Universidad Zaragoza

Trabajo Fin de Máster

Análisis del Fast Roaming en redes WiFi Mesh para comunicaciones robóticas

Analysis of Fast Roaming in WiFi Mesh Networks for Robotics Communications

Autor

Guillermo Cruz Rojas

Directora

Natalia Ayuso Escuer

Máster Universitario en Ingeniería Informática

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2024

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por: el Programa de Becas y Ayudas del Instituto de Investigación en Ingeniería de Aragón (I3A); el proyecto PID2022-139615OB-I00, Robótica y comunicaciones en entornos complejos (ROBOCOMPLEX), Ministerio de Ciencia e Innovación, 2023-2026 y T45_23R: Robótica, Visión por Computador e Inteligencia Artificial.

Resumen

La robótica móvil es una de las alternativas más populares para tareas de inspección que conllevan cierto nivel de riesgo, como la búsqueda y rescate o la inspección de edificios inseguros. A pesar de ser aplicaciones que cada vez cuentan con un mayor nivel de autonomía, en ocasiones los robots no son capaces de determinar la mejor acción debido a la falta de datos o escenarios imprevisibles, por lo que la supervisión humana sigue teniendo un papel importante. El supervisor normalmente va a tener la necesidad de enviar comandos al robot en forma de teleoperación y al mismo tiempo necesita *feedback* sobre la situación del robot, que llega a través de la transmisión de datos recogidos por sensores, siendo la transmisión de vídeo desde una cámara integrada en el robot un escenario muy común.

Evidentemente esta interacción entre el supervisor y el robot precisa de un canal de comunicaciones fiable, con baja latencia para tener un tiempo de respuesta inmediato y un ancho de banda apropiado para la transmisión de vídeo, entre otros requisitos. En este proyecto se ha considerado una red WiFi Mesh para estudiar su viabilidad para soportar una aplicación robótica de teleoperación con transmisión de vídeo, debido a la posibilidad de obtener una mayor cobertura y ancho de banda frente a otro tipo de tecnologías comúnmente utilizadas como 5G o Zigbee. Concretamente, la característica de esta red que genera más interés es el proceso conocido como *Fast Roaming*, el cual permite que los dispositivos móviles cambien rápidamente de un punto de acceso a otro dentro de la misma red, sin interrupciones notables en la conexión, a priori.

En el presente trabajo se ha llevado a cabo un estudio teórico con el objetivo de comprender los procesos involucrados en el *Fast Roaming*, caracterizando posteriormente con experimentos prácticos cuándo y cómo ocurre el *Fast Roaming*, haciéndolo funcionar satisfactoriamente en el hardware con el que se ha trabajado en este proyecto. Este estudio y experimentación ha permitido adquirir conocimientos clave de configuración de red en sistemas Linux, de los que se ha sacado provecho para optimizar el rendimiento del *Fast Roaming* y evitar conflictos y puntos de fallo entre distintos módulos del sistema.

La completa caracterización del funcionamiento del *Fast Roaming* permitió a posteriori cuantificar la validez de la red WiFi Mesh para una teleoperación basada en transmisión de vídeo. Para ello se calcularon métricas derivadas de la transmisión de información entre dispositivos dentro de la red, con uno de ellos moviéndose por una amplia zona para provocar situaciones de *roaming* y analizar si el proceso es lo suficientemente eficiente para evitar una degradación notable en aplicaciones con requisitos de tiempo real. Estudiando los resultados se llegó a la conclusión de que si bien el *Fast Roaming* presenta un avance en cuanto a la velocidad con la que se completa la transición entre puntos de acceso, es inevitable la pérdida de datos durante un tiempo aproximado de 300 ms, al menos, con el enfoque aplicado en este trabajo. Estos datos aportan información valiosa para investigaciones futuras, en las que se podrían analizar medidas preventivas o estudiar otras alternativas en caso de que la aplicación final no tolere pérdidas durante ese pequeño tiempo.

Índice general

Capítulo 1

Introducción

1.1. Motivación

La robótica móvil se ha establecido como una de las alternativas más importantes y prácticas en aplicaciones como la búsqueda y rescate, la inspección de edificios inseguros o la exploración de ambientes tóxicos [1]. En estas aplicaciones, incluso en el caso de robots con un alto grado de autonomía, se precisa de supervisión humana, que es llevada a cabo en forma de teleoperación. En situaciones críticas, los algoritmos para navegación autónoma pueden no ser capaces de determinar la mejor acción debido a la falta de datos o a escenarios imprevisibles. es por ello que la supervisión humana sigue teniendo un papel importante. La Figura 1.1 ilustra los componentes básicos de la arquitectura de teleoperación, que se pueden agrupar en tres grandes bloques; i) el robot/sistema robotizado, ii) el canal de comunicaciones y iii) el operador/estación de usuario. Los comandos de control son enviados al robot remoto a través de un canal de comunicaciones y la información de los sensores, como vídeo procedente de una cámara o nubes de puntos de un sensor LiDAR (*Light Detection and Ranging*), son enviados de vuelta por el mismo canal. La teleoperación de robots en los entornos mencionados requiere satisfacer unos requisitos mínimos en cuanto a latencia, ancho de banda, *jitter* (variación de la latencia) y pérdidas de paquetes, entre otros, según la aplicación robótica [2].

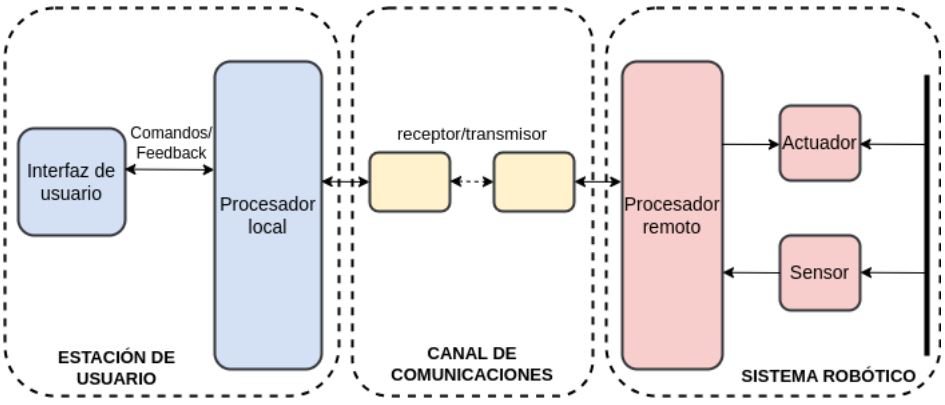


Figura 1.1: Diagrama de bloques de la arquitectura de teleoperación.

En general, las tecnologías inalámbricas usadas para teleoperación comprenden las tecnologías celulares (5G, 4G, 3G, 2.5G) inviables en nuestro ámbito de aplicación, y no celulares

(Satélite, WiFi, Bluetooth, Zigbee) [3]. Entre las que podrían utilizarse en entornos confinados, se ha considerado para este proyecto una red WiFi Mesh debido a la posibilidad de obtener una mayor cobertura y ancho de banda frente a las otras tecnologías. Estas redes poseen una naturaleza distribuida, como se puede observar en la Figura 1.2, con capacidad de reconfiguración sin intervención humana ante la posible caída de uno de sus nodos y también permiten a un cliente de la red transicionar rápidamente entre puntos de acceso, proceso conocido como *Fast Roaming*, cuando la intensidad de señal recibida en el cliente comienza a caer por debajo de cierto umbral. La implementación de ciertos protocolos para el *Fast Roaming* presenta una mejora notable frente al *roaming* convencional, donde era habitual la pérdida de datos por la fuerte degradación de la señal o por tiempos demasiado largos sin asociación a un punto de acceso.

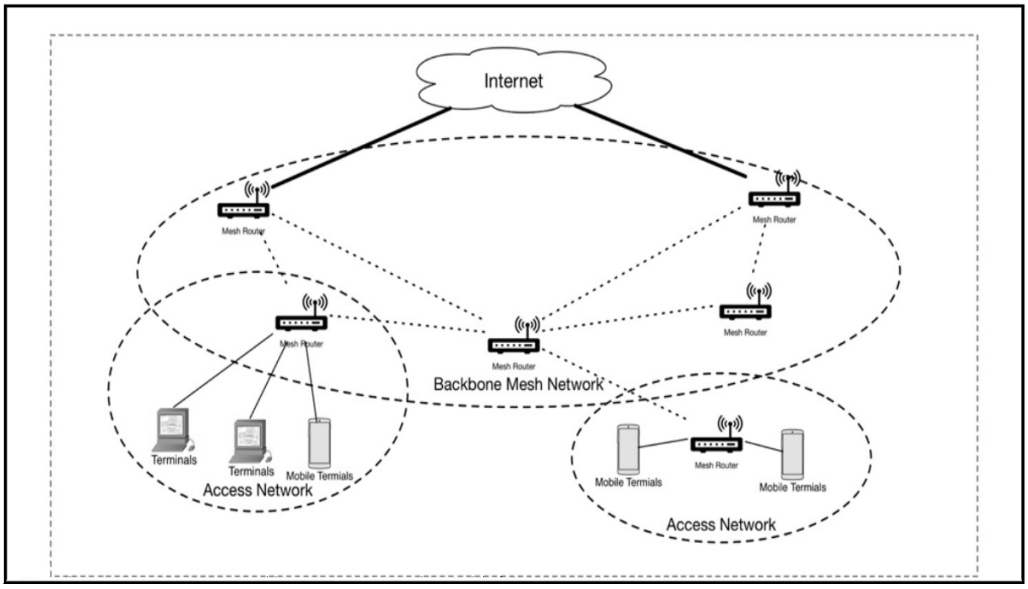


Figura 1.2: Red inalámbrica Mesh con puntos de acceso (routers) y clientes (fijos o móviles) conectados a ellos. Fuente: [4]

Por tanto, la motivación principal de este trabajo es caracterizar la validez de la tecnología WiFi Mesh para una teleoperación robótica basada en vídeo, poniendo especial énfasis en lo que ocurre durante el *Fast Roaming*, comprendiendo los protocolos que lo hacen posible y estudiando cómo evolucionan las métricas de prestaciones de red durante este proceso.

El proyecto surge como continuación al trabajo final realizado en la asignatura Sistemas Empotrados y Ubicuos del Máster Universitario en Ingeniería Informática, en el que se tuvo una primera toma de contacto con estas tecnologías a través de un cliente con cámara que ofrecía un pequeño streaming de vídeo mientras se desplazaba por una red WiFi Mesh comercial con tres puntos de acceso. En dicho trabajo surgieron algunos problemas respecto al cambio de punto de acceso, pues en numerosas situaciones la señal se degradaba hasta niveles que hacían inviables la transmisión de vídeo durante un tiempo considerable. En el ámbito de la asignatura no dio tiempo a comprender las técnicas y procesos necesarios para que se produjese un cambio de punto de acceso eficiente, y por ello nace este proyecto. El trabajo propuesto forma parte de los objetivos de mejora del alcance de la comunicación y el ancho de banda en entornos subterráneos no estructurados del proyecto ROBOCOMPLEX (PID2022-139615OB-I00) del

Robotics, Computer Vision and Artificial Intelligence group (Ropert) de la Universidad de Zaragoza.

1.2. Objetivos

El alcance del trabajo incluye:

- Estudio de la tecnología WiFi Mesh y del proceso de *Fast Roaming*.
- Configuración de sistemas GNU/Linux para clientes WiFi Mesh y análisis de eventos del sistema.
- Análisis del tráfico de red para la caracterización de los protocolos de *Fast Roaming*.
- Pruebas experimentales. Medidas de los parámetros que más afectan a la teleoperación y análisis de resultados.

1.3. Estructura de la memoria

El Capítulo 2 introduce los conceptos teóricos de la tecnología WiFi Mesh y el *Fast Roaming*, profundizando especialmente en los protocolos que lo hacen posible. El Capítulo 3 introduce la metodología que se ha llevado a cabo y los resultados obtenidos para caracterizar experimentalmente cómo se produce el *Fast Roaming* en los dispositivos de este trabajo. El Capítulo 4 presenta la metodología que se ha seguido y los resultados obtenidos al medir el desempeño del *Fast Roaming*, poniendo especial énfasis en su impacto en comunicaciones robóticas con restricciones de tiempo real, valorando si el rendimiento obtenido es suficiente comparado con los requisitos mínimos que deben cumplir este tipo de aplicaciones. El Capítulo 5 contiene las conclusiones y posible trabajo futuro.

Capítulo 2

Tecnología WiFi Mesh y Fast Roaming

2.1. WiFi Mesh

WiFi (*Wireless Fidelity*) es el nombre comercial propiedad de la organización *Wi-Fi Alliance* para designar la familia de protocolos de comunicación inalámbrica basados en el estándar 802.11 del IEEE (*Institute of Electrical and Electronics Engineers*). El estándar 802.11 especifica el conjunto de protocolos de control de acceso al medio y de la capa física para implementar la comunicación informática en redes de área local inalámbricas. Este estándar es parte del conjunto de estándares IEEE 802 que definen todos los protocolos para redes de área local, entre los que se encuentra también el IEEE 802.3 o *Ethernet* (red de área local cableada).

El IEEE denomina al estándar en vigor por 802.11 seguido de la fecha de publicación, siendo el 802.11-2020 la versión publicada en el momento en el que se desarrolla este trabajo y que reemplaza a las versiones anteriores. El estándar se puede actualizar mediante enmiendas, que son creadas por distintos grupos de trabajo dentro del IEEE. Tanto el grupo de trabajo como su documento final se describen con 802.11 seguido de una o dos letras minúsculas, por ejemplo, 802.11a o 802.11ax. La nomenclatura WiFi 5 o WiFi 6 es simplemente el nombre con el que la *Wi-Fi Alliance* comercializa los dispositivos compatibles con los distintos estándares del IEEE. La Figura 2.1 ilustra las normas del IEEE junto al nombre comercial escogido por la *Wi-Fi Alliance*. Por ejemplo, WiFi 6 designa a los dispositivos compatibles con la norma 802.11ax del IEEE, publicada en 2020.

Rel. Year	1999	2007	2009	2013	2020	2023(?)
Freq. Band	2.4 GHz	2.4 GHz	2.4 + 5 GHz	5 GHz	2.4 + 5 + 6 GHz (6E)	2.4 + 5 + 6 GHz
Bandwidth	20 MHz	20 MHz	40 MHz	80 MHz, 160 MHz	80 MHz, 160 MHz	240 MHz, 320 MHz

Figura 2.1: Avance histórico del estándar 802.11. Fuente: https://forum.huawei.com/enterprise/en/wi-fi-7-vs-other-wi-fi-standards/thread/703630656388612096-667213855346012160 (accedido: 28/11/2024)

Una red WiFi Mesh es una red formada por puntos de acceso WiFi organizados en topología de malla y con una interconexión entre ellos que presenta ciertas ventajas frente a la topología convencional de estrella. La naturaleza distribuida y descentralizada de las redes Mesh las hace apropiadas para aplicaciones robóticas o para aplicaciones IoT (*Internet of Things*) cuando es necesario desplegar redes con un amplio rango de cobertura que evitan tener un único punto de fallo. La principal diferencia entre las redes Mesh y las redes en estrella es la habilidad de las redes Mesh de autoorganizarse y autoconfigurarse dinámicamente, estableciendo la conectividad entre nodos de manera automática, mientras que la topología en estrella convencional, por definición, implica que todos los nodos están conectados a un único punto central, el cual se conecta al nivel superior de la red. La Figura 2.2 ilustra la topología de ambas redes.

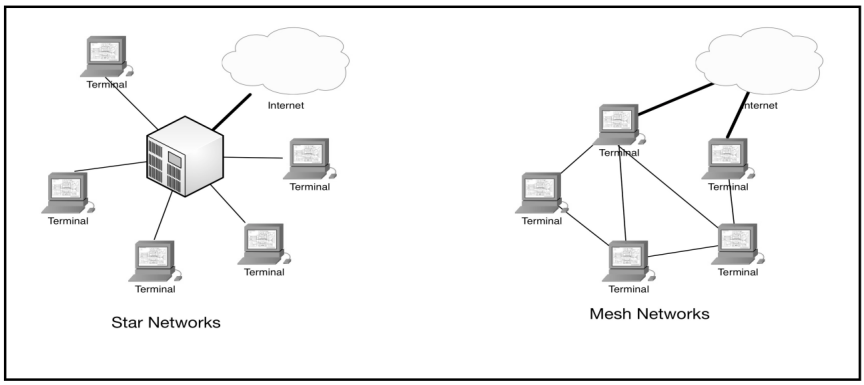
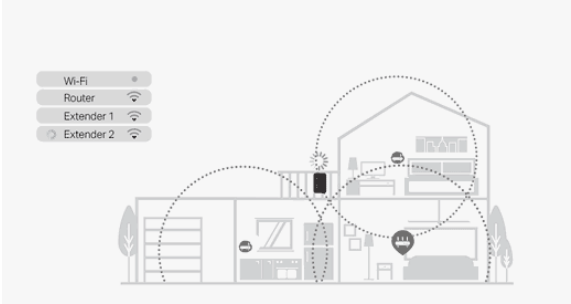


Figura 2.2: Topología en estrella y topología en malla. Fuente: [4]

Como ya se ha introducido en la Figura 1.2, las redes inalámbricas Mesh están formadas por nodos Mesh (routers) que forman una infraestructura inalámbrica a la que los clientes pueden conectarse. Los clientes dentro del área de un nodo Mesh crean una topología en estrella con este mientras que los nodos mesh forman una malla de enlaces autoconfigurables y autorreparables entre ellos. Esto significa que una vez que la red ha sido configurada por primera vez, añadir un nuevo nodo Mesh para ampliar la cobertura es una tarea sencilla, pues basta con colocar el nuevo dispositivo directamente en el área dentro del alcance de la red existente. Al tratarse de routers, los nodos Mesh también cuentan con la funcionalidad de puerta de enlace (*gateway*), por lo que es posible conectar cualquiera de ellos a Internet para proporcionar a la red conectividad con el exterior.

Los sistemas comerciales WiFi Mesh han ganado popularidad en los últimos años y existen gamas a distintos precios que se ajustan a las necesidades concretas de cada usuario. Estos sistemas buscan principalmente eliminar zonas muertas de WiFi allá donde no llega el router principal. Tradicionalmente, este problema se ha solucionado con extensores de rango, pero existen algunas diferencias clave entre ambos sistemas. La Figura 2.3 ilustra una posible solución al problema de las zonas muertas de WiFi con el sistema tradicional de extensores de rango. La primera contraindicación de este sistema, es que la mayoría de los extensores de rango levantan una nueva red WiFi, con un SSID (*Service Set Identifier*, nombre que identifica a la red WiFi) y una contraseña distintos e independientes a los de la red WiFi que levanta el router principal. Como se puede observar en la Figura 2.3a, en este caso el sistema está compuesto por el router principal y dos extensores, desencadenando en tres redes WiFi. Si se quiere ampliar la cobertura, manualmente hay que configurar una nueva red WiFi que el cliente debe aprender y recordar.

(a) Situación inicial. Cliente conectado al router principal.



(b) Situación intermedia, proceso de *roaming*. Cambio entre redes.

(c) Situación final. Cliente conectado a un extensor.

Figura 2.3: Zona residencial cubierta con un router principal y dos extensores de rango (tres redes WiFi en total). Se ilustra el proceso de *roaming*. Fuente: https://www.tp-link.com/en/mesh-wifi/ (accedido: 28/11/2024)

La segunda contraindicación es consecuencia de la primera, y es que cuando el cliente comienza a moverse, alejándose del router principal y entrando en el rango de un extensor, como se muestra en las Figuras 2.3b y 2.3c, el hecho de tener que cambiar entre redes hace que este proceso de *roaming* sea lento e ineficiente, con una desconexión total de la red por un intervalo de tiempo que no es despreciable.

Figura 2.4: Zona residencial cubierta con tecnología WiFi Mesh (una única red WiFi). Fuente: https://www.tp-link.com/en/mesh-wifi/ (accedido: 28/11/2024)

La Figura 2.4 ilustra la misma zona residencial cubierta con tecnología WiFi Mesh en lugar

de extensores de rango. Se puede observar que en este caso hay una única red WiFi, por lo que no es necesario realizar ninguna configuración específica en el cliente para añadir nuevos nodos, y el proceso de *roaming* no implica desconectarse para conectar a la nueva red. Los nodos Mesh se comunican entre ellos formando una red WiFi con un único SSID. Por último, los extensores de rango tradicionales simplemente amplifican la señal del router por el mismo canal, y esto significa que el extensor tiene que alternar entre recibir datos del router y transmitirlos al cliente (y viceversa), dividiendo por tanto por 2 el ancho de banda disponible en cada salto, como se puede ver en la Figura 2.5 (*half-duplex*: por el canal se puede enviar o recibir, pero no ambos a la vez). Sin embargo, los nodos Mesh generan su propia señal y tienen la capacidad de gestionar la red de manera eficiente, con canales dedicados para comunicarse con el router y canales dedicados para la comunicación con el cliente (*full-duplex*: una comunicación en la que se puede enviar y recibir al mismo tiempo). También tienen la capacidad de establecer nuevas rutas de encaminamiento si alguna de las existentes se encuentra congestionada.

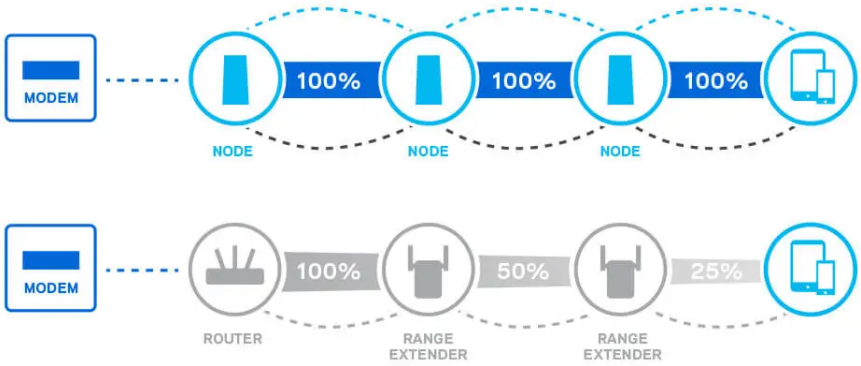


Figura 2.5: Ilustración gráfica de cómo un extensor de rango tradicional divide por 2 el ancho de banda original que provee el router (parte inferior de la figura). Con los nodos Mesh esto no ocurre, ya que utilizan canales dedicados para router y para clientes (parte superior de la figura). Fuente: https://datafeature.com/wifi-extender-vs-mesh-wifi/ (accedido: 28/11/2024)

La forma en la que los nodos de una red WiFi Mesh se comunican entre ellos para gestionar las rutas de encaminamiento, levantar una red con un único SSID, la adición de nuevos nodos, etc., se establece en la enmienda 802.11s, emitida en 2011 y absorbida en 2012 por el estándar 802.11-2012.

2.2. Marco teórico Fast Roaming

Fast Roaming es el término inglés acuñado para referirse al *roaming* que ocurre siguiendo unos protocolos definidos por el IEEE. Estos protocolos están recogidos en los estándares 802.11k [5], 802.11v [6] y 802.11r [7], en los que se profundizará más adelante.

2.2.1. Qué es el roaming

En primer lugar, es conveniente comprender qué es el *roaming*. Se puede explicar con la experiencia que se tiene cuando alguien conectado con un móvil a la red WiFi de un hotel o

una universidad se mueve a lo largo de pasillos y salas, donde normalmente hay varios puntos de acceso WiFi (*Access Point*, AP) en techos o paredes, visibles u ocultos. Cada uno de estos APs proporciona cobertura a una zona determinada. Para moverse por toda la zona sin perder la cobertura WiFi, es necesario que el móvil pueda ir saltando entre los APs y para ello debe detectar que recibe una señal débil, desconectarse y conectarse al siguiente AP con mejor señal. Esto es lo que se conoce como *roaming* o itinerancia en redes WiFi empresariales y es un proceso transparente para el usuario.

