Universitat Oberta de Catalunya

uoc.edu

Implementación de un punto de acceso residencial con servicio de monitorización WiFi

Sergio Díaz Camacho
Máster Universitario en Ingeniería de Telecomunicación UOC-URL
Sistemas de Comunicación

Profesor Colaborador: Raúl Parada Medina

Profesor Responsable: Carlos Monzo Sánchez

Junio 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Implementación de un punto de acceso residencial con servicio de monitorización WiFi*
Nombre del autor:	*Sergio Díaz Camacho*
Nombre del consultor/a:	*Raúl Parada Medina*
Nombre del PRA:	*Carlos Monzo Sánchez*
Fecha de entrega (mm/aaaa):	*06/2019*
Titulación:	*Máster Universitario en Ingeniería de Telecomunicación UOC-URL*
Área del Trabajo Final:	*Sistemas de Comunicación*
Idioma del trabajo:	*Castellano*
Palabras clave	*Acceso WiFi Monitorizar Seguridad Autenticar*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

El boom de internet y el auge de la tecnología WiFi ha cambiado la forma y patrones de navegación de los usuarios. De un usuario que se conectaba a internet gracias a un PC cableado a su *router* y hacía un uso puntual de la conexión de banda ancha, a un usuario que se conecta de forma inalámbrica al *router* y hace un uso constante de la conexión. Esto ha provocado una cierta sensación de inseguridad o descontrol en los usuarios, ya que las conexiones inalámbricas no son tangibles. Quién o qué está conectado a la conexión WiFi es una pregunta recurrente hoy en día.

Actualmente existen diferentes soluciones aisladas, incompletas o de pago, para responder a la pregunta planteada. Los *softwares* de control comerciales consultados se muestran incompletos (únicamente permiten o deniegan el servicio) y no monitorizan el uso del WiFi.

Este proyecto se apoya en un microordenador Raspeberry de bajo coste y de sistema operativo Raspbian totalmente gratuito (tecnología creada por The Raspberry Pi Foundation con fines académicos) y sobre programas libres y *scripts* de código abierto. Con esta base se implementa un punto de acceso totalmente operativo con funcionalidades de:

- Autenticación.
- Monitorización del entorno WiFi.
- Validación de clientes confiables.
- Envío de alertas en base a eventos.

De esta forma, este trabajo viene a dotar al usuario del conocimiento suficiente para implementar un sencillo punto de acceso residencial con funcionalidades básicas de monitorización y seguridad sin tener que depender de diferentes softwares de pago de terceros.

Abstract (in English, 250 words or less):

Internet boom and WiFi technology emergence have changed the way that final customers use their devices. Few years ago, users typically owned a laptop which was wired to the router. Furthermore, this laptop was used in short periods of time or for a few tasks. Nowadays that has changed: a continued use of electronic devices is common. Moreover, wireless is the preferred connection for these devices. This fact has provoked a feeling of insecurity in the final users due to wireless connections are not properly known. Who o which device is connected to the router would be a common question.

There are several options in order to answer the question raised. The solutions that can be found are incomplete or 'pay-per-use'.

This project is supported by a cheap Raspeberry Pi single board computer and its operating system Raspbian (technology developed by The Raspberry Pi Foundation with academic purposes). In addition, the scripts presented in this document are open source. In view of the above this project gives the necessary instructions in order to implement an WiFi access point with the following functionalities:

- Authentication
- WiFi monitoring
- WiFi client validation
- Alert monitoring and notification

Thanks to this project final users will be able to configure an easy access point with a Raspberry Pi board.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Actualmente, la tecnología WiFi se ha convertido en indispensable para la gran mayoría de los usuarios: PCs, *smartphones*, *tablets*, impresoras, elementos de domótica, vigilancia enel hogar… hacen uso de esta tecnología. Una tecnología robusta, rápida, altamente instaurada y en constante evolución. Sin embargo, también es una completa desconocida para la gran mayoría de las personas que la utilizan. Una de sus grandes ventajas, apoyarse en un medio no guiado, lo que confiere movilidad al usuario, es también uno de sus aspectos más críticos. El saber quién o qué está intentando conectarse a nuestra red, o ya está conectado, es uno de los puntos que los usuarios más reclaman.

Hoy en día, el usuario puede consultar que equipos están conectados a su interfaz WiFi. Esto se puede hacer desde la propia GUI del *router* hasta en aplicaciones de terminales móviles. Sin embargo, estas funcionalidades son raramente utilizadas por los usuarios por puro desconocimiento o por falta de formación a la hora de interpretar los datos. Además, las funcionalidades comentadas están orientadas principalmente a la visualización de la información, pero no a la monitorización y aviso al usuario en caso de que haya equipamiento desconocido conectado al interfaz WiFi. Por otro lado, esta el término uso del aire u ocupación del aire cuando se habla de las conexiones inalámbricas. Estos términos hacen referencia a cómo se gestionan los recursos WiFi cuando hay conexiones en dicho interfaz. Aplicaciones para controlar estas conexiones (los denominados *sniffer* WiFi) hay muchas y variadas. Sin embargo, todas tienen un carácter académico/técnico (no comercial) y no son fácilmente interpretables. Por tanto, el comentado uso u ocupación del aire es una realidad ajena completamente al usuario medio, debido a que no sabe como funciona la tecnología WiFi y el estándar 802.11 en el que se apoya. Esto hace que el usuario quede totalmente indefenso a la hora de vigilar, optimizar y tomar decisiones en base a sus dispositivos y el uso que se hace de la conexión.

Como se puede apreciar, los usuarios hacen un uso masivo de una tecnología desconocida para ellos. En caso de querer una red de área local (LAN) WiFi optimizada y monitorizada deben de hacer uso de uno o varios SW, usualmente de pago, que no tienen todas las funcionalidades requeridas (autenticación, ocupación del aire y envío de notificaciones) para una tecnología inalámbrica.

Este proyecto, por tanto, viene a dotar al usuario del conocimiento suficiente para implementar un sencillo punto de acceso residencial con funcionalidades

básicas de monitorización y seguridad sin tener que depender de diferentes softwares de pago de terceros.

1.2 Objetivos del Trabajo

El presente proyecto tiene como objetivos:

- Dar al usuario unas pautas claras y sencillas para tener un punto de acceso WiFi totalmente operativo y monitorizado.
- Acercar la tecnología WiFi al usuario final.

El punto de acceso tendrá las capacidades de:

- Interfaz WiFi completo (SSID y contraseña).
- *Sniffer* para monitorizar la ocupación del aire y el uso del interfaz WiFi.
- Contabilización de ataque WiFi.
- Englobar una lista de equipos confiables.
- Mandar un mail según una serie de eventos predefinidos.

El proyecto se apoyará en:

- Microordenador Raspberry Pi: Tecnología de alta capacidad computacional y de bajo coste.
- Sistema operativo Raspbian: Sistema operativo gratuito, proveído por *The Raspberry Pi Foundation* con fines académicos.
- *Scripts* de código abierto: *Scripts* públicos que pueden ser modificados y usados libremente.
- Programas gratuitos: Programas totalmente funcionales de libre uso y gratuitos.

1.3 Enfoque y método seguido

El proyecto englobará cuatro partes diferenciadas:

1. Estudio del Hardware: Dentro de las opciones que da Raspberry, se estudiará que placa se ajusta mejor a las necesidades para realizar el trabajo en concreto. A su vez, se tendrá que escoger que tarjeta SD será la elegida para el proyecto.

2. Creación de *scripts* y elección de programas: Se deberá estudiar que programas y *scripts* albergará la Raspberry Pi para tener un sistema completo y operativo. Para ello se tendrá en cuenta las posibilidades técnicas que hay actualmente para realizar las funciones planteadas anteriormente.

3. Montaje: Se creará una guía de usuario donde, paso por paso, se explicarán todos los puntos a tener en cuenta para tener un primer montaje funcional.

4. Pruebas Funcionales: Se definirán y se pasarán una serie de pruebas. De esta forma se estudiarán los límites del montaje y se cambiarán, si fuese pertinente, la parametrización cargada.

1.4 Planificación del Trabajo

A continuación, se listan los pasos a realizar a lo largo del trabajo:

1. Raspberry y Raspbian: Se deberá estudiar que placa, SD y SO son los más apropiados para el proyecto. Se tendrá en cuenta:

    a. Funcionalidades: El HW escogido debe ser lo suficientemente potente como para poder soportar todos los puntos explicados en el ámbito del proyecto.

    b. Precio: Al tener un carácter académico, se evaluarán las distintas opciones buscando la que mejor calidad/precio presente.

    c. SO: Se escogerá de entre los distintos SO Raspbian disponibles, teniendo en cuenta las funcionalidades a desarrollar.

2. *Scripts* y Programas: Se deberá definir que funcionalidades se implementarán, acotando de forma clara un primer alcance del proyecto.

    a. Investigación: Se deberá hacer un estudio de que programas hay disponibles para realizar las funcionalidades a implementar. Asimismo, se estudiará que líneas de investigación o proyectos análogos se han realizado anteriormente. De esta forma, se podrá evaluar que líneas de desarrollo son más interesantes y aportan más al ámbito académico.

    b. Definición: Se definirán detalladamente las funcionalidades a desarrollar teniendo en cuenta el punto anterior.

    c. Desarrollo: Se desarrollarán las funcionalidades que serán implementadas en el diseño final.

3. Batería de pruebas: Con el fin de probar de forma precisa todas las funcionalidades y fiabilidad del diseño, se definirán una serie de pruebas.

4. Guía de Usuario: A lo largo del proyecto, se irá creando una guía de usuario donde se recojan, paso por paso, todos los puntos relevantes.

5. Valoración: Se valorará económicamente el montaje final.

6. Redacción del trabajo: Se hará una memoria del trabajo, al igual que una presentación de este, donde se recogerán todos los puntos relevantes y conclusiones.

Figura 1 - Plan de Trabajo

1.5 Breve sumario de productos obtenidos

En este proyecto se ha conseguido implementar un sencillo punto de acceso WiFi gracias a una tecnología al alcance del usuario medio, las placas Raspberry Pi. Además, se han desarrollado dos servicios los cuales pueden permitir a cualquier usuario tener un control de su WiFi:

- Ocupación de canal: Con un sencillo *script* en lenguaje bash se ha conseguido desarrollar un servicio por el cual el usuario final es avisado vía *mail* si cierto umbral, parametrizable, de ocupación de canal es superado. De esta forma el usuario puede tomar decisiones respecto al interfaz WiFi.

- Usuarios conectados: Con dos *scripts,* uno en lenguaje bash y otro en Python, se ha conseguido un servicio por el cual el usuario final es notificado vía *mail* si:
    - Se ha conectado correctamente un equipo desconocido.
    - Se ha intentado conectar, de forma infructuosa, un equipo.

Estos dos servicios se han definido de tal forma que son lanzados de forma automática por la propia Raspberry Pi sin necesitar ningún tipo de acción por parte del usuario. Tras diversas pruebas funcionales, se ha conseguido que el montaje final sea estable en largos periodos de tiempos:

- Raspberry Pi funcionando como punto de acceso durante semanas, sin merma alguna en sus capacidades.

- Con los servicios funcionando durante tres cuartas partes del día de forma concurrente.

Adicionalmente, se ha desarrollado una guía de usuario donde se detalla, paso por paso, todos los puntos que hay que seguir para tener lo anteriormente comentado. A destacar que la dificultad de la guía es mínima, de tal forma que no hay que tener conocimiento previo de la tecnología WiFi o de lenguajes de programación.

1.6 Breve descripción de los otros capítulos de la memoria

El documento se ha estructurado en tres grandes bloques:

- Descripción y montaje HW.
- Descripción y desarrollo SW.
- Pruebas de usuario y estabilidad.

Partiendo de esta base, tendríamos:

- Capítulo 2. Raspberry y Raspbian: Este capítulo se hace una pequeña introducción al ecosistema Raspberry Pi. Más adelantes, se detallan las opciones escogidas, tanto HW como de SO, para el desarrollo del proyecto.

- Capítulo 3. Proyecto y funcionalidades a desarrollar: En este capítulo se realiza una investigación donde se tratan dos puntos fundamentales:
    - Herramientas existentes actualmente en las que, en parte o totalmente, el proyecto se puede sustentar.
    - Proyectos análogos al presente. En este punto se hallan similitudes y diferencias respecto al enfoque y finalidad del desarrollo de este proyecto.

  Tras estos dos puntos de estudio, se desarrollan los servicios finales a implementar.

- Capítulo 4. Batería de Pruebas: Este punto se detallan y realizan una serie de pruebas. Con estos *tests* se buscan dos puntos:
    - La comprobación de que cada una de las herramientas utilizadas y servicios desarrollados se comportan correctamente tanto de forma individual como cuando trabajan de forma orquestada.
    - Se estudian los límites del funcionamiento del montaje, analizando y resolviendo, en la medida de lo posible, los puntos de fallo que afloren por restricciones ya sean HW o SW.

- Capítulo 5. Impedimentos Encontrados: Muy ligado al punto anterior, tenemos el capítulo 5. En este apartado se resumen los impedimentos encontrados, buscando los posibles orígenes del problema.

- Capítulo 6. Guía de Usuario: Uno de los puntos clave del proyecto es acercar la tecnología al usuario medio. Con esta pequeña guía, se indica como implementar un punto de acceso WiFi totalmente funcional con servicio de alertas integrado. Esta guía se enfoca de tal forma que el usuario final no deba porqué tener conocimientos técnicos previos para poder seguirla.

- Capítulo 7. Inversión Económica Final: En este capitulo se hace un pequeño desglose de la inversión económica que un usuario pudiera realizar este montaje.

- Capítulo 8. Conclusiones: En este apartado se hace una valoración del proyecto en sí, teniendo en cuenta los objetivos del proyecto. Además, se hará una revisión de en qué medida se ha seguido la metodología. De forma adicional, se especifican ciertos puntos de mejora.

2. Raspberry Pi y Raspbian

The Raspberry Pi Foundation es una empresa asentada en Reino Unido la cual fue la creadora de los microordenadores Raspberry Pi y de su sistema operativo oficial, Raspbian (basado en Debian). Esta compañía tiene como misión la de crear pequeños ordenadores de bajo coste y alta capacidad, con fines académicos. Desde su fundación en 2009 han conseguido vender millones de unidades de los distintos modelos de Raspeberry Pi, creando una comunidad de usuarios muy activa y organizada.

2.1 Elección de placa

Lo que comúnmente se conoce como Raspberry Pi es, básicamente, un ordenador de placa simple (en inglés, *Single Board Computer*, o SBC en su abreviatura). Este pequeño ordenador ha ido evolucionando a lo largo de los años, adaptándose a las necesidades de los usuarios e implementando ciertos avances tecnológicos. En concreto hay ocho evoluciones de las Raspeberry Pi, siendo la más nueva la Raspberry Pi 3 Model B+. Este modelo es el más completo hasta la fecha y el elegido para el desarrollo de este proyecto. Esto es debido a dos razones:

- Tecnológicas: Frente a su predecesor, la Raspberry Pi 3 Model B (lanzada en 2016 y hasta la fecha la SBC más completa), tiene las siguientes mejoras:
    - Un procesador más potente, de 1.4 GHz, frente a los 1.2 GHz de la Raspberry Pi Model B.
    - WiFi dual band, 802.11b/g/n/ac, en 2.4GHz y 5GHz. La Raspberry Pi Model B solo trabaja en 2.4GHz (sin estándar 802.11ac).
    - Bluetooth 4.2, frente al Bluetooth 4.1 de su predecesora.
    - Puerto ethernet con capacidad de hasta 300 Mbps. El modelo anterior solo tenía capacidad de 100 Mbps.

- Precio: La Raspberry Pi 3 Model B+ conserva el mismo precio que predecesora, oscilando este (según el *reseller* consultado) entre los 38€ y los 40€.

Figura 2 - Raspberry Pi 3 Model B+ https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

De forma añadida a lo anteriormente comentado, las especificaciones de la Raspeberry Pi 3 B+ serían las siguientes:

- Procesador Quad-Core Cortex A53 de 1.4GHz
- 4 puertos USB 2.0
- 40 pines GPIO
- 1 puerto HDMI
- 1 puerto Ethernet (PoE)
- 1 combo audio/mic
- 1 interfaz de cámara (CSI)
- 1 interfaz de pantalla (DSI)
- 1 núcleo gráfico 3D
- Dual-band 802.11ac *Wireless* LAN y Bluetooth 4.2

Figura 3 - Raspberry Pi 3 Model B+ Front
https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

2.2 Elección de memoria SD

La tarjeta SD es uno de los puntos clave en una Raspberry Pi. Esta tarjeta será la que albergue el sistema operativo y todos los ficheros generados. Como mínimo se deberá tener una SD de 8GB, aunque se recomienda optar por una de 16 GB. Muchos de los modelos de Raspberry vienen con una tarjeta incorporada (con un Raspbian ya preconfigurado).
Específicamente, para el modelo 3 B+, la tarjeta que se debe integrar es del tipo microSD. Para este proyecto se ha optado por una microSD Integral de 16GB, con capacidad de escritura de 10MB/sec.

Figura 4 - SD Card
https://www.amazon.es/Integral-Tarjeta-memoria-microSDHC-adaptador/dp/B004IIA488

2.3 Elección de Sistema Operativo (SO)

Las Raspeberry Pi trabajan con un sistema operativo propio denominado Raspbian. Este software, de código abierto, es una distribución Linux basada en Debian. Sin embargo, la Raspberry Pi soporta otros sistemas como Ubuntu, Windows 10 IoT Core, RISC OS… Esto hace que el "ecosistema" Raspeberry sea rico en alternativas, cada una de ellas enfocada a un uso o unas especificaciones concretas del usuario final. Para este proyecto, por simplicidad

y practicidad, se ha utilizado la última versión Raspbian: *RaspbianStretch with desktop and recommended software*.

