Universidad Zaragoza
1542

Trabajo Fin de Grado

Análisis e implementación de nuevas funcionalidades en un entorno WLAN Enterprise basado en SDN

Analysis and development of new functionalities in a WLAN Enterprise environment based on SDN

Autor

Rubén Munilla Hernández

Director

José Mª Saldaña Medina

Ponente

Julián Fernández Navajas

ESCUELA DE INGENIERIA Y ARQUITECTURA

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Rubén Munilla Hernández,

con nº de DNI 16609036T en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Análisis e implementación de nuevas funcionalidades en un entorno WLAN Enterprise basado en SDN,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 24 de Noviembre de 2017

Fdo: Rubén Munilla Hernández

Agradecimientos

A José Mª, Pepe y Julián, por vuestra ayuda y orientación.

A mi familia, por apoyarme y permitirme llegar hasta aquí.

Resumen

Vivimos en un mundo conectado, en el que disfrutar de acceso a las redes de información se ha convertido en algo prioritario. Uno de los medios para conseguirlo es el uso de redes Wi-Fi, cuya implantación generalizada ha hecho surgir la necesidad de coordinar el despliegue para dar un mejor servicio, sobre todo en localizaciones de gran densidad de usuarios como centros comerciales, aeropuertos o centros de negocios.

En este contexto, el Proyecto H2020 Wi-5, haciendo uso de tecnologías SDN (Software Defined Networking, Redes Definidas por Software) y de equipamiento doméstico, pretende poner al alcance de todos unas funcionalidades avanzadas de optimización y coordinación entre Puntos de Acceso, que actualmente sólo están disponibles mediante un desembolso económico importante.

Enmarcado dentro del Proyecto Wi-5, el presente trabajo tiene como objetivo estudiar, optimizar y desarrollar las herramientas utilizadas para ese fin, obteniendo un sistema de código abierto comparable con las soluciones empresariales de las grandes marcas del sector.

El traspaso transparente de terminales entre puntos de acceso es una parte esencial de la solución, ya que permite la movilidad, así como el balanceo de carga, sin disminuir la calidad percibida por el usuario. En el presente trabajo se ha desplegado un escenario distribuido para analizar, mejorar y medir al detalle esa funcionalidad. Tomando como punto inicial un procedimiento reactivo, se mejora y se caracteriza su comportamiento, para posteriormente desarrollar una versión proactiva que corrija los puntos débiles de la solución original.

Asimismo, se ha integrado el trabajo de otros socios del proyecto en la infraestructura SDN, demostrando la flexibilidad del sistema y su capacidad para recopilar información y utilizarla para mejorar el servicio ofrecido. La integración requiere el uso de la información de monitorización obtenida de la red, la ejecución de algoritmos desarrollados en otros Paquetes de Trabajo, y la implementación de sus decisiones en la red inalámbrica.

En noviembre de 2017 se ha realizado una demostración de la solución en La Haya (Países Bajos) ante el *Operator Board*, un panel de operadores de red que participa de los resultados del proyecto, y cuyos consejos son esenciales para la mejora de las soluciones propuestas. Esta demonstración se mejorará para su presentación final ante la Comisión Europea, planificada para mediados de 2018.

Abstract

We live in a connected world, where enjoying access to information networks has become a priority. One of the means to achieve this is the use of Wi-Fi networks. Its widespread implementation has given rise to the need of a better coordination, thus providing a better service, especially in dense scenarios, such as shopping malls, airports or business centers.

In this context, the Wi-5 Project, making use of SDN (Software Defined Networking) technologies and domestic equipment, aims to make available to everybody functionalities that nowadays are exclusively targeted for enterprise customers.

Within the framework of the Wi-5 Project, the present work aims to study, optimize and develop the tools used for this purpose, obtaining an open source system comparable with the business solutions offered by the major brands in the sector.

The seamless handover of terminals between access points is an essential part of the solution, as it allows the mobility, as well as load balancing, without reducing the quality level perceived by the user. A distributed scenario has been deployed in order to examine, improve and analyze that functionality in detail. Taking a reactive procedure as a starting point, its behaviour is improved and characterized, and then a proactive version that corrects the weak points of the original solution is developed and tested.

Likewise, the work of other partners of the project has been integrated into the SDN infrastructure, demonstrating the flexibility of the system and its ability to collect information and use it to improve the service offered. The integration requires the use of monitoring tools, calling the algorithms developed within other Work Packages, and the implementation of their decisions in the wireless network.

In November 2017, a demonstration of the solution was carried out in The Hague (Netherlands) in the presence of the members of the Operator Board, a panel of network operators that participates in the results of the project, whose advice is essential for the improvement of the solutions. This demonstration will be improved for the final review of the project with the European Commission, planned for mid-2018

Índice

1. Introducción

1.1 Problemática

En los últimos años se ha experimentado un gran crecimiento en el uso de dispositivos móviles (teléfonos, tabletas, portátiles, etc.) y esto ha llevado a un aumento de la necesidad de permanecer conectado a Internet en todo momento y lugar (casa, trabajo, ocio, etc.). Aunque actualmente el despliegue de redes de datos 3G o 4G está generalizado en muchos países, las redes inalámbricas IEEE 802.11 [1], conocidas como redes Wi-Fi, son preferidas por muchos usuarios, debido a su compatibilidad con diversos dispositivos, su ancho de banda y, por supuesto, por su gratuidad en la mayor parte de los lugares públicos (bares, restaurantes, hoteles, aeropuertos, etc.).

Actualmente el número de redes Wi-Fi que podemos encontrar en una misma localización es muy alto, compartiendo todas ellas unos recursos espectrales limitados. En la Figura 1 podemos ver un ejemplo de esa situación, lo que se conoce como *Wi-Fi Jungle* [2], donde los puntos de acceso (*Access Point*, APs) comparten canales y la interferencia de otros terminales es elevada.

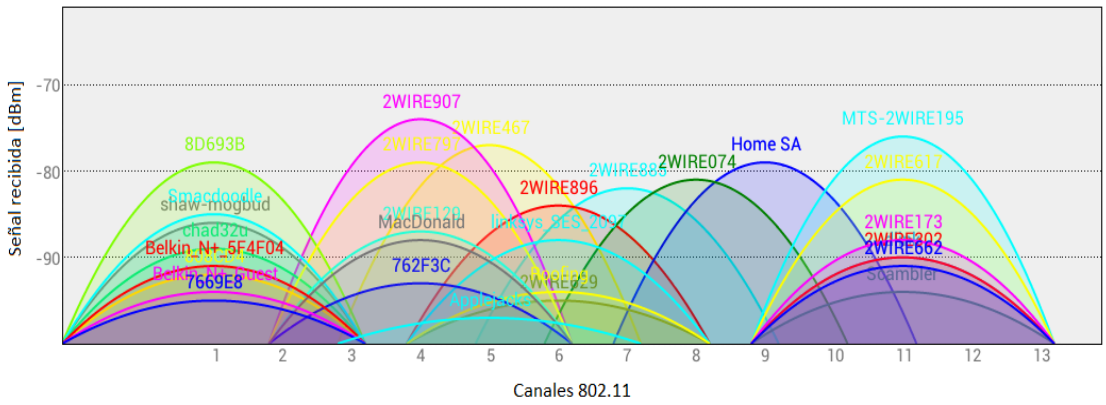


Figura 1. Ejemplo de saturación de redes inalámbricas 802.11 en una zona.

En lugares en los que es posible coordinar los puntos de acceso, como centros comerciales, empresas o edificios de servicios públicos, se despliegan soluciones denominadas *WLAN Enterprise*, redes inalámbricas de clase empresarial que mejoran en ciertos aspectos la versión para uso personal, que se suele denominar SOHO (*Small Office* / *Home Office*).

Entre los aspectos que diferencian ambas soluciones destacan varios que se detallan a continuación:

- Seguridad: La mayor parte de los puntos de acceso, incluso los productos SOHO, soportan diferentes tipos de opciones de cifrado y ofrecen incluso autenticación de usuario. Pero en un entorno empresarial podemos encontrar una gran variedad de terminales, por lo que los puntos de acceso deben ser capaces de operar simultáneamente con diferentes opciones de seguridad. Puede que con securizar la conexión no sea suficiente, por lo que la solución *Enterprise* puede añadir por ejemplo Detección de Intrusos, para evitar el acceso no autorizado o comprometer los datos.

- Acceso de invitados: Aunque no sea esencial, en empresas o edificios públicos, el ofrecer una red de cortesía para visitantes o invitados mejora notablemente la imagen. El separar dicha red de la utilizada por los trabajadores es una medida de seguridad básica para evitar filtraciones o usos no adecuados. Esta segmentación puede realizarse por medio de VLANs (*Virtual Local Area Networks*, Redes de Área Local Virtuales), pero en las redes *WLAN Enterprise* se da un paso más. Se utilizan opciones como portales cautivos, en los que el usuario tiene acceso limitado; usuarios temporales, con los que se controla el uso; e incluso se consigue impacto en redes sociales dando la opción al usuario de utilizar su cuenta para conectarse al sistema.

- Movilidad: Existen situaciones en las que los usuarios de la red no se encuentran constantemente en la misma ubicación y se desplazan por diferentes partes de un edificio. Durante ese desplazamiento se pierde la zona de cobertura del punto de acceso al que está asociado y accede a la de otro AP, apareciendo el problema denominado *"sticky client"*: el terminal permanece asociado al punto de acceso inicial hasta que la señal que recibe se encuentra por debajo de la sensibilidad del interfaz inalámbrico. En muchos casos es demasiado tarde y el empeoramiento del enlace ha producido perdidas y bajada de *throughput* (tasa de transferencia efectiva) [3]. Esto se podría evitar si el terminal se hubiera reasociado a otro AP más cercano. Para que el usuario no perciba el cambio y la conexión no caiga, algunas soluciones empresariales pueden realizar traspasos suaves *(seamless handover),* que gestionan la reasociación del terminal de una manera transparente y efectiva, evitando la degradación de la conexión y minimizando las pérdidas y retardos. Esta funcionalidad es muy importante en aplicaciones en tiempo real como VoIP (Voz sobre IP, *Voice over IP)* o *streaming*.

- Precio: Por último, pero no menos importante, la diferencia en coste de un punto de acceso *Enterprise* comparado con los modelos sencillos es muy a tener en cuenta. Mientras que los APs SOHO se encuentran en el orden de los 100€, los empresariales se posicionan en torno a los 400€ o 500€ para fabricantes de primer nivel.

1.2 Marco del trabajo

El desarrollo del TFG se enmarca dentro del proyecto europeo H2020 *What to do With the Wi-Fi Wild West* (Wi-5) [4], en el que se estudian e implementan soluciones SDN (*Software Defined Network*, Redes Definidas por Software) para gestionar y coordinar los distintos puntos de acceso de una red inalámbrica mediante software libre.

Debido a que las soluciones *WLAN Enterprise* actuales son tecnología propietaria y de alto coste, el proyecto Wi-5 pretende estudiar y desarrollar herramientas de código abierto que realicen ese servicio utilizando hardware de bajo coste, centrándose en la movilidad y el uso eficiente de los recursos inalámbricos.

