Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

Estudio sobre el uso de redes inalámbricas en la estimación de aforos

---

Alumno: Alejandro García Gómez

Tutor: Jesús M. Vegas Hernández

Agradecimientos

A mis padres, por apoyarme en todo momento durante estos años.

A mis amigos, por estar ahí en cualquier situación, ayudándome en los peores momentos y compartiendo con ellos este camino.

Resumen

El reciente aumento del número de dispositivos inalámbricos que lleva la gente en su día a día supone un nuevo reto para los algoritmos de rastreo y captura de paquetes, afectando directamente a aquellos programas dedicados a detectar la ocupación de una sala. Esto, sumado a la poca fiabilidad de sistemas para el cálculo del aforo ya existentes, como los sistemas basados en el dióxido de carbono o en cámaras de vídeo, generan la necesidad de diseñar nuevos métodos más fiables y adaptados a las nuevas políticas de privacidad. Por ello, en este proyecto, se diseñarán dos algoritmos que permitan identificar dispositivos Bluetooth y relacionar los dispositivos WiFi entre sí, algo que se ha tratado en otros trabajos y artículos pero sin llegar a una conclusión definitiva sobre ello. Con estos algoritmos se tendrá una aproximación más fiel a la realidad alejándose de la hipótesis de que cada persona lleva un único dispositivo, detectando así la ocupación de una forma más adecuada. Para probar el sistema y ajustar sus parámetros se realizará una prueba en un entorno real junto con un estudio de campo.

Abstract

The recent growth in the number of the personal wireless devices, represents a new challenge for packet capture and tracking algorithms, directly impacting room ocupation detecting softwares. This, added to the unreliability of existing ocuppancy detection systems, such as CO2 or videocams based programs, create the need to design new more reliable methods which are adapted to recent privacy policies. Therefore, the objective of this proyect is to design two algorithms which will allow to identify Bluetooth devices and link WiFi devices to each other. This algorithms will allow us to have a more realistic approach to a real world scenario dispelling the idea that each person carries a single device, making easier to detect room ocupation. In order to probe the system and adjust its parameters a test will be carried out in a real enviroment together with a survey.

Índice general

Agradecimientos **I**

Resumen **III**

Abstract **V**

Lista de figuras **XI**

Lista de tablas **XIII**

1. Introducción **1**

2. Estado del arte **3**

2.1. Trabajo relacionado 3

2.1.1. Sistemas basados en factores naturales 3

2.1.2. Sistemas basados en vídeo 4

2.1.3. Sistemas basados en redes inalámbricas 4

2.1.4. Sistemas de rastreo 4

2.2. Características utilizables para la identificación 5

2.2.1. Bluetooth y BLE 5

2.2.2. Direcciones MAC 5

2.2.3. SSID 7

2.2.4. RSSI 8
2.2.5. *Probe request* y *Probe response* 8
2.2.6. Huella WiFi 9
2.3. Tecnología utilizada 9
2.3.1. Raspberry Pi 3 9
2.3.2. Kali 10
2.3.3. Python 10
2.3.4. SQLite3 10
2.3.5. TFG de partida 11

3. Desarrollo **13**

3.1. Planificación 13
3.1.1. Coste 14
3.2. Desarrollo del sniff BLE 15
3.2.1. Análisis 15
3.2.2. Modelo de datos 17
3.2.3. Diseño del algoritmo 18
3.2.4. Implementación 19
3.3. Desarrollo del algoritmo WiFi 21
3.3.1. Análisis 21
3.3.2. Modelo de datos 23
3.3.3. Diseño 24
3.3.4. Implementación 24
3.4. Script de Ejecución 26
3.5. Prueba en un escenario real 27
3.5.1. Objetivos 27
3.5.2. Diseño del experimento 27

3.5.3. Diseño de la encuesta complementaria 28
3.5.4. Ejecución del experimento 29

4. Resultados **31**
4.1. Resultados del experimento 31
4.2. Resultados de la encuesta 31
4.2.1. Encuesta del experimento 32
4.2.2. Encuesta pública 32

5. Discusión **33**

6. Conclusiones **37**

A. Manual de usuario **39**
A.1. Script de ejecución 39

B. Preguntas de la Encuesta **41**

C. Contenidos del .ZIP **43**

Bibliografía **45**

Índice de figuras

2.1. Formato de una dirección MAC 6
2.2. Tipos de direcciones MAC en Bluetooth 7

3.1. Diagrama de Grantt de este proyecto 14
3.2. Diagrama de flujo del algoritmo BLE 19
3.3. Diagrama de flujo del algoritmo WiFi 25

5.1. Resultados de la encuesta en cuanto a Bluetooth 33
5.2. Personas detectadas según los parámetros del algoritmo 34

Índice de Tablas

3.1. Duración de cada fase del proyecto 14
3.2. Coste de los materiales utilizados 15
3.3. Requisito Bluetooth Nº1 16
3.4. Requisito Bluetooth Nº2 16
3.5. Requisito Bluetooth Nº3 16
3.6. Requisito Bluetooth Nº4 16
3.7. Requisito Bluetooth Nº5 17
3.8. Requisito Bluetooth Nº6 17
3.9. Requisito Bluetooth Nº7 17
3.10. Requisito WiFi Nº1 21
3.11. Requisito WiFi Nº2 21
3.12. Requisito WiFi Nº3 21
3.13. Requisito WiFi Nº4 22
3.14. Requisito WiFi Nº5 22
3.15. Requisito WiFi Nº6 22
3.16. Requisito WiFi Nº7 22
3.17. Requisito WiFi Nº8 23
3.18. Requisito WiFi Nº9 23

Capítulo 1

Introducción

El reciente aumento del número de dispositivos inalámbricos y su creciente popularidad ha supuesto un incremento en la cantidad de dispositivos que lleva encima la gente en su día a día, como teléfonos móviles, pulseras y relojes inteligentes, cascos inalámbricos o incluso sensores biomédicos como parches para controlar la glucosa en personas diabéticas. Todos estos dispositivos utilizan redes inalámbricas para conectarse a internet o entre sí, siendo las más frecuentes WiFi y Bluetooth, respectivamente.

Por ello la pregunta que se va a intentar responder en este proyecto es si se puede calcular el aforo de una sala a partir del número de dispositivos que haya en esta, siendo capaces de relacionar los dispositivos entre sí para tener la estimación tanto de dispositivos como de personas. Además de intentar dar una solución para este problema, se realizará un experimento en un entorno real para comprobar la eficacia de esta solución, y un estudio de campo para poder ajustar más detalladamente parámetros como el número medio de dispositivos que suele llevar un usuario normal.

Existen otro tipo de soluciones y métodos para calcular de forma automática el aforo de una sala, siendo los más comunes sensores de CO2 y cámaras de vídeo, aunque con varios problemas como la distribución del CO2 por la sala o la identificación de las personas, respectivamente. Por ello, si se logra dar una solución válida en este proyecto, supondrá un avance en la utilización de este tipo de tecnología y método en la estimación de la ocupación, sin poner en peligro la privacidad de las personas ya que no se interactúa con los dispositivos en ningún momento. Además, puede ser útil para varias situaciones tanto en el ámbito personal como empresarial, por ejemplo para comprobar qué salas son las más ocupadas de una empresa e invertir más en su comodidad o tener una valor constantemente actualizado del aforo para negocios pequeños, siendo una solución barata que podrían permitirse.

El principal objetivo de este proyecto es dar una solución válida al problema de la relación de dispositivos entre sí, ya que se ha llegado a varias soluciones para resolver el problema de detectar el número de dispositivos que hay en una sala pero no se ha llegado a profundizar en la relación que tienen esos dispositivos, dando en la mayoría de ocasiones soluciones con

la hipótesis de que cada persona lleva un único dispositivo. Una vez se consiga diseñar un algoritmo que permita esa relación, se diseñará un estudio de campo y un experimento para probar su eficacia en un entorno real ajustando parámetros del algoritmo, basándose tanto en nuestro estudio como en estudios ajenos, hechos por compañías como *Cisco*.

La estructura que seguirá en este proyecto es la siguiente. En el *Capítulo 2* se tratarán los trabajos previos y relacionados que hay para resolver este problema además de una descripción del sistema propuesto y de la tecnología utilizada. En el *Capítulo 3* se detallará la solución y el sistema implementado completo así como el diseño y desarrollo del experimento y del estudio de campo. En el *Capítulo 4* se expondrán los resultados tanto del experimento realizado como de la encuesta, pasando a analizarlos y discutirlos en el *Capítulo 5*. Por último, en el *Capítulo 6* se expondrán las conclusiones obtenidas en este proyecto.

Capítulo 2

Estado del arte

En este capítulo se expondrán los trabajos y métodos existentes para detectar la ocupación de una sala, así como los trabajos que han servido de referencia para realizar este proyecto. Se expondrá también el sistema propuesto junto a una explicación de los conceptos más importantes en los que se basa el trabajo. Además, se explicará también la tecnología y programas utilizados para el desarrollo del sistema.

2.1. Trabajo relacionado

Existen varios sistemas para detectar cuántas personas se encuentran en una sala, el principal objetivo de este trabajo, y varios programas y algoritmos para rastrear dispositivos bien sea a través de Bluetooth o de WiFi, algo que no se abarca aquí pero que ha servido como influencia para este proyecto.

2.1.1. Sistemas basados en factores naturales

El sistema más utilizado con este funcionamiento es el que mide el volumen de CO2 que hay en la sala, a partir del cual se calcula el aforo, conservando totalmente la privacidad de estas personas ya que no se obtiene ningún dato sobre esas personas [1].
La contra de este sistema es que la distribución del CO2 por la sala no es totalmente uniforme, ya que depende de la distribución de las personas y de la ventilación que haya, por lo que el valor medido puede variar fácilmente haciendo que el aforo sea muy distante al real.
Para buscar una solución más precisa se ha integrado a este sistema la detección de otras medidas como la humedad, acercándose así al valor real pero sin ser suficientemente preciso como para que sea fiable.