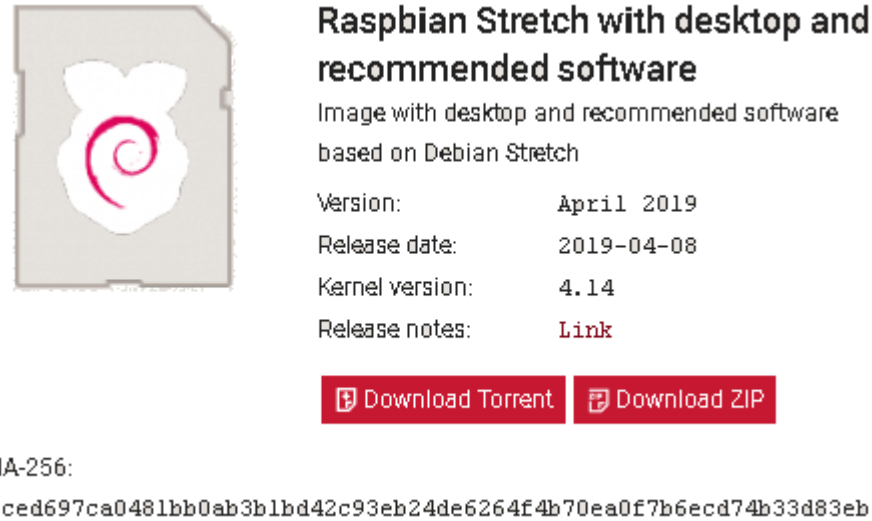


Figura 5 - Raspbian https://www.raspberrypi.org/downloads/raspbian/

2.4 WiFi *Dongle*

Para poder analizar tráfico WiFi primeramente se debe poder capturar sus tramas. La Raspberry Pi, como tal, tiene integrado dos interfaces WiFi, uno correspondiente a la banda de 2.4GHz y otro a la banda de 5GHz. Sin embargo, estos interfaces no vienen configurados en modo *monitor*, requisito indispensable si se quiere capturar las transmisiones WiFi. Aún probando distintas configuraciones que habilitan el comentado modo *monitor* en la Raspberry Pi, no se ha conseguido capturar tráfico WiFi. Esto puede ser por el tipo de *chipset* Broadcom que la Raspberry Pi monta, el cual no es capaz de capturar las tramas buscadas. Para suplir esta deficiencia técnica, se ha optado por incorporar un *dongle* WiFi, para que haga de *sniffer*, al montaje final.

Figura 6 - WiFi Dongle https://www.wifi-antennas.co.uk/alfa-awus036nh2w-2-4ghz-150mbps-usb-wifi-dongle-with-detachable-5dbi-rp-sma-antenna.html

El equipo escogido, por su simplicidad y precio, es el Alfa AWUS036NH. Este equipo está dotado con un chipset WiFi Ralink RT3070. La antena es 1x1, es decir, una antena transmisora y una antena receptora. A efectos prácticos, no tiene relevancia a la hora de realizar tareas de *sniffer*.

2.5 Montaje Hardware

Como se ha visto en el punto anterior, la Raspberry Pi cuenta con diferentes recursos físicos para poder trabajar con periféricos de distinta índole. En un vistazo rápido, la placa escogida para este proyecto tendría:

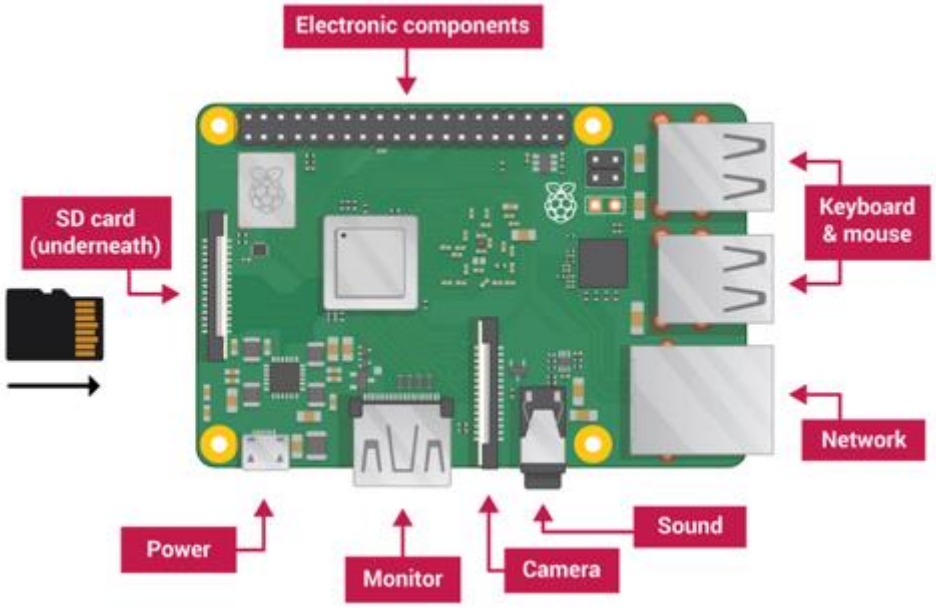


Figura 7 - Componentes Placa

Particularizando para el montaje del proyecto, se utilizarán las entradas:

- Power: Para alimentar la placa.
- Monitor: Para trabajar sobre el entorno gráfico que nos ofrece Raspbian.
- Entrada USB 1: Para conectar un pequeño transmisor USB que hará de anclaje para el ratón y el teclado.
- Entrada USB 2: Para conectar el *dongle*, el cual nos permitirá capturar y analizar tramas WiFi.
- Ethernet: Para tener conectividad hacia el *router* principal.

El montaje final sería el siguiente:

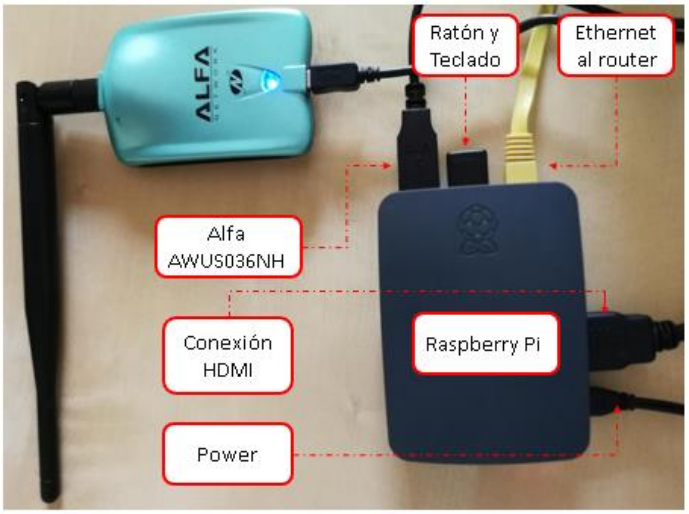


Figura 8 - Montaje Final

3. Proyecto y funcionalidades a desarrollar

En este apartado se hará una revisión completa del proyecto a abordar:

- Visión final del proyecto, especificando el ideal de este.
- Funcionalidades en las que se va a apoyar el proyecto.
- Proyectos análogos al que se va a desarrollar.

De esta forma se tendrá una idea de que grado de novedad y que demanda viene a satisfacer el proyecto que se va a realizar.

3.1 Proyecto Final

En el presente proyecto hay cinco grandes funcionalidades a desarrollar sobre la placa Raspberry Pi. Sobre estas actividades, y su interacción entre ellas, se sustentará el funcionamiento del prototipo final ideado. A grandes rasgos, se tendría:

- La funcionalidad de punto de acceso.
- *Sniffer* para monitorizar la ocupación del aire y el uso del interfaz WiFi.
- Funcionalidad de contabilización de contraseñas erróneas.
- Funcionalidad de listado de equipos confiables.
- Envío de mails en base a eventos.

- Funcionalidad de Punto de Acceso:

Englobaría la funcionalidad de Punto de Acceso WiFi: validará y autenticará a los usuarios de acuerdo al estándar de seguridad y encriptación elegido. Creará un SSID en la banda de 2.4GHz bajo el estándar 802.11n.

- *Sniffer* WiFi:

Una herramienta capaz de monitorizar las comunicaciones WiFi que se hagan en el entorno del *router*. De esta forma se podrá controlar que equipos reportan un mayor uso del espectro WiFi, así cómo las autenticaciones fallidas detectadas.

- Listado de Equipos Confiables:

Se elaborará una lista de equipos confiables, es decir, un listado donde se tengan las MACs de aquellos dispositivos los cuales el administrador reconozca como suyos o como usuarios habituales. Esta lista no aplicará ninguna restricción en cuanto a la autenticación y conexión de usuarios al interfaz WiFi, ni al uso que hagan estos usuarios de este interfaz.

- Contabilización de Ataque WiFi:

Se recogerán las veces que un usuario ha introducido una contraseña errónea a la hora de autenticarse. De esta forma, se tendrá un sencillo mecanismo que permitirá tener un primer esbozo de cuantas veces se está intentando, a priori, romper por fuerza bruta la seguridad de nuestra red WiFi.

- Envío de Mails:

Se realizarán tareas de envío de mails automáticas según unos eventos definidos. Estos eventos se formarán según tres grandes puntos:

1. Equipo confiable: Si un usuario fuera de la lista de equipos confiables se conecta a nuestra red WiFi, se generará un envío de un mail predeterminado advirtiendo de está cuestión. Como antes se ha comentado, este cliente "inusual" podrá hacer un uso normal del interfaz WiFi ya que no se le aplicará ninguna restricción.

2. Ocupación del canal: En base a la ocupación del canal, y gracias a un *sniffer* WiFi, se enviará un mail automático cuando se detecte una saturación en el canal de emisión. Esto permitirá al usuario tomar decisiones.

3. Contabilización de ataques: Gracias al *sniffer* WiFi, se contabilizarán las veces que un cliente se autentica erróneamente en el interfaz WiFi. Este hecho se notificará automáticamente vía mail.

De forma resumida, tendríamos un montaje final como el siguiente:

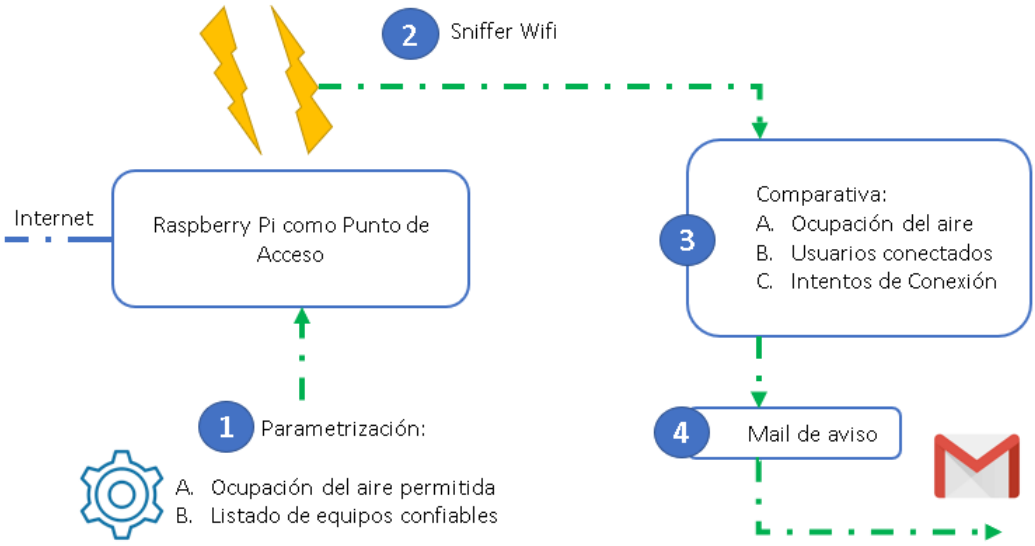


Figura 9 - Servicio Completo

3.2 Investigación sobre funcionalidades

En mayor o menor medida, existen funcionalidades ya definidas por usuarios o programas comerciales y/o académicos que desarrollan las acciones anteriormente comentadas. A continuación, se hace un breve resumen de aquellos programas o tutoriales que desarrollan, de forma aproximada, las actividades a desarrollar en la placa Raspberry Pi:

- Funcionalidad de *Router*:

Uno de los usos más extendidos de las placas Raspberry es como *router*. Este uso de la placa está ampliamente documentado en internet, con mayor o menor detalle, sobre distintos SO Raspbian. Así, se pueden encontrar ejemplos a modo de tutoriales gratuitos como los que a continuación se en *Instructables* [1] o en *GitHub* [2]

- *Sniffe*r WiFi:

El programa más ampliamente utilizado en el ámbito académico sería *Wireshark* [3]. Esta herramienta, de uso libre, es un analizador de protocolos: captura, en base a una parametrización flexible, los paquetes que fluyen en una red o interfaz determinado. Una herramienta más ligera y orientada a terminal (CLI) sería *TShark* [4]. Esta herramienta también está dentro del dominio de *Wireshark*.

Otra herramienta, también de descarga y uso libre, sería *Tcdump* [5] Al igual que *TShark*, es una herramienta por línea de comandos (CLI) que permite capturar los paquetes en tiempo real transmitidos en una red o interfaz indicado.

- Listado de Equipos Confiables:

Actualmente, hay distintas formas de validar la confiabilidad de los usuarios. La más sencilla y extendida en el ámbito residencial sería el uso de una contraseña para conectarse al interfaz WiFi: una vez el usuario elige un SSID, debe introducir una contraseña, conocida a priori, para validarse. Otra opción más robusta que la anteriormente expuesta sería la de filtrado MAC. Tras la introducción de la contraseña por parte del usuario, se valida si la MAC del cliente pertenece a una lista blanca o negra, según la implementación del filtrado, donde se le permitirá o no la conexión. Como último ejemplo tendríamos la implementación de un servidor RADIUS. El protocolo RADIUS gestiona la autenticación y la autorización de clientes sobre un sistema

determinado. Como punto positivo, el servidor RADIUS, además, asigna unas credenciales individuales para cada usuario.
Estas tres formas de validación de usuarios tienen en común la, a priori, nula información que se envía al propietario del sistema cuando la autenticación ha sido correcta.

- Contabilización de Ataque WiFi

En internet se pueden encontrar diversos programas que monitorizan el interfaz WiFi, detectando y previniendo los intrusos.

Waidps [6] cumpliría los tres puntos comentados anteriormente. Esta herramienta permite realizar pruebas de penetración WiFi, detección de intrusos e, incluso, prevención y detención de ataques.

Otra solución sería *Kismet* [7] Esta herramienta monitoriza y, entre otras aplicaciones, detecta posibles intrusos en la red WiFi (WIDS – Wireless intrusion detection).

- Envío de Mails:

Para esta funcionalidad, hay infinidad de opciones disponibles. Por ligereza y flexibilidad, los *mails client* por línea de comando son idóneos para la automatización del envío de mails.

Sendmail [8] es un *mail transfer agent* (MTA) que soporta infinidad de tipos de transferencia de mail, entre ellos SMTP. Este programa, con modalidad de uso libre, permite, de forma sencilla enviar mails por línea de comandos.

Otra opción disponible, y ampliamente utilizada, sería Mutt [9] Este programa configura un *mail user agent* (MUA) para sistemas operativos basados en Unix. Una de las grandes ventajas de este programa es el pequeño espacio que ocupa, ideal por tanto para Raspberry Pi.

Por último, tendríamos *SSMTP* [10], el cual es un *mail transfer agent* (MTA) simple de configurar. SSMTP es ligero, fácil de configurar y, sobre todo, demanda pocos recursos computacionales.

3.3 Investigación sobre proyectos análogos

A continuación, se hace un breve resumen de aquellos proyectos análogos a este Trabajo Final de Master, o con similitudes en alguno de sus puntos de investigación:

- PiMonitor: A Wi-Fi Monitoring Device Based on Raspberry-Pi:

En este trabajo, Milad Ghantous et al [11], proponen una herramienta de control de consumo de datos sobre el interfaz WiFi por parte los usuarios. Para realizar este proyecto, se apoyan en una Raspberry Pi la cual hará de *sniffer* del interfaz WiFi. De forma periódica, la Rasberry Pi manda los paquetes capturados para que se haga una disección de ellos, con la cual se puede controla que MACs (y por tanto usuarios) están haciendo un mayor uso del interfaz WiFi. Además, en este diseño se implementa una aplicación móvil por la cual el dueño del sistema puede imponer restricciones de navegación, a nivel de datos transmitidos, a los usuarios.

- Building up knowledge through passive WiFi probes:

En este caso, Redondi y Cesana [12], conforman un sistema para caracterizar el ambiente WiFi de una zona. Para ello se valen de Raspberry Pi utilizada como *sniffer* WiFi. Gracias a estas placas base, obtienen información que, posteriormente, es tratada.

- Low cost smart security camera with night vision capability using Raspberry Pi and OpenCV:

En este proyecto, Feipeng et al [13], implementan un sistema de seguridad apoyado sobre Raspberry Pi. La placa Raspberry Pi es utilizada como centro neurálgico del sistema, siendo está la que configura la parametrización de los distintos *triggers* del sistema de seguridad. De forma adicional, la Raspberry Pi es la encargada de lanzar una notificación, gracias a un servidor SMTP, según los eventos detectados.

- Raspberry *Sniffing*: El juguete oculto:

En la web hacking-etico, Camacho [14], propone una configuración sobre Raspberry Pi que permite, tras insertarla en una red, capturar paquetes. Estos paquetes, posteriormente, son enviados automáticamente con un mail para su posterior estudio.

3.4 Comparativa con otros proyectos

Como se puede apreciar, hay proyectos, análogos al que se encuadra en el presente Trabajo Final de Máster, ya implementados y de acceso libre. Proyectos o propuestas de calidad donde se abordan distintas cuestiones como la monitorización del espectro WiFi, un control exhaustivo de usuarios que hacen uso del interfaz, la detección de eventos y el envío de mails en automáticos en base a estos.

Todos los proyectos citados, junto con el que nos ocupa, son complementarios, ya que vienen a dotar al usuario del conocimiento suficiente para monitorizar la red WiFi de su entorno doméstico.

Por su parte, el presente proyecto, tiene como fin una monitorización proactiva del entorno WiFi del usuario desde tres perspectivas diferentes:

- ¿Alguien está intentado conectarse al interfaz WiFi?
- ¿Cómo se encuentra el interfaz WiFi?

Estas dos cuestiones, abordadas en otros proyectos análogos, no son respondidas a la vez, lo que resta potencial a las herramientas o prototipos antes comentados. Además, la propuesta aquí recogida tiene un enfoque menos académico, pero más práctico y adaptable a las necesidades del usuario final.