En todo momento, es el cliente quien decide cuándo cambiar de AP. Por ello, la eficiencia del *roaming* depende en última instancia de la implementación en el cliente, que decide cuándo cambiar entre los distintos estados del diagrama de estados del estándar 802.11, y por tanto, terminar asociándose con un AP distinto al que ya estaba. El diagrama de estados del 802.11 y el tipo de tramas que se pueden enviar en cada estado [8] se pueden consultar en el Anexo A. Al tratarse de una decisión que depende de la implementación en el cliente, se pueden encontrar dispositivos que únicamente cambian de AP cuando la señal es realmente débil, con previa pérdida de datos si la señal ya lleva un rato sin ser suficientemente buena, o dispositivos en los que el proceso de *roaming* es lento, si tienen que escanear el espectro para encontrar nuevos APs, produciéndose microcortes en la conexión.

2.2.1.1. Factores de decisión en el cliente

Una vez comprendido qué es el *roaming*, es importante conocer los diversos métodos que utilizan los clientes para decidir cambiar de AP. Los factores varían considerablemente entre los dispositivos cliente, pero los más comunes son cuatro [10]:

1. Tramas *Beacon* perdidas: en algunos clientes, cuando se pierde un número específico de este tipo de tramas (ver Anexo A para comprender los distintos tipos de tramas del estándar 802.11), se activa el proceso de *roaming*.

2. Intensidad de señal: la intensidad de señal recibida en el cliente, explicada en detalle en la Sección 3.1.1, se utiliza como un umbral para el *roaming*. Cuando la intensidad cae bajo cierto umbral, se activa el proceso de *roaming*.

3. Tramas reintentadas: los reintentos de tramas pueden hacer que un cliente realice el *roaming* hacia otro AP cuando existe un problema de nodo oculto. Este problema se presenta cuando un nodo de la red no puede "escuchar" a otro nodo, y ambos intentan comunicarse con un tercer nodo, lo que puede generar colisiones y pérdidas de datos. Se trata de una situación en la que dos nodos están fuera del alcance de la señal de cada uno, pero ambos pueden interferir con un nodo central o intermediario que sí está al alcance de ambos. Esto puede provocar corrupción de tramas en el AP debido a interferencias, lo que resulta en reintentos en los clientes. Si estos incluyen un umbral de reintentos de tramas, se puede activar el proceso de *roaming*.

4. Histéresis: se refiere a la diferencia en la intensidad de señal entre un AP objetivo potencial y el AP actualmente asociado. Si los clientes implementan este factor, al cumplirse una histéresis mínima se activará el proceso de *roaming*.

2.2.2. Fast Roaming: protocolos

Los estándares 802.11k, 802.11v y 802.11r del IEEE, establecen cada uno de ellos protocolos para mejorar diversas funcionalidades de la red, y si los dispositivos que implementan estos protocolos saben aprovecharse de los beneficios que aportan cuando se utilizan coordinadamente, es cuando se puede conseguir el *Fast Roaming* con el entendimiento entre clientes y APs. La Figura 2.6 muestra la interrelación entre los tres estándares y el papel que juegan antes y durante el proceso de *roaming*. Aunque el estándar 802.11r es uno de los tres que aporta al *Fast Roaming*, se queda fuera del alcance de este proyecto debido a que hace énfasis en reducir el tiempo de autenticación en redes WiFi protegidas con contraseña y en el ámbito de de este proyecto las aplicaciones se levantarán en entornos controlados que no requieren protección.

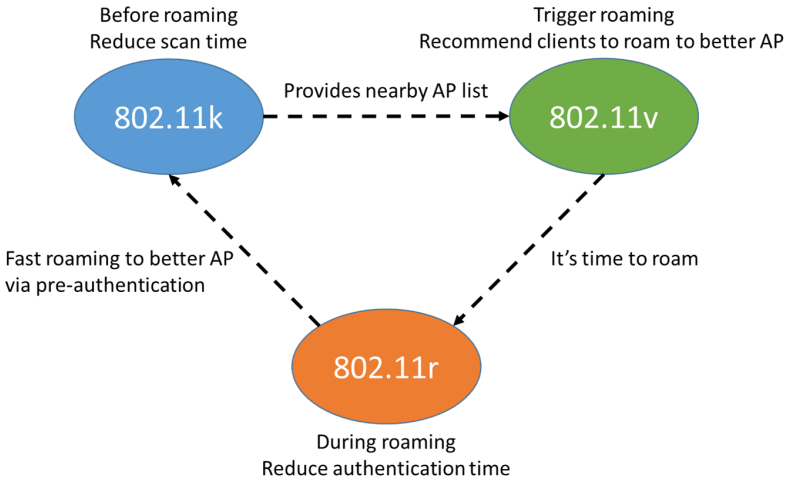


Figura 2.6: Interrelación entre los tres estándares del IEEE que favorecen el *Fast Roaming*. 802.11k permite conocer la situación de otros APs, 802.11v permite recomendar a un cliente que debe comenzar el proceso de *roaming* y 802.11r permite reducir el tiempo de autenticación contra un nuevo AP dentro de la misma red. Fuente: https://community.tp-link.com/en/home/forum/topic/180194 (accedido: 28/11/2024)

A continuación se profundiza en el marco teórico de los estándares 802.11k y 802.11v, que sí que se analizarán en los experimentos prácticos del proyecto.

2.2.2.1. 802.11k: Medición del espectro radioeléctrico en WLANs

El estándar 802.11k proporciona mecanismos para la medición del espectro radioeléctrico en WLANs (*Wireless Local Area Network*), de manera que tanto clientes como APs comprenden el entorno radioeléctrico en el que se encuentran. Esto significa, por ejemplo, que un cliente puede ser capaz de obtener una lista de los posibles APs a los que puede asociarse antes de que la señal con su AP actual comience a degradarse.

La manera de anunciar qué mecanismos del estándar implementa un dispositivo dentro de la WLAN depende de si este actúa como cliente o AP. Si actúa como cliente, anunciará los mecanismos que implementa en las tramas *Association Request* o *Authentication Request*.

Si se trata de un AP, este anunciará los mecanismos en la trama *Beacon*. A continuación se exponen los mecanismos del estándar más relevantes para el *Fast Roaming*.

2.2.2.1.1 Beacon Report

El mecanismo *Beacon Report* permite a un dispositivo de la red solicitar a otro dispositivo una lista de los APs con los que este podría asociarse en un canal o canales determinados. La estación que solicita la medición, envía una trama del tipo *Radio Measurement Request* a la estación objetivo, indicando en el subtipo de la trama que se trata de un *Beacon Report*. La estación objetivo que recibe la solicitud, envía una trama *Probe Request* en modo *broadcast* a todas las estaciones bajo el mismo SSID. Los APs que reciban esta petición, responderán con una trama *Probe Response* y la estación objetivo almacenará los resultados, formando una lista con la identificación y el nivel de señal de cada AP que ha respondido, para enviársela en una trama *Radio Measurement Response* al AP que comenzó la solicitud de medición en primera instancia. De esta manera el AP es consciente de la situación radioeléctrica en la que se encuentra este cliente, pudiendo valorar si existen mejores APs para él. En la Figura 2.7 se puede observar el diagrama de secuencia de este proceso.

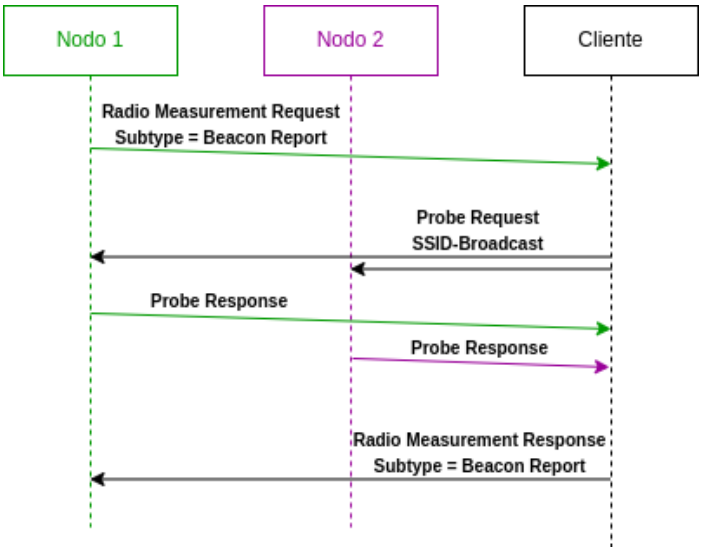


Figura 2.7: Diagrama de secuencia de la medición tipo *Beacon Report*. El AP que comienza la petición es el nodo 1, que sería el nodo al que el cliente está asociado.

2.2.2.1.2 Neighbor Report

En este mecanismo, es el cliente quien solicita a su AP actual información sobre sus APs vecinos. El cliente envía una trama del tipo *Radio Measurement Request* a su AP, indicando en el subtipo de la trama que se trata de un *Neighbor Report*. El AP contesta con una trama del tipo *Radio Measurement Response* y subtipo *Neighbor Report*, con información sobre los APs vecinos conocidos que son candidatos para comenzar el *roaming*. Este par de tramas permiten a un cliente obtener información sobre los vecinos del AP asociado para utilizarlos como candidatos sin la necesidad de que sea él quien tenga que medir el espacio radioeléctrico. En el estándar se establece que el mecanismo por el cuál el AP obtiene información sobre sus APs vecinos queda fuera del alcance del mismo, por lo que puede recogerse información con

informes de medición previos recibidos de otros clientes, información recibida por una interfaz de gestión, etc. La Figura 2.8 recoge el diagrama de secuencia de este proceso.

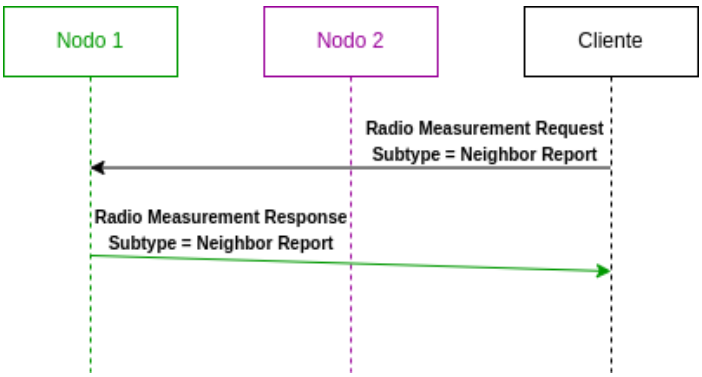


Figura 2.8: Diagrama de secuencia de la medición tipo *Neighbor Report*. El cliente es quien comienza la petición, que se encuentra asociado al nodo 1. El mecanismo por el cual el nodo 1 obtiene información sobre sus APs vecinos queda fuera del estándar.

2.2.2.2. 802.11v: Gestión de redes inalámbricas

El estándar 802.11v define mecanismos y servicios para la gestión inalámbrica de la red entre los distintos dispositivos de la misma. Los dispositivos intercambian información con el propósito de mejorar el rendimiento general de la red, de manera que estos pueden solicitar o aconsejar ciertas acciones a otros dispositivos para evitar la presencia de interferencias o ajustar ciertos parámetros basándose en las condiciones de la red.

2.2.2.2.1 BSS Transition Management

Para el propósito de un *roaming* rápido y efectivo, el servicio más interesante que proporciona este estándar es el llamado *BSS Transition Management*. BSS (*Basic Service Set*) se refiere a la cobertura y servicio que aporta un único AP dentro de la red. Si se está hablando de una red inalámbrica formada por varios APs, cada uno de ellos representa un BSS, y el conjunto de todos ellos forma un *Extended Service Set* (ESS). Una red WiFi Mesh es un ESS que está formada por varios BSS (nodos Mesh). *BSS Transition Management* permite que un AP envíe a uno de sus clientes asociados una solicitud de transición a otro AP específico o a un conjunto de APs si hay varios candidatos óptimos. La decisión de cambiar o no de AP es en última instancia del cliente. Por ello, el cliente recibe una trama del tipo *BSS Transition Management Request* enviada por su AP actual con un AP candidato que podría presentar un mejor nivel de señal, y el cliente responde con una trama del tipo *BSS Transition Management Response*, en la que indica si acepta la solicitud, en cuyo caso comenzará el proceso de *roaming*, o si por el contrario la rechaza, rehusando así tomar acción. La figura 2.9 ilustra las tramas intercambiadas en este servicio.

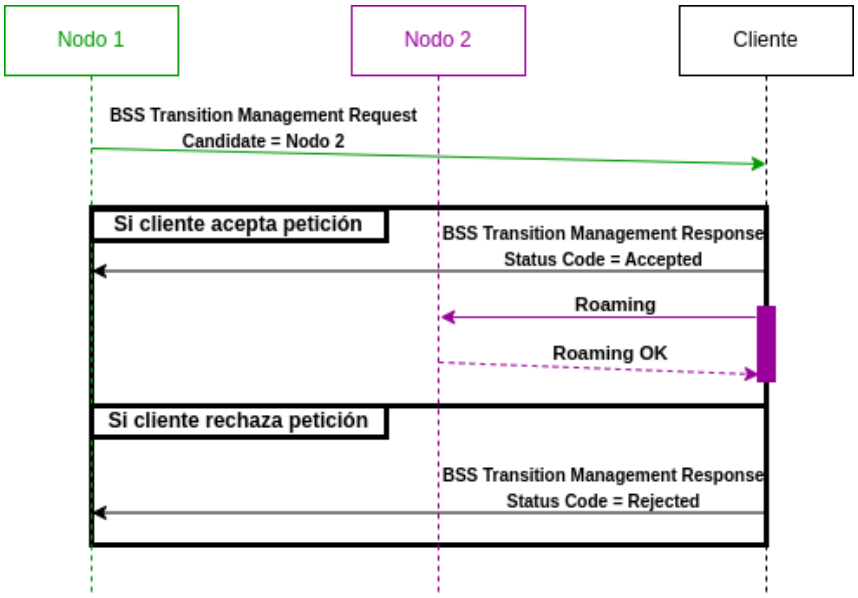


Figura 2.9: Diagrama de secuencia del mecanismo *BSS Transition Management*. El cliente se encuentra asociado al nodo 1, y este le aconseja cambiar al nodo 2. El cliente puede aceptar la petición, en cuyo caso realiza el *roaming* contra el nodo 2, o rechazar la petición y no tomar acción. Las tramas de gestión involucradas en el proceso de *roaming* se han excluido para facilitar la legibilidad.

Capítulo 3

Análisis experimental del comportamiento del Fast Roaming

El presente trabajo cuenta con dos experimentos troncales. El primero de ellos, introducido en este capítulo, pretende caracterizar experimentalmente la teoría y protocolos que se han introducido hasta ahora para comprender cómo funciona el *Fast Roaming*. El segundo, introducido en el Capítulo 4, pretende medir el desempeño del *Fast Roaming* una vez comprendido su funcionamiento. La Tabla 3.1 muestra el hardware utilizado en esta primera fase.

Nombre	**Descripción**
Tp-Link AC1300 Deco M5[1](x3)	Los 3 nodos WiFi Mesh que actuarán como AP a lo largo de este proyecto.
Lenovo ThinkPad E14 Gen4	Dispositivo móvil utilizado en esta fase. Sistema operativo: Ubuntu 22.04.1 (*kernel* 6.8.0-49-generic).
Intel Wi-Fi 6 AX201[2](**Intel**)	Tarjeta de red WiFi integrada en el Lenovo.
Alfa AWUS036ACHM[3](**Alfa1**)	Tarjeta de red WiFi externa.
Alfa AWUS036AXML[4](**Alfa2**)	Tarjeta de red WiFi externa.

Tabla 3.1: Hardware utilizado en la primera fase del trabajo.

Tanto los nodos Mesh como las dos tarjetas Alfa fueron adquiridos por el grupo de investigación de la directora de este trabajo. La guía de usuario[5]de los nodos indica que los clientes necesitan implementar los protocolos 802.11k/v para que el *Fast Roaming* funcione correctamente. En las especificaciones técnicas de las tarjetas Alfa[3,4] no se ha encontrado evidencia de que implementen los protocolos, por lo que habrá que descubrirlo experimentalmente. El Lenovo es propiedad del autor de este trabajo y también se aprovecha la tarjeta Intel que incorpora para caracterizar el *Fast Roaming*. Sus especificaciones técnicas[2] (ver sección *"IEEE WLAN Standard"* tras descargar información complementaria) establecen que implementa los protocolos 802.11k/v. De aquí en adelante se hará referencia a las tarjetas por **Intel**, **Alfa1**

[1]https://www.tp-link.com/es/home-networking/deco/deco-m5/v3.20/ (accedido: 28/11/2024)

[2]https://www.intel.la/content/www/xl/es/products/sku/130293/intel-wifi-6-ax201-gig/specifications.html (accedido: 28/11/2024)

[3]https://www.alfa.com.tw/products/awus036achm?variant=39477247082568 (accedido: 28/11/2024)

[4]https://www.alfa.com.tw/products/awus036axml?variant=39754360684616 (accedido: 28/11/2024)

[5]https://www.tp-link.com/es/support/download/deco-m5/ (accedido: 28/11/2024)

y **Alfa2**, tal y como se indica en la Tabla 3.1. Para que el Lenovo reconociera correctamente la tarjeta Alfa2, fue necesario actualizar el *kernel* y el sistema operativo del mismo, pues esta tarjeta requería una versión de *kernel* superior a la que incorporaba. El Anexo D recoge las dificultades encontradas durante este proceso, así como las versiones de *drivers* y *firmware* para cada una de las tarjetas.

3.1. Métricas

3.1.1. RSSI

La RSSI (*Received Signal Strength Indicator*) es una de las métricas principales utilizadas para evaluar la calidad de un enlace inalámbrico. Otras métricas comunes son SINR (*Signal-to-Interference-plus-Noise Ratio*), PDR (*Packet-Delivery Ratio*) y BER (*Bit-Error Rate*) [9], pero por el momento en el ámbito de este experimento se ha escogido trabajar con la RSSI para centrarse únicamente en el nivel sobre el que ocurre el *roaming*. Esta métrica representa la intensidad de señal observada en la antena del receptor durante la recepción de paquetes. Se calcula como una relación logarítmica con respecto al valor de referencia de 1 mW (ver Ecuación 3.1).

$$P(dBm) = 10 \cdot log_{10} \frac{P_{mW}}{1\ mW} \tag{3.1}$$

Valores superiores a -50 dBm representan una intensidad de señal excelente. Entre -50 y -60 dBm, la intensidad de la señal es buena. De -60 a -70 dBm, la intensidad de la señal se considera regular, y para valores menores a -70 dBm, se considera débil.

3.1.2. Tasa de transmisión

La tasa de transmisión, también conocida como *bitrate*, es una medida utilizada en telecomunicaciones, informática y procesamiento de señales para describir la cantidad de datos transmitidos o procesados en un período de tiempo determinado. Se expresa generalmente en bits por segundo (bps) y sus múltiplos como kilobits por segundo (kbps), Megabits por segundo (Mbps) o Gigabits por segundo (Gbps).

3.2. Configuración del experimento

Como punto de partida, se levantó la red WiFi emplazando dos de los tres nodos tal y como se muestra en la Figura 3.1. Los objetivos de este primer experimento son los siguientes:

- Verificar si el cliente ejecuta el *roaming* sin realizar aún ningún tipo de configuración específica, para establecer el punto del que se parte.
- Medir la RSSI durante el trayecto, para comprender sobre qué nivel de intensidad de señal se realiza el *roaming* y si existe algún patrón de los ya mencionados en la Sección 2.2.1.1.
- Transmitir datos a una estación base colocada junto al nodo 1, para medir la fluctuación en la tasa de transmisión durante el trayecto.

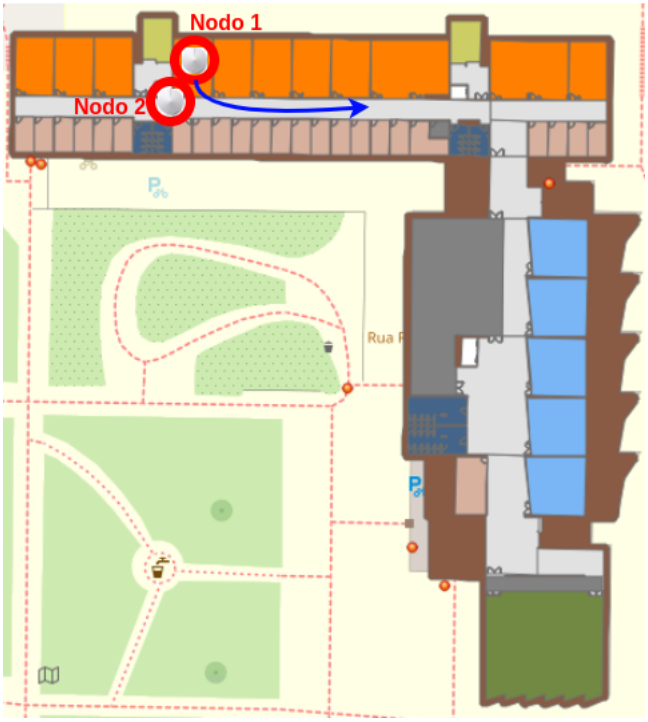


Figura 3.1: Despliegue de nodos en el primer experimento[6]. Se trata de la planta segunda del edificio Ada Byron. El nodo 1 está dentro del laboratorio 2.08 y el nodo 2 en el pasillo, junto a la puerta del laboratorio. La flecha azul indica la trayectoria del recorrido de la estación móvil.

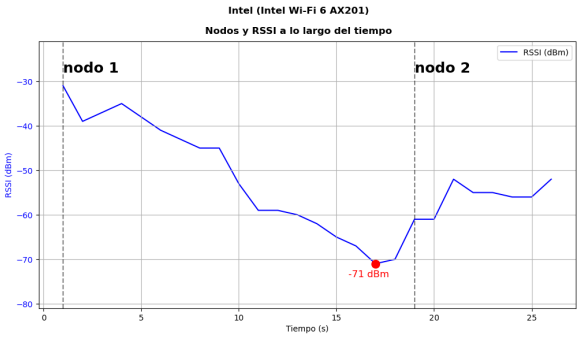
El experimento consiste en conectarse a la red junto al nodo 1 y comenzar a andar alejándose de este, para salir del laboratorio y comenzar a andar por el pasillo, provocando el *roaming* al nodo 2. Se realizaron varias iteraciones del recorrido con las tres tarjetas de red mencionadas en la Tabla 3.1 para ver si se observaba repetitividad en el comportamiento de cada una de ellas.