2.1.2. Sistemas basados en vídeo

El principal método dentro de estos sistemas es la detección del aforo a través cámaras de vídeo, mucho más preciso que el anterior pero con el problema de la privacidad, ya que identifica a todas las personas de la sala, por lo que previamente deben haber dado su consentimiento para ello. Para evitar este problema y prevenir que se reconozcan las caras se utilizan cámaras de muy baja resolución o se colocan a una altura suficientemente baja para que no se vean las caras. Normalmente se utiliza este sistema combinado con el de CO2 para aumentar la precisión en salas con baja iluminación o mucha ventilación, casos en los que ambos métodos por separado son menos fiables [1].

2.1.3. Sistemas basados en redes inalámbricas

Este tipo de sistemas calculan el aforo tanto en interiores como exteriores a partir de la información obtenida de los paquetes WiFi y Bluetooth que se transmiten. Se analizan los paquetes de descubrimiento de red, especialmente los paquetes *probe request* (en la siguiente sección se explican más detalladamente), para obtener los valores RSSI (Receive Signal Strenght Indicator) y CSI (Channel State Information), con los que a partir de diferentes algoritmos y modelos se obtienen estimaciones cercanas a la realidad. El problema con este método es que no escala bien con muchas personas, limitando así las pruebas y experimentos que se han hecho a un máximo de 10 personas [1, 2].

Otros sistemas se basan en obtener las direcciones MAC de los dispositivos y así hacer una estimación más directa del número de dispositivos alrededor, que combinado con el RSSI del método anterior obtienen una aproximación de los dispositivos en un umbral más cercano o más lejano. Uno de los problemas de este método, como se explicará en la siguiente sección, es la aleatoriedad de la dirección MAC, la cuál hace que si no se detecta correctamente se obtengan más dispositivos de los que realmente hay en la sala [3].

Otra contra que surge de ambos tipos de sistemas es que están hechos para detectar dispositivos, no personas. Con el uso cada vez más común de dispositivos inalámbricos, como auriculares o pulseras inteligentes, se detectarían muchos más dispositivos de las personas que realmente habría en la sala, por lo que no se puede establecer una relación uno a uno entre dispositivo y persona, siendo necesario buscar nuevos métodos dentro de esta categoría con los que tener una aproximación más real [1, 4].

2.1.4. Sistemas de rastreo

Este tipo de sistemas han servido de referencia, aunque difieran de lo que se hace en este proyecto, por su conocimiento de las direcciones MAC tanto en Bluetooth como en WiFi, así como por la información que se obtiene de los paquetes de descubrimiento de ambas redes. Uno de los conceptos en los que se basan este tipo de sistemas es lo que se conoce como "*huella dactilar*", explicado en profundidad en la siguiente sección. Al recolectar todas las redes conocidas y asociarlas con su dirección MAC, se puede establecer un identificador único

con el que seguir la pista al dispositivo. Teniendo varios puntos de acceso usados para rastreo, se puede trazar la trayectoria del usuario, ver por donde ha pasado y cuándo, pudiendo establecer un patrón solamente con la información de su huella [3, 5, 6].

2.2. Características utilizables para la identificación

El sistema propuesto para este proyecto es una estimación del aforo de una sala a partir del rastreo de paquetes WiFi y Bluetooth que emitan sus dispositivos, con una Raspberry Pi, siendo capaces de relacionar los dispositivos WiFi entre sí acercándonos a una aproximación real de las personas que se encuentren en el interior. Este sistema irá acompañado de un estudio de campo sobre los dispositivos que lleva encima la gente, lo que permitirá estimar la media de dispositivos que tiene una persona o las redes que lleva conectadas, y de un experimento realizado en la Escuela de Ingeniería Informática donde se recogerán datos y se probarán los algoritmos desarrollados sobre un grupo de usuarios controlados.
Para sentar unas bases sobre estas estimaciones tanto por WiFi como por Bluetooth, se explicará a continuación la diferencia entre los dos estándares Bluetooth que existen, así como el funcionamiento de los paquetes de descubrimiento y la información que contienen.

2.2.1. Bluetooth y BLE

Comenzando por este protocolo, existe una variante creada a partir de la actualización de Bluetooth 4.0 llamada *Bluetooth Low Energy*, conocida como BLE. Este nuevo subprotocolo se diseñó para ser utilizado en dispositivos menos potentes y más pequeños, siendo un chip de menor tamaño que el de Bluetooth normal. La mayoría de teléfonos móviles y ordenadores incorporan ambos protocolos, que se van alternando dependiendo del dispositivo al que se haya conectado [7].
Ambos protocolos funcionan en la misma banda de frecuencia, 2.4 GHz, pero con la diferencia de que BLE permanece en suspensión hasta que activamente se inicia una conexión y se transmiten datos, a diferencia de Bluetooth que mientras esté activado está en funcionamiento constante. Por esta diferencia, Bluetooth se utiliza para conexiones duraderas en la que se va a intercambiar una gran cantidad de datos y BLE para dispositivos que no van a transmitir tantos datos.
En este proyecto se trabajará con BLE, ya que consume menos energía y se pueden detectar fácilmente los dispositivos con este protocolo.

2.2.2. Direcciones MAC

Una dirección MAC, también llamado dirección física, es un conjunto de 48 bits que se utiliza como un identificador único del dispositivo en la capa de enlace del modelo de red OSI.
Aunque este identificador supone lo mismo tanto para Bluetooth como WiFi, el significado

de sus bytes varía:

- **WiFi**: Los 3 primeros bytes, llamados *OUI,* identifican al fabricante de la interfaz de red, mientras que los 3 últimos bytes identifican al dispositivo, siendo la parte que se altera si se trata de una MAC aleatoria [8].

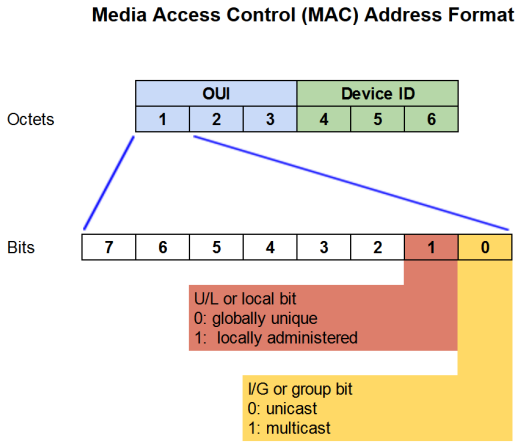

Figura 2.1: Formato de una dirección MAC

Todos estos fabricantes están registrados en una lista, dentro de *IEEE*, en la que cada uno tiene asignados un conjunto de bytes propios.
Por tanto, gracias a estos 3 bytes, se puede tener un pequeño conocimiento sobre el dispositivo WiFi del que se está obteniendo información y resolver uno de los problemas que se plantearán más adelante, la aleatoriedad de las direcciones MAC.

- **Bluetooth**: Dependiendo de si la dirección es pública o aleatoria, los significados de los bytes varían. Si la MAC es pública los bits tienen el mismo significado que en las direcciones WiFi, pero si la MAC es aleatoria, todos sus bits serán generados al azar excepto los 2 más a la derecha, que nos indicarán si es aleatoria pero estática (no varía a lo largo del tiempo) o si es privada (la dirección cambia cada cierto tiempo) [9].
En la figura 2.2 se pueden ver los distintos tipos de direcciones que hay, de los cuales para este proyecto solo nos centraremos en la capa superior, siendo capaces de diferenciar MACs públicas de aleatorias.

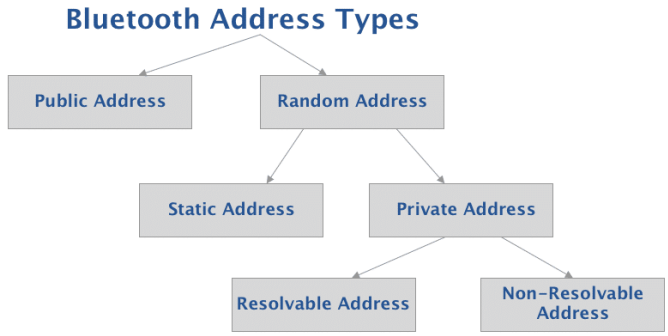

Figura 2.2: Tipos de direcciones MAC en Bluetooth

Una vez hecha la distinción entre las dos redes con las que se va a trabajar, veamos uno de los principales problemas que surgen de estas direcciones, la aleatoriedad.

MAC Aleatoria

Las MACs aleatorias surgieron de la necesidad de mantener la privacidad de los usuarios. Antes de que fuese un estándar común, los dispositivos tenían una única MAC durante toda su vida útil, por lo que era muy fácil rastrear estos dispositivos [5].
Con estos nuevos algoritmos se complica mucho más seguir el rastro a un dispositivo, ya que se tiene constancia solamente del fabricante del dispositivo. Por otro lado, para el estudio del aforo que se quiere hacer en este proyecto, basta con conocer solamente el *OUI* de la MAC, por lo que teniendo estos 3 bytes se puede saber si el mismo dispositivo sigue en la sala o no, ya que son bytes fijos que no van a cambiar con el resto de bits.
Esta aleatoriedad de las direcciones físicas está presente tanto en Bluetooth como en WiFi, con la diferencia del *OUI* explicada en el apartado anterior.

2.2.3. SSID

El SSID (*Service Set Identifier*) es una secuencia de 32 caracteres que se utiliza como identificador de una red, bien sea Bluetooth o Wifi. Todos los puntos de acceso a los que se intente conectar un dispositivo tienen que tener un nombre de red, por ejemplo `eduroam`, que es el SSID de la red Wifi de la universidad. Cabe destacar que mientras que la MAC es un identificador único del dispositivo (asociado a la capa de enlace), el SSID no lo es, siendo un identificador usado para ocultar la MAC del punto de acceso [10].
Este identificador se utiliza también en la búsqueda de redes si se está haciendo una búsqueda activa, es decir, si el dispositivo busca las redes conocidas transmitiendo sus SSIDs [2].