El trabajo se centra en dos paquetes de trabajo (*Work Packages*, WP) del proyecto: el WP3 *(Smart AP Solutions),* liderado por la Universidad de Zaragoza, está dedicado a las funcionalidades que corren en los puntos de acceso, coordinadas por un controlador central. El WP5 *(Integration and Field Trials),* liderado por AirTies (una empresa de Turquía dedicada al desarrollo y fabricación de APs WiFi), se dedica a la

integración de todas las soluciones, así como a la realización de pruebas de rendimiento y validación de las soluciones propuestas.

Además, se da soporte al resto de miembros del proyecto Wi-5 para el despliegue y prueba de las soluciones, incluidos en el WP5, teniendo en cuenta sus necesidades y dificultades, para lograr una mejora constante de la implementación. Dicho soporte se realiza tanto por correo electrónico como por teleconferencia. También se ha realizado una reunión del proyecto en La Haya (Países Bajos) en noviembre de 2017, y una demostración al *Operator Board* (un panel de operadores de red que participa de los resultados del proyecto)[1]. Esta realimentación por parte de los socios (especialmente las empresas del consorcio) es de una gran importancia, para conseguir que las soluciones sean correctas, pero a la vez sean realizables en la práctica, evitando aproximaciones demasiado teóricas.

1.3 Objetivos

Como objetivos prioritarios se establecen los siguientes puntos:

- Mejora de la gestión de la movilidad: Partiendo de la aplicación **MobilityManager**[2], se debe estudiar el comportamiento de los traspasos (*handoff* o *handover*) en un escenario distribuido, medir la respuesta y prestaciones de la solución y mejorarlas. Todo ello mediante el despliegue del escenario en varios laboratorios de la EINA y la configuración de un sistema de captura sincronizado que refleje fielmente los resultados del sistema.

- Integración del algoritmo de asignación de canales: Desarrollo de una aplicación llamada **ChannelAssignment**[3] que integre los algoritmos de asignación de canales desarrollados por el equipo de la *Liverpool John Moores University* (LJMU) en el WP4, siendo los parámetros de entrada la información recogida por la funcionalidad de monitorización externa de la solución Wi-5. Con ello se pretende minimizar la interferencia introducida por los puntos de acceso vecinos. En este aspecto se cuenta con la asistencia del equipo de AirTies, que muestra la visión práctica y aconseja sobre el uso de los equipos en entornos reales.

- Desarrollo de la aplicación de balanceo de carga: Implementación de una nueva aplicación llamada **SmartApSelection**[4] que permita decisiones proactivas, analizando el estado de la red y sus elementos para seleccionar el punto de acceso idóneo para cada terminal, comparando los resultados con los obtenidos para la aplicación **MobilityManager**.

- Preparación de demostración de la plataforma: Se planeará una serie de pruebas para mostrar las diferentes funcionalidades de la solución, que servirán para la presentación al *Operator Board*, y para la revisión final del proyecto con la Comisión Europea en Bruselas.

En este documento se resumen las contribuciones realizadas al proyecto. Se debe estudiar la tecnología desarrollada previamente, analizar su rendimiento mejorando

[1] Wi-5 Operator Board members: http://www.wi5.eu/?page_id=193 Accedido noviembre de 2017.
[2] MobilityManager: https://github.com/Wi5/odin-wi5/wiki/Application-MobilityManager Acc. Nov. 2017.
[3] ChannelAssignment: https://github.com/Wi5/odin-wi5/wiki/Applications-for-Channel-Assignment Accedido noviembre de 2017.
[4] SmartAPSelection: https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection Acc. Nov. 2017.

partes del mismo y desarrollar nuevas funcionalidades. Todo el desarrollo y sus sucesivas actualizaciones se coordinan en un repositorio de *GitHub*[5], web que aloja el código y ofrece herramientas para trabajo en equipo. Otro cometido es crear la documentación necesaria para entender el funcionamiento y puesta en marcha del sistema[6].

1.4 Estructura del documento

La memoria se ha organizado con la siguiente disposición:

- Capítulo 1: se presenta la descripción del trabajo, la problemática del mismo y los objetivos a alcanzar.

- Capítulo 2: se repasa el estado del arte, las tecnologías utilizadas y la plataforma utilizada en Wi-5 en detalle.

- Capítulo 3: se trata de la movilidad en el sistema, presentando el problema, analizando la solución implementada previamente, así como las modificaciones y resultados obtenidos en el presente trabajo.

- Capítulo 4: estudiamos la asignación de canales, presentando el problema, explicando la integración realizada y los resultados obtenidos.

- Capítulo 5: se detallan los pasos para el desarrollo de la aplicación de balanceo de carga, presentando el problema, mostrando la implementación y los resultados obtenidos.

- Capítulo 6: se exponen las conclusiones finales del trabajo y posibles líneas futuras, así como la planificación del mismo.

Finalmente, se incluyen los siguientes anexos:

- Anexo A: Instalación ESXi en Intel Nuc, alojamiento de máquinas virtuales
- Anexo B: Configuración de red vSphere
- Anexo C: Configuraciones OpenWrt
- Anexo D: Configuración del Switch con Trunking
- Anexo E: Esquema Click Modular Router
- Anexo F: Archivo de configuración “poolfile”
- Anexo G: Códigos en GitHub
- Anexo H: Fotografías La Haya 2017

[5] Repositorio del proyecto Wi-5 en GitHub: https://github.com/Wi5 Accedido noviembre de 2017.
[6] Wiki del Proyecto Wi-5 en GitHub: https://github.com/Wi5/odin-wi5/wiki Accedido noviembre de 2017.

2. Estado del arte

2.1 Redes definidas por software

En los equipos de comunicaciones clásicos, el envío de paquetes de datos y las decisiones de encaminamiento se realizan en el mismo dispositivo. Esta situación conlleva una muy baja flexibilidad a la hora de realizar cambios en la topología global de la red, teniendo que depender de los distintos algoritmos de encaminamiento.

Por ello surgen las redes definidas por software (SDN), una abstracción del hardware donde se diferencia claramente el plano de datos, por el que circula la comunicación entre los terminales de la red, y el plano de control, utilizado por los dispositivos de comunicaciones para coordinar y ordenar modificaciones. Esta abstracción se basa en transformar los elementos de la red en dispositivos sencillos que se encargan de seguir las órdenes de un controlador central, superior jerárquicamente, que tiene toda la información del despliegue y decide cómo se trata cada conexión.

La arquitectura SDN no dispone de un estándar, pero dentro del IRTF *(Internet Research Task Force)*[7], el grupo *Software-Defined Networking Research Group* (SDNRG[8], Investigación en Redes Definidas por Software) la definió desde un punto de vista funcional. Su propuesta [5] se puede observar en la Figura 2.

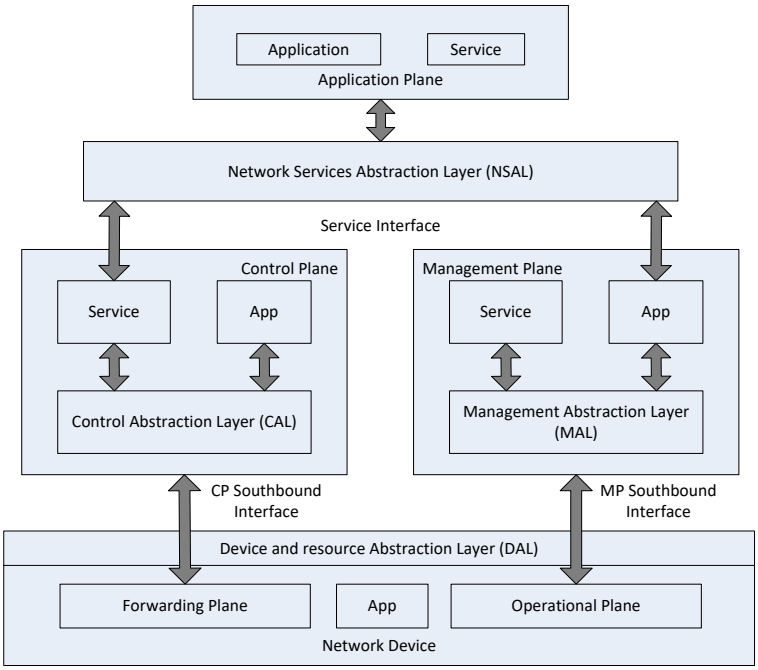


Figura 2. Arquitectura SDN según RFC 7426.

La comunicación entre el controlador y los elementos de la red necesita de un protocolo, siendo OpenFlow [6] el más utilizado. Inicialmente se desarrolló para entornos académicos, pero en la actualidad lo implementan aparatos de marcas de primera línea como Cisco o Juniper.

[7] The Internet Research Task Force: https://www.irtf.org/ Accedido noviembre de 2017.
[8] Software Defined Networking: https://datatracker.ietf.org/rg/sdnrg/about/ Accedido noviembre de 2017.

2.2 Virtualización

Una de las tendencias con más fuerza en la actualidad es la virtualización, que crea a través de software una versión virtual de un recurso tecnológico. La posibilidad de levantar servicios mediante un solo "*click*", realizar copias de seguridad o crear nuevas estaciones de trabajo en segundos han revolucionado el mercado.

Con una inversión inicial ligeramente mayor a la desembolsada para un equipo tradicional, la virtualización permite disponer de nuevas posibilidades que nos ofrecen una versatilidad, flexibilidad y rapidez de acción que con los equipos físicos es imposible lograr. Otros beneficios de las soluciones virtualizadas son la redundancia y la ágil recuperación ante fallos, teniendo en cuenta que el coste medio por minuto de un centro de datos sin servicio alcanza la cifra de 9000$ [7].

Como se observa en la Figura 3, hoy en día podemos encontrar en la misma máquina física todo tipo de elementos:

- Terminales de usuario: se acceden en remoto mediante clientes ligeros, "*thin clients*", que son equipos sencillos cuyo coste es mucho menor que una estación de trabajo tradicional, reduciendo sensiblemente el presupuesto.

- Almacenamiento: abstracción de los discos duros hardware, que permite copias de seguridad instantáneas, redundancia para evitar pérdidas de datos y alta disponibilidad.

- Elementos de red: es posible crear dispositivos de red virtuales que se utilizarán para la conexión entre los equipos reales y virtuales. Entre ellos destacan el conmutador virtual (vSwitch), la red de superposición (*Overlay Network*), el hipervisor de red y las redes virtuales extendidas (*Virtual extensible local area networks*, VXLANs) [8].

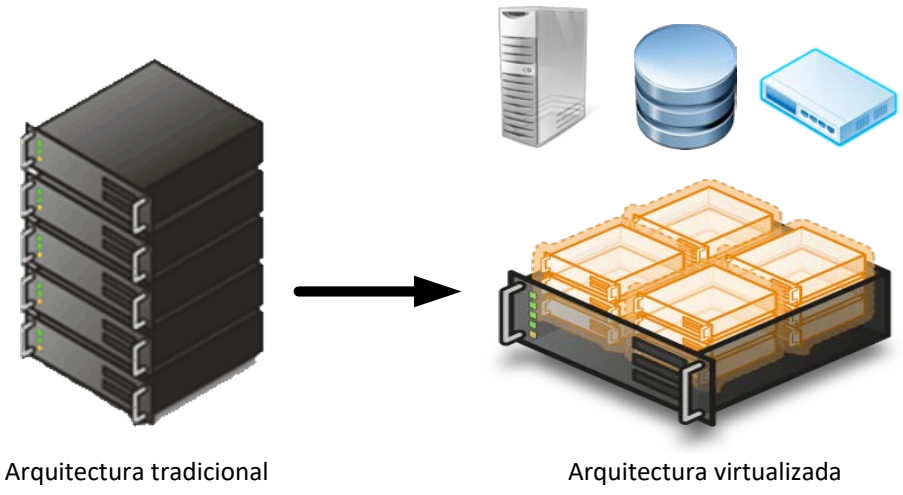


Figura 3. Transformación hacia elementos virtualizados.

