ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Universidad Politécnica de Cartagena

**Trabajo Fin de Master**

**MASTER EN INGENIERÍA DE TELECOMUNICACIÓN**

# Evaluación de un sistema de ocupación de aulas mediante el análisis de datos de la red Wifi Eduroam

etsit escuela técnica superior de ingeniería de telecomunicación

AUTOR: Alejandro González Redel

DIRECTOR: Alejandro S. Martínez Sala

Codirector: Mathieu Kessler

Julio / 2022


| | |
|---|---|
| **Autor** | Alejandro González Redel |
| **E-mail del autor** | alejandrogonzalezredel@gmail.com |
| **Director /mail** | Alejando S. Martínez Sala / alejandros.martinez@upct.es |
| **Co-director / mail** | Mathieu Kessler / mathieu.kessler@upct.es |
| **Título del TFM** | **Evaluación de un sistema de ocupación de aulas mediante el análisis de datos de la red Wifi Eduroam** |
| **Resumen** | En la UPCT hay una infraestructura de red empresarial y, en particular, una infraestructura Wifi-inalámbrica gestionada por el Servicio de Informática. La red Wifi UPCT está basada en Access Point de Aruba y una controladora Aurba y gestiona dos redes principales de usuarios; Open-UPCT y Eduroam. Un elevado número de usuarios de la UPCT (alumnos, profesores y personal de administración) usan diariamente la red Wifi corporativa en su labor en la universidad.<br><br>Se ha hecho un estudio de la viabilidad de estimar la ocupación de aulas de la UPCT a partir de la captura y análisis de datos gestionados en la red Wifi de la UPCT.<br><br>Un primer objetivo del proyecto ha sido estudiar la infraestructura de la red Wifi de la UPCT, automatizar la recopilación de datos mediante consultas SNMP a la controladora Aruba y preprocesar los datos y almacenarlos en ficheros CSV para su posterior análisis. Todos los datos se tratan como anónimos, para garantizar el cumplimiento de la normativa de privacidad y protección de datos de carácter personal.<br>El siguiente objetivo ha sido hacer un análisis exploratorio de los datos para estimar la ocupación de aulas. Se ha hecho un primer estudio en aulas del sótano del Hospital de Marina y en la biblioteca de Antigones. |
| **Titulación** | Master en Ingeniería de Telecomunicación. |
| **Departamento** | Tecnología de la Información y las Comunicaciones. |
| **Fecha de presentación** | Julio - 2022 |

# Agradecimientos

A todas las personas que han estado durante este periodo de mi vida, donde nada ha sido fácil, pero vamos encontrando el camino. Simplemente Gracias.

También he de destacar que sin la colaboración y apoyo de los vicerrectorados de Campus infraestructuras y de Nuevas Tecnología no hubiera sido posible este proyecto. En particular a Natalio, Ana Belén y Alejandro del servicio de Informática de la UPCT por su fundamental colaboración y apoyo para crear una infraestructura segura de trabajo, resolución rápida de problemas y poder obtener datos reales.

# Índice

# Índice de figuras

# Índice de tablas

# Capítulo 1. Introducción y objetivos

## 1.1 Introducción y objetivos

En la actualidad, aunque todos los usuarios tengan una conexión de datos mediante la red móvil es una realidad que la gran mayoría del tiempo estamos conectados a redes wifi, ya sea en una cafetería en el hospital o incluso dentro de un avión. Y en nuestro caso la universidad no es una excepción, los alumnos, profesores y personal externo a la universidad se conecta a la red Wi-Fi desplegada en el entorno universitario ya sea en las redes privadas (Eduroam) o en la pública (Open-UPCT).

La red Wi-Fi de la universidad está compuesta por una gran cantidad de puntos de acceso o AP (del inglés, Access Point), la cual genera un área de cobertura muy amplia que engloba todos los edificios pertenecientes a la Universidad Politécnica de Cartagena (UPCT) permitiendo así que los usuarios de la misma tengan una conexión rápida y estable durante su estancia en el complejo universitario. Todos estos AP envían información de cada usuario a una controladora del fabricante Aruba para la monitorización del rendimiento tanto de la red como de los recursos hardware.

El objetivo de este proyecto está basado en estimar el número de usuarios que sen encuentran en un aula concreta de la universidad, mediante la obtención, a través del protocolo SNMP, y tratamiento de los parámetros que nos ofrece la controladora de cada usuario con una latencia baja.

Para conseguir el objetivo general debemos estudiar y validar las siguientes hipótesis:

- Se puede detectar si un aula está ocupada o no.
- Se puede estimar la cantidad de personas que hay en un aula en un intervalo de tiempo.
- Se puede estimar la cantidad de personas que hay en un aula con una latencia baja o en tiempo real.

Para conseguir este objetivo deberemos desarrollar un programa que realice consultas SNMP (Simple Network Management Protocol o en español, Protocolo Simple de Gestión de Red) desde un servidor interno en la universidad hacia la controladora Aruba, almacenarlos en ficheros CSV (Comma Separated Values) dentro del servidor de la universidad para su posterior tratamiento, obteniendo como resultado una estimación del número de usuarios en una clase. Para la mejor compresión del entorno de trabajo se incluye un esquema simple del sistema.

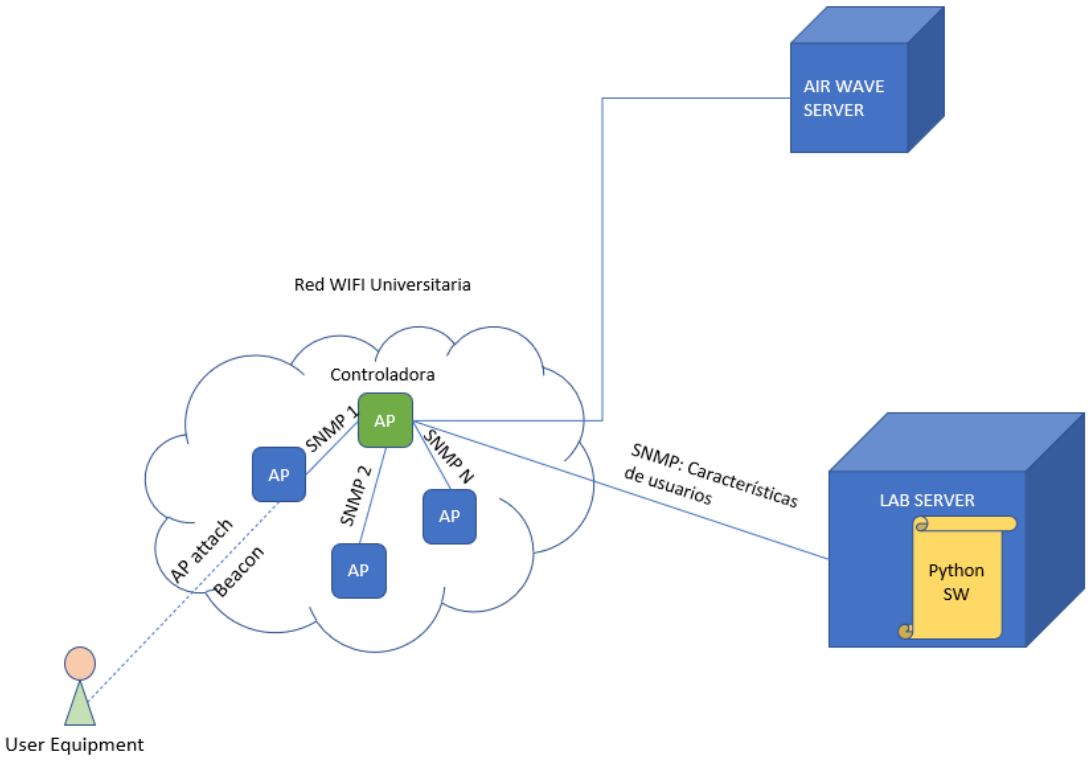



*Figura 1. Esquema simplificado del entorno de trabajo.*

Para este proyecto hemos usado como método de consulta dos herramientas del fabricante Aruba, cada una de ellas nos proporcionaba la información necesaria para caracterizar la red de una manera más rápida y sencilla:

**Aruba Airwave**

Aruba AirWave es un sistema de operaciones de red potente y fácil de usar que no solo administra la infraestructura cableada e inalámbrica de Aruba y una amplia gama de fabricantes de terceros, sino que también brinda visibilidad granular de los dispositivos, usuarios y aplicaciones en la red. Con una visión sin precedentes y un control centralizado para gestionar de forma eficaz las infraestructuras empresariales globales, AirWave permite que las organizaciones de TI optimicen de forma proactiva el rendimiento de la red, fortalezcan la seguridad inalámbrica y mejoren la experiencia del usuario final.

A través de una interfaz de usuario centralizada e intuitiva, AirWave brinda monitoreo en tiempo real, alertas proactivas, informes históricos y solución de problemas rápida y eficiente. Las vistas de panel dedicadas ayudan rápidamente a ver los posibles problemas de cobertura de RF, el tráfico de comunicaciones unificadas y colaboración (UCC), el rendimiento de las aplicaciones y el estado de los servicios de red.

La ubicación y el mapeo de VisualRF ofrecen vistas de toda la red de todo el entorno de RF. Los mapas de cobertura Wi-Fi y la topología cableada subyacente muestran una imagen clara y precisa de quién está en la red, su ubicación y el rendimiento de la red. Además, las superposiciones de la salud del cliente y el rendimiento de la aplicación

pueden ayudar a diagnosticar rápidamente problemas específicos de un cliente, un plano de planta o una ubicación específica.” [11]

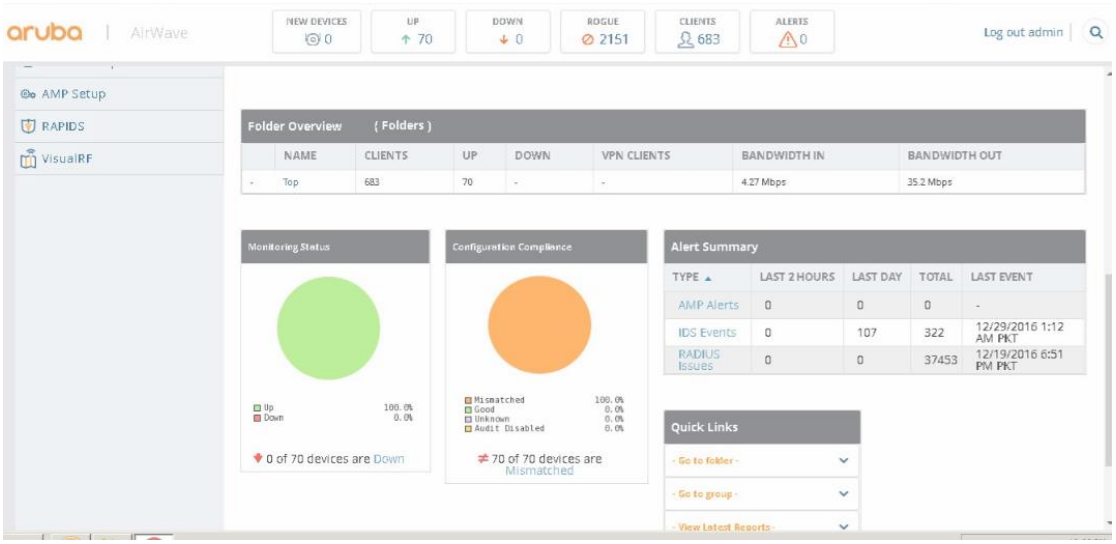



*Figura 2. Página principal del gestor Aruba Airwave*

**Aruba Virtual Controller**

“Instant no requiere un controlador de movilidad externo para regular y administrar la red Wi-Fi. En cambio, un IAP en cada red asume el rol de controlador virtual. Coordina, almacena y distribuye la configuración necesaria para proporcionar una funcionalidad centralizada para regular y administrar la red Wi-Fi. El controlador virtual es el único punto de configuración y administración de firmware. Cuando está configurado, el controlador virtual configura y administra el túnel VPN a un Controlador de movilidad en el centro de datos.

El controlador virtual también funciona como cualquier otro AP con escalabilidad completa de RF. También actúa como un nodo, coordinando la asignación de direcciones DHCP para clientes traducidos de direcciones de red, lo que garantiza la movilidad de los clientes cuando se desplazan entre diferentes IAP.” [12]

## 1.2 Herramientas software y elementos usados

Para el desarrollo de este trabajo se han utilizado los siguientes recursos hardware y software:

- Puntos de Acceso
  - Aironet 1240 IOS 12.3(8)JA2
  - Aironet 1100 IOS 12.3(8)JEE
  - Aironet 1600 IOS 15.3
- Linux Ubuntu 18.04 LTS. OS libre que es el que usaba el servidor donde hemos corrido todos los programas realizados en el proyecto.
- Aruba Operating System Software Version: 6.5.4.9. Version de la controladora Aruba para la gestion de los Puntos de acceso.

- AirWave Management Platform 8.2.13.1. Programa encargado de ver todas las características de la red y de los usuarios conectados a ella.
- Python. Lenguaje de programación utilizado para la creacion de todos los programas del proyecto.
- Librerías de la biblioteca de python.
    - Glob
    - Datetime
    - Pathlib
    - Numpy
    - Pandas
    - Matplotlib
- Pycharm. Software utilizado para el uso del lenguaje de programación de python por su sencillez a la hora de empezar a crear un proyecto. Version 2021.3.3
- SNMP y MIBs de Aruba. Usamos las MIBs de Aruba para realizar las consultas en su version 5.0.
- Windows 10 con Office 365 para la redacción del proyecto y la manipulación de tablas obtenidas de la controladora.

## 1.3 Estructura y organización del proyecto

En la memoria de este proyecto vamos a estudiar la arquitectura de una red wifi profundizando en su protocolo de conexión de usuario 802.11 en sus diferentes versiones, así como el protocolo simple de gestión de red o SNMP y aprenderemos la estructura de un árbol de MIBs.

En el tercer capítulo de este trabajo una vez estudiado los protocolos que corren en nuestro entorno Wi-Fi, desarrollaremos un programa en Python que nos ayude a recolectar todos los datos ofrecidos por la controladora que nos sean de utilidad de forma periódica.

En capitulo número cuatro haremos un programa que nos ayude a hacer un análisis exploratorio de la gran cantidad de datos recogidos de la controladora para encontrar patrones o posibles puntos de estudio nuevos y obtener resultados acerca de la ocupación de las aulas, para esto nos centraremos en una zona concreta de la universidad, pero permitiendo poder realizar este análisis en cualquier zona del complejo universitario

Por último, se explican las conclusiones que se han obtenido del análisis de los datos, así como todo lo que hemos aprendido con el trabajo realizado y los objetivos que hemos alcanzado. Además, se proponen posibles trabajos futuros.

# Capítulo 2. Tecnologías y conceptos clave

## 2.1 Resumen protocolo Wifi IEEE 802.11

"La especificación IEEE 802.11 (ISO/IEC 8802-11) es un estándar internacional que define las características de una red de área local inalámbrica (WLAN). La palabra Wi-Fi (Wireless Fidelity, Fidelidad inalámbrica, incorrectamente abreviado wifi) es el nombre de la certificación otorgada por la Wi-Fi Alliance, anteriormente WECA (Wireless Ethernet Compatibility Alliance), grupo que garantiza la compatibilidad entre dispositivos que utilizan el estándar 802.11. Por el uso indebido de los términos (y por razones de marketing) el nombre del estándar se confunde con el nombre de la certificación. Una red WIFI es en realidad una red que cumple con el estándar 802.11. A los dispositivos certificados por la Wi-Fi Alliance se les permite usar este logotipo:


*Figura 3. Logotipo de certificación Wi-Fi*

Con wifi se pueden crear redes de área local inalámbricas de alta velocidad siempre y cuando el equipo que se vaya a conectar no esté muy alejado del punto de acceso. En la práctica, wifi admite ordenadores portátiles, equipos de escritorio, asistentes digitales personales (PDA) o cualquier otro tipo de dispositivo de alta velocidad con propiedades de conexión también de alta velocidad (11 Mbps o superior) dentro de un radio de varias docenas de metros en ambientes cerrados (de 20 a 50 metros en general) o dentro de un radio de cientos de metros al aire libre." [3]

"El estándar original 802.11 fue publicado en junio de 1997. Este especifica tecnologías en la capa física y en la subcapa Data Link Layer de MAC. El grupo de trabajo que se encargó de la capa física definió tres especificaciones:

- Infrarrojo. Para las implementaciones de redes inalámbricas esta tecnología es obsoleta.
- Frecuency Hoppping Spread Spectrum (FHSS). Una señal de RF se considera Spread Spectrum cuando el ancho de banda de la señal es más amplio de lo que se requiere para transportar dicha señal. Esta tecnología fue patentada durante la segunda guerra mundial.
- Direct Sequence Spread Spectrum(DSSS). DSSS es otro tipo de tecnología Spread Spectrum el cual utiliza canales fijos, es decir, a

diferencia de FHSS, esté no realiza saltos de canales para transmitir información. Cluase 16.

El estándar 802.11 define que FHSS y DSSS puede transmitir en la banda de 2.4 GHz:

- DSSS puede transmitir en el rango de 2.4 a 2.4835.
- FHSS puede transmitir en el rango de 2.402 a 2.480 en donde solo se permite transmitir 1 MHz de información." [4]

"Los años posteriores a la publicación del estándar 802.11 surgieron nuevos grupos de trabajo que continuaron con la ratificación de nuevos protocolos 802.11. Los más importantes son los siguientes.

- IEEE 802.11: el estándar que sirve de base en la comunicación de las redes inalámbricas. El primer estándar Wi-Fi del año 1997 permitió transferir datos a 1 Mbps.
- IEEE 802.11a: se desarrolló sobre la base del estándar IEEE 802.11. Llegó en 1999, funcionaba en la banda de 5 GHz y alcanzó una velocidad máxima de 54 Mbps.
- IEEE 802.11b: fue el primer estándar desarrollado a finales de los años noventa. Es capaz de transferir dados a un máximo de 11 Mbps en la banda de 2,4 GHz.
- IEEE 802.11g: también utiliza la banda de 2,4 GHz. Con este estándar, la velocidad máxima de transmisión se incrementó hasta los 54 Mbps. Llegó a partir de 2003.
- IEEE 802.11n: se ratificó en septiembre de 2009. Funciona tanto en la banda de 2,4 GHz como en la de 5 GHz y alcanza velocidades de hasta 600 Mbps.
- IEEE 802.11ac: se estandarizó a finales de 2013. Opera en la banda de 5 GHz y puede alcanzar velocidades de 1.300 Mbps.
- IEEE 802.11ax: un avance importante que alcanza velocidades de hasta 10 Gbps.
- IEEE 802.11be: será el próximo gran salto en conectividad Wi-Fi. Está previsto para 2024, trabajará en las bandas de 2,4 GHz, 5 GHz y 6 GHz, y promete velocidades de hasta 30 Gbps." [6]

"Las tramas 802.11 incluyen los siguientes campos:

1. Campo Versión de protocolo: la versión de la trama 802.11 en uso. Campos Tipo y Subtipo: identifican una de las tres funciones y subfunciones de la trama (control, datos y administración).
2. Campo A DS: se establece en 1 para las tramas de datos destinadas al sistema de distribución (dispositivos en la estructura inalámbrica).
3. Campo Desde DS: se establece en 1 para las tramas de datos que salen del sistema de distribución.□ Campo Más fragmentos: se establece en 1 para las tramas que tienen otro fragmento.

4. Campo Reintentar: se establece en 1 si la trama es una retransmisión de una trama anterior.
5. Campo Administración de energía: se establece en 1 para indicar que un nodo estará en el modo de ahorro de energía.
6. Campo Más datos: se establece en 1 para indicarle a un nodo en el modo de ahorro de energía que se almacenan más tramas en búfer para ese nodo.
7. Campo Privacidad equivalente por cable (WEP): se establece en 1 si la trama contiene información encriptada mediante WEP para propósitos de seguridad
8. Campo Orden: se establece en 1 en una trama de tipo de datos que utiliza la clase de servicio Estrictamente ordenada (no requiere reordenamiento).
9. Campo Duración/ID: según el tipo de trama, representa el tiempo que se requiere en microsegundos para transmitir la trama o una identidad de asociación (AID) para la estación que transmitió la trama.
10. Campo Dirección de destino (DA): contiene la dirección MAC del nodo de destino final en la red.
11. Campo Dirección de origen (SA): contiene la dirección MAC del nodo que inició la trama.
12. Campo Dirección del receptor (RA): contiene la dirección MAC que identifica al dispositivo inalámbrico que es el destinatario inmediato de la trama.
13. Campo Número de fragmento: indica el número de cada fragmento de la trama.
14. Campo Número de secuencia: indica el número de secuencia asignado a la trama. Las tramas retransmitidas se identifican con números de secuencia duplicados.
15. Campo Dirección del transmisor (TA): contiene la dirección MAC que identifica al dispositivo inalámbrico que transmitió la trama.
16. Campo Cuerpo de la trama: contiene la información que se transporta. En las tramas de datos; generalmente se trata de un paquete IP.
17. Campo FCS: contiene una comprobación de redundancia cíclica (CRC) de 32 bits de la trama.” [5]

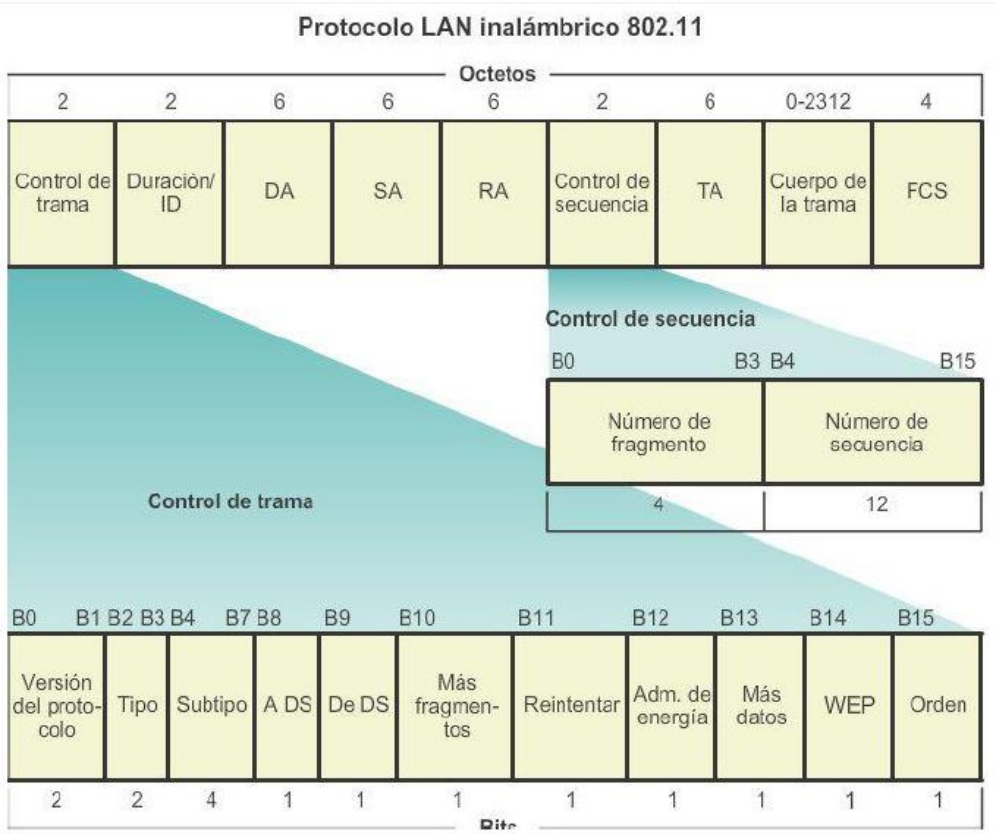



*Figura 4. Trama protocolo 802.11*

Una vez conocido el estándar de conexión wifi sus diferentes protocolos y como se forman sus tramas también es importante saber cómo un usuario haciendo uso de esos protocolos comparte una serie de tramas con los puntos de acceso para establecer una conexión.