3.5 Funcionalidades integradas

3.5.1 Punto de Acceso

En este apartado se recogen los pasos que hay que seguir para tener un punto de acceso WiFi totalmente funcional. Hay que remarcar que la Raspberry Pi, con esta configuración, hará de *bridge*, delegando las funciones de *routing*, *NAT*… en el *router* residencial del usuario. A continuación, se muestra el código empleado, el cual deberá ser escrito tras la apertura de un LXTerminal, el cual es el interfaz de comandos de Raspbian y es representado por el siguiente símbolo:

Figura 10 - LXTerminal

```
#Descargamos actualizaciones
sudo apt-get update
sudo apt-get upgrade

#Instalamos Hostpad: Para crear puntos de acceso y servers de autenticacion.
sudo apt-get install hostapd bridge-utils

#Paramos SW descargado anteriormente
sudo systemctl stop hostapd

#Paramos el DHCP para los interfaces
#Abrimos el fichero dhcpd.conf
sudo nano /etc/dhcpcd.conf
denyinterfaces wlan0
denyinterfaces eth0

#Creamos una interfaz nueva. Será un bridge.
sudo brctl addbr br0
sudo brctl addif br0 eth0

#Retocamos los interfaces
sudo nano /etc/network/interfaces

-- Levantamos le bridge tras un boot
auto br0
-- Asignamos una IP automáticamente
iface br0 inet manual
-- Conectamos el interfaz eth0 y el wlan0
bridge_ports eth0 wlan0

#Retocamos el fichero hostapd. Aquí elegiremos el ssid, contraseña…
sudo nano /etc/hostapd/hostapd.conf
interface=wlan0
bridge=br0
ssid=ProyectoUOC
hw_mode=g
channel=1
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=ProyectoUOC
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP

#Aplicamos los cambios
sudo nano /etc/default/hostapd
-- Quitamos comentario de DAEMON_CONF y añadimos el nuevo directorio creado
DAEMON_CONF="/etc/hostapd/hostapd.conf"


sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd

sudo reboot
```

3.5.2 *Sniffer* WiFi

Para tener un *sniffer* WiFi totalmente funcional, se deben satisfacer dos puntos clave:

- Tener una herramienta que nos permita analizar las tramas WiFi.
- Tener una tarjeta de red compatible con este tipo de análisis, es decir, un interfaz que trabaje en modo monitor.

Para el primer punto comentado, se utilizará la herramienta *TShark*, anteriormente introducida. Por otro lado, y para completar la dupla de requisitos, se usará un *dongle* WiFi, también descrito anteriormente, que permitirá trabajar en modo monitor y capturar las tramas WiFi.

El primer punto sería conectar el *dongle* WiFi a uno de los puertos USB libres. Automáticamente, el LED que posee el *dongle* empezará a brillar, indicativo de que está correctamente alimentado y listo para usarse. Por otro lado, se descargará el *TShark*. A continuación, se muestra el código a emplear tras la apertura de un LXTerminalOk:

```
#Descargamos Tshark
sudo apt-get install tshark
```

Tras tener el *TShark* instalado, se debe chequear en que interfaz de la Raspberry Pi se analizará tráfico. Para ello se tiene el siguiente comando:

```
#Chequeamos en que interfaz debemos capturar tráfico
sudo tshark -D
```

Tras lanzar el comando, aparecerán una serie de interfaces o formas de capturar tráfico. La elegida para este proyecto deberá ser la opción que presente una nomenclatura como la siguiente:

- phyX.mon (siendo X un número)

Si ninguno de los interfaces cumple con la estructura marcada, se recomienda hacer un reboot a la Raspberry Pi de la forma:

```
#Hacemos reboot
sudo reboot
```

Una vez identificado el interfaz, se estará en disposición de capturar tráfico WiFi del tipo:

```
pi@raspberrypi:~ $ sudo tshark -D
Running as user "root" and group "root". This could be dangerous.
1. eth0
2. wlan0
3. br0
4. phy1.mon
5. any
6. lo (Loopback)
7. wlan1
8. bluetooth0
9. nflog
10. nfqueue
11. usbmon1
12. ciscodump (Cisco remote capture)
13. randpkt (Random packet generator)
14. sshdump (SSH remote capture)
15. udpdump (UDP Listener remote capture)
pi@raspberrypi:~ $ sudo tshark -i4
Running as user "root" and group "root". This could be dangerous.
Capturing on 'phy1.mon'
    1 0.000000000 Mitrasta_0e:00:77 → Broadcast    802.11 291 Beacon frame, SN=62, FN=0, Flags=........, BI=100, SSID=MOVISTAR_0075
    2 0.017597364 6a:6a:b0:9b:44:7e → Broadcast    802.11 257 Beacon frame, SN=1802, FN=0, Flags=........, BI=100, SSID=MiFibra-447A-24G
    3 0.026184041 Espressi_d4:42:a3 → Broadcast    802.11 282 Data, SN=3804, FN=0, Flags=.p....F.
    4 0.051623028 Netgear_48:37:8e (c4:04:15:48:37:8e) (TA) → AskeyCom_4c:ab:25 (e8:d1:1b:4c:ab:25) (RA) 802.11 34 Request-to-send, Flags=........
    5 0.051937143 Netgear_48:37:8e (c4:04:15:48:37:8e) (TA) → AskeyCom_4c:ab:25 (e8:d1:1b:4c:ab:25) (RA) 802.11 34 Request-to-send, Flags=........
    6 0.052246258 Netgear_48:37:8e (c4:04:15:48:37:8e) (TA) → AskeyCom_4c:ab:25 (e8:d1:1b:4c:ab:25) (RA) 802.11 34 Request-to-send, Flags=........
    7 0.054284750 Netgear_48:37:8e (c4:04:15:48:37:8e) (TA) → AskeyCom_4c:ab:25 (e8:d1:1b:4c:ab:25) (RA) 802.11 34 Request-to-send, Flags=........
    8 0.064767575 f8:8b:86:ad:a1:39 → Broadcast    802.11 287 Beacon frame, SN=350, FN=0, Flags=........, BI=100, SSID=MOVISTAR_A130
    9 0.066932109 Tp-LinkT_a3:e3:be → Broadcast    802.11 264 Beacon frame, SN=2512, FN=0, Flags=........, BI=100, SSID=MOVISTAR_9049
```

Figura 11 - Interfaz sniffer

3.5.2.1 Ocupación de canal

En las comunicaciones inalámbricas, el uso del espectro juega un papel fundamental. Quién y cómo está utilizando cierto ancho de banda marcará la calidad de la comunicación.

En las comunicaciones tratadas en este proyecto, WiFi sobre la banda de 2.4GHz, se debe tener en cuenta dos puntos:

- La banda de frecuencia que utiliza la tecnología WiFi es no licenciada. Esto significa que cualquiera, con cualquier tecnología (bluetooth, por ejemplo) puede radiar en la misma porción de espectro que nuestro punto de acceso WiFi.
- Los canales WiFi en 2.4GHz sobre el estándar 802.11n se solapan.

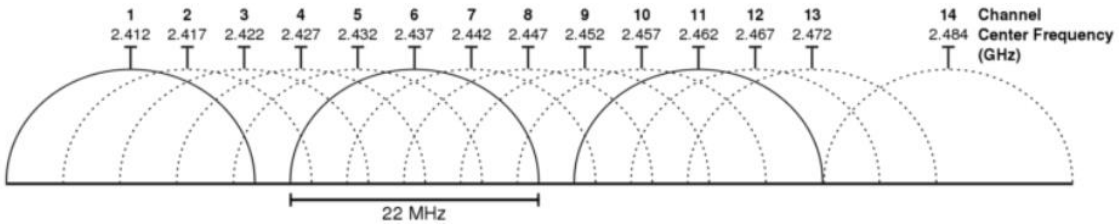


Figura 12 - Canales WiFi 2.4GHz https://www.adslzone.net/2018/05/21/canales-1-6-11-wifi/

Por tanto, y de forma resumida, existen tres grandes tipos de interferencia WiFi que tener en cuenta, las cuales marcarán la calidad de la transmisión y, por tanto, la experiencia de usuario resultante:

- Interferencia por Canal Solapado: Es aquella interferencia producida por las estaciones que radian en el mismo canal que el punto de acceso

WiFi. Dicho punto de acceso WiFi, gracias a CSMA/CA, podrá coordinarse con otros puntos de acceso WiFi que radien en su mismo canal. De esta forma, las estaciones se reparten el tiempo de aire de la forma más equilibrada posible.

- Interferencia por Canal Adyacente: Esta interferencia es producida por otras estaciones que radian en un canal próximo al del punto de acceso de usuario. Esta interferencia es muy dañina y puede provocar una rápida degradación de la señal.

- Interferencia externa: Serían aquellas interferencias No WiFi que se detectan en el ancho de banda de trabajo del punto de acceso. Estas interferencias son también muy dañinas.

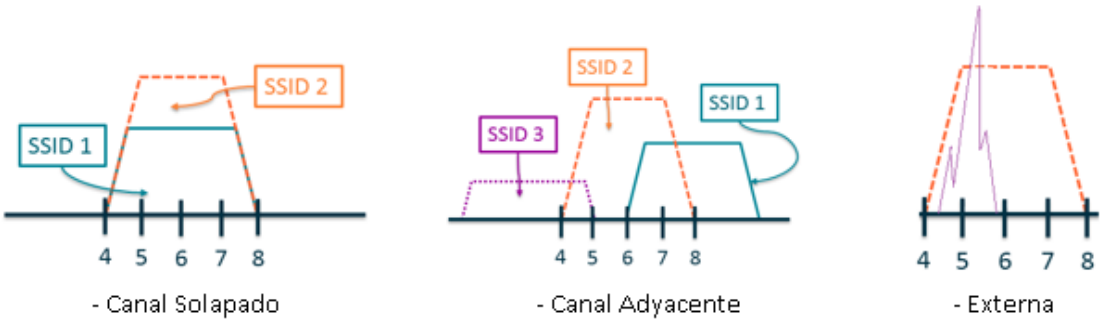


Figura 13 - Interferencias WiFi

A la hora de tener en cuenta la ocupación de canal, se tendrán en cuenta las mediciones que periódicamente reportan los puntos de acceso WiFi vecinos. Para ello se utilizará un parámetro clave:

- **wlan.qbss.cu**

Este parámetro es uno entre los múltiples que el estándar 802.11 define para marcar QoS (*Quality of Service*). En concreto, este parámetro es utilizado por los puntos de acceso WiFi para informar a los terminales de que utilización del canal (*channel utilization*) están midiendo. Este dato toma valores de 0 a 255, calculándose la utilización del canal de la siguiente forma:

Utilización de canal = (wlan.qbss.cu * 100) / 255

Teniendo lo anteriormente expuesto en cuenta, se podrá filtrar las tramas gracias a *TShark* de la siguiente forma:

```
#Ejemplo de ocupacion de canal
sudo tshark -i4 -Y"wlan_radio.channel == 1 or wlan_radio.channel == 2 or wlan_radio.channel == 3 and wlan.qbss.cu > 150" -T fields -e wlan.ta -e wlan.qbss.cu -e wlan_radio.channel
```

Cabe destacar que se analizará la ocupación del canal de trabajo del punto de acceso de usuario (1 en el caso del proyecto) y en los adyacentes (2 y 3, en este proyecto).
La información recogida será plasmada en un fichero de texto, donde se anotará que punto de acceso ha tomado la muestra y que valor ha reportado.

3.5.2.2 *Handshake*

A la hora de tener en cuenta los clientes confiables y los equipos que, insatisfactoriamente, intentan conectarse al WiFi de usuario, se inspeccionará el denominado *4-Way Handshake*.

Esta forma de autenticar a los usuarios es propia del protocolo de cifrado WPA2, el escogido para este proyecto, y más seguro hasta la fecha. También hay que remarcar que este proyecto se apoya en el protocolo de compartición de claves PSK (*Pre Shared Key*).
El *4-Way Handshake* se diseñó de tal forma que el punto de acceso WiFi (denominado *authenticator* en el proceso) y el cliente (*supplicant* en este caso) pudieran autenticarse mutuamente sin poner en riesgo la PSK/PMK (en el caso de estudio, el primer término, PSK). Este punto es crítico ya que la PSK/PMK se genera, junto a otros campos, con la *passphrase*. La *passphrase* sería la contraseña que se debe introducir cuando un cliente quiere conectarse a una red WiFi.
Teniendo lo anteriormente comentado en cuenta, los pasos para la autenticación serían:

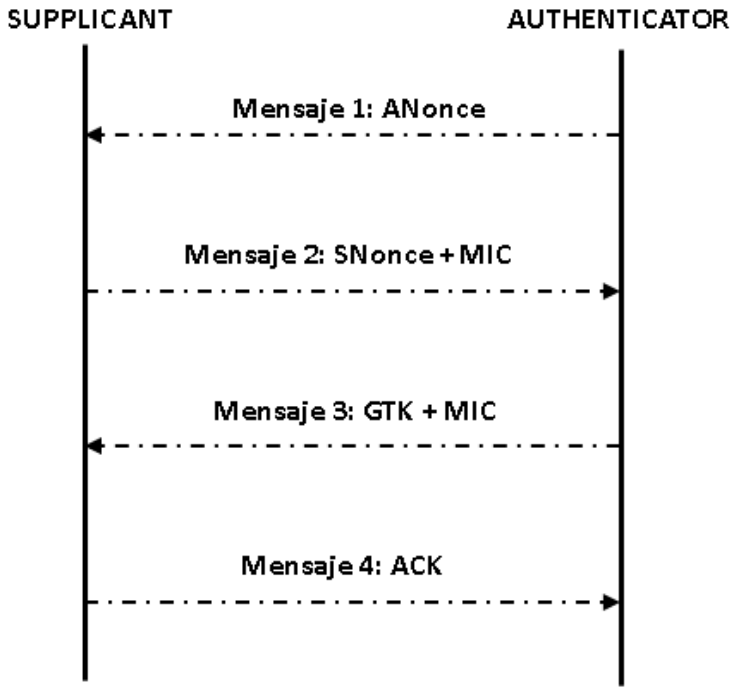


Figura 14- 4 Way Handshake

Particularizando para el presente Trabajo Final de Master, para la detección de estaciones que quieran conectarse al punto de acceso WiFi de usuario con una clave errónea, se tomará el Mensaje 2. En este paso, la estación conforma un mensaje con los campos:

- PSK
- ANonce
- SNonce
- MAC Origen
- MAC destino

Este mensaje no podrá ser autenticado por el punto de acceso WiFi de usuario ya que estará mal conformado. Esto es debido a que el campo PSK se calcula, de forma obligatoria, con el *passphrase* (la contraseña del SSID WiFi), el cual no es conocido por la estación (*supplicant*).

De cara a poder analizar las sesiones de los equipos que se conectan al punto de acceso WiFi, se tendrá la siguiente línea de comando de *TShark*:

```
#Capturar Handshake
sudo tshark -i4 -Y"wlan_rsna_eapol.keydes.msgnr and wlan.addr == b8:27:eb:09:eb:11" -T fields -e wlan.ta -e wlan.ra -e wlan_rsna_eapol.keydes.msgnr > /home/pi/paw_services/handshake_raw.paw
```

Los parámetros clave serán:

- wlan_rsna_eapol.keydes.msgnr: Este parámetro, perteneciente al *4-Way Handshake* (eapol), marca en que mensaje, del 1 al 4, se encuentra la autenticación.
- wlan.addr: Este punto filtrará solo tráfico que sea iniciado o dirigido hacia nuestra Raspberry Pi.

La señalización capturada será guardada en un fichero de texto. En dicho fichero, línea por línea, se especificará el transmisor, receptor y paso del *handshake* en el que se produjo la comunicación.

3.5.3 Envío de mails

Un punto crítico del proyecto es el aviso al usuario en caso de que se detectara alguna anomalía, ya sea por la presencia de equipos desconocidos en el interfaz WiFi, por la detección de un posible ataque WiFi o una sobreocupación del canal. Para hacer esto posible, se configurará la herramienta SSMTP. Primeramente, se deberán descargar los paquetes SW necesarios para su correcto funcionamiento:

```
#Descargamos SW
sudo apt-get install ssmtp mailutils mpack
```

Tras esto, se configurará la herramienta:

```
#Configuramos el SSMTP
sudo nano /etc/ssmtp/ssmtp.conf

--Añadimos la linea
AuthUser=sergio.proyectouoc@gmail.com
AuthPass=xxxxxxxxx
--Descomentamos la linea
FromLineOverride=YES
--Retocamos la linea
mailhub=smtp.gmail.com:587
--Añadimos la linea
UseSTARTTLS=YES
```

Una vez hecho, se podrán enviar *mails* con el siguiente formato:

```
#Ejemplo de envío de mail
sudo echo “Cuerpo del Mail” | mail -s “Asunto del Mail” sergio.proyecto@gmail.com
```

Se tendrá, como ejemplo, el siguiente *mail*:

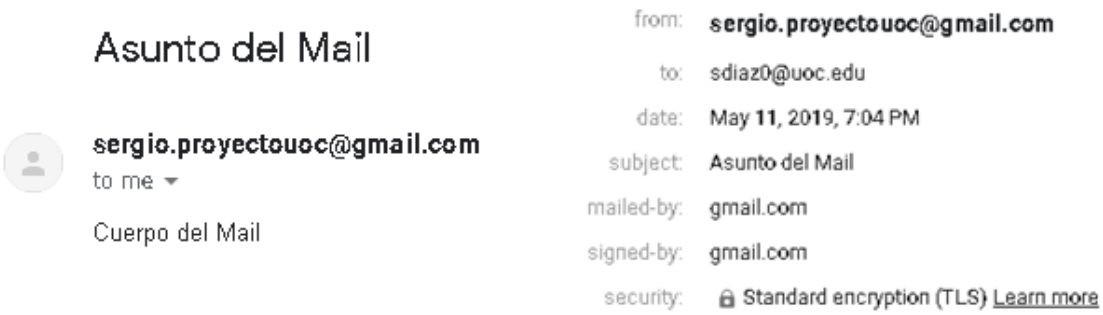


Figura 15 - Mail

3.5.4 Automatización de Tareas

Uno de los puntos centrales del proyecto será la automatización de tareas. Para poder lanzar los debidos *scripts* en automático, se debe tener un programa que pueda ejecutar las ordenes, en forma y tiempos, de modo que el usuario final tenga poca interacción con el punto de acceso WiFi. Para este fin, se ha escogido la herramienta Crontab. Para acceder a él se debe teclear lo siguiente en un LXTerminalOK:

```
#Demonio crontab
sudo crontab -e
```

Por pantalla se pedirá una forma de editar el demonio Crontab, por simplicidad, se recomienda la segunda alternativa: /bin/nano. Una vez dentro del demonio, se configurarán las órdenes a lanzar de la forma:

Minuto	Hora	Día	Mes	Día de la semana	Usuario	Comando a Ejecutar
Rango: 0-59	Rango: 0-23	Rango: 1-31	Rango: Enero: 1 .... Diciembre: 12	Rango: Domingo: 0 .... Domingo: 7		

Figura 16 - Configuración Crontab

Crontab, por defecto, envía un *mail* de confirmación de realización, correcta o incorrectamente, de la tarea programada. En caso de que se requiera recibir el *mail*, como en el caso de este proyecto, para chequear que el sistema funciona correctamente, se añadirá la siguiente linea de código a la configuración del crontab:

```
#Reporte a un correo en concreto
MAILTO=”sergio.proyectouoc@gmail.com”
```

En caso de que no se quiera recibir este tipo de *mail*s, se podrá añadir el siguiente código:

```
#Sin reporte
MAILTO=””
```

3.6 Programa Final: Punto de acceso WiFi con servicio de Alerta por Mail

En este punto se detallan los dos procesos creados (uno para controlar los accesos al WiFi y otro para analizar la ocupación del canal). Los *scripts* y archivos generados por ambos procesos se albergan en una carpeta llamada paw_services, que puede ser encontrada en la siguiente ruta: /home/pi/paw_services. Para crear la comentada carpeta, se puede utilizar el siguiente comando:

```
#Partiendo del directorio pi
mkdir paw_services
```

3.6.1 Ocupación de Canal

Para automatizar la medición y el reporte, si fuera pertinente, de una excesiva ocupación de canal, se ha creado un *script* en bash llamado ocupación_canal_v2.sh.