2.2.4. RSSI

El RSSI (*Recive Signal Strenght Indicator*) es una medida que indica como de buena es la señal que le llega a un dispositivo desde un punto de acceso. Su valor siempre es negativo, donde más próximo a 0 significa que llega una mayor señal y, por tanto, indica que ese dispositivo se encuentra más próximo al punto de acceso [11].
Esta medida viene dada como un índice relativo, no como decibelios, aunque se puede aproximar el valor del RSSI a un equivalente en dBm.
Además de indicar la calidad de la señal del punto de acceso, se puede utilizar también como un indicador para ver la distancia de los dispositivos al punto de acceso. Aunque influyen más factores como la calidad de la interfaz de red, los objetos que haya entre ambos y la calidad del router WiFi, su impacto en cuanto a la distancia va a ser muy leve por lo que se puede realizar la estimación igualmente. Esta distancia no será en valores absolutos si no relativos, estableciendo unos umbrales para distancias pequeñas, medianas o grandes [12].

2.2.5. *Probe request* y *Probe response*

Los paquetes de tipo *probe request* son los paquetes que envían los dispositivos WiFi a los puntos de acceso (AP) para obtener información sobre estos AP y establecer una conexión con ellos. Los paquetes *probe response* son las respuestas de estos AP hacia los dispositivos. Este tipo de paquetes se intercambian cuando el dispositivo realiza búsqueda activa, es decir, cuando busca explícitamente las redes que ya tiene guardadas, transmitiendo en cada paquete el SSID de esas redes [1].
En este proyecto nos interesa quedarnos con los paquetes *probe request* que son los que se utilizarán para detectar los dispositivos.

Estos mensajes de descubrimiento se envían por todos los canales WiFi, los cuales se transmiten en la banda de frecuencias de 2.4 GHz yendo desde 2.401 hasta 2.483 MHz. Los envíos en los distintos canales van rotando entre ellos, permitiendo así que cualquier AP vea al menos un mensaje de difusión en alguno de los canales que detecte.
Cada mensaje esta compuesto de varia información transmitida en texto plano, de la cuál interesa para este trabajo los 3 campos explicados anteriormente, la dirección MAC, el SSID conocido que transmita y el RSSI. El campo del SSID puede no estar presente en todos los mensajes, ya que dependiendo del fabricante de la interfaz de red y del sistema operativo puede omitirse ese campo, transmitir solamente los SSIDs de redes públicas o los SSIDs de redes públicas y privadas. Por una parte esto se debe a la brecha de seguridad que supone transmitir todas las SSIDs privadas a las que se ha conectado, ya sea por ser fácilmente rastreable o por la posibilidad de recibir ataques de tipo *"man in the middle"*, por lo que cada vez se transmite menos información privada que pueda suponer un riesgo para el usuario, incrementado además por el constante cambio de las políticas de privacidad.
En cuando a Bluetooth, existen también unos paquetes de descubrimiento muy similares a estos *probe request* llamados *inquiry packets*, los cuales hacen difusiones por todos los canales disponibles. De forma similar a los paquetes WiFi, nos interesa obtener de estos paquetes la dirección MAC y el RSSI. Las redes conocidas se enviaban en este tipo de paquetes pero tras el riesgo que suponían quitaron este campo de la comunicación.

2.2.6. Huella WiFi

Se le denomina huella dactilar del dispositivo al conjunto de SSIDs conocidos que se envían en la búsqueda de nuevas redes.
La mayoría de dispositivos móviles realizan una búsqueda activa de nuevas redes, estando presente en los sistemas operativos más utilizados hoy en día (Android, Windows, iOS, etc). Esto quiere decir que los dispositivos envían paquetes de tipo *probe request* cada cierto tiempo buscando las redes que ya conocen, por lo que se tiene un conjunto de paquetes de difusión con redes conocidas distintas que se envían por todos los canales.
Capturando todos estos paquetes y viendo los SSIDs que contienen se puede establecer entonces una especie de identificador para cada dispositivo, combinado con la dirección MAC, que permitirá detectar cada dispositivo aislado y establecer patrones gracias a ello.
Esta huella es lo que se utilizará en el algoritmo diseñado para detectar si dos dispositivos son de una misma persona.
Al ser necesario conocer las redes conocidas que transmite el dispositivo, este tipo de identificador está presente únicamente en WiFi [3].

2.3. Tecnología utilizada

En esta sección se detallará la tecnología, tanto hardware como software, que se ha utilizado para desarrollar este trabajo.

2.3.1. Raspberry Pi 3

Comenzando por la parte hardware, se ha utilizado una Raspberry Pi 3 modelo B+ para la captura de los paquetes de ambas redes y el desarrollo de los algoritmos. Este modelo incluye tanto conexión BLE como conexión de red inalámbrica, por lo que no ha sido necesario utilizar otro ordenador o dispositivo para realizar la captura. Las principales características de este dispositivo son [13]:

- Procesador Broadcom BCM2837B0 con arquitectura ARM8 de 64-bit, con una frecuencia de 1,4 GHz.
- Puerto de red Gigabit Ethernet sobre USB2.0.
- Conectividad inalámbrica en bandas de 2,4GHz y 5GHz.
- WiFi IEEE 802.11b/g/n/ac.
- Bluetooth 4.2, BLE.
- Puertos HDMI, USB, Micro SD, Micro USB.

2.3.2. Kali

Como sistema operativo para la Raspberry se ha utilizado Kali Linux, un sistema operativo especializado en ciberseguridad que cuenta con multitud de herramientas para ello [14]. Entre las distintas herramientas, las que se han utilizado han sido Scapy, Aircrack-ng y Wireshark, todas ellas de código abierto.

- Scapy: Es un programa hecho en Python para enviar, capturar y crear paquetes de red, como TCP, UDP, Bluetooth, etc. Esta herramienta es la que se ha utilizado para capturar el tráfico de paquetes por la red, concretamente los paquetes de descubrimiento *probe request* para WiFi y los *inquiry packets* para Bluetooth. Además permite usarse como una librería en programas Python, por lo que se ha utilizado también en el desarrollo de la aplicación [15].
- Aircrack-ng: Es un conjunto de herramientas utilizado para seguridad en redes WiFi. De este conjunto se ha usado la herramienta `airmon-ng` para poner la interfaz inalámbrica de red en modo monitor, es decir, para que a través de la interfaz se pueda ver que paquetes WiFi llegan hasta ella [16, 17].
- Wireshark: Es una aplicación utilizada para analizar el tráfico de una red, la cuál captura toda la información posible de cada paquete. Esta herramienta se ha utilizado para escanear la información de los paquetes de descubrimiento Bluetooth y ver su estructura, permitiendo así tener una base para después poder capturarlos manualmente, y para ver que información transmiten los paquetes de descubrimiento WiFi, viendo que además de su MAC pueden transmitir los SSIDs que tienen guardados, lo que permitirá establecer una conexión entre dispositivos como se verá más adelante [18].

2.3.3. Python

Para todo el desarrollo del proyecto se ha usado el lenguaje de programación Python.
Python es un lenguaje interpretado, dinámico y multiplataforma, por lo que se puede utilizar fácilmente en Kali Linux, donde tiene soporte nativo. Además es un lenguaje de código abierto y multiparadigma, que soporta programación funcional, programación imperativa u orientada a objetos [19].
La versión utilizada ha sido Python3, la última versión que existe del lenguaje y en la cual hay multitud de librerías, entre ellas `Scapy`, que como se comentaba anteriormente se ha utilizado para la captura de paquetes WiFi, y `Bluepy`, para capturar los paquetes Bluetooth [15, 20].

2.3.4. SQLite3

Para guardar toda la información capturada que resulte útil se ha utilizado la base de datos SQLite3, escrita en `C` para el motor de bases de datos SQL. SQLite se encuentra presente en la mayoría de dispositivos de forma nativa, tanto dispositivos móviles como ordenadores,

por lo que resulta muy útil y sencillo trabajar con ello.
Es una base de datos relacional, es decir, que los datos se recogen en tablas y, dentro de estas tablas, se agrupan por filas, por tanto se tendrán los datos relacionados en tuplas mientras que por columnas tendremos los atributos. De esta forma resulta muy sencillo relacionar los datos entre las distintas tablas, hacer búsquedas sobre ellas y garantizar las propiedades ACID: Atomicidad, Consistencia, Aislamiento(Isolation) y Durabilidad [21].
Se ha decidido utilizar esta base de datos ya que la aplicación base de este proyecto guardaba sus datos en SQLite3 y, puesto que se van a guardar datos similares y se van a relacionar distintas tablas con las que ya había creadas, es una buena opción continuar con este motor.

2.3.5. TFG de partida

La base de esta aplicación y algoritmo ha sido la aplicación desarrollada durante el año 2020 por Aitor Ojeda Bilbao en su TFG, la cual consistía en saber cuántos dispositivos se encuentran en una habitación mediante paquetes WiFi. Partiendo de esta aplicación y su investigación se ha continuado el desarrollo sobre el algoritmo para redes WiFi, junto a la tecnología y herramientas que utilizaba en su aplicación como Scapy o SQLite3 [4].

Capítulo 3

Desarrollo

En este capítulo se explicarán los distintos algoritmos desarrollados así como el experimento que se ha llevado a cabo en un entorno real y la encuesta complementaria. Se ha hecho una distinción entre el desarrollo en la parte Bluetooth y en la parte WiFi para facilitar la lectura y comprensión de los dos algoritmos, por lo que los contenidos de *diseño* o *implementación* se encontrarán en las dos secciones.

3.1. Planificación

El enfoque que se le ha dado a este proyecto es el de desarrollo en cascada, en el que se tendrán hitos que marcarán la finalización de esa parte del proyecto, hasta llegar al análisis de los resultados del experimento y la redacción de este documento.

El proyecto tiene como fecha de inicio el día 17 de febrero de 2021, durando un total de 4 meses, hasta el día 21 de junio. Como se puede ver en la tabla 3.1, se han marcado unas fechas de hitos asequibles, basándose así en el enfoque más probable, teniendo fechas que no sean extremas de cumplir y con un pequeño margen por los contratiempos que puedan surgir durante el desarrollo de este.