"Los tres estados de conexión 802.11 son:

- No autenticado ni asociado.
- Autenticado, pero aún no asociado.
- Autenticado y asociado.

Una estación móvil debe estar en un estado autenticado y asociado antes de que ocurra la transmisión de datos. Para ello, la estación móvil y el AP intercambiarán una serie de frames de administración 802.11 para llegar a un estado autenticado y asociado.

Una estación móvil comienza como no autenticada y asociada.

1. Una estación móvil envía solicitudes de sondeo para descubrir redes 802.11 dentro de su proximidad. Las solicitudes de sondeo anuncian las velocidades de datos admitidas por las estaciones móviles y las capacidades 802.11, como 802.11n. Debido a que la solicitud de sondeo se envía desde la estación móvil a la dirección de capa 2 de destino y al BSSID de ff:ff:ff:ff:ff:ff, todos los AP que la reciban responderán.

2. Los puntos de acceso que reciben la solicitud de sondeo verifican si la estación móvil tiene al menos una velocidad de datos compatible común. Si tienen velocidades de datos compatibles, se envía una respuesta de sondeo anunciando el SSID (nombre de la red inalámbrica), las velocidades de datos admitidas, los tipos de encriptación si se requieren y otras capacidades 802.11 del AP.

Una estación móvil elige redes compatibles a partir de las respuestas de sondeo que recibe. La compatibilidad podría basarse en el tipo de cifrado. Una vez que se descubren las redes compatibles, la estación móvil intentará la autenticación 802.11 de bajo nivel con puntos de acceso compatibles. Tenga en cuenta que la autenticación 802.11 no es lo mismo que los mecanismos de autenticación WPA2 o 802.1X que se producen después de autenticar y asociar una estación móvil. Originalmente, los marcos de autenticación 802.11 se diseñaron para el cifrado WEP; sin embargo, se ha demostrado que este esquema de seguridad es inseguro y, por lo tanto, está obsoleto. Debido a esto, los marcos de autenticación 802.11 están abiertos y casi siempre tienen éxito.

3. Una estación móvil envía un marco de autenticación 802.11 de bajo nivel a un AP configurando la autenticación para abrir y la secuencia en 0x0001.

4. El AP recibe el marco de autenticación y responde a la estación móvil con el marco de autenticación configurado para abrir, lo que indica una secuencia de 0x0002.

Si un AP recibe cualquier trama que no sea una solicitud de autenticación o sondeo de una estación móvil que no está autenticada, responderá con una trama de desautenticación que coloca al móvil en un estado no autenticado o no asociado. La estación deberá comenzar el proceso de asociación desde el paso de autenticación de bajo nivel. En este punto, la estación móvil está autenticada pero aún no está asociada. Algunas capacidades de 802.11 permiten que una estación móvil se autentique a bajo nivel en múltiples puntos de acceso. Esto acelera el proceso de asociación cuando se mueve entre puntos de acceso. Una estación móvil se puede autenticar 802.11 en varios puntos de acceso; sin embargo, solo se puede asociar activamente y transferir datos a través de un único punto de acceso a la vez.

5. Una vez que una estación móvil determina a qué AP le gustaría asociarse, enviará una solicitud de asociación a ese AP. La solicitud de asociación contiene los tipos de cifrado elegidos si es necesario y otras capacidades 802.11 compatibles.
Si un AP recibe una trama de una estación móvil que está autenticada pero aún no asociada, responderá con una trama de desasociación que coloca al móvil en un estado autenticado, pero no asociado.

6. Si los elementos en la solicitud de asociación coinciden con las capacidades del AP, el AP creará una ID de asociación para la estación móvil y responderá con una respuesta de asociación con un mensaje de éxito que otorga acceso a la red a la estación móvil.

7. Ahora la estación móvil está asociada con éxito al AP y puede comenzar la transferencia de datos.” [7]

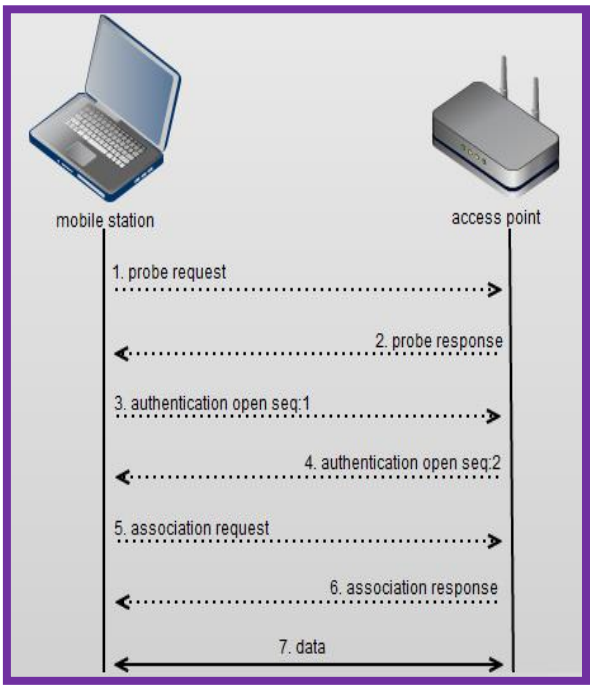



*Figura 5. Proceso de asociación de un UE*

## 2.2 Arquitectura Eduroam

### 2.2.1 Que es Eduroam

“Eduroam (contracción de education roaming) es el servicio mundial de movilidad segura desarrollado para la comunidad académica y de investigación. Eduroam persigue el lema "abre tu portátil y estás conectado".

El servicio permite que estudiantes, investigadores y personal de las instituciones participantes tengan conectividad Internet a través de su propio campus y cuando visitan otras instituciones participantes.

Eduroam ES es una iniciativa englobada en el proyecto Red IRIS que se encarga de coordinar a nivel nacional los esfuerzos de instituciones académicas con el fin de conseguir un espacio único de movilidad. En este espacio de movilidad participa un amplio grupo de organizaciones que, en base a una política de uso y una serie de requerimientos tecnológicos y funcionales, permiten que sus usuarios puedan desplazarse entre ellas disponiendo en todo momento de conectividad.

Por otro lado, eduroam ES forma parte de la iniciativa eduroam a nivel internacional, financiada a través de GEANT 3, y operada por varias redes académicas europeas y TERENA. Esta iniciativa amplía el espacio de movilidad al ámbito académico europeo, a través de eduroam Europa, y tiende puentes con eduroam Canadá, eduroam US, y eduroam APAN (Asia y Pacífico).” [8]


*Figura 6. Mapa de las redes Eduroam desplegadas en España*

### 2.2.2 Despliegue de Eduroam en las Universidades.

Para conseguir que este proyecto de conexión a internet entre universidades se lleve a cabo se necesita básicamente 2 cosas:

Una arquitectura hardware que soporte la creación de una red Wifi

Un sistema de autenticación común 802.1X

En primer lugar, para la arquitectura hardware lo que se necesitan es una gran cantidad de puntos de acceso repartidos por las diferentes aulas y espacios de la

universidad generando con ello un área de cobertura que permita mantener una señal WIFI de calidad.

Todos estos puntos de acceso van integrados dentro de lo que se conoce como controladora, que es un gestor de la infraestructura desplegada que se gestiona mediante una aplicación web. En esta controladora están definidos todos los parámetros de los puntos de acceso como autenticación, SSID, direccionamiento IP… además a través de ella se puede comprobar mediante estadísticas que recoge cada uno de los APs información acerca de la calidad de la red.

También necesitamos una arquitectura LAN compuesta por switches y routers que nos permitirán la conexión con la controladora y la salida a internet además de un servidor DHCP que asigne IP a cada usuario cuando se conecta.

Por otro lado, para que sea una red wifi segura se necesita que haya un control de acceso que autentique al usuario antes de permitirle navegar para ello se hace uso de un servidor de autenticación mediante el protocolo 802.1X Radius. En este servidor de autenticación Radius se comprueban los datos de los usuarios que deben estar registrados en el LDAP para que puedan tener acceso a la red Eduroam. Para que esto funcione entre universidades de España y del extranjero es necesario que se le permitan entre servidores de autenticación la consulta a los datos de usuario.

Toda esta arquitectura se muestra en la siguiente figura de la red Eduroam y será muy similar en el resto de universidades.

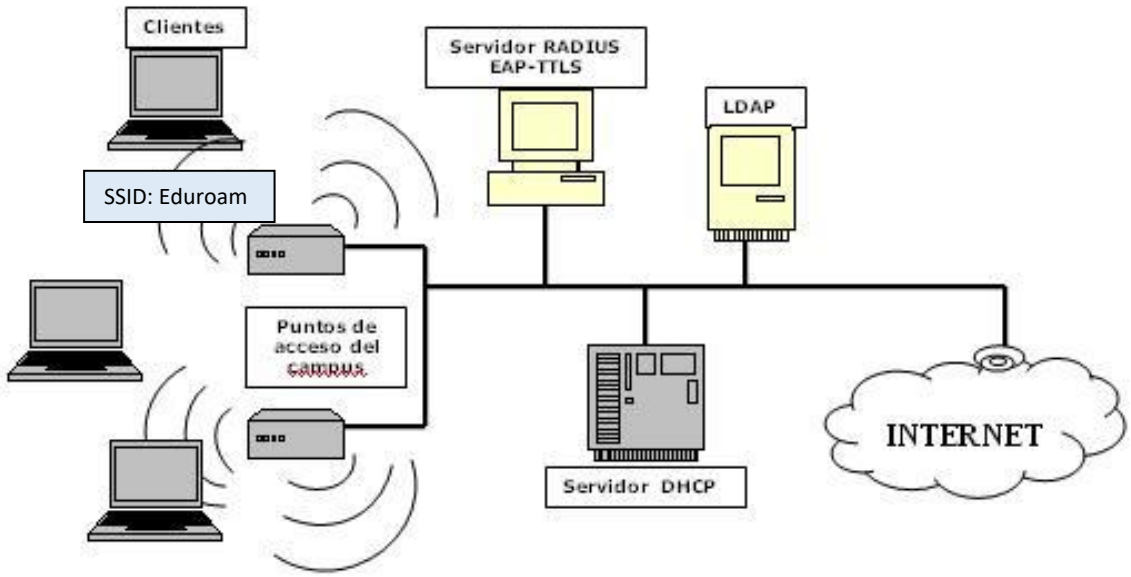



*Figura 7. Esquema simplificado de una Red Wifi con Eduroam*

## 2.3 Protocolo SNMP y MIB

### 2.3.1 Definición SNMP

SNMP, Simple Network Management Protocol o en español Protocolo simple de administración de redes es un protocolo de red que se utiliza para la administración y

monitorización de dispositivos conectados a la red en redes IP. El protocolo SNMP está integrado en los dispositivos locales como enrutadores, conmutadores, servidores, cortafuegos y puntos de acceso inalámbricos accesibles mediante su dirección IP. SNMP proporciona un mecanismo común para que los dispositivos de red transmitan información de gestión dentro de entornos LAN o WAN de uno o varios proveedores. Es un protocolo de capa de aplicación en el marco del modelo OSI.

Normalmente, el protocolo SNMP se implementa mediante el Protocolo de datagramas de usuario (UDP), donde UDP es un protocolo sin conexión que funciona como el Protocolo de control de transmisión (TCP), pero asume que no se requieren servicios de recuperación y verificación de errores.

Las bases de información de administración de SNMP (llamadas MIB para abreviar) son estructuras de datos que definen qué se puede recopilar del dispositivo local y qué se puede cambiar y configurar. Hay muchas MIB definidas por organismos de normalización como IETF e ISO, así como MIB patentadas definidas por proveedores de equipos de TI específicos como Cisco o Aruba y proveedores de software como Microsoft y Oracle.

Hay tres versiones diferentes de SNMP:

- SNMP versión 1 (SNMPv1): esta fue la primera implementación, que opera dentro de la especificación de información de administración de estructura y se describe en RFC 1157.
- SNMP versión 2 (SNMPv2): esta versión se mejoró para admitir un manejo de errores más eficiente y se describe en RFC 1901. Se introdujo por primera vez como RFC 1441. A menudo se denomina SNMPv2c.
- SNMP versión 3 (SNMPv3): esta versión mejora la seguridad y la privacidad. Fue introducido en RFC 3410.

La versión 2 de SNMP es la versión del protocolo SNMP más comúnmente implementada en la actualidad. La versión más reciente, SNMP versión 3, incluye nuevas características de seguridad que agregan soporte para la autenticación y encriptación de mensajes SNMP, así como también protegen los paquetes durante el tránsito.

### 2.3.2 SNMP Runtime Components

Estos son los principales componentes de tiempo de ejecución en un entorno habilitado para SNMP:

- Dispositivos y recursos administrados por SNMP: estos son los dispositivos y elementos de red en los que se ejecuta un agente.
- Agente SNMP: este software se ejecuta en el hardware o servicio que SNMP supervisa y recopila datos sobre diversas métricas, como el uso de la CPU, el uso del ancho de banda o el espacio en disco. Según lo consultado por el administrador de SNMP, el agente encuentra y envía esta información a los sistemas de administración de SNMP.

- Administrador SNMP: (también denominado servidor SNMP) Este componente funciona como una estación de administración centralizada que ejecuta una aplicación de administración SNMP en muchos entornos de sistemas operativos diferentes. Solicita activamente a los agentes que envíen actualizaciones SNMP a intervalos regulares.
- Base de información de administración (MIB): esta estructura de datos es un archivo de texto (con una extensión de archivo .mib) que describe todos los objetos de datos utilizados por un dispositivo en particular que se pueden consultar o controlar mediante SNMP, incluido el control de acceso. Dentro de la MIB hay muchos objetos gestionados diferentes que pueden identificarse mediante identificadores de objeto. Un identificador de objeto (OID) es un identificador de MIB que se utiliza para delinear entre dispositivos dentro de la MIB. Los OID se generan de forma única como identificadores numéricos que se utilizan para acceder a los objetos MIB.

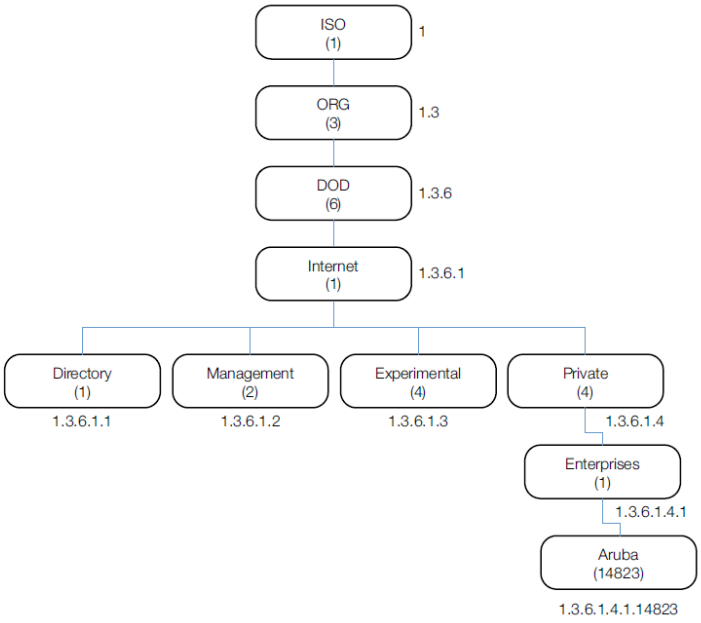



*Figura 8. Estructura MIB Aruba [2]*

En funcionamiento, el Protocolo simple de administración de red utiliza uno o varios administradores SNMP administrativos, que supervisan grupos de computadoras en red y dispositivos asociados. Un programa de software que se ejecuta continuamente, llamado agente, proporciona información a los administradores a través de SNMP. Los agentes crean variables a partir de los datos y las organizan en jerarquías descritas por las MIB.

SNMP es uno de los protocolos de la industria de redes más implementados y es compatible con una variedad de hardware, desde elementos de red comunes como enrutadores, conmutadores y puntos de acceso inalámbricos hasta puntos finales como impresoras, escáneres y

dispositivos de Internet de las cosas (IoT). Además del hardware, el software SNMP se puede utilizar para monitorear los servicios de configuración del Protocolo de configuración dinámica de host (DHCP).

Si bien SNMP se usa en una red de cualquier tamaño, su mayor valor es cuando se usa en redes más grandes. Al usar SNMP, un administrador de red podrá administrar y monitorear todos los dispositivos SNMP desde una única interfaz.

### 2.3.3 Comandos SNMP

Las herramientas SNMP realizan muchas funciones que se basan en una combinación de comunicaciones push y pull entre los dispositivos de red y el sistema de gestión de la red. En su conjunto básico de funciones, puede ejecutar comandos de lectura o escritura, cómo restablecer una contraseña o cambiar una configuración. También puede encontrar cuánto ancho de banda de red, CPU y memoria están en uso. Algunos administradores de SNMP pueden enviar automáticamente al administrador una alerta por correo electrónico o mensaje de texto si se excede un umbral predefinido. Las siguientes PDU, o unidades de datos de protocolo, describen los comandos de mensajería admitidos por el protocolo:

- Get Request: una solicitud para recuperar el valor de una variable o lista de variables.
- Set Request: enviada por el administrador de SNMP al agente para que emita configuraciones o comandos.
- GetNextRequest: enviada por el administrador SNMP al agente para encontrar los valores del siguiente registro en la jerarquía de la MIB.
- GetBulk Request: enviada por el administrador de SNMP al agente para obtener tablas de datos grandes mediante la ejecución de varios comandos de solicitud GetNext.
- SNMP Response: enviada por el agente al administrador SNMP
- SNMP Trap: los mensajes de captura asíncrona de los agentes SNMP alertan al administrador SNMP de que ha ocurrido un evento significativo, como un error.
- SNMP Inform: confirma la recepción de un Trap.

### 2.3.4 Puertos SNMP

Los puertos SNMP se utilizan a través de UDP 161 para los administradores SNMP que se comunican con los agentes SNMP (es decir, sondeo) y UDP 162 cuando los agentes envían capturas no solicitadas al administrador SNMP. [1]

# Capítulo 3. Diseño e implementación sistema de captación de datos Wifi

De forma general nuestro sistema se basa en dos elementos:

- Red Wifi: formada por una red principal de puntos de acceso gestionados por una controladora que se encarga de recibir la información de cada uno de los puntos de acceso. A su vez, la información que envían estos puntos de acceso es recolectada de cada uno de los usuarios que se registran en un SSID.

-Servidor: Donde se está ejecutando un programa en Python que consulta mediante el protocolo SNMP a la controladora para que le envíe la información que recibe de los usuarios conectados a la red. Estos datos se guardan diariamente en un archivo .csv para su posterior procesado.

Para el desarrollo de este programa en Python ha sido necesario conocer cómo funciona una arquitectura WIFI, cuáles son sus elementos y que tipo de protocolos se utilizan para el envío de datos. Para ello hemos buscado en la documentación oficial de Aruba ayudándonos de los libros para las certificaciones oficiales de gestión y mantenimiento de una red wifi CWNA (Certified Wireless Network Administrator).

Una vez teníamos toda esta información creamos la primera consulta SNMP en Python para observar empíricamente como nos devolvía los datos, para posteriormente hacer consultas de manera automática y guardarlas en vectores por usuario.

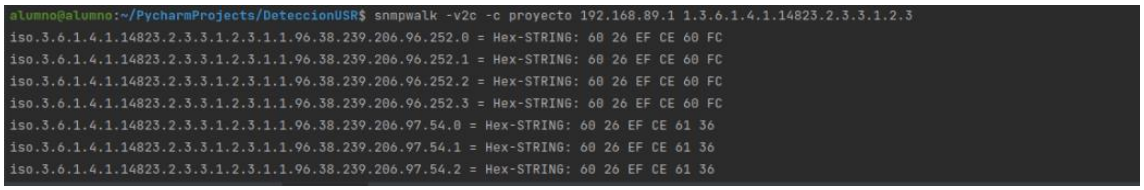



*Figura 9. Extracción de datos de la primera consulta SNMP por cli*

## 3.1 Arquitectura lógica del sistema de monitorización

Mediante el programa de monitorización para la adquisición de datos de usuario en tiempo real conseguimos obtener la información que se genera a partir de cada usuario conectado en la red wifi en intervalos de tiempo de 3 minutos.

En el siguiente diagrama se muestra cada uno de los pasos que ocurren hasta obtener los datos en nuestro servidor.

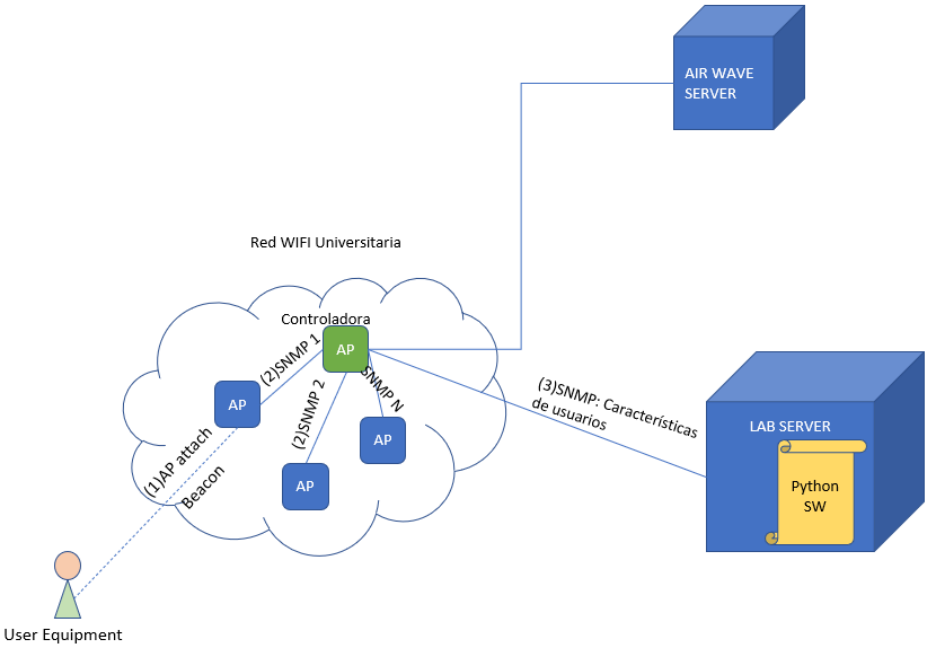



*Figura 10. Arquitectura del sistema de captación de datos.*

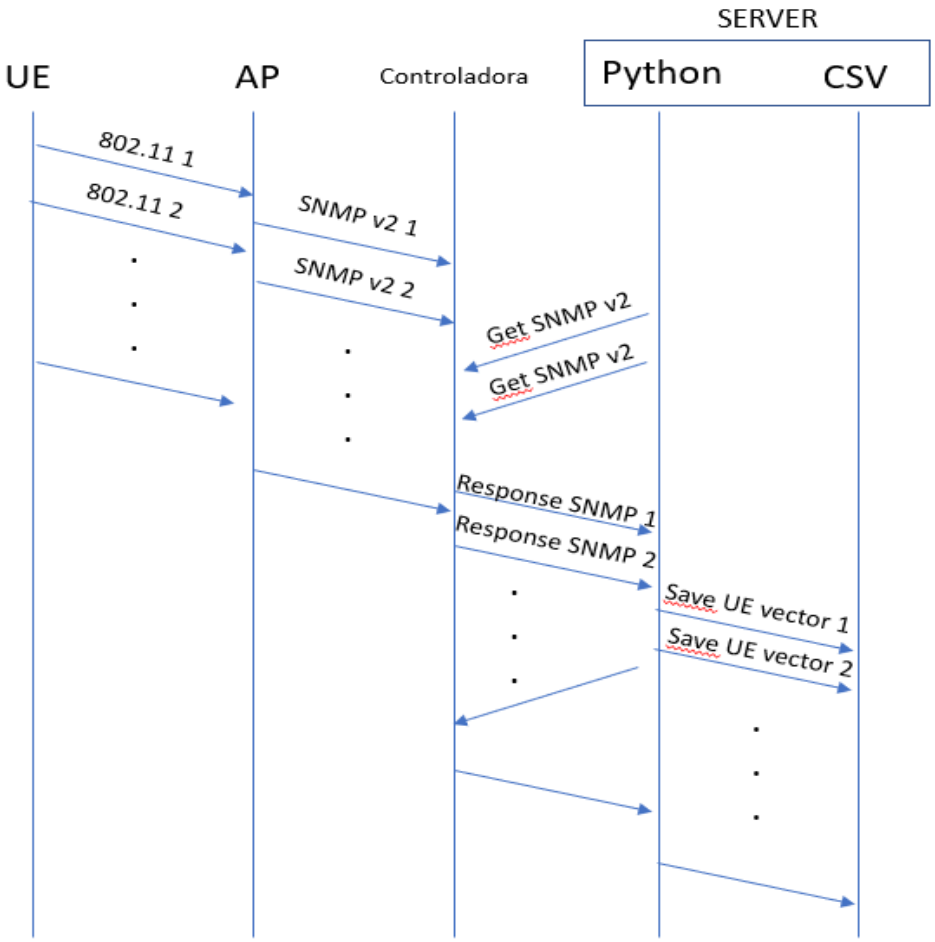



*Figura 11. Cronograma alto nivel con la secuencia de mensajes*

Tenemos una red WIFI con un área de cobertura que abarca cada uno de los edificios que forman parte de la Universidad Politécnica de Cartagena. Esta red WIFI esta compuesta por multitud de puntos de acceso los cuales son gestionados por un punto de acceso master o controladora, esto es así debido a que el software de gestión que se utiliza en la universidad es del fabricante Aruba y utiliza un punto de acceso como referente para el resto de equipos en vez de tener un software externo cargado en otro servidor y al cual apuntarían todos los puntos de acceso como si ocurre con otros fabricantes como Extreme Network o Cisco.