3.3. Análisis de resultados

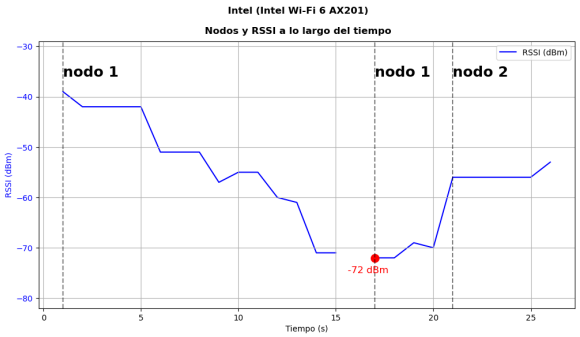
En la Figura 3.2 se muestran los resultados de tres repeticiones representativas del experimento para la tarjeta Intel. Las gráficas muestran en azul la evolución de la RSSI a lo largo del tiempo (con un punto rojo destacando la peor RSSI registrada). Las líneas discontinuas verticales muestran un cambio de AP, esto es, el *roaming*. La Figura 3.2a muestra lo que sería un comportamiento óptimo, produciéndose el *roaming* una vez se han alcanzado los -70 dBm aproximadamente. Las Figuras 3.2b y 3.2c muestran un comportamiento un tanto más extraño, el cual no puede considerarse como un *roaming* óptimo. En la Figura 3.2b se puede observar que, tras llegar aproximadamente a los -70 dBm, se da durante un par de segundos una pérdida de conexión total con la red (no hay registro de RSSI). Cuando la conexión vuelve, es importante destacar que se realiza contra el nodo 1 en lugar del nodo 2, cuando este último ofrece realmente mejor RSSI, en torno a los -55 dBm, como se puede observar finalmente cuando sí que se produce el *roaming*. Por último, en la Figura 3.2c el *roaming* es aún más insatisfactorio, pues en este caso se llega a los -85 dBm y se produce también un corte de

[6]https://sigeuz.unizar.es/ (accedido: 28/11/2024)

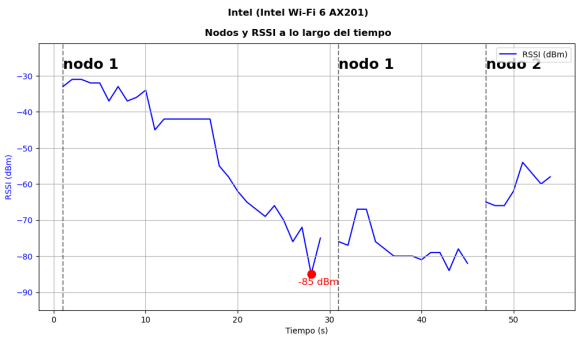
red, volviendo a conectar con el nodo 1 por un tiempo considerablemente más largo que en la Figura 3.2b y con un valor débil de RSSI, para finalmente producirse el *roaming* hacia el nodo 2.



(a) Roaming satisfactorio al nodo 2.



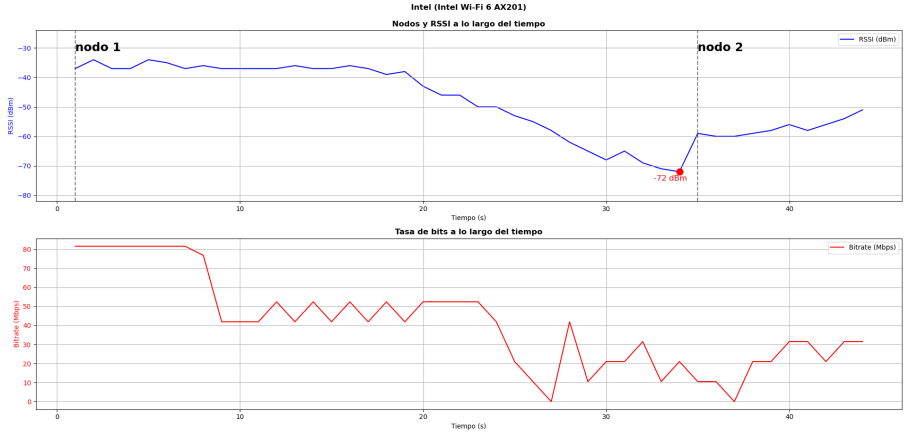
(b) Roaming fallido, reconectando al nodo 1.



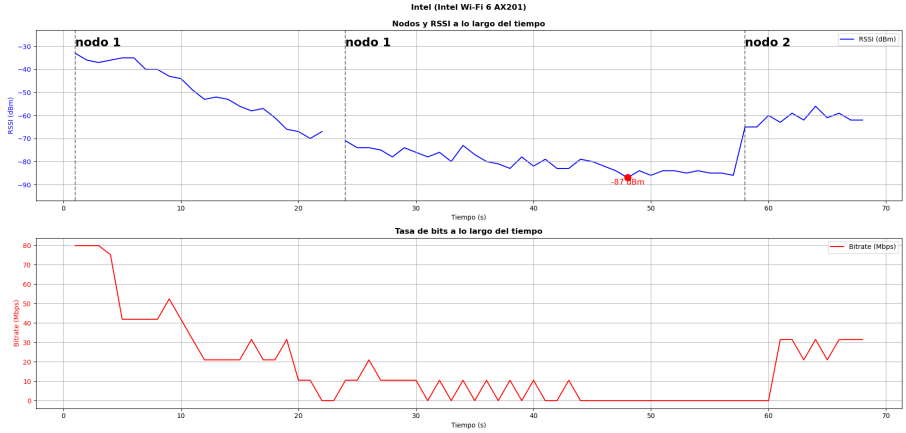
(c) Roaming fallido, reconectando al nodo 1 por un mayor intervalo de tiempo.

Figura 3.2: Gráficas representativas del primer experimento para la tarjeta Intel. Se muestra la evolución de la RSSI a lo largo del tiempo conforme se avanza por el trayecto del experimento.

La Figura 3.3 muestra dos repeticiones del experimento para la tarjeta Intel añadiendo una transmisión de datos con la herramienta `iPerf3` (ver el Anexo B para consultar la información relevante sobre esta herramienta). La transmisión consiste en el envío de paquetes desde el cliente móvil hacia la estación base por medio de una conexión TCP. Las gráficas muestran en rojo la evolución de la tasa de transmisión a lo largo del tiempo. La Figura 3.3a muestra un *roaming* satisfactorio, produciéndose en torno a los -70 dBm, tal y como muestra el registro de RSSI. En la gráfica inmediatamente inferior, se puede observar que la tasa de bits comienza a descender cuando la RSSI se encuentra en torno a los -60 dBm. Desde este momento hasta que se produce el *roaming*, la tasa de bits presenta subidas y bajadas, pudiendo dar a entender que para una transferencia de datos satisfactoria la RSSI debería mantenerse superior a los -60 dBm. La Figura 3.3b muestra un proceso de *roaming* deficiente, como se puede observar en el registro de la RSSI. Se vuelve a dar el fenómeno de desconexión de la red y reconexión contra el nodo menos óptimo, lo que tiene notoria implicación en el desempeño de la transmisión de datos. En la gráfica inferior se puede observar como la tasa de bits llega a 0 Mbps cuando se produce el corte de red, para posteriormente tener picos entre los 0 y 10 Mbps y finalmente cortarse totalmente la transmisión de datos durante unos 15 segundos hasta que el dispositivo realiza el *roaming* contra el nodo 2.



(a) Roaming tardío al nodo 2.



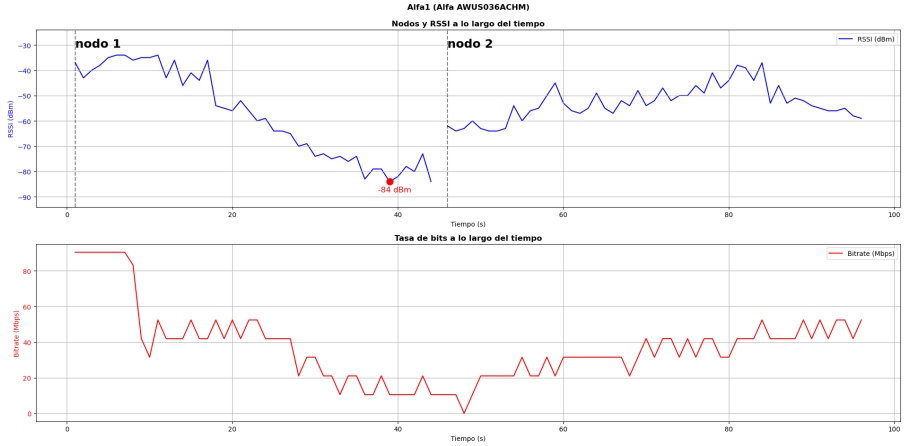
(b) Roaming fallido, con reconexión al nodo 1 por un largo tiempo.

Figura 3.3: Representaciones representativas del primer experimento para la tarjeta Intel. Se muestra la evolución de la RSSI y la tasa de bits a lo largo del tiempo conforme se anda por el trayecto del experimento.

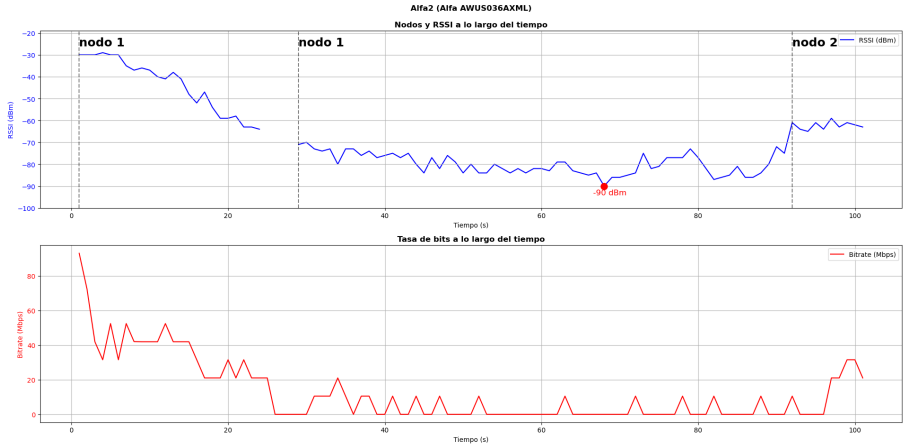
El comportamiento visto hasta ahora en las muestras del experimento con la tarjeta Intel, se repite también con las tarjetas Alfa1 y Alfa2, sin ninguna diferencia destacable. Ocasionalmente se produce un *roaming* óptimo entre repeticiones, pero es más frecuente un *roaming* ineficiente. La Figura 3.4 muestra una repetición con la tarjeta Alfa1 (Figura 3.4a) y otra con la tarjeta Alfa2 (Figura 3.4b). Para la Alfa1 se puede observar como se llega a un valor bastante débil de señal (en torno a -85 dBm), ocurre un pequeño corte a continuación y se conecta con nodo 2, mejorando RSSI y tasa de bits. Para la Alfa2, se puede observar como tras el corte de red, se reconecta con nodo 1, con unos valores de RSSI y tasa de bits muy bajos hasta que se produce el *roaming* al nodo 2. El resto de resultados de este experimento se pueden consultar en el repositorio público de GitHub del proyecto[7], dado que la información conjunta de todas las repeticiones ya se ha sintetizado en esta sección.

Tras analizar los resultados y tomando como referencia los objetivos planteados al inicio de la sección, se pueden establecer las siguientes conclusiones:

[7]https://github.com/gcr-UZ/tfm-mesh



(a) Roaming tardío al nodo 2, con pequeño corte de red.



(b) Roaming fallido, con reconexión al nodo 1 por un largo tiempo.

Figura 3.4: Representaciones representativas del primer experimento para las tarjetas Alfa1 y Alfa2. Se muestra la evolución de la RSSI y la tasa de bits a lo largo del tiempo conforme se anda por el trayecto del experimento.

- Como punto de partida, el *roaming* no funciona bien con ninguna de las tres tarjetas. Es necesario investigar si hacen falta configuraciones específicas en el sistema operativo y analizar qué ocurre, recurriendo a los *logs* del sistema y estudiando el intercambio de tramas entre las tarjetas de red y los APs.
- En la mayoría de las repeticiones, conforme el cliente se aleja del nodo 1 y el nivel de RSSI oscila aproximadamente entre los -70 y -75 dBm, ocurre un evento. Este evento puede ser:
  a) Un corte de red para reconectar con el nodo 1 por un largo tiempo (bastante común).
  b) Un corte de red para reconectar con el nodo 2 (poco común).
  c) Un *roaming* satisfactorio al nodo 2 (poco común).
- El hecho de transmitir datos, parece no influir a la hora de realizar el *roaming*. No se

han encontrado diferencias en la ocurrencia de los eventos citados en el punto anterior en función de si se transmitían datos con iPerf3 o no.

3.4. Configuración y depuración en sistemas Linux

Tras los insatisfactorios resultados obtenidos en el primer experimento, se profundizó en el estudio de las herramientas de configuración de red en sistemas Linux, enfocándose en Ubuntu y el ámbito inalámbrico. También se registraron y consultaron los *logs* del sistema en busca de anomalías o información valiosa para mejorar los resultados.

3.4.1. El Stack 802.11

La naturaleza open source de los sistemas Linux impulsa un amplio abanico de opciones para configurar sus interfaces de red. Para comprender completamente lo que ocurre en el sistema cuando se ejecutan determinadas acciones de configuración de red, se profundizó en los módulos del *stack* inalámbrico de Linux, empezando por las herramientas que cualquier administrador a nivel de usuario puede utilizar. En el momento en el que se desarrolla este trabajo, se puede establecer que fundamentalmente son dos las alternativas más populares: a) el paquete `ifupdown`, con cuyos comandos se pueden controlar las interfaces de red definidas en el fichero '`/etc/network/interfaces`' y b) el demonio `NetworkManager`.

En el fichero '`/etc/network/interfaces`' se pueden definir las interfaces de red del sistema, y su manipulación, ya sea directamente con editores de texto o indirectamente con los comandos `ifup` o `ifdown`, del paquete `ifupdown`, ha sido la manera tradicional de configurar nodos de red en sistemas Linux basados en Debian, como Ubuntu. A día de hoy, la literatura establece que estas herramientas están ya obsoletas, pero conviene conocerlas porque se siguen encontrando máquinas configuradas con ellas. La otra opción que se ha mencionado, el demonio `NetworkManager`, es una herramienta que trata de simplificar y automatizar la configuración de interfaces de red. En los sistemas Linux, un demonio es un proceso que se ejecuta en segundo plano y realiza diversas tareas esenciales para el sistema. Concretamente, `NetworkManager` es capaz de detectar las interfaces de red y automatizar la mayoría de tareas, como asignarles una dirección IP o escanear las redes inalámbricas disponibles, sustituyendo en gran medida la necesidad de una configuración manual. Ambas opciones, tanto '`/etc/network/interfaces`' como `NetworkManager` pueden coexistir en un mismo sistema, ya que `NetworkManager` ha sido desarrollado para que por defecto ignore las interfaces de red definidas explícitamente en el fichero '`/etc/network/interfaces`', para evitar conflictos. No obstante, lo habitual es decantarse por una única opción.

El Lenovo con el que se ha realizado el experimento, se basa en `NetworkManager` para la configuración de las interfaces de red. Respecto a interfaces inalámbricas, hay algunas tareas de configuración que el `NetworkManager` delega en otra herramienta llamada `wpa_supplicant`, la cual se encarga principalmente de aspectos de bajo nivel de seguridad inalámbrica como autenticación, cifrado y exploración de la red. En algunas ocasiones será el `NetworkManager` quien se comunicará directamente con el *kernel* de Linux para interactuar con el hardware, y en otras ocasiones tiene definidos en su propio código valores de configuración para el

`wpa_supplicant` que establecen cómo este debe abordar esas tareas específicas de las que no se encarga el `NetworkManager`.

La Figura 3.5 muestra un diagrama de los módulos principales del *stack* 802.11 de Linux y cómo se comunican entre ellos. Hasta ahora se ha hablado del espacio de usuario, con las herramientas `NetworkManager`, `wpa_supplicant` y otras que existen, como `hostapd`, que no interviene en este proyecto pero que se pueden utilizar. Para que todo sistema operativo pueda comunicarse con el dispositivo hardware por medio de conectores PCIe, USB, etc., es necesario que el sistema incorpore los *drivers* específicos de ese dispositivo. Esto es lo que refleja la parte más inferior del diagrama, la parte final del *stack* en la que el hardware recibe las instrucciones específicas gracias al *driver*. Lo que se puede ver entre medio, son los módulos del *kernel* de Linux que recogen las operaciones estándar que los desarrolladores de *drivers* utilizan para conseguir que una interfaz de red inalámbrica funcione en un sistema Linux. Se puede ver como en esta parte del diagrama hay dos bloques, SoftMAC y FullMAC.

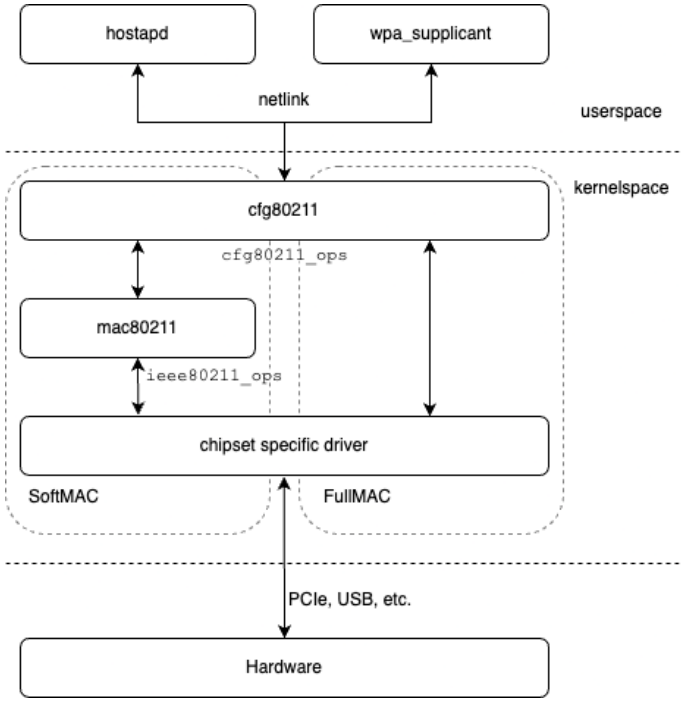


Figura 3.5: *Stack* de los módulos principales que intervienen en la implementación del protocolo 802.11 para sistemas Linux. Fuente: https://wifidiving.substack.com/p/linux-kernel-wifi-stack-basics (accedido: 9/11/2024)

- **FullMAC**: tarjeta de red inalámbrica en la que la MLME (*Media Access Control subLayer Management Entity*) es gestionada completamente por el hardware. MLME es la entidad que implementa la máquina de estados de la capa MAC (*Medium Access Control*) del protocolo 802.11 (la máquina de estados se basa en el diagrama de estados introducido en el Anexo A). Tal y como se puede ver en el diagrama, el *kernel* de Linux ofrece las operaciones del módulo `cfg80211` para implementar los *drivers* de este tipo de tarjetas.
- **SoftMac**: tarjeta de red inalámbrica en la que la MLME es gestionada completamente por el software. El módulo del *kernel* de Linux que implementa la MLME es el `mac80211`,

y como puede observarse en el diagrama, se posiciona entre el módulo `cfg80211` y el *driver* del dispositivo. Esto es lógico ya que los desarrolladores utilizan las operaciones que ofrece el módulo `mac80211` para implementar el *driver*, pero el comportamiento de la máquina de estados depende en gran medida de aspectos de configuración, manejados principalmente con el módulo `cfg80211`. Por ello ambos módulos trabajan conjuntamente.

Para definir por completo el diagrama y volviendo a la parte superior del mismo, se puede observar que la comunicación entre los procesos del espacio de usuario, como es `wpa_supplicant`, y los procesos del *kernel*, como el *stack* inalámbrico, se controla por medio de `netlink`, la interfaz de Linux basada en *sockets* para comunicar ambos tipos de procesos.

Una vez completado el estudio de los distintos módulos que intervienen en el *stack* inalámbrico de Linux, se puede realizar un análisis de los *logs* del sistema con un mayor nivel de entendimiento, pues en los distintos mensajes registrados, es habitual encontrar los nombres de los módulos que se han introducido en esta sección.

3.4.2. Logs del sistema

Para depurar correctamente es necesario conocer dónde se puede buscar información sobre los distintos eventos ocurridos en sistemas Linux.

- **Journal**: Sistema de registro de eventos en sistemas Linux que ofrece una interfaz centralizada de escritura de mensajes para los distintos componentes del sistema. Los eventos quedan registrados en '`/var/log/journal`' o '`/run/log/journal`'[8] y normalmente se utiliza la herramienta `journalctl` para consultarlos, que proporciona distintas opciones para filtrar por los eventos de interés.
- **dmesg**: Herramienta que lista el búfer de mensajes específicos del *kernel*. Este búfer almacena mensajes relacionados con dispositivos hardware, el arranque del sistema, *drivers*, etc.
- **/var/log**: Directorio en el que las distintas aplicaciones del sistema también pueden escribir sus mensajes de depuración. Se pueden usar distintas herramientas de procesamiento de texto como `grep`, `sed` y `awk` para consultar los distintos archivos.

Con la herramienta `journalctl` se puede filtrar por los eventos registrados en el `journal` por un componente específico, por lo que se comenzó estudiando los distintos mensajes que las herramientas del espacio de usuario como `NetworkManager` o `wpa_supplicant` registraban durante las distintas iteraciones del primer experimento. Investigando los mensajes de `wpa_supplicant`, se descubrieron dos sucesos que se repetían en bastantes de las iteraciones y que resultaban llamativos teniendo en cuenta lo estudiado hasta ahora. La Figura 3.6 muestra ambos sucesos con información detallada a pie de figura para entenderlos correctamente.

1. La Figura 3.6a muestra como al llegar a cierto umbral de RSSI, el propio sistema inicia un escaneo del espectro en busca de nuevos APs. Esto resulta curioso porque no se está

[8] Sistema de ficheros temporal que contiene datos de ejecución volátiles que muestra el sistema desde que se inició.

siguiendo ninguno de los métodos que ofrecen los estándares 802.11k/v para favorecer al *Fast Roaming*, sino que el sistema, por iniciativa propia, inicia un escaneo para recabar información sobre otros APs.

2. La Figura 3.6b muestra como se produce un timeout en la fase de asociación cuando se está ejecutando el *roaming* contra un nuevo AP, desencadenado una desconexión total con la red.

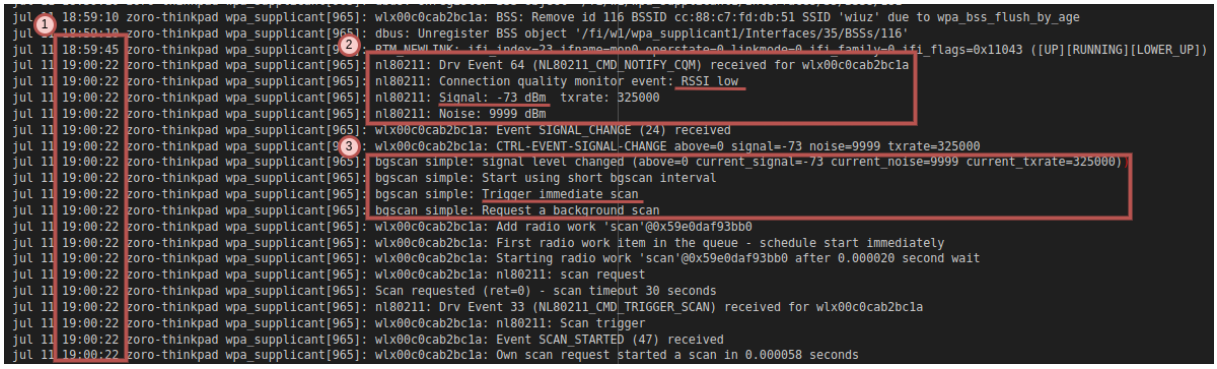
3.4.3. Estudio de código fuente y medidas adoptadas

Para comprender el origen de los dos eventos introducidos en la sección anterior, se recurrió al estudio de manuales, documentación *online* y código fuente de las herramientas `NetworkManager` y `wpa_supplicant`.