En nuestro caso utilizaremos varios elementos virtualizados, destacando el conmutador virtual. Al permitir su configuración mediante reglas OpenFlow, nos dará la posibilidad de dirigir los flujos de datos a discreción del controlador. Asimismo, se virtualizarán varias piezas clave del sistema que detallaremos en apartados posteriores.

2.3 Redes inalámbricas definidas por software

La adaptación de las ideas de SDN a entornos inalámbricos se suele denominar SDWN (*Software Defined Wireless Network*). Esta es la aproximación que se está siguiendo en el proyecto Wi-5. Simultáneamente, se introduce el término "punto de acceso virtual ligero" (*Light Virtual Access Point*, LVAP) [9]: se trata de una tupla creada para cada terminal en el momento que accede a la red.

Gracias a esta abstracción, un mismo punto de acceso físico puede albergar varios LVAP, dando acceso a cada terminal mediante diferente BSSID (*Basic Service Set Identifier*). Todo ello facilita la gestión del controlador a la hora de decidir qué AP da servicio a una determinada estación (STA), siendo capaz de "trasladar" el LVAP al punto de acceso correspondiente de una manera transparente para el terminal asociado [10].

La Figura 4 ilustra el funcionamiento del LVAP: el controlador puede moverlo entre diferentes AP físicos, de forma que la estación siempre recibe las tramas con la misma dirección MAC origen, y "piensa" que siempre está conectada al mismo AP. Esto requiere un "parche" en el *driver* de la tarjeta inalámbrica del AP, para que se modifique la MAC en función de la STA destino. Es decir, un mismo AP envía tramas con distinto BSSID en función de la STA destino.

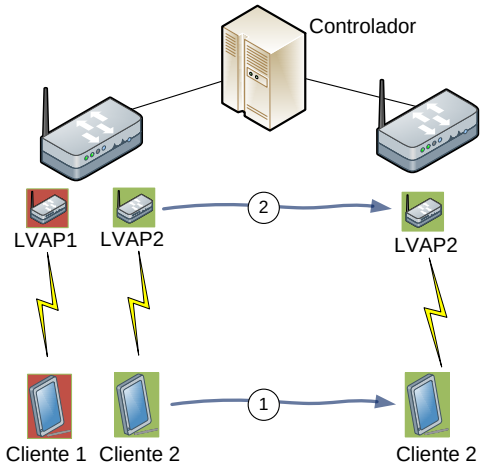


Figura 4. Traslado de LVAP por orden del controlador.

El LVAP consiste en una tupla con el conjunto de parámetros asociados a una estación:

- Dirección de control de acceso al medio (*Media Access Control*, MAC): identificador de 48 bits asociado al interfaz de red de un dispositivo. Es único, permitiendo asociar inequívocamente el origen o destino en la capa 2 del modelo OSI.
- Dirección IP (*Internet Protocol*): identificador lógico y jerárquico de 32 bits asociado al interfaz de red de un terminal. En este caso corresponde a la dirección de capa 3 del modelo OSI.
- Nombre visible de la red (*Service Set Identifier*, SSID): secuencia de un máximo de 32 bits presente en todas las tramas inalámbricas. Asocia las tramas a una red concreta, siendo necesario el mismo SSID para lograr la comunicación entre dos dispositivos.

- Dirección MAC del LVAP: identificador creado a partir de la dirección MAC del terminal al que se da servicio. Es la dirección de capa 2 que utilizará el punto de acceso para comunicarse con la estación (BSSID).

Para implementar correctamente esta arquitectura son necesarios dos protocolos que dirijan los puntos de acceso.

Por una parte, OpenFlow se encargará de que los flujos de datos sigan el camino correcto, y por la otra un protocolo (Odin) [10] que se dedicará a gestionar el plano inalámbrico y sus funcionalidades. Dicho protocolo y la arquitectura utilizada se estudia en detalle en el siguiente apartado.

2.4 Solución Odin-Wi-5

Existen diferentes modelos de arquitectura que consideran el uso de SDWN y LVAPs. En el proyecto Wi-5 se ha decidido utilizar como base del despliegue el *framework* Odin [10]. Se trata de un entorno SDN en el que el controlador utiliza OpenFlow para gestionar los elementos de red, en este caso conmutadores virtuales (vSwitch).

2.4.1 Controlador

El controlador elegido para esta tarea es Floodlight[9], un controlador SDN de clase *Enterprise* de código abierto. Está desarrollado en Java, lo que nos da la flexibilidad de poder ejecutarlo en el amplio número de sistemas que lo soportan, y es compatible con OpenFlow. La característica que lo hace idóneo para la solución es la posibilidad de añadir Odin como un módulo del mismo y lanzar aplicaciones que se ejecutan en un plano superior. Se basa en un modelo de tres capas, Aplicación-Control-Datos, pudiendo ver en detalle las interacciones entre los diferentes actores del proceso en la Figura 5 [11].

Las aplicaciones se basan la utilización de las funcionalidades inteligentes (*smart functionalities*) que ofrece el controlador a través de un interfaz de acceso a capas superiores (*Northbound*). Este gestiona dichas opciones en el plano correspondiente, aislando y coordinando las diferentes peticiones a los elementos de la red (*Southbound*).

Estas órdenes que lanza el controlador deben ser atendidas por los elementos de menor jerarquía en la red, denominados *agentes*, que incluyen módulos capaces de dar respuesta al controlador (*Handlers*).

[9] Proyecto Floodlight: http://www.projectfloodlight.org/floodlight/ Accedido noviembre 2017

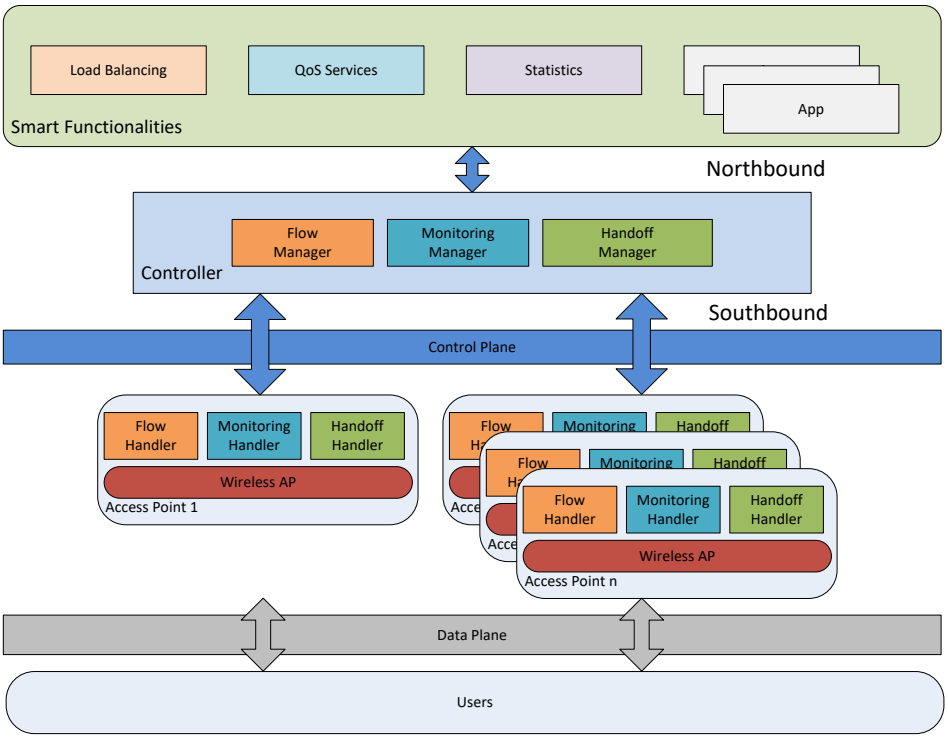


Figura 5. Arquitectura Odin-Wi5.

Entre las funcionalidades que se encuentran operativas actualmente podemos destacar las siguientes:

- Monitorización interna y externa: Gracias a la adquisición de estadísticas de todos los paquetes que gestiona o escucha el agente, se puede obtener información actualizada del estado de la red inalámbrica. Se denomina *interna* cuando es realizada por el interfaz principal del agente, y *externa* si la obtiene el interfaz auxiliar.
- Detección de flujos: Un agente especifico[10] se encarga de detectar los tipos de tráfico que circulan por la red de datos y notifica al controlador. Esta información se puede usar para mejorar la gestión de Calidad de Servicio (*Quality of Service*, QoS) de la red.
- Movilidad: Gestión del agente que da servicio a un determinado usuario, siendo posible el traspaso de LVAP a otros APs del sistema.

Haciendo uso de una o varias de estas funcionalidades, se pueden desarrollar aplicaciones[11] que realizan una tarea o un servicio.

2.4.2 Puntos de acceso

Respecto a la implementación en el punto de acceso, siguiendo las directrices expuestas anteriormente sobre el coste de los dispositivos, se elige un modelo de gama baja cuyo *chipset* admite la modificación necesaria en el driver para conseguir el uso de LVAPs. En nuestro caso se ha optado por modelos con el driver Atheros Ath9k[12], en los que el primer paso es modificar la conmutación de su *switch* interno, de forma que dependa de las órdenes de controlador.

[10] Agente detector de flujos: https://github.com/Wi5/odin-wi5-flow-detection Accedido Nov. 2017
[11] Aplicaciones Odin-Wi5.: https://github.com/Wi5/odin-wi5/wiki/Applications Acc. Nov. 2017
[12] Modelos compatibles: https://github.com/Wi5/odin-wi5/wiki/Supported-hardware Acc. Nov. 2017

Para lograrlo (Figura 6), se reemplaza el sistema operativo de los puntos de acceso con OpenWrt[13], versión de Linux ligero para sistemas embebidos, especialmente diseñado para correr en APs de entornos SOHO. Se instala la aplicación Open vSwitch[14], que implementa un conmutador virtual que interpretará las ordenes OpenFlow y redireccionará flujos de datos en función de reglas establecidas. Asimismo, los interfaces inalámbricos físicos del punto de acceso se configuran en modo *monitor* para su control mediante software.