A esta red se conectan cientos de dispositivos diariamente ya sean smartphone tablets, portátiles, televisiones o raspberries cada uno con el protocolo de conexión wifi que soporte 802.11_ _. Cuando estos equipos se conectan a los diferentes puntos de acceso se envían cantidad de mensajes que podríamos dividir en 3 Conexión, Navegación y desconexión. Cuando el terminal se conecta a uno de los puntos de acceso y empieza a navegar ese punto de acceso recolecta toda la información del usuario y se la envía al punto de acceso que funciona como controladora para que tenga la información de dicho terminal y pueda crear sus propias estadísticas para ver el comportamiento de la red. Estos datos son los que nuestro programa en Python pregunta a la controladora mediante el protocolo SNMP haciendo referencia al OID de las consultas que quiera hacerle. Una vez la controladora recibe cada una de las consultas esta va respondiendo al servidor el cual coge la información la procesa y la almacena en un archivo .csv para su posterior estudio. El programa Python realiza las consultas cada 3 min principalmente para no saturar a la controladora y evitar provocar una caída de red.

## 3.2 Consultas SNMP a la controladora Aruba

“La función principal de un controlador de red local inalámbrica tradicional (WLC) consiste en la configuración de los puntos de acceso inalámbricos (AP). Pero, en principio, puede tener muchas otras aplicaciones.

Debido a las características propias de la tecnología Wi-Fi y las arquitecturas de red inalámbrica que se han utilizado tradicionalmente, así como a las propias estrategias comerciales de los fabricantes, un WLC tradicional puede realizar muchas otras funciones.

- Gestión y operación

Son las tareas que permiten desplegar y operar la red local inalámbrica de una manera unificada y sencilla, sin necesidad de tener que repetir las mismas operaciones en todos y cada uno de los AP de la red local. Estas funciones permiten configurar, monitorizar y diagnosticar problemas en la red. También tienen la ventaja de permitir enviar y recibir alarmas cuando se detecta algún problema. Para un funcionamiento

homogéneo de la red, incluimos en este grupo la actualización del software de funcionamiento de los AP, para que todos tengan la misma versión." [10]

"En contraste con Aruba OS 6, el cual opera sobre un modelo plano de configuración que contiene la configuración global y local, Aruba OS 8 utiliza una arquitectura centralizada de múltiples niveles bajo una nueva Interface de Usuario que proporciona una clara separación entre las funciones de administración, control y reenvío. La configuración completa para el Mobility Conductor y para los dispositivos administrados se efectúa desde una ubicación centralizada– proporcionando mejor visibilidad y monitoreo, así como simplificando y optimizando el proceso de configuración y minimizando repeticiones.

- Seguridad

Para asegurar la red empresarial, Aruba OS 8 efectúa autenticación, control de acceso y cifrado para usuarios y dispositivos. Con la arquitectura de Aruba, la autenticación es estándar y se puede implementar para redes alámbricas e inalámbricas. Para redes inalámbricas, 802.1X es un componente de los protocolos WPA2 y 802.11i ampliamente reconocidos como el estado del arte para seguridad Wi-Fi." [9]

Como ejemplo grafico de los datos que recibe la controladora del resto de puntos de acceso podemos ver la siguiente tabla cogida de la interfaz gráfica de esta.

| Name | IP Address | MAC address | OS | ESSID | Access Point | Channel | Type | Role | IPv6 Address | Signal | Speed (Mbps) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| [redacted] | 192.168.251.... | e0:cc:f8:5e:c... | Linux | eduroam | ETSII-P1-S-EE | 1 | GN | eduroam | 2001:4610:4b... | 0 | 0 |
| | 192.168.62.191 | f8:a2:d6:84:e... | Win 10 | Open-UPCT | ANT-PB-S-E | 100E | AC | Open-U... | fe80::9d2f:17... | 48 | 390 |
| | 192.168.250.... | 66:62:92:e4:b... | OS X | eduroam | ANT-P1-S-O | 6 | GN | eduroam | 2001:4610:4b... | 24 | 117 |
| | 192.168.250.... | 32:ee:69:bb:0... | Linux | eduroam | IDI-PB-E | 36E | AC | eduroam | 2001:4610:4b... | 5 | 6 |
| | 192.168.61.40 | 3c:22:fb:00:3... | Apple | Open-UPCT | Eldi-blqO-p2-N | 11 | GN | Open-U... | 2001:4610:4b... | 22 | 78 |
| | 192.168.102.... | 00:13:ef:4f:11... | NOFP | IoTUT | ETSII-PS-Aula4 | 116E | AC | IoTUT | 2001:4610:4b... | 43 | 390 |
| | 192.168.251.14 | f0:6e:0b:d5:4... | Win 10 | eduroam | ANT-PB-S-E | 100E | AC | eduroam | 2001:4610:4b... | 50 | 866 |

*Figura 12. Muestra de datos de la controladora Aruba en la versión 8*

En la captura anterior podemos ver que se consiguen vectores con las características de cada usuario, aunque no vemos el tiempo que ha estado conectado a la red simplemente obtenemos en tiempo real la característica de cada usuario. Debido a que no tenemos un report de los usuarios de la red diario tenemos que hacer consultas SNMP a la controladora cada 3min y guardarlo en un fichero para su posterior estudio.

La controladora Aruba versión de software 6.2.0.0 (igual que la nueva versión Aruba OS 8) tenemos cuatro grandes bloques de MIBs las cuales vamos a estudiar y valorar su utilidad para más tarde poder hacer la consulta de las peticiones que nos

sean realmente interesantes para nuestro objetivo. En la siguiente tabla vemos un esquema global de la MIBs que se pueden interrogar:

| Group | Description |
|---|---|
| aiAccessPointTable | Contains all the access points connected to the virtual controller. This table is indexed by the MAC Address of the IAP. |
| aiRadioTable | Contains all the radios of the access points connected to the virtual controller. This table is indexed by the MAC Address and radio number. |
| aiWlanTable | Contains all the BSSIDs that are active on the virtual controller. This table is indexed by the MAC address and a WLAN Index of the IAP. |
| aiClientTable | Contains information about all the clients connected to the virtual controller. When a client roams from one access point to another, all the counters in this table are reset to 0. |

*Tabla 1. MIB soportadas por Aruba*

Con esta información de cada uno de los bloques debemos de seleccionar cuales de ellos nos serán necesarios para obtener todos los datos de interés que necesitamos para poder generar las tablas con los valores que nos ayudarán a obtener el número de usuarios por aula.

En primer lugar, nos encontramos los datos de los puntos de Acceso o AP (Access Point):

| Object | Object ID | |
|---|---|---|
| aiAccessPointEntry | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1 | aiAccessPointTable 1 |
| aiAPMACAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.1 | aiAccessPointEntry 1 |
| aiAPName | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.2 | aiAccessPointEntry 2 |
| aiAPIPAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.3 | aiAccessPointEntry 3 |
| aiAPSerialNum | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.4 | aiAccessPointEntry 4 |
| aiAPModel | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.5 | aiAccessPointEntry 5 |
| aiAPModelName | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.6 | aiAccessPointEntry 6 |
| aiAPCPUUtilization | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.7 | aiAccessPointEntry 7 |
| aiAPMemoryFree | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.8 | aiAccessPointEntry 8 |
| aiAPUptime | 1.3.6.1.4.1.14823.2.3.3.1.2.1.1.9 | aiAccessPointEntry 9 |

*Tabla 2. MIBs para las características del Punto de Acceso*

Estos datos no varían a excepción de los tres últimos (debido a que son los parámetros de utilización del AP y conforme varíe la carga del AP así variarán los parámetros de memoria y uso de CPU) o en el caso de que algún AP nuevo sea añadido

o por el contrario se caiga. En nuestro caso tenemos un total de 62 puntos de acceso y los datos que obtenemos están almacenados en el documento tablas_estaticas_Wifi.xlsx en la hoja con el nombre DATOS Puntos de ACCESO.

Estos datos serán útiles simplemente para realizar un inventario de la red dándonos información de los puntos de acceso que hay y ayudándonos a identificar cuáles son lo que podremos bajo estudio.

El siguiente bloque de MIBs son las correspondientes a los parámetros Radio de los APs (aiRadioTable)

| Object | Object ID | |
|---|---|---|
| aiRadioEntry | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1 | aiRadioTable 1 |
| aiRadioAPMacAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.1 | aiRadioEntry 1 |
| aiRadioIndex | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.2 | aiRadioEntry 2 |
| aiRadioMACAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.8 | aiRadioEntry 3 |
| aiRadioChannel | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.4 | aiRadioEntry 4 |
| aiRadioTransmitPower | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.5 | aiRadioEntry 5 |
| aiRadioNoiseFloor | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.6 | aiRadioEntry 6 |
| aiRadioUtilization4 | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.7 | aiRadioEntry 7 |
| aiRadioUtilization64 | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.8 | aiRadioEntry 8 |
| aiRadioTxTotalFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.9 | aiRadioEntry 9 |
| aiRadioTxMgmtFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.10 | aiRadioEntry 10 |
| aiRadioTxDataFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.11 | aiRadioEntry 11 |
| aiRadioTxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.12 | aiRadioEntry 12 |
| aiRadioTxDrops | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.13 | aiRadioEntry 13 |
| aiRadioTxTotalFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.14 | aiRadioEntry 14 |
| aiRadioRxDataFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.15 | aiRadioEntry 15 |
| aiRadioRxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.16 | aiRadioEntry 16 |
| aiRadioRxMgmtFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.17 | aiRadioEntry 17 |
| aiRadioRxBad | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.18 | aiRadioEntry 18 |
| aiRadioPhyEvents | 1.3.6.1.4.1.14823.2.3.3.1.2.2.1.19 | aiRadioEntry 19 |

*Tabla 3. MIBs para los parámetros Radio de los Puntos de Acceso*

En esta MIB vamos a ir identificando cada uno de los valores que podemos obtener de manera individual puesto que alguno de ellos puede resultarnos de utilidad.

- aiRadioAPMacAddress: Con esta MIBs obtenemos la dirección MAC del AP en cada uno de los OID de las Radios, como se muestra en la siguiente tabla.

| |
|---|
| iso.3.6.1.4.1.14823.2.3.3.1.2.2.1.1.24.100.114.193.212.12.0 = Hex-STRING: 18 64 72 C1 D4 0C |
| iso.3.6.1.4.1.14823.2.3.3.1.2.2.1.1.24.100.114.193.212.12.1 = Hex-STRING: 18 64 72 C1 D4 0C |
| iso.3.6.1.4.1.14823.2.3.3.1.2.2.1.1.24.100.114.193.212.152.0 = Hex-STRING: 18 64 72 C1 D4 98 |
| iso.3.6.1.4.1.14823.2.3.3.1.2.2.1.1.24.100.114.193.212.152.1 = Hex-STRING: 18 64 72 C1 D4 98 |

*Tabla 4. Ejemplo de datos de la MIBs para obtener las vMAC de las radios*

- aiRadioIndex : Con esta MIB identificamos cada Radio de cada AP y se identifica con un 0 y un 1.

- aiRadioMACAddress: Con esta MIB identificamos cada una de las Radios con una MAC distinta.

- aiRadioChannel : Con esta MIB sabemos en qué canal está operando cada una de las radios de cada uno de los APs.

- aiRadioTransmitPower :Con esta MIB obtenemos la potencia con la que transmite cada una de las radios del AP

- aiRadioNoiseFloor: Con esta MIB obtenemos el ruido de fondo que obtenemos por cada una de las radios

  El resto de parámetros de esta MIB no nos ofrece información relevante.

La siguiente MIB que vamos a analizar es aiWLANTable con la que podemos obtener:

| Object | Object ID | |
|---|---|---|
| aiWlanEntry | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1 | aiWlanTable 1 |
| aiWlanAPMACAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.1 | aiWlanEntry 1 |
| aiWlanIndex | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.2 | aiWlanEntry 2 |
| aiWlanESSID | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.3 | aiWlanEntry 3 |
| aiWlanMACAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.4 | aiWlanEntry 4 |
| aiWlanTxTotalFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.5 | aiWlanEntry 5 |
| aiWlanTxDataFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.6 | aiWlanEntry 6 |
| aiWlanTxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.7 | aiWlanEntry 7 |
| aiWlanRxTotalFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.8 | aiWlanEntry 8 |
| aiWlanRxDataFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.9 | aiWlanEntry 9 |
| aiWlanRxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.3.1.10 | aiWlanEntry 10 |

*Tabla 5. MIBs de las características de las WLAN por AP*

De estos OIDs solo vamos a necesitar la entrada 1 y la 4 que son las que nos ofrecen la MAC del AP en referencia a la WLAN que se está transmitiendo y la MAC de cada uno de las WLAN que se transmiten en 2.4 y 5 GHz.

Estas tablas van a ser estáticas puesto que estos valores de MAC no cambian con el tiempo por lo que podremos generar una tabla que relacione las dos entradas y que posteriormente nos servirá para identificar si un usuario está trabajando en la banda de 2.4 o 5GHz.

La tabla correspondiente se encuentra ubicada en el archivo tablas_estaticas_Wifi.xlsx en la hoja con el nombre Relación WLAN con Frecuencia.

El resto de OID no nos aportan información relevante acerca de los usuarios por lo tanto no se requiere de su explicación.

Por último, la MIB más importante para el proyecto que nos ocupa es aiClientTable, la cual nos muestra cada uno de los parámetros que se pueden obtener de los usuarios.

| Object | Object ID | |
|---|---|---|
| aiClientTable Entry | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1 | aiClientTable 1 |
| aiClientMACAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.1 | aiClientEntry 1 |
| aiClientWlanMACAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.2 | aiClientEntry 2 |
| aiClientIPAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.3 | aiClientEntry 3 |
| aiClientAPIPAddress | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.4 | aiClientEntry 4 |
| aiClientTxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.5 | aiClientEntry 5 |
| aiClientTxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.6 | aiClientEntry 6 |
| aiClientTxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.7 | aiClientEntry 7 |
| aiClientRxDataFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.8 | aiClientEntry 8 |
| aiClientTxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.9 | aiClientEntry 9 |
| aiClientRxRetries | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.10 | aiClientEntry 10 |
| aiClientTxRate | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.11 | aiClientEntry 11 |
| aiClientRxDataFrames | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.12 | aiClientEntry 12 |
| aiClientRxDataBytes | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.13 | aiClientEntry 13 |
| aiClientRxRetries | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.14 | aiClientEntry 14 |
| aiClientRxRate | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.15 | aiClientEntry 15 |
| aiClientUptime | 1.3.6.1.4.1.14823.2.3.3.1.2.4.1.16 | aiClientEntry 16 |

*Tabla 6. MIBs de las características de los usuarios conectados a los Puntos de Acceso*

- MAC Cliente: Dirección MAC del dispositivo del cliente

- MAC WLAN: Dirección MAC de la Wireless LAN a la que esté conectado el cliente. En nuestro caso tendremos dos: OPEN-UPCT y EDUROAM. Debemos saber que esta MAC WLAN variará si el usuario se ha conectado en la red de 2.4 o de 5GHz y lo podremos saber mediante la tabla tablas_estaticas_Wifi.xlsx en la hoja con el nombre Relación WLAN con Frecuencia.

- IP Cliente: Dirección IP del cliente

- IP AP conectado: Dirección IP del AP al que está conectado el cliente.

- Nombre del Cliente: Nombre del usuario con el que se haya registrado.

- SO: sistema operativo del dispositivo

- SNR: Relación Señal a Ruido (dB)

## 3.3 Programa Python de automatización consultas

Explicación del programa de una parte, incluir y explicar módulos en los que se descompone una parte del programa, diagrama flujo seudocódigo, formato y tipo de mensajes entre módulos, etc.

En nuestro programa vamos a realizar consultas SNMP GETBULK para obtener información de los usuarios. El diagrama de flujo de la figura 3 muestra el funcionamiento del programa.

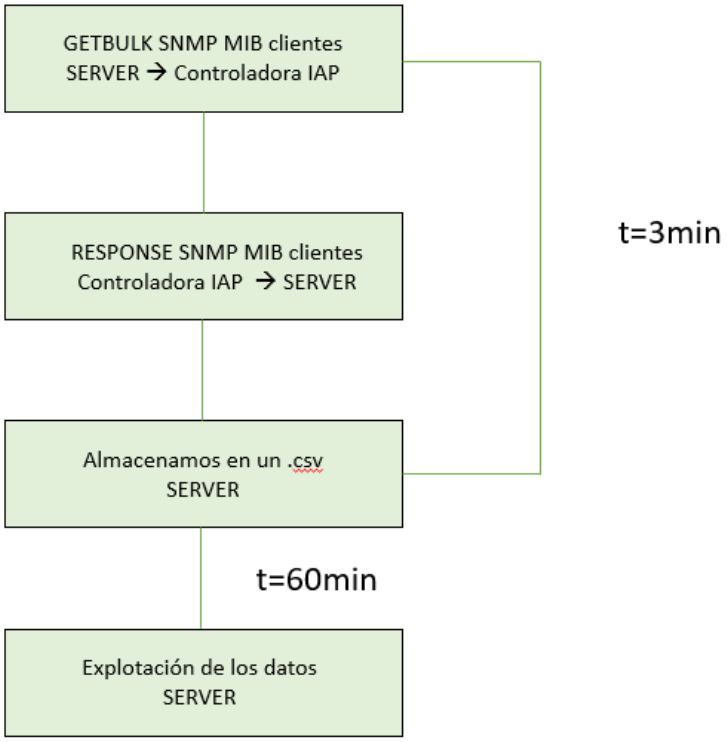



*Figura 13. Diagrama flujo sensor Maestro*

Para el desarrollo de nuestro programa para interrogar a la controladora de puntos de acceso a través de un programa en Python utilizaremos la librería que nos ofrece este lenguaje de programación que se denomina PySNMP. Para ellos como primer paso deberemos importar esta característica a nuestro entorno.

```
from pysnmp import hlapi
from pysnmp.entity.rfc3413.oneliner import cmdgen
```

Para hacer uso de esta librería deberemos instalarla en primer lugar:

>> pip install pysnmp

De esta manera cuando la importemos para hacer uso de sus características en nuestro programa no nos dará problemas.

Nuestro programa se podría dividir en 3 secciones:

1. Funciones necesarias para realizar las consultas SNMP.

   En esta sección encontramos la definición de las funciones que más tarde usaremos para interrogar a la controladora.

   La función principal la hemos llamado GET:

```
def get(target, oids, credentials, port=161, engine=hlapi.SnmpEngine(), context=hlapi.ContextData()):
    handler = hlapi.getCmd(
        engine,
        credentials,
        hlapi.UdpTransportTarget((target, port)),
        context,
        *construct_object_types(oids)
    )
    return fetch(handler, 1)[0]
```

   Esta función permite obtener una lista de objetos dentro de una MIB. Para la función GET hacemos uso de la API de Python para SNMP que se denomina hlapi-High Level API. Para esta función debemos de pasarle los siguientes parámetros fundamentales:

   - Target: Es la dirección IP del dispositivo que queremos interrogar, en nuestro caso es la controladora Aruba la cual usa la siguiente dirección IP- 192.168.89.1
   - OID: como ya hemos definido en el capítulo 2 Object ID es el identificador del objeto de la MIB por el que queremos consultar.
   - Credencial: La comunidad SNMP que estamos utilizando para que cuando hagamos las consultas y la controladora nos vea pueda reconocernos como un equipo amigo.
   - Port: es el puerto UDP para SNMP, en nuestro caso es el 161 pero podría darse el caso que hubiese sido modificado por temas de seguridad y deberíamos de poner el que corresponda.

Para la función Get se pueda ejecutar de manera correcta se necesita hacer uso de otras dos funciones como son construct_object_types y fech

```
def construct_object_types(list_of_oids):
    object_types = []
    for oid in list_of_oids:
        object_types.append(hlapi.ObjectType(hlapi.ObjectIdentity(oid)))
    return object_types
```

Tambien necesitamos la funcion fetch la cual se encarga de preguntar tantas veces como valor haya en el count, en nuestro caso count sera el numero de usuarios

que haya conectados en la red cada vez que nosotros interroguemos pero eso lo explicaremos en la siguiente sección.

```
def fetch(handler, count):
    result = []
    for i in range(count):
        try:
            error_indication, error_status, error_index, var_binds = next(handler)
            if not error_indication and not error_status:
                items = {}
                for var_bind in var_binds:
                    items[str(var_bind[0])] = cast(var_bind[1])

                result.append(items)
            else:
                raise RuntimeError('Got SNMP error: {0}'.format(error_indication))
        except StopIteration:
            break
    return result
```

La función fetch nos indica cuando hay algún error en la consulta y detiene el programa.

Por último, necesitamos crear la función cast que es la que nos convierte los datos recibidos en int, float o strign.

```
def cast(value):
    try:
        return int(value)
    except (ValueError, TypeError):
        try:
            return float(value)
        except (ValueError, TypeError):
            try:
                return str(value)
            except (ValueError, TypeError):
                pass
    return value
```

Como hemos explicado en la función fetch necesitamos indicarle cual es el valor de count/número de iteraciones que tiene que realizar puesto que al querer obtener información de todos los usuarios no nos valdría solo 1 por tanto vamos a crear una función Get que nos pida como valor de entrada el número de iteraciones que tiene que realizar.

```
def get_bulk(target, oids, credentials, count, start_from=0, port=161,
             engine=hlapi.SnmpEngine(), context=hlapi.ContextData()):
    handler = hlapi.bulkCmd(
        engine,
        credentials,
        hlapi.UdpTransportTarget((target, port)),
        context,
        start_from, count,
        *construct_object_types(oids)
    )
    return fetch(handler, count)
```

2. Función principal de interrogación.

En este punto vamos a explicar cada uno de los comandos que hemos utilizado para interrogar a la controladora con las consultas OID que nos interesan haciendo uso de la función explicada en el punto anterior Get_Bulk.