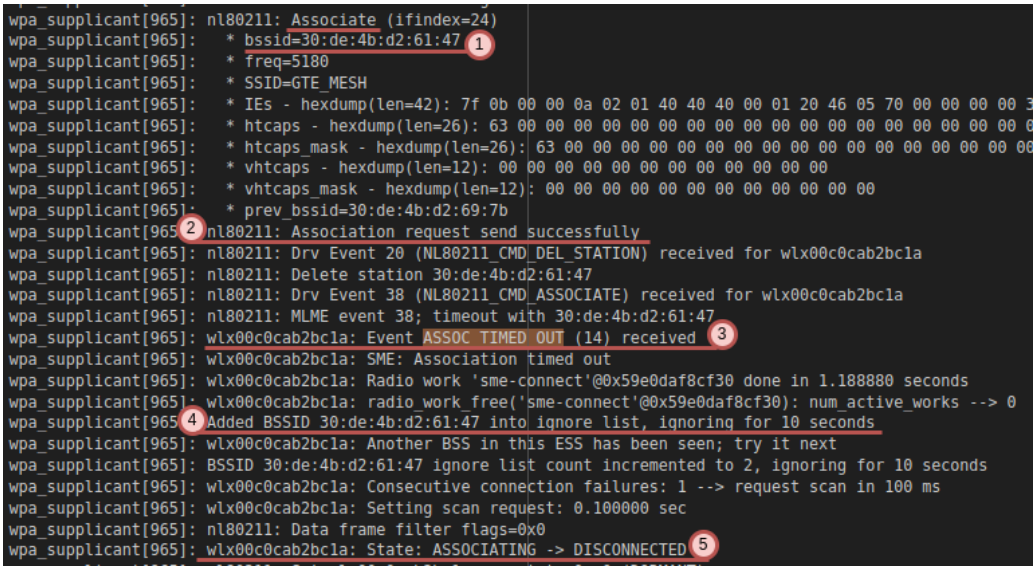
Respecto al evento introducido en la Figura 3.6a, en el que se inicia un escaneo en busca de APs tras caer bajo cierto umbral de RSSI, resulta que `wpa_supplicant` cuenta con un parámetro de configuración llamado *bgscan* que precisamente ordena comenzar un escaneo cuando la señal cae bajo cierto umbral. *bgscan* es la abreviatura de *background scan*, y la sintaxis específica puede consultarse en el Anexo C. Tras consultar los archivos de configuración del `NetworkManager` y el `wpa_supplicant`, no se encontró en ninguno de ellos una mención explícita a bgscan, pero consultando los *logs* del sistema, se encontró lo mostrado en la Figura 3.7. De alguna manera, `NetworkManager` comanda los valores de *bgscan*, ofreciendo 2 posibles valores:

- **'simple:30:-65:300'**: obliga a `wpa_supplicant` a realizar un escaneo cada 30 segundos si la señal cae por debajo de los -65 dBm. En caso contrario, escanear cada 300 segundos.
- **'simple:30:-70:86400'**: obliga a `wpa_supplicant` a realizar un escaneo cada 30 segundos si la señal cae por debajo de los -70 dBm. En caso contrario, escanear cada 86400 segundos.

Al no encontrar nada en los distintos archivos de configuración, se recurrió al estudio del código fuente de ambos módulos. Estudiando el código de `NetworkManager`, se descubrió que los dos valores que se ven en la Figura 3.7 se encuentran *"hardcodeados"*, es decir, explícitamente incrustados en sus líneas de código. En el Anexo C se muestran las secciones de código que establecen ambos valores. `NetworkManager` escoge entre uno u otro valor en función de si la red en la que el dispositivo se encuentra conectado cuenta con varios APs ofreciendo el mismo SSID o cuenta con un único AP. Si hay varios APs se utiliza el valor "`simple:30:-65:300`", pues con él se busca el *roaming* más agresivamente que con el otro valor. Dado que el protocolo 802.11k ya establece una serie de protocolos en los que clientes y APs trabajan conjuntamente para tener conocimiento sobre la situación del espectro, es posible que el inicio de un escaneo causado por el sistema operativo del cliente en una línea independiente a estos protocolos entorpezca el proceso de *roaming*. Para verificar esta hipótesis, se realizaron varias iteraciones del experimento estableciendo un valor de bgscan con el que nunca se fuese a iniciar un escaneo por esta vía, con el valor "`simple:30:-120:86400`", el cual fuerza el escaneo si y solo si el nivel de RSSI cae por debajo de los -120 dBm (este nivel es impracticable y los protocolos definidos en los estándares 802.11k/v deberían ser suficiente



(a) Tres puntos importantes en los que fijarse en esta secuencia de mensajes. El primero de todos, en la parte izquierda de la Figura, donde se puede observar que desde las 18:59:45 hasta las 19:00:22 no se registra nada, y a partir de ese instante comienzan a registrarse mensajes continuamente. El segundo punto, en la parte central superior, define lo que ha ocurrido: ha llegado un aviso por parte de la tarjeta de red (identificada por `wlx00c0cab2bc1a`) que notifica un nivel bajo de RSSI (-73 dBm). Por último, el tercer punto, en la parte central, registra una serie de mensajes con el prefijo "*bgscan simple*" que vienen a decir que se inicia un "*immediate scan*", esto es, se va a analizar el espectro en busca de nuevos APs.



(b) Cinco puntos en los que hay que detenerse, en orden de arriba a abajo, en esta secuencia de mensajes. El primero de ellos aparece junto a una dirección MAC, y en la línea inmediatamente superior se puede ver el evento "*Associate*". Esto indica que se inicia el proceso de asociación contra un nuevo AP, identificado por esa MAC. En el segundo punto, aparece el mensaje que informa de que se ha enviado la petición al nuevo AP correctamente. El tercer punto muestra un mensaje crítico, pues indica que ha saltado un timeout en el proceso de asociación "*ASSOC_TIMED_OUT*". Esto desencadena en el cuarto punto, en el que se puede ver que el AP contra el que se pretendía asociar, se añade a una "lista de ignorados", para ignorarlo durante 10 segundos. Por último, el quinto punto muestra una transición en la máquina de estados del protocolo 802.11. Se transiciona al estado "*DISCONNECTED*", esto es, se pierde la conexión de red.

Figura 3.6: Información observada en los *logs* durante las iteraciones del primer experimento que podrían explicar algunos de los resultados desfavorables.

para evitarlo). El Anexo C muestra cómo se hizo que el sistema trabajase con este valor. Aplicar este ajuste, junto a la comprensión del motivo del timeout que se explicará a continuación, fue

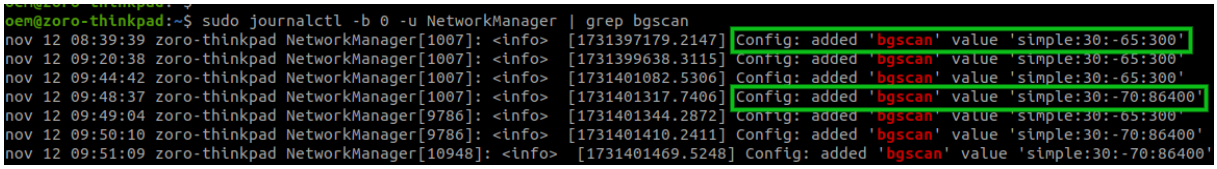


Figura 3.7: Salida del comando `journalctl`, que muestra los mensajes registrados desde el inicio del sistema (`-b 0`) por el módulo `NetworkManager` (`-u NetworkManager`) y filtrando por la palabra *bgscan* (`| grep bgscan`). En los rectángulos verdes se destacan los dos posibles valores de *bgscan* que `NetworkManager` configura.

uno de los factores esenciales para hacer funcionar el *roaming* correctamente. Posteriormente, en los resultados mostrados en la Sección 4.3, con la diferencia visible de rendimiento entre las Figuras 4.3 y 4.8, se ilustra gráficamente la importancia de aplicar este ajuste.

Respecto al evento de timeout introducido en la Figura 3.6b, se buscó la raíz del problema para averiguar si era posible incrementar el tiempo de timeout y evitar así que el nodo que ofrece un mejor valor de RSSI fuera ignorado durante 10 segundos, lo cual es fatal para el *Fast Roaming*. La investigación consistió en el estudio del código fuente de `NetworkManager`, `wpa_supplicant` y los módulos 802.11 del *kernel* de Linux. Se descubrió que el tiempo de timeout depende de una constante definida en el código fuente del *kernel* de Linux, y por tanto cambiar este valor supondría recompilar el *kernel* por completo. El procedimiento para encontrar este detalle se puede consultar en el Anexo C. Una vez encontrado el origen del timeout se pensó si el entorno en el que se estaba desarrollando el experimento podía estar lo suficientemente sobrecargado para que por motivos de colisiones o esperas en el envío de tramas terminase saltando el timeout. Los laboratorios colindantes al que se estaba trabajando pertenecen al grupo Communications Networks and Information Technologies (CeNIT) de la Universidad de Zaragoza, que trabajan con redes experimentales WiFi en la banda de los 5GHz. Para examinar si esta podía ser la posible causa, se realizaron varias iteraciones del primer experimento en una ubicación menos sobrecargada, comenzando el trayecto en el laboratorio 0.05b, en la planta baja de la escuela. En esta ubicación no volvió a darse el evento de timeout en ninguna iteración.

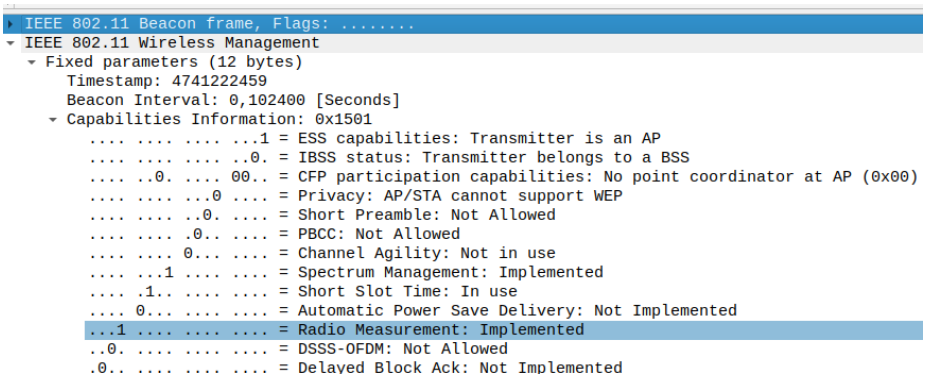
3.5. Análisis del tráfico de red

Junto al estudio y medidas adoptadas introducidos en la sección anterior, se analizó el intercambio de tramas que tenía lugar entre los APs y las tarjetas de red en las múltiples repeticiones del primer experimento para terminar de comprender las causas por las cuales el *roaming* no era satisfactorio. Para ello, una de las dos tarjetas de red que no actuaban como cliente en la repetición, se configuraba en modo monitor, lo que permite capturar todas las tramas que están circulando en determinado canal del medio radioeléctrico con herramientas como `Wireshark`, la utilizada en este trabajo. En el Anexo B se detalla todo lo relativo a la configuración de una tarjeta en modo monitor para poder capturar tramas con `Wireshark`.

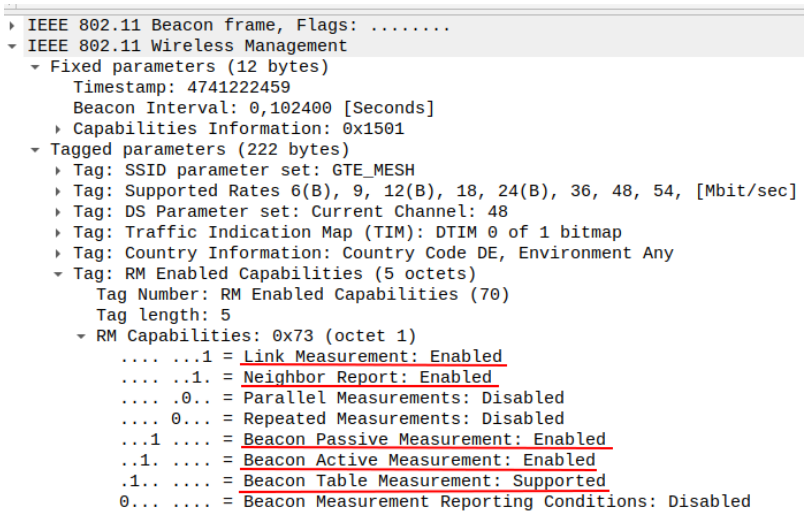
En primer lugar, se buscó evidencia de que tanto los APs como las tres tarjetas implementaran alguno de los protocolos establecidos en los estándares 802.11k/v. Los estándares establecen que un AP debe anunciar las funcionalidades que implementa en sus tramas *Bea-*

con, y la Figura 3.8 muestra la trama de uno de los APs. En la Figura 3.8a se puede ver que el AP anuncia que implementa la funcionalidad *Radio Measurement*, dentro del elemento *Capabilities Information*. Esto le obliga, según el estándar 802.11k, a incluir también en la trama el elemento *RRM Enabled Capabilities* indicando los servicios específicos que implementa. Este elemento se puede observar en la Figura 3.8b, que anuncia que el AP implementa los servicios *Link Measurement*, *Neighbor Report* y *Beacon Passive/Active/Table Measurement*.

Esto es lo que los APs ofrecen respecto al estándar 802.11k. Para buscar evidencia de implementación del estándar 802.11v, hay que fijarse en el elemento *Extended Capabilities*, también de las tramas *Beacon*. Este se muestra en la Figura 3.9, y como se puede observar, los APs implementan los servicios *TFS*, *WNM Sleep Mode*, *TIM Broadcast* y *BSS Transition*.



(a) Funcionalidad *Radio Measurement* implementada, anunciado dentro del elemento *Capabilities Information*.



(b) Subrayado en rojo, los servicios específicos del estándar 802.11k que el AP implementa, dentro del elemento *RRM Enabled Capabilities*.

Figura 3.8: Anuncio de los servicios del estándar 802.11k que los APs implementan por medio de sus tramas *Beacon*.

Para aprovechar estos servicios que implementan los APs, es necesario que las tarjetas de red también los implementen. Las tarjetas de red, o clientes, anuncian sus funcionalidades en las tramas *Association Request* que envían hacia un AP, de esta manera el AP conoce qué

servicios puede utilizar con dicho cliente. La estructura de los elementos a consultar dentro de la trama es la misma que en el caso de los APs, por lo que las figuras con la evidencia se pueden consultar en el repositorio público de GitHub del proyecto[9]. En este caso, las tres tarjetas, implementan únicamente el servicio *Beacon Passive/Active/Table Measurement* en lo referente al estándar 802.11k, por lo que los APs solo podrán utilizar ese a pesar de implementar otros servicios de este estándar. En lo que se refiere al 802.11v, las tres tarjetas implementan los servicios *WNM Sleep Mode* y *BSS Transition*, no obstante, de estos dos servicios el único que está relacionado con el *Fast Roaming* es el *BSS Transition*, introducido ya en la Sección 2.2.2.2.1, por lo que es el que los APs podrán utilizar para favorecer el *roaming*.

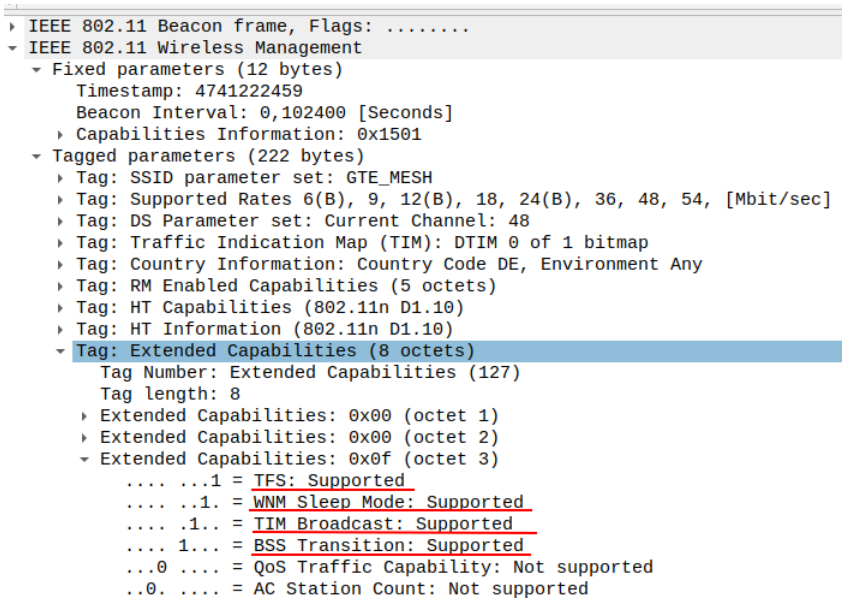
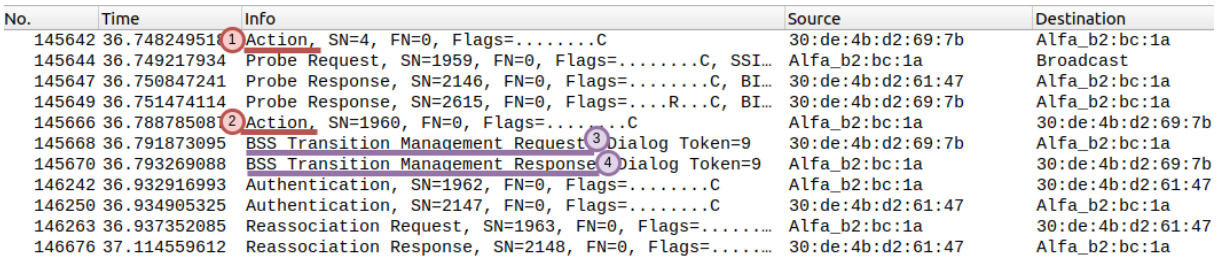


Figura 3.9: Anuncio de los servicios del estándar 802.11v que los APs implementan por medio de sus tramas *Beacon*. Subrayado en rojo, las funcionalidades que implementan, recogidas dentro del elemento *Extended Capabilities*.

Conociendo ya las funcionalidades que tanto APs como tarjetas de red anuncian que implementan, se analizaron las tramas intercambiadas durante el punto crítico en el que debe darse el *roaming*, ese momento en torno a los -70 dBm, en el que parece que ocurre algo como ya se ha comentado al concluir la Sección 3.2. La Figura 3.10 ilustra el intercambio de tramas cuando el *roaming* es satisfactorio.

Una vez comprendida la correcta secuencia de tramas que debe darse en un *roaming* guiado por los protocolos de los estándares 802.11k/v, y subsanadas también las dos situaciones de error que se han introducido en la Sección 3.4.2, se avanzó a la siguiente fase del trabajo, en la que se analizó si este proceso de *roaming* es suficiente para cubrir los requisitos de calidad de servicio que requieren las comunicaciones en aplicaciones robóticas con restricciones de tiempo real.

[9]https://github.com/gcr-UZ/tfm-mesh



No.	Time	Info	Source	Destination
145642	36.748249518	Action, SN=4, FN=0, Flags=........C	30:de:4b:d2:69:7b	Alfa_b2:bc:1a
145644	36.749217934	Probe Request, SN=1959, FN=0, Flags=........C, SSI…	Alfa_b2:bc:1a	Broadcast
145647	36.750847241	Probe Response, SN=2146, FN=0, Flags=........C, BI…	30:de:4b:d2:61:47	Alfa_b2:bc:1a
145649	36.751474114	Probe Response, SN=2615, FN=0, Flags=....R...C, BI…	30:de:4b:d2:69:7b	Alfa_b2:bc:1a
145666	36.788785087	Action, SN=1960, FN=0, Flags=........C	Alfa_b2:bc:1a	30:de:4b:d2:69:7b
145668	36.791873095	BSS Transition Management Request, Dialog Token=9	30:de:4b:d2:69:7b	Alfa_b2:bc:1a
145670	36.793269088	BSS Transition Management Response, Dialog Token=9	Alfa_b2:bc:1a	30:de:4b:d2:69:7b
146242	36.932916993	Authentication, SN=1962, FN=0, Flags=........C	Alfa_b2:bc:1a	30:de:4b:d2:61:47
146250	36.934905325	Authentication, SN=2147, FN=0, Flags=........C	30:de:4b:d2:61:47	Alfa_b2:bc:1a
146263	36.937352085	Reassociation Request, SN=1963, FN=0, Flags=.....	Alfa_b2:bc:1a	30:de:4b:d2:61:47
146676	37.114559612	Reassociation Response, SN=2148, FN=0, Flags=.....	30:de:4b:d2:61:47	Alfa_b2:bc:1a

Figura 3.10: Secuencia de tramas intercambiadas en un *roaming* satisfactorio, provocado por los protocolos de los estándares 802.11k/v. Cuando el nodo 1 detecta que la RSSI del cliente cae bajo cierto umbral, le pide al cliente que ejecute una medición del espectro (trama 1, *Radio Measurement Request*). En este momento, el cliente genera una trama *Probe Request* en modo *broadcast*, pero marcando que únicamente contesten los APs con el SSID al que se encuentra conectado. Esto provoca que únicamente respondan los dos APs de la red, como se puede ver en las siguientes 2 tramas *Probe Response*. Con la respuesta de los 2 APs, el cliente genera y manda la trama marcada como 2, *Radio Measurement Report,* que es el informe que el AP espera con la información del entorno del cliente. En este momento, el AP aconseja un cambio al cliente por medio de la trama marcada como 3, *BSS Transition Management Request*, pues es consciente gracias al informe de que hay un mejor nodo candidato. El cliente contesta que acepta la solicitud en la trama marcada con el 4, *BSS Transition Management Response*, y las tramas posteriores representan las fases de autenticación y asociación contra el nuevo AP (nodo 2).

Capítulo 4

Análisis experimental del desempeño del Fast Roaming

Tras comprender cómo debe funcionar el *Fast Roaming* con las funcionalidades concretas que implementa el hardware disponible en este proyecto, en la segunda fase del trabajo se recopilan y analizan distintas métricas de rendimiento que tienen importancia en el ámbito de las comunicaciones en aplicaciones robóticas con restricciones de tiempo real. Por ejemplo, una teleoperación basada en transmisión de vídeo, en la que es deseable que ante un nuevo estímulo los teleoperadores respondan dentro de 0 a 3 segundos, con un tiempo de respuesta mínimo de 0.2 segundos [3]. Por tanto, a las métricas de RSSI y ancho de banda medidas en la primera fase, se añaden la tasa de pérdida de paquetes, el número de paquetes recibidos, la latencia, la variación en el tiempo de latencia, conocida como *jitter*, y el tiempo exacto que dura el proceso de *roaming*.

4.1. Métricas

4.1.1. Tasa de pérdida de paquetes

La tasa de pérdida de paquetes es una métrica clave cuando se pretende caracterizar el rendimiento de una red. Se refiere al porcentaje de paquetes de datos que se pierden durante su transmisión a través de una red. Los paquetes pueden perderse por diversas razones, como congestión de red, fallos en los equipos, interferencias, errores en el protocolo o condiciones desfavorables en la conexión. En este experimento se quiere medir cuántos paquetes se pierden durante el proceso de *roaming*. La Ecuación 4.1 muestra cómo calcular esta métrica.

$$Tasa\ de\ pérdida\ de\ paquetes\ (\%) = \frac{Número\ de\ paquetes\ perdidos}{Número\ total\ de\ paquetes\ enviados} \times 100 \quad (4.1)$$

4.1.2. Número de paquetes recibidos

Dado que la métrica anterior, la tasa de pérdida de paquetes, se trata de un porcentaje, no se puede interpretar por sí sola. Es necesario conocer el número total de paquetes recibidos, esto es, la diferencia entre el denominador y el numerador de la Ecuación 4.1. La importancia de este dato se puede comprender con el siguiente ejemplo: si se pierde 1 paquete de un total de

10 paquetes enviados, se obtiene una tasa de pérdida de paquetes del 10 %. De igual manera, si se pierden 1000 paquetes de un total de 10000 paquetes enviados, la tasa de pérdida de paquetes es del 10 %. Si en media, la red es capaz de transmitir 10000 paquetes por unidad de tiempo, analizando únicamente la tasa de pérdida de paquetes, se estaría perdiendo el dato que permite descubrir una degradación en el número total de paquetes recibidos, lo cual desvela una bajada del rendimiento en la transmisión de datos.