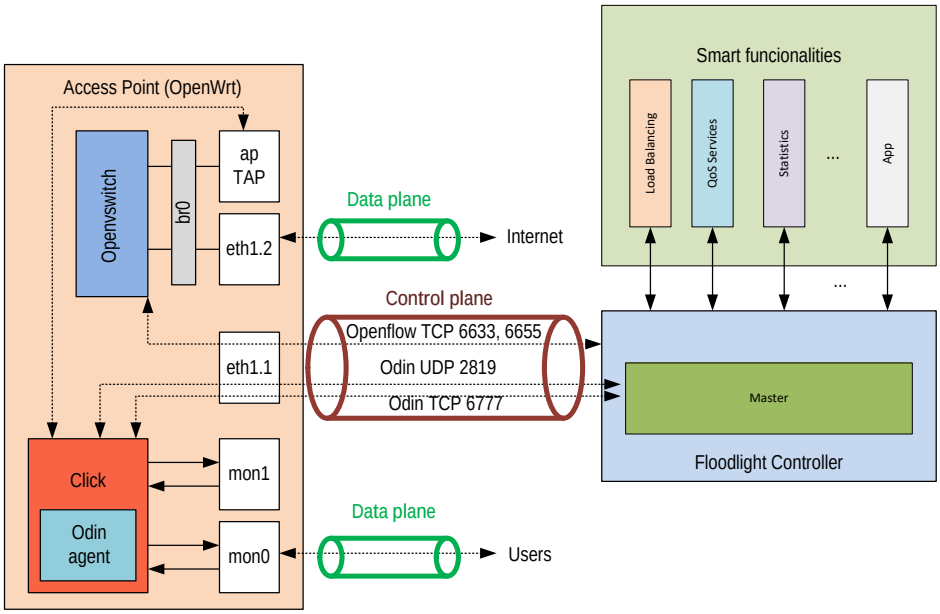


Figura 6. Esquema conexiones Odin-Wi5

La tarea de albergar el agente Odin recae en la aplicación *Click Modular Router* [12]. Se trata de un desarrollo liderado por el Instituto Tecnológico de Massachusetts (MIT) que permite realizar tareas de encaminamiento y procesado de paquetes. Se basa en la concatenación de módulos con tareas específicas. En nuestro caso, se añade el módulo *Odin agent*, que será el encargado de gestionar las órdenes del controlador y ejecutar las funcionalidades correspondientes. El esquema completo de la implementación se detalla en el Anexo E.

2.4.3 Protocolo Odin

El controlador y el agente se comunicarán mediante una conexión TCP en el puerto 6777 y UDP en el puerto 2819. Los datos viajan en texto en claro y son "parseados" en destino, utilizando la conexión TCP para las tareas de gestión y la de UDP para señalización.

Estas tareas de gestión se dividen en:

- Recopilación de información: Precedidas del preámbulo *"READ"*, ordenan al agente la comunicación de un parámetro concreto en su configuración o un conjunto de datos. Se encargan de ello los módulos incluidos en el agente llamados "*read_handler*". Algunos ejemplos son la recepción de las estadísticas internas o del canal en uso.

[13] OpenWrt, Wireless Freedom: https://openwrt.org/ Accedido noviembre 2017
[14] Open vSwitch: http://openvswitch.org/ Accedido noviembre 2017

- Modificación de parámetros y órdenes activas: En este caso comienzan con la orden *"WRITE"*, por lo que serán los *"write_handler"* quienes les den servicio. Son los encargados de realizar cambios en el comportamiento del agente, por ejemplo, añadir o eliminar un LVAP o cambiar el canal en uso.

2.5 Herramientas utilizadas

A continuación, se enumeran las herramientas, lenguajes de programación y plataformas utilizadas para el desarrollo de este trabajo:

- VMware ESXi: Sistema operativo de la compañía VMware[15] que nos permite utilizar un equipo físico como un servidor de virtualización. En él es posible levantar tanto equipos como redes virtuales de manera segura, aprovechando toda la potencia del equipo físico. Se ha utilizado en un dispositivo Intel NUC[16] 5i3RYH, que cuenta con un procesador i3-5010U, 16 GB de memoria y disco duro SSD. Las instrucciones para su instalación y configuración se encuentran en el Anexo A.

- VMware vSphere: Hipervisor de la empresa VMware. Con él somos capaces de gestionar el repositorio de elementos virtuales contenido en el servidor ESXI. Pone a nuestra disposición todas acciones necesarias para controlar los equipos, acceder a ellos mediante consola y configurar los elementos de red virtualizados. El Anexo B detalla los pasos necesarios para desplegar el montaje de red interno.

- Debian[17]: Sistema operativo GNU basado en Linux utilizado en las maquinas virtualizadas (controlador y router DHCP, *Dynamic Host Configuration Protocol*, protocolo de configuración dinámica de host) y en los equipos físicos de generación y captura de datos. Nos permite crear un entorno estable y compatible con el resto de herramientas utilizadas.

- OpenWrt: Sistema operativo ligero de código abierto, basado en Linux y especialmente diseñado para utilizarlo en puntos de acceso. Tiene una comunidad muy activa en la que podemos encontrar versiones del *firmware* de un gran número de dispositivos. Es la base utilizada en los puntos de acceso de la solución. En el Anexo C se detallan los pasos necesarios para actualizar los dispositivos y configurarlos.

- tcpdump[18]: Aplicación de captura de trafico de red por línea de comandos. Nos permite capturar y analizar las comunicaciones que suceden tanto en interfaces físicos como inalámbricos.

- Wireshark[19]: Software de análisis de capturas de red. En este caso se trata de una aplicación con interfaz gráfico de usuario con la que visualizar los datos capturados e interpretarlos de una manera más amigable. Es el complemento perfecto para tcpdump.

[15] VMware: https://www.vmware.com/es.html Accedida noviembre 2017
[16] Intel Nuc: https://www.intel.es/content/www/es/es/products/boards-kits/nuc.html Acc. Nov. 2017
[17] Debian: https://www.debian.org/ Accedida noviembre 2017
[18] Tcpdump: http://www.tcpdump.org/ Accedido noviembre 2017
[19] Wireshark: https://www.wireshark.org/ Accedido noviembre 2017

- Distributed Internet Traffic Generator (D-ITG)[20]: Plataforma capaz de generar diversos patrones de tráfico a nivel de capa de red [13], dándonos la posibilidad de observar el comportamiento de la solución mediante las medidas de pérdidas y retardos introducidos. En nuestro caso generaremos trafico UDP de tamaño de segmento fijo, reflejando en el *throughput* recibido las prestaciones del sistema.

- GNU Bourne-Again Shell (BASH)[21]: Aplicación Unix de código abierto con funciones de intérprete de comandos (*Command Line Interface*, CLI). La utilidad que nos ofrece es la creación de *script* para la ejecución secuencial de órdenes, facilitando la configuración o lanzamiento de aplicaciones.

- Python[22]: Lenguaje de programación interpretado que facilita el desarrollo de acciones complejas con un código sencillo y legible. En este trabajo se utiliza para crear los archivos de configuración de *Click Modular Router*.

- Perl[23]: Lenguaje de programación originalmente concebido para el tratamiento eficaz y ágil de textos. Es en ese ámbito en el que le daremos uso, para la creación de un *script* de procesado.

- Java: Lenguaje de programación orientada a objetos de propósito general, muy extendido y con escasas dependencias de implementación. Hace uso de una máquina virtual Java (JVM) para ejecutarse, puede ejecutarse en cualquier arquitectura. Se utiliza en la implementación del controlador Odin y en las aplicaciones que corren sobre él.

- C++: Lenguaje de programación orientada a objetos, extensión de C, que da la posibilidad de trabajar a bajo nivel y de manera eficiente. Se utiliza para la implementación del módulo correspondiente al agente dentro de *Click Modular Router*.

- GitHub[24]: plataforma de trabajo en equipo que permite la gestión de proyectos, alojando las diferentes versiones e identificando a los autores de cada una de ellas. Pone a nuestra disposición la creación de páginas de documentación ("Wikis"), resolución de errores ("Issues") y lanzamiento de versiones ("Releases").

[20] D-ITG: http://traffic.comics.unina.it/software/ITG/ Accedido noviembre 2017
[21] GNU Bourne-Again SHell: https://tiswww.case.edu/php/chet/bash/bashtop.html Acc. Nov. 2017
[22] Python: https://www.python.org/ Accedido noviembre 2017
[23] Perl: https://www.perl.org/ Accedido noviembre 2017
[24] GitHub: https://github.com/ Accedido noviembre 2017

3. Gestión de la movilidad

3.1 Problema

Como ya hemos comentado en los capítulos anteriores, la necesidad de movilidad con un impacto transparente para el usuario es una de las prioridades de este proyecto.

La solución Wi-Fi nativa incluye el concepto de Conjunto de Servicios Extendidos (*Extended Service Set*, ESS), una red inalámbrica con puntos de acceso configurados como *Routed APs.* Este conjunto de puntos de acceso comparte un mismo SSID (*Service Set Identifier*, nombre visible de la red), pero cada uno tiene un BSSID diferente. Una buena práctica es elegir canales diferentes en cada AP, evitando así el solapamiento espectral que empeora la calidad del enlace.

En la Figura 7 se muestra la situación comentada con anterioridad, donde ocurre con frecuencia el fenómeno *"sticky client"*: la estación se asocia inicialmente a un punto de acceso (AP15 en el ejemplo) y, hasta que la conexión comienza a fallar al recibir menor potencia que la sensibilidad de la tarjeta inalámbrica, no se plantea el cambio de AP (se asocia al AP13), por lo que se generan pérdidas y retransmisiones evitables. Según sean las distancias entre los APs, puede ocurrir que la STA nunca se conecte al AP14, sino que pase directamente del AP15 al AP13. Asimismo, el tiempo de desconexión en ese *handover* es elevado ya que el terminal debe hacer una reasociación al AP destino.

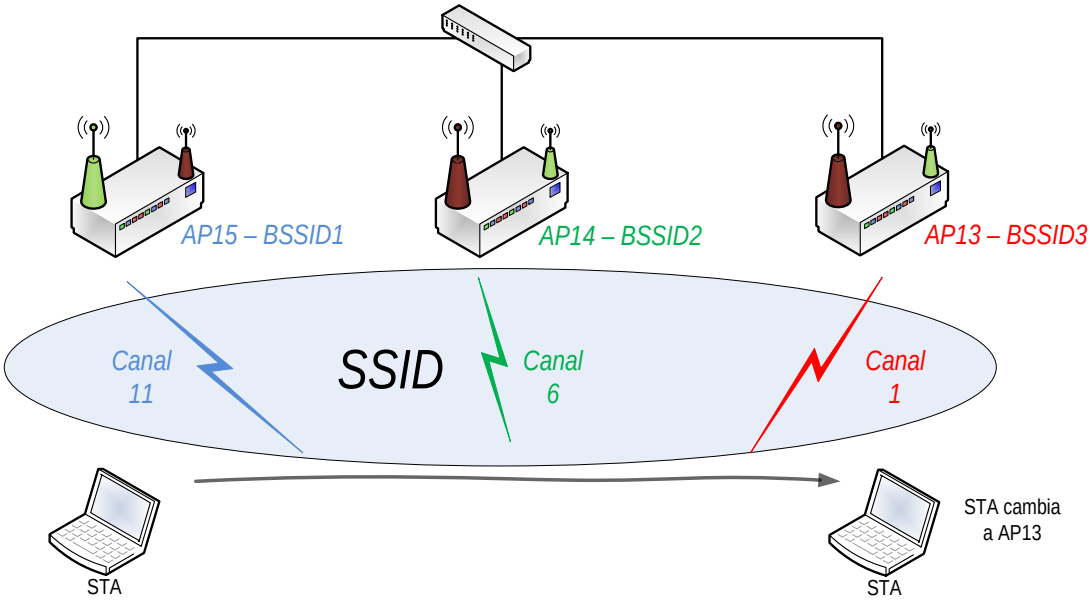


Figura 7. Representación del fenómeno "*sticky client*".

Este comportamiento hace pasar por alto el servicio que está ofreciendo el punto de acceso intermedio, perdiendo la posibilidad de una conexión estable durante todo el desplazamiento. Es especialmente dañino si el usuario se encuentra realizando comunicaciones en tiempo real, descargando un fichero o reproduciendo contenido en *streaming*.