Pero antes de usar la función Get_Bulk necesitamos saber cuántos usuarios hay conectados en cada instante que lanzamos la consulta a la controladora para ello nos ayudamos de otra función que nos muestra las direcciones IP que tiene la controladora, que serán guardas en un data frame para usarse más tarde y generar los vectores de cada usuario.

```
    ip="192.168.89.1"
    community="proyecto"


    generator = cmdgen.CommandGenerator()
    comm_data = cmdgen.CommunityData("server", community, 1) # 1 means version SNMP v2c
    transport = cmdgen.UdpTransportTarget((ip, 161))

    real_fun = getattr(generator, "nextCmd")
#IP usuario
    oidipusr=(1,3,6,1,4,1,14823,2,3,3,1,2,4,1,3)
    IPusr = (errorIndication, errorStatus, errorIndex, varBinds) = real_fun(comm_data, transport, oidipusr)
    if not errorIndication is None  or errorStatus is True:
        print ("Error: %s %s %s %s" % IPusr)
    else:
        c = int(1)
        n = len(varBinds)+1
        IPuser = [None] * n
        IPuser[0] = 0

        for var_bind in varBinds:
            IPuser[c] =[x.prettyPrint() for x in var_bind]
            c=c+1
```

Esta función real_fun es otra API que tiene Python para interrogar por SNMP y nos guarda todo lo que le responde la controladora en el data frame IPusr. Con este la longitud de este dataframe ya podemos utilizar la función getbulk y consultar a la controladora todos los OID que nos interesen para nuestro estudio.

```
nentradas = len(IPuser)
Name = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.5'], hlapi.CommunityData('proyecto'), nentradas)
SO = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.6'], hlapi.CommunityData('proyecto'), nentradas)
ruido = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.7'], hlapi.CommunityData('proyecto'), nentradas)
TXDataFr = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.8'], hlapi.CommunityData('proyecto'), nentradas)
TXDataBy = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.9'], hlapi.CommunityData('proyecto'), nentradas)
TXRetries = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.10'], hlapi.CommunityData('proyecto'), nentradas)
TXRate = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.11'], hlapi.CommunityData('proyecto'), nentradas)
RXDataFr = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.12'], hlapi.CommunityData('proyecto'), nentradas)
RXDataBy = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.13'], hlapi.CommunityData('proyecto'), nentradas)
RXRetries = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.14'], hlapi.CommunityData('proyecto'), nentradas)
RXRate = get_bulk('192.168.89.1', ['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.15'], hlapi.CommunityData('proyecto'), nentradas)
```

Y para cada uno de los datos que recibimos lo convertimos eliminando la información que no nos es útil también para que sea más fácil de crear los vectores de usuario. Para ello, usamos el siguiente bucle que nos deja únicamente el valor útil que se obtiene de la controladora.

iso.3.6.1.4.1.14823.2.3.3.1.2.4.1.1.0.23.196.116.86.45 = Hex-STRING: 00 17 C4 74 56 2D

Únicamente nos guardaría el valor de la MAC o en el ejemplo el nombre de usuario.

```
# Nombre del usuario
c = int(1)
n = int(nentradas + 1)
nombre = [None] * n
nombre[0] = 0

for it in Name:
    for k, v in it.items():
        nombre[c] = "{1}".format(k, v)
    c = c + 1
```

Este condigo se repite para todas las consultas SNMP que realizamos y una vez que ya tenemos todos los vectores con la información de cada OID generemos nuestro data frame con los vectores de cada usuario de la siguiente forma:

```
# Generamos el Dataframe con cada una de las peticiones que hemos realizado al servidor
data = [timestamp, MACUSR, MACW, IPuser,IPAP, nombre, sistema, snr, TXframes, TXBytes, TRetries, TRate, RXframes, RXBytes, RRetries, RRate]

df = pd.DataFrame(data, index=['timestamp', 'MAC','MACWLAN', 'IP', 'IPAP', 'Nombre', 'sistema operativo', 'relacion SN', 'TXDataFr', 'TXDataBy',
                               'TXRetries', 'TXRate', 'RXDataFr', 'RXDataBy', 'RXRetries', 'RXRate'])
df = df.transpose()
df.columns = {'timestamp', 'MAC','MACWLAN', 'IP', 'IPAP', 'Nombre', 'sistema operativo', 'relacion SN', 'TXDataFr', 'TXDataBy',
              'TXRetries', 'TXRate', 'RXDataFr', 'RXDataBy', 'RXRetries', 'RXRate'}
with open('13062022.csv', 'a') as f:
    df.to_csv(f, mode='a', header=False)
```

Y en la última parte es donde creamos nuestro fichero csv y guardamos todos los data frame que recibimos.

3. Ejecución del programa.

Cuando ya tenemos definidas todas nuestras funciones que realizan las consultas SNMP y guardan los datos ya filtrados en vectores dentro de un dataframe y lo almacenan en un archivo csv. Debemos crear un main donde se hagamos una llamada a nuestra función principal y que se repita cada 3 min, lo cual es un tiempo bastante bajo para poder obtener suficiente información de los usuarios, pero sin llegar a saturar la controladora.

```
crearcsv()
print('recolectando datos del servidor: Espere 3 min ')
time.sleep(180)  # 3 min
```

fetch.py

*Figura 14. Fichero con el código completo.*

Para concluir este capítulo mostramos como se ven esos archivos csv que se generan diariamente en tramos de 3 min con los vectores de información de cada usuario y añadimos un fichero csv donde podemos ver la información de un día completo.

2021-10-26 09:19:01
['SNMPv2-SMI::enterprises.14823.2.3.3.1.2.4.1.1.0.8.34.199.46.149 = 0x000822c72e95']
['SNMPv2-SMI::enterprises.14823.2.3.3.1.2.4.1.2.0.8.34.199.46.149 = 0x186472a62a22']
['SNMPv2-SMI::enterprises.14823.2.3.3.1.2.4.1.3.0.8.34.199.46.149 = 192.168.36.143']
['SNMPv2-SMI::enterprises.14823.2.3.3.1.2.4.1.4.0.8.34.199.46.149 = 192.168.57.210']
Nombre de usuario con el que se registra (LOPD)
Linux
40
1831
1250534
28
72
3421
500032
34
65

En el vector anterior los parámetros que vemos en ese mismo orden son los siguientes:

| | |
|---|---|
| Timestamp | Fecha y hora de la muestra |
| MAC del usuario | Dirección MAC del dispositivo del usuario |
| MAC de la WLAN | Dirección MAC virtual de la WLAN a la que se conecta |
| IP del usuario | Dirección que se le asigna a ese usuario |
| IP del punto de acceso | Dirección IP del punto de acceso donde el usuario está ubicado |
| Usuario de registro en la red | Nombre con el que el usuario se ha logado en la red eduroam |
| Sistema Operativo | Sistema operativo que corre el dispositivo |
| Relación SN | Relación señal a ruido entre ese usuario y el punto de acceso que le da servicio |
| TXData frame | Tramas totales que envía el usuario |
| TX Data Byte | Número total de byte transmitidos por el usuario |
| TX Retries | Número total de intentos de transmisión de frames por el usuario |
| TX Rate | Velocidad de transmisión/subida en Mbps |
| RXData frame | Tramas totales que envía el usuario |
| RX Data Byte | Número total de byte recibidos por el usuario |
| RX Retries | Número total de intentos de recepción de frames por el usuario |
| RX Rate | Velocidad de recepción/bajada en Mbps |

*Tabla 7. Definicion de parametros del vector de usuario*

# Capítulo 4. Análisis exploratorio de datos, pruebas y resultados

Para el análisis exploratorio de datos se han tomado dos emplazamientos diferentes:

- Planta Sótano del Antiguo Hospital de Marina. (Actualmente Edificio de la Escuela de Industriales)
- Planta baja del Edificio de Antigones. (Actualmente Edificio de la Escuela de Telecomunicaciones)

Para este análisis exploratorio hemos creado unos scripts de Python que cogen los datos brutos obtenidos de las consultas a la controladora para representarlos gráficamente.

## 4.1 Primer escenario: registro de datos y estudio de las aulas del sótano de Hospital de Marina.

En este primer escenario situado en el edificio de industriales en la planta sótano nos encontramos con una distribución de puntos de acceso, dando cobertura a las 13 clases más la reprografía que se encuentra en el centro del sótano, como se muestra en el siguiente mapa.

*Figura 15. Mapa de calor de los Puntos de Acceso del sótano de H. Marina*

Como se ve en el mapa de calor cada par de aulas recibe la mayor parte de la cobertura de un único punto de acceso, lo cual es un dato importante que nos ayudara a identificar más fácilmente si un aula está ocupada o no, además de poder estimar el número de usuarios que se encuentran en un aula.

Para poder realizar un análisis de esta zona concreta de la red wifi de la universidad vamos a hacer uso de los datos que nos ofrecen estos puntos de acceso los cuales los caracterizamos por su dirección IP.

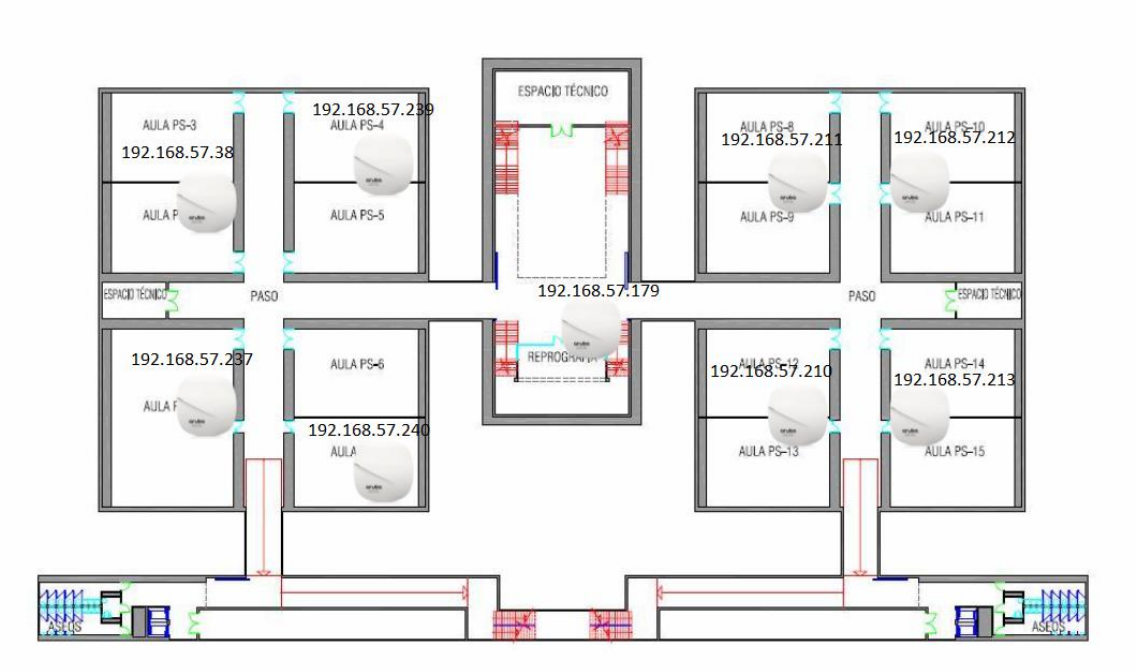



*Figura 16. Mapa de distribución de los Puntos de Acceso del sótano del H. Marina*

Direccionamiento de los puntos de acceso:

| NAME | ADDRESS |
|---|---|
| etsii-sot-ps14/15-ap213 | 192.168.57.213 |
| etsii-sot-ps4/5-ap239 | 192.168.57.239 |
| etsii-sot-ps10/11-ap212 | 192.168.57.212 |
| etsii-sot-ps6/7-ap240 | 192.168.57.240 |
| etsii-sot-ps2/3-ap238 | 192.168.57.238 |
| etsii-sot-ps8/9-ap211 | 192.168.57.211 |
| etsii-sot-ps1-ap237 | 192.168.57.237 |
| etsii-sot-ps12/13-ap210 | 192.168.57.210 |
| etsii-sot-repro-ap179 | 192.168.57.179 |

*Tabla 8. Direcciones de los AP bajo estudio*

Con estos datos podemos empezar nuestro análisis para empezar a caracterizar los datos, para ello se han tomado datos de varias semanas completas en un tramo horario de 8 de la mañana a 22 de la noche para estudiar diferencias entre días de la semana y las variaciones de cantidad de personas en las diferentes franjas horarias.

Para la primera representación hemos calculado la ocupación de las aulas en función de cada punto de acceso y esto lo hacemos gracias a los scripts que encontramos en el Anexo.

- Cantidad de usuarios por punto de acceso:

Días muestreados:


*Figura 17. Fechas de muestreo*

Con estas muestras vamos a observar cómo varía el número de usuarios a lo largo de varios días, primero en días lectivos y en segundo lugar fines de semana:

Días lectivos:

Jueves 31 de marzo podemos ver en la gráfica de que representa el número de usuarios conectados al punto de acceso (192.168.57.211) situado entre las aulas PS-8 y PS-9 que tiene una acumulación de usuarios constante durante dos tramos horarios uno por la mañana y otro por la tarde cada uno de ellos con una duración aproximada de 2 horas completas lo cual corresponde al tiempo que dura una clase, con lo que podríamos estimas que en esas dos aulas había en esos instante alrededor de unas 15 personas en el horario de mañana y alrededor de 20 en el horario de tarde.

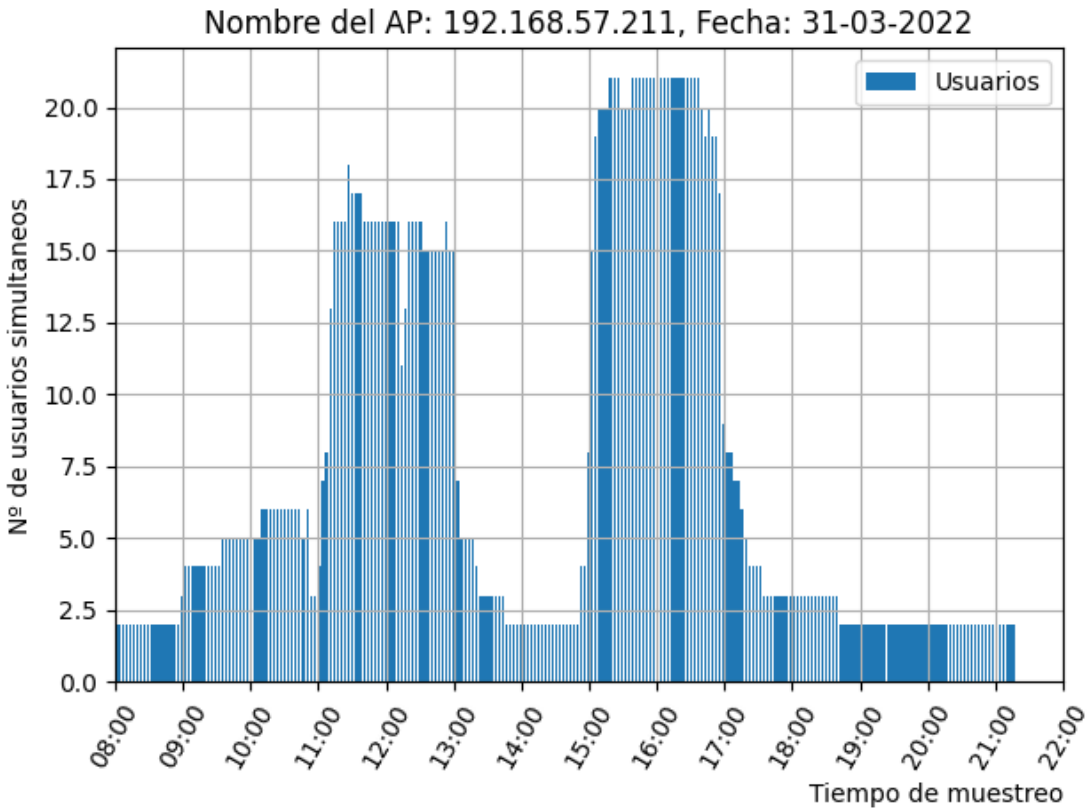



*Figura 18. Grafica de cantidad de usuarios por AP*

Para ese mismo día podemos comprobar otro punto de acceso (192.168.57.239) situado en el ala izquierda del sótano dando servicio a las aulas PS-4 y PS-5

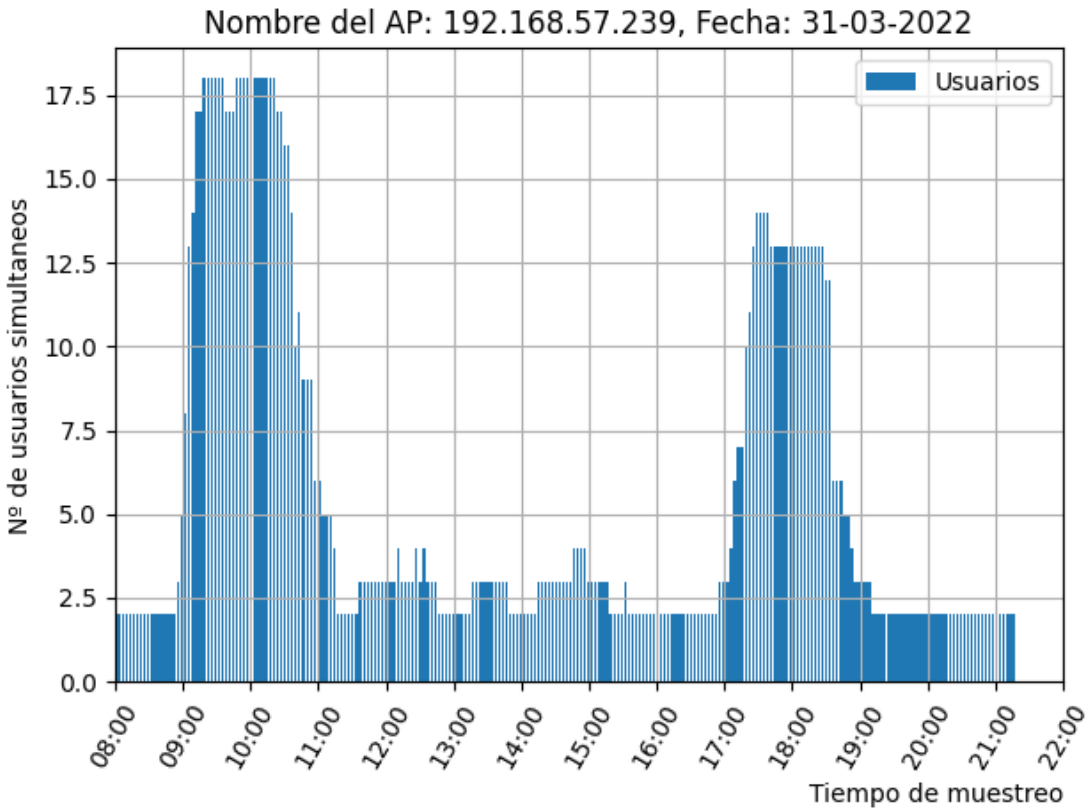



*Figura 19. Grafica de cantidad de usuarios por AP*

En este caso se puede observar un comportamiento muy parecido a lo sucedido en el ala derecha, nos encontramos con dos tramos bien diferenciados uno por la mañana y otro por la tarde donde tenemos una acumulación constante de usuarios durante 2 horas completas, entre medias en el rango de 11 a 5 vemos pequeñas fluctuaciones de usuarios que se conecten a ese punto de acceso simplemente por pasar al lado del aula, pero estas pequeñas variaciones nos son irrelevantes para nuestro estudio.

Fin de semana:

A diferencia de los días lectivos en los fines de semana nos encontramos una situación muy diferente, como se puede observar en las gráficas siguientes:

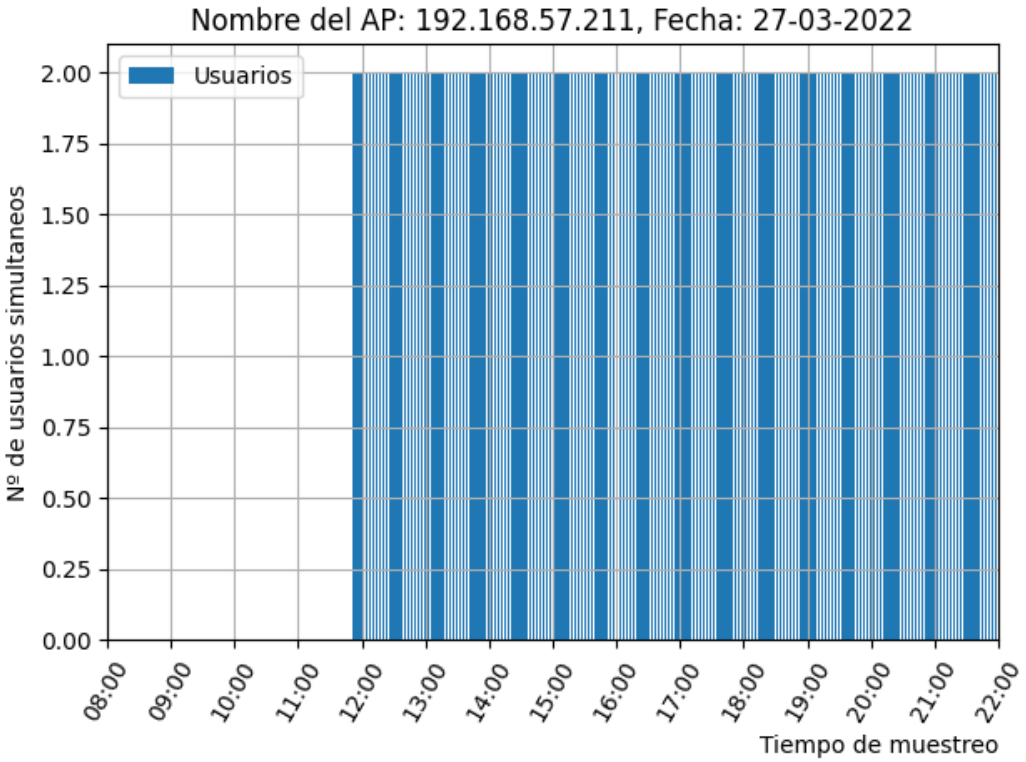



*Figura 20. Grafica de cantidad de usuarios por AP*

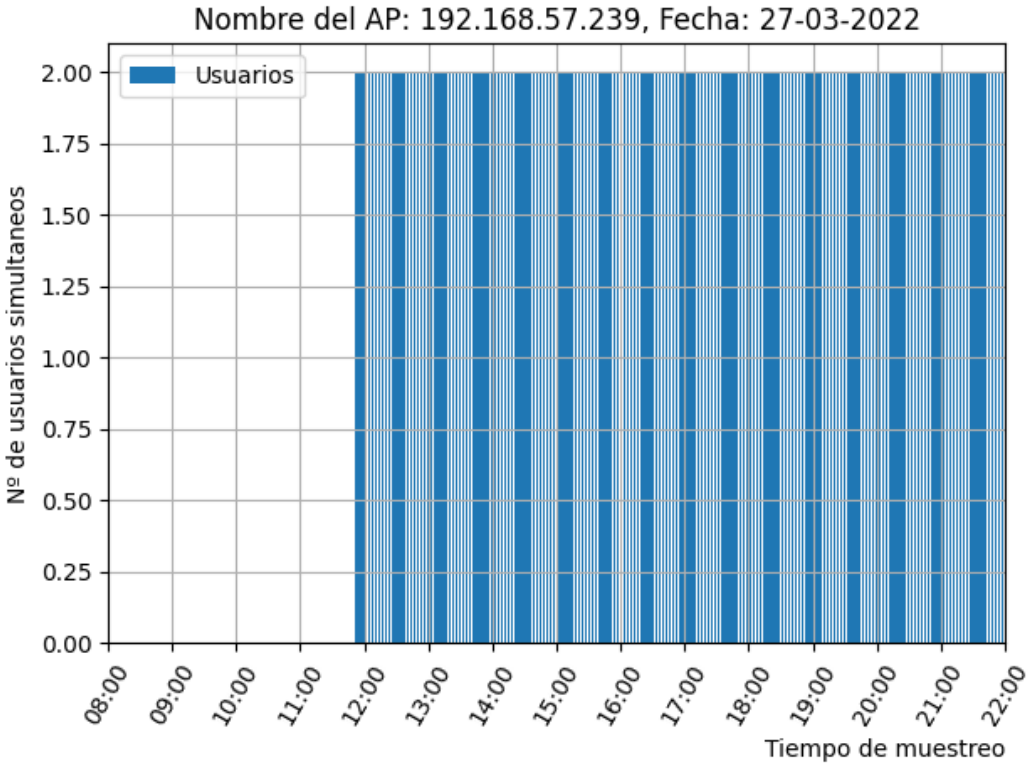



*Figura 21. Grafica de cantidad de usuarios por AP*

En primer lugar, mencionar que la falta de muestras de 8:00 a 12 :00 corresponde con una caída del servidor que recolecta los datos.