4.1.3. Latencia

La latencia en una red de comunicaciones es el tiempo que tarda un paquete de datos en viajar desde su origen hasta su destino dentro de la red. Se mide generalmente en milisegundos y es un factor crítico en el rendimiento de aplicaciones que requieren interacciones en tiempo real, como videollamadas, videojuegos en línea, teleoperación o transmisión de datos críticos. Existen dos tipos de latencia:

- Unidireccional: Tiempo que tarda un paquete en viajar desde el origen hasta el destino.
- Bidireccional o de ida y vuelta (RTT, *Round Trip Time*): Tiempo total para que un paquete viaje al destino y su respuesta regrese al origen.

4.1.4. Jitter

El *jitter* es la fluctuación que se da en la latencia de una red, es decir, la variabilidad en el tiempo que un paquete tarda en viajar desde el origen hasta el destino. En una red ideal, los paquetes deberían llegar a intervalos constantes y predecibles, esto es, con una latencia constante. Sin embargo, en redes reales, factores como la congestión o errores en los nodos intermedios pueden causar que los paquetes lleguen con tiempos desiguales. El *jitter* se mide en el orden de los milisegundos.

El *jitter* ideal es igual a 0 ms. Esto es, que todos los paquetes tardan el mismo tiempo en atravesar la red y llegan al receptor igualmente espaciados. El *jitter* puede ser positivo o negativo. Un *jitter* positivo significa que los paquetes tardan cada vez más en llegar, pudiendo ser descartados y desencadenando pérdida de información. Un *jitter* negativo significa que los paquetes llegan cada vez con menos separación, esto es, antes de lo previsto, lo que podría desencadenar una saturación en el receptor. El *jitter* tiene especial impacto en aplicaciones como videoconferencias, *streaming* o juegos en línea, pues puede causar interrupciones, retrasos o pérdida de sincronización. Normalmente, el nodo destino decodifica los paquetes que recibe en una transmisión de vídeo y/o audio a una tasa constante, y una variación en la latencia, si no está prevista, puede producir el efecto visual de congelamiento/emborronamiento de la imagen o desincronización entre vídeo y audio. El Anexo B recoge el procedimiento de `iPerf3` para calcular el *jitter*.

4.1.5. Tiempo de roaming

Una de las métricas más importantes para caracterizar completamente el proceso de *roaming* es el tiempo exacto que dura, ya que conociendo a priori la cantidad de tiempo que el cliente va a estar ocupado gestionando el cambio, se pueden implementar técnicas preventivas

a nivel de aplicación. Para obtener esta métrica, se pueden seguir varias aproximaciones. En los experimentos o aplicaciones en los que se transmiten datos de manera constante, probablemente lo más coherente sea considerar el tiempo de *roaming* como la diferencia temporal que existe entre el primer paquete que se recibe por medio del nuevo AP y el último paquete que se recibió por el AP anterior. La Figura 4.1 ilustra gráficamente lo que sería este tiempo.

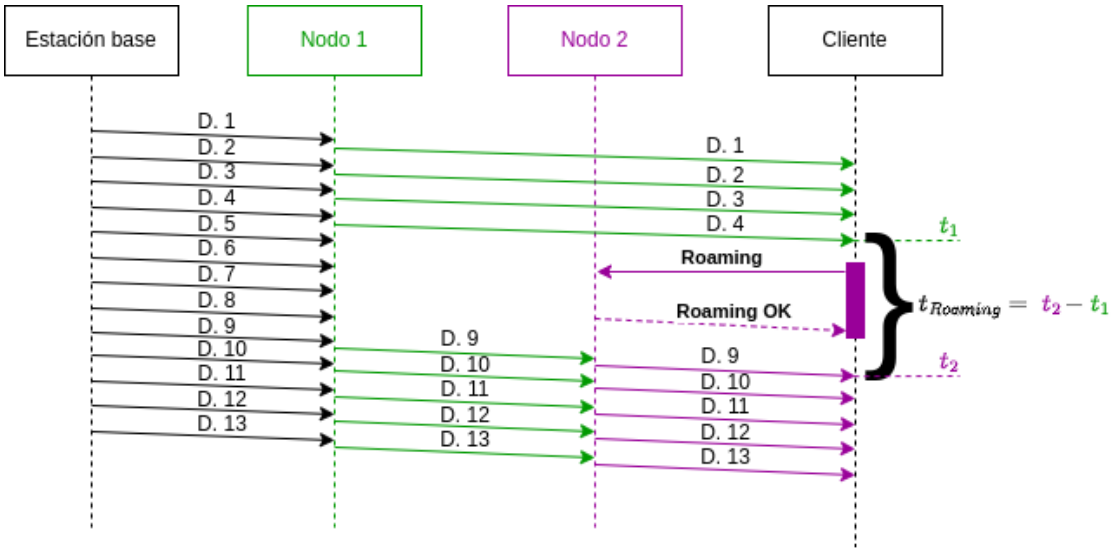


Figura 4.1: Cálculo del tiempo de *roaming* basándose en el instante de recepción de datagramas en el cliente (t_1 y t_2). Las tramas de gestión involucradas en el proceso de *roaming* se han excluido para facilitar la legibilidad.

En los experimentos sin transmisión de datos, en los que simplemente se quiere observar sobre qué punto del recorrido se da el cambio o sobre qué nivel de RSSI, se puede medir el tiempo que se tarda en completar la autenticación y la asociación contra el nuevo AP. Esto implica medir la diferencia temporal entre la última trama de esta fase, *Reassociation Response*, enviada desde el AP hacia el cliente, y la primera trama, *Authentication Request*, enviada desde el cliente hacia el AP. No obstante para este experimento, al haber transmisión de datos, se considera más importante y se utilizará el cálculo ilustrado en la Figura 4.1.

4.1.6. Requisitos para una aplicación robótica de teleoperación basada en vídeo

En una teleoperación basada en vídeo, en primer lugar, se necesita que la red ofrezca velocidades de transferencia de datos de subida y de bajada equilibradas, ya que los comandos del operador deben enviarse al sistema remoto (subida), mientras que los datos de video, y otros, como telemetría, deben transmitirse de vuelta al operador (bajada), tal y como se ilustra en la Figura 1.1. Ambos sentidos son igualmente críticos para la operación en tiempo real. En principio, una red WiFi Mesh debe soportar sin problemas una velocidad adecuada en ambos sentidos.

En cuanto a los requisitos específicos para una teleoperación fluida y en tiempo real de un sistema remoto a través de la red, las restricciones clave son las siguientes:

1. Baja latencia: la baja latencia garantiza que los comandos enviados por el operador sean ejecutados casi en tiempo real por el sistema remoto. Una alta latencia puede causar retrasos, dificultando el control del sistema o incluso volviéndolo peligroso en aplicaciones críticas (drones o vehículos autónomos).

2. Elevada fiabilidad: la red debe entregar los paquetes de manera fiable, esto es, con una mínima tasa de pérdida de paquetes. Un paquete perdido podría significar la pérdida de control del sistema remoto, desencadenando situaciones poco seguras.

3. Bajo *jitter*: la consistencia en los tiempos de llegada de los paquetes es crucial para el *feedback* en tiempo real. La fluctuación en los tiempos de llegada de los paquetes puede producir en el sistema remoto movimientos retardados o inconsistentes, entorpeciendo la habilidad del operador para manejarlo fluidamente.

Los requisitos clave para una transmisión de vídeo exitosa dependen en gran medida de factores como la calidad del vídeo (la cual, a su vez, depende principalmente de la resolución, el *bitrate* y los FPS (Frames Por Segundo)), el protocolo de transmisión de vídeo utilizado (RTP (*Real Time Protocol*), RTSP (*Real Time Streaming Protocol*), UDP, TCP, etc.) y las condiciones de la red. A la baja latencia, la elevada fiabilidad y el bajo *jitter*, se añade la necesidad de contar con suficiente ancho de banda, pues habitualmente es el recurso más consumido en una transmisión de vídeo. Los siguientes valores suelen ser los mínimos necesarios para transmitir con la resolución indicada:

- *Standard Definition* (SD): 1-3 Mbps.
- *High Definition* (HD): 5-10 Mbps.
- *Full HD* (1080p): 10-20 Mbps.
- *4K Ultra HD:* 25-50 Mbps.

La Tabla 4.1 recoge los valores ideales para que una teleoperación basada en vídeo tenga un rendimiento óptimo [2].

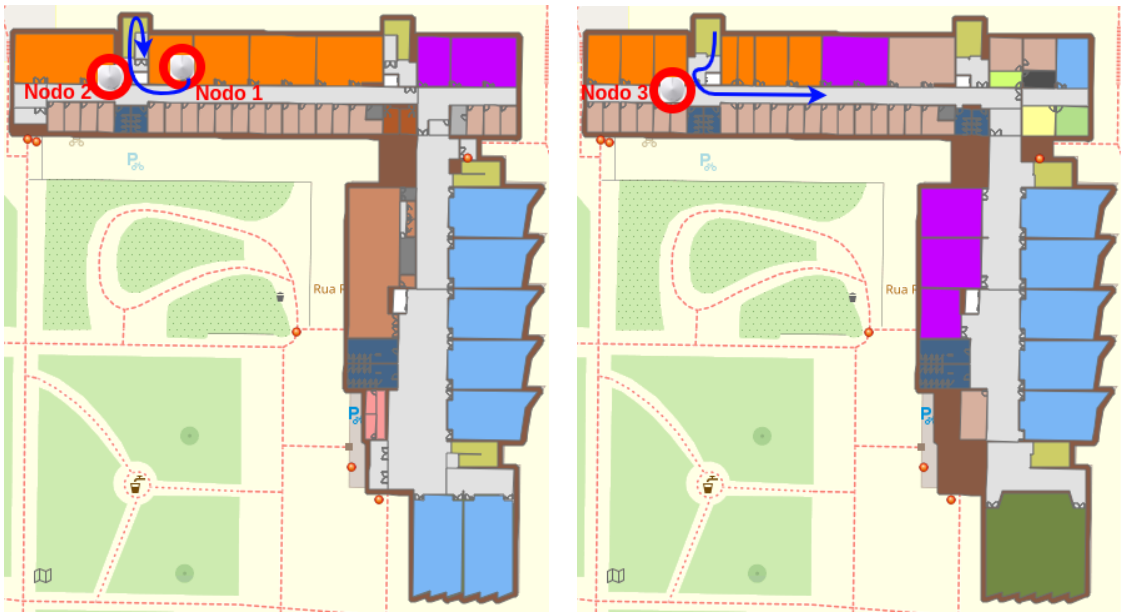
Métrica	**Valor ideal**
Latencia[1]	< 100 ms
Tasa de pérdida de paquetes	$< 1\,\%$
Jitter	$< 20 - 30$ ms
Ancho de banda	3 ó 10 Mbps (resolución SD o HD, respectivamente)

Tabla 4.1: Valores mínimos necesarios para el correcto funcionamiento de una teleoperación basada en transmisión de vídeo.

[1]En general, 100 ms se considera el límite para aplicaciones críticas, en algunas aplicaciones de control remoto son admisibles 300 ms o incluso 500 ms.

4.2. Configuración del experimento

En este experimento se desplegaron los tres nodos Mesh disponibles. Para evitar interferencias con las redes de investigación de la planta segunda, se desplegaron dos nodos en la planta baja del edificio y un tercer nodo nada más subir las escaleras para acceder a la primera planta. La Figura 4.2 muestra el plano en planta que ayudará a comprender el despliegue. El trayecto del experimento comienza junto al nodo 1, dentro del laboratorio 0.05b. Tras salir del laboratorio, se gira a la derecha para acercarse al nodo 2, para posteriormente subir las escaleras y pasar por el nodo 3, accediendo al pasillo de despachos de la primera planta y andando a través de este en sentido izquierdo. En cada repetición del experimento, se intenta que transcurran 60 segundos desde el punto inicial hasta el final, intentando andar al mismo ritmo durante todo el recorrido para obtener la mayor reproducibilidad posible.



(a) Planta baja del edificio Ada Byron. El nodo 1 se sitúa dentro del laboratorio 0.05b y el nodo 2 junto a la puerta de acceso a las escaleras. La trayectoria, indicada con la flecha azul, consiste en salir del laboratorio para subir por las escaleras a la planta primera.

(b) Planta primera del edificio Ada Byron. El nodo 3 se sitúa junto a la puerta de acceso a las escaleras, en el pasillo. La trayectoria, indicada con la flecha azul, consiste en continuar el trayecto hacia la izquierda del pasillo después de haber subido las escaleras.

Figura 4.2: Despliegue de nodos en el segundo experimento del proyecto[2].

Este segundo experimento del proyecto cuenta con dos fases:

1. Caracterización objetiva de métricas: por medio de las herramientas introducidas en la Sección 4.2.1.1, se van a cuantificar las métricas clave explicadas en la Sección 4.1.6 para analizar, especialmente en los puntos críticos en los que debe darse el *roaming*, si tanto la red WiFi Mesh como los clientes cumplen con los requisitos para una teleoperación basada en transmisión de vídeo.

2. Caracterización subjetiva de la transmisión de vídeo: con las métricas ya cuantificadas, se realizarán varias iteraciones del recorrido mientras se produce una transmisión de vídeo

[2]https://sigeuz.unizar.es/ (accedido: 28/11/2024)

desde la estación móvil hacia la estación base. El propósito de esta fase es observar si en la estación base, en la cual el teleoperador tomaría acciones en función de la señal de vídeo que visualiza, el vídeo se recibe de manera fluida, sin pixelación o cortes, con especial atención en los puntos críticos en los que se produce el *roaming*.

4.2.1. Herramientas utilizadas

4.2.1.1. Caracterización objetiva de métricas

Se ha utilizado de nuevo `iPerf3` para la transmisión de datos. En este caso se ha optado por una conexión UDP, el protocolo de transporte más comúnmente utilizado en aplicaciones basadas en transmisión de vídeo o teleoperación en tiempo real, dado que es más rápido que TCP al no implementar un control de flujo ni retransmisiones en caso de pérdida de paquetes. Las aplicaciones de vídeo en *streaming* pueden tolerar la pérdida ocasional de paquetes sin un impacto significativo en la calidad percibida por el usuario. En las aplicaciones de teleoperación en tiempo real, se busca una latencia mínima, para la cual UDP es más favorable que TCP, y un pequeño porcentaje de pérdida de datos puede ser aceptable porque los sistemas suelen actualizarse constantemente con nuevos comandos o estados. Además, `iPerf3` informa del *jitter* únicamente con UDP, pues es una métrica más relevante en protocolos sin conexión como UDP que en protocolos orientados a conexión como TCP, en los que las distintas técnicas que garantizan una entrega confiable y en orden suavizan las variaciones en los tiempos de llegada de los paquetes.

En este experimento es la estación base la que envía datos al cliente que transita por el recorrido, simulando el sentido de transmisión que se daría en una teleoperación desde la estación base. Se decidió trabajar con 3 tasas de transmisión distintas, para ver si la cantidad de datos transmitidos influye en el comportamiento o desempeño del *roaming*. Se optó por que la estación base transmitiese paquetes siempre al mismo ritmo, 1 paquete por ms, independientemente de la tasa. Si el ritmo se mantiene constante, entonces para cambiar la tasa lo que tiene que modificarse es el tamaño de cada paquete. La Tabla 4.2 muestra el tamaño de paquete necesario para conseguir dicha tasa a un ritmo de 1 paquete por ms.

Tasa (Mbps)	**Tamaño de paquete (Bytes)**
1	125
5	625
11.68	1460

Tabla 4.2: Tamaño de paquete necesario para conseguir la tasa de transmisión indicada a ritmo de 1 paquete por ms.

Para conocer con el mayor nivel de detalle lo que ocurre en cada punto del recorrido, se estableció que `iPerf3` informase del rendimiento cada 0.1 segundos (es el máximo nivel de detalle que permite). Esto es, que cada 0.1 s, `iPerf3` muestra por salida estándar una línea con la siguiente información calculada en ese intervalo específico de 0.1 segundos: número total de datos enviados (KB), tasa de transmisión (Mbps), *jitter* (ms) y el número de datagramas perdidos sobre el total de recibidos. La Figura 4.7 de la Sección 4.3.4 muestra un ejemplo

gráfico de la salida de iPerf3. Dado que iPerf3 permite muestrear cada 0.1 s, es el mismo intervalo que se ha establecido para preguntar al sistema por el nivel de RSSI y nodo asociado. Esta información se obtiene por medio de la ejecución de un *script* paralelamente a la ejecución de iPerf3, como ya se hizo en la primera fase del trabajo. Todos los *scripts* utilizados durante el trabajo se pueden encontrar en el repositorio público[3].

Cabe destacar, que dado que se envía 1 paquete por ms e iPerf3 muestra resultados cada 0.1 s, o lo que es lo mismo, cada 100 ms, en una situación teórica ideal, iPerf3 debería mostrar que se han recibido satisfactoriamente 100 paquetes en cada intervalo de muestreo.

Para calcular la latencia se ha utilizado la herramienta ping, la cual mide la latencia bidireccional (RTT). Paralelamente a iPerf3, se lanza esta herramienta desde la estación móvil por medio del comando 'ping -i 0.01 <ip_estacion_base>'. La opción '-i 0.01' indica que se debe enviar un paquete cada 10 milisegundos (ping no permite enviar cada milisegundo, como sí permite iPerf3).

Para calcular correctamente el tiempo de *roaming* como se ha explicado en la Sección 4.1.5, es necesario conocer el tiempo de recepción exacto de todos los paquetes en el cliente. Se ha utilizado la herramienta Wireshark para ello. Como se ha comentado en anteriores secciones, el disponer de varias tarjetas de red permite que mientras una está trabajando y recibiendo datos, siendo esta sobre la que se realizan métricas, otra tarjeta se pueda configurar en modo monitor para escuchar todas las tramas que circulan en el medio. El procedimiento y los filtros de Wireshark aplicados para obtener los paquetes de interés se recogen en el Anexo B, así como los comandos utilizados para configurar las tarjetas en modo monitor.

Para cada una de las tres tarjetas de red disponibles en el trabajo, Alfa1, Alfa2 e Intel, se realizó al menos una repetición con cada una de las tres tasas especificadas en la Tabla 4.2.

4.2.1.2. Caracterización subjetiva de la transmisión de vídeo

Para la transmisión de vídeo, se ha utilizado la herramienta FFmpeg[4]. FFmpeg es una herramienta de software libre y de código abierto que permite grabar, convertir, manipular y transmitir audio y vídeo en una gran variedad de formatos. Los comandos utilizados para hacer posible la transmisión se pueden consultar en el Anexo B. La transmisión consiste en capturar la imagen que proviene de la cámara integrada en la estación móvil y enviarla por la red a la estación base, la cual muestra la imagen en un reproductor. En el ámbito de este proyecto no es necesaria una calidad de vídeo extremadamente alta, por lo que una resolución de 720p (HD) y un bitrate de entre 1 y 3 Mbps es suficiente para percibir con nitidez la imagen de la cámara integrada de la estación móvil. Respecto a la tasa de fotogramas por segundo, con 10 fps, la imagen es suficientemente fluida en el ámbito de este trabajo.

[3] https://github.com/gcr-UZ/tfm-mesh

[4] https://www.ffmpeg.org/ffmpeg-all.html (accedido: 28/11/2024)

4.3. Análisis de resultados

4.3.1. Tasa de pérdida de paquetes y número de paquetes recibidos

La Figura 4.3 muestra una repetición del experimento para la tarjeta Alfa1 y la tasa de transmisión de 5 Mbps. La evolución de la RSSI (gráfica superior) es la esperable según lo introducido en el Capítulo 3, con el cambio de nodo en torno a los -75 dBm. La tasa de paquetes perdidos, ilustrada en la gráfica del medio, indica que el rendimiento es bueno durante prácticamente todo el trayecto, a excepción del momento en el que se da el cambio de AP, con esos dos picos del 100 % en torno a los segundos 17 y 48 del recorrido. Por último, la gráfica inferior, muestra el número total de paquetes recibidos cada 100 ms, para ver si realmente se está cerca de los 100 paquetes teóricos que deberían recibirse en este intervalo. Como se puede observar, hay una fluctuación que se incrementa cuando el cliente se encuentra conectado al nodo 3. Esta fluctuación se anula habitualmente entre muestreos sucesivos, es decir, si en una muestra `iPerf3` registra que se han recibido 70 paquetes, en la siguiente registra que se han recibido 130, teniendo como media los 100 paquetes. La Tabla 4.3 muestra la media y desviación estándar de los paquetes recibidos cada 0.1 s diferenciando por el nodo al que el cliente se encuentra asociado. Como se puede observar, la desviación aumenta cuando el cliente se encuentra asociado al nodo 3.

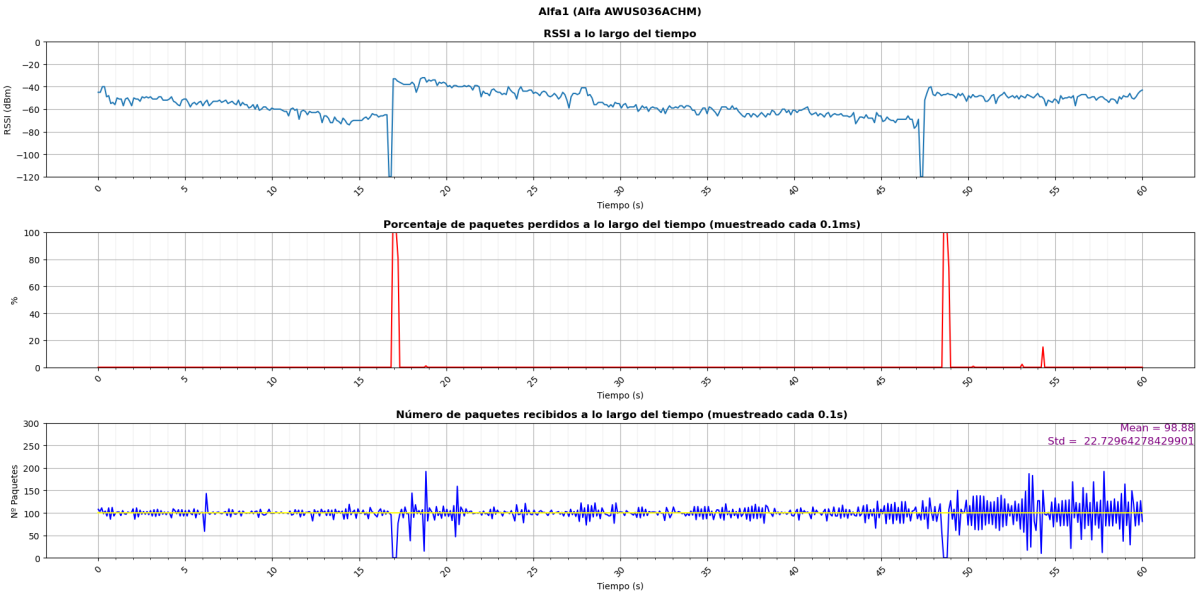


Figura 4.3: Recopilación de RSSI, tasa de pérdida de paquetes y número de paquetes recibidos para una repetición de la tarjeta Alfa1 a una tasa de 5 Mbps.