En este *script*, básicamente, se recogen las ocupaciones de canal mayores de una ocupación de canal marcada en 217 (un 85% de ocupación de canal) en un periodo de 23 horas y 55 minutos. Todas las tramas que cumplan las condiciones marcadas serán volcadas a un fichero de texto (con extensión .paw, para saber que es un fichero del proyecto de punto acceso WiFi). El fichero recogerá que medida ha sido reportada y que punto de acceso WiFi lo hizo. Si el fichero no está en blanco (es decir, su tamaño es distinto de 0) se mandará un *mail* de alerta. El fichero creado, por tanto, quedará guardado durante 23 horas y 55 minutos, tiempo suficiente para que el usuario final reconsidere si cambiar o no el canal en función de las muestras recogidas que superan el umbral fijado. Se ha dejado una salvaguarda de 5 minutos para que la Raspberry Pi haga el tratamiento del fichero y el envío del *mail* si es pertinente, antes del siguiente periodo de *sniffing*.
La automatización de este servicio correrá a cargo de Crontab. La configuración del fichero bash será la siguiente:

```
#Partiendo del directorio pi
#!/bin/bash
sudo tshark -i4 -Y"wlan_radio.channel == 1 or wlan_radio.channel == 2 or wlan_radio.channel == 3 and wlan.qbss.cu >217" -a duration:86100 -T fields -e wlan.ta -e wlan.qbss.cu > /home/pi/paw_services/ocupacion_canal_raw.paw
declare -i n
declare -i zero=0
n=$(cat ocupacion_canal_raw.paw|wc -w)
echo $n
test $n -gt $zero && echo "Alta ocupacion de canal. Se recomienda cambiar de canal." | mail -s "Canal saturado" sergio.proyectouoc@gmail.com || echo "Todo Ok"
```

3.6.2 Clientes Confiables y Posible Intrusión en el WiFi

En este punto se dispondrá de dos *scripts* que se ocuparán de:

- Capturar el *handshake*.
- Discernir si el *handshake* ha sido correcto o incorrecto (posible ataque).
- Controlar si los dispositivos que se han conectado al WiFi son confiables o no.
- Enviar los mensajes pertinentes.

El primer *script* que se tiene está especificado en lenguaje bash, llamado handshake.sh. Este *script* también estará albergado en la carpeta paw_services. El código en sí será el siguiente:

```
#!/bin/bash

tshark -i4 -Y"wlan_rsna_eapol.keydes.msgnr and wlan.addr == b8:27:eb:09:eb:11" -a duration:85900 -T fields -e wlan.ta -e wlan.ra -e wlan_rsna_eapol.keydes.msgnr > /home/pi/paw_services/handshake_raw.paw
python /home/pi/paw_services/handshake_service.py
#rm /home/pi/paw_services/handshake_raw.paw
```

Este sencillo *script* lanza dos líneas. La primera correspondiente al tráfico generado por aquellos clientes WiFi que quieren conectarse al punto de acceso WiFi de usuario, y una segunda correspondiente al tratamiento de los datos.

El segundo *script*, en lenguaje de programación Python, se ocupa de tratar el fichero y mandar los correspondientes mensajes. En la lectura del fichero hay tres puntos clave que discernir:

- Sesiones: Por cada vez que un usuario diferente se conecte al WiFi, ya sea correcta o incorrectamente, se sumará una sesión.

- Clientes: Hay que separar cuidadosamente las sesiones de cada usuario. Solo así se podrá saber cómo ha finalizado el proceso de *handshake*.

- Retransmisiones: Enel proceso de *handshake* puede haber retransmisiones. Estas retransmisiones deben ser computadas en la misma sesión de usuario. Si no fuera así, cada retransmisión se interpretaría como un fallo de conexión, lo que sería erróneo.

Debido a lo anteriormente comentado, para el correcto tratamiento del fichero, se ha decidido realizar un diccionario. Este diccionario albergará:

- Keys: Identificará unívocamente al usuario con su sesión. De esta forma se podrá conocer que MAC y cuantas veces se ha conectado al Punto de Acceso WiFi.

- Value: Recogerá el estado de cada sesión. Es decir, guardará la información de cada paso del Handshake.

El código utilizado ha sido el siguiente. Para facilidad de seguimiento y explicación del comentado código, se ha divido este en tres fases diferenciadas. Cada una de sus partes se explica a continuación:

- Fase 1: En primera instancia, se declaran las variables a utilizar. A destacar la línea de *send_OK* y *send_KO*, las cuales se utilizarán para mandar los *mails* de cliente desconocido conectado y de posible ataque WiFi. También remarcar la variable *_dict*, la cual se utilizará para crear un diccionario de tipo (*key,value*) y la *mac_whitelist*, la cual marcará los clientes confiables.
  En la primera función declarada, *prepare_debugging*, se recorrerá el fichero leído hasta encontrar un 1 en la posición que marca el paso del *handshake*. Con esto se busca el primer 4-Way Handshake iniciado correctamente. De esta forma se descartará aquellos *handshake* ya iniciados en el momento en el que el *sniffer* se conectó y que, a efectos prácticos, serán ignorados.

```
#!/usr/bin/env python

import subprocess
#Envío de mails
send_OK_mail_command = '"echo "Se ha conectado un cliente desconocido." | mail -s "Cliente desconocido" sergio.proyectouoc@gmail.com"'
send_KO_mail_command = '"echo "Se ha intentado conectar un cliente." | mail -s "Posible ataque por fuerza bruta" sergio.proyectouoc@gmail.com"'
start_debuging = False
_dict = {}
handshake_path='/home/pi/paw_services/handshake_raw.paw'
mac_whitelist = ['10:44:00:19:0a:b2']
def prepare_debugging(file):
        prepared = False
        while not prepared:
                pos = file.tell()
                current_fake_line = file.readline()
                prepared = int([word for word in current_fake_line.split('\t')][2]) == 1
                if prepared:
                        file.seek(pos)
```

Fase 2: Una vez se encuentre el primer paso 1 del *handshake* se procederá a leer el fichero. Primeramente, se separarán las líneas del fichero por tabulaciones. De esta forma se tendrán los campos separados como: [Transmisor, Receptor, Paso del *handshake*]. Una vez hecho esto, se buscará en que paso del *handshake* se encuentra la comunicación. Sabiendo esto, se buscará la posición del equipo de cliente gracias a la operación módulo: si la comunicación se encuentra en el paso 1 o 3, el equipo de cliente será el receptor del mensaje y, por tanto, ocupará la posición 2 de la línea leída (posición 1 del array que representa a la línea). Por el contrario, si la comunicación se encuentra en la fase 2 o 4, el equipo del cliente vendrá recogido en la posición 1 (posición 0 del array que representa a la línea leída).

Una vez hecho esto, y según se vaya leyendo el fichero, se deberá discernir si la línea leída es el siguiente paso de la comunicación de un cliente, una retransmisión del cliente o un nuevo equipo de cliente. Para ello se valida si el paso del *handshake* actual sea igual o menor al anterior (potencial retransmisión o nuevo cliente) y si el equipo del cliente es igual al de la anterior línea leída (potencial nuevo cliente). De esta forma las sesiones de cada usuario se tendrán ordenadas (las retransmisiones, por tanto, se tendrán en cuenta dentro de la misma sesión de usuario, contabilizándose además únicamente cada tipo de paso del *handshake* -paso 1, 2, 3 o 4- diferente y no el número de retransmisiones de cada paso). Cabe destacar que cada sesión se formará con una *key* del tipo: MACCliente__sesión.

Una vez realizado todo lo anteriormente expuesto, se irá completando el diccionario, añadiendo cada línea a la sesión del usuario.

```
def do_scanner():
        with open(handshake_path,'r') as file:
                prepare_debugging(file)
                current_step = 0
                current_session = 0
                last_device = 0
                for line_raw in file:
                        print('LINEA --> ',line_raw)
                        line = [word for word in line_raw.split('\t')] #Separamos el fichero
                        current_line_step = int(line[2]) #Paso del handshake
                        current_line_device = line[1] if int(line[2]) % 2 == 1 else line[0] #Posicion equipo de cliente
                        allow_continue = True
                        if last_device == 0:
                                last_device = current_line_device
                    #Se chequea si se ha cambiado de sesión o es un retransmision
                        if ((current_line_step <= current_step) or (current_line_device != last_device)):
                                if current_line_device != last_device:
                                        if current_step > 0:
                                                current_session += 1
                                else:
                                        allow_continue = False
                        if allow_continue:
                                last_device = current_line_device
                                current_step = current_line_step
                                key = current_line_device + '__' + str(current_session)
                                print(key)
                                value = [] #Guardaremos cada paso del handshake
                                print(line)
                                if key not in _dict:
                                        value = [line]
                                else:
                                        value = _dict[key] #Se guarda primero en value el diccionario
                                        value += [line] #Añades a value la nueva linea
                                _dict[key] = value #Guardas en value la nueva linea y lo que el dicc guardaba
                                if current_step is 4:
                                        current_line_step = 0
                                        current_step = 0
                                        current_session += 1
                print(_dict)
```

Fase 3: Una vez se haya tratado todo el fichero, se tomarán las decisiones según cada sesión. Si para una misma sesión tenemos más de dos pasos del *handshake* registrados, se evaluará como una asociación correcta. De forma adicional, se analizará si dicha asociación correcta ha sido realizada por un usuario confiable (perteneciente a la *White_list* de la Fase 1) o no. En caso de que no sea confiable, se emitirá un correo de cliente desconocido. Por otro lado, si para una misma sesión tenemos dos o menos pasos del *handshake* registrados, se evaluará como una asociación incorrecta y se emitirá un correo de posible ataque WiFi.

```
def do_mailling():
        for key,value in _dict.items():
                if len(value) <= 2:
                        process                                                 =
subprocess.Popen(send_KO_mail_command,stderr=subprocess.PIPE,stdout=subprocess.PIPE,shell=True).
communicate()
                elif len(value) > 2 and key.split('__')[0] not in mac_whitelist: #Comparamos contra nuestra
whitelist.
                        process                                                 =
subprocess.Popen(send_OK_mail_command,stderr=subprocess.PIPE,stdout=subprocess.PIPE,shell=True).
communicate()

do_scanner()
do_mailling()
```

De forma gráfica, se tendrá una distribución de las sesiones de los usuarios como la siguiente:

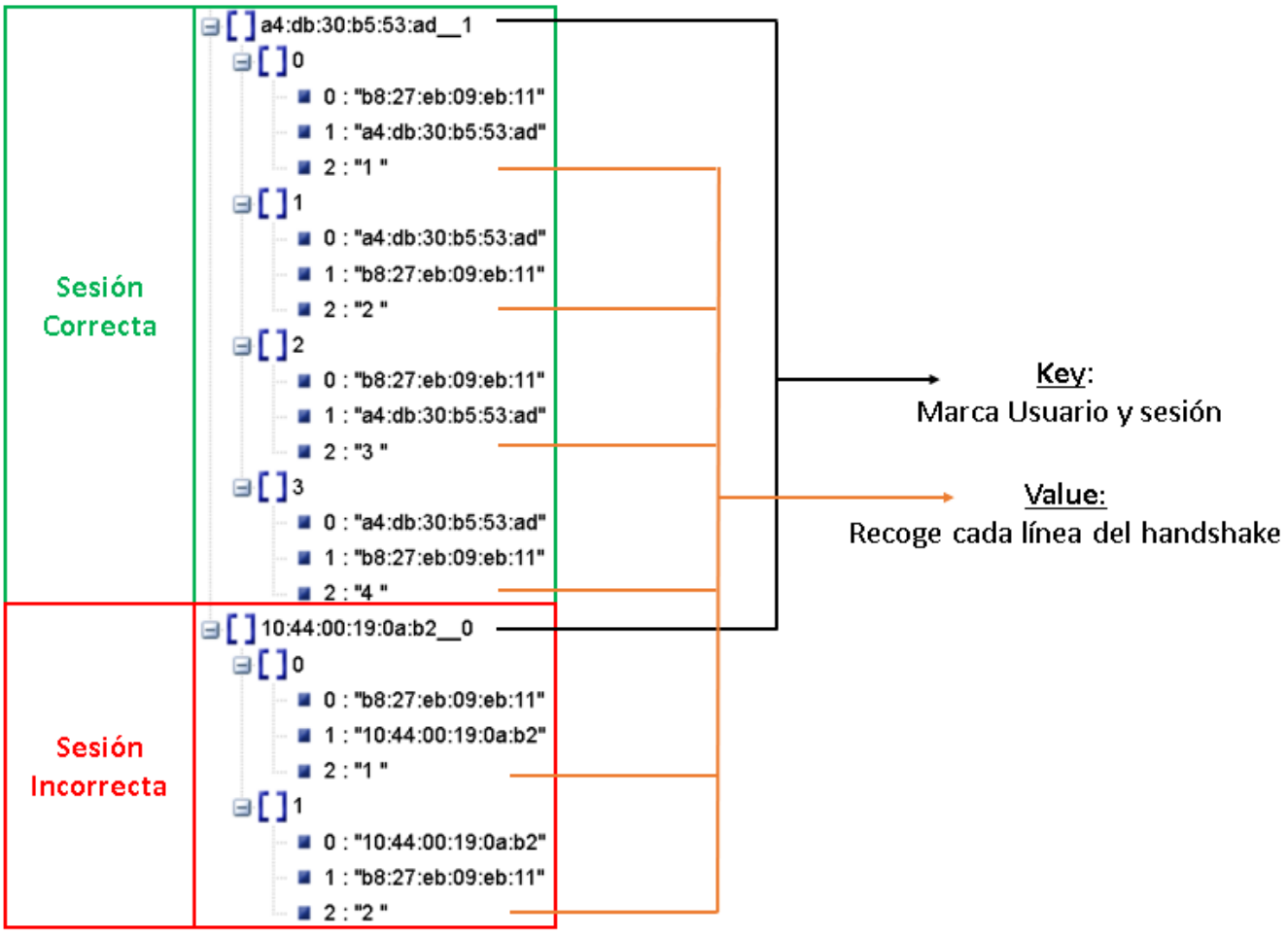


Figura 17 - Mapa de sesiones

3.6.3 Automatización: Crontab

Se abrirá Crontab, donde se añadirán las siguientes líneas:

```
MAILTO=”sergio.proyectouoc@gmai.com”
00 10 * * * /home/pi/paw_serices/ocupacion_canal_v2.sh
00 10 * * * /home/pi/paw_services/handshake.sh
```

Cabe destacar la notación usada: la utilización de asteriscos (*) implica que el comando será lanzado sin importar el día, mes o día de la semana. De esta forma se recibirá un reporte diario.
Además, se ha añadido un *mail* donde se recibirá la finalización, correcta o incorrecta, de cada tarea.

4. Batería de Pruebas

En este apartado se definirán una serie de pruebas, validando que cada una de las herramientas implementadas, *sniffer* WiFi (para ocupación de canal y *handshake*), envío de *mails* y automatización de tareas, funcionan correctamente. Además, se probará también el servicio en conjunto, donde cada una de las herramientas implementadas deben trabajar de forma concurrente y ordenada.

4.1 Definición de batería de pruebas

A. **Cliente permitido conectado**: Se marcará la MAC 10:44:00:19:0a:b2 como cliente confiable. En este punto se validará que no se produce ningún mensaje ya que el cliente se valida correctamente y está dentro de la lista de clientes confiables.

B. **Ciente desconocido**: Se iniciará un proceso exitoso de *4-Way Handshake* con un cliente desconocido (fuera de la lista de clientes confiables). En este caso, se validará que el mensaje correspondiente a un cliente desconocido es lanzado.

C. **Posible Ataque WiFi:** Se iniciará un proceso erróneo (mal *passphrase*) de *4-Way Handshake*. En este punto, se validará que los reintentos erróneos del cliente se computan en una única sesión y se envía un único *mail*.

D. **Cliente desconocido + Posible Ataque WiFi**: Se lanzarán dos procesos de *4-Way Handshake* independientes de dos terminales WiFi. Uno de ellos será de un cliente desconocido, pero que se valida correctamente contra el punto de acceso WiFi de usuario. El segundo terminal, por el contrario, fallará en la autenticación. En este escenario, dos *mail*s distintos deberán ser lanzados: uno advirtiendo de un cliente WiFi desconocido y otro sobre un posible ataque por fuerza bruta.

E. **Dos clientes permitidos conectados**: En este caso, se conectarán dos terminales distintos. Ambos terminales tendrán la categoría de confiables, por lo que ningún *mail* deberá ser lanzado.

F. **Captura de Channel Utilization:** Se capturará la utilización del canal con un umbral muy bajo, de forma que se capturarán paquetes y, por tanto, se enviará el pertinente *mail*.

G. **Concurrencia de procesos Crontab**: Se comprobará que los dos crontab configurados pueden confluir en el tiempo sin ningún tipo de

problema. Para ello se analizará durante 60 segundos el canal WiFi. Para recoger eventos, se marcará la utilización del canal como 5, valor muy bajo, que desembocará en el envío de un *mail*. Además, en los 60 segundos de la muestra, se conectará un usuario correctamente con un cliente WiFi que este fuera de la lista de clientes confiables. Este punto hará que se envíe automáticamente un *mail* por cliente desconocido.

H. Estabilidad de los servicios (I): Se programará un crontab de forma que los procesos subyacentes de análisis de tráfico sean concurrentes. Además, los *sniffer* quedarán programados de tal forma que tomarán muestras de las tramas WiFi durante 6 horas.

I. Estabilidad de los servicios (II): La prueba será similar a la comentada en el punto H, con la única diferencia de que se programarán los *sniffer* para que analicen durante 23 horas y 55 minutos.