	Fase del desarrollo	Inicio	Fin	Duración (días)
1	Planificación	17/02/21	18/02/21	2
2	Búsqueda de información	17/02/21	02/03/21	10
3	Desarrollo de los prototipos	03/03/21	18/03/21	12
4	Desarrollo del algoritmo BLE	19/03/21	08/04/21	15
5	Desarrollo del algoritmo WiFi	09/04/21	28/04/21	14
6	Preparación del experimento y encuesta	29/04/21	30/04/21	2
7	Desarrollo del experimento	03/05/21	03/05/21	1
8	Análisis de resultados	07/06/21	11/06/21	5
9	Desarrollo de la memoria	04/05/21	21/06/21	35

Cuadro 3.1: Duración de cada fase del proyecto

Dentro de las fases mencionadas en el cuadro anterior se encuentran también el análisis de requisitos, el diseño de los algoritmos o la implementación de estos, pero englobadas dentro de tareas más generales para tener una mejor estimación y percepción de la duración de cada parte.
En la figura 3.1 se puede encontrar el diagrama de Grantt de este proyecto, viendo más gráficamente las fechas de las tareas. Dejando de lado la redacción de la memoria, se puede ver que lo que más tiempo necesita es el desarrollo de los dos algoritmos. El análisis de resultados se dejará para el final para así poder obtener más resultados en la encuesta que sirvan para probar los algoritmos con distintos parámetros sobre los resultados del experimento. Además de marcar el desarrollo de la memoria, se ha ido haciendo a la par que se hacía el resto del proyecto, pero al final del proyecto se ha trabajado en mayor medida en la memoria, por lo que viene marcada también como tarea.

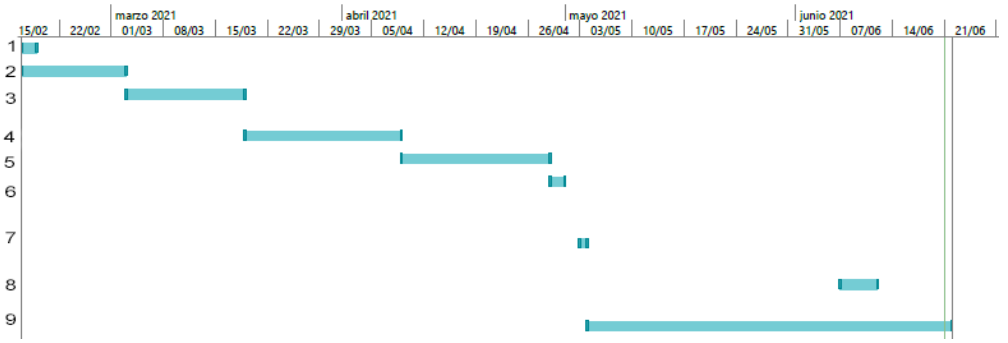

Figura 3.1: Diagrama de Grantt de este proyecto

3.1.1. Coste

El coste de desarrollar estos algoritmos y de realizar el experimento será muy pequeño, ya que los algoritmos están programados con software de código libre y diseñados para su

uso en una Raspberry Pi, cuyo coste no supera los 50€ dependiendo del modelo. Además, el experimento se realizará dentro de la escuela de ingeniería informática con alumnos del centro, por lo que no será necesario alquilar ninguna sala donde realizarse.

Objeto	**Precio(€)**
Raspberry Pi 3 B+ (con tarjeta de memoria)	48
Teclado	7
Ratón	5
Monitor	70
Total	**130**

Cuadro 3.2: Coste de los materiales utilizados

A pesar del coste que se muestra en la tabla 3.2, el coste real del proyecto ha sido inferior, ya que se trabajó en un laboratorio de la escuela que disponía de todo lo mencionado, además de material extra que no fue necesario en este proyecto.

3.2. Desarrollo del sniff BLE

En esta sección se detallará todo el desarrollo del algoritmo que se ha implementado para hacer la captura de paquetes BLE y, por tanto, detectar los dispositivos Bluetooth. Se explicará tanto el diseño del algoritmo como las herramientas utilizadas, los datos que se guardarán en la base de datos y los fundamentos que se han utilizado para diseñar el algoritmo.

3.2.1. Análisis

En esta sección se detallarán los requisitos de software necesarios para este algoritmo. Un requisito de sofware se define como una descripción de las funcionalidades del sistema que se va a desarrollar, haciendo distinción entre dos tipos de requisitos, funcionales y no funcionales [22]:

- **Requisitos Funcionales**: Son los servicios o funcionalidades que el sistema/algoritmo debe proporcionar.
- **Requisitos No Funcionales**: Son las cualidades que deben tener las funcionalidades del sistema, relacionado con la fiabilidad, la capacidad, el tiempo de respuesta o la forma de guardar los datos.

En las siguientes tablas se pueden encontrar los requisitos tanto funcionales como no funcionales que implementará el algoritmo Bluetooth:

RF1 - Captura de paquetes	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de capturar los paquetes necesarios para el proyecto, obteniendo solo los de tipo *probe request*.
Motivo	Esta funcionalidad es principal para relacionar los dispositivos, por lo que es esencial implementarla.

Cuadro 3.3: Requisito Bluetooth Nº1

RF2 - Almacenamiento de datos	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de guardar los datos de los dispositivos y SSIDs capturados.
Motivo	Esta funcionalidad servirá para relacionar los dispositivos manteniéndolos en la base de datos, por lo que es necesaria implementarla.

Cuadro 3.4: Requisito Bluetooth Nº2

RF3 - Actualización de los datos	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de actualizar los datos de la base de datos con los nuevos datos que se capturen.
Motivo	Deben actualizarse los registros de la DB para tener registrado las nuevas MACs aleatorias o el tiempo de vida del registro.

Cuadro 3.5: Requisito Bluetooth Nº3

RF4 - Detección del tipo de MAC	
Tipo de requisito	Funcional
Prioridad	Media
Descripción	El algoritmo debe ser capaz de diferenciar si la MAC del dispositivo es aleatoria o pública.
Motivo	Esta funcionalidad resulta útil para tener constancia de los fabricantes de los dispositivos y de cuántas direcciones hay de cada tipo.

Cuadro 3.6: Requisito Bluetooth Nº4

RF5 - Permanencia del dispositivo	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de detectar si un dispositivo sigue estando en la sala o no.
Motivo	Esta funcionalidad será principal para calcular correctamente los dispositivos que hay.

Cuadro 3.7: Requisito Bluetooth Nº5

RNF1 - Almacenamiento en tiempo real	
Tipo de requisito	No Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de guardar en tiempo real la información que capture en la base de datos.
Motivo	Es necesario que se guarden los datos en tiempo real para tener los dispositivos guardados según se capturen.

Cuadro 3.8: Requisito Bluetooth Nº6

RNF2 - Actualización en tiempo real	
Tipo de requisito	No Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de mantener actualizados los datos de la base de datos constantemente en tiempo real.
Motivo	Es necesario que se actualicen los datos en tiempo real para tener constancia de los dispositivos que hay en la sala actualmente.

Cuadro 3.9: Requisito Bluetooth Nº7

3.2.2. Modelo de datos

Para este algoritmo se ha utilizado una única tabla de la basde de datos llamada *Data-Bluetooth*, en la cuál se guardarán los datos recogidos de cada dirección MAC. Esta tabla se creará en la base de datos cada vez que se inicie la captura de paquetes de BLE y, en caso de que estuviese creada, se vaciará al inicio de la captura, por lo que no se conservarán los datos de los dispositivos en la base de datos a lo largo del tiempo. Los datos que se guardarán en esta tabla son:

- **mac**: Dirección MAC del dispositivo del que se ha capturado el paquete.
- **mac_type**: Nombre del fabricante de la interfaz Bluetooth o *Random* en caso de que se una MAC aleatoria y no se conozca el fabricante.

- **rssi**: Intensidad de la señal del dispositivo.
- **date**: Fecha y hora a la que se ha capturado el paquete.
- **TTL**: Contador de tiempo de vida del paquete, inicialmente establecido en 1.

Todos estos atributos de la tabla son de tipo *text*, lo que facilita guardarlos en la base de datos y las comparaciones entre ellos.

3.2.3. Diseño del algoritmo

En esta sección se detallará el algoritmo con el que se obtienen los distintos paquetes y dispositivos Bluetooth, explicando la hipótesis para la detección de dispositivos y el algoritmo creado.

Hipótesis de captura

La hipótesis en la que se basará este algoritmo es una simplificación del número de dispositivos que puede llevar una persona, que se establecerá en que cada dispositivo BLE es una persona. Por tanto, para cada MAC distinta que tengamos en la base de datos habrá una persona en la sala.

Algoritmo

El primer paso que ejecuta este algoritmo es iniciar el rastreo de los paquetes con un bucle que capture cada 3 segundos y, una vez se analizan todos esos paquetes, volver a capturar. Estos paquetes son todos paquetes de descubrimiento de difusión, por lo que no es necesario aplicar un filtro para quedarse solo con esos paquetes.
Para el análisis de cada paquete lo primero que se comprueba es si la dirección MAC se encuentra en la base de datos, donde si el identificador ya está guardado se pasa al siguiente paquete capturado. Tras esto se obtiene si la dirección MAC es pública o privada, gracias a una función de la librería utilizada, y, en caso de ser pública, se comprueba si es una dirección registrada en *IEEE*. En caso de estar registrada se guardará en la base de datos el fabricante de ese chip, si no lo está pero es una dirección pública se guardará la cadena "*public, unknown manufacter*" y si es una MAC aleatoria se guardará la cadena "*random*". Como se comentaba en la sección anterior, se obtiene también la hora a la que se está analizando el paquete y el rssi del dispositivo, que se incluyen en la base de datos en una única tupla junto a los datos de la MAC. Tras este paso se vuelve al inicio del bucle y se empiezan a capturar de nuevo paquetes.