3.2 Solución inicial

La solución Wi5 utiliza la aplicación MobilityManager para solucionar este problema**.** Esta aplicación permite que el controlador gestione y ordene un *handoff* reactivo, es decir, el AP avisa al controlador, mediante una suscripción, que puede ser necesario cambiar de punto de acceso a una estación en concreto.

En Wi-5, es posible que el controlador establezca una serie de suscripciones en el AP. Se trata de acciones que se ejecutan si se cumplen unas condiciones. En este caso, el AP envía un mensaje (*publish*) si la potencia de la STA se encuentra por debajo de un umbral, notificando al controlador dicha situación. Podemos ver el detalle del proceso en la Figura 8: la STA está asociada al AP15, pero se aleja de él (1). El AP15 envía entonces un *publish* al controlador (2).

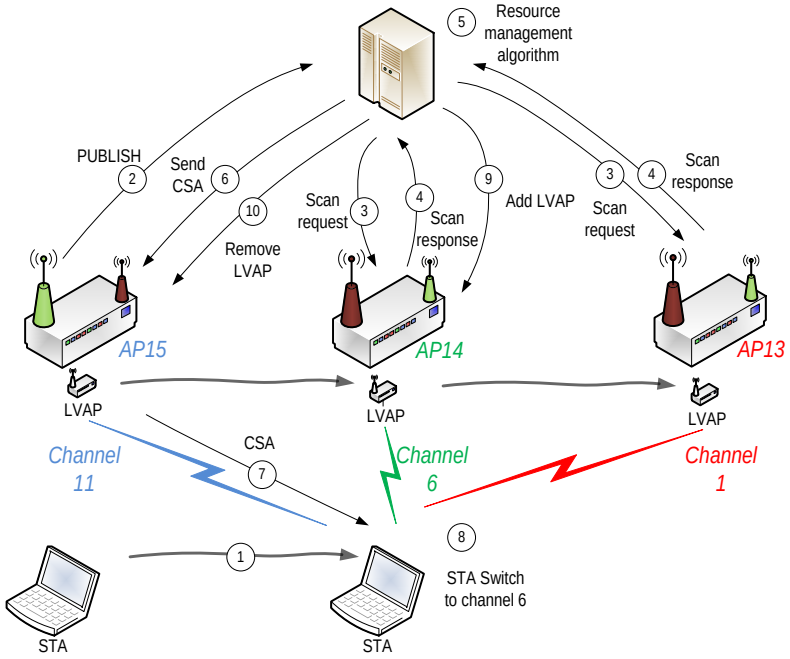


Figura 8. Esquema de un handover reactivo mediante la aplicación MobilityManager.

El controlador, al recibir el aviso, manda al resto de APs escanear (3) en el canal donde se encuentra la STA para comparar las potencias que escuchan. Una vez recibida esa información (4), ejecuta un algoritmo para decidir qué AP es el idóneo en base a la *RSSI* recibida por cada uno de los candidatos (5).

El cambio de AP se consigue mediante una estrategia muy hábil: se ordena al punto de acceso *origen* (6) que informe al terminal del cambio de canal mediante CSA (*Channel Switch Announcement*, anuncio de cambio de canal). A partir de ese momento, el AP envía una serie de CSAs (7) a la STA. Estos CSA se pueden ver como una cuenta atrás. Su significado es: *"dentro de 10 beacon, cambia al nuevo canal"*. Cuando el número llega a 0, la STA debe cambiarse al nuevo canal (8).

Para que el cambio sea rápido, el controlador hace coincidir el envío del último CSA (8) con el cambio del LVAP del AP anterior al nuevo (9 y 10). De esta forma, se consigue un *handover* muy rápido (entre 30 y 100 ms en función del hardware utilizado) en el que las pérdidas son muy bajas, resultando prácticamente imperceptible para el usuario. De cara a la STA, el cambio ha sido simplemente un cambio de canal, dado que siempre ha recibido del AP las tramas con la misma MAC.

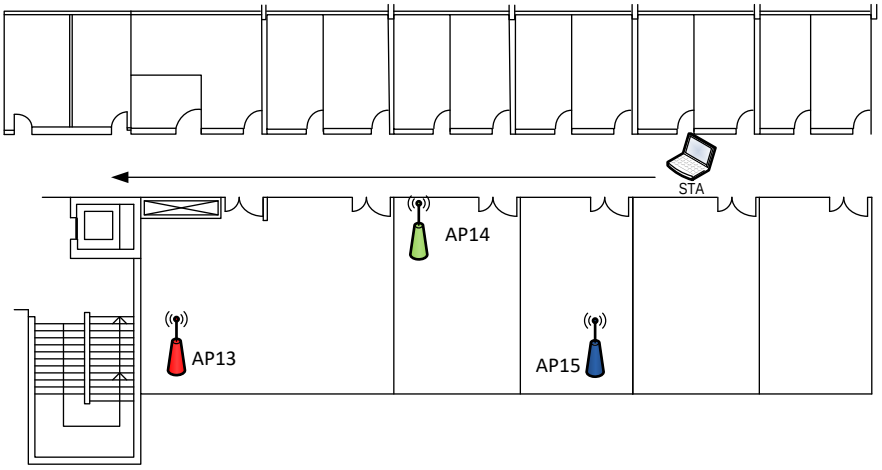
Se parte de esta solución, en un escenario con dos puntos de acceso en la misma estancia y la aplicación funcionando con parámetros locales, realizando un “ping-pong” entre ambos AP. Esta solución fue probada y medida inicialmente en [14], con dos APs y una STA estática situada entre ellos.

3.3 Mejoras introducidas y comparativa de resultados

Los objetivos son: *a)* probar la aplicación con tres APs y una STA en movimiento; *b)* mejorarla y ajustar los parámetros (umbrales) para que tenga un comportamiento correcto; *c)* medirla en detalle.

3.3.1 Escenario

Para comprobar el funcionamiento de la solución Wi-5, pasamos a un escenario distribuido de tres puntos de acceso con STA en movimiento, ilustrado en la Figura 9.



(a)

(b)

Figura 9. a) Escenario 3APs distribuidos en una planta del edificio Ada Byron; b) fotografía del pasillo.

El esquema de la red utilizada en las pruebas se ilustra en la Figura 10. Tanto el controlador como el servidor DHCP, que también realiza la función de *Router* del escenario, se encuentran virtualizados en un dispositivo Intel Nuc (Figura 11).

Se desea obtener trazas del tráfico enviado y recibido. Es interesante que se hagan en la misma máquina, para que ambas trazas tengan la misma base de tiempos. Para ello, utilizaremos un equipo extra como generador *(traffic generator)*, en una subred que comparte con el equipo de captura *(sniffer)* y la estación asociada a nuestra red a estudio (STA). De esta manera, el tráfico no se genera en la STA, sino que ésta tiene el rol de *Router* para su subred. Por tanto, el tráfico viaja desde el generador hasta el servidor *(server)* a través de la STA, que lo recibe por la tarjeta Ethernet y lo reenvía por la tarjeta Wi-Fi. Se generan paquetes UDP de 80 bytes con tasa 100 pps.

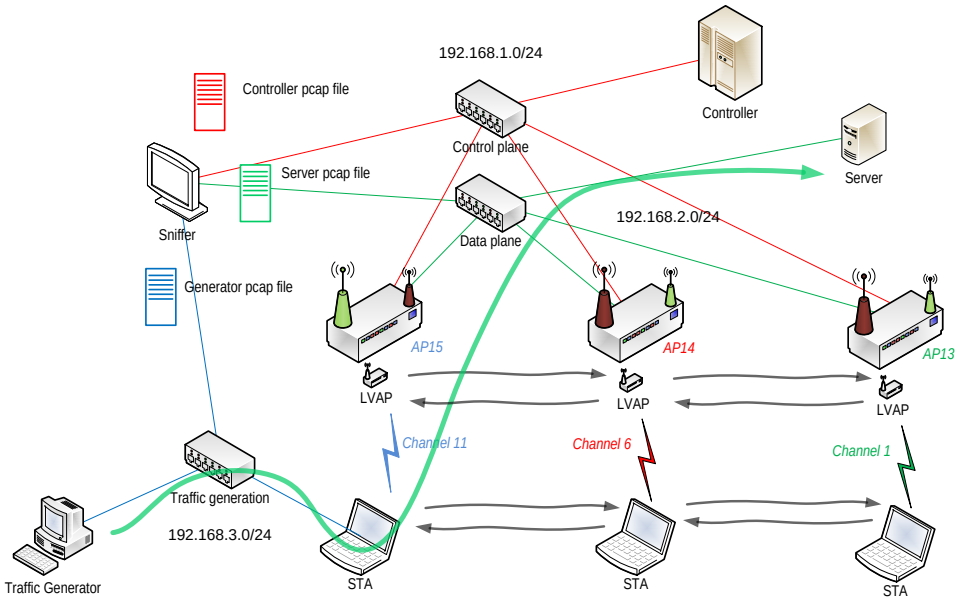


Figura 10. Despliegue de escenario con plano de captura.

Por tanto, el equipo de captura (*Sniffer*), tiene acceso al tráfico enviado, al plano de datos, y al plano de control (tráfico entre el controlador y el AP). De esa manera obtenemos tres capturas sincronizadas con las que somos capaces de estudiar al detalle el comportamiento del sistema.

Figura 11. Intel Nuc sobre el concentrador segmentado utilizado para capturar en los diversos planos.

3.3.2 Modificaciones y mejoras

a) Mejoras en el fichero de configuración. El controlador necesita una serie de parámetros para inicializarse y ejecutar las aplicaciones correspondientes. Estos datos se encuentran en un archivo llamado "*poolfile*"[25]. Se ha mejorado el proceso de obtener dichos parámetros para evitar errores y optimizar la agregación de nuevas aplicaciones. Asimismo, se han implementado funciones específicas para asignar los parámetros a cada aplicación, creando un sistema flexible y permitiendo realizar baterías de pruebas sin necesidad de compilar de nuevo la solución.

b) Cálculo de la media para lanzar el *publish* y para ejecutar el *handover*. En el desarrollo inicial solamente se tenían en cuenta valores instantáneos, tanto para la potencia que iniciaba el *publish,* como para las potencias recibidas por el controlador de los diferentes APs. Un paso esencial para obtener un comportamiento estable es obtener un valor correlado que nos permita lanzar el cambio de punto de acceso únicamente cuando la situación se mantenga, evitando valores esporádicos que falseen el estado de la estación. Por lo tanto, se ha introducido un cálculo en esas adquisiciones que realiza una media acumulada móvil (*Cumulative moving average*). Esta modificación se ha realizado tanto en la aplicación que corre en el controlador como en el agente que se encuentra en el punto de acceso. De esta manera, uno de los parámetros nos permite fijar el número de muestras con las que se calculará la media que debe estar por debajo de un umbral, antes de notificar la situación al controlador.

c) Archivo de todos los paquetes en un fichero. Como hemos comentado en capítulos anteriores, el agente guarda una estadística de todos los paquetes que gestiona, y esta información es accesible mediante las *Smart functionalities* de monitorización interna y externa. En nuestro caso, para realizar un análisis exhaustivo era necesario conocer los datos concretos de cada paquete, por lo que se ha desarrollado la capacidad de guardar en archivo un registro de todos ellos, adquiriendo directamente la información desde la cabecera inalámbrica Radiotap[26].