Estos resultados muestran que mientras la señal es óptima, la red podría soportar una aplicación con los requisitos mencionados en la Sección 4.1.6, con una tasa de pérdida de paquetes menor del 1 % durante estos tramos. No obstante, se debe ser consciente de que las pérdidas son inevitables en el punto crítico de *roaming*, para así aplicar medidas preventivas u optar por otras soluciones si las restricciones de tiempo real son críticas durante todo el trayecto, incluidos los puntos de cambio.

Nodo	Media	Desviación estándar
1	99.52	10.84
2	99.45	15.85
3	99.81	39.31

Tabla 4.3: Media y desviación estándar del número de paquetes recibidos cada 0.1 s diferenciando por nodo asociado (calculado para la iteración de la Figura 4.3).

4.3.2. Latencia

La Figura 4.4 muestra la latencia a lo largo del recorrido. Como se puede observar, con cada cambio de AP la latencia se incrementa ligeramente, la gráfica presenta un pequeño escalón. Esto entra dentro de lo normal pues los paquetes recorren un trayecto mayor. En el peor de los casos, cuando se está asociado al nodo 3, la latencia en promedio se encuentra en torno a los 10 ms, que es un valor aceptable. En los dos momentos críticos de *roaming* se puede observar que se dan picos de latencia debido a la pérdida de paquetes ya ilustrada en la Figura 4.3.

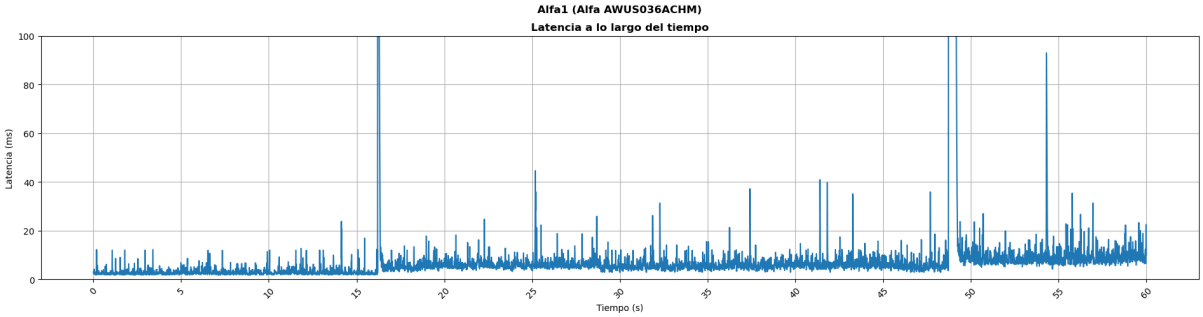


Figura 4.4: Latencia registrada para la misma repetición de la tarjeta Alfa1 a una tasa de 5 Mbps que la mostrada en la Figura 4.3.

Al igual que ocurre con la tasa de pérdida de paquetes, la latencia cumple con los requisitos mínimos mientras la señal es óptima, con valores por debajo de los 100 ms, pero en el punto crítico de *roaming* se debe ser consciente de que esta restricción no se cumple durante unos pocos milisegundos.

4.3.3. Jitter

La Figura 4.5 muestra la variación en el tiempo de llegada de los paquetes, el *jitter*, calculado por `iPerf3` según el Anexo B. Pertenece a la misma repetición que la Figura 4.3. A simple vista, lo más destacable de la gráfica podría considerarse que es el mismo fenómeno observado con la fluctuación de los paquetes recibidos en la Figura 4.3. Esto es, que mientras el cliente se encuentra asociado al nodo 3, la fluctuación en el *jitter* es mayor, con la presencia de algunos picos altos. También se puede concluir que a medida que se avanza en el recorrido, el *jitter* aumenta muy ligeramente, pues al principio se observa que el valor es cercano a 0.6 ms y al final se puede tener la percepción de que ha incrementado ligeramente. Esto se confirma analíticamente con los resultados mostrados en la Tabla 4.4, donde se muestran la media y

desviación estándar del *jitter* en función del nodo asociado. La media en el nodo 3 es mayor y la elevada desviación estándar respecto a los otros dos nodos confirma también la mayor fluctuación en el último tramo.

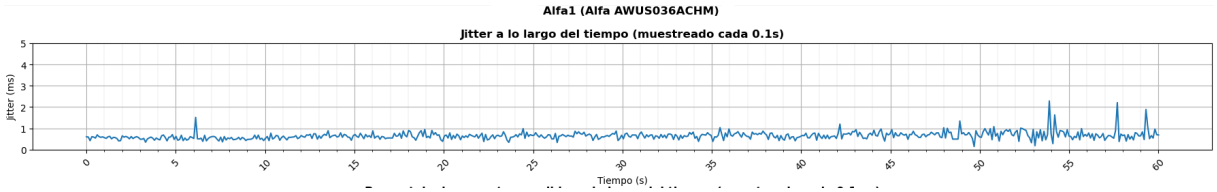


Figura 4.5: *Jitter* registrado para la misma repetición de la tarjeta Alfa1 a una tasa de 5 Mbps que la mostrada en la Figura 4.3.

Nodo	Media	Desviación estándar
1	0.58	0.11
2	0.66	0.12
3	0.73	0.31

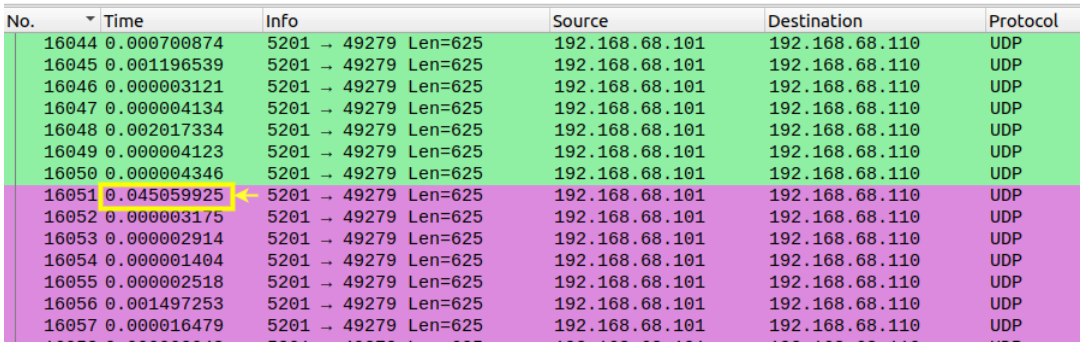
Tabla 4.4: Media y desviación estándar del *jitter* cada 0.1 s diferenciando por nodo asociado (calculado para la repetición de la Figura 4.5).

El *jitter* cumple con el requisito mínimo de permanecer por debajo de los 20 o 30 ms durante todo el trayecto. No obstante, `iPerf3` congela el cálculo del *jitter* durante la pérdida de paquetes, así que al igual que con la tasa de pérdida de paquetes y la latencia, se debe tener en cuenta esto en el punto crítico de *roaming*.

4.3.4. Tiempo de roaming

La Figura 4.6 muestra en `Wireshark` los paquetes interceptados por la tarjeta en modo monitor para la misma repetición que se ilustra en las Figuras 4.3, 4.4 y 4.5. Se puede observar en la figura un cambio de color de verde a morado. Los paquetes en verde son los que provienen del nodo 1. En morado, los que provienen del nodo 2. En la columna *Time*, se ha escogido que se muestren los segundos transcurridos desde la captura del paquete inmediatamente anterior. Justamente en la transición de verde a morado, `Wireshark` muestra un tiempo de 0.045 segundos (45 ms).

Sin embargo, esto no es suficiente para concluir que el tiempo de *roaming* ha sido exactamente de 45 ms. Si se echa un vistazo al informe de la herramienta `iPerf3` en ese mismo instante, el cual se muestra adaptado para visualizarse correctamente en la Figura 4.7, se puede observar como por tres líneas consecutivas (destacadas por el rectángulo rojo) se reciben 0 paquetes (0/0). En la siguiente línea, `iPerf3` desvela la información de que se han perdido 319 paquetes en los últimos 400 ms. O lo que es lo mismo, de los 400 paquetes que teóricamente se tendrían que haber recibido, sólo se han recibido 81.



No.	Time	Info	Source	Destination	Protocol
16044	0.000700874	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16045	0.001196539	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16046	0.000003121	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16047	0.000004134	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16048	0.002017334	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16049	0.000004123	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16050	0.000004346	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16051	0.045059925	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16052	0.000003175	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16053	0.000002914	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16054	0.000001404	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16055	0.000002518	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16056	0.001497253	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP
16057	0.000016479	5201 → 49279 Len=625	192.168.68.101	192.168.68.110	UDP

Figura 4.6: Captura de paquetes con la herramienta `Wireshark`. Se muestra el momento exacto en el que se da el *roaming*, pasando de los paquetes que provienen del nodo 1 (verde) a los paquetes que provienen del nodo 2 (morado). Resaltado en el rectángulo amarillo, se indica el tiempo que `Wireshark` muestra que ha transcurrido entre el primer paquete del nodo 2 y el último paquete del nodo 1.

```
20240925-18:35:09.411106 Connecting to host 192.168.68.101, port 5201
20240925-18:35:09.411349 Reverse mode, remote host 192.168.68.101 is sending
20240925-18:35:09.411404 [  5] local 192.168.68.110 port 49279 connected to 192.168.68.101 port 5201
20240925-18:35:09.411450 [ ID]  Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
20240925-18:35:09.411494 [  5]  0.00-0.10   sec  65.9 KBytes  5.40 Mbits/sec  0.610 ms  0/108      (0%)
20240925-18:35:09.511102 [  5]  0.10-0.20   sec  62.9 KBytes  5.15 Mbits/sec  0.595 ms  0/103      (0%)
                                (...)
                                (...)
20240925-18:35:26.211955 [  5] 16.80-16.90  sec  58.6 KBytes  4.76 Mbits/sec  0.551 ms  0/96       (0%)
20240925-18:35:26.311161 [  5] 16.90-17.00  sec  0.00 Bytes   0.00 bits/sec   0.551 ms  0/0        (0%)
20240925-18:35:26.411336 [  5] 17.00-17.10  sec  0.00 Bytes   0.00 bits/sec   0.551 ms  0/0        (0%)
20240925-18:35:26.511123 [  5] 17.10-17.20  sec  0.00 Bytes   0.00 bits/sec   0.551 ms  0/0        (0%)
20240925-18:35:26.611222 [  5] 17.20-17.30  sec  47.0 KBytes  3.84 Mbits/sec  0.771 ms  319/396 (81%)
20240925-18:35:26.711043 [  5] 17.30-17.40  sec  59.2 KBytes  4.86 Mbits/sec  0.636 ms  0/97       (0%)
20240925-18:35:26.811186 [  5] 17.40-17.50  sec  65.9 KBytes  5.40 Mbits/sec  0.648 ms  0/108      (0%)
```

Figura 4.7: Salida de `iPerf3` en la que se muestran las métricas de rendimiento cada 0.1 segundos. Subrayado en rojo se indica la columna de paquetes perdidos sobre el total de recibidos. En el rectángulo rojo se muestra el momento exacto de *roaming*, en el que por 3 muestras consecutivas, es decir 300 ms, no se reciben datos.

Esto indica que aunque `Wireshark` desvela por medio de la captura de paquetes que en un intervalo de 45 ms se ha dado un cambio de AP, lo registrado a nivel de aplicación no dice lo mismo, pues iPerf3 desvela que se han perdido paquetes durante aproximadamente 300 ms.

4.3.5. Influencia en sistemas Linux del parámetro configurable *bgscan*

La Figura 4.8 muestra una repetición para la tarjeta Alfa1 en la que no se deshabilitó el escaneo en segundo plano que inicia el sistema cuando se cae por el umbral de los -70 dBm (consecuencia del valor del parámetro *bgscan* de `wpa_supplicant` introducido en la Sección 3.4.3). Observando la gráfica de la tasa de pérdida de paquetes, en el medio, se ratifica la importancia de los descubrimientos del primer experimento, pues el desempeño es bastante pobre, con tasas del 100 % de pérdidas durante varios segundos. Como ya se ha comentado en la Sección 3.4.3, si los dispositivos implementan funcionalidades concretas de los estándares 802.11k/v, es de vital importancia desactivar cualquier otro procedimiento que pueda entorpecer.

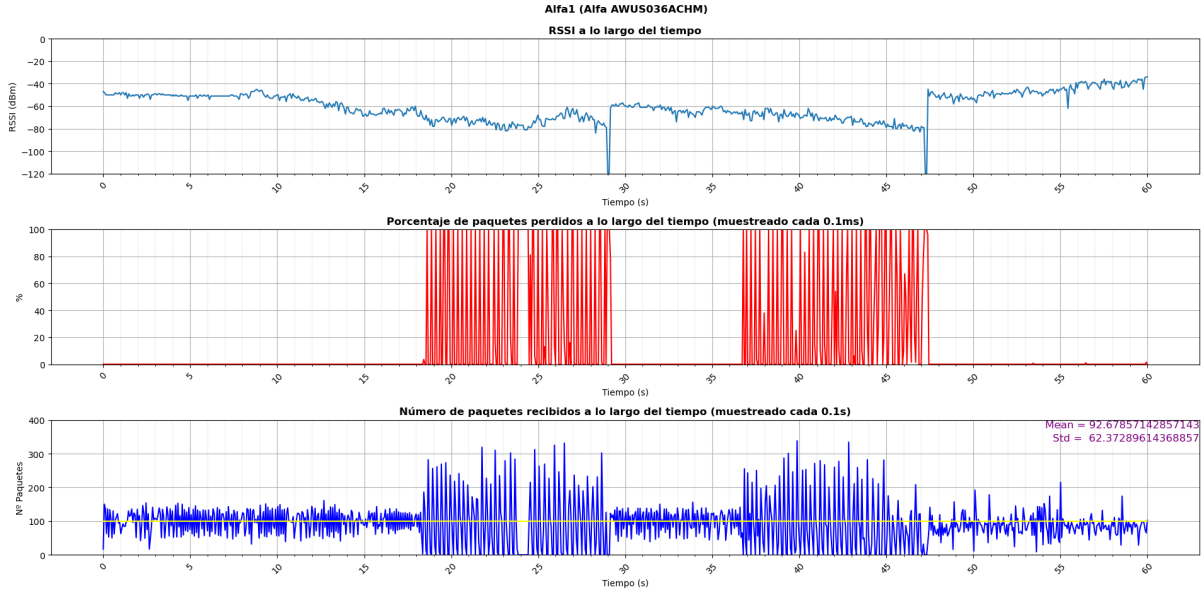


Figura 4.8: Recopilación de RSSI, tasa de pérdida de paquetes y número de paquetes recibidos para una repetición de la tarjeta Alfa1 en la que no se ha modificado el valor por defecto de *bgscan*.

4.3.6. Comportamiento anómalo de la tarjeta Intel

En las iteraciones con la tarjeta Intel, a pesar de que anuncia que implementa la funcionalidad *BSS Transition Management* en sus tramas *Association Request* exactamente igual que las tarjetas Alfa1 y Alfa2, se observó que en ningún caso el AP le enviaba la trama *BSS Transition Management Request* aconsejando el cambio de AP al nodo que presenta una mejor señal, lo cual es clave para que el *roaming* se produzca eficientemente. Esto desencadena los resultados que se muestran en la Figura 4.9, los cuales están muy lejos de ser satisfactorios, pues se puede observar que en ningún caso se produce el cambio de AP, quedándose asociado al primer nodo durante todo el recorrido. El motivo exacto por el cuál se da este comportamiento con la tarjeta Intel no se ha llegado a determinar, pero los resultados sí que pueden mejorar si para este caso específico no se desactiva el escaneo en segundo plano desencadenado por el valor por defecto del parámetro `bgscan`.

4.3.7. Transmisión de vídeo

En las iteraciones del recorrido en las que se transmitía vídeo desde la estación móvil a la estación base, se observó que la recepción era fluida a excepción de los momentos exactos en los que ocurría el *roaming*, en los que la imagen recibida presentaba congelamiento y pixelación por un intervalo de 1 a 2 segundos. La Figura 4.10 ilustra este fenómeno, donde se muestra en la Figura 4.10a la imagen nítida que se recibe en el punto justo antes de darse el *roaming* entre el nodo 1 y el nodo 2, la subida por las escaleras, y en la Figura 4.10b la imagen pixelada que se recibe una vez el dispositivo móvil se ha asociado ya al nodo 2 pero en la estación base la reproducción aún se está recuperando de los paquetes perdidos durante el proceso de *roaming*. Como al reproductor le falta información para decodificar correctamente los *frames* actuales en función de los *frames* previos (que se han perdido), se da la pixelación.

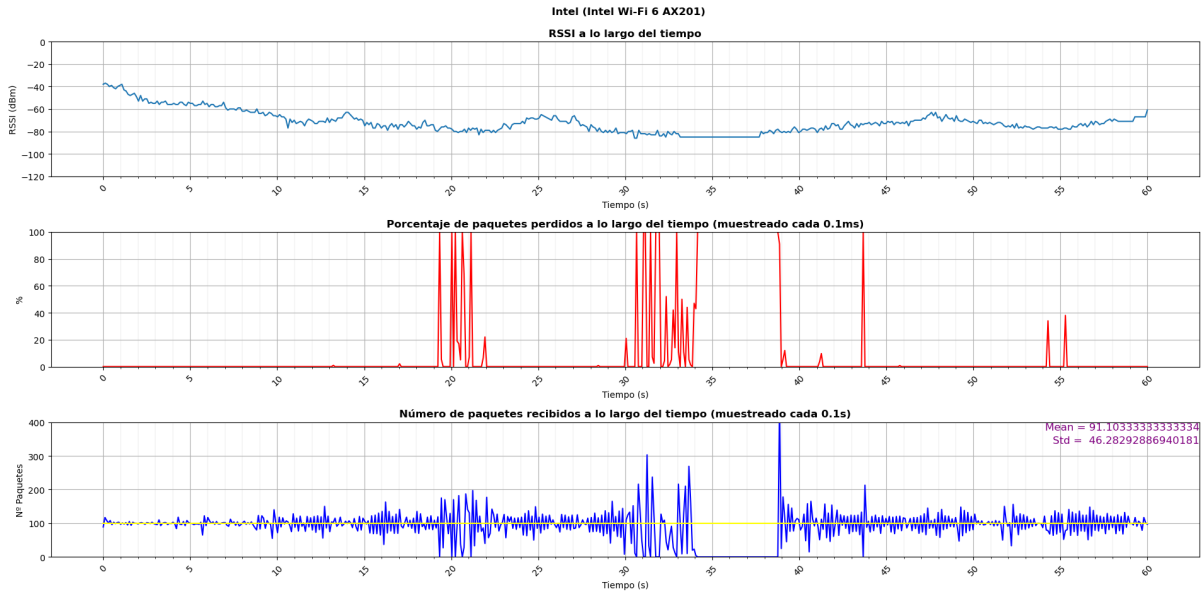
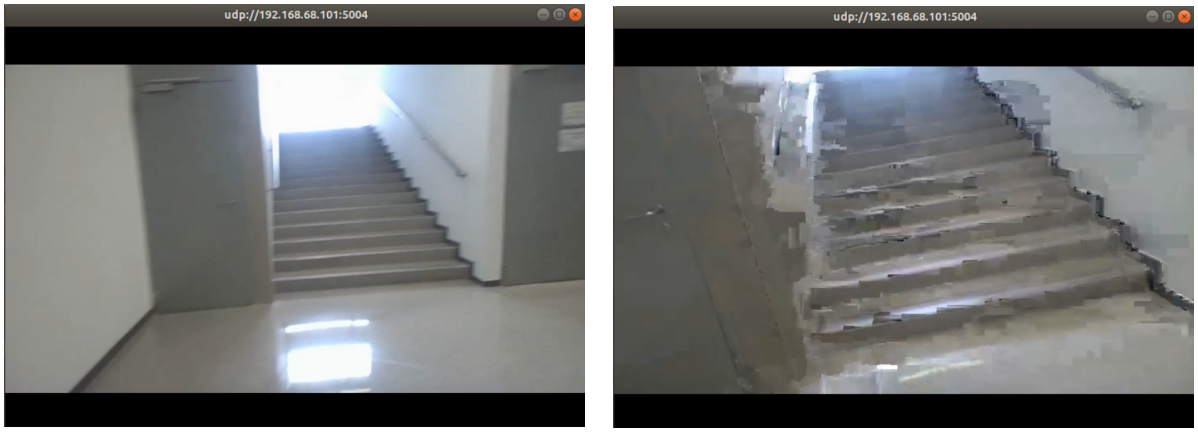


Figura 4.9: Recopilación de RSSI, tasa de pérdida de paquetes y número de paquetes recibidos para una repetición de la tarjeta Intel. Como se puede observar, el desempeño es pobre, pues no se produce en ningún momento un cambio de AP al no ejecutarse la funcionalidad *BSS Transition Management Request*.



(a) Imagen nítida justo antes de producirse el *roaming*.

(b) Imagen pixelada tras producirse el *roaming*.

Figura 4.10: Imagen recibida en la estación base en el punto donde se produce el *roaming* del nodo 1 al nodo 2.

Capítulo 5

Conclusiones

Este trabajo estudia la validez del *Fast Roaming* para las comunicaciones en aplicaciones robóticas en entornos inseguros o de difícil acceso para los humanos, especialmente aquellos entornos en los que es necesario desplegar varios puntos de acceso para dar cobertura de red adecuada. Concretamente, se ha caracterizado el *Fast Roaming*, que es uno de los beneficios que las redes WiFi Mesh comerciales venden para diferenciarse del resto de alternativas convencionales para cubrir grandes áreas de cobertura, como es el caso de los extensores de rango.

En el trabajo se han buscado y estudiado las fuentes de la literatura más notables que explican el *Fast Roaming*. Concretamente, se han leído los estándares 802.11k y 802.11v del IEEE, que explican en detalle cómo deben ser implementados los protocolos que favorecen al *Fast Roaming* tanto en los puntos de acceso como en las estaciones cliente. Este estudio ha permitido tener los conocimientos necesarios para verificar, mediante la captura y análisis de los paquetes que circulan por el medio inalámbrico, las funcionalidades concretas que implementan cada una de las interfaces de red disponibles en el trabajo. Además de esto, ha sido necesario un estudio y mitigación de diversos puntos de fallo en el sistema operativo que aloja a las interfaces de red, identificando ciertos módulos que es necesario desactivar cuando el hardware implementa los protocolos de los estándares. Esto recalca la importancia de una correcta administración y depuración de sistemas cuando se trabaja con hardware externo al equipo.

Tras la completa caracterización del intercambio de tramas que existe entre los puntos de acceso y las estaciones cliente en una situación de *Fast Roaming*, se ha estudiado la validez de la red WiFi Mesh para una aplicación robótica de teleoperación basada en transmisión de vídeo. Los resultados obtenidos muestran que, en una situación de señal óptima en el cliente, la red WiFi Mesh muestra un rendimiento similar al de cualquier red inalámbrica convencional, con unos valores de latencia, *jitter* y tasa de pérdida de paquetes dentro de los rangos habituales en cualquier red, que son suficientemente buenos para soportar una aplicación de estas características (una latencia inferior a 100 ms, un *jitter* por debajo de los 30 ms y una tasa de pérdidas menor al 1 %). No obstante, en el momento crítico en el que la señal empeora y sucede el *Fast Roaming*, se ha demostrado que por un intervalo de entre 300 y 400 ms hay una pérdida de paquetes recibidos en la estación móvil, con una tasa del 100 % en este intervalo, y que en la reproducción de vídeo en la estación base se da una congelación y pixelación de la señal entre 1 y 2 segundos. Como conclusión, se cree que el *Fast Roaming* hace más eficiente el proceso de *roaming*, pero es necesario conocer, si se desean tomar las

medidas de prevención oportunas, que en aplicaciones con fuertes restricciones de respuesta en tiempo real es inevitable ese pequeño corte en la comunicación, al menos con los protocolos estudiados en este trabajo. Si la aplicación tiene cierto grado de tolerancia, el *Fast Roaming* que se ha estudiado puede ser suficientemente útil.