4.2 Bateria de pruebas y refinamiento

En este apartado se revisarán, uno a uno, los casos prácticos planteados. Para cada caso, se especificará la información relevante para la realización de la prueba. De forma adicional, se añadirán observaciones para cada una de ellas. Como apunte, los archivos generados en crudo se podrán consultar en el Anexo I. Solo se recogerán, a modo de muestra, los archivos generados para la prueba A. Cliente permitido conectado y la prueba F. Captura de Channel Utilization.

A. Cliente permitido conectado:

- Esquema de la prueba:

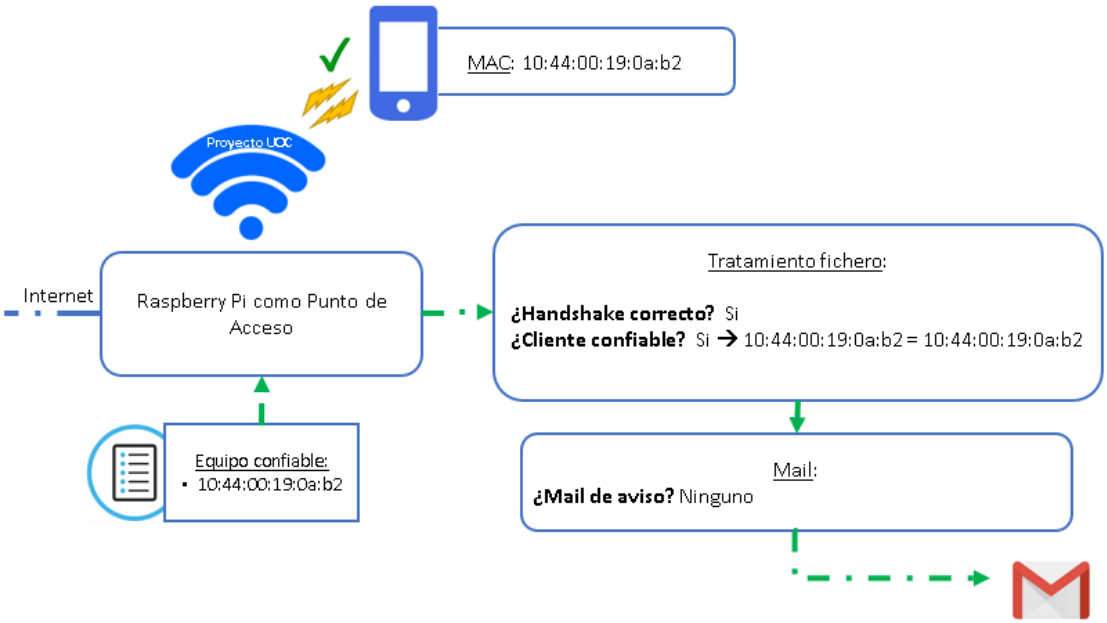


Figura 18 - Prueba A Cliente Confiable

- Archivo generado:

```
b8:27:eb:09:eb:11  10:44:00:19:0a:b2  1
b8:27:eb:09:eb:11  10:44:00:19:0a:b2  1
10:44:00:19:0a:b2  b8:27:eb:09:eb:11  2
10:44:00:19:0a:b2  b8:27:eb:09:eb:11  2
b8:27:eb:09:eb:11  10:44:00:19:0a:b2  3
10:44:00:19:0a:b2  b8:27:eb:09:eb:11  4
```

- Observaciones: En este caso, la validación es correcta y el cliente está marcado como confiable, por lo que ningún *mail* será enviado al usuario final. En el archivo se puede observa que hay dos retransmisiones, una del paquete 1 y otra del paquete 2. El *script*

hace que esas retransmisiones no las considere como una mala autenticación ni como parte de la sesión de otro usuario distinto.

B. Ciente desconocido:

- Esquema de la prueba:

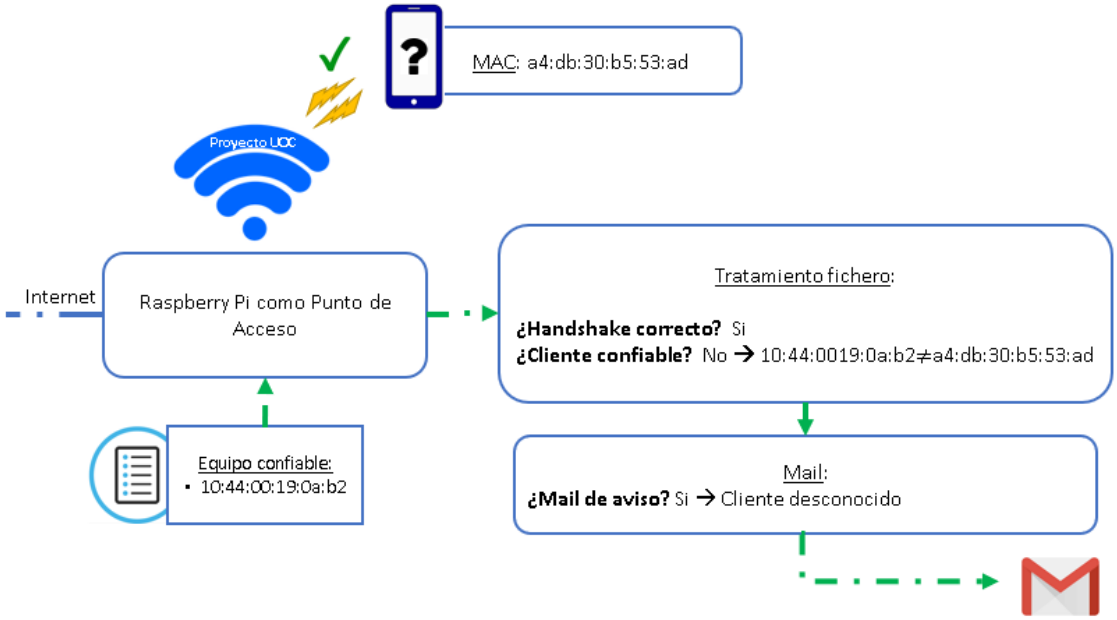


Figura 19 - Prueba B Cliente Desconocido

- *Mail* generado: Cliente desconocido.

Figura 20 - Prueba B Mail Cliente Desconocido

- Archivo generado: En Anexo I.

- Observaciones: Se pueden ver que se cumplen todos los pasos del *handshake* de forma limpia. El cliente, al no estar en la lista de clientes confiables, es tomado como un cliente desconocido, lo que hace que un *mail* se genere en automático.

C. Posible Ataque WiFi:

- Esquema de la prueba:

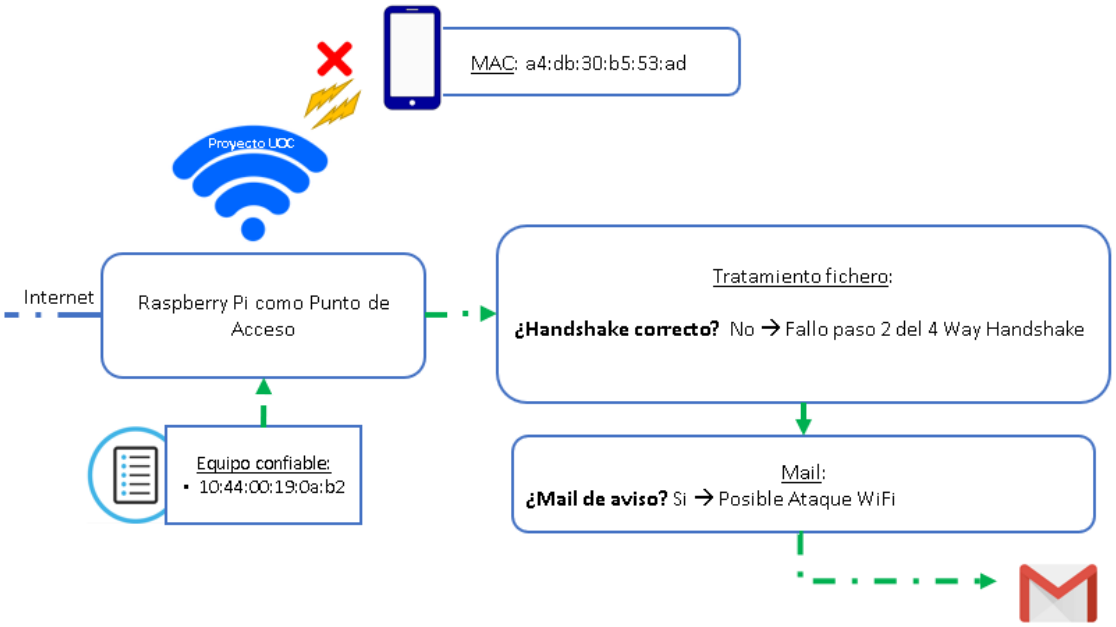


Figura 21 - Prueba C Posible Ataque WiFi

- *Mail* generado: Posible ataque WiFi.

Figura 22 - Prueba B *Mail* Posible Ataque por Fuerza Bruta

- Archivo generado: En Anexo I.

- Observaciones: En el archivo generado se puede observar que el cliente WiFi intenta autenticarse varias veces, tantas como la política de reintentos que el fabricante haya dispuesto. Como se describe en puntos anteriores, el *4-Way Handshake* falla en el segundo mensaje. En dicho paso, el punto de acceso WiFi no puede autenticar al cliente ya que la PSK construida por este es errónea. Cabe destacar que únicamente se computa una única sesión que derivará en un único *mail* de aviso.

D. Cliente desconocido + Posible Ataque WiFi:

- Esquema de la prueba:

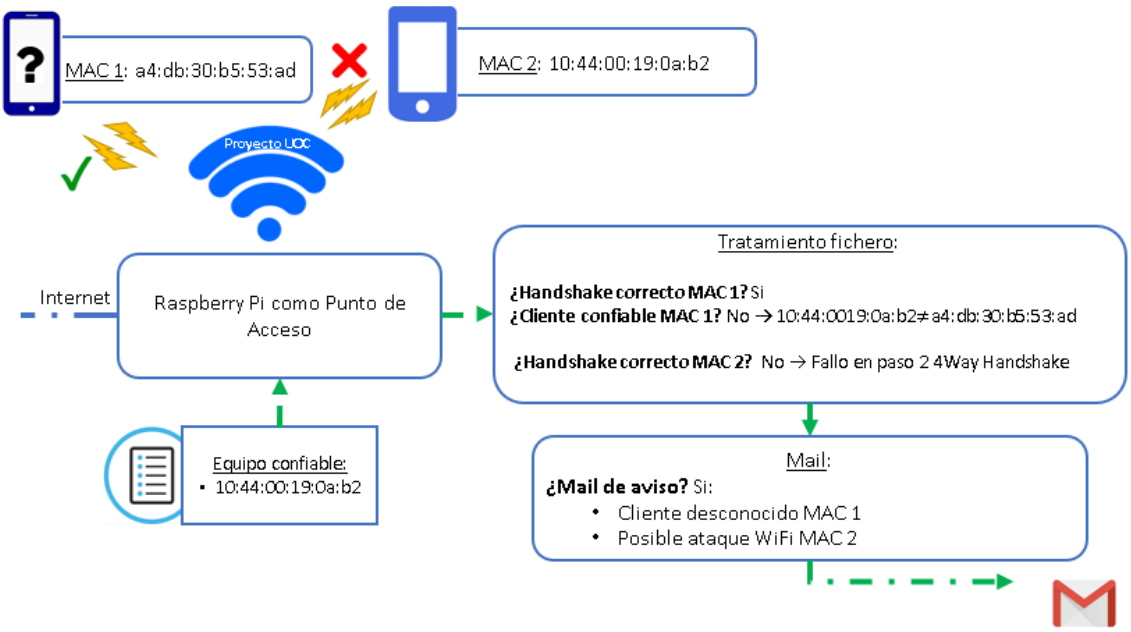


Figura 23 - Prueba D Cliente Desconocido + Posible Ataque WiFi

- *Mail* generado: Cliente desconocido y posible ataque.

Figura 24 - Prueba D Mail Cliente Desconocido + Posible Ataque WiFi

- Archivo generado: En Anexo I.

- Observaciones: Enel archivo generado se puede ver como el terminal con MAC a4:db:30:b5:53:ad, se conecta sin ningún tipo de problema, cumpliendo los 4 pasos del *handshake*. Por el contrario, el terminal con MAC 10:44:00:19:0a:b2, entra en una política de reintentos donde no se progresa más allá del paso 2, como era de esperar.

E. Dos clientes permitidos conectados:

- Esquema de la prueba:

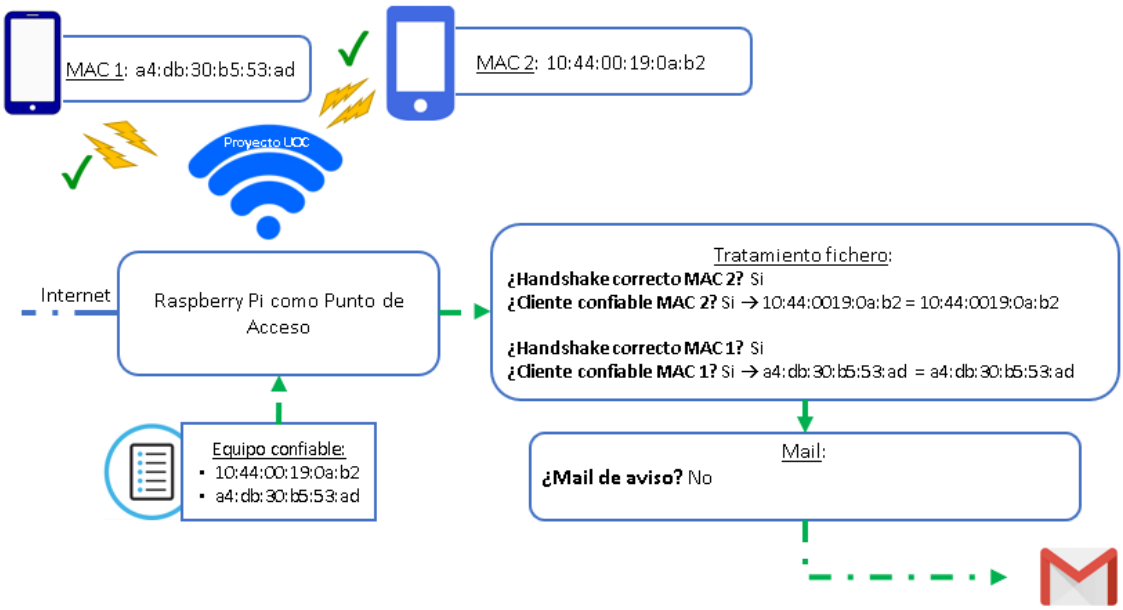


Figura 25 - Prueba E MACs Confiables

- Mac conectada: 10:44:00:19:0a:b2 y a4:db:30:b5:53:ad

- Código cambiado:

```
mac_whitelist = ['10:44:00:19:0a:b2','a4:db:30:b5:53:ad']
```

- *Mail* generado: Posible ataque WiFi.

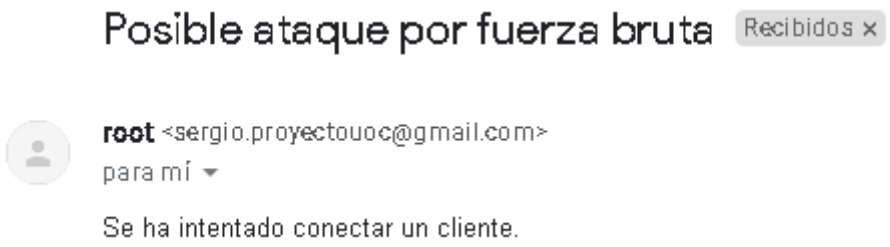


Figura 26 - Prueba E Mail MACs Confiables

- Archivo generado: En Anexo I.

- Observaciones: Como se puede observar en el archivo generado, hay una primera comunicación entre la punto de acceso WiFi del usuario y el cliente 10:44:00:19:0a:b2. Esta comunicación termina de forma correcta pero no es capturada por el *sniffer*. Esto puede deberse a la pérdida o descarte de paquetes. Este comportamiento erróneo puede ser debido a alguna deficiencia

del *dongle* utilizado o por el *TShark*. El hecho de no haber capturado los paquetes suficientes dentro de la comunicación hace que nuestro sistema lo interprete como un posible ataque WiFi.

Por otro lado, se puede observar que hay dos clientes WiFis conectados y que completan todos los pasos de autenticación. Ningún *mail* de cliente desconocido es generado, por lo que la parametrización de los clientes confiables es correcta.

F. Captura de *Channel Utilization*:

- Esquema de la prueba:

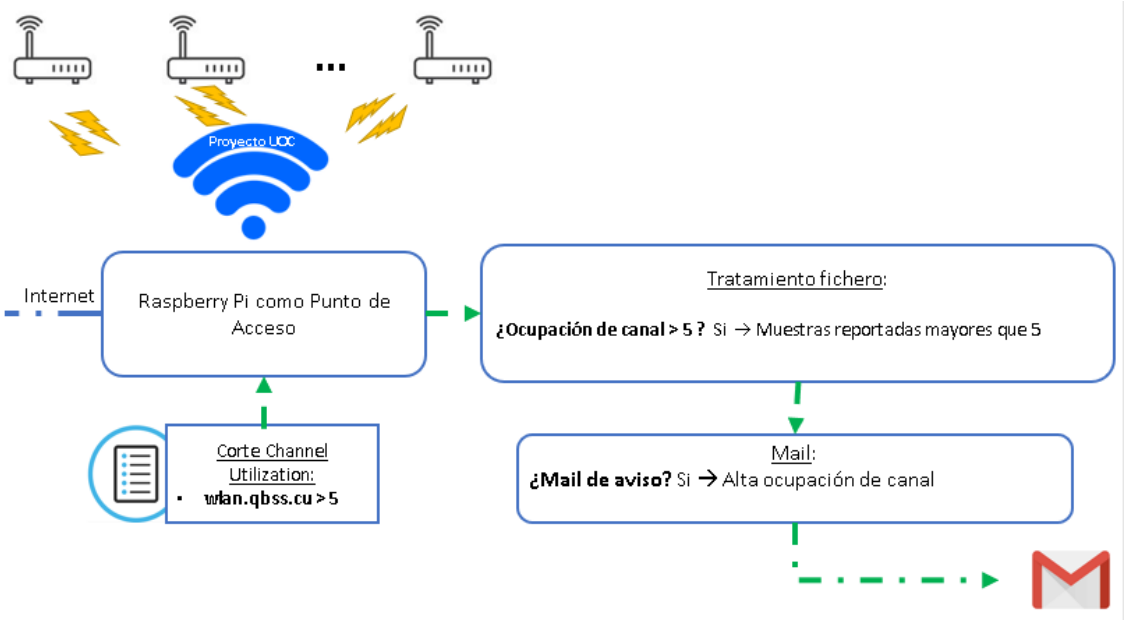


Figura 27 - Prueba F Channel Utilization

- *Mail* generado: Ocupación de canal.

