eliminara todos los
grupos persistentes recordados por el dipositivo

        Log.d(TAG, "conectarPeersDisponibles: Peers detectados: " +
tamaño);
        // Condicion if que revisa antes el array peers si es vacio y no
peermitir ejecutar la instruccion  enviar mensaje y reescanear
        if (peersAux.size() >= 1) {

            for (int i = 0; peersAux.size() > i; i++) { // conectamos a
cada peer disponible en la lista
                    synchronized (this) {
                        int count=0;
                        WifiP2pDevice device = (WifiP2pDevice)
peersAux.get(i);
                        if (device.deviceName.contains("ubb-messenger")
&& !device.deviceName.contains("GOW")&&
```

```
!device.deviceName.contains("RM")) { // filtrar por el nombre del
dispositivo y no este nombrado como GOW
                            do {
                                Log.d(TAG, "conectarPeersDisponibles:
Ocupado esperando");
                                count= count+1;
                                try {
                                    wait(1000);   // espero 1 segundo y
vuelvo a chequear si la conexion esta ocupada
                                } catch (InterruptedException e) {
                                    e.printStackTrace();
                                }
                                if (count==15){ // si el contador llega a
15 se cancela el intento de conexion
                                   Log.d(TAG, "conectarPeersDisponibles:
Se espero 15 segundos la conexion cancelando");
                                    mManager.cancelConnect(mChannel,null);
                                     setConectado(false);
                                }

                            } while (conectado);

                            WifiP2pConfig config = new WifiP2pConfig();
                            config.groupOwnerIntent = 15;  // reduzco
tendencia de que el dispositivo que selecciona conectar sea el group
owner
                            config.deviceAddress = device.deviceAddress;
                            config.wps.setup = WpsInfo.PBC; // tal vez no
necesario
                            connect(config);
                        } else {
                            restPeerConectado();// si se descarto lo
quito de la variable de control
                            Log.e(TAG, "conectarPeersDisponibles:
Dispositivo descartado" + device.deviceName.toString(), null);
                            }
                    }

            } //for

        } else{
                Toast.makeText(Main2Activity.this, "No existen peers
disponibles Continuando busqueda",
                        Toast.LENGTH_SHORT).show();
                midiscoverPeers();

            }
        }
```

Método **enableLocationUpdates**; este método permite obtener la ubicación del dispositivo cada 2 segundos por medio del GPS y las Google Play Services.

```
private void enableLocationUpdates() {
```

```
        locRequest = new LocationRequest();
        locRequest.setInterval(2000);
        locRequest.setFastestInterval(1000);
        locRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);


        LocationSettingsRequest locSettingsRequest =
                new LocationSettingsRequest.Builder()
                        .addLocationRequest(locRequest)
                        .build();

        PendingResult<LocationSettingsResult> result =
                LocationServices.SettingsApi.checkLocationSettings(
                        apiClient, locSettingsRequest);

        result.setResultCallback(new
ResultCallback<LocationSettingsResult>() {
            @Override
            public void onResult(LocationSettingsResult
locationSettingsResult) {
                final Status status = locationSettingsResult.getStatus();
                switch (status.getStatusCode()) {
                    case LocationSettingsStatusCodes.SUCCESS:

                        Log.i(LOGTAG, "Configuración correcta");
                        startLocationUpdates();

                        break;
                    case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:
                        try {
                            Log.i(LOGTAG, "Se requiere actuación del
usuario");

status.startResolutionForResult(Main2Activity.this,
PETICION_CONFIG_UBICACION);
                        } catch (IntentSender.SendIntentException e) {
                            btnActualizar.setChecked(false);
                            Log.i(LOGTAG, "Error al intentar solucionar
configuración de ubicación");
                        }