3.3.3 Comparativa y resultados

Para comprobar la mejora de utilizar la solución respecto a la configuración nativa de los APs, se utilizarán dos configuraciones: en primer lugar, una configuración habitual (nativa), donde los puntos de acceso se configuran como *Routed APs*, ofreciendo una red wifi de la que serán puerta de enlace. Las STA generarán tráfico y obtendremos resultados cualitativos con los que comparar la solución. En segundo lugar, utilizaremos la solución Wi-5.

Ambos escenarios se prueban utilizando el mismo hardware, tanto la tarjeta inalámbrica como APs, usando en este caso una potencia de transmisión de 10dBm para limitar el área de cobertura en las pruebas, y limitar así la distancia a recorrer.

En el Anexo C se pueden ver los pasos necesarios para llegar a una conectividad nativa en los APs. Ejecutamos D-ITG desde la STA para que el flujo de datos atraviese esa red y podamos medir su comportamiento.

El comportamiento de la solución Wi-Fi nativa se muestra en la Figura 12: el trafico recibido (línea negra) se mantiene estable aunque nos alejemos del AP 15 (potencia azul). Pero llega un momento (t=25 s) en el que la potencia recibida por el AP 14 (verde),

[25] Archivo poolfile: https://github.com/Wi5/odin-wi5-controller/blob/master/poolfile Acc. Nov. 2017
[26] Radiotap: http://www.radiotap.org/ Accedido noviembre 2017

y más tarde por el AP 13 (rojo, t=40s), nos ofrecerían un servicio más estable. Sin embargo, la estación se mantiene asociada al AP 15. Finalmente, en t=45 s, la STA recibe una potencia de -70 dBm del AP15, por lo que comenzamos a ver grandes variaciones en el *throughput*, introducidas por las pérdidas y los intentos de retransmisión. Esta situación se mantiene hasta que la conexión se pierde y la STA se reasocia al AP más cercano.

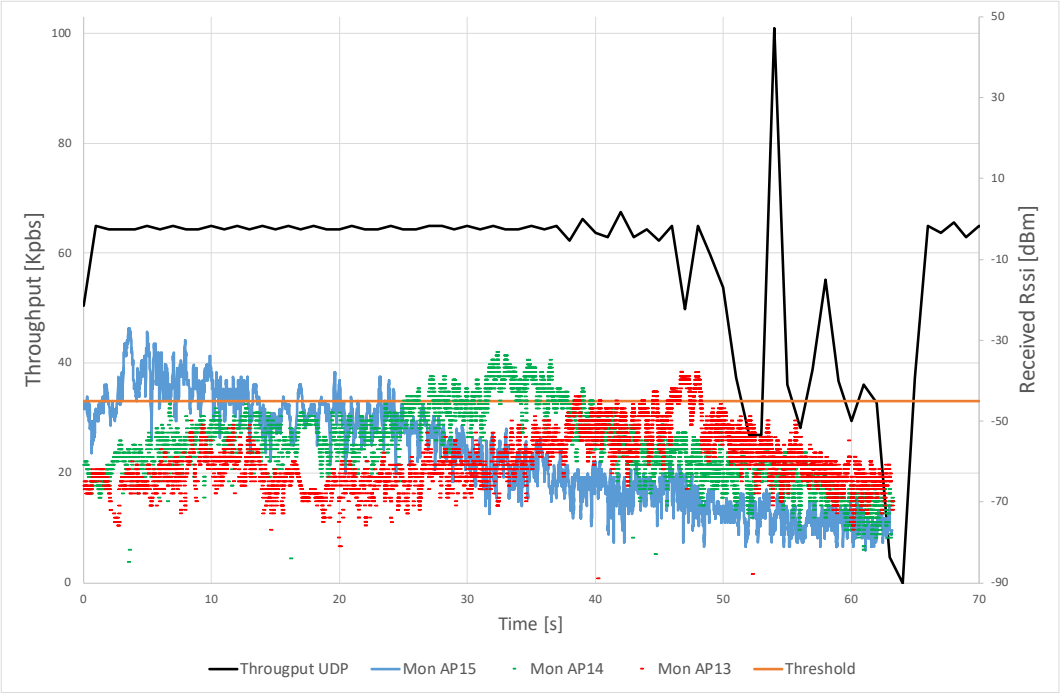


Figura 12. Tráfico recibido y potencias en 3 APs, solución nativa.

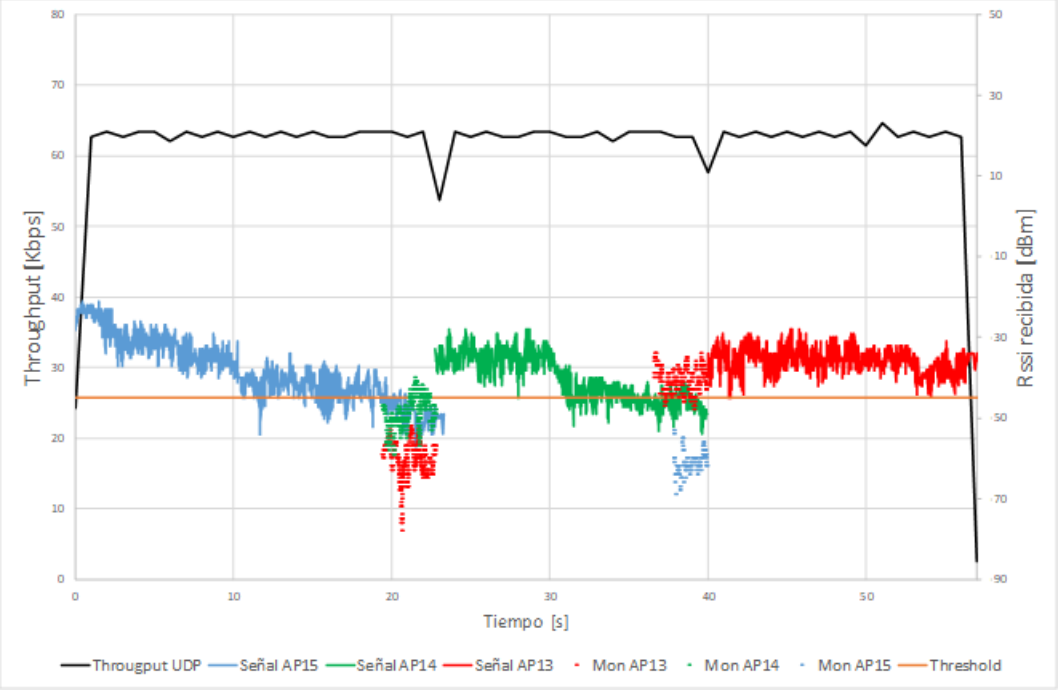


Figura 13. Tráfico recibido y potencias en 3 APs, solución Odin-Wi-5 reactiva.

Al utilizar el mismo escenario con MobilityManager, explotamos la disponibilidad de los puntos de acceso intermedios (Figura 13). Cuando la potencia del terminal asociado baja del umbral establecido (-45 dBm) en t=20s, el resto de APs escanean y notifican al controlador de la potencia con la que reciben a la STA. Se comparan los datos con la potencia que ha lanzado la petición, siendo en ese momento (t=23 s) cuando se realiza el *handover* si es necesario. La situación se repite en t=40 s.

Comparando con la solución nativa, observamos que en este caso sí estamos usando en cada momento el AP más cercano, que nos da mejor servicio. En esta realización, el tráfico recibido sufre una pequeña alteración en los momentos del *handover* (segundos 23 y 40). Esta caída contrasta con el comportamiento nativo, en el que las pérdidas son cuantiosas y se pierde la conexión.

Por lo tanto, con un traspaso entre los 30 y los 100 ms, Wi-5 y la aplicación MobilityManager nos permiten que el usuario no perciba el cambio de punto de acceso y su conexión se vea alterada mínimamente, consiguiendo el objetivo del *seamless handover*.

Finalmente, en la Figura 14, que incluye varios traspasos (se obtuvo recorriendo varias veces el escenario) podemos ver cómo en algunos casos, gracias al *buffer* del interfaz inalámbrico, incluso esas pérdidas se recuperan. Los traspasos (triángulos verdes) siempre van precedidos de órdenes de escaneo (rombos naranjas), coincidiendo de manera exacta el traspaso con las pérdidas. Estas varían según las condiciones del entorno y, en algunos casos favorables, son tan bajas que pasan inadvertidas como pérdidas o retransmisiones debidas a la propia transmisión inalámbrica.

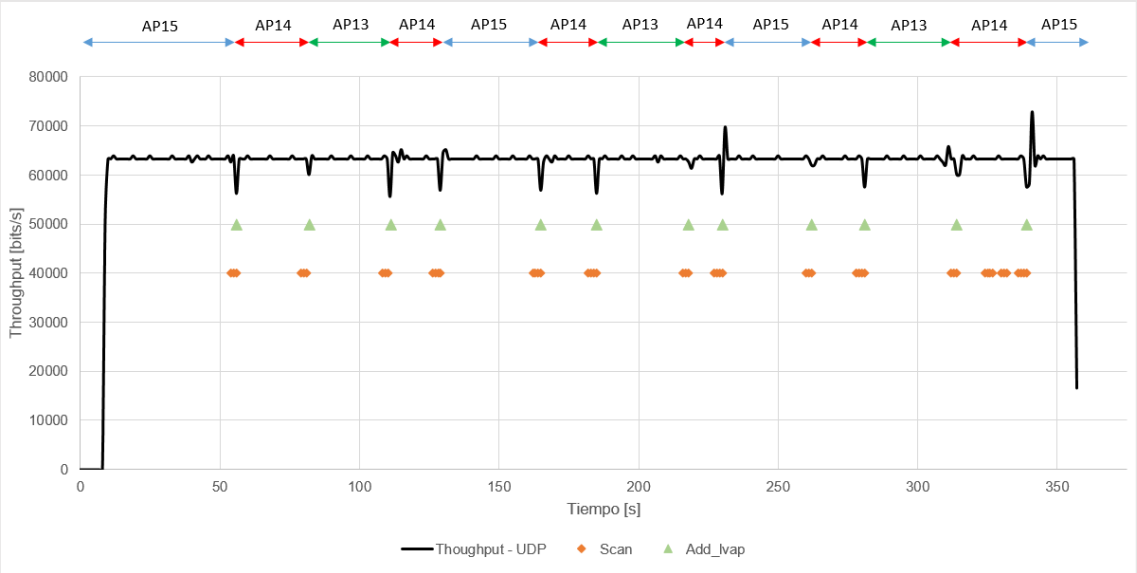


Figura 14. Tráfico recibido con varios traspasos, detalle de escaneo.

4. Asignación de canales

4.1 Problema

En el Capítulo 1 hablábamos de la gran densidad de redes inalámbricas que podemos encontrar en una localización (*Wi-Fi jungle*). En muchos casos, cada AP tiene un propietario distinto (por ejemplo, un bloque de apartamentos), pero en otros (un aeropuerto, un centro de negocios), los AP están gestionados por la misma entidad. Por ello, podemos ser capaces de coordinarlos y seleccionar una configuración que optimice la utilización del espectro radioeléctrico en la zona. Esta posibilidad mejoraría la experiencia de los usuarios y maximizaría los anchos de banda conseguidos por los terminales.