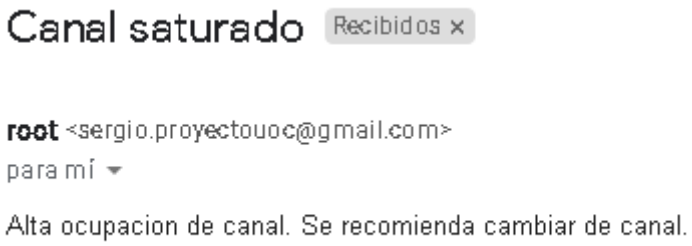


Figura 28 - Prueba F Mail Channel Utilization

- Archivo generado:

```
98:97:d1:a3:90:4a  55      1
d4:7b:b0:d2:42:54  43      1
98:97:d1:a3:90:4a  55      1
```

98:97:d1:a3:90:4a	55	1
98:97:d1:a3:90:4a	55	1
98:97:d1:a3:90:4a	55	1
98:97:d1:a3:90:4a	56	1
98:97:d1:a3:90:4a	56	1
98:97:d1:a3:90:4a	56	1
98:97:d1:a3:90:4a	56	1
98:97:d1:a3:90:4a	56	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	77	1
98:97:d1:a3:90:4a	65	1
98:97:d1:a3:90:4a	65	1
98:97:d1:a3:90:4a	65	1
98:97:d1:a3:90:4a	65	1
98:97:d1:a3:90:4a	65	1
98:97:d1:a3:90:4a	65	1
98:97:d1:a3:90:4a	65	1
98:97:d1:a3:90:4a	65	1
98:97:d1:a3:90:4a	65	1
fc:52:8d:20:e2:de	27	1
98:97:d1:a3:90:4a	56	1
98:97:d1:a3:90:4a	56	1
98:97:d1:a3:90:4a	56	1

- Observaciones: Como se puede observar en el archivo generado, se capturan los paquetes relativos a la utilización de canal. Sin embargo, en una inspección detallada del archivo, no se detecta ninguna trama de los canales dos y tres. Esto puede deberse a un fallo del *dongle* ya que, chequeando con *Wireshark*, sí se observa que el *dongle* está capturando varios canales y no únicamente el 1.

G. Concurrencia de procesos Crontab:

- Esquema de la prueba:

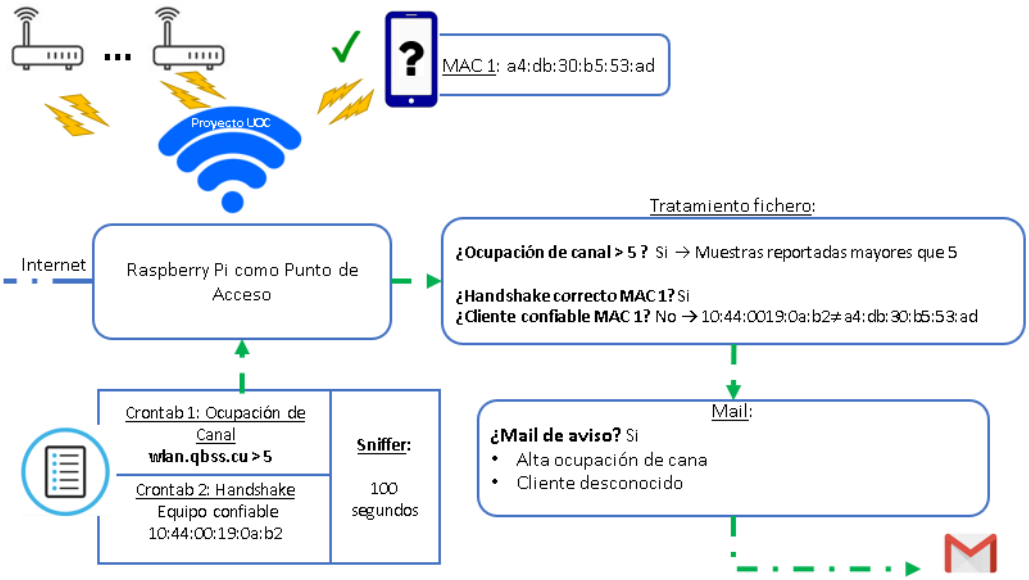


Figura 29 - Prueba G Concurrencia Crontab

- Tiempo de *sniffer*: 100 segundos.

- *Mail* generado: 4 *mail*s. 2 propios de los *scripts* implementados: uno informando del canal saturado y otro por la conexión de un cliente desconocido. De forma adicional, se envían dos *mail*s con los logs de las dos tareas programadas.

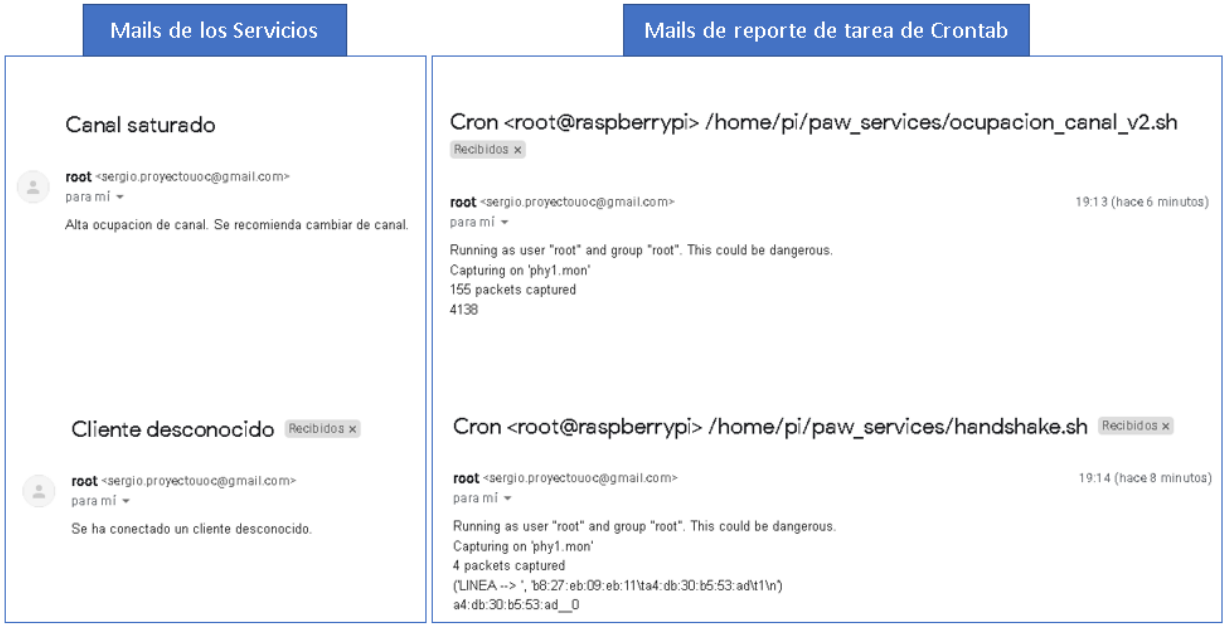


Figura 30 - Prueba G Mails Concurrencia Crontab

- Archivo generado: En Anexo I

- Observaciones: Como se puede observar, las dos tareas automáticas pueden convivir perfectamente sin ningún tipo de problemas. Tanto los *mail*s de log, propios del Crontab, como los *mail*s propios de nuestros servicios implementados funcionan correctamente.

H. Estabilidad de los servicios (I):

- Esquema de la prueba:

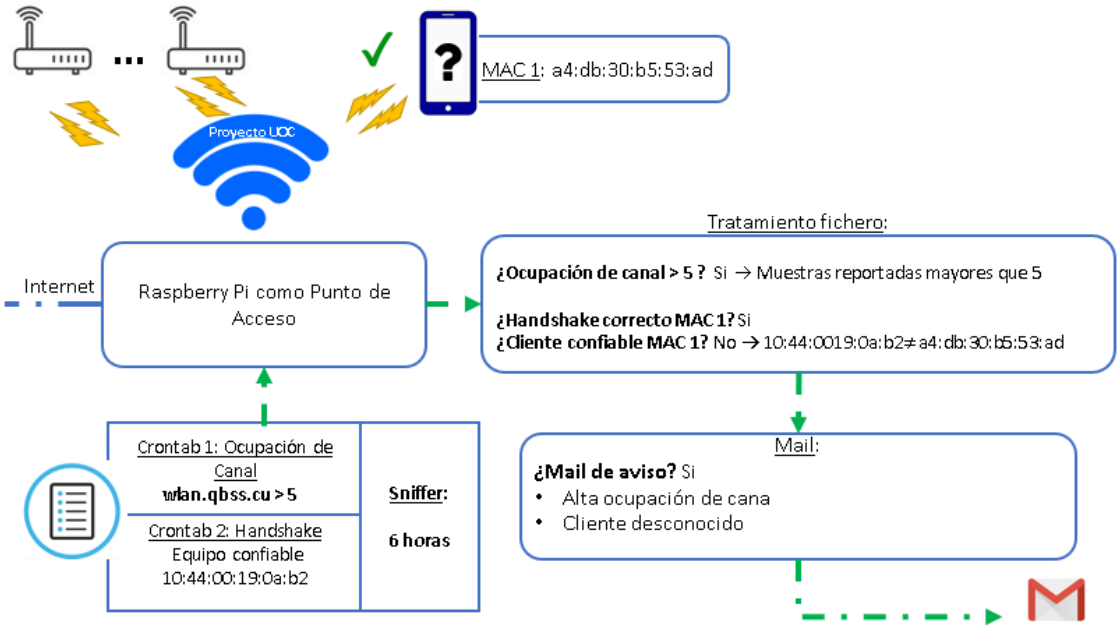


Figura 31 - Prueba H Estabilidad de los Servicios I

- Tiempo de *sniffer*: 21600 segundos.

- *Mail* generado: 4 *mail*s. 2 propios de los *scripts* implementados: uno informando del canal saturado y otro por la conexión de un cliente desconocido. De forma adicional, se envían dos *mail*s con los logs de las dos tareas programadas.

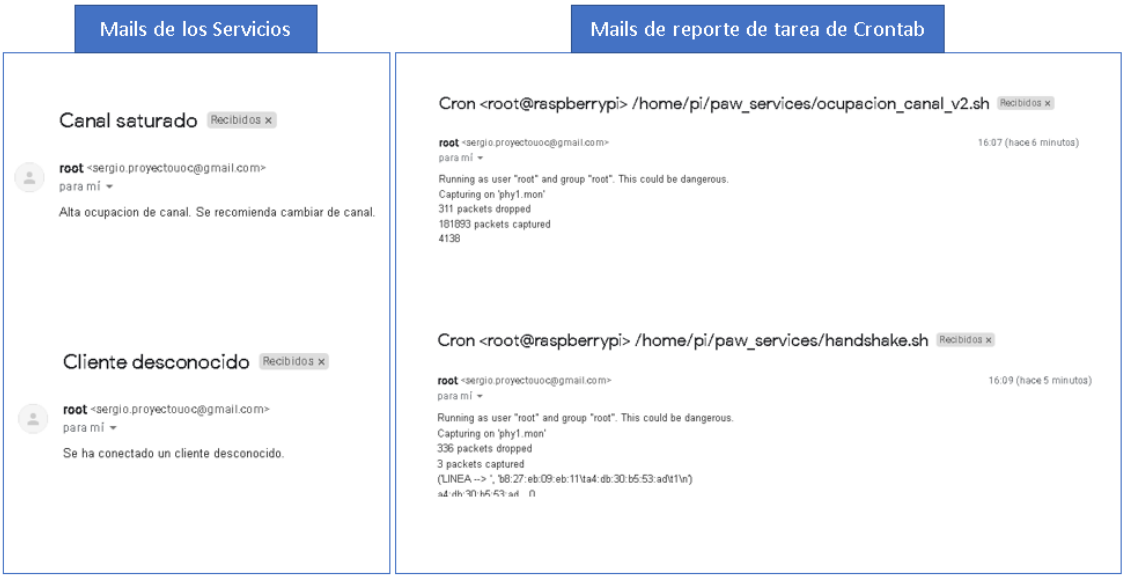


Figura 32 - Prueba H Mail de Estabilidad de los Servicios I

- Observaciones: La usabilidad de la Raspberry Pi no queda mermada por utilizar durante tanto tiempo (6 horas) recursos de la misma.

I. Estabilidad de los servicios (II):

- Esquema de la prueba:

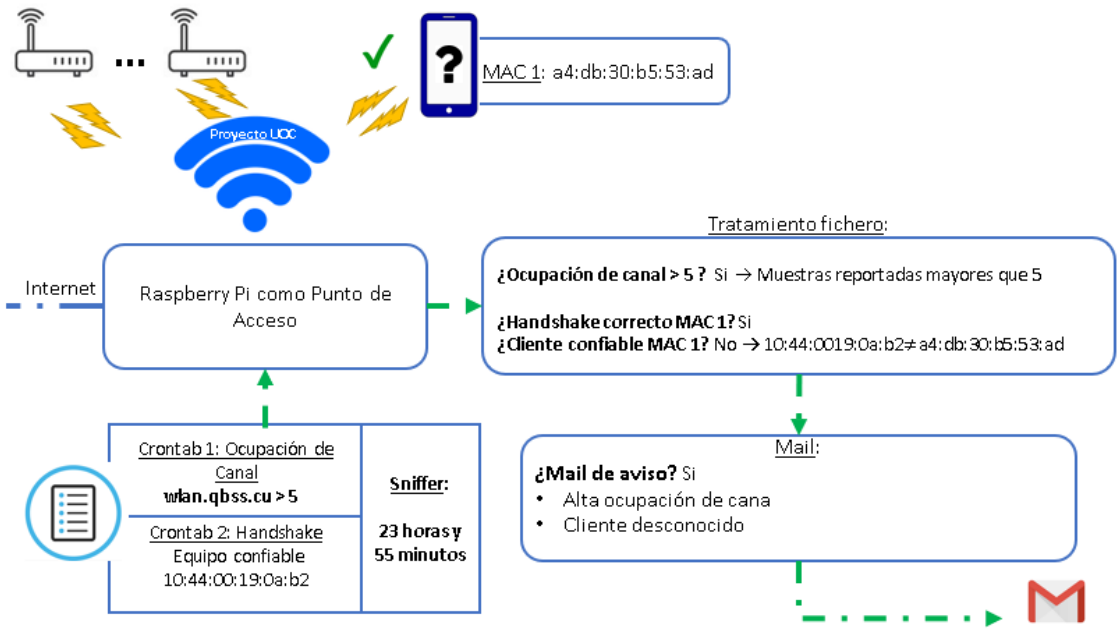


Figura 33 - Prueba I Estabilidad de los Servicios II

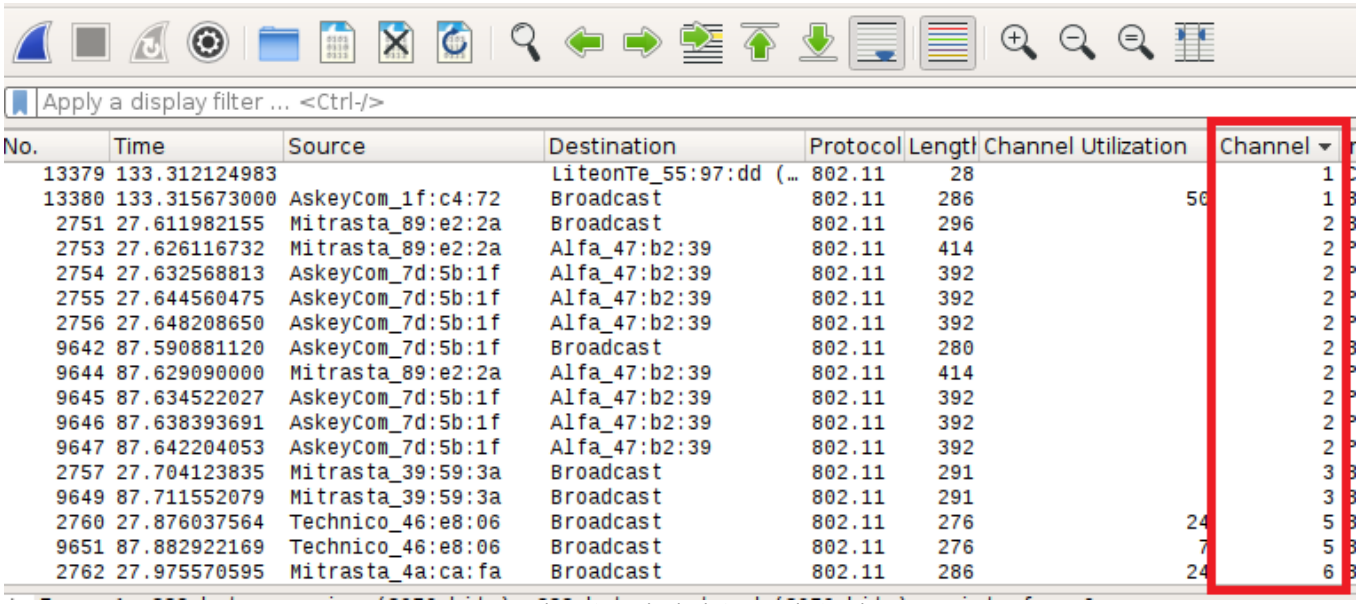
- Tiempo de *sniffer*: 86100 segundos.

- Observaciones: En este caso, la Raspberry Pi no se comporta correctamente. Tras diversas pruebas, no se ha llegado a tener una muestra de la funcionalidad completa cuando los *sniffer* trabajan 24 horas. En estas pruebas se ha visto que, llegado un punto, uno de los dos procesos de *sniffer* falla, por lo que una de las comprobaciones (ocupación de canal o *handshake*) no llega a realizarse. Adicionalmente, se ha comprobado que la usabilidad de la Raspberry Pi cae, dejando la interfaz gráfica mermada. Todo apunta a que un sobrecalentamiento de la placa (no se ha registrado un uso de la CPU excesivamente alto) es el causante de los fallos reportados.

5. Impedimentos Encontrados

A continuación de listarán los impedimentos técnicos encontrados en las pruebas realizadas:

- Captura de canales: Aún sabiendo que el *dongle* Alfa captura tráfico de distintos canales, no se han recuperado tramas de los canales 2 y 3. Para asegurar tráfico en dichos canales, se conectaron dos puntos de acceso, los cuales radiaban en los canales 2 y 3. Además, a estos puntos de acceso se conectaron diversos dispositivos para asegurar tráfico en el interfaz WiFi. Aun implementando estas medidas, se recuperaron muy pocas tramas de los comentados canales.