                        break;
                    case
LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
                        Log.i(LOGTAG, "No se puede cumplir la
configuración de ubicación necesaria");
                        btnActualizar.setChecked(false);
                        break;
                }
            }
        });
    }
```

13 Anexo: Diagrama de estados

Diagrama que muestra los estados que la aplicación pasa de manera autónoma durante su ciclo de ejecución.

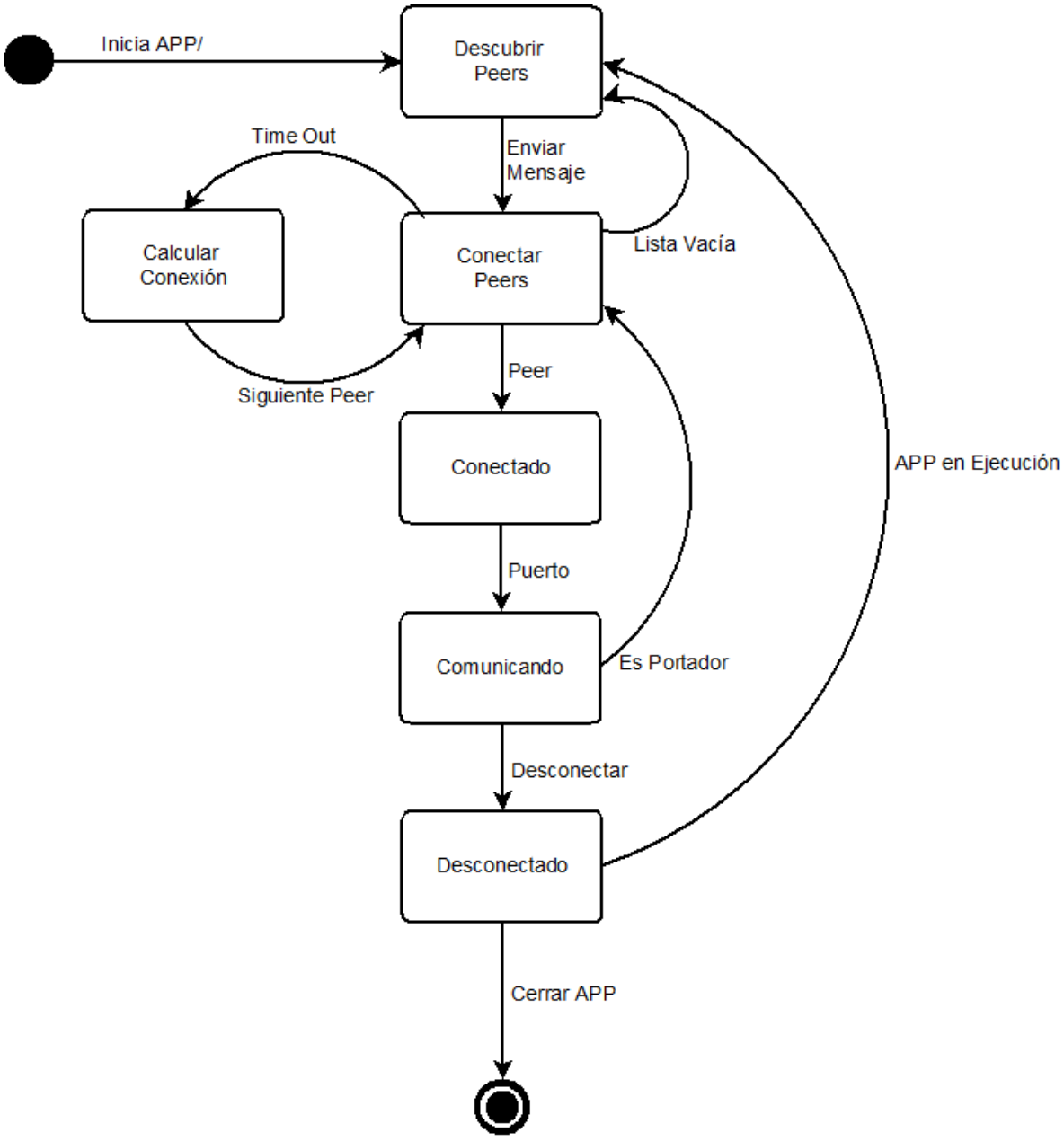

Figura 33: Diagrama de estados Aplicación/ WiFi Direct

14 Anexo: Wi-Fi Direct observaciones y recomendaciones del framework

Wi-Fi Direct es un protocolo de comunicación dispositivo a dispositivo que ha sido estandarizada y lanzada por la Wi-Fi Alliance, bajo este estándar teóricamente la comunicación entre múltiples grupos estaría habilitada. En principio sería posible realizar una red inalámbrica con múltiples grupos y comunicar los dispositivos entre si [36].

El Framework Wi-Fi direct disponible en Android aun siendo certificado y compilado bajo este estándar se encuentran ciertas limitaciones, como la imposibilidad de crear redes inalámbricas con múltiples grupos.

Utilizando este framework el desarrollador puede descubrir y conectarse a otros dispositivos que soportan Wi-Fi Direct permitiendo comunicarse a través de una conexión dispositivo a dispositivo. Su uso requiere interacción con el hardware Wi-Fi del dispositivo, para esto existe la clase *WifiP2pManager* la cual provee los métodos que permiten descubrir, conectar y desconectar a otros dispositivos. Dado a que interactúan con el hardware y que todos estos métodos son asíncronos y el *framework* utiliza *listeners* ubicados en la clase *WiFiDirectBroadcastReceiver* los cuales nos permiten saber y notificar el estado de una llamada a los métodos que interactúan con el hardware.