Como trabajo futuro se puede continuar con un nuevo experimento que surgió en la fase final del trabajo, basado en el control de un robot con ROS[1] que fue cedido del grupo de robótica, RoPeRT. Concretamente, esta última parte proporcionó un aprendizaje valioso en lo que se refiere a administración de sistemas, pues a pesar de utilizar las mismas tarjetas de red que durante todo el proyecto, estas presentaban diferencias notables de comportamiento al ser instaladas en el sistema operativo con ROS que fue prestado, incluso con el mismo *kernel* presente en este sistema que el utilizado en el Lenovo del trabajo. Básicamente, se descubrió tras dedicar tiempo de nuevo a depurar el sistema y analizar el tráfico de red, que las tarjetas no anunciaban la funcionalidad *BSS Transition Management* y por tanto el *roaming* no podía ser eficiente. Este aprendizaje destaca la importancia de diseñar entornos de prueba fácilmente replicables y controlados, para garantizar la consistencia y fiabilidad en los desarrollos de un proyecto. Si en futuros trabajos se consigue subsanar este problema, se podrá avanzar con todo lo aprendido hasta ahora y abrir nuevas posibilidades de aplicación en escenarios reales.

[1] https://www.ros.org/ (accedido: 28/11/2024)

Bibliografía

[1] Moniruzzaman, MD & Rassau, Alexander & Chai, Douglas & Islam, Syed. (2021). *Teleoperation methods and enhancement techniques for mobile robots: A comprehensive survey. Robotics and Autonomous Systems.* 150. 103973. 10.1016/j.robot.2021.103973.

[2] Opiyo, S., Zhou, J., Mwangi, E. *et al. A Review on Teleoperation of Mobile Ground Robots: Architecture and Situation Awareness. Int. J. Control Autom. Syst.* 19, 1384–1407 (2021). https://doi.org/10.1007/s12555-019-0999-z.

[3] Kamtam, Sidharth & Lu, Qian & Bouali, Faouzi & Haas, Olivier & Birrell, Stewart. (2024). *Network Latency in Teleoperation of Connected and Autonomous Vehicles: A Review of Trends, Challenges, and Mitigation Strategies. Sensors.* 24. 3957. 10.3390/s24123957.

[4] KLiu, Yu & Tong, Kin-Fai & Qiu, Xiangdong & Liu, Ying & Ding, Xuyang. (2017). *Wireless Mesh Networks in IoT Networks.* 183-185. 10.1109/iWEM.2017.7968828.

[5] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 1: Radio Resource Measurement of Wireless LANs" in IEEE Std 802.11k-2008 (Amendment to IEEE Std 802.11-2007) , vol., no., pp.1-244, 12 June 2008, doi: 10.1109/IEEESTD.2008.4544755.

[6] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 8: IEEE 802.11 Wireless Network Management" in IEEE Std 802.11v-2011 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11w-2009, IEEE Std 802.11n-2009, IEEE Std 802.11p-2010, and IEEE Std 802.11z-2010) , vol., no., pp.1-433, 9 Feb. 2011, doi: 10.1109/IEEESTD.2011.5716530.

[7] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition," in IEEE Std 802.11r-2008 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008) , vol., no., pp.1-126, 15 July 2008, doi: 10.1109/IEEESTD.2008.4573292.

[8] M. S. Gast, *"802.11 wireless networks: The definitive guide"*, 2nd edition, O'Reilly, April 2005.

[9] A. Vlavianos, L. K. Law, I. Broustis, S. V. Krishnamurthy and M. Faloutsos, .ᴬssessing link quality in IEEE 802.11 Wireless Networks: Which is the right metric?,"2008 IEEE 19th

International Symposium on Personal, Indoor and Mobile Radio Communications, Cannes, France, 2008, pp. 1-6, doi: 10.1109/PIMRC.2008.4699837.

[10] 7SIGNAL. *"Mysteries of Wi-Fi roaming revealed" [White paper].* https://www.7signal.com/resources/whitepapers/9-mysteries-of-wi-fi-roaming-revealed-part-1 (accedido: 26/11/2024).

Anexos

Anexo A

Tramas del estándar 802.11 en detalle

Existen tres tipos principales de tramas. Tramas de datos, tramas de control y tramas de gestión. Las tramas de datos transportan los datos de aplicación de una estación a otra. Las tramas de control se utilizan para controlar el acceso al medio y para la confirmación de tramas. Solo contienen una cabecera y el FCS (*Frame Check Sequence*), sin datos. Las tramas de control y datos trabajan juntas para entregar los datos de manera confiable de estación a estación. Por último, las tramas de gestión, que han sido las estudiadas a fondo en este trabajo, realizan funciones de supervisión; se utilizan para unirse y salir de redes inalámbricas y para mover asociaciones de un punto de acceso a otro.

A.1. Tipos de tramas de gestión

Existen varios tipos de tramas de gestión que se utilizan para diversas funciones de mantenimiento de la capa de enlace.

A.1.1. Beacon

Las tramas *Beacon* anuncian la existencia de una red y son una parte importante de muchas tareas de mantenimiento de la misma. Se transmiten a intervalos regulares para permitir que las estaciones móviles encuentren e identifiquen una red, así como para que coincidan los parámetros necesarios para unirse a esta. En una red inalámbrica, el punto de acceso es el responsable de transmitir las tramas *Beacon*, y dado que toda la comunicación se realiza a través de este, las estaciones de la red deben estar lo suficientemente cerca como para escuchar las *Beacons*.

A.1.2. Probe Request

Las tramas *Probe Request* son usadas por las estaciones móviles para escanear un área en búsca de redes 802.11 existentes. Esta trama contiene dos campos: el SSID y las tasas de transmisión soportadas por la estación móvil. Las estaciones que reciben las *Probe Request* utilizan esta información para determinar si la estación móvil puede unirse a la red. Para lograr una unión exitosa, la estación móvil debe ser compatible con las tasas de datos que ofrece la red y debe querer unirse a la red identificada por el SSID. El campo del SSID puede estar configurado con el SSID de una red específica o configurado para unirse a cualquier red

compatible, marcando este campo como *broadcast* para que contesten todas las estaciones en el rango en lugar de aquella que coincida con un SSID específico.

A.1.3. Probe Response

Si una trama *Probe Request* encuentra una red con parámetros compatibles, la estación compatible envía una trama *Probe Response* (el punto de acceso en el ámbito de este proyecto). Esta trama cuenta con los mismos parámetros que una trama *Beacon*, lo que permite a las estaciones móviles conocer los parámetros de la red y unirse a ella.

A.1.4. Autenticación

Para autenticarse contra un punto de acceso, las estaciones móviles intercambian con este una serie de tramas de autenticación, que varían dependiendo del algoritmo de autenticación empleado. La secuencia más simple consistiría en una trama *Authentication Request* por parte de la estación móvil hacia el punto de acceso y una trama *Authentication Response* por parte del punto de acceso hacia la estación móvil. Este intercambio mínimo es obligatorio aunque no se haya configurado autenticación en la red, como es el caso de este proyecto.

A.1.5. Association Request

Una vez que una estación móvil identifica una red compatible y se autentica contra el punto de acceso, le envía a este una trama *Association Request* para unirse a la red.

A.1.6. Association Response

El punto de acceso debe responder con una trama *Association Response* cuando recibe una trama *Association Request* por parte de una estación móvil. Antes de responder, el punto de acceso revisa el SSID, las tasas de transmisión y otra información relevante presente en la trama *Association Request* para verificar que los parámetros que envía la estación móvil coinciden con los requerimientos de la red. Si todo es correcto la trama *Association Response* incluirá el código de estado de "operación completada correctamente", y en caso contrario, el de "asociación denegada".

A.1.7. Reasociación

Existen dos tramas, *Reassociation Request* y *Reassociation Response*, que son equivalentes a las tramas *Association Request* y *Association Response* y que son las utilizadas por las estaciones móviles cuando van a asociarse con otro punto de acceso dentro de la misma red, esto es, dentro del mismo SSID. Las tramas *Association Request* y *Reassociation Request* únicamente se diferencian en que esta última incluye la dirección física del punto de acceso previo, para que el nuevo punto de acceso pueda conectar con el anterior y transferir distintos datos de asociación.

A.1.8. Desasociación y desautenticación

La trama *Disassociation* es enviada para poner fin a una asociación, y la trama *Deauthentication* es enviada para poner fin a una autenticación. Ambas tramas incluyen un código con el motivo y pueden ser enviadas tanto por un nodo móvil como por un punto de acceso.

A.2. Diagrama de estados general del estándar 802.11

Las tramas que se pueden transmitir cambian dependiendo del estado de asociación y autenticación en el que se encuentra una estación. Las estaciones pueden estar autenticadas o sin autenticar y pueden estar asociadas o sin asociar. Estas dos variables se pueden combinar en tres estados, resultando en la jerarquía de desarrollo de redes 802.11:

1. Estado inicial: estación no autenticada y no asociada.
2. Estación autenticada pero no asociada.
3. Estación autenticada y asociada.

Cada estado representa una etapa en el desarrollo de una conexión 802.11. Todas las estaciones móviles comienzan en el estado 1, y los datos solo pueden ser transmitidos cuando se está en el estado 3. La Figura A.1 ilustra este diagrama general de estados para la transmisión de tramas del estándar 802.11. Cada una de las tramas introducidas en la Sección A.1 pertenecen a una clase. Las tramas de clase 1 pueden ser transmitidas en el estado 1. En el estado 2 se pueden transmitir tramas de clase 1 y 2. En el estado 3 se pueden transmitir tramas de clase 1, 2 y 3. La Tabla A.1 define a qué clase pertenecen cada una de las tramas.

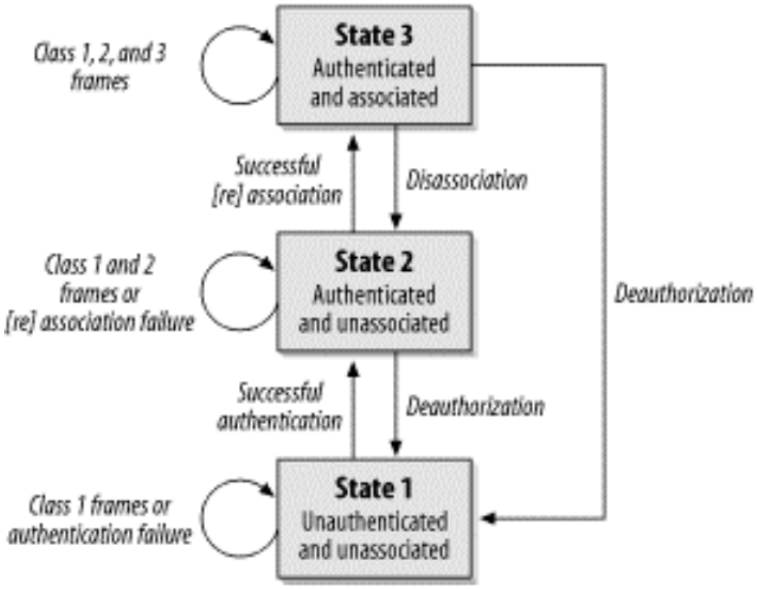


Figura A.1: Diagrama de estados general del estándar 802.11. Fuente: [8]

Clase	**Trama**
1	*Beacon*, *Probe Request/Response*, *Authentication Request/Response*, *Deauthentication*
2	*Association Request/Response*, *Reassociation Request/Response*, *Disassociation*
3	*Deauthentication*

Tabla A.1: Clases de las distintas tramas de gestión del estándar 802.11.

Anexo B

Herramientas

B.1. iPerf3

La herramienta `iPerf3` permite realizar mediciones activas del ancho de banda máximo alcanzable en redes IP. Permite ajustar diversos parámetros relacionados con el tiempo, los búferes y los protocolos (TCP, UDP, SCTP con IPv4 e IPv6). Para cada prueba, informa sobre el ancho de banda, la pérdida de paquetes y otros parámetros[1]. Principalmente, consiste en un nodo que actúa como servidor, escuchando constantemente nuevas conexiones, y un segundo nodo que actúa como cliente, el cual cuando es lanzado conecta con el servidor y comienza la transmisión de datos según se haya indicado en el comando utilizado.

B.1.1. Comandos utilizados en el experimento del Capítulo 3

En el experimento introducido en el Capítulo 3 se pretendía como objetivo caracterizar las funcionalidades de *Fast Roaming* que implementaban tanto los puntos de acceso como las tres tarjetas de red disponibles. No se requería una transmisión de datos compleja, únicamente observar si el hecho de que existiera un flujo de datos o no durante el recorrido influía en el momento y circunstancias bajo las que se daba el *roaming*. Estos son los comandos necesarios para comenzar una transmisión de datos simple con `iPerf3` entre el servidor (estación base fija) y el cliente (estación móvil):

- Servidor: **"`iperf3 -s`"**.
  - **`-s`**: Ejecutar `iPerf3` en modo servidor.
- Cliente: **"`iperf3 -c <ip_servidor> -t <tiempo>`"**.
  - **`-c <ip_servidor>`**: Ejecutar `iPerf3` en modo cliente y conectar contra el servidor especificado por `<ip_servidor>`.
  - **`-t <tiempo>`**: El tiempo en segundos durante el que transmitir.

Por defecto, si no se indica lo contrario con la opción `-R`, es el cliente quien envía datos al servidor. También por defecto, si no se indica lo contrario, el protocolo de la capa de transporte utilizado es TCP. Y si no se indica una tasa de transmisión concreta, `iPerf3` trata de transmitir al máximo ancho de banda que la red permite en conexiones TCP.

[1] https://iperf.fr/ (accedido: 28/11/2024)

B.1.2. Comandos utilizados en el experimento del Capítulo 4

En el experimento introducido en el Capítulo 4 se buscó enviar paquetes siempre al mismo ritmo (1 paquete por ms) independientemente de la tasa de transmisión empleada (1, 5 y 11.68 Mbps), que la conexión fuera UDP y que el sentido de envío fuera desde el servidor (estación base) hacia la estación móvil (cliente) para simular el flujo de datos que podría darse en una teleoperación. Estos son los comandos necesarios para comenzar una transmisión con dichas características:

- Servidor: **"`iPerf3 -s`"**.
  - **`-s`**: Ejecutar `iPerf3` en modo servidor.
- Cliente: **"`iperf3 -c <ip_servidor> -u -b <tasa> -l <tamaño> -t <tiempo> -i <intervalo> --forceflush -R`"**.
  - **`-c <ip_servidor>`**: Ejecutar `iPerf3` en modo cliente y conectar contra el servidor especificado por `<ip_servidor>`.
  - **`-u`**: Usar UDP en lugar de TCP.
  - **`-b <tasa>`**: Define la tasa de transmisión objetivo. Para establecer 1, 5 o 11.68 Mbps, se tendría `-b 1M`, `-b 5M` o `-b 1.68M`, respectivamente.
  - **`-l <tamaño>`**: Tamaño que ocupan los datos en cada uno de los paquetes a transmitir. Para conseguir enviar siempre 1 paquete por ms con las tasas de 1, 5 o 11.68 Mbps, se tendría `-l 125`, `-l 625` o `-l 1460`, respectivamente. Si no se indica unidad, `iPerf3` lo toma por defecto como Bytes.
  - **`-t <tiempo>`**: El tiempo en segundos durante el que transmitir.
  - **`-i <intervalo>`**: Establece el intervalo de tiempo en segundos entre los informes periódicos de ancho de banda, *jitter* y pérdida de paquetes.
  - **`--forceflush`**: Fuerza a mostrar por salida estándar los informes en cada periodo.
  - **`-R`**: Ejecutar en modo reverso (servidor envía, cliente recibe).

B.1.3. Jitter

Analizando el código fuente de `iPerf3` para analizar cómo realiza el cálculo del *jitter*, en los comentarios que los desarrolladores han introducido en el código se puede encontrar que el cálculo está basado en las secciones 6.3.1 y A.8 del RFC 1889[2].

B.2. FFmpeg

`FFmpeg` permite crear una transmisión de vídeo entre dos dispositivos por medio de línea de comandos. En el ámbito de este proyecto se han realizado experimentos con los protocolos RTP y UDP, para comparar si el protocolo utilizado influía en el rendimiento global de la transmisión y porque con `FFmpeg` estos protocolos simplifican el proceso de creación de una

[2]https://datatracker.ietf.org/doc/html/rfc1889 (accedido: 28/11/2024)

transmisión de vídeo. Para una transmisión de vídeo con FFmpeg, se precisa una máquina actuando como servidor, la cual va a recibir el vídeo, y una máquina como cliente, la cual envía el vídeo. En este proyecto la estación base ha sido la que recibe vídeo, por tanto, el servidor. Los comandos utilizados para levantar un servidor RTP o un servidor UPD son los siguientes:

- RTP: "`ffplay -protocol_whitelist ''file,crypto,data,udp,rtp'' -fflags nobuffer -flags low_delay -i video.sdp`".
  - `-protocol_whitelist ''file,crypto,data,udp,rtp''`: lista de protocolos permitidos. Incluye `rtp`.
  - `-fflags nobuffer`: Reduce la latencia introducida por el *buffering* durante el análisis inicial del flujo de entrada.
  - `-fflags low_delay`: reduce la latencia en operaciones de streaming o reproducción de vídeo y audio.
  - `-i video.sdp`: este argumento indica la entrada multimedia, que en este caso es un archivo `.sdp`. Este es un archivo que se genera en el cliente, como se indicará a continuación, que contiene la información de la transmisión de vídeo, pues el protocolo RTP utiliza el protocolo SDP para negociar las características de la comunicación que se va a levantar entre las dos máquinas.
- UDP: "`ffplay -fflags nobuffer -flags low_delay udp://<ip_servidor>:<puerto>`".
  - `udp://<ip_servidor>:<puerto>`: indica la IP y el puerto en los que escuchar conexiones entrantes del protocolo UPD. En este caso, la IP debe ser la IP local de la estación base dentro de la red WiFi Mesh.

Una vez que el servidor está escuchando para la conexión entrante de vídeo, desde el cliente se puede comenzar la transmisión del vídeo procedente de su cámara integrada. Los comandos necesarios son los siguientes:

- RTP: "`ffmpeg -f v4l2 -video_size 1280x720 -framerate 30 -input_format mjpeg -i /dev/video0 -vcodec libx264 -preset ultrafast -tune zerolatency -b:v 1M -f rtp rtp://<ip_servidor>:<puerto> -sdp_file video.sdp`".
  - `-f v4l2`: especifica el formato de entrada. `v4l2` es una API para capturar video de dispositivos como cámaras web en sistemas Linux.
  - `-video_size 1280x720`: configura la resolución de vídeo de la cámara.
  - `-framerate 30`: establece la velocidad de fotogramas del video capturado.
  - `-input_format mjpeg`: define el formato de entrada que la cámara está utilizando para enviar el video (secuencia de imágenes JPEG comprimidas).
  - `-i /dev/video0`: especifica el dispositivo de entrada (ruta del dispositivo de vídeo en Linux).
  - `-vcodec libx264`: selecciona el códec de video que se utilizará para la compresión (H.264 en este caso).

- `-preset ultrafast`: priorización de velocidad sobre la compresión. Genera archivos más grandes, pero es ideal para transmisiones en vivo donde la latencia debe ser mínima.
- `-tune zerolatency`: reduce los búferes internos del codificador para minimizar la latencia.
- `-b:v 1M`: establece la tasa de bits del video.
- `-f rtp`: especifica el formato de salida.
- `rtp://<ip_servidor>:<puerto>`: para indicar la IP y puerto en los que está escuchando el servidor RTP.
- `-sdp_file video.sdp`: para generar el archivo `.sdp` necesario en el servidor (estación base) con las características de transmisión.

- UDP: "`ffmpeg -f v4l2 -video_size 1280x720 -framerate 30 -input_format mjpeg -i /dev/video0 -vcodec libx264 -preset ultrafast -tune zerolatency -b:v 1M -f mpegts udp://<ip_servidor>:<puerto>`".

Los valores que se observan de número de fotogramas (`-framerate 30`) y tasa de bits (`-b:v 1M`) se fueron modificando a lo largo de las distintas iteraciones del experimento del Capítulo 4, para analizar si influían en el rendimiento, y se observó que con los valores de 10 FPS (`-framerate 10`) y 1Mbps (`-b:v 1M`) se obtenía una imagen suficientemente buena en la estación base y además los instantes de congelación y pixelación eran más pequeños que con otras configuraciones, como ya se ha explicado en el Capítulo 4.

B.3. Configuración de tarjeta de red en modo monitor

Para poder analizar el tráfico que circula por el medio inalámbrico se debe configurar la tarjeta de red en modo monitor, que es el modo que ofrecen las tarjetas de red para capturar y analizar todo el tráfico que pasa por un canal específico, independientemente de si está destinado a la máquina en la que está instalada la tarjeta. Primero es necesario conocer si la tarjeta permite el modo monitor, lo cual se puede consultar por medio del comando '`iw list`'. Este comando muestra información de las tarjetas de red instaladas en el sistema y en la sección '`Supported interface modes`' debe aparecer la línea '`* monitor`'. En este caso, ya se puede configurar la tarjeta en este modo por medio de la siguiente secuencia de comandos:

1. **`sudo ifconfig <iface> down`**: Desactivar la interfaz con nombre `<iface>`. `<iface>` es el nombre de la interfaz de red que muestra el comando '`ip a`' o '`iwconfig`'.
2. **`sudo iwconfig <iface> mode monitor`**: Configurar la interfaz con nombre `<iface>` en modo monitor.
3. **`sudo ifconfig <iface> up`**: Activar la interfaz con nombre `<iface>`.
4. **`sudo iw <phy> set channel ‘‘<canal>’’ ‘‘<ancho>’’`**: Configurar el canal y el ancho de canal que la tarjeta debe analizar. `<phy>` es el nombre de la interfaz de red que muestra el comando '`iw dev`'.

B.4. Wireshark

Con la herramienta `Wireshark` se pueden analizar gran cantidad de paquetes y tramas circulando por el medio inalámbrico, y es necesario aplicar diversos filtros de visualización para centrarse en los datos de interés. En el caso de la Figura 4.6, en el que se ilustra cómo se realiza el cálculo de tiempo de *roaming* por medio de este procedimiento, se utilizó el siguientes filtro:

```
(((udp) && (udp.srcport == 5201)) &&
(data.len == 625)) && (wlan.ra == 00:c0:ca:b2:bc:1a)
```

- `udp`: filtra por los paquetes transmitidos únicamente por UDP.
- `udp.srcport == 5201`: filtra por los paquetes que salieron por el puerto 5201 en el nodo origen (el puerto que utiliza el servidor `iPerf3`)
- `data.length == 625`: filtra por los paquetes cuya longitud de datos es exactamente 625 Bytes (para obtener únicamente los que se están transmitiendo con `iPerf3`, que dependiendo de la tasa serán de 125, 625 o 1460 Bytes)
- `wlan.ra == 00:c0:ca:b2:bc:1a`: filtra por los paquetes que únicamente van dirigidos al dispositivo con la dirección MAC indicada, esto es, la estación móvil. Si se usa otra tarjeta, simplemente hay que cambiar la dirección MAC.

Posteriormente, para aplicar colores y diferenciar más fácilmente el punto de acceso por el cuál se están recibiendo los datos, basta con ir a `Visualización -> Reglas de coloreado` en la barra de herramientas de la interfaz de `Wireshark` y añadir el filtro `wlan.ta == <MAC>` donde `<MAC>` es la dirección MAC del nodo.