No.	Time	Source	Destination	Protocol	Lengt	Channel Utilization	Channel
13379	133.312124983		LiteonTe_55:97:dd (…	802.11	28		1
13380	133.315673000	AskeyCom_1f:c4:72	Broadcast	802.11	286	56	1
2751	27.611982155	Mitrasta_89:e2:2a	Broadcast	802.11	296		2
2753	27.626116732	Mitrasta_89:e2:2a	Alfa_47:b2:39	802.11	414		2
2754	27.632568813	AskeyCom_7d:5b:1f	Alfa_47:b2:39	802.11	392		2
2755	27.644560475	AskeyCom_7d:5b:1f	Alfa_47:b2:39	802.11	392		2
2756	27.648208650	AskeyCom_7d:5b:1f	Alfa_47:b2:39	802.11	392		2
9642	87.590881120	AskeyCom_7d:5b:1f	Broadcast	802.11	280		2
9644	87.629090000	Mitrasta_89:e2:2a	Alfa_47:b2:39	802.11	414		2
9645	87.634522027	AskeyCom_7d:5b:1f	Alfa_47:b2:39	802.11	392		2
9646	87.638393691	AskeyCom_7d:5b:1f	Alfa_47:b2:39	802.11	392		2
9647	87.642204053	AskeyCom_7d:5b:1f	Alfa_47:b2:39	802.11	392		2
2757	27.704123835	Mitrasta_39:59:3a	Broadcast	802.11	291		3
9649	87.711552079	Mitrasta_39:59:3a	Broadcast	802.11	291		3
2760	27.876037564	Technico_46:e8:06	Broadcast	802.11	276	24	5
9651	87.882922169	Technico_46:e8:06	Broadcast	802.11	276	7	5
2762	27.975570595	Mitrasta_4a:ca:fa	Broadcast	802.11	286	24	6

Figura 34 - Wireshark Captura de Canal

La calidad del *dongle* Alfa puede ser la causa de esta falta de precisión en ciertas bandas de frecuencia pertenecientes al 2.4GHz.

- Pérdida de paquetes: Ligado al punto anterior, se han detectado casos donde no se han capturado paquetes que, con seguridad, se saben transmitidos. Esto puede ser causado por el propio *TShark*, descartando paquetes por la ingente cantidad de información que debe filtrar, o por el *dongle* Alfa escogido, el cual puede no detectar alguna trama.

- Usabilidad de Raspberry Pi tras uso intensivo: Se ha detectado que la Raspberry Pi sufre un deterioro en cuanto a la usabilidad de la interfaz gráfica por parte del usuario cuando los servicios quedan activos varios días. Sin apreciarse variaciones bruscas o excesivas del uso de CPU, los indicios parecen apuntar a un sobrecalentamiento de la Raspberry Pi. Este hecho está documentado en distintas páginas especializadas en Raspberry Pi. Una mejora podría ser la instalación de unos disipadores para disminuir así la temperatura de la placa.

6. Guía de Usuario

El montaje y puesta en marcha de las Raspberry Pi es rápido y sencillo. Además, existen muchos tutoriales en internet explicando, paso a paso, cada punto del proceso. Como documentación de referencia respecto a esto cabe destacar el tutorial *Setting Up Your Raspberry Pi* [15], por su simplicidad y enfoque didáctico.
A continuación, se hace un breve resumen de los pasos a seguir:

1) Descarga del SO: Primeramente, se descargará el sistema operativo. En la página oficial se podrá escoger el más adecuado (https://www.raspberrypi.org/downloads/raspbian/).

2) Descarga de Etcher e instalación: Este programa de apoyo permitirá grabar la imagen del SO en la tarjeta SD.

3) Grabación del SO en la tarjeta SD: Se introducirá la SD en su soporte correspondiente, y se arrancará el programa *Etcher*, donde se grabará la SO en la SD.

Figura 35 - Etcher Grabación SD

4) Instalación de la SD y periféricos: Se instalarán los periféricos que, en un principio, serán necesarios para poder configurar la Raspberry Pi: teclado, ratón y monitor de pantalla. Además, se introducirá la SD en la ranura especificada de la Raspberry Pi.

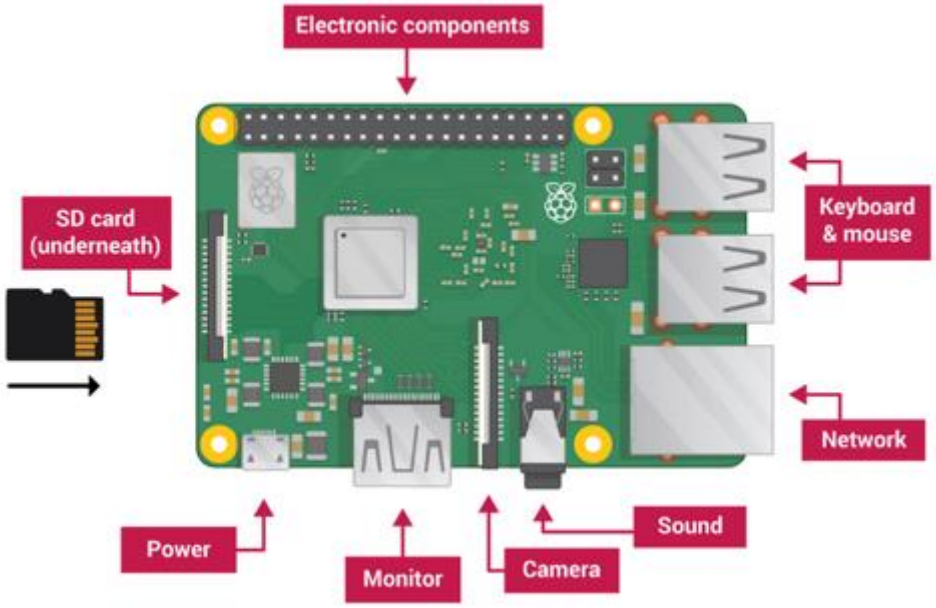


Figura 36 - Entradas Placa

5) A continuación, se conectará vía cable ethernet la Raspberry Pi al *router* comercial proporcionado por el ISP.

6) Por último, se conectará la Raspberry Pi a la corriente eléctrica: El puerto de alimentación será micro USB y deberá proveer 5V/2.5A para que el correcto funcionamiento de la placa.

7) Una vez hecho esto, la Raspberry Pi se encenderá bajo un proceso ordenado en el cual nos pedirá ciertas elecciones:
   a) Mensaje de Bienvenida:

Figura 37 - Pantalla Inicio Raspberry Pi

b) Configuración del idioma:

Figura 38 - Idioma Raspberry Pi

c) Cambio de *password*: Por defecto las RaspberyPi vienen preconfiguradas con el usuario: pi y la contraseña: pi.

Figura 39 - Password Raspberry Pi

d) WiFi *Network*: En este caso, al tener ya conexión vía Ethernet, se seleccionará la opción *Skip*.

Figura 40 - Network Raspberry Pi

e) *Update Software*: La Raspberry Pi buscará de forma automática la última *release* de SW disponible. Se recomienda ejecutar este paso.

Figura 41 - Update SW Raspberry Pi

f) *Reboot*: Es necesario hacer un *reboot* para cargar toda la configuración anteriormente comentada.

Figura 42 - Reboot Raspberry Pi

En este punto se tendría ya una Raspberry Pi 3 Model B+ totalmente operativa y lista para usar.

8) Se abrirá un LXTerminal:

Figura 43 - LXTerminal II

9) Implementación de punto de acceso WiFi: Para la implementación de un punto de acceso WiFi, se deberán seguir las siguientes instrucciones, las cuales deberán ser introducidas en el LXTerminal abierto en el paso 8. Todos los puntos marcados como **<Palabra>** deberán ser sustituidos por el valor que el usuario crea conveniente.

```
#Descargamos actualizaciones
sudo apt-get update
sudo apt-get upgrade

#Instalamos Hostpad: Para crear puntos de acceso y servers de autenticacion.
sudo apt-get install hostapd bridge-utils

#Paramos SW descargado anteriormente
sudo systemctl stop hostapd

#Paramos el DHCP para los interfaces
#Abrimos el fichero dhcpd.conf
sudo nano /etc/dhcpcd.conf
denyinterfaces wlan0
denyinterfaces eth0

#Creamos una interfaz nueva. Será un bridge.
sudo brctl addbr br0
sudo brctl addif br0 eth0

#Retocamos los interfaces
sudo nano /etc/network/interfaces

-- Levantamos le bridge tras un boot
auto br0
-- Asignamos una IP automáticamente
iface br0 inet manual
-- Conectamos el interfaz eth0 y el wlan0
bridge_ports eth0 wlan0

#Retocamos el fichero hostapd. Aquí elegiremos el ssid, contraseña…
sudo nano /etc/hostapd/hostapd.conf
interface=wlan0
bridge=br0
ssid=<Nombre_WiFi>
hw_mode=g
channel=<Canal>
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=<Contraseña_WiFi>
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP

#Aplicamos los cambios
sudo nano /etc/default/hostapd
-- Quitamos comentario de DAEMON_CONF y añadimos el nuevo directorio creado
DAEMON_CONF="/etc/hostapd/hostapd.conf"


sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd

sudo reboot
```

10) Se descargarán los SW necesarios para el envío de *mails*:

```
#Descargamos SW
sudo apt-get install ssmtp mailutils mpack
```

11) Se configurará el servidor de *mail*: La parametrización establecida es en base a un servidor Gmail. Todos los puntos marcados como **<Palabra>** deberán ser sustituidos por el valor que el usuario crea conveniente.

```
#Configuramos el SSMTP
sudo nano /etc/ssmtp/ssmtp.conf

--Añadimos la linea
AuthUser=<Mail_de_usuario>
AuthPass=<Contraseña_mail_de_usuario>
--Descomentamos la linea
FromLineOverride=YES
--Retocamos la linea
mailhub=smtp.gmail.com:587
--Añadimos la linea
UseSTARTTLS=YES
```

12) A continuación, se descargará el programa *TShark*

```
#Descargamos Tshark
sudo apt-get install tshark
```

13) Se chequeará en que interfaz está mapeado el *dongle* Alfa (si no apareciera, se recomienda hacer un apagado/encendido de la Raspberry Pi):

```
#Chequeamos en que interfaz debemos capturar tráfico
sudo tshark -D

#Si no apareciera el dongle Alfa, se realiza apagado/encendido:
sudo reboot
```

14) El interfaz con el cual se trabajará tendrá una posición dentro de la lista mostrada y será presentado con la siguiente notación:

- phyX.mon (siendo X un número)

La posición que ocupe dicho interfaz debe ser tenida en cuenta para próximos pasos.

15) Todos los ficheros deberán ser creados y guardados en una carpeta común. El *path* a esa carpeta será representado en los diferentes códigos como:

- **<Path_del_archivo>**

16) En la carpeta creada en el punto 15, se creará un nuevo archivo con extensión .sh. Para ello se clicará con el botón derecho, escogiendo la opción crear nuevo > archivo vacío. Una vez hecho esto, se especificará el nombre del archivo (**<Nombre_archivo>**) con la terminación .sh, donde se especificará el código para el *script* de ocupación de canal.

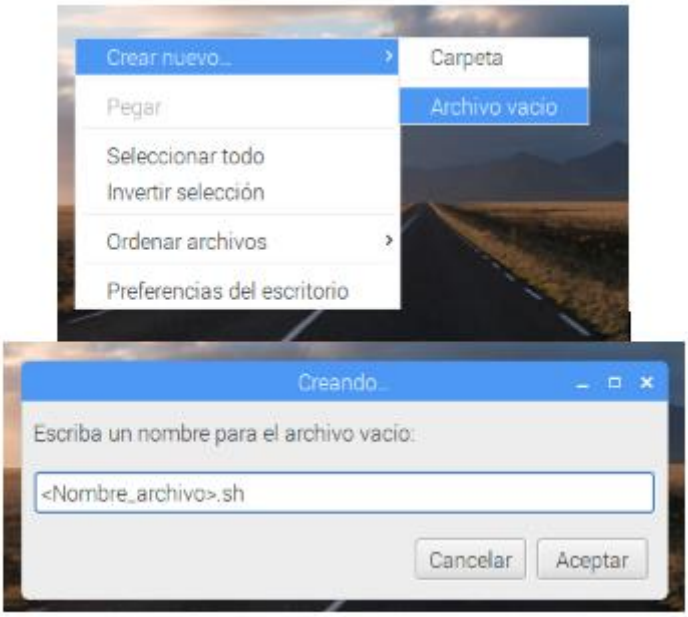


Figura 44 - Nuevo Archivo

17) El paso 16 se repetirá otras dos veces: en una se creará un archivo con extensión .sh (para la automatización del *handshake*) y otro archivo con extensión .py (para el tratamiento del fichero del *handshake*). En total se tendrá tres archivos como:
    - **<Nombre_archivo_ocupacion_canal>.sh**
    - **<Nombre_archivo_handshake_1>.sh**
    - **<Nombre_archivo_handshake_2>.py**

Todos los ficheros deberán ser guardados en la carpeta creada en el punto 15.

18) Se abrirá el fichero <Nombre_archivo_ocupacion_canal>.sh, editándose las partes reflejadas cómo **<Palabra>.** El siguiente código editado deberá ser pegado en el fichero abierto:

```
#Editaremos
#!/bin/bash
sudo tshark -i<Número_de_interfaz_paso_14> -Y"wlan_radio.channel == <Canal_paso_9> or wlan_radio.channel == <Canal_paso_9_menos_1> or wlan_radio.channel == <Canal_paso_9_mas_1> and wlan.qbss.cu > <Valor_ocupacion>" -a duration:86100 -T fields -e wlan.ta -e wlan.qbss.cu > <Path_del_archivo>/<Nombre_del_archivo_ocupacion_canal.txt>
declare -i n
declare -i zero=0
n=$(cat <Nombre_del_archivo_ocupacion_canal.txt>|wc -w)
echo $n
test $n -gt $zero && echo "Alta ocupacion de canal. Se recomienda cambiar de canal." | mail -s "Canal saturado" <Dirección_Mail_de_recepcion_de_aviso> || echo "Todo Ok"
```

En caso de que el valor del **<Canal_paso_9>** sea:

- Valor 1:
  - **<Canal_paso_9_mas_1>:** Tomará el valor 2.
  - **<Canal_paso_9_menos_1>:** Tomará el valor 3.
- Valor 13:
  - **<Canal_paso_9_mas_1>:** Tomará el valor 12.
  - **<Canal_paso_9_menos_1>:** Tomará el valor 11.

Se recomienda que el parámetro **<Valor_ocupacion>** tome valor 217 (85% ocupación de canal). Un nuevo valor de ocupación puede calcularse de la forma:

Utilización de canal = (wlan.qbss.cu * 100) / 255

Siendo el valor 255 el 100% de ocupación de canal.

19) Se abrirá el fichero creado en el paso 16, <Nombre_archivo_handshake_1>.sh. Se editarán las partes especificadas como **<Palabra>** y se copiará el siguiente código tras ser modificado.

```
#!/bin/bash

tshark -i4 -Y"wlan_rsna_eapol.keydes.msgnr and wlan.addr == <MAC_Raspberry_Pi>" -a duration:86000 -T fields -e wlan.ta -e wlan.ra -e wlan_rsna_eapol.keydes.msgnr > <Path_del_archivo>/<Nombre_del_archivo_handshake.txt>
python <Path_del_archivo>/<Nombre_archivo_handshake_2>.py
```

20) Se abrirá el fichero <Nombre_archivo_handshake_2>.py creado en el paso 17 y se pegarán las siguientes líneas de código. Se deberán cambiar los puntos marcados cómo **<Palabra>**. Hay que tener en cuenta que:

- **<Dirección_Mail_de_recepcion_de_aviso>**: Será el *mail* de usuario que reciba las notificaciones.

- **<Path_del_archivo>/<Nombre_del_archivo_ocupacion_canal.txt>**: Deberá coincidir con los especificados en el punto 19. En caso contrario, el fichero (ruta hasta él y el nombre de este) no podrá ser leído ni tratado.

```
#!/usr/bin/env python

import subprocess
send_OK_mail_command = '"echo "Se ha conectado un cliente desconocido." | mail -s "Cliente desconocido" <Dirección_Mail_de_recepcion_de_aviso>'"
send_KO_mail_command = '"echo "Se ha intentado conectar un cliente." | mail -s "Posible ataque por fuerza bruta" <Dirección_Mail_de_recepcion_de_aviso>'"
start_debuging = False
_dict = {}
handshake_path='<Path_del_archivo>/<Nombre_del_archivo_ocupacion_canal.txt>'
mac_whitelist = ['<MAC_confiable_1>','<MAC_confiable_2']

def prepare_debugging(file):
        prepared = False
        while not prepared:
                pos = file.tell()
                current_fake_line = file.readline()
                prepared = int([word for word in current_fake_line.split('\t')][2]) == 1
                if prepared:
                        file.seek(pos)

def do_scanner():
        with open(handshake_path,'r') as file:
                prepare_debugging(file)
                current_step = 0
                current_session = 0
                last_device = 0
                for line_raw in file:
                        print('LINEA --> ',line_raw)
                        line = [word for word in line_raw.split('\t')]
                        current_line_step = int(line[2])
                        current_line_device = line[1] if int(line[2]) % 2 == 1 else line[0]
                                                allow_continue = True
                        if last_device == 0:
                                last_device = current_line_device
                        if ((current_line_step <= current_step) or (current_line_device != last_device)):
                                if current_line_device != last_device:
                                        if current_step > 0:

                                                current_session += 1
                                else:
                                        allow_continue = False
                        if allow_continue:
                                last_device = current_line_device
                                current_step = current_line_step
                                key = current_line_device + '__' + str(current_session)
                                print(key)
                                value = []
                                print(line)
                                if key not in _dict:
                                        value = [line]
                                else:
                                        value = _dict[key]
                                        value += [line]
                                _dict[key] = value
                                if current_step is 4:
                                        current_line_step = 0
                                        current_step = 0
                                        current_session += 1
                print(_dict)

def do_mailling():
        for key,value in _dict.items():
                if len(value) <= 2:
                        process                                                                                 =
subprocess.Popen(send_KO_mail_command,stderr=subprocess.PIPE,stdout=subprocess.PIPE,shell=True).communicate()
                elif len(value) > 2 and key.split('__')[0] not in mac_whitelist: #Comparamos contra nuestra whitelist.
                        process                                                                                 =
subprocess.Popen(send_OK_mail_command,stderr=subprocess.PIPE,stdout=subprocess.PIPE,shell=True).communicate()

do_scanner()
do_mailling()
```

21) Tras tener los tres archivos creados, se editará el crontab. Esta herramienta permitirá la automatización de las tareas:

a) Se abrirá Crontab en una ventana LXTerminal:

```
#Demonio cron
sudo crontab -e
```

b) Se pedirá al usuario la forma de editar el archivo. Se escogerá la opción 2.

c) El archivo se modificará de acuerdo con las especificaciones de usuario, sustituyendo los parámetros marcados como **<Palabra>:**

```
<minuto> <hora> <día> <mes> <día_de_la_semana> <Path_del_archivo>/<Nombre_archivo_handshake_2>.sh
<minuto> <hora> <día> <mes> <día_de_la_semana> <Path_del_archivo>/<Nombre_archivo_ocupacion_canal>.sh
```

Los valores que se pueden dar a las variables arriba especificadas serían:

Minuto	Hora	Día	Mes	Día de la semana
Rango: 0-59	Rango: 0-23	Rango: 1-31	Rango: Enero: 1 .... Diciembre: 12	Rango: Domingo: 0 .... Domingo: 7

Figura 45 - Configuración Crontab II

En caso de que se quiera hacer el reporte todos los días, las siguientes variables deberían tomar el valor de:

- **<día>**: *
- **<mes>**: *
- **<día_de_la_semana>**: *

7. Inversión económica final

Este proyecto tiene un carácter eminentemente didáctico, de forma que dota al usuario final del conocimiento suficiente para monitorizar una red WiFi de una manera sencilla y efectiva. Por tanto, el montaje final debe tener un coste económico bajo, pudiendo ser accesible para la gran mayoría de usuarios.