Por otro lado, hemos representado los dos mismos puntos de acceso que dan cobertura a las aulas antes mencionadas. Como se puede comprobar a simple vista vemos unas graficas más homogéneas en cuanto a ocupación del punto de acceso teniendo ambas 2 ‘usuarios’ de manera continuada durante todo el día. Este comportamiento también lo podemos observar en las gráficas del día 31 de marzo en los extremos de las mismas, donde vemos que se mantiene de manera constante 2 usuarios.

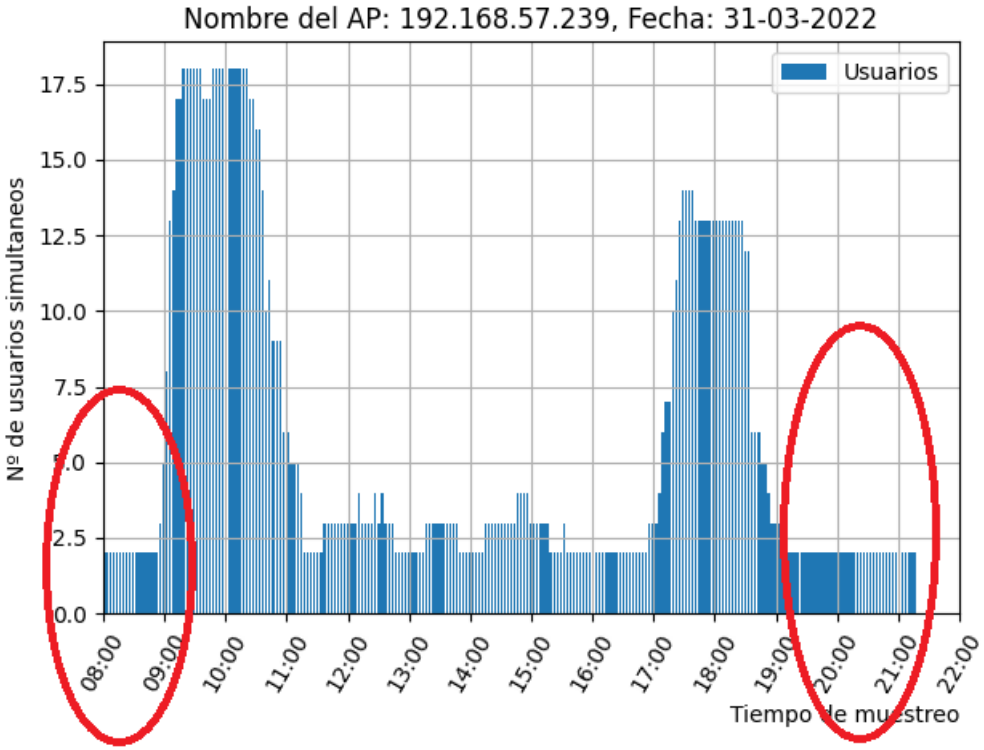



*Figura 22. Grafica de extracción del nº de equipos fijos en el entorno de estudio*

Este comportamiento se debe a que en estas aulas hay dispositivos conectados de manera continuada para la recolección de otro tipo de datos, por lo tanto, para nuestro estudio también es importante tener esta información del número de ‘dispositivos’ que no usuarios se encuentran en cada punto de acceso. Gracias a esta información de los días no lectivos donde las gráficas son muy estables con pocas o ninguna fluctuación de usuarios para por tener un valor más real de la cantidad de usuarios localizados en un aula. Simplemente deberíamos de restar al número de usuarios que nos encontremos en un punto de acceso durante un día lectivo la cantidad de usuarios constante que más se repita durante el fin de semana. En nuestra muestra observamos claramente 2 dispositivos durante los fines de semana por lo tanto un valor más ajustado de usuarios en esas aulas seria para cada uno de los puntos de acceso:

- 192.168.57.211

  11:00-13:00 à Ocupación = 16-2 =14 usuarios
  15:00-17:00 à Ocupación = 22-2 =20 usuarios

- 192.168.57.239

  9:00-11:00 à Ocupación = 17-2 =15 usuarios
  17:00-19:00 à Ocupación = 13-2 =11 usuarios

Como vemos en el ejemplo anterior, todo el tiempo nos referimos a un área bastante reducida como es el rango de cobertura de un AP, pero lo que realmente nos interesa es poder calcular el número de usuarios que se encuentran congregados en el mismo instante temporal dentro de un aula para controlar asistencia, evitar aglomeraciones y respetar la ocupación… Para ello otro factor que nos podría ser útil a la hora de identificar si se encuentran dentro de esa área de cobertura en un aula u otra es la frecuencia a la que se conectan 2.4 o 5GHz.

Como sabemos cada una de estas frecuencias tiene unos pros y unos contras que podríamos resumir de la siguiente manera:

| **DIFERENCIAS** | **2.4GHz** | **5GHz** |
|---|---|---|
| Canales | 14 canales no superpuestos | 25 canales no superpuestos |
| Interferencias | Más interferencias | Menos interferencias |
| Velocidad máxima | Menos velocidad de conexión | Más velocidad de conexión |
| Rango de red | Mayor rango | Menor rango |

*Tabla 9. Diferencia entre 2.4 y 5 GHz*

Por lo tanto, sabiendo que en la frecuencia de 5Ghz tenemos un menor rango, es decir, menor distancia al punto de acceso ya que la señal se atenúa mucho debido a las interferencias del entorno (muros de piedra, cristales, etc.) y asumiendo que la gran mayoría de usuarios/terminales tienen conectividad con ambas bandas de frecuencia, podríamos suponer que si observamos una mayor afluencia de usuarios en 5Ghz estarán situados en el aula donde el punto de acceso está ubicado físicamente y si por el contrario la mayoría de usuarios están en 2.4 GHz asumiremos que están situados en la otra aula donde reciben cobertura.

Para poder representar estas graficas en primer lugar tuvimos que identificar si un usuario estaba conectado en 2.4 o 5 GHz ya que esto no es un valor que nos indique de manera automática la controladora cuando le hacemos consultas. Para esto

utilizamos el dato que si nos indica de cada usuario que es la dirección MAC de la BSSID a la que se ha conectado el usuario, la cual es una dirección MAC virtual que asigna el punto de acceso a cada BSSID para diferenciar entre BSSID tanto para 2.4 como para 5Gz y la hemos ido comparando con cada una de las vMAC que publica cada punto de acceso que hemos utilizado para el estudio. Con estos hemos conseguido identificar si un usuario estaba conectado en 2.4 GHz o en 5Ghz.

Para seguir trabajando con el mismo entorno anterior hemos vuelto a coger las gráficas correspondientes al día 31 de marzo en los mismos puntos de acceso 192.168.57.211 y 192.168.57.239 con los siguientes resultados:

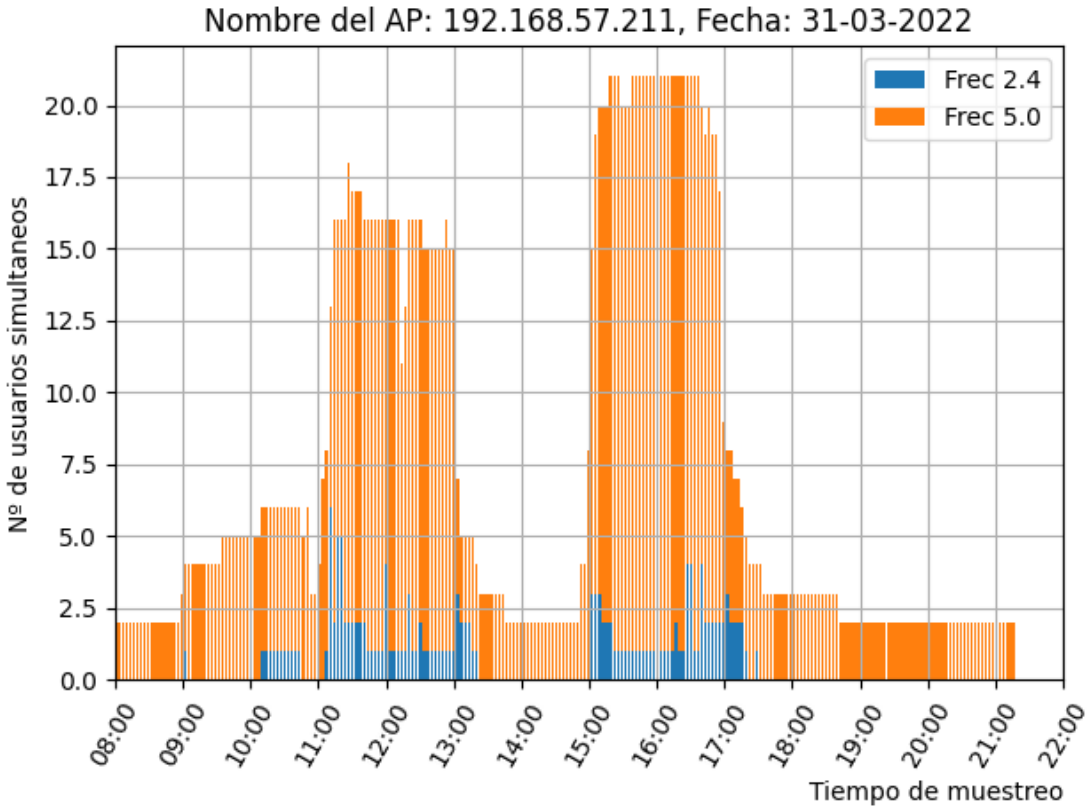



*Figura 23. Grafica de cantidad de usuarios por AP y Frecuencia*

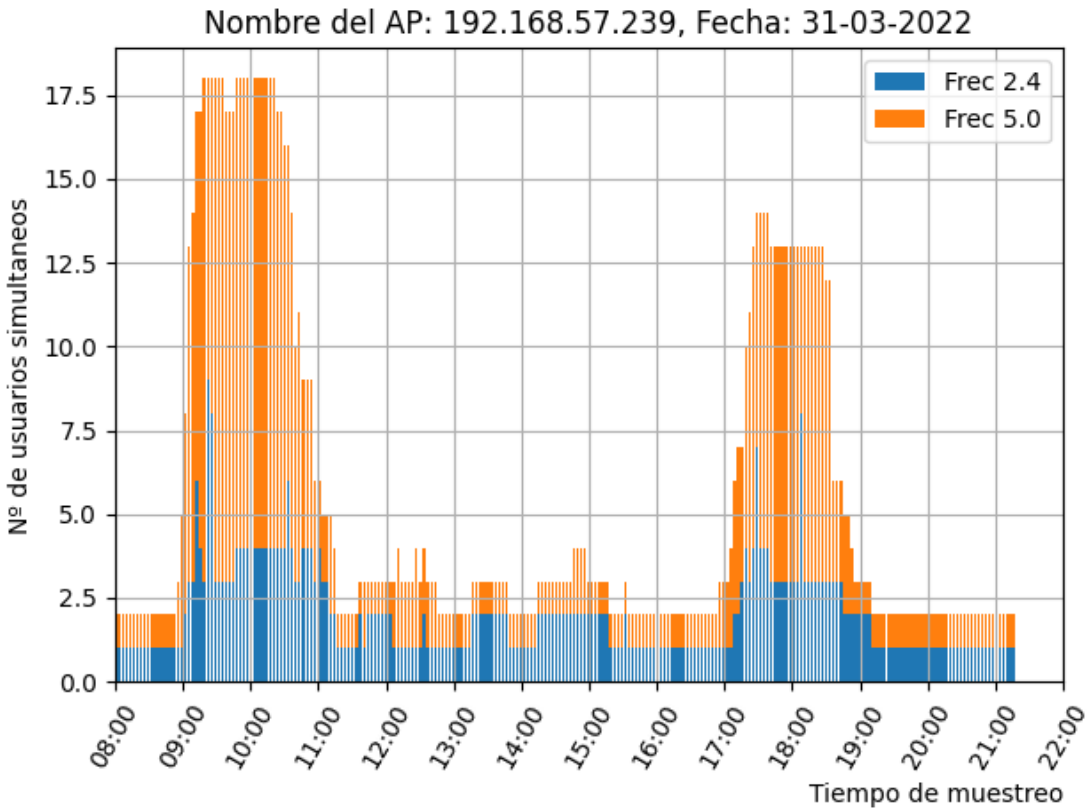



*Figura 24. Grafica de cantidad de usuarios por AP y Frecuencia*

Como podemos observar existe una mayor concentración de terminales conectados a la frecuencia de 5Ghz lo cual según nuestra hipótesis del rango que podemos alcanzar con esta frecuencia, podemos suponer que la concentración de usuarios se ha realizado en el aula donde se encuentra físicamente el punto de acceso que como podemos observar en el plano de infraestructura es la siguiente para cada zona:

- 192.168.57.211
  El aula PS-8 para el ala derecha del sótano

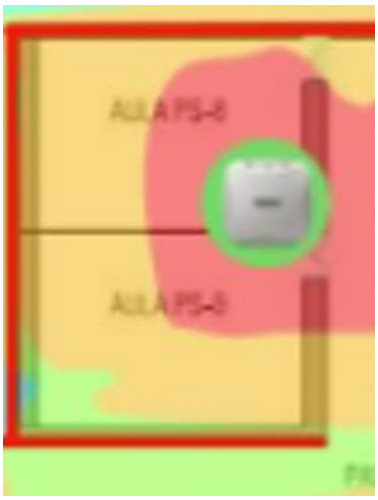


*Figura 25. Ampliación del Mapa de cobertura*

- 192.168.57.239
  El aula PS-4 para el ala izquierda del sótano

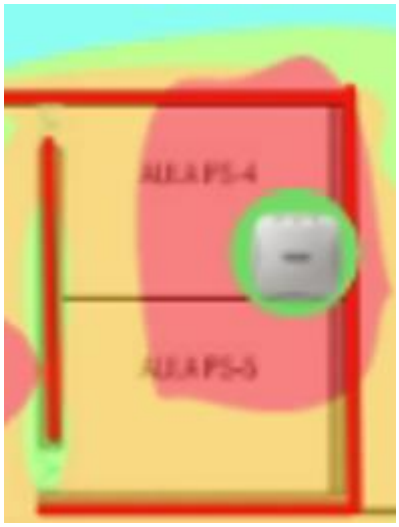


*Figura 26. Ampliación del mapa de cobertura del ala derecho del sótano*

Este estudio y conclusión de los datos obtenidos en este entorno para que fuese más fiable deberían de tomarse lo que se conoce como datos etiquetados, donde se capturan datos de los usuarios sabiendo realmente que terminales utilizan el aula en el que están y la frecuencia a la que se conectan para poder comprobar si realmente nuestra hipótesis es real.

Para cumplimentar estos datos también vamos a mostrar otro punto de acceso donde por el contrario existe una mayor concentración de equipos a la frecuencia de 2.4 GHz lo que supondría el otro caso de nuestra hipótesis, que los usuarios se encontrasen en el aula donde no se encuentra físicamente el punto de acceso. Como por ejemplo para el punto de acceso con que da servicio a las aulas PS-6 y PS-7

*Figura 27. Ampliación del mapa de cobertura de un AP*

Como vemos en la imagen este punto de acceso se encuentra ubicado claramente lejos del aula PS-6 y la gráfica que obtenemos en este caso es la siguiente:

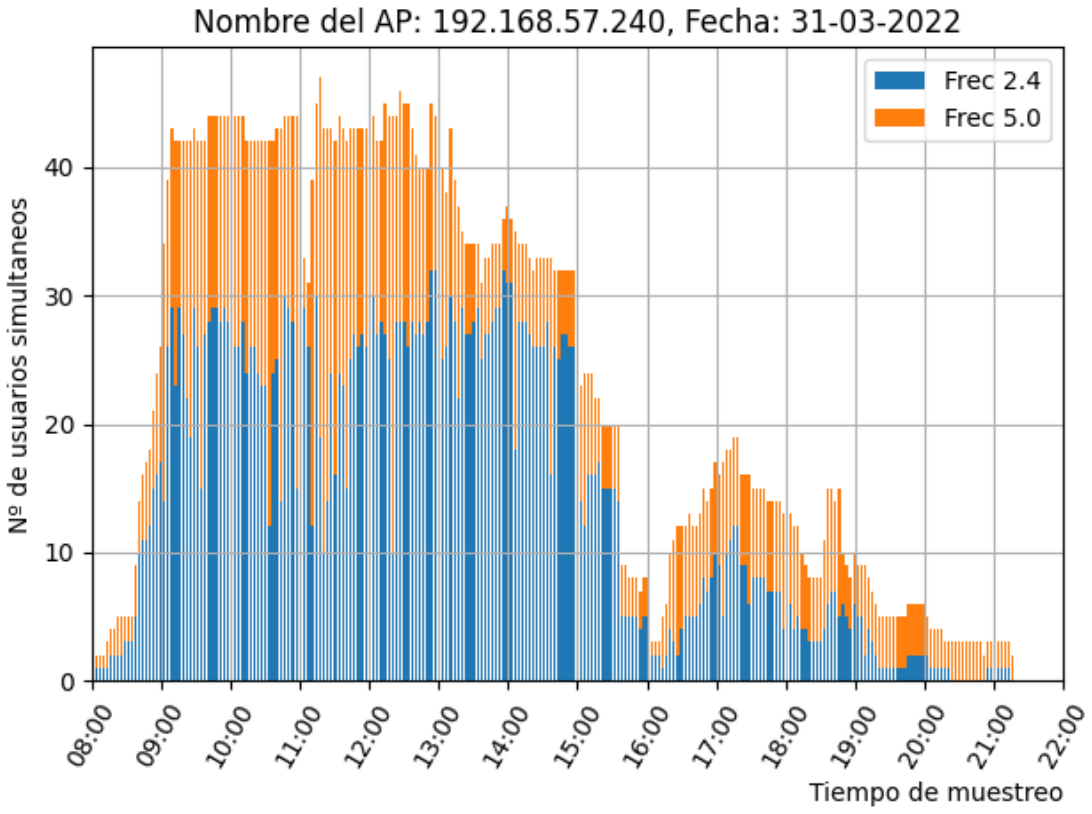



*Figura 28. Grafica de cantidad de usuarios por AP y Frecuencia*

En este caso vemos una mayor concentración de terminales funcionando en 2.4 GHz, aunque también vemos usuarios conectados en 5Ghz, la proporción sería de unos ¾ para 2.4 GHz y ¼ para 5Ghz. Con estos datos y nuestra hipótesis tendríamos dos casos bastante probables:

En primer lugar, podríamos tener usuarios que se encuentran en el aula PS-6 la cual está más alejada del punto de acceso por que tendría sentido mayor número de terminales en 2.4 GHz y lo equipos que están conectados en 5Ghz podrían corresponder con terminales de última generación con mejor tecnología de antenas.

La otra situación que también es posible es que en este caso las dos clases tuviesen usuarios dentro, donde los usuarios en 2.4 serían los que se encuentran en su mayoría en la PS-6 y los usuarios de 5Ghz se encuentran en la PS-7. Como hemos dicho anteriormente para poder comprobar estos resultados deberíamos tener datos etiquetados con estas características y ver el comportamiento de la red.

## 4.2 Segundo escenario: registro de datos y estudio de las aulas de la Biblioteca del Edificio de Antigones.

Para este segundo escenario hemos elegido la biblioteca situada en el edificio de Antigones debido a que es otro punto de gran afluencia de personas, pero esta vez en un área abierta sin muros que limiten las áreas de cobertura de los puntos de acceso.

La distribución de estos puntos de acceso es la siguiente:

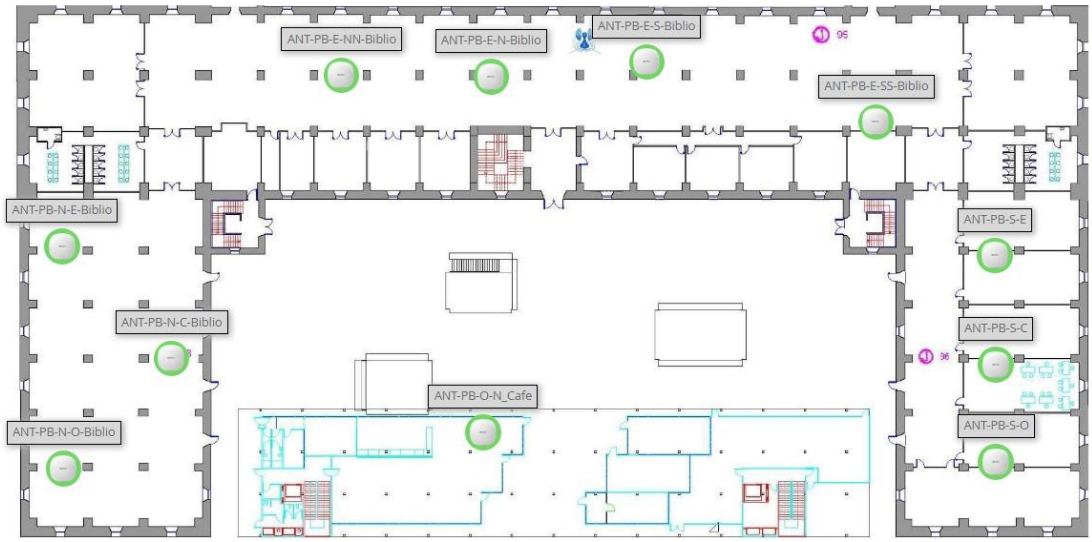



*Figura 29. Mapa del segundo entorno de estudio Biblioteca de Antigones*

Y su mapa de calor cubre el área completa de la siguiente forma:

El direccionamiento para estos puntos de acceso es el siguiente:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ANT-PB-S-O | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 0 | - | 0 bps | 192.168.89.183 | Aruba AP 505 |
| ANT-PB-S-E | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 1 | - | 0 bps | 192.168.89.188 | Aruba AP 505 |
| ANT-PB-S-C | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 0 | - | 0 bps | 192.168.89.186 | Aruba AP 505 |
| ANT-PB-O-N_Cafe | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 9 | - | 7,21 Kbps | 192.168.89.117 | Aruba AP 505 |
| ANT-PB-N-O-Biblio | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 16 | - | 315 Kbps | 192.168.89.189 | Aruba AP 515 |
| ANT-PB-N-E-Biblio | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 18 | - | 1,32 Mbps | 192.168.89.111 | Aruba AP 515 |
| ANT-PB-N-C-Biblio | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 16 | - | 27,3 Mbps | 192.168.89.187 | Aruba AP 515 |
| ANT-PB-E-SS-Biblio | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 4 | - | 1,12 Mbps | 192.168.89.249 | Aruba AP 515 |
| ANT-PB-E-S-Biblio | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 14 | - | 1,7 Mbps | 192.168.89.182 | Aruba AP 515 |
| ANT-PB-E-NN-Biblio | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 2 | - | 0 bps | 192.168.89.184 | Aruba AP 515 |
| ANT-PB-E-N-Biblio | Up | Good | VC_MURALLA | Antigones | Cluster_Muralla | 10 | - | 2,38 Mbps | 192.168.89.181 | Aruba AP 515 |

*Tabla 10. Direccionamiento de los Puntos de Acceso del la biblioteca*

Y los días muestreados son los siguientes

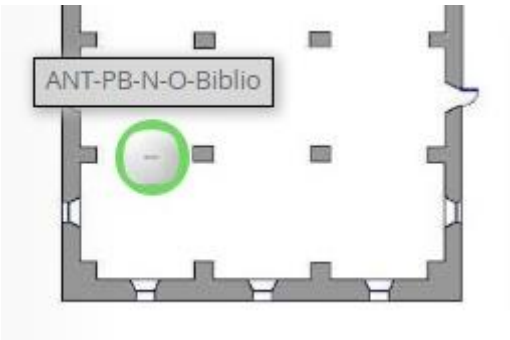

*Figura 30. Dias de muestreo en la biblioteca*

Con estas muestras vamos a observar cómo varía el número de usuarios a lo largo de varios días, primero en días lectivos y en segundo lugar fines de semana:

Días lectivos:

El día que hemos tomado para representar en las gráficas es el 30 de mayo, en este primer caso no diferenciaremos por frecuencia a la que se conecta el usuario, simplemente por la cantidad de usuarios que vemos conectados.