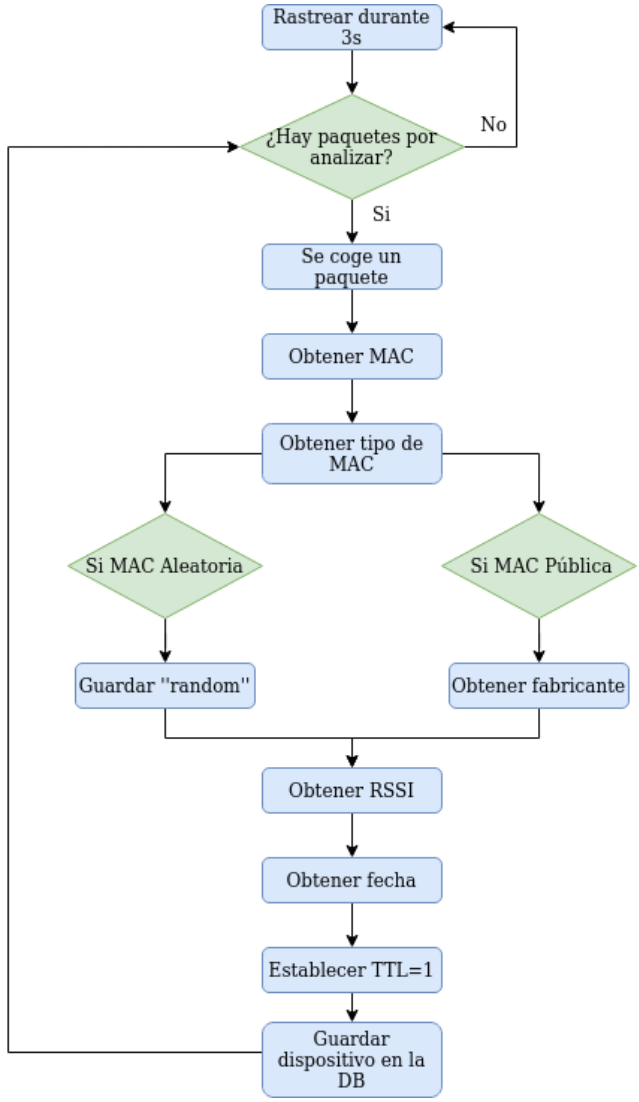

Figura 3.2: Diagrama de flujo del algoritmo BLE

3.2.4. Implementación

En esta sección se detallará el procedimiento seguido para el desarrollo de este algoritmo, así como las herramientas utilizadas.

Entorno de trabajo

Antes de empezar la programación del algoritmo fue necesario preparar el entorno de trabajo e instalar las librerías que se iban a utilizar en la Raspberry Pi.
Como se ha utilizado la Raspberry que se utilizó en el proyecto de Aitor Ojeda, no ha hecho falta instalar ningún software, ya que *Wireshark*, *Scapy*, *Python* y *SQLite3* estaban ya instalados.
Las versiones utilizadas de estos programas han sido:

- Kali Linux 2020.2
- Python 3.9.2
- Wireshark 3.4.4
- SqLite3 3.33.0

Además de estos programas, se ha utilizado el entorno de programación que trae por defecto Kali Linux, *Mousepad*, que funciona como editor para diferentes lenguajes de texto.

Las librerías que se utilizan para este algoritmo son:

- **bluepy**: Se utilizará para capturar los paquetes de descubrimiento BLE, ya que es una herramienta bien optimizada y más sencilla de utilizar para Bluetooth que *Scapy*, además de que filtra automáticamente los paquetes de difusión de descubrimiento [20].
- **mac_vendor_lookup**: Se utilizará para obtener el fabricante del chip Bluetooth, ya que directamente busca si se encuentra la MAC registrada en *IEEE* [23].
- **db**: Se ha creado esta clase, que se utilizará como librería, para crear, insertar y consultar la base de datos que se ha utilizado. Aunque ya estaba implementada en el proyecto de partida, se han añadido nuevos métodos para la tabla de dispositivos Bluetooth
- **datatime**: Se ha utilizado esta librería del sistema para obtener la hora en la que se analiza el paquete y poder así guardar la fecha y hora exacta en la base de datos.

Las primeras dos librerías se han instalado con el comando `pip3`, siguiendo sus respectivas páginas de documentación:

```
pip3 install <librería>
```

Para este algoritmo se ha creado un único archivo, llamado `sniff_bl.py`, que contiene todo el proceso descrito anteriormente, además de utilizar la clase `db.py` para la base de datos.

3.3. Desarrollo del algoritmo WiFi

En esta sección se detallará todo el desarrollo del algoritmo implementado para relacionar los dispositivos WiFi entre sí y determinar así el aforo de la sala. Se explicará tanto el diseño del algoritmo como las herramientas utilizadas, los datos que se guardarán en la base de datos y la hipótesis utilizada para diseñar el algoritmo.

3.3.1. Análisis

En esta sección se detallarán los requisitos tanto funcionales como no funcionales del algoritmo, de forma similar a los requisitos del algoritmo Bluetooth en la sección anterior.

RF1 - Captura de paquetes	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de capturar los paquetes necesarios para el proyecto, obteniendo solo los de tipo *probe request*.
Motivo	Esta funcionalidad es principal para relacionar los dispositivos, por lo que es esencial implementarla.

Cuadro 3.10: Requisito WiFi Nº1

RF2 - Almacenamiento de datos	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de guardar los datos de los dispositivos y SSIDs capturados.
Motivo	Esta funcionalidad servirá para relacionar los dispositivos manteniéndolos en la base de datos, por lo que es necesaria implementarla.

Cuadro 3.11: Requisito WiFi Nº2

RF3 - Actualización de los datos	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de actualizar los datos de la base de datos con los nuevos datos que se capturen.
Motivo	Deben actualizarse los registros de la DB para tener registrado las nuevas MACs aleatorias o el tiempo de vida del registro.

Cuadro 3.12: Requisito WiFi Nº3

RF4 - Detección del tipo de MAC	
Tipo de requisito	Funcional
Prioridad	Media
Descripción	El algoritmo debe ser capaz de diferenciar si la MAC del dispositivo es aleatoria o pública.
Motivo	Esta funcionalidad resulta útil para tener constancia de los fabricantes de los dispositivos y de cuántas direcciones hay de cada tipo.

Cuadro 3.13: Requisito WiFi Nº4

RF5 - Permanencia del dispositivo	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de detectar si un dispositivo sigue estando en la sala o no.
Motivo	Esta funcionalidad será principal para calcular correctamente el aforo sabiendo cuántos dispositivos hay.

Cuadro 3.14: Requisito WiFi Nº5

RF6 - Relación de dispositivos	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de relacionar los dispositivos entre sí.
Motivo	Esta funcionalidad permitirá detectar a las personas en la sala además de los dispositivos.

Cuadro 3.15: Requisito WiFi Nº6

RF7 - Relación dispositivo/persona	
Tipo de requisito	Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de detectar cuantos dispositivos llevan las personas de la sala basándose en el requisito Nº 6.
Motivo	Esta funcionalidad permitirá detectar el número de personas que hay en la sala independientemente de los dispositivos.

Cuadro 3.16: Requisito WiFi Nº7

RNF1 - Almacenamiento en tiempo real	
Tipo de requisito	No Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de guardar en tiempo real la información que capture en la base de datos.
Motivo	Es necesario que se guarden los datos en tiempo real para tener los dispositivos guardados según se capturen.

Cuadro 3.17: Requisito WiFi Nº8

RNF2 - Actualización en tiempo real	
Tipo de requisito	No Funcional
Prioridad	Alta
Descripción	El algoritmo debe ser capaz de mantener actualizados los datos de la base de datos constantemente en tiempo real.
Motivo	Es necesario que se actualicen los datos en tiempo real para tener constancia de los dispositivos que hay en la sala actualmente.

Cuadro 3.18: Requisito WiFi Nº9

3.3.2. Modelo de datos

Para este algoritmo se han utilizado 4 tablas de la base de datos llamadas *DataSniff*, *DataSniffRandom*, *SsidRegister* y *Person*. Las dos primeras se utilizan para guardar los dispositivos con MAC pública y MAC aleatoria, respectivamente. Estas dos tablas fueron creadas en el proyecto de partida y se han mantenido para simplificar el trabajo con los paquetes WiFi. La 3ª tabla se utilzia para guardar los SSIDs que se capturen de los mensajes *probe request*, y la última tabla se utilizará para guardar los dispositivos relacionados entre sí, teniendo una "persona" por fila.
Los campos de la tabla *SsidRegister* son:

- **mac**: Dirección MAC del dispositivo que transmite el paquete.
- **ssid**: SSID capturado del paquete.
- **date**: Fecha en la que se capturó el paquete.

Con estos campos se tiene la información necesaria para poder relacionar dos dispositivos como se verá en el siguiente apartado. Todos los atributos son de tipo *text* para guardar y comparar más fácilmente los datos.

La tabla *Person* guarda únicamente las direcciones MACs relacionadas entre sí, por lo que tendrá de 2 a 4 campos, dependiendo del parámetro de dispositivos por personas del algoritmo, el cuál se explica más adelante, en los que se guardarán esas direcciones. Igual que el resto de tablas, todos sus campos son de tipo *text*.

3.3.3. Diseño

En esta sección se detallará el algoritmo con el que se relacionan los distintos dispositivos WiFi entre sí y se detectan cuantas personas hay en la sala, explicando la hipótesis para la relación de estos y el algoritmo creado.

Hipótesis de relación

La hipótesis de este algoritmo es, fundamentalmente, que cada persona lleva consigo un número medio de dispositivos, por lo que relacionandolos entre si se obtendrá una estimación más precisa del aforo de esa sala.

Algoritmo

Este algoritmo se ejecuta simultáneamente junto con el de captura de dispositivos WiFi cuando el paquete capturado tenga un SSID dentro de los datos que transmite. Antes de ejecutar este algoritmo se comprueba si el SSID está en la base de datos y, en caso de que no esté, se guarda en la tabla *SsidRegister*.
El primer paso de este algoritmo es obtener todos los SSIDs que el dispositivo ha transmitido y, por tanto, se encuentran en la base de datos. Tras esto, si hay al menos dos redes se continúa con el proceso y, si no, se pasa a la siguiente captura. Este número de redes mínimas viene establecido por la hipótesis explicada en el apartado anterior. Tras esto se crea un diccionario en el que guardar las direcciones MAC como "*key*" y un valor numérico como "*value*", que será el número de SSIDs coincidentes por cada dirección MAC.
Una vez se tienen las redes asociadas a ese dispositivo, se pasan a analizar individualmente. De cada una de ellas se obtienen las direcciones MACs que han transmitido esa SSIDs y se añaden al diccionario, incrementando en uno su valor si ya estaban en él o añadiendo el identificador con un valor igual a 1.
Por último se trata cada MAC del diccionario por sí sola. Se comprueba si el valor de esa MAC es mayor que lo establecido como el número mínimo de redes coincidentes necesarias para que dos dispositivos pertenezcan a la misma persona (en este caso 2, con el desarrollo del experimento más adelante explicado este valor podrá cambiar para ajustarse a un caso más real), y en caso de ser así se crea o actualiza la tupla de la tabla *Person*. En caso de que haya que actualizar la tupla se introducirá la dirección MAC del diccionario en la misma fila que está la dirección MAC del dispositivo y, en caso de estar llena, se creará una nueva tupla con este identificador como único valor.