Artículo	Unidades	Precio Unidad	Total
Raspberry Pi 3 Model B+	1	38,00 €	38,00 €
microSD Integral de 16GB	1	8,50 €	8,50 €
Dongle Alfa AWUS036NH	1	17,00 €	17,00 €
		Suma Total	64,00 €

8. Conclusiones

En el presente Trabajo Final de Máster se ha implementado un punto de acceso, perfectamente funcional, con servicio de monitorización WiFi y envío de alertas de acuerdo con una serie de eventos. El proyecto ha sido enfocado en base a dos vertientes. Una primera didáctica, donde se pretendía acercar la tecnología al usuario medio y potenciar su autonomía, de tal forma que se alentara la búsqueda de conocimiento. La segunda vertiente, de carácter técnico, se ha centrado en la de implementación del comentado punto de acceso y los servicios de monitorización y alerta. Remarcar que el gran punto de confluencia de las comentadas vertientes, a parte del desarrollo del trabajo en sí mismo, ha sido la tecnología WiFi: un estándar que ha emergido recientemente y ha supuesto una revolución en la vida de las personas, cambiando de formas inimaginables el patrón de navegación de los usuarios. Una tecnología que cualquier persona utiliza diariamente y que, de forma paradójica, muy poca gente conoce su funcionamiento.

El proyecto ha sido abordado en fases diferenciadas:

- Elección de HW y SW.
- Estudio de proyectos y solucionas análogas a la propuesta.
- Desarrollo de los servicios a implementar.
- Pruebas funcionales.
- Desarrollo de guía de usuario.

Desarrollando de esta forma el proyecto, se ha ido de un plano abstracto, donde se ha investigado la tecnología más apropiada y las herramientas o SW actuales en el mercado, a un plano más concreto, aterrizando y parametrizando una solución propia que se ajustara a las necesidades detectadas. De esta forma, la elaboración del proyecto ha seguido una consecución lógica en búsqueda de la mejor solución, teniendo en cuenta que se debía guardar un equilibrio entre complejidad (de forma que un usuario medio pudiera realizar todo el montaje), funcionalidad (una implementación que funcionara de forma correcta) y coste (que supusiera una inversión al alcance del usuario medio).

La tecnología escogida para abordar el proyecto, la placa Raspberry Pi 3 Model B+, se ha presentado como un acierto: una tecnología robusta y desarrollada expresamente para aplicaciones académicas, perfecta para usuarios sin conocimientos profundos de programación y/o de sistemas operativos, altamente configurable y con infinidad de sencilla documentación albergada en diversas páginas de Internet. Todo lo anteriormente comentado, a un precio económico y sin depender de licencias o de SW de terceros. Sin embargo, y como punto negativo, la placa Raspberry Pi se tuvo que completar con un *dongle* WiFi. La inclusión de este *dongle* se presentó como la solución

adoptada a una limitación del *chipset* WiFi Broadcom que incorporaba la placa, el cual no podía trabajar en *monitor mode*. Este modo de trabajo era imprescindible para poder capturar las “tramas” WiFi.

En cuanto a las funcionalidades desarrolladas, había cinco grandes puntos a abordar:

- Implementación del punto de acceso WiFi.
- Envío de *mail*s desde el propio punto de acceso WiFi.
- Estudio de la conexión de usuarios (*sniffer*).
- Estudio de la ocupación de canal (*sniffer*).
- Automatización de tareas.

El éxito del trabajo residía en el desarrollo de estas funcionalidades y del nivel de orquestación que se pudiera alcanzar a la hora de hacerlas trabajar de forma concurrente, sin que ninguna de ellas quedara mermada. Esto se considera logrado, ya que se ha desarrollado un servicio global en el cual las funcionalidades comentadas trabajaban de forma coordinada y se retroalimentan. Las pruebas realizadas sobre el montaje rebelan que el entregable final es estable y funcional, registrándose pocos fallos en las mismas. El único aspecto negativo sería la estabilidad y usabilidad de la Raspberry Pi bajo condiciones de trabajo de 24 horas, donde los *sniffer* deben capturar y analizar una ingente cantidad de datos. En estas condiciones el servicio global funciona de forma incompleta, siendo incapaz de tener levantados ambos *sniffer* a la vez y mermando la usabilidad de la interfaz gráfica de usuario. Este punto puede ser debido a la temperatura de la placa, la cual aumenta considerablemente.

Por otro lado, y no menos importante, la documentación y el desarrollo de los servicios debían ser explicados de forma sencilla. Acercar y hacer accesible la tecnología y, en concreto, algunos aspectos del estándar 802.11n (WiFi) era uno de los puntos clave del proyecto. A la par que se han ido desarrollando cada una de las funcionalidades, se han añadido comentarios que han permitido un mejor seguimiento del trabajo. De esta forma se han explicado aspectos técnicos sin entrar a muy bajo nivel, evitando puntos considerados superfluos, pero manteniendo el debido rigor técnico. Las gráficas desarrolladas para cada apartado, además, han servido de apoyo para las explicaciones, haciendo que estas fueran más comprensibles. A todo lo anteriormente comentado hay que sumar la creación de una guía de usuario. Esta guía se ha estructurado de tal forma que, siguiendo los pasos secuencialmente, se consiga un entregable totalmente funcional. Engloba puntos como el montaje *hardware* (equipos a emplear, conexiones…), la elección y grabación del sistema operativo o la creación de los *scripts* de cada una de las funcionalidades. Cabe destacar que el enfoque de la guía es puramente ejecutivo, pensado para los usuarios de más bajo nivel, los cuales

no deben más que realizar las pertinentes conexiones y personalizar las partes marcadas en los *scripts*. De forma añadida, y de cara al usuario final, se ha hecho una pequeña valoración económica del montaje final, remarcado así el hecho de que la tecnología puede y debe estar al alcance de todas las personas.

Este trabajo, dado su enfoque y su campo de acción, puede servir de punto de partida para otros proyectos. Como posibles mejoras se podrían tener:

- Inclusión de nuevos disipadores: Esta mejora de la placa haría que la temperatura de esta bajara. La estabilidad y usabilidad, por tanto, se verían mejoradas.
- Envío de *mail*s: Se podría mejorar la forma en la que se envían los *mail*s. Enviar un único *mail* resumen con todas las incidencias facilitaría la gestión y la toma de decisiones al usuario final.
- Ocupación de canal: El tratamiento de la ocupación de canal podría ir un paso mas allá. A parte de detectar cuando hay una ocupación de canal alta, se podría indicar al usuario a qué canal moverse. Este desarrollo, bastante complejo, debería tener en cuenta en qué momentos se ha detectado la ocupación de canal alta y, tras analizar los reportes a lo largo del tiempo del resto de canales, proponer un nuevo canal de funcionamiento.
- Cambio de canal automático: Derivado del punto anterior, se podría automatizar el cambio de canal. De esta forma, se descargaría de esta gestión al usuario.
- Banda de frecuencia: El modelo de Raspberry Pi escogido tiene un interfaz WiFi de 5GHz el cual puede trabajar bajo el estándar 802.11ac. Este estándar puede utilizar anchos de banda cuatro veces superiores al 802.11n utilizado en este proyecto. Además, es más rápido y eficiente. Se podría implementar un punto de acceso WiFi dual, con un WiFi sobre 2.4GHz y otro sobre 5GHz.
- Ampliación del *4-Way Handshake*: Al igual que en el presente proyecto se analiza quién se conecta al punto de acceso vía WiFi en 2.4GHz, se podría hacer un estudio similar sobre el nuevo WiFi de 5GHz.

Gracias a la flexibilidad que da el ecosistema Raspberry Pi, todos los puntos anteriormente comentados podrían ser desarrollados, sumando así nuevas vías de exploración dentro del ámbito académico.

9. Glosario

MUA – *Mail User Agent*.
MTA – *Mail Transfer Agent*.
RADIUS – Remote *Authtentication Dial-In User*.
MAC – *Media Access Control*.
CLI – *Command Line Interface*.
IoT – *Internet of things*.
SBC – *Single Board Computer*.
SO – Sistema Operativo.
SW – *Software*.
HW – *Hardware*.
Script – Documento que contiene instrucciones en un código de programación.
SD *Card* – *Secure Digital Card*.
SSID – *Service Set IDentifier*.
PSK – *Pre-Shared Key*.
PMK – *Pairwise Master Key*.
Passphrase – Contraseña del SSID WiFi.
Sniffer – Analizador de protocolos.
***Dongle* WiFi** – Adaptador que añade funcionalidades de *sniffer* WiFi al montaje inicial.
ISP – Internet *Service Provider*

10. Bibliografía

[1] minProjects. (2018). https://www.instructables.com/id/Use-Raspberry-Pi-3-As-Router/

[2] SurferTim. (2017). https://github.com/SurferTim/documentation/blob/6bc583965254fa292a470990c40b145f553f6b34/configuration/wireless/access-point.md#using-the-raspberry-pi-as-an-access-point-to-share-an-internet-connection

[3] The Wireshark Team. (2019). Wireshark (3.0.0). https://www.wireshark.org/#download

[4] The Wireshark Team. (2019). Tshark https://www.wireshark.org/#download

[5] Tcdump. (4.9.2). (2017). https://www.tcpdump.org/#latest-releases

[6] GitHub. (3). (2007). https://github.com/SYWorks/waidps

[7] Kismet. (2019). https://www.kismetwireless.net/docs/readme/packages/

[8] Proofpoint. (8.15.2). (2015). https://www.proofpoint.com/us/open-source-email-solution

[9] Mutt. (1.11.4). (2019). http://www.mutt.org/

[10] Unknown. https://wiki.debian.org/es/sSMTP

[11] Ghantous. M, Jaber. H, Darwish. Z y Tahan.O. PiMonitor: A Wi-Fi Monitoring Device Based on Raspberry-Pi. https://0-ieeexplore-ieee-org.cataleg.uoc.edu/document/8460248

[12] Redondi y Cesana.M. Building up knowledge through passive WiFi probes. https://0-www-sciencedirect-com.cataleg.uoc.edu/science/article/pii/S0140366417302773

[13] Abaya. W.F, Basa.J, Sy.M, Abad.A.C y Dadios.E.P. Low cost smart security camera with night vision capability using Raspberry Pi and OpenCV. https://0-ieeexplore-ieee-org.cataleg.uoc.edu/document/7016253/authors#authors

[14] Camacho.M. (2014). https://hacking-etico.com/2014/10/29/raspberry-sniffing-el-juguete-oculto/

[15] Raspberry.org. Setting Up your Raspberry Pi https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up

11. Anexo I

- **Cliente Desconocido:**

b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	3
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	4

- **Posible Ataque WiFi:**

a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1

a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2

- **Cliente Desconocido + Posible Ataque WiFi:**

b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	3
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	4
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1

10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1

- **Dos clientes permitidos conectados**:

b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	3
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	4
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	1
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	10:44:00:19:0a:b2	3
10:44:00:19:0a:b2	b8:27:eb:09:eb:11	4

- **Concurrencia de Proceso Crontab**:
  - **Ocupación de canal**:

d4:7b:b0:d2:42:54	23	1
e8:d1:1b:4c:ab:25	25	1
d4:7b:b0:1f:c4:72	81	1
e8:d1:1b:4c:ab:25	29	1
e8:d1:1b:4c:ab:25	29	1
e8:d1:1b:4c:ab:25	21	1
d4:7b:b0:d2:42:54	45	1
e8:d1:1b:4c:ab:25	21	1
d4:7b:b0:d2:42:54	44	1
e8:d1:1b:4c:ab:25	22	1
d4:7b:b0:d2:42:54	44	1
d4:7b:b0:d2:42:54	44	1
d4:7b:b0:d2:42:54	47	1
d4:7b:b0:d2:42:54	42	1
d4:7b:b0:1f:c4:72	24	1
d4:7b:b0:1f:c4:72	24	1
d4:7b:b0:d2:42:54	42	1
d4:7b:b0:1f:c4:72	23	1
fc:52:8d:20:e2:de	44	1
d4:7b:b0:d2:42:54	45	1
e8:d1:1b:4c:ab:25	25	1
d4:7b:b0:d2:42:54	45	1
98:97:d1:a3:90:4a	62	1
98:97:d1:a3:90:4a	62	1
98:97:d1:a3:90:4a	62	1
d4:7b:b0:d2:42:54	28	1
d4:7b:b0:d2:42:54	16	1
d4:7b:b0:d2:42:54	16	1

d4:7b:b0:d2:42:54	16	1
d4:7b:b0:d2:42:54	16	1
d4:7b:b0:d2:42:54	16	1
d4:7b:b0:d2:42:54	16	1
d4:7b:b0:d2:42:54	58	1
d4:7b:b0:d2:42:54	58	1
d4:7b:b0:d2:42:54	58	1
d4:7b:b0:1f:c4:72	24	1
e8:d1:1b:4c:ab:25	23	1
d4:7b:b0:d2:42:54	58	1
d4:7b:b0:d2:42:54	19	1
e8:d1:1b:4c:ab:25	19	1
e8:d1:1b:4c:ab:25	19	1
d4:7b:b0:d2:42:54	19	1
e8:d1:1b:4c:ab:25	19	1
e8:d1:1b:4c:ab:25	19	1
d4:7b:b0:d2:42:54	18	1
d4:7b:b0:1f:c4:72	25	1
d4:7b:b0:d2:42:54	44	1
d4:7b:b0:d2:42:54	20	1
d4:7b:b0:d2:42:54	20	1
d4:7b:b0:d2:42:54	20	1
d4:7b:b0:d2:42:54	20	1
d4:7b:b0:d2:42:54	60	1
d4:7b:b0:d2:42:54	60	1
d4:7b:b0:d2:42:54	53	1
d4:7b:b0:d2:42:54	52	1
d4:7b:b0:d2:42:54	49	1

- ***4-Way Handshake***:

b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	3
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	4

- **Estabilidad de los Servicios (I)**:

  - **Ocupación de canal**:

d4:7b:b0:d2:42:54	16	1
d4:7b:b0:d2:42:54	16	1
d4:7b:b0:d2:42:54	16	1
d4:7b:b0:d2:42:54	45	1
d4:7b:b0:d2:42:54	45	1
98:97:d1:a3:90:4a	57	1
e8:d1:1b:4c:ab:25	28	1
e8:d1:1b:4c:ab:25	29	1
d4:7b:b0:d2:42:54	55	1

e8:d1:1b:4c:ab:25	29	1
e8:d1:1b:4c:ab:25	29	1
e8:d1:1b:4c:ab:25	19	1
e8:d1:1b:4c:ab:25	19	1
d4:7b:b0:d2:42:54	37	1
e8:d1:1b:4c:ab:25	19	1
e8:d1:1b:4c:ab:25	26	1
d4:7b:b0:d2:42:54	63	1
e8:d1:1b:4c:ab:25	26	1
d4:7b:b0:d2:42:54	63	1
e8:d1:1b:4c:ab:25	26	1
d4:7b:b0:d2:42:54	39	1
d4:7b:b0:d2:42:54	39	1
d4:7b:b0:d2:42:54	39	1
e8:d1:1b:4c:ab:25	20	1
d4:7b:b0:d2:42:54	41	1
e8:d1:1b:4c:ab:25	21	1
d4:7b:b0:d2:42:54	42	1
d4:7b:b0:d2:42:54	50	1
d4:7b:b0:d2:42:54	50	1
d4:7b:b0:d2:42:54	50	1
d4:7b:b0:d2:42:54	50	1
e8:d1:1b:4c:ab:25	23	1
e8:d1:1b:4c:ab:25	23	1
e8:d1:1b:4c:ab:25	30	1
e8:d1:1b:4c:ab:25	30	1
d4:7b:b0:1f:c4:72	64	1
d4:7b:b0:d2:42:54	28	1
d4:7b:b0:1f:c4:72	64	1
d4:7b:b0:1f:c4:72	64	1
d4:7b:b0:1f:c4:72	64	1
98:97:d1:a3:90:4a	56	1
d4:7b:b0:d2:42:54	55	1
d4:7b:b0:d2:42:54	55	1
d4:7b:b0:d2:42:54	55	1
98:97:d1:a3:90:4a	60	1
d4:7b:b0:d2:42:54	55	1
d4:7b:b0:1f:c4:72	62	1
98:97:d1:a3:90:4a	60	1
d4:7b:b0:d2:42:54	55	1
98:97:d1:a3:90:4a	60	1
d4:7b:b0:d2:42:54	55	1
d4:7b:b0:1f:c4:72	63	1
d4:7b:b0:1f:c4:72	63	1
d4:7b:b0:1f:c4:72	63	1
d4:7b:b0:1f:c4:72	63	1
d4:7b:b0:1f:c4:72	63	1
d4:7b:b0:d2:42:54	27	1
98:97:d1:a3:90:4a	57	1
d4:7b:b0:d2:42:54	48	1

- ***4-Way Handshake*:**

b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	1
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	2
b8:27:eb:09:eb:11	a4:db:30:b5:53:ad	3
a4:db:30:b5:53:ad	b8:27:eb:09:eb:11	4