La primera zona que vamos a analizar es la cubierta por el punto de acceso más a la izquierda del plano 192.168.89.189




Con esto vamos a evitar la interferencia con otros puntos de acceso en este primer análisis.

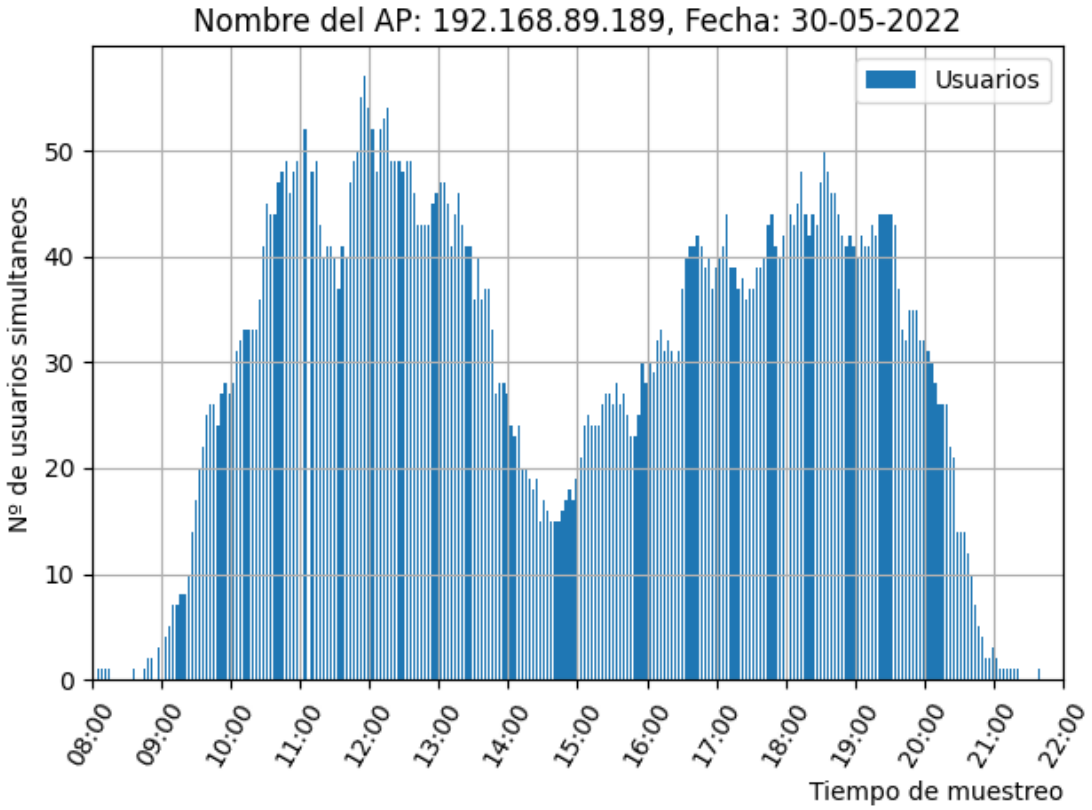



*Figura 31. Grafica de cantidad de usuarios por AP*

Como podemos ver en primer lugar estas graficas son bastante diferentes a las que se han conseguido en las aulas de sótano del Hospital de Marina, debido a que la asistencia a la biblioteca no tiene un horario definido en calendario si no que está abierta de manera ininterrumpida de 9:00 a 21:00 horas. La segunda observación y por la cual comprobamos que es una representación de la realidad es que el cupo de usuarios aumenta de 9:00 a 11:00 luego hay una pequeña caída debido al descanso de media mañana alrededor de 30 min y ya desciende a la hora de comer sobre las 14:00 y por la tarde vuelve a haber otro máximo de ocupación. Este patrón es algo que se repite de manera similar en el resto de puntos de accesos de esa ala izquierda de la biblioteca como vemos en las siguientes graficas:

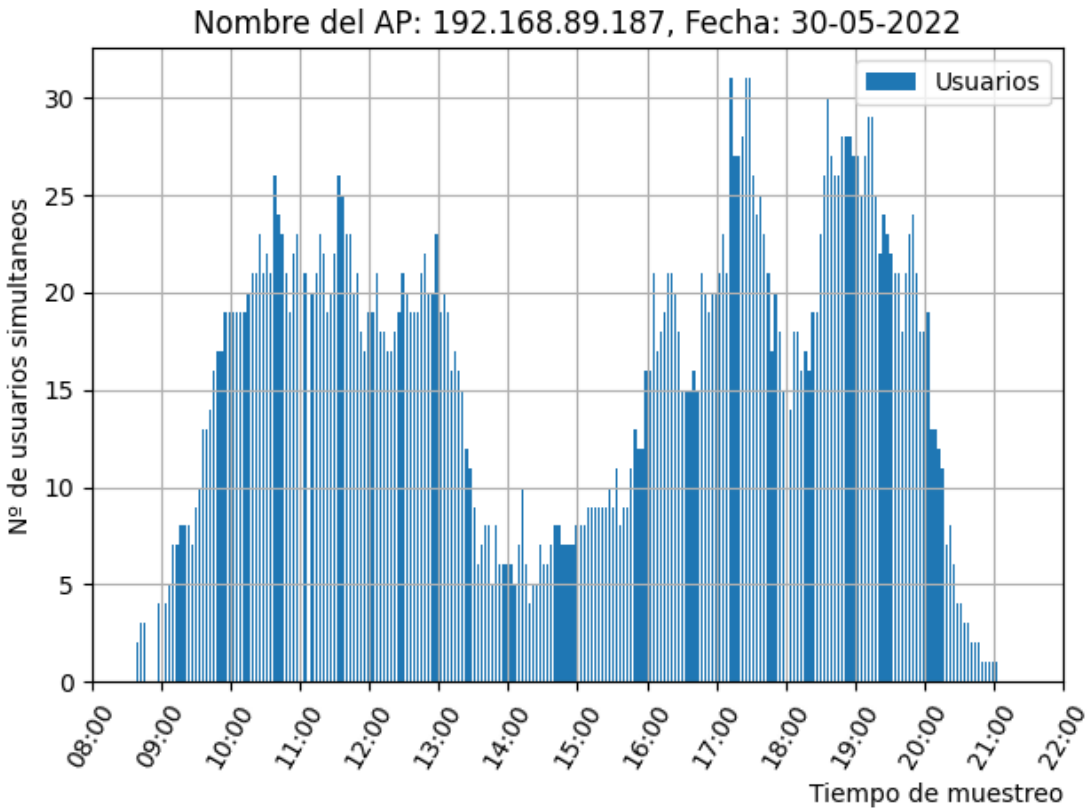



*Figura 32. Grafica de cantidad de usuarios por AP*

Esta grafica corresponde con el punto de acceso contiguo al anterior por lo que cubren una misma área.

Fines de semana:

En este caso los fines de semana al contrario que en el edificio de industriales que está cerrado y encontrábamos un comportamiento muy estable, en la biblioteca vemos que si hay usuarios y la ocupación fluctúa, pero principalmente en el ala derecha donde se encuentran las aulas de estudio.

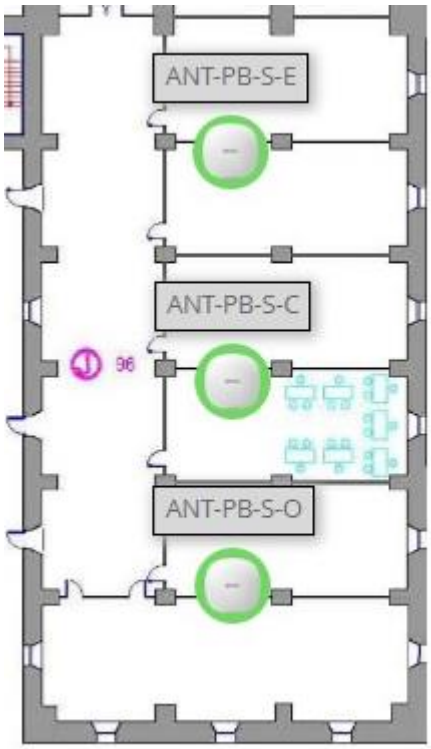



*Figura 33. Ala derecha de la biblioteca de Antigones*

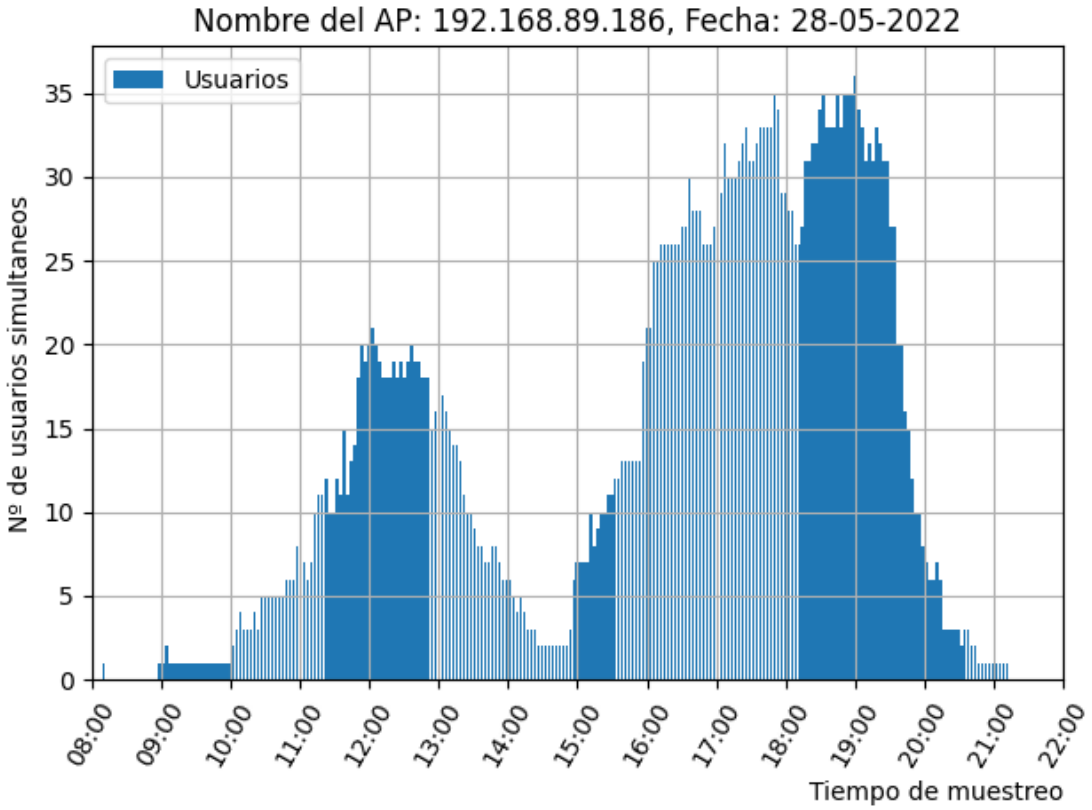



*Figura 34. Grafica de cantidad de usuarios por AP*

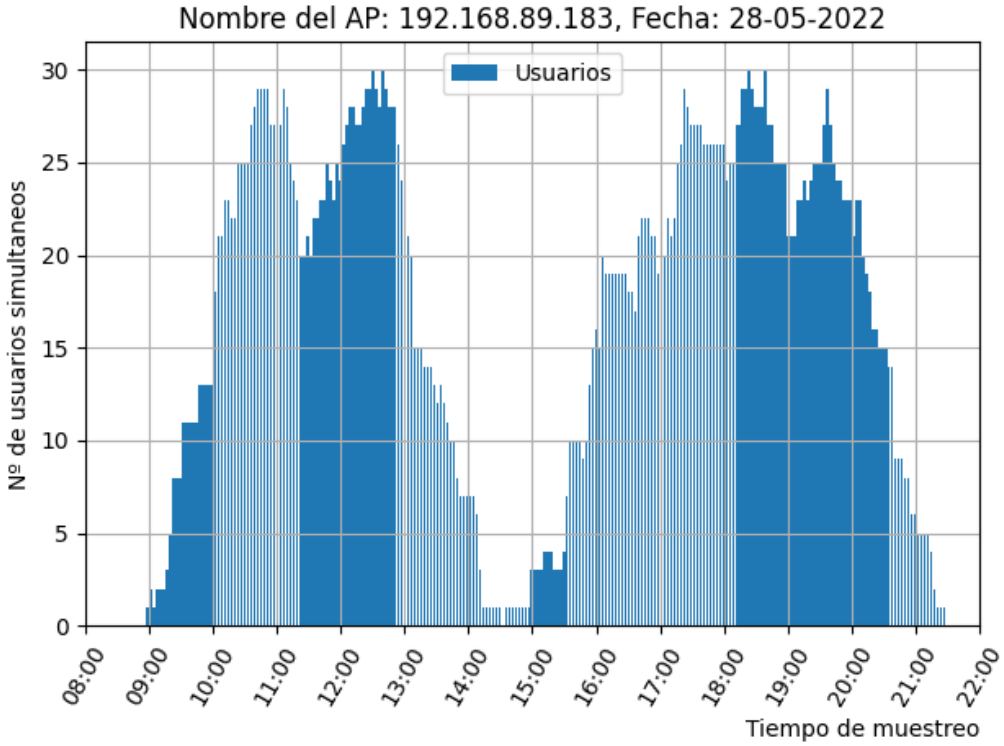



*Figura 35. Grafica de cantidad de usuarios por AP*

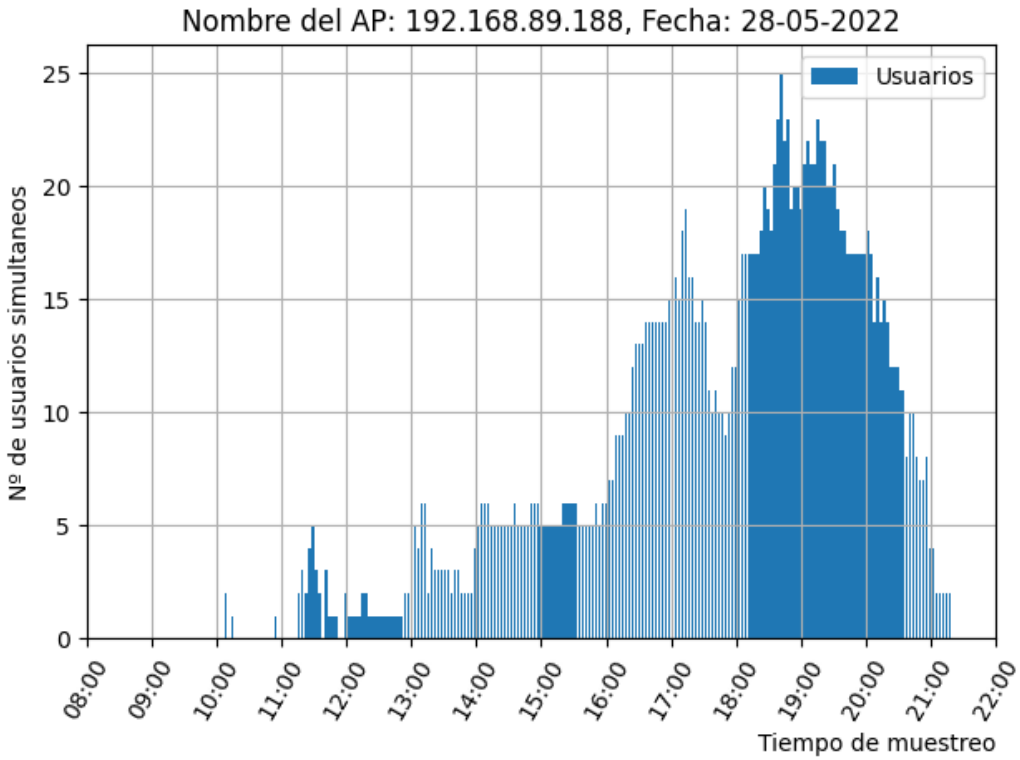



*Figura 36. Grafica de cantidad de usuarios por AP*

Y como se puede observar en la última grafica correspondiente al punto de acceso 192.168.89.188 tiene un aumento de tráfico solamente por la tarde debido a que hasta que no estén ocupadas las primeras aulas no abren las siguientes.

En este escenario también vamos a hacer un estudio diferenciando por tipo de frecuencia que se utiliza para conectarse a la red wifi.

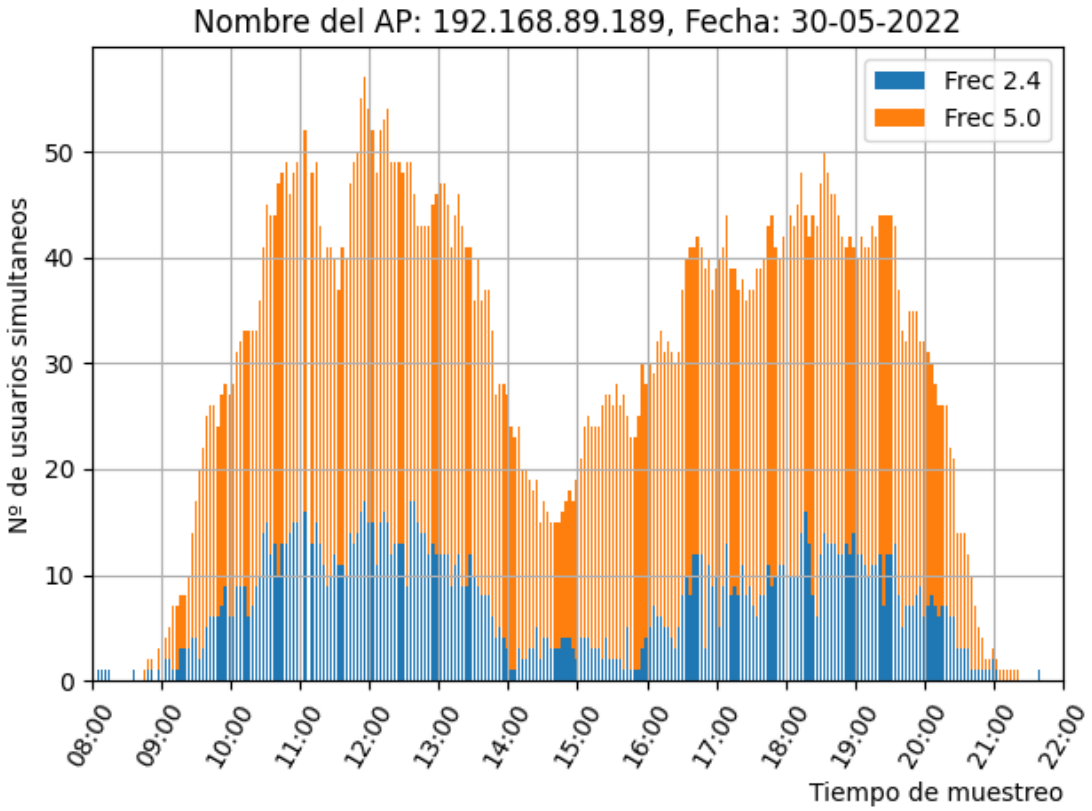



*Figura 37. Grafica de cantidad de usuarios por AP y Frecuencia*

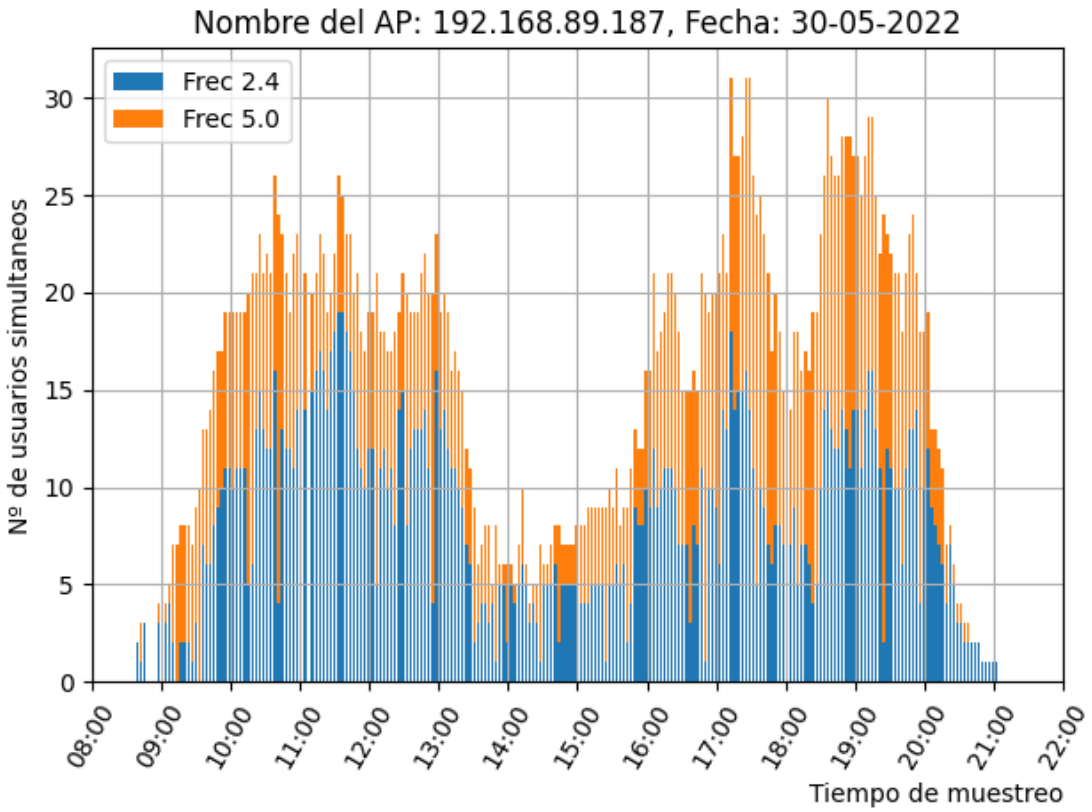



*Figura 38. Grafica de cantidad de usuarios por AP y Frecuencia*

En este caso al ser un área abierta sin muros que delimiten ninguna zona es muy complicado sacar conclusiones a partir de la frecuencia de conexión al punto de acceso. Como se ve en las gráficas los usuarios se conectan en 2.4 o 5Ghz dependiendo de la distancia al punto de acceso sin observarse ningún patrón reseñable para el resto de los días.

En este caso donde tenemos un área grande como es la biblioteca donde no hay zonas delimitas deberemos tomarlo como un conjunto y para comprobar que no se supera el aforo de la biblioteca y para ello necesitaremos datos etiquetados, pero eso lo analizaremos en el siguiente punto.

## 4.3 Análisis de los usuarios conectados por AP de la biblioteca de antigones con los datos de ocupación ofrecidos por el personal de la biblioteca.

En este punto vamos a utilizar los datos de conteo recogidos por el personal de la biblioteca durante un periodo de 4 días para comprobar si los datos obtenidos mediante la controladora nos serian útiles para estimar la ocupación de la biblioteca.