3.3.4. Implementación

En esta sección se detallará el procedimiento seguido para el desarrollo de este algoritmo, así como las herramientas utilizadas.

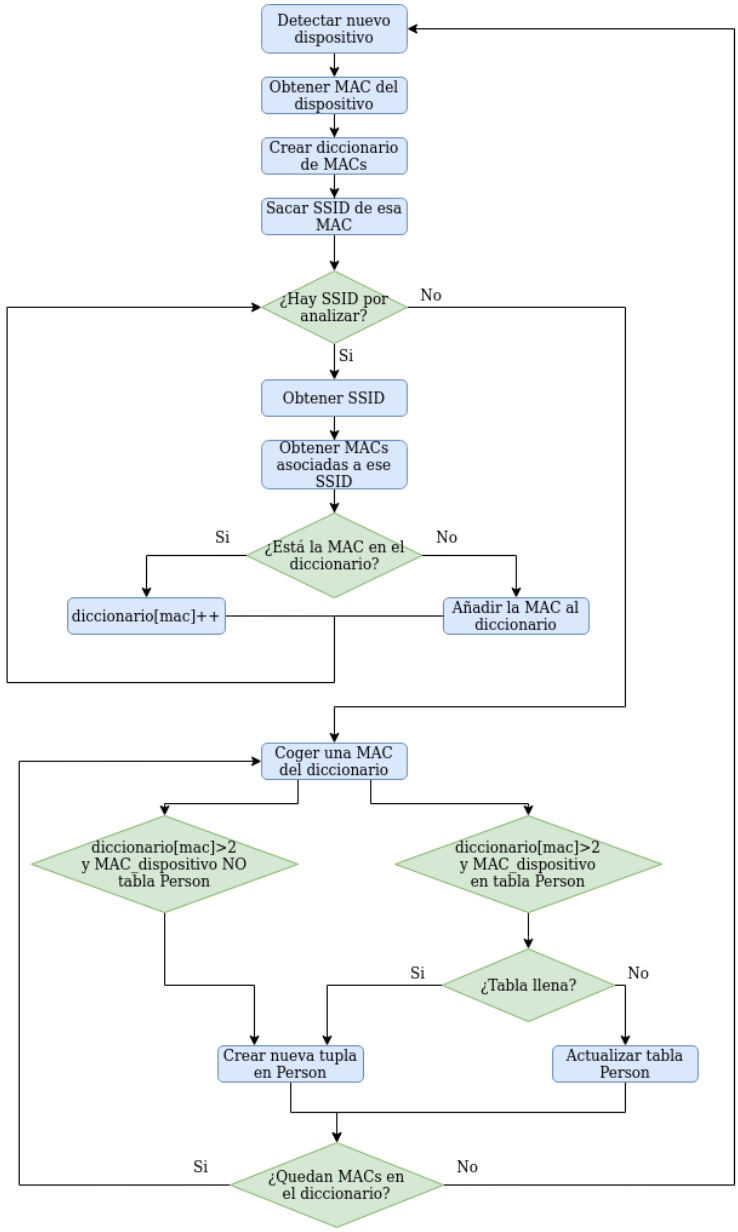

Figura 3.3: Diagrama de flujo del algoritmo WiFi

Entorno de trabajo

Igual que se explicaba en el algoritmo BLE, antes de comenzar con la programación del algoritmo fue necesario preparar el entorno de trabajo e instalar las librerías que se iban a utilizar en la Raspberry Pi.

Como se ha utilizado la Raspberry que utilizó Aitor Ojeda en su proyecto y se han desarrollado los dos algoritmos a la vez, no ha hecho falta instalar ningún software nuevo.
Las versiones de los softwares utilizadados fueron:

- Kali Linux 2020.2
- Python 3.9.2
- Wireshark 3.4.4
- SqLite3 3.33.0

Para este algoritmo se ha utilizado Wireshark para ver la información que transmiten los paquetes *probe request* en la búsqueda de redes, comprobando que algunos dispositivos transmiten los SSIDs conocidos y en que campo se encuentra esa cadena.
Igual que el algoritmo Bluetooth, el entorno de programación que se ha utilizado ha sido *Mousepad*.
Las librerías que se utilizan en este algoritmo son:

- **Scapy**: Se utiliza para capturar los paquetes de descubrimiento WiFi, como se utilizaba en el proyecto de partida, pero con la diferencia de que además ahora se captura el SSID [15].
- **netaddr**: Se utiliza para comprobar si la dirección MAC es aleatoria o no y, en caso de que no lo sea, para obtener el fabricante de ese chip [24].
- **db**: Igual que en el algoritmo Bluetooth, se utiliza esta clase para crear, insertar y consultar datos de la base de datos, creando y modificando tablas tanto ya existentes como tablas nuevas creadas para este proyecto.
- **Librerías del sistema**: Se han utilizado librerías como **datatime** para obtener la fecha y hora en la que se captura el paquete y guardarla en la DB, **sys** u **os**.

Las dos primeras librerías se han instalado con el comando `pip3`, de forma similar a las librerías utilizadas en la parte de Bluetooth.

Este algoritmo consta de un único archivo, llamado `sniff_wifi.py`, que contiene todo el proceso y funcionalidad descrita anteriormente, además de la clase db alojada en `db.py`, utilizada para tratar con la base de datos.

3.4. Script de Ejecución

Para ejecutar más fácilmente los algoritmos se creó un pequeño script en lenguaje `Python` para lanzar las capturas junto al temporizador, el cual elimina dispositivos de la base de datos si ha pasado suficiente tiempo desde que se detectó por última vez.

Para ello se han utilizado hilos, con los que se ejecutan tareas independientemente del programa principal, por lo que asignando a la captura de Bluetooth un hilo, a la de WiFi otro y al temporizador otro, se van a estar ejecutando los 3 simultáneamente. Además, son hilos *daemon*, por lo que además se ejecutan en segundo plano, permitiendo realizar otras acciones en el programa principal. Previamente a la ejecución de los hilos se establece la interfaz WiFi en modo monitor.

3.5. Prueba en un escenario real

Para probar la eficacia de los dos algoritmos descritos y, por tanto, como de exacta es la aproximación que se estudia en este proyecto, se ha diseñado un experimento que se realizará durante una clase de la asignatura de Fundamentos de Redes del primer curso de Ingeniería Informática con los alumnos de dicha materia.

3.5.1. Objetivos

El objetivo de esta prueba es probar la eficacia de los dos algoritmos diseñados y comprobar si con ellos se puede hacer una estimación real y precisa de los dispositivos y personas que hay dentro de la sala.
En cuanto a Bluetooth, se busca obtener el número de dispositivos que tienen esta red activada, siempre y cuando ese dispositivo tenga BLE como se explicaba anteriormente.
Por la parte del algoritmo WiFi, lo que se busca es obtener una estimación real de las personas en la sala, detectando más dispositivos que personas y relacionando los dispositivos que pertenezcan al mismo usuario entre sí, probando el algoritmo con diferentes parámetros para obtener una solución más real.

3.5.2. Diseño del experimento

El experimento se realizará en la Escuela de Ingeniería Informática durante la clase de Fundamentos de Redes, impartida por el tutor de este proyecto.
Se colocará la Raspberry en el aula y se dejarán funcionando los algoritmos de captura de WiFi y Bluetooth, sin eliminar nada de la base de datos, durante 15 minutos aproximadamente. Estos paquetes WiFi se tratarán manualmente, para ver cuántos dispositivos están relacionados entre sí y cuántos dispositivos se han capturado, y después se analizarán con el algoritmo diseñado, probando distintos parámetros como el número mínimo de SSIDs comunes y el número medio de dispositivos que lleva una persona.
El análisis manual consistirá en visualizar la base de datos y comprobar cuantos dispositivos ha detectado, tanto Bluetooth como WiFi, y ver cuantas SSIDs ha detectado, de cuántos dispositivos y si hay alguna común.
Tras esto se compararán los resultados obtenidos manualmente con los obtenidos con el algoritmo, viendo con que parámetros se obtiene un aforo más próximo al real.

Mientras se estén capturando los datos, se pasará a los alumnos presentes en el aula una encuesta que servirá como un pequeño estudio de campo, cuyos datos se usarán para definir los parámetros del algoritmo.

3.5.3. Diseño de la encuesta complementaria

Como estudio de campo para este experimento y proyecto se ha diseñado una encuesta para ajustar los parámetros de los algoritmos basándose en datos reales.
La encuesta consta de 7 preguntas de respuesta única o múltiple y de 2 respuesta de redacción corta, de las cuales no se obtiene ningún dato personal ni privado del usuario, por lo que se respeta completamente su privacidad.
De estas preguntas se hicieron 4 de carácter general. La primera pregunta era para conocer el número de dispositivos que la gente suele llevar encima en su día a día, la siguiente pregunta era para saber de que marca son esos dispositivos que lleva la gente, lo que nos dará una idea de que marcas son las más frecuentes y si se podrá detectar bien a las personas, debido al problema de privacidad que se explicaba en el capítulo 2. La tercera pregunta de la encuesta era para la gente que lleva pulseras o relojes inteligentes, cascos inalámbricos o cualquier dispositivo que normalmente suele estar conectado siempre, para saber si efectivamente lo llevan siempre conectado, algo que nos indicará el porcentaje de gente que lleva estos dispositivos y que, en caso de que llevarlos, ya tendremos dos dispositivos mínimo para cada uno de esos usuarios. La última pregunta de las generales era para saber que redes suelen llevar conectadas los encuestas, siendo las opciones Bluetooth, WiFi, ambas o ninguna. Esta última pregunta nos permitirá establecer un peso a los dispositivos detectados por WiFi o por Bluetooth, donde si la gente suele llevar apagado Bluetooth estos dispositivos tendrán más peso para detectar una persona.
Tras estas se hicieron 3 preguntas relacionadas con WiFi y 2 relacionadas con Bluetooth. Empezando por las de WiFi, las preguntas eran para saber cuántas redes, cuáles y de qué tipo son esas redes. La primera pregunta servirá para conocer cuántas redes se tiene guardadas de media y después, con el experimento, ver cuantas de ellas se transmiten. La segunda pregunta era para indicar que redes suelen ser a las que más se conectan, por ejemplo *eduroam*, la red WiFi de casa o la red privada de su sitio de trabajo, y la tercera pregunta para saber cuántas de las redes guardadas son públicas. Con la respuesta de estas dos preguntas se complementará a la anterior y se tendrá un conocimiento sobre cuántas redes públicas tienen guardadas y si transmiten todas ellas.
Las dos preguntas de Bluetooth eran cuántos dispositivos enlazados tiene cada encuestado y de que tipo son. Con esto se obtendrá una estimación de qué tipo de dispositivos se tienen enlazados y así ajustar mejor si se tienen guardados dispositivos como cascos inalámbricos, complementando así a una de las primeras preguntas relacionada con las pulseras inteligentes y cascos inalámbricos.
Estas preguntas se encuentran listadas en el Apéndice B junto con las respuestas que se podían dar a cada una.