Para un escenario pequeño se puede recurrir a una selección manual de los canales, pero la inclusión de nuevos puntos de acceso (propios o ajenos) o cambios en el despliegue hacen necesaria una estrategia de automatización. Por ello, uno de los WP del proyecto Wi-5 se encarga del desarrollo de algoritmos que, dados como parámetros de entrada datos obtenidos por las funcionalidades de monitorización, facilitan la selección de canales utilizados para el despliegue [15].

4.2 Solución inicial

Partiendo de dichos algoritmos[27] desarrollados en Matlab por el grupo de la LJMU (Liverpool John Moores University), se realizará la integración con el controlador Odin-Wi-5 y se analizarán sus prestaciones. Los parámetros de entrada utilizados son el *pathloss* (pérdidas medidas en dBs) entre los diferentes puntos de acceso.

Para calcular el valor de esas pérdidas, nos basamos en la aplicación ShowMatrixOfDistancedBs[28]. En esta aplicación, el controlador ordena transmitir a un AP en un canal concreto unas tramas de administración (*beacon*) con una SSID especial (Figura 15). Mientras tanto, el resto de puntos de acceso escuchan en ese canal con el interfaz auxiliar, y comunican al controlador la potencia que reciben. El proceso se repite para cada AP, obteniéndose una matriz en la que se encuentran las “distancias en dB” entre cada par de puntos de acceso.

En agosto de 2017 se organizó en Zaragoza una reunión de los socios del proyecto involucrados en incorporar los algoritmos de selección de canal (se denominó la *2ª semana de la integración*[29]). Asistieron representantes de la LJMU, desarrolladores del algoritmo, de la empresa AirTies (líderes del WP5 centrado en la integración), así como miembros de la Universidad de Zaragoza. El objetivo era coordinar y resolver cuestiones técnicas para llevar a cabo el desarrollo de la aplicación para asignación de canales. Se obtuvieron sugerencias, que era necesario integrar en la solución.

[27] Algoritmo de selección de canal: https://github.com/Wi5/Wi-5-Channel-Assignment Acc. Nov. 2017
[28] ShowMatrixOfDistancedBs: https://github.com/Wi5/odin-wi5/wiki/Application-ShowMatrixOfDistancedBs Acc. Nov. 2017
[29] 2ª Semana de la Integración: http://www.wi5.eu/?p=688 Acc. Nov. 2017

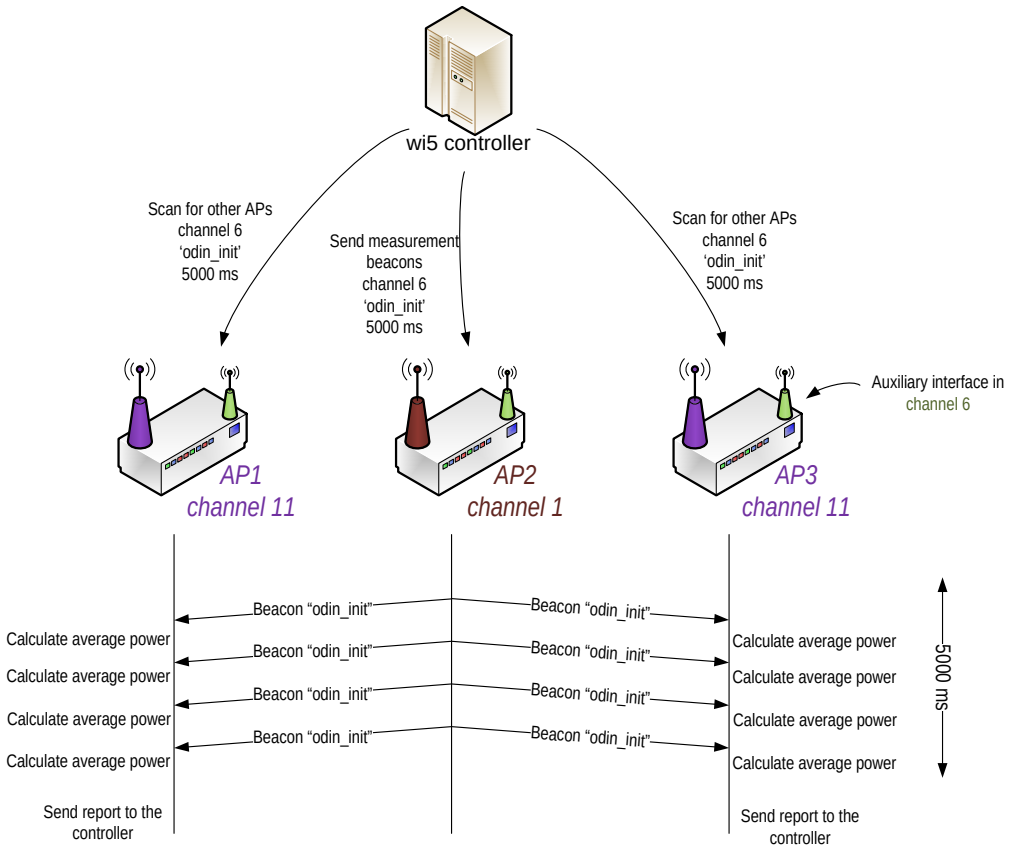


Figura 15. Procedimiento para el cálculo de matriz de distancias.

4.3 Mejoras introducidas, integración con el algoritmo y resultados

4.3.1 Integración de código Matlab

Para integrar el código de Matlab con nuestro controlador, implementado en Java, necesitamos hacer uso de la herramienta de compilación que nos ofrece el propio Matlab. Se trata de *MATLAB Compiler Runtime* (MCR), un entorno de ejecución donde se lanzarán las ordenes relacionadas con el algoritmo, pudiendo llamarlas directamente desde nuestra aplicación Java.

4.3.2 Modificaciones en agente y controlador

a) Nuevo *handler* de potencia transmitida. Para calcular las pérdidas entre APs, además de la potencia recibida entre ellos que nos brinda el procedimiento de la matriz de distancias, necesitamos saber la potencia de transmisión que utilizan los puntos de acceso. Al no contar con ese dato, fue necesario implementar un nuevo *handler* en el agente que corre en el AP. Se trata de un "*read_handler*", que nos comunica el valor de la potencia que utiliza el AP.

Una vez incluido en el agente, debemos dar al controlador la posibilidad de hacer esa petición (*Southbound*), por lo que fueron incluidas sendas funciones en el núcleo del mismo, así como en las capas de comunicación entre las aplicaciones y el controlador (*Northbound*).

b) Construcción de matriz y visualización. Partiendo de la aplicación de distancias antes mencionada, debemos almacenar los datos en una estructura que nos permita acceder a ellos para realizar cálculos, ya que en la versión inicial la información obtenida solamente se representaba por pantalla. Debe ser una estructura compatible con los tipos de dato que utilizará el algoritmo de Matlab.

Asimismo, es necesario mostrar de manera amigable la información, por lo que modificamos la sección encargada de mostrar la información por pantalla.

c) Modificación *handler* de cambio de canal. Finalmente, una vez decidido el nuevo canal para cada AP, se debe hacer un cambio coordinado, de forma que no sólo el AP, sino todas las STA asociadas, cambien simultáneamente. Para ello se modificó el *handler* encargado del cambio de canal. Se trata de un "*write_handler*", ya que en este caso estamos realizando cambios sustanciales en el comportamiento del AP. Era necesario que se contemplara la opción de mandar la cuenta atrás de *beacon* CSA a todas las estaciones asociadas. Una vez que la cuenta atrás llega a 0, las STA cambiarán de canal, y en ese momento se realiza también el cambio en el AP.

4.3.3 Comportamiento y resultados

Para el estudio de la implementación y el correcto funcionamiento de las modificaciones realizadas, se desarrollan tres aplicaciones[30]:

a) ChannelPrompt. Aplicación que pide por pantalla el canal que se quiere asignar al agente en cuestión. Se utiliza para comprobar que la estación asociada recibe los CSAs y cambia al canal elegido.

b) ChannelLoop. Bucle que cada 10 segundos cambia el canal del AP realizando un barrido desde el 1 al 11. Se ha utilizado para medir pérdidas y duración del cambio de canal. Para ello se ha generado un tráfico UDP de 50 pps de tamaño 80 bytes, obteniendo una ráfaga de pérdidas con un tamaño medio cercano a los 10 paquetes. Estimamos por tanto el tiempo de cambio de canal alrededor de los 200 ms (Figura 16). Podemos observar cómo el *throughput* cae momentáneamente en cada salto frecuencial. Este tiempo es superior al obtenido a la hora de realizar un *seamless handover*, ya que en este caso las STAs deben esperar a que el punto de acceso cambie realmente de canal (recordamos que en el *handover* cambiaban a otro AP que ya estaba trabajando en el canal destino), existiendo limitaciones *hardware* del propio AP.

[30] Applications for Channel Assignment https://github.com/Wi5/odin-wi5/wiki/Applications-for-Channel-Assignment Acc. Nov 2017

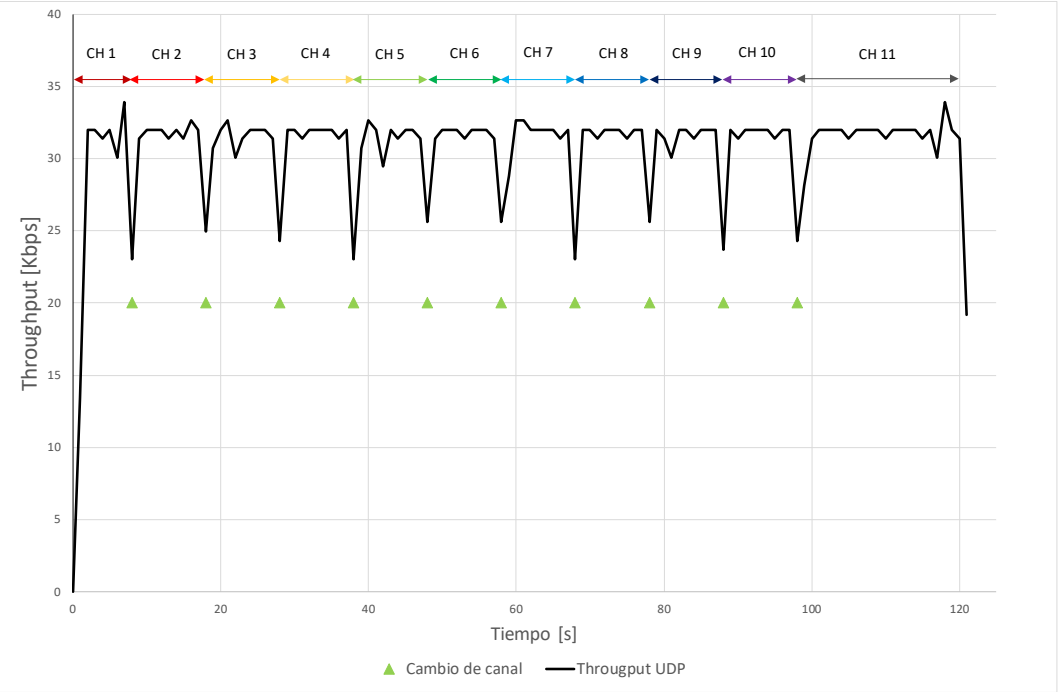


Figura 16. Tráfico recibido durante los cambios de canal del AP y sus STA asociadas.

c) ChannelAssignment. Es la aplicación principal. Se encarga de recopilar los datos de los diferentes puntos de acceso, calculando la matriz de *pathloss* y representando los resultados por pantalla en cada iteración, siguiendo el proceso descrito en la Figura 17.