Anexo C

Depuración del sistema

C.1. Parámetro bgscan de wpa_supplicant

C.1.1. Valor incrustado en el código del NetworkManager

Como se ha comentado en la Sección 3.4.3, el valor del parámetro `bgscan` con el que el `wpa_supplicant` trabaja por defecto, está incrustado en las líneas de código del `NetworkManager`. La Figura C.1 muestra las lineas específicas, 614 y 628. Se trata del código fuente de la versión `1.36.6` del `NetworkManager`, que es la instalada en el Lenovo con el que se ha trabajado.

```
609     /* Default to a very long bgscan interval when signal is OK on the assumption
610      * that either (a) there aren't multiple APs and we don't need roaming, or
611      * (b) since EAP/802.1x isn't used and thus there are fewer steps to fail
612      * during a roam, we can wait longer before scanning for roam candidates.
613      */
614     bgscan = "simple:30:-70:86400";
615
616     /* If using WPA Enterprise, Dynamic WEP or we have seen more than one AP use
617      * a shorter bgscan interval on the assumption that this is a multi-AP ESS
618      * in which we want more reliable roaming between APs. Thus trigger scans
619      * when the signal is still somewhat OK so we have an up-to-date roam
620      * candidate list when the signal gets bad.
621      */
622     if (nm_setting_wireless_get_num_seen_bssids(s_wifi) > 1
623         || ((s_wsec = nm_connection_get_setting_wireless_security(connection))
624             && NM_IN_STRSET(nm_setting_wireless_security_get_key_mgmt(s_wsec),
625                             "ieee8021x",
626                             "wpa-eap",
627                             "wpa-eap-suite-b-192")))
628         bgscan = "simple:30:-65:300";
629
630     return nm_supplicant_config_add_option(self, "bgscan", bgscan, -1, FALSE, error);
631 }
```

Figura C.1: Valores de `bgscan` "hardcodeados" en el código fuente de `NetworkManager`. Se encuentran en la función `nm_supplicant_config_add_bgscan` del archivo '`src/core/supplicant/nm-supplicant-config.c`.'

C.1.2. Sintaxis

El parámetro bgscan usa el siguiente formato: "<módulo bgscan>:<parámetros del módulo>"[1]. Los módulos disponibles son los siguientes:

- simple - Escaneos periódicos en segundo plano dependiendo del nivel de intensidad de señal. La sintaxis si se usa este módulo es
  bgscan="simple:<intervalo corto>:<umbral>:<intervalo largo>".
  - <intervalo corto>: cada cuántos segundos realizar escaneos si la señal cae por debajo de <umbral>.
  - <umbral>: si se obtiene un nivel de intensidad de señal menor que <umbral>, comenzará un escaneo y este se repetirá cada <intervalo corto> segundos siempre y cuando el nivel de señal no se recupere superando el umbral.
  - <intervalo largo>: cada cuántos segundos realizar escaneos si la señal está por encima de <umbral>.

  Ejemplo: bgscan="simple:30:-70:300"
- learn - Aprender los canales utilizados por la red e intentar evitar escaneos en canales no utilizados (experimental, en desarrollo).

C.1.3. Establecer manualmente el valor de bgscan con wpa_cli

Se intentó, sin éxito, establecer un valor persistente de bgscan en el sistema por medio de los archivos de configuración de NetworkManager y wpa_supplicant. Este valor es bgscan="simple:30:-120:86400", forzando así con un valor umbral de -120 dBm, que nunca se dispare un escaneo en segundo plano desde el wpa_supplicant (nunca se va a llegar a -120 dBm si los protocolos de los estándares 802.11k/v funcionan correctamente). Se desconoce el motivo por el cual por los archivos de configuración no se reconoce el parámetro bgscan para así modificar el valor por defecto que establece NetworkManager. Buscando otras alternativas, se encontró que el paquete de wpa_supplicant aporta la utilidad wpa_cli, que es una herramienta de línea de comandos que permite interactuar con el demonio wpa_supplicant y gestionar la red a nivel de usuario. Por medio de esta utilidad, se consiguió modificar el valor de bgscan al comienzo de cada experimento. La secuencia de comandos que lo permiten se ilustran en la Figura C.2. El comando get_network permite preguntar por distintos parámetros, así es como se puede ver que el valor actual de bgscan es "simple:30:-70:86400", y por medio del comando set_network se puede establecer un nuevo valor para el parámetro que se desee. El entero que sucede a ambos comandos, un 0 en este caso, es para identificar a la tarjeta de red que se quiere gestionar, en caso de que el sistema disponga de varias.

C.2. Timeout durante la asociación/autenticación

Como se introduce en la Sección 3.4.2, en los *logs* del sistema se registra un timeout en la estación móvil durante la fase de asociación o autenticación contra el punto de acceso.

[1] https://web.mit.edu/freebsd/head/contrib/wpa/wpa_supplicant/wpa_supplicant.conf (accedido: 24/11/2024)

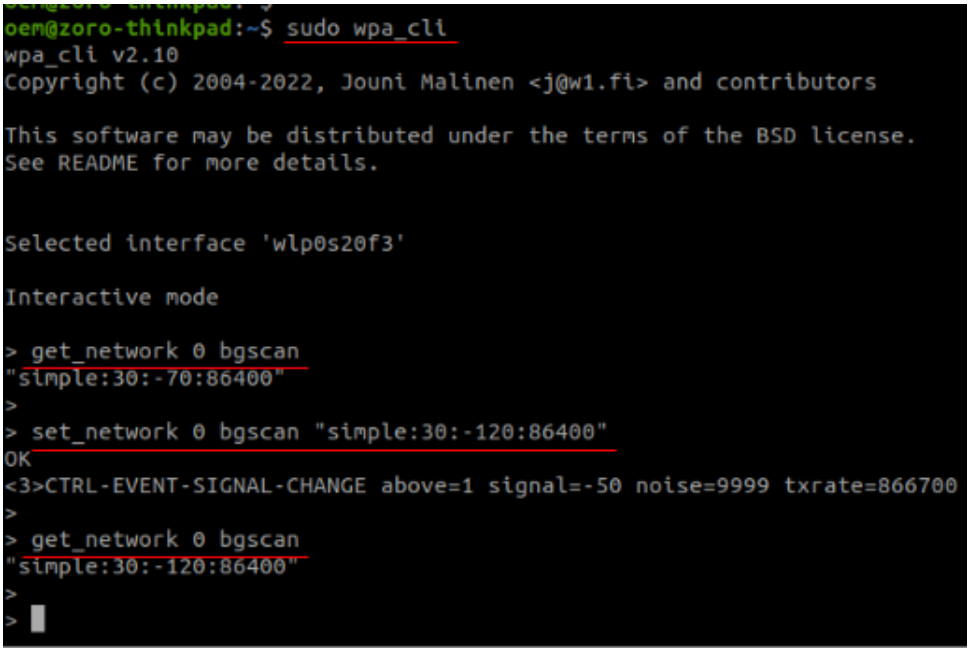


Figura C.2: Modificación del valor de `bgscan` por medio de `wpa_cli`. Subrayado en rojo los comandos utilizados.

Para encontrar el motivo exacto, se busco en el código fuente de `wpa_supplicant` (versión 2.10) por las cadenas clave `ASSOC_TIMED_OUT`, `AUTH_TIMED_OUT`, `SME: Association timed out` y `SME: Authentication timed out`, las cuales aparecen en los *logs* cuando ocurre este problema, como se puede observar en la Figura 3.6. Esto llevó a la línea de código exacta del `wpa_supplicant` en la que está la instrucción de debug con la cadena `SME: Association timed out`, la cual se ilustra en la Figura C.3. Yendo hacia atrás, siguiendo toda la traza de funciones que terminan en ese punto, se encontró que el origen del evento de timeout proviene de un mensaje que llega del *driver*. El `wpa_supplicant` recibe este mensaje por medio de `netlink`, el protocolo de comunicación de Linux que se encarga de la interacción entre procesos del espacio de usuario y de *kernel*, como se ha introducido en la Sección 3.4.1. Dado que el mensaje proviene del *driver*, gestionado por el *kernel* de Linux, se continuó profundizando en la traza de ejecución en el código fuente del *kernel* de Linux, versión 6.8.0. Finalmente se encontró la constante que define este tiempo de timeout, como se puede observar en la Figura C.4.

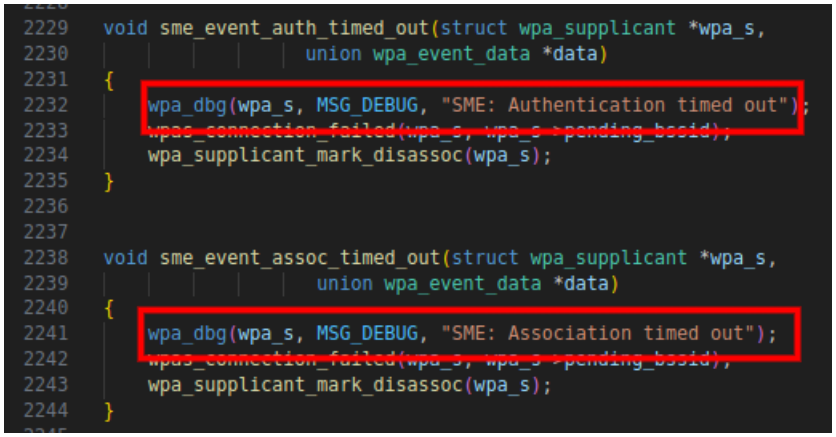
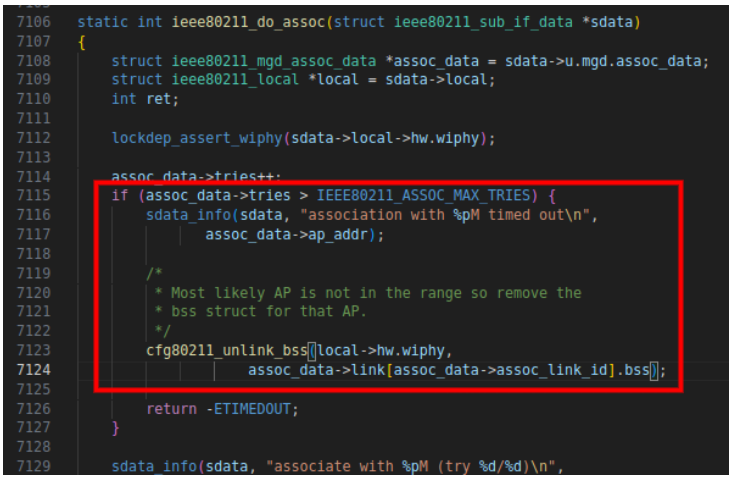
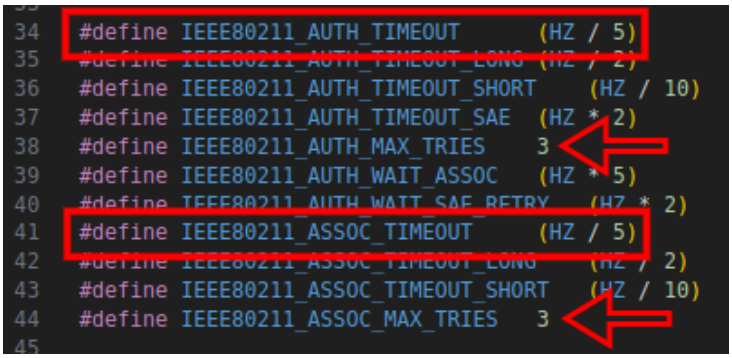


Figura C.3: Funciones y líneas de código exactas del `wpa_supplicant` que registran el timeout (resaltadas con el rectángulo rojo). Se encuentran en el archivo '`wpa_supplicant/sme.c`'.



```
7106  static int ieee80211_do_assoc(struct ieee80211_sub_if_data *sdata)
7107  {
7108      struct ieee80211_mgd_assoc_data *assoc_data = sdata->u.mgd.assoc_data;
7109      struct ieee80211_local *local = sdata->local;
7110      int ret;
7111
7112      lockdep_assert_wiphy(sdata->local->hw.wiphy);
7113
7114      assoc_data->tries++;
7115      if (assoc_data->tries > IEEE80211_ASSOC_MAX_TRIES) {
7116          sdata_info(sdata, "association with %pM timed out\n",
7117                     assoc_data->ap_addr);
7118
7119          /*
7120           * Most likely AP is not in the range so remove the
7121           * bss struct for that AP.
7122           */
7123          cfg80211_unlink_bss(local->hw.wiphy,
7124                              assoc_data->link[assoc_data->assoc_link_id].bss);
7125
7126          return -ETIMEDOUT;
7127      }
7128
7129      sdata_info(sdata, "associate with %pM (try %d/%d)\n",
```

(a) En el rectángulo rojo, se indica el momento en el que salta el timeout debido a sobrepasar un número de intentos máximo. Además, en la línea 7123, se observa como se llama a la función que posteriormente provoca que el AP vaya a ser ignorado.



```
34  #define IEEE80211_AUTH_TIMEOUT      (HZ / 5)
35  #define IEEE80211_AUTH_TIMEOUT_LONG (HZ / 2)
36  #define IEEE80211_AUTH_TIMEOUT_SHORT    (HZ / 10)
37  #define IEEE80211_AUTH_TIMEOUT_SAE  (HZ * 2)
38  #define IEEE80211_AUTH_MAX_TRIES    3
39  #define IEEE80211_AUTH_WAIT_ASSOC   (HZ * 5)
40  #define IEEE80211_AUTH_WAIT_SAE_RETRY   (HZ * 2)
41  #define IEEE80211_ASSOC_TIMEOUT     (HZ / 5)
42  #define IEEE80211_ASSOC_TIMEOUT_LONG    (HZ / 2)
43  #define IEEE80211_ASSOC_TIMEOUT_SHORT   (HZ / 10)
44  #define IEEE80211_ASSOC_MAX_TRIES   3
45
```

(b) Constantes que establecen el tiempo de timeout tanto de autenticación como de asociación (indicadas en los rectángulos rojos) y el número de intentos máximos (indicadas con las flechas). La constante `HZ`, de la cual dependen los tiempos de timeout, está definida en el archivo 'include/asm-generic/param.h'. Esta constante establece la frecuencia interna del timer del *kernel* y depende del sistema operativo.

Figura C.4: Fragmentos de código del *kernel* de Linux que muestran el momento exacto en el que se da un timeout de asociación tras un número de intentos fallidos y las constantes que definen cuál es el tiempo de timeout y el número de intentos. Ambos fragmentos se encuentran en el archivo '`net/mac80211/mlme.c`'

Anexo D

Drivers y firmware de las interfaces de red

A continuación se muestra la salida de los comandos `ethtool` y `lshw` que muestran la información más relevante en lo que a *drivers*, *firmware* y configuración de las interfaces de red instaladas en un sistema. Para la tarjeta Alfa2, se detallan también los pasos de actualización que fueron necesarios en el sistema para que fuera reconocida. Otro comando que muestra información muy detallada sobre el hardware es `lsusb`, pero su salida es muy extensa y se ha añadido como archivo de texto en el repositorio del proyecto[1].

D.1. Intel

```
~$ sudo ethtool -i wlp0s20f3
driver: iwlwifi
version: 6.8.0-47-generic
firmware-version: 86.fb5c9aeb.0 so-a0-hr-b0-86.uc
expansion-rom-version:
bus-info: 0000:00:14.3
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
~$
~$
~$ sudo lshw -C network
  *-network:0
       descripción: Interfaz inalámbrica
       producto: Alder Lake-P PCH CNVi WiFi
       fabricante: Intel Corporation
       id físico: 14.3
```

[1] https://github.com/gcr-UZ/tfm-mesh

```
       información del bus: pci@0000:00:14.3
       nombre lógico: wlp0s20f3
       versión: 01
       serie: 70:d8:23:d3:b4:bd
       anchura: 64 bits
       reloj: 33MHz
       capacidades: pm msi pciexpress msix bus_master cap_list ethernet
                    physical wireless
       configuración: broadcast=yes driver=iwlwifi
                      driverversion=6.8.0-47-generic
                      firmware=86.fb5c9aeb.0 so-a0-hr-b0-86.uc
                      ip=192.168.68.113 latency=0 link=yes multicast=yes
                      wireless=IEEE 802.11
       recursos: iomemory:600-5ff irq:16 memoria:601d1cc000-601d1cffff
```

D.2. Alfa1

```
~$ sudo ethtool -i wlx00c0cab2bc2c
driver: mt76x0u
version: 6.8.0-47-generic
firmware-version: 0.1.00-b7640
expansion-rom-version:
bus-info: 3-9:1.0
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
~$
~$
~$ sudo lshw -C network
  *-network
       descripción: Interfaz inalámbrica
       id físico: d
       información del bus: usb@3:9
       nombre lógico: wlx00c0cab2bc2c
       serie: 00:c0:ca:b2:bc:2c
       capacidades: ethernet physical wireless
       configuración: broadcast=yes driver=mt76x0u
                      driverversion=6.8.0-47-generic
                      firmware=0.1.00-b7640 ip=10.1.64.217
```

```
                link=yes multicast=yes wireless=IEEE 802.11
```

D.3. Alfa2

```
~$ sudo ethtool -i wlx00c0cab3c2de
driver: mt7921u
version: 6.8.0-47-generic
firmware-version: ____010000-20231109190959
expansion-rom-version:
bus-info: 3-9:1.3
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
~$
~$
~$ sudo lshw -C network
  *-network
       descripción: Interfaz inalámbrica
       id físico: d
       información del bus: usb@3:9
       nombre lógico: wlx00c0cab3c2de
       serie: 00:c0:ca:b3:c2:de
       capacidades: ethernet physical wireless
       configuración: broadcast=yes driver=mt7921u
                      driverversion=6.8.0-47-generic
                      firmware=____010000-20231109190959
                      ip=10.1.64.251 link=yes multicast=yes
                      wireless=IEEE 802.11
```

D.3.1. Actualización del sistema para reconocer la tarjeta Alfa2

El *driver* de esta tarjeta, `mt7921u`, requiere una versión 5.18 o superior del *kernel* de Linux[2], y el Lenovo con el que se ha realizado este trabajo contaba con el *kernel* 5.15.0 en Ubuntu 20.04. Por ello, se actualizó concretamente a la versión de *kernel* 5.19.0, pero surgieron

[2]https://wireless.docs.kernel.org/en/latest/en/users/drivers/mediatek.html (accedido: 28/11/2024)

diversos problemas de dependencias entre librerías del sistema. Buscando la raíz del problema se llegó a la conclusión de que la solución óptima era actualizar el sistema operativo, por lo que se actualizó a Ubuntu 22.04 con el *kernel* 6.8.0-49-generic, siendo esta la configuración con la que se ha desarrollado el proyecto.

No obstante, ya con esta versión de *kernel*, la tarjeta todavía no era reconocida. Se buscó por diversas fuentes la causa del problema y no fue hasta que se siguieron los pasos recogidos en la sección 4 del repositorio público de GitHub[3] indicado a pie de página cuando se consiguió hacer funcionar la tarjeta. Era necesario descargar manualmente el *firmware* específico de esta tarjeta y guardarlo en el directorio de Ubuntu "`/lib/firmware`", donde se almacenan los archivos de *firmware* necesarios para que el sistema operativo y sus dispositivos funcionen correctamente.

[3]https://github.com/morrownr/USB-WiFi/blob/main/home/How_to_Install_Firmware_for_Mediatek_based_USB_WiFi_adapters.md (accedido: 28/11/2024)

Anexo E

Resumen temporal del trabajo

En este anexo se recoge una visión de la evolución temporal del trabajo, desde su comienzo a mediados de mayo de 2024 hasta su finalización en noviembre de 2024. En la Figura E.1 se muestra el diagrama de Gantt que describe las distintas tareas realizadas en este intervalo de tiempo.

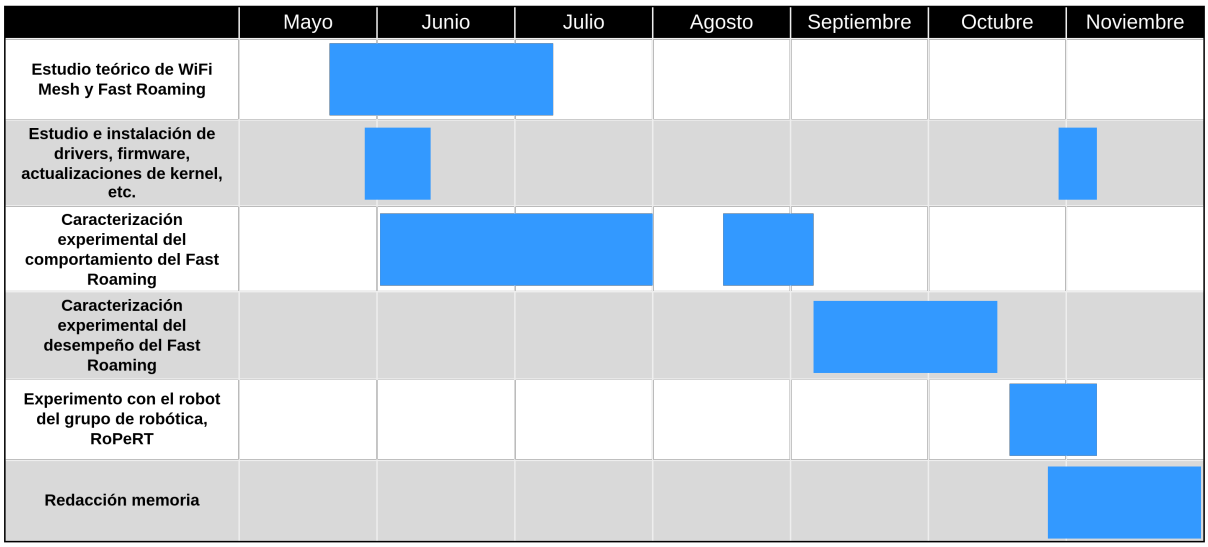


Figura E.1: Diagrama de Gantt de las tareas realizadas a lo largo del proyecto.

En el diagrama se distinguen las siguientes fases:

- **Estudio teórico de WiFi Mesh y *Fast Roaming*:** estudio y comprensión de los componentes que forman una red WiFi Mesh, así como su funcionamiento, ventajas y desventajas. Búsqueda de fuentes fiables que documenten el proceso de *Fast Roaming* y lectura de los estándares 802.11k y 802.11v del IEEE.
- **Estudio e instalación de *drivers*, *firmware*, actualizaciones de *kernel*, etc.:** solución de problemas relacionados con las tarjetas de red del proyecto, especialmente con la tarjeta Alfa2. Búsqueda y lectura de fuentes que aportasen información específica sobre los *drivers* y *firmware* necesarios para el funcionamiento de las tarjetas Alfa en Ubuntu. Al final del proyecto se volvió a hacer énfasis en este estudio para sacar adelante el experimento del robot.

- **Caracterización experimental del comportamiento del *Fast Roaming*:** caracterización experimental para comprender el funcionamiento del *Fast Roaming* en los dispositivos de este proyecto. Esta parte incluye la depuración en sistemas Linux para comprender y subsanar posibles puntos de fallo que impidiesen un *Fast Roaming* correcto.
- **Caracterización experimental del desempeño del *Fast Roaming*:** cálculo de métricas de rendimiento relevantes en cuanto a aplicaciones robóticas con restricciones de tiempo real para analizar la validez de la red WiFi Mesh y el *Fast Roaming* en una aplicación de teleoperación con transmisión de vídeo.
- **Experimento con el robot del grupo de robótica, RoPeRT:** pruebas con el experimento que surgió en la fase final del trabajo basado en el control del robot con ROS del grupo de robótica, RoPeRT.