3.5.4. Ejecución del experimento

El experimento se llevó a cabo el día 3 de mayo en el aula 05 de la Escuela de Ingeniería Informática, con una duración de 18 minutos.
Contó con una participación activa de 13 personas, 11 alumnos, el profesor Jesús Vegas y yo, más la participación indirecta de los alumnos presentes en al aula 06, colindante a la que se realizó el experimento, y los alumnos de los laboratorios de la 1ª planta. De estos usuarios se recogió información también debido a la proximidad con la Raspberry y al alcance que tiene tanto WiFi como Bluetooth, por lo que los resultados no se obtuvieron en condiciones ideales pero sí en una situación real que podría darse si se ponen en marcha estos algoritmos.

Se colocó la Raspberry en la mesa del profesor, con conexión a internet a través de cable de red, lo que permitió poder acceder remotamente a ella por SSH. Tras esto se pusieron en marcha los algoritmos de captura y se dejaron capturando durante todo el experimento. Mientras estaba en ejecución, se explicó a los alumnos la finalidad de este proyecto, pidiendo su participación en el experimento, además de pasarles a través de la herramienta Campus Virtual la encuesta mencionada en el apartado anterior.
Una vez rellenaron la encuesta y se capturaron suficientes paquetes como para analizarlos detenidamente, se paró la captura y se realizó una copia de seguridad de la base de datos.
Tras la finalización de este se analizaron los datos con distintos parámetros, tanto manualmente como con el algoritmo.

Capítulo 4

Resultados

En este capítulo se detallarán los resultados obtenidos en el experimento realizado, explicado en el capítulo anterior, además de exponer los resultados obtenidos en la encuesta realizada. La discusión de estos resultados y la validez del algoritmo se expondrán en el siguiente capítulo.

4.1. Resultados del experimento

En total se recogieron datos de 13 personas, capturando un total de 11 dispositivos Bluetooth, 25 dispositivos WiFi con una MAC conocida, 600 entradas en la base de datos de dispositivos con MAC aleatoria y 46 SSIDs, de las cuales 21 eran distintas. Dentro de los dispositivos WiFi, 12 de ellos han transmitido mínimo 2 SSIDs, por lo que se puede aplicar el algoritmo de relación entre ellas.
De los dispositivos Bluetooth, 1 de ellos tienen un RSSI menor a -78, por lo que ese dispositivo se considerará externo a la sala.
De los dispositivos WiFi 9 de ellos tienen un RSSI menor a -78, por lo que estos dispositivos se descartarán ya que se considera que están demasiado lejos del punto de rastreo.
En cuanto al análisis de las SSIDs, se descartó la red "*eduroam*" evitando que se capturasen los paquetes que transmitían ese identificador, ya que el experimento se realizó en la universidad y la mayoría de dispositivos iban a difundir esa red, como se vio en pruebas que se hizo antes del experimento.

4.2. Resultados de la encuesta

En esta sección se hará una distinción entre los resultados de la encuesta de los alumnos presentes en el experimento, la cuál realizaron durante este, y los resultados de la encuesta pública.

4.2.1. Encuesta del experimento

Durante el experimento se realizó la encuesta que se encuentra en el anexo B, contando con la participación de los 11 alumnos presentes en el aula.
En total había en el aula 23 dispositivos, con una media de 2,09 dispositivos por persona. La principal marca de estos dispositivos es *Xiaomi*, habiendo un total de 7, seguida por *Samsung* con 4. De las 11 personas del aula, 9 llevan conectada la red WiFi y 7 llevan conectado Bluetooth, coincidiendo en 5 usuarios las dos redes encendidas simultáneamente.
Por la parte de WiFi, la media de redes que tienen guardadas los usuarios es 4,9, de las cuales todos los usuarios coinciden en que tienen menos de 2 redes públicas guardadas, siendo las más frecuentes la red de la universidad y la red privada de casa.
En cuanto a Bluetooth, 7 usuarios tienen guardados menos de 5 dispositivos a los que se han conectado, 2 usuarios tienen guardados entre 6 y 10 dispositivos y los otros 2 usuarios restantes tienen guardados entre 11 y 15 dispositivos a los que se han conectado previamente. De todos estos dispositivos guardados, los principales son cascos inalámbricos, relojes o pulseras inteligentes, altavoces o reproductores multimedia y ordenadores. Con menos respuestas se encuentran los navegadores de los coches y las tablets.

4.2.2. Encuesta pública

La encuesta ha contado con una participación de 153 usuarios. La media de dispositivos que llevan los encuestados es 1,87, algo menor que en la encuesta del experimento. La marca más común entre esos dispositivos es Xiaomi, con un total de 69, seguido por Samsung con 42 dispositivos. 130 de los encuestados llevan normalmente encendida la red WiFi, 76 llevan encendido Bluetooth y 72 de ellos coinciden en llevar ambas redes encendidas.
En cuanto a WiFi, la media de redes que tienen guardadas los usuarios es de 10,46, de las cuales la mayoría coinciden en que públicas son menos de 2.
En cuanto a Bluetooth, 125 usuarios tienen guardados menos de 5 dispositivos a los que se han conectado, siendo la mayoría de respuestas. De esos dispositivos, los más comunes a los que se conectan los encuestado son cascos inalámbricos, relojes o pulseras inteligentes, reproductores multimedia y navegadores de coches, en ese orden.

Capítulo 5

Discusión

En este capítulo se van a discutir los resultados obtenidos en el experimento relacionándolos con la encuesta, tanto la propia del experimento como con la general. Además se compararán los resultados obtenidos en la encuesta con otros estudios hechos por compañías como Cisco, y se definirán los parámetros del algoritmo WiFi diseñado en este proyecto para que sea lo más óptimo posible.

Bluetooth

Comenzando por los dispositivos Bluetooth obtenidos en el experimento, se han detectado un total de 11 dispositivos. Relacionando este resultado con las de la encuesta, 7 alumnos llevaban encendido Bluetooth durante el experimento. El número total de dispositivos que llevaban estos alumnos que indicaron que solían llevarlo encendido era 17, de los cuales 4 tienen una pulsera inteligente que llevan siempre conectada al teléfono móvil. En la figura 5.1 se puede ver una parte de los resultados de la encuesta con los resultados pertenecientes a Bluetooth.

Dispositivos	Marca Dispositivos	MiBand	Dispositivos enlacados
3	Apple, Xiaomi	Sí	Auriculares, Pulsera inteligente, Coches, Tablets, Reproductor multimedia
3	Samsung, Xiaomi	Sí	Auriculares, Pulsera inteligente, Coches, Reproductor multimedia, PC
2	Samsung	Sí	Auriculares, Pulsera inteligente
2	Xiaomi, HP	Sí	Auriculares, Pulsera inteligente, Reproductor multimedia
2	Xiaomi, HP	No	Auriculares Tablets, PC
3	Xiaomi	No	Auriculares, Reproductor multimedia
2	Huawei	No	Auriculares, PC

Figura 5.1: Resultados de la encuesta en cuanto a Bluetooth

De los 7 alumnos que tenían encendido Bluetooth, 4 llevaban pulsera inteligente o cascos inalámbricos conectados, por lo que ahí tenemos 2 dispositivos BLE para cada uno, teniendo así por ahora 8 dispositivos reales.

Teniendo en cuenta que en el experimento realizado se detectaron 11 dispositivos, puede ser una estimación bastante acertada teniendo en cuenta sabemos con certeza que hay mínimo 8 dispositivos con el Bluetooth activado, más algún dispositivo de los usuarios que suelen llevar encendido Bluetooth pero no tienen pulsera inteligente o cascos siempre conectado, el número de dispositivos detectados es una estimación aceptable de los dispositivos reales que había.

WiFi

En el experimento se detectaron 600 entradas en la base de datos procedentes de dispositivos con una dirección MAC aleatoria. Dado que no había tantos dispositivos en el aula, la causa de este número tan alto de registros fue un problema con el tiempo de vida, *TTL*, de las entradas en la tabla.
Al capturar paquetes en tiempo real, si una dirección MAC no se ha vuelto a detectar en un lapso de 3 minutos, se elimina de la base de datos y se considera que ese dispositivo no se encuentra en la sala. En la ejecución del experimento se deshabilitó la función de eliminar registros de la base de datos para así tener la mayor cantidad de información posible, pero no se tuvo en cuenta que esto podría darse. Por este motivo, el análisis de los datos se harán basándose en las direcciones MAC públicas y en las SSIDs capturadas, que aunque pertenezcan a un dispositivo con una MAC aleatoria servirá para ajustar los parámetros del algoritmo y aproximar el aforo del aula.
Se detectaron 21 SSIDs distintas de un total de 46, por lo que había dispositivos que estaban relacionados entre sí, por lo que se aplicó el algoritmo WiFi diseñado en este proyecto con distintos parámetros, para comprobar con que combinación de ellos daban unos resultados más reales. Entre todas estas SSIDs se descartó la red *eduroam*, ya que el experimento se realizó en un aula de la universidad y todos los dispositivos iban a transmitir esa red como una red conocida.