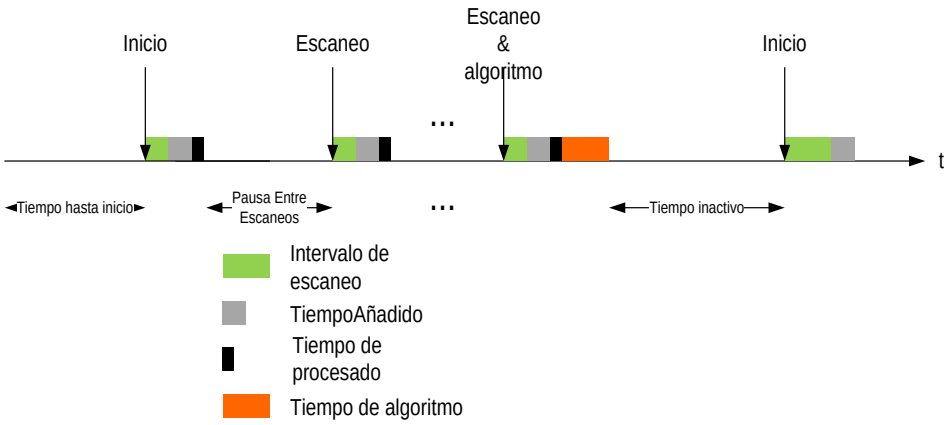


Figura 17. Ejecución temporal de ChannelAssignment.

Se realizan una serie de escaneos, con una pausa entre ellos. Al cumplirse el número de escaneos establecido en los parámetros, se ejecuta el algoritmo de asignación de canales y, si es necesario, ordena a los puntos de acceso que cambien al canal elegido junto con sus estaciones asociadas (Figura 18). Se añaden como parámetros el tiempo añadido tras el escaneo y el tiempo tras la ejecución del algoritmo, consiguiendo flexibilidad a la hora de planificar la ejecución.

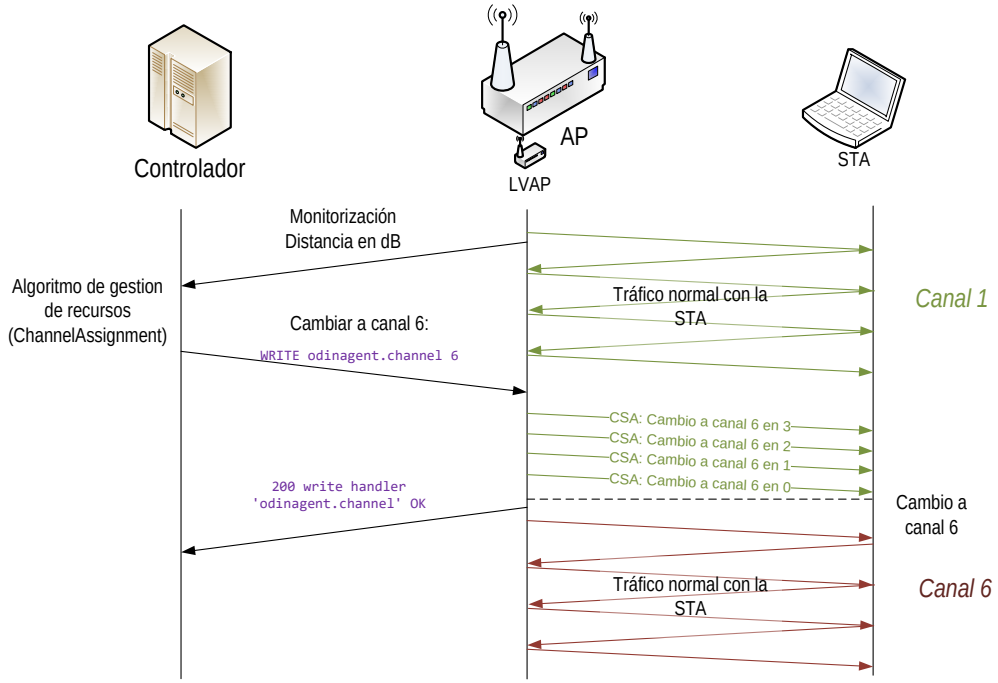


Figura 18. Esquema de funcionamiento ChannelAssignment.

En este caso el algoritmo únicamente tiene en cuenta información de los puntos de acceso pertenecientes a la red Wi-5, siendo de utilidad en un despliegue inicial o en el momento de modificar la topología del mismo.

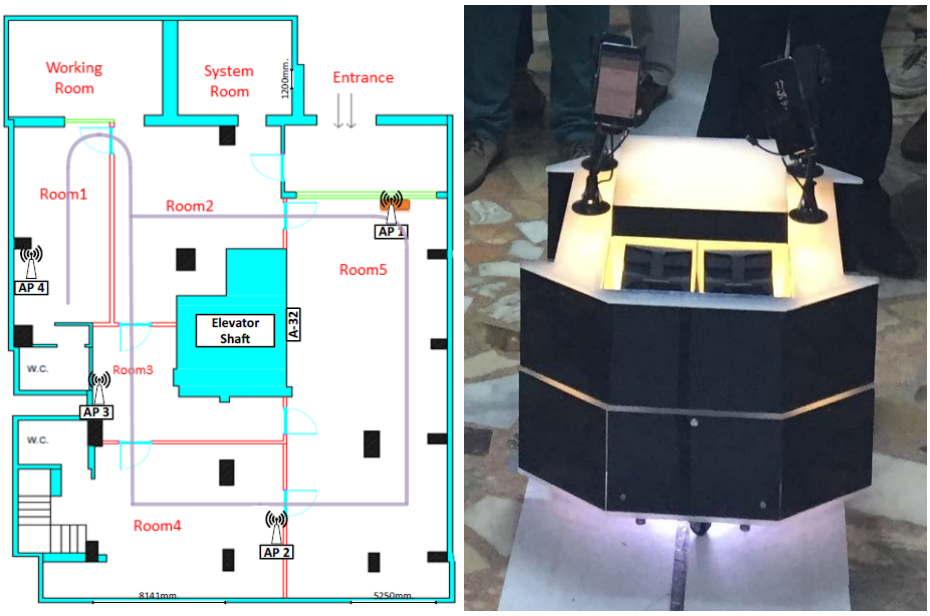
Con esta aplicación se cumple el objetivo de integrar el algoritmo de asignación de canales, demostrando la capacidad del controlador para conseguir información de la red inalámbrica, y utilizarla para modificar la configuración del despliegue.

5. Implementación de una aplicación de balanceo de carga

5.1 Problema

Tras su análisis y mejora, la aplicación de *handover* reactivo MobilityManager se analizó al detalle y quedó lista para las pruebas en un entorno controlado. Este examen exhaustivo del comportamiento del sistema es esencial para comprobar su viabilidad en un contexto real.

Estar pruebas fueron realizadas por AirTies (uno de los socios del proyecto Wi-5) en Estambul. Se desplegó la solución en un piso piloto dotado de puntos de acceso distribuidos en dos plantas (Figura 19 a). Para automatizar los desplazamientos y conseguir realizaciones repetibles, se hizo uso de un robot que portaba varios terminales móviles (Figura 19 b) siguiendo un camino fijo por el piso.



(a) (b)

Figura 19. Distribución piso piloto Estambul y robot utilizado para desplazamientos.

El informe elaborado por AirTies tras las pruebas indicaba problemas de escalabilidad en el caso de contar con numerosas STAs asociadas a un punto de acceso y que varias de ellas se desplazaran a la vez. El problema se debe a que el controlador recibe simultáneamente varios *publish,* y algunos quedan en espera hasta que el primer terminal haya sido escaneado y realizado el *handover* si llega el caso. Otro problema de esta aplicación es que es difícil de compatibilizar con el balanceo de carga: las STAs simplemente se asocian al AP más cercano, sin tener en cuenta si está más o menos saturado.

Por lo tanto, se planteó el desarrollo de una aplicación proactiva que escanee periódicamente el entorno usando el interfaz auxiliar y tome decisiones para realizar *handover*. En un principio se ha implementado tomando como referencia la potencia recibida por los puntos de acceso con el objetivo de comparar los resultados con MobilityManager, pero también se podría utilizar la información de monitorización interna

y externa para realizar labores de balanceo de carga. Asimismo, recopilando datos de la funcionalidad de detección de flujos, podría dividir las STAs en función del servicio que están utilizando.

5.2 Solución inicial

En este caso nos basamos en la aplicación ShowScannedStationsStatistics[31], que utiliza la funcionalidad de monitorización externa para escanear todos los canales y enviar al controlador las estadísticas de ese periodo.

Este método nos facilita una gran cantidad de datos, siendo capaces de ver el estado del canal (número de puntos de acceso y estaciones en él), la actividad de un punto de acceso concreto o de un terminal, asociado a nuestra red o a una ajena (Figura 20).

```
(...)
Station MAC: 40:A5:EF:05:9B:A0
               num packets: 5
               avg rate: 24000 kbps
               avg signal: -70 dBm
               avg length: 116 bytes
               air time: 0.219111111111 ms
               init time: 1501053420.460779354 sec
               end time: 1501053425.467050619 sec
               AP of client: /192.168.1.14
               Channel of AP: 6
               Code: Wi-5 STA
(...)
```

Figura 20. Detalle información monitorización externa.

5.3 Implementación de aplicación y resultados

5.3.1 Aplicación SmartApSelection

Nuestro caso es más concreto, ya que solo debemos monitorizar los canales utilizados y almacenar la información que se utilizará para decidir el punto de acceso al que asociar las STAs. Se desarrolla para ello una nueva aplicación llamada SmartApSelection[32], que tras recopilar información de los APs que componen su despliegue, ordena escanear en los canales utilizados y compone una base de datos con la información recopilada. Se añade la posibilidad de dar un peso específico a los valores anteriores, para crear una correlación que nos ofrezca un comportamiento mejorado. La aplicación decide mediante un algoritmo si el punto de acceso al que la estación está asociada es el idóneo, realizando el cambio si es necesario.

5.3.2 Nuevo *handler* de potencia de STAs recibida

Debido a la gran cantidad de información que el controlador recibía del agente en cada AP, se producían unos retardos de procesado de unos 7 segundos. Esto no era aceptable para una aplicación que debe gestionar usuarios andando en un escenario

[31] Aplicación ShowScannedStationsStatistics: https://github.com/Wi5/odin-wi5/wiki/Application-ShowScannedStationsStatistics Accedido noviembre 2017
[32] Aplicación SmartApSelection: https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection Accedido noviembre 2017

con varios APs, por lo que se decidió optimizar los datos que se enviaban al controlador. Para ello se creó un nuevo *handler*, dentro del grupo asignado a los "*read_handler*", que únicamente transmite la información relativa a la RSSI recibida de terminales.

Asimismo, se opta por concentrar el procesado de los datos recibidos tras la recepción de todos ellos, disminuyendo en gran medida el tiempo dedicado a la organización de la información recopilada.

5.3.3 Comportamiento y resultados

La mejora conseguida respecto a la aplicación reactiva es significativa, ya que con este método conseguimos la información de todas las estaciones en el mismo periodo de escaneo, y se toman todas las decisiones en el mismo momento, evitando los problemas de escalabilidad (Figura 21).