Los datos ofrecidos son los siguientes:

| Tiempo de muestreo | 30 de mayo | 31 de mayo | 1 de junio | 2 de junio |
|---|---|---|---|---|
| 12:00 | X | X | 185 | 169 |
| 18:00 | 319 | 332 | 317 | 270 |

*Tabla 11. Cantidad de usuarios contados a mano por los responsables de la biblioteca*

En este caso solo se utiliza la siguiente área de la biblioteca, las aulas del ala derecha no entran en el conteo.

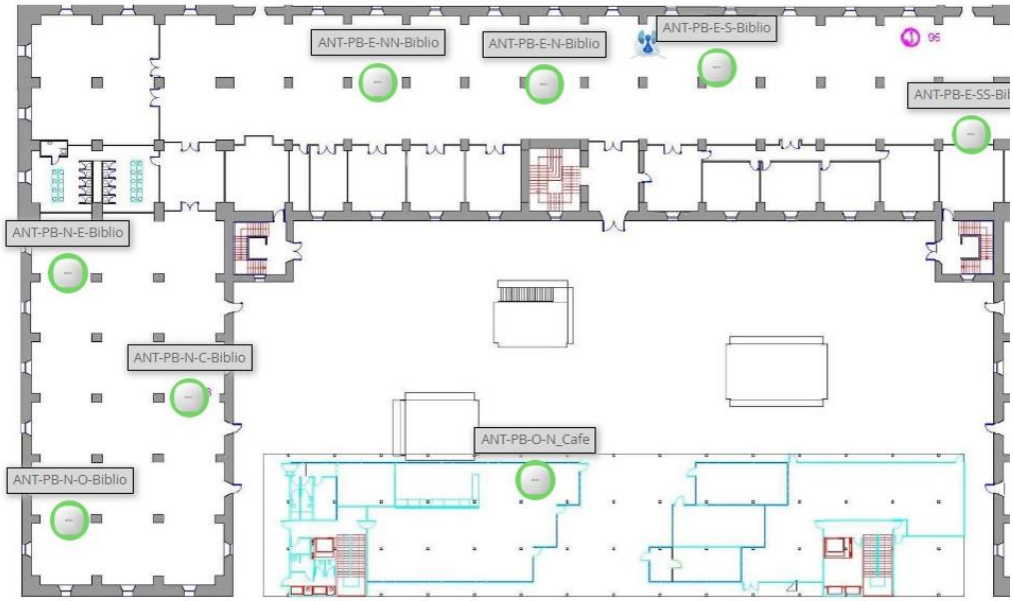



*Figura 39. Mapa del ala izquierda de la biblioteca.*

Las gráficas de esta zona para esos días son las siguientes:

**30 de mayo:**

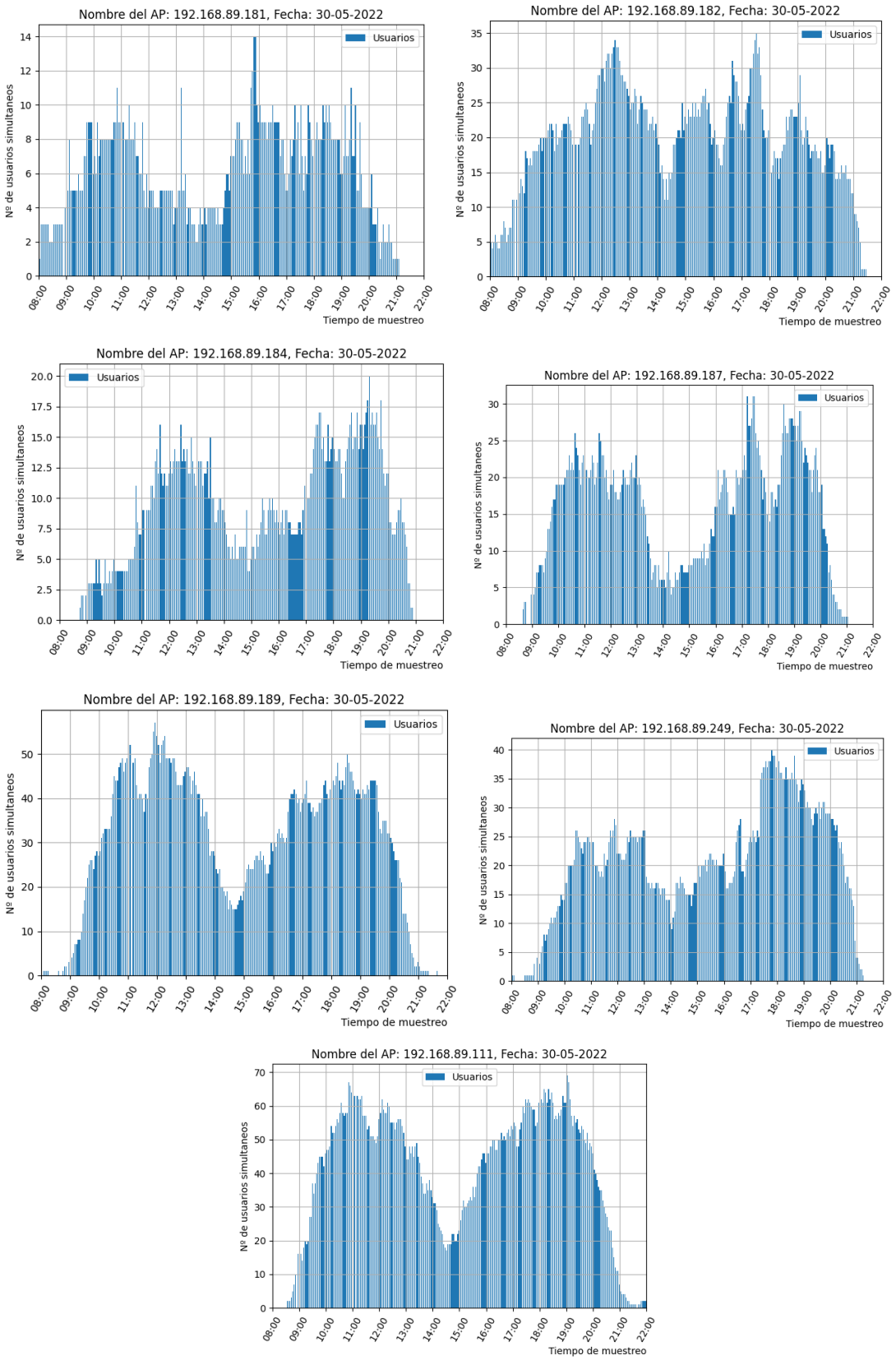



*Figura 40. Graficas de cantidad de usuarios para los AP correspondientes al ala izquierda de la biblioteca 30-5*

**31 de mayo:**

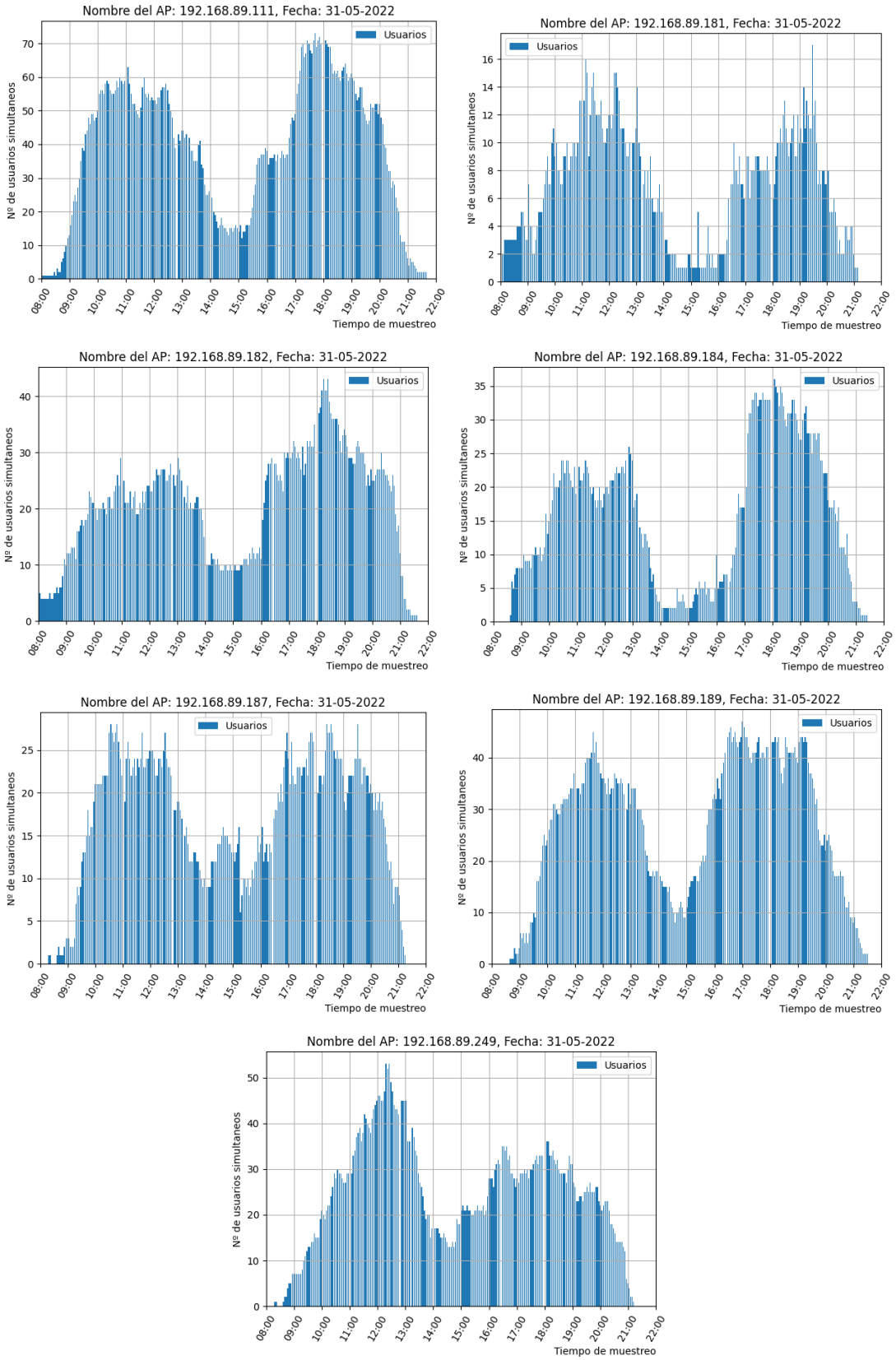



*Figura 41. Graficas de cantidad de usuarios para los AP correspondientes al ala izquierda de la biblioteca 31-5*

**1 de junio:**

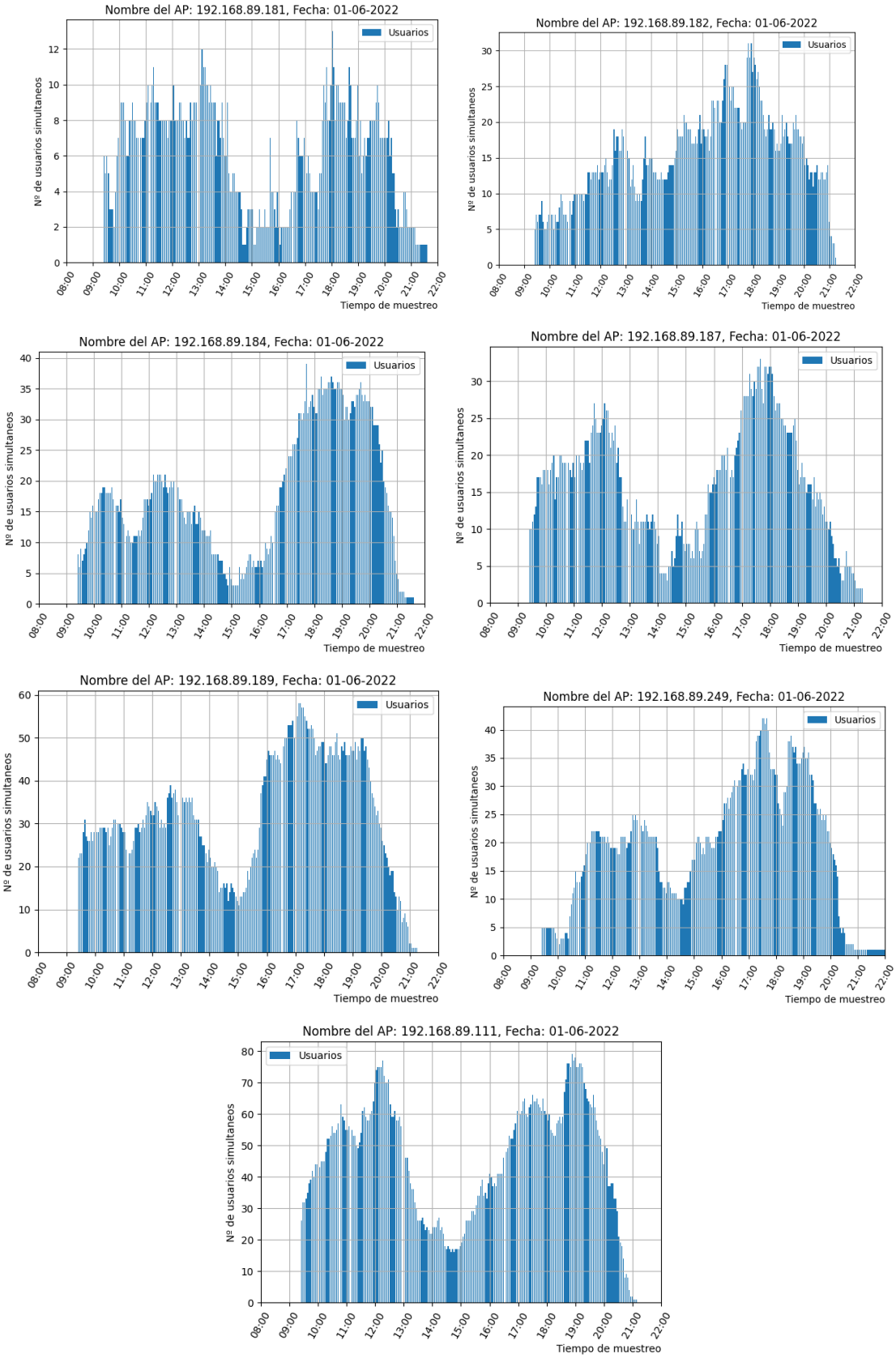



*Figura 42. Graficas de cantidad de usuarios para los AP correspondientes al ala izquierda de la biblioteca 1-6*

**2 de junio:**

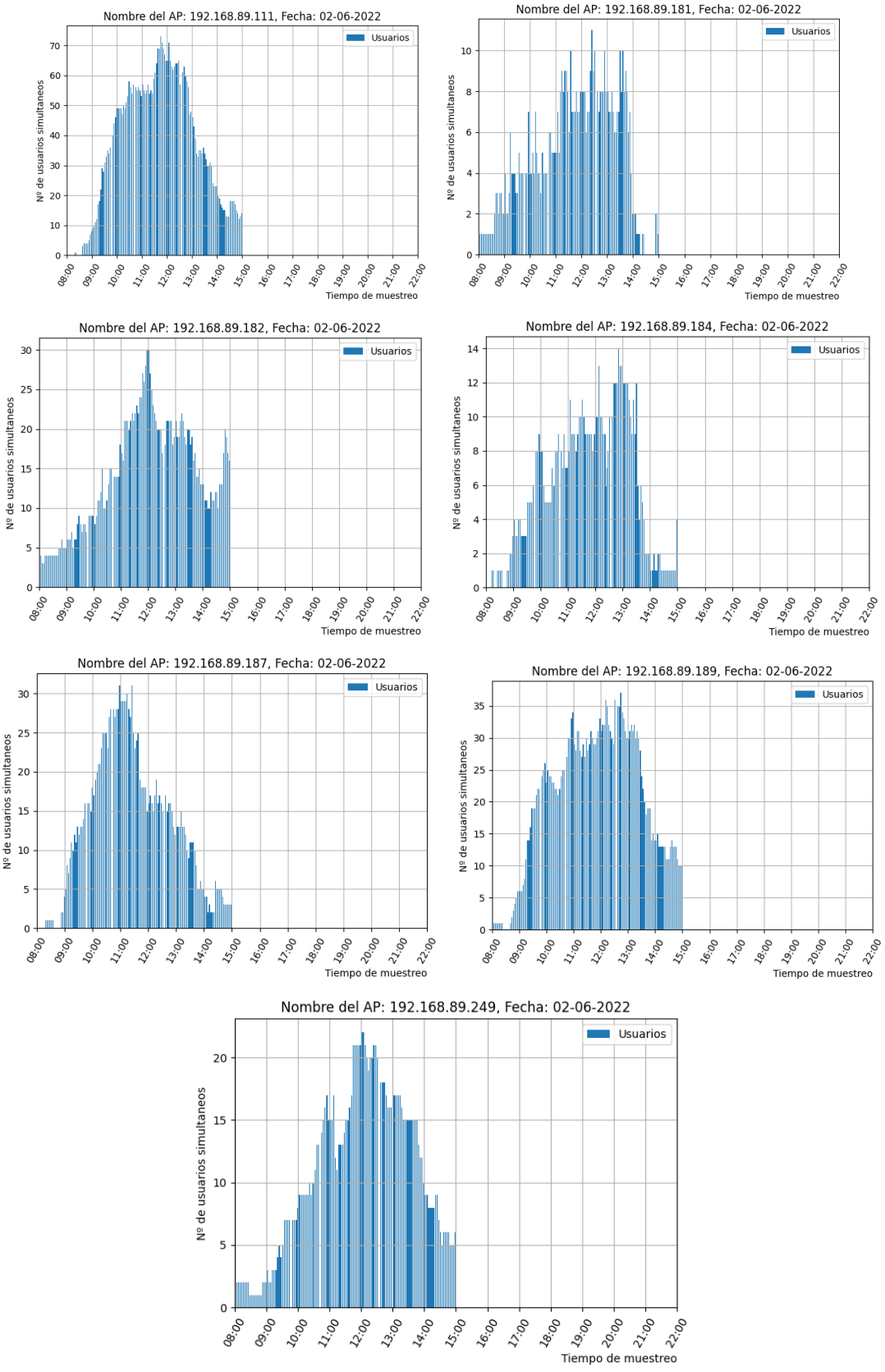



*Figura 43. Graficas de cantidad de usuarios para los AP correspondientes al ala izquierda de la biblioteca 2-6*

Con estas graficas la tabla que obtenemos con la cantidad de usuarios en los instantes donde se tomaron las muestras es la siguiente:

| Tiempo de muestreo | 30 de mayo | 31 de mayo | 1 de junio | 2 de junio |
|---|---|---|---|---|
| 12:00 | 204 | 206 | 190 | 183 |
| 18:00 | 221 | 260 | 249 | X |

*Tabla 12. Cantidad de usuarios medios medidos por los puntos de acceso*

Comparando con la tabla de medidas reales observamos gran diferencia

| Tiempo de muestreo | 30 de mayo | 31 de mayo | 1 de junio | 2 de junio |
|---|---|---|---|---|
| 12:00 | X | X | 185 | 169 |
| 18:00 | 319 | 332 | 317 | 270 |

*Tabla 13. Cantidad de usuarios medios medidos por los bibliotecarios*

Esta diferencia podría deberse a diferentes factores como, por ejemplo:

I. Método de muestreo real.
II. Usuarios que no utilizan la red WIFI
III. Usuarios que estaban en el patio/cantina pero contaron su asiento.

# Capítulo 5. Conclusiones y trabajos futuros

## 5.1 Conclusiones

En este proyecto hemos utilizado una arquitectura de red wifi real, esto nos ha permitido conocer cómo funcionan las tecnologías reales que utilizan los fabricantes de estos sistemas como Aruba en nuestro caso.

Otro punto importante que hemos aprendido es el protocolo SNMP para la gestión de redes, sus diferentes versiones y formas de utilizarlo para hacer consultas SNMP a equipos remotos para obtener su información y poder estudiarla.

También hemos aprendido sobre el lenguaje de programación Python y tanto para realizar las consultas SNMP, trabajar con dataframe y analizar grandes cantidades de datos.

En esta primera aproximación al sistema final hemos conseguido automatizar la recolección de datos mediante SNMP, almacenamiento de datos en ficheros csv, tratamiento de los datos obtenidos, generación de graficas para su análisis exploratorio y primeras conclusiones de la estimación de usuarios.

## 5.2 Grado de consecución de los objetivos y formación adquirida

**Resumen de los objetivos propuestos y alcanzados**

De los objetivos propuestos para este proyecto hemos logrado los siguientes:

1. Conocimiento de una red inalámbrica real, punto de acceso, protocolos de gestión, controladora, datos que caracterizan a los dispositivos…
2. Diseño de un programa para la consecución de ficheros de los usuarios conectados a la red mediante consultas SNMP a la controladora.
3. Tratamiento de grandes cantidades de datos con el lenguaje de programación en Python.
4. Estimación de usuarios sin datos etiquetados para su comprobación.
5. Aprendizaje de los diferentes estándares wifi y sus características principales.

**Curva de aprendizaje y principales dificultades encontradas**

El punto de mayor dificultad ha sido aprender a utilizar el lenguaje de programación Python puesto que era un lenguaje que no había utilizado y he tenido que aprender a utilizar librerías y crear funciones …

También ha sido importante aprender el uso de SNMP con sus OID específicas del fabricante, como generar funciones que hagan consultas y como utilizar las comunidades SNMP.

**Reflexión sobre la formación adquirida en conocimientos, metodología y competencias**

Este proyecto me ha servido para conocer cómo funcionan las redes de wifi de otros fabricantes como se gestionan y las características que ofrecen a nivel de seguridad, despliegue, información de usuarios centralizada…

Además de aprender cómo funciona básicamente el lenguaje de programación Python con sus librerías específicas para SNMP como pysnmp.

## 5.3 Trabajos futuros

Los trabajos futuros relacionados con este proyecto serian:

- Toma de datos etiquetados en los diferentes entornos de estudio como son las aulas del sótano de industriales y la biblioteca.
- Estudio del número de personas que utilizan la red eduroam entre todas las personas que van a la universidad
- Creación de un programa que mediante el aprendizaje con los datos etiquetados estime el número de usuarios que hay en un aula utilizando algoritmos de machine learning.

# Bibliografía y referencias

[1] Web UPCT: www.upct.es

[2] CWNA Certified Wireless Network Administrator de David D. Coleman (Autor), David A. Westcott (Autor)

[3] https://es.ccm.net/contents/789-introduccion-a-wifi-802-11-o-wifi

[4]https://community.cisco.com/t5/blogs-wireless-mobility/est%C3%A1ndares-inal%C3%A1mbricos/ba-p/3365301

[5] http://www.ingenieriasystems.com/2016/12/Trama-inalambrica-80211-y-Resumen-CCNA1-V5-CISCO-C4.html#:~:text=Las%20redes%20802.11%20tambi%C3%A9n%20utilizan,intactos%2C%20se%20retransmite%20la%20trama.

[6] https://www.muycomputer.com/2021/03/21/estandares-wi-fi-mas-utilizados/#:~:text=IEEE%20802.11%3A%20el%20est%C3%A1ndar%20que,velocidad%20m%C3%A1xima%20de%2054%20Mbps.

[7]https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/802.11_Association_Process_Explained

[8] https://www.eduroam.es/index.es.php

[9] https://www.arubanetworks.com/assets/_es/ds/DS_ArubaOS8.pdf

[10]https://www.teldat.com/blog/es/controlador-wireless-lan-y-su-implantacion-en-la-nube/#:~:text=Son%20las%20tareas%20que%20permiten,diagnosticar%20problemas%20en%20la%20red.