Dispositivos/persona	Nº SSIDs coincidentes	Personas detectadas	Error
2	1	**19**	7
2	2	**27**	15
3	1	**16**	4
3	2	**26**	14
4	1	**14**	2
4	2	**26**	14

Figura 5.2: Personas detectadas según los parámetros del algoritmo

En la figura 5.2 se pueden ver los valores que se han dado a los parámetros del algoritmo y a las personas que se han detectado con cada uno de esos parámetros, junto al error de detección que hay en cada caso. Aunque se detectan más personas de las que había en el aula (13, aunque con el WiFi encendido solo había 12 personas), se debe a que el experimento no se realizó de forma aislada y había más alumnos tanto en las clases contiguas como en las aulas del piso de arriba, por lo que las condiciones no eran ideales.
Al haber quitado *eduroam* del conjunto de SSIDs, lo más óptimo según se aprecia en la figura anterior es ajustar el parámetro del número de SSIDs coincidentes a 1 para este experimento, y a 2 cuando se esté probando en un entorno real.

Con este parámetro a 1, se puede ajustar el parámetro de número máximo de dispositivo que puede llevar una persona. Tanto en la encuesta del experimento como en la encuesta pública, la media de dispositivos por persona era de 2, por lo que con estos dos parámetros ajustados en este entorno, se obtienen 19 personas dentro del aula (primera fila de la figura 5.2). Este resultado difiere ligeramente del número de usuarios reales en la sala, que eran 12 con el WiFi activado, pero se podría considerar una estimación aceptable teniendo en cuenta que varía en 7 personas, que podrían estar en algún aula contiguo como se explicaba anteriormente o que no se hayan podido capturar tantas SSIDs como para relacionar dos dispositivos.
Según el estudio realizado por Cisco [25], la media de dispositivos que lleva una persona pasará desde 2,4 que había en 2018 hasta 3,6 en 2023, por lo que con el creciente aumento de dispositivos inalámbricos con redes WiFi, se podría establecer este parámetro del algoritmo en 3 dispositivos como máximo. Con esto, se tendrían un total de 16 personas en el aula, más próximo a la realidad aunque cambiando un parámetro obtenido en la encuesta. De estos dispositivos que se indican en el estudio de Cisco, habrá dispositivos que no tengan WiFi, como pueda ser una pulsera inteligente o cascos inalámbricos, por lo que se dejará establecido en 2 el número de dispositivos por persona para el algoritmo WiFi.

La mayoría de SSIDs capturados eran de redes públicas, con alguna red privada de Movistar o Vodafone, pero en su mayoría públicas. Esto supone un problema a la hora de estimar el aforo correctamente, ya que no se tienen suficientes redes para relacionar dos dispositivos que puedan ser de la misma persona, debido como se explicaba en el capítulo 2 a que los dispositivos suelen transmitir solamente redes públicas debido a brechas de seguridad. Esto también justifica que se hayan obtenido 19 personas en el aula, ya que no se pueden obtener más SSIDs ni información sobre los dispositivos relacionados sin interactuar directamente con ellos, algo fuera del alcance de este proyecto.

Capítulo 6

Conclusiones

En este proyecto se han presentado dos algoritmos completamente funcionales con los que detectar tanto los dispositivos Bluetooth como las personas que se encuentran en una sala, dando así una aproximación del aforo de esta. Además, se han probado estos sistemas en un entorno real, obteniendo así unos resultados que han servido para perfeccionar el segundo algoritmo y obtener así una aproximación más precisa. La encuesta contó con más participación de la esperada, obteniendo un total de 153 respuestas, por lo que ha resultado útil para ajustar esos parámetros que se comentaban durante este documento. Los resultados obtenidos, pese a diferir de la realidad, son una estimación aceptable teniendo en cuenta que el experimento no se realizó en condiciones ideales, por lo que el algoritmo diseñado para relacionar dispositivos WiFi resuelve el problema planteado al inicio del proyecto. El algoritmo Bluetooth también es completamente funcional, detectando los dispositivos que tenga cerca el punto de rastreo, dando así una aproximación de estos.

Comparado con el resto de métodos para calcular el aforo, que se comentaban en el capítulo 2, el método diseñado en este trabajo tiene un coste mucho menor que otro tipo de sensores, además de mantener la privacidad de los usuarios, por lo que puede ser interesante seguir trabajando sobre este método en un futuro e intentar ajustar más los algoritmos, intentando relacionar los dispositivos Bluetooth entre sí también o incluso los dispositivos Bluetooth y WiFi de la misma persona, dando así una estimación aún más precisa de la ocupación de la sala.

Apéndice A

Manual de usuario

A.1. Script de ejecución

```
import os
import threading
import time
from tempo import *
from sniff_bl import *
from sniff_wifi import main_wifi

os.system("airmon-ng check kill")
os.system("airmon-ng start wlan0")


thread_tempo = threading.Thread(target=tempo, daemon=True)
thread_bl = threading.Thread(target=main_bl, daemon=True)
thread_wifi = threading.Thread(target=main_wifi, daemon=True)

thread_tempo.start()
thread_bl.start()
thread_wifi.start()
while True:
        time.sleep(10)
```

Para ejecutar el script, y por tanto los 2 algoritmos, lo único necesario será moverse a la carpeta donde se encuentra el script y ejecutarlo con el comando:

```
python3 start.py
```

Apéndice B

Preguntas de la Encuesta

Normalmente, ¿Cuántos dispositivos con WiFi o Bluetooth llevas encima?

- 0
- 1
- 2
- 3
- 4 o más

¿De qué marca son los dispositivos que sueles llevar encima? (Móvil, PC, Smartwatch, etc)

Respuesta múltiple entre varias marcas con opción de "Otro".

En caso de utilizar Mi Band, Smartwatch, cascos inalámbricos, etc ¿Lo llevas conectado la mayor parte del tiempo al teléfono móvil?

- Si
- No

¿Que redes llevas encendidas normalmente?

- WiFi
- Bluetooth

- WiFi y Bluetooth
- Ninguna de las dos

¿Cuántas redes WiFi tienes guardadas en tu teléfono móvil?

Respuesta corta.

¿Cuáles son las redes a las que te conectas más frecuentemente?

Respuesta corta.

¿Cuántas de esas redes guardadas son públicas?

- Menos de 2
- Entre 3 y 5
- Entre 5 y 8
- 9 o más

En el caso de Bluetooth en el teléfono móvil, ¿Cuántos dispositivos tienes enlazados?

- Menos de 5
- Entre 6 y 10
- Entre 11 y 15
- 16 o más

¿Qué tipo de dispositivos tienes enlazados?

Respuesta múltiple entre varios dispositivos con opción de "Otro".

Apéndice C

Contenidos del .ZIP

En el archivo ZIP entregado para para este proyecto se encuentran los siguientes archivos:

- Memoria del proyecto. Se adjuntará este documento de forma digital.
- Código de los algoritmos, script y clases necesarias. Se adjuntarán todos los archivos necesarios para su funcionamiento.

Bibliografía

[1] Edoardo Longo, Alessandro E.C. Redondi, and Matteo Cesana. Accurate occupancy estimation with wifi and bluetooth/ble packet capture. *Computer Networks*, 163:106876, 2019.

[2] Julien Freudiger. How talkative is your mobile device? an experimental study of wi-fi probe requests. In *Proceedings of the 8th ACM Conference on Security Privacy in Wireless and Mobile Networks*, WiSec '15, New York, NY, USA, 2015. Association for Computing Machinery.

[3] Mathieu Cunche, Mohamed-Ali Kaafar, and Roksana Boreli. Linking wireless devices using information contained in wi-fi probe requests. *Pervasive and Mobile Computing*, 11:56–69, 2014.

[4] Aitor Ojeda Bilbao. Estimación de la ocupación de un espacio físico a través de la captura de paquetes wifi, Jun 2020.

[5] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik Rye, and Dane Brown. A study of mac address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies*, 2017, 03 2017.

[6] Johannes Becker, David Li, and David Starobinski. Tracking anonymized bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019:50–65, 07 2019.

[7] Cómo funciona bluetooth low energy. `https://www.welivesecurity.com/la-es/2020/03/17/como-funciona-bluetooth-low-energy/`.

[8] Mac address. `https://wintelguy.com/`.

[9] Bluetooth addresses privacy in bluetooth low energy. `https://www.novelbits.io/bluetooth-address-privacy-ble/`.

[10] Qué es y cómo cambiar el ssid o nombre de nuestra red wifi. `https://www.redeszone.net/2018/04/24/ssid-red-wi-fi-modificacion/`.

[11] Rssi-a changing definition. `https://www.cwnp.com/rssi-changing-definition/`.

[12] Understanding rssi levels. `https://www.metageek.com/training/resources/understanding-rssi.html`.

[13] Análisis: Raspberry Pi 3 Modelo B+. https://hardzone.es/reviews/perifericos/analisis-raspberry-pi-3-modelo-b/.

[14] Kali docs. https://www.kali.org/docs/.

[15] Scapy. https://scapy.readthedocs.io/en/latest/introduction.html.

[16] Aircrack-ng. https://www.aircrack-ng.org/.

[17] Airmon-ng. https://www.aircrack-ng.org/doku.php?id=airmon-ng.

[18] Wireshark doc. https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html.

[19] Python faq. https://docs.python.org/3/faq/general.html#what-is-python.

[20] bluepy - a bluetooth le interface for python. http://ianharvey.github.io/bluepy-doc/.

[21] Sqlite home page. https://www.sqlite.org/index.html.

[22] Especificación de requisitos software. https://es.wikipedia.org/wiki/Especificaci%C3%B3n_de_requisitos_de_software#Tipos_de_requisitos.

[23] mac-vendor-lookup. https://pypi.org/project/mac-vendor-lookup/.

[24] netaddr0.8.0 documentation. https://netaddr.readthedocs.io/en/latest/.

[25] Cisco annual internet report - cisco annual internet report (2018–2023) white paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, Mar 2020.