Dado que los métodos son asíncronos un correcto manejo de los *listeners* es fundamental para poder hacer uso exitoso de *Wi-Fi Direct* en Android. El método utilizado fue implementar banderas de control en los *listeners* ubicados en la clase *WiFiDirectBroadcastReceiver* las cuales controlan el acceso de los métodos que interactúan con el hardware. Por ejemplo: el método *discoverPeers()* es utilizado para registrarse en el canal permitiendo que otros dispositivos sean capaces de detectar el dispositivo actual, escanear y listar los dispositivos disponibles por medio del método *onPeersAvaliable()*. Este proceso se detiene bajo dos condiciones; se inició el proceso de conexión con un dispositivo disponible o se hizo una llamada explicita al método que detiene este proceso *stopPeerDiscovery()*. Si se llama al método *discoverPeers()* cuando ya existe un método *discoverPeers()* ejecutado anteriormente arrojara un error del tipo *Ocupado* bloqueando completamente el Wi-Fi para escanear dispositivos o incluso iniciar una conexión, de igual manera si se ha llamado al método *conect()* y este método no se maneja con *listeners* que impidan el llamado al mismo tiempo del método *discoverPeers()* causara

una colisión con el uso del hardware del Wi-Fi provocando que la aplicación detecte un estado de *Ocupado* quedando completamente bloqueado el uso del Wi-FI.

Dado que estos métodos son provistos y utilizados por todas las aplicaciones que utilicen el *framework Wi-Fi Direct* no son de acceso exclusivo para la aplicación desarrollada, inclusive si se desarrolla una aplicación que no llame a estos métodos de manera automática y repetitiva se recomienda el uso de banderas de control en los *listeners* para evitar tener problemas al intentar acceder al hardware habiendo una aplicación externa que esté haciendo uso al mismo tiempo de este, aun siendo una situación poco probable, tomar esta precaución podría evitar problemas en el proceso de desarrollo.

Otra limitación encontrada es que por razones de seguridad en una versión de stock de Android sin modificaciones es el usuario el que debe permitir la conexión con otro dispositivo limitando crear redes de manera automática por la aplicación. La única manera de saltarse esta limitación es por medio de privilegios root en la aplicación lo que implica modificar el S.O del dispositivo viéndose expuesta la seguridad del mismo.

15 Anexo: Contexto

Para aportar información del contexto a la aplicación se preparó una encuesta dirigida al estudiante, esta fue diseñada para obtener información relevante y lograr establecer un patrón respecto al comportamiento que podría presentar un alumno respecto a los lugares que visita de la Universidad.

Las preguntas realizadas fueron las siguientes:

- ¿Qué carrera estudia?
- ¿A qué facultad pertenece?
- ¿Cuántos edificios de la Universidad frecuenta (aulas, facultades, etc.)?
- Nombre el edificio que más frecuente de la Universidad.
- ¿Cuántas horas pasa al día en la Universidad?
- ¿Almuerza en dependencias de la Universidad?
- ¿En qué horarios almuerza en la Universidad?
- ¿En qué horario llega regularmente a la Universidad?
- ¿En qué horario se retira regularmente de la Universidad?

El universo total de alumnos que respondieron la encuesta fue de 48. Se pudo observar que el 80% de los que respondieron la encuesta nombraron a su facultad o escuela como uno de los edificios más visitados. De la pregunta que se pudo observar un patrón es la que solicita indicar la cantidad de edificios que frecuenta de la Universidad siendo la moda 3 edificios frecuentados. Por lo tanto, se decidió solicitar información del contexto al usuario en el primer inicio de la aplicación, donde se le pide ingresar sus datos y seleccionar 3 edificios que frecuente de la Universidad (Figura y figura).

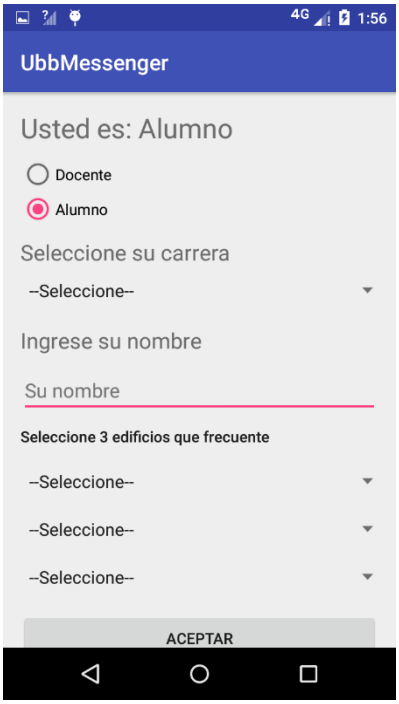

Figura 34: UbbMessengrer: "Primer inicio"

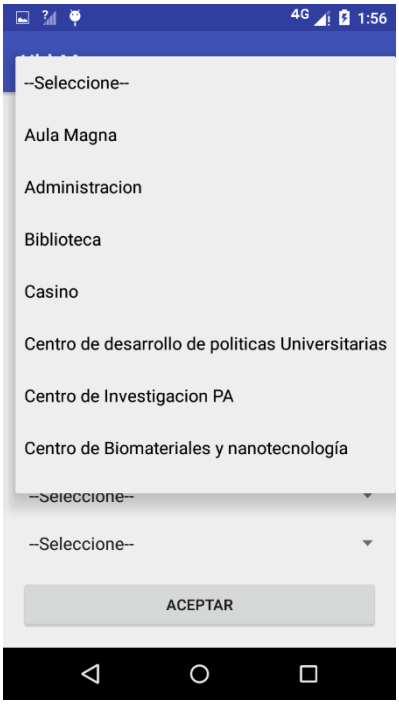

Figura 35: UbbMessengrer: " Primer inicio Lista edificios"

16 Anexo: Planificación inicial

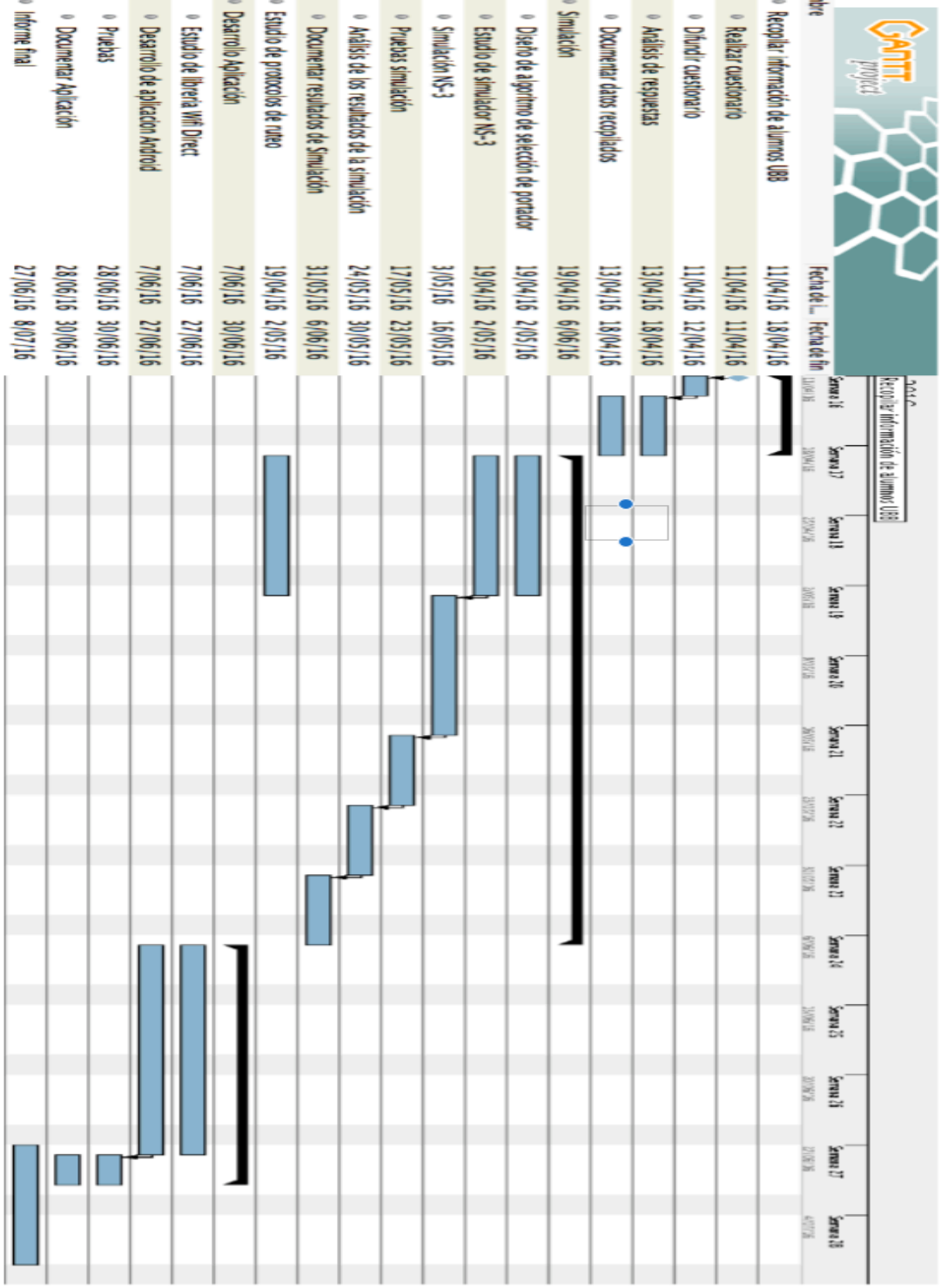

Figura 36: Planificación inicial