[11] https://www.securewirelessworks.com/AirWave.asp

[12] https://www.arubanetworks.com/techdocs/Instant_40_Mobile/Advanced/Content/UG_files/virtual_controller/Master_Election_Protocol.htm

# Anexos

## Anexo 1. Estructura y organización del código

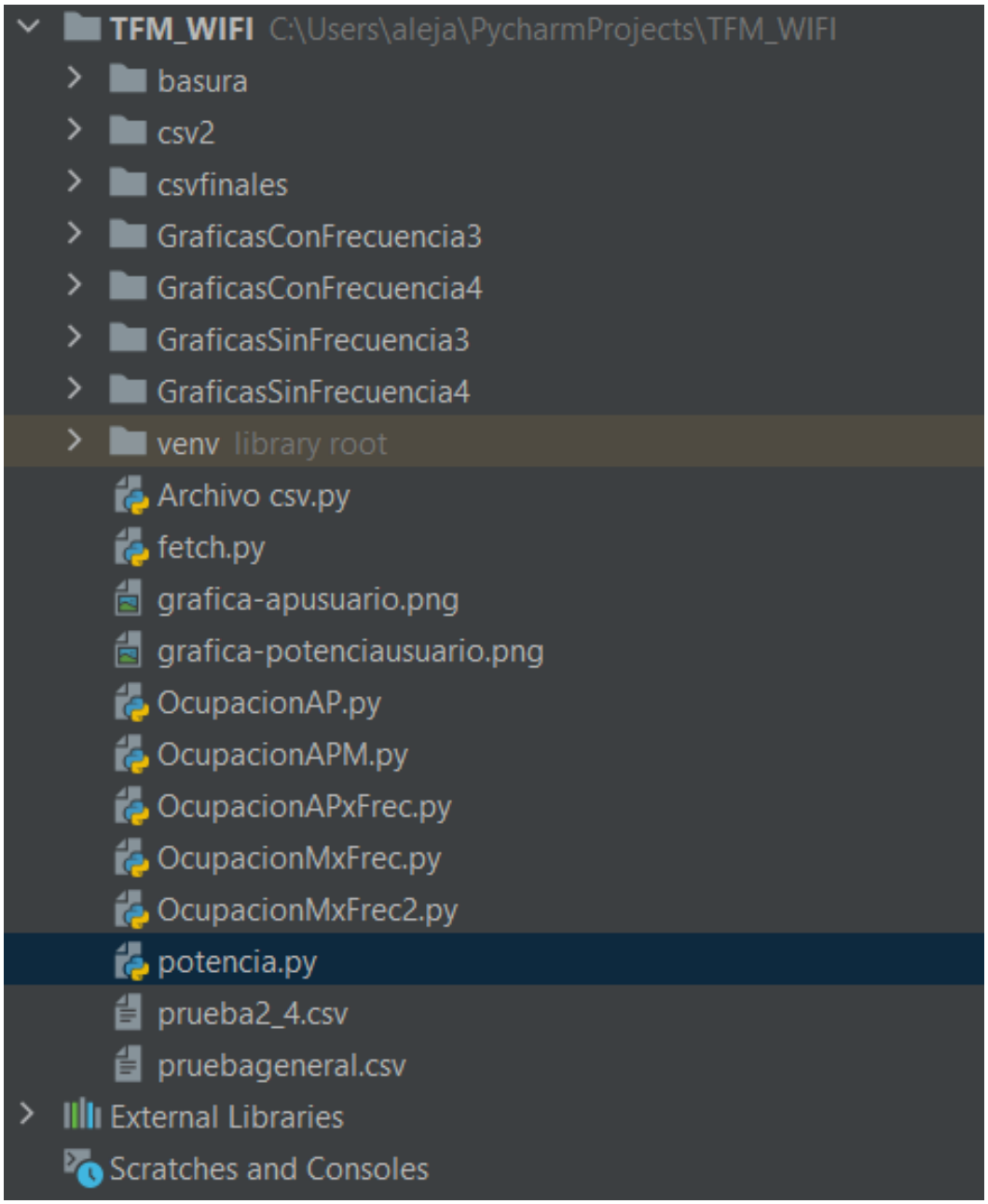



*Figura 44. Estructura código y programas desarrollados*

La estructura de carpetas es sencilla, por un lado, tenemos los ficheros Python para realizar las consultas a la controladora y generar los csv que se guardaran en la carpeta csv finales, después tenemos los demás archivos Python que se utilizan para analizar los datos generando graficas que se guardan respectivamente en las carpetas con el nombre GraficasConFrecuencia y GraficasSinFrecuencia.

# Anexo 2. Código fuente script Py consultas a la controladora Aruba y generación fichero CSV del día.

```
from pysnmp import hlapi
from pysnmp.entity.rfc3413.oneliner import cmdgen
import time
import pandas as pd


#### Funciones SNMP para manejar las consultas al servidor
def construct_object_types(list_of_oids):
    object_types = []
    for oid in list_of_oids:

object_types.append(hlapi.ObjectType(hlapi.ObjectIdentity(oid)))
    return object_types


def cast(value):
    try:
        return int(value)
    except (ValueError, TypeError):
        try:
            return float(value)
        except (ValueError, TypeError):
            try:
                return str(value)
            except (ValueError, TypeError):
                pass
    return value


def fetch(handler, count):
    result = []
    for i in range(count):
        try:
            error_indication, error_status, error_index, var_binds =
next(handler)
            if not error_indication and not error_status:
                items = {}
                for var_bind in var_binds:
                    items[str(var_bind[0])] = cast(var_bind[1])

                result.append(items)
            else:
                raise RuntimeError('Got SNMP error:
{0}'.format(error_indication))
        except StopIteration:
            break
    return result


def get(target, oids, credentials, port=161,
engine=hlapi.SnmpEngine(), context=hlapi.ContextData()):
    handler = hlapi.getCmd(
        engine,
        credentials,
        hlapi.UdpTransportTarget((target, port)),
        context,
```

```
        *construct_object_types(oids)
    )
    return fetch(handler, 1)[0]


def get_bulk(target, oids, credentials, count, start_from=0, port=161,
             engine=hlapi.SnmpEngine(), context=hlapi.ContextData()):
    handler = hlapi.bulkCmd(
        engine,
        credentials,
        hlapi.UdpTransportTarget((target, port)),
        context,
        start_from, count,
        *construct_object_types(oids)
    )
    return fetch(handler, count)


def get_bulk_auto(target, oids, credentials, count_oid, start_from=0,
port=161,
                  engine=hlapi.SnmpEngine(),
context=hlapi.ContextData()):
    count = get(target, [count_oid], credentials, port, engine,
context)[count_oid]

    return get_bulk(target, oids, credentials, count, start_from,
port, engine, context)


#########


#######Funcion para crear el csv apartir de las peticiones SNMP.


def crearcsv():
    hora = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
#Peticiones snmp de mac e ip con otro metodo
    ip="192.168.89.1"
    community="proyecto"


    generator = cmdgen.CommandGenerator()
    comm_data = cmdgen.CommunityData("server", community, 1) # 1 means
version SNMP v2c
    transport = cmdgen.UdpTransportTarget((ip, 161))

    real_fun = getattr(generator, "nextCmd")
#IP usuario
    oidipusr=(1,3,6,1,4,1,14823,2,3,3,1,2,4,1,3)
    IPusr = (errorIndication, errorStatus, errorIndex, varBinds) =
real_fun(comm_data, transport, oidipusr)
    if not errorIndication is None  or errorStatus is True:
        print ("Error: %s %s %s %s" % IPusr)
    else:
        c = int(1)
        n = len(varBinds)+1
        IPuser = [None] * n
        IPuser[0] = 0
```

```
        for var_bind in varBinds:
            IPuser[c] =[x.prettyPrint() for x in var_bind]
            c=c+1
    ##MACC
    oidmacuser=(1,3,6,1,4,1,14823,2,3,3,1,2,4,1,1)
    MAC = (errorIndication, errorStatus, errorIndex, varBinds) =
real_fun(comm_data, transport, oidmacuser)

    if not errorIndication is None  or errorStatus is True:
        print ("Error: %s %s %s %s" % MAC)
    else:
        c = int(1)
        n = len(varBinds)+1
        MACUSR = [None] * n
        MACUSR[0] = 0

        for var_bind in varBinds:
            MACUSR[c] =[x.prettyPrint() for x in var_bind]
            c=c+1
    ##MACWLAN
    oidmacWLAN=(1,3,6,1,4,1,14823,2,3,3,1,2,4,1,2)
    MACWLAN = (errorIndication, errorStatus, errorIndex, varBinds) =
real_fun(comm_data, transport, oidmacWLAN)

    if not errorIndication is None  or errorStatus is True:
        print ("Error: %s %s %s %s" % MACWLAN)
    else:
        c = int(1)
        n = len(varBinds)+1
        MACW = [None] * n
        MACW[0] = 0

        for var_bind in varBinds:
            MACW[c] =[x.prettyPrint() for x in var_bind]
            c=c+1

#IP Punto de acceso AP
    oidipap=(1,3,6,1,4,1,14823,2,3,3,1,2,4,1,4)
    IPap = (errorIndication, errorStatus, errorIndex, varBinds) =
real_fun(comm_data, transport, oidipap)
    if not errorIndication is None  or errorStatus is True:
        print ("Error: %s %s %s %s" % IPap)
    else:
        c = int(1)
        n = len(varBinds)+1
        IPAP = [None] * n
        IPAP[0] = 0

        for var_bind in varBinds:
            IPAP[c] =[x.prettyPrint() for x in var_bind]
            c=c+1




    # Hacemos la peticion SNMP al Servidor de todos los parametros que
queremos
    nentradas = len(IPuser)
```

```
    Name = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.5'],
hlapi.CommunityData('proyecto'), nentradas)
    SO = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.6'],
hlapi.CommunityData('proyecto'), nentradas)
    ruido = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.7'],
hlapi.CommunityData('proyecto'), nentradas)
    TXDataFr = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.8'],
hlapi.CommunityData('proyecto'), nentradas)
    TXDataBy = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.9'],
hlapi.CommunityData('proyecto'), nentradas)
    TXRetries = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.10'],
hlapi.CommunityData('proyecto'), nentradas)
    TXRate = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.11'],
hlapi.CommunityData('proyecto'), nentradas)
    RXDataFr = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.12'],
hlapi.CommunityData('proyecto'), nentradas)
    RXDataBy = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.13'],
hlapi.CommunityData('proyecto'), nentradas)
    RXRetries = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.14'],
hlapi.CommunityData('proyecto'), nentradas)
    RXRate = get_bulk('192.168.89.1',
['1.3.6.1.4.1.14823.2.3.3.1.2.4.1.15'],
hlapi.CommunityData('proyecto'), nentradas)

    # hora
    c = int(1)
    n = int(nentradas + 1)
    timestamp = [None] * n
    timestamp[0] = 0
    while c < n:
        timestamp[c] = hora
        c = c + 1
    # print(timestamp)

    # Nombre del usuario
    c = int(1)
    n = int(nentradas + 1)
    nombre = [None] * n
    nombre[0] = 0

    for it in Name:
        for k, v in it.items():
            nombre[c] = "{1}".format(k, v)
        c = c + 1

    #print(Name)

    # SO
    c = int(1)
    n = int(nentradas + 1)
```

```
    sistema = [None] * n
    sistema[0] = 0

    for it in SO:
        for k, v in it.items():
            sistema[c] = "{1}".format(k, v)
        c = c + 1
    # print(sistema)
    # SeñalRuido
    c = int(1)
    n = int(nentradas + 1)
    snr = [None] * n
    snr[0] = 0

    for it in ruido:
        for k, v in it.items():
            snr[c] = "{1}".format(k, v)
        c = c + 1
    # print(snr)
    # TXDataFrames
    c = int(1)
    n = int(nentradas + 1)
    TXframes = [None] * n
    TXframes[0] = 0

    for it in TXDataFr:
        for k, v in it.items():
            TXframes[c] = "{1}".format(k, v)
        c = c + 1
    # print(TXframes)
    # TXDataBytes
    c = int(1)
    n = int(nentradas + 1)
    TXBytes = [None] * n
    TXBytes[0] = 0

    for it in TXDataBy:
        for k, v in it.items():
            TXBytes[c] = "{1}".format(k, v)
        c = c + 1
    # print(TXBytes)
    # TXRetries
    c = int(1)
    n = int(nentradas + 1)
    TRetries = [None] * n
    TRetries[0] = 0

    for it in TXRetries:
        for k, v in it.items():
            TRetries[c] = "{1}".format(k, v)
        c = c + 1
    # print(TRetries)
    # TXRate
    c = int(1)
    n = int(nentradas + 1)
    TRate = [None] * n
    TRate[0] = 0

    for it in TXRate:
        for k, v in it.items():
```

```
            TRate[c] = "{1}".format(k, v)
        c = c + 1
    # print(TRate)
    # RXDataFrames
    c = int(1)
    n = int(nentradas + 1)
    RXframes = [None] * n
    RXframes[0] = 0

    for it in RXDataFr:
        for k, v in it.items():
            RXframes[c] = "{1}".format(k, v)
        c = c + 1
    # print(RXframes)
    # RXDataBytes
    c = int(1)
    n = int(nentradas + 1)
    RXBytes = [None] * n
    RXBytes[0] = 0

    for it in RXDataBy:
        for k, v in it.items():
            RXBytes[c] = "{1}".format(k, v)
        c = c + 1
    # print(RXBytes)
    # RXRetries
    c = int(1)
    n = int(nentradas + 1)
    RRetries = [None] * n
    RRetries[0] = 0

    for it in RXRetries:
        for k, v in it.items():
            RRetries[c] = "{1}".format(k, v)
        c = c + 1
    # print(RRetries)
    # RXRate
    c = int(1)
    n = int(nentradas + 1)
    RRate = [None] * n
    RRate[0] = 0

    for it in RXRate:
        for k, v in it.items():
            RRate[c] = "{1}".format(k, v)
        c = c + 1
    # print(RRate)
    # Generamos el Dataframe con cada una de las peticiones que hemos
realizado al servidor
    data = [timestamp, MACUSR, MACW, IPuser,IPAP, nombre, sistema,
snr, TXframes, TXBytes, TRetries, TRate, RXframes, RXBytes, RRetries,
RRate]

    df = pd.DataFrame(data, index=['timestamp', 'MAC','MACWLAN', 'IP',
'IPAP', 'Nombre', 'sistema operativo', 'relacion SN', 'TXDataFr',
'TXDataBy',
                                   'TXRetries', 'TXRate', 'RXDataFr',
'RXDataBy', 'RXRetries', 'RXRate'])
    df = df.transpose()
    df.columns = {'timestamp', 'MAC','MACWLAN', 'IP', 'IPAP',
```

```
'Nombre', 'sistema operativo', 'relacion SN', 'TXDataFr', 'TXDataBy',
                'TXRetries', 'TXRate', 'RXDataFr', 'RXDataBy',
'RXRetries', 'RXRate'}
    with open('13062022.csv', 'a') as f:
        df.to_csv(f, mode='a', header=False)

    #df = pd.read_csv('06122021.csv')
    #print(df)
    print(1)
while True:
    try:
        crearcsv()
        print('recolectando datos del servidor: Espere 3 min ')
        time.sleep(180)  # 3 min
    except:
            crearcsv()
```

## Anexo 3. Código fuente script Py análisis exploratorio datos.

Código generación de graficas sin frecuencia

```
import glob
from datetime import datetime, timedelta
from pathlib import Path

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

FIGURES_DIRECTORY = Path("GraficasSinFrecuencia4")
BIN_SIZE = 10  # número de mínutos de ancho de cada barra


def plot_recuento(recuento, accesspoint, fig_path, bin_size,
show=False):
    """recuento tiene dos columns frec y MAC, MAC contiene el número
de MAC observadas
    en el instante o intervalo correspondiente al index"""
    recuento_wide = recuento.pivot(columns="frec", values="MAC")
    recuento_wide.fillna(0, inplace=True)

    fig, ax = plt.subplots()
    if 2.4 in recuento_wide.columns:
        print('OK')
    else:
        recuento_wide[2.4] = 0
    if 5.0 in recuento_wide.columns:
        print('OK')
    else:
        recuento_wide[5.0] = 0
    ax.bar(
```

```
        recuento_wide.index,
        (recuento_wide[2.4]+recuento_wide[5.0]),
        label="Usuarios",
        width=timedelta(minutes=bin_size),
    )
    # ax.bar(
    #     recuento_wide.index,
    #     recuento_wide[5.0],
    #     label="Frec 5.0",
    #     width=timedelta(minutes=bin_size),
    #     bottom=recuento_wide[2.4],
    # #)

    ax.legend()
    ax.set_xlabel("Tiempo de muestreo", loc="right")
    ax.grid(axis="both", visible=None, which="major")
    ax.tick_params(axis="x", rotation=60)

    min_sampling_time = recuento.index.min()
    max_sampling_time = recuento.index.max()

    left_xlim = min_sampling_time.replace(hour=8, minute=0, second=0)
    right_xlim = max_sampling_time.replace(hour=22, minute=0,
second=0)
    ax.set_xlim([left_xlim, right_xlim])
    ticks_sequence = pd.date_range(start=left_xlim, end=right_xlim,
freq="1H")
    ax.set_xticks(ticks_sequence,
labels=ticks_sequence.strftime("%H:%M"))

    ax.set_title(
        f"Nombre del AP: {accesspoint}, Fecha:
{left_xlim.strftime('%d-%m-%Y')}"
    )

    ax.set_ylabel("N° de usuarios simultaneos")

    fig.tight_layout()

    plt.savefig(fig_path)

    if show:
        plt.show()

    plt.close(fig)


csv_files = glob.glob("csvfinales/*31052022frecuencias.csv")

for filename in csv_files:

    data = pd.read_csv(
        filename,
        usecols=["timestamp", "MAC", "IPAP", "frec"],
        dtype={"frec": float},
    )

    data.replace({"timestamp": "0"}, np.nan, inplace=True)

    print(data.dtypes)
    ###Arreglamos el csv eliminando todos los NaN y espacios blancos
```

```
para que no de problemas a la hora de procesarlo
    # data.drop(['index'],axis=1)
    data.replace("", np.nan, inplace=True)
    # data.replace(0, np.nan, inplace=True)
    print(data)
    data = data.dropna(subset=["timestamp", "IPAP"])

    data["AP"] = data["IPAP"].str.extract(r" ([\d\.]+)']$")
    data['timestamp'] = pd.to_datetime(data['timestamp'], format="%Y-
%m-%d %H:%M:%S")
    data["floor_timestamp"] =
data["timestamp"].dt.floor(f"{BIN_SIZE}min")

    for i, (accesspoint, datos_AP) in enumerate(data.groupby("AP")):
        # if i > 0:
        #     break

        recuento = datos_AP.groupby(["timestamp", "frec"]).count()
        recuento.reset_index("frec", inplace=True)

        recuento = recuento.loc[:, ["frec", "MAC"]]

        fecha = recuento.index[0].strftime('%d-%m-%Y')
        bin_figures_directory = FIGURES_DIRECTORY / f"{BIN_SIZE}-
minutes-bin"
        bin_figures_directory.mkdir(parents=True, exist_ok=True)

        plot_recuento(
            recuento,
            accesspoint,
            FIGURES_DIRECTORY / f"AP-{accesspoint}-{fecha}.png",
            bin_size=2,
            show=False,
        )

        recuento = datos_AP.groupby(["floor_timestamp",
"frec"]).count()
        recuento.reset_index("frec", inplace=True)

        recuento = recuento.loc[:, ["frec", "MAC"]]

        plot_recuento(
            recuento,
            accesspoint,
            bin_figures_directory / f"AP-{accesspoint}-{fecha}.png",
            bin_size=10,
            show=False,
        )
```

Código generación de graficas por frecuencia

```
# %%
import glob
from datetime import datetime, timedelta
from pathlib import Path

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

```
FIGURES_DIRECTORY = Path("GraficasConFrecuencia4")
BIN_SIZE = 10  # número de mínutos de ancho de cada barra


def plot_recuento(recuento, accesspoint, fig_path, bin_size,
show=False):
    """recuento tiene dos columns frec y MAC, MAC contiene el número
de MAC observadas
    en el instante o intervalo correspondiente al index"""
    frecuencias_observadas = np.sort(recuento["frec"].unique())
    print(f"frecuencias observadas {frecuencias_observadas}")
    recuento_wide = recuento.pivot(columns="frec", values="MAC")
    recuento_wide.fillna(0, inplace=True)

    fig, ax = plt.subplots()

    ax.bar(
        recuento_wide.index,
        recuento_wide[frecuencias_observadas[0]],
        label=f"Frec {frecuencias_observadas[0]}",
        width=timedelta(minutes=bin_size),
    )

    if len(frecuencias_observadas) > 1:
        ax.bar(
            recuento_wide.index,
            recuento_wide[5.0],
            label="Frec 5.0",
            width=timedelta(minutes=bin_size),
            bottom=recuento_wide[2.4],
        )

    ax.legend()
    ax.set_xlabel("Tiempo de muestreo", loc="right")
    ax.grid(axis="both", visible=None, which="major")
    ax.tick_params(axis="x", rotation=60)

    min_sampling_time = recuento.index.min()
    max_sampling_time = recuento.index.max()

    left_xlim = min_sampling_time.replace(hour=8, minute=0, second=0)
    right_xlim = max_sampling_time.replace(hour=22, minute=0,
second=0)
    ax.set_xlim([left_xlim, right_xlim])
    ticks_sequence = pd.date_range(start=left_xlim, end=right_xlim,
freq="1H")
    ax.set_xticks(ticks_sequence,
labels=ticks_sequence.strftime("%H:%M"))

    ax.set_title(
        f"Nombre del AP: {accesspoint}, Fecha:
{left_xlim.strftime('%d-%m-%Y')}"
    )

    ax.set_ylabel("N° de usuarios simultaneos")

    fig.tight_layout()

    plt.savefig(fig_path)

    if show:
```

```
        plt.show()

    plt.close(fig)


csv_files = glob.glob("csvfinales/*31052022frecuencias.csv")
# csv_files = ["2603frecuencia.csv"]
for filename in csv_files:

    data = pd.read_csv(
        filename,
        usecols=["timestamp", "MAC", "IPAP", "frec"],
        dtype={"frec": float},
    )

    data.replace({"timestamp": "0"}, np.nan, inplace=True)

    print(data.dtypes)
    ###Arreglamos el csv eliminando todos los NaN y espacios blancos
para que no de problemas a la hora de procesarlo
    # data.drop(['index'],axis=1)
    data.replace("", np.nan, inplace=True)
    # data.replace(0, np.nan, inplace=True)
    print(data)
    data = data.dropna(subset=["timestamp", "IPAP"])

    data["AP"] = data["IPAP"].str.extract(r" ([\d\.]+)']$")
    data['timestamp'] = pd.to_datetime(data['timestamp'], format="%Y-
%m-%d %H:%M:%S")
    data["floor_timestamp"] =
data["timestamp"].dt.floor(f"{BIN_SIZE}min")

    for i, (accesspoint, datos_AP) in enumerate(data.groupby("AP")):
        # if i > 0:
        #     break

        recuento = datos_AP.groupby(["timestamp", "frec"]).count()
        recuento.reset_index("frec", inplace=True)

        recuento = recuento.loc[:, ["frec", "MAC"]]

        fecha = recuento.index[0].strftime('%d-%m-%Y')
        bin_figures_directory = FIGURES_DIRECTORY / f"{BIN_SIZE}-
minutes-bin"
        bin_figures_directory.mkdir(parents=True, exist_ok=True)

        plot_recuento(
            recuento,
            accesspoint,
            FIGURES_DIRECTORY / f"AP-{accesspoint}-{fecha}.png",
            bin_size=2,
            show=False,
        )

        recuento = datos_AP.groupby(["floor_timestamp",
"frec"]).count()
        recuento.reset_index("frec", inplace=True)

        recuento = recuento.loc[:, ["frec", "MAC"]]

        plot_recuento(
```

```
            recuento,
            accesspoint,
            bin_figures_directory / f"AP-{accesspoint}-{fecha}.png",
            bin_size=10,
            show=False,
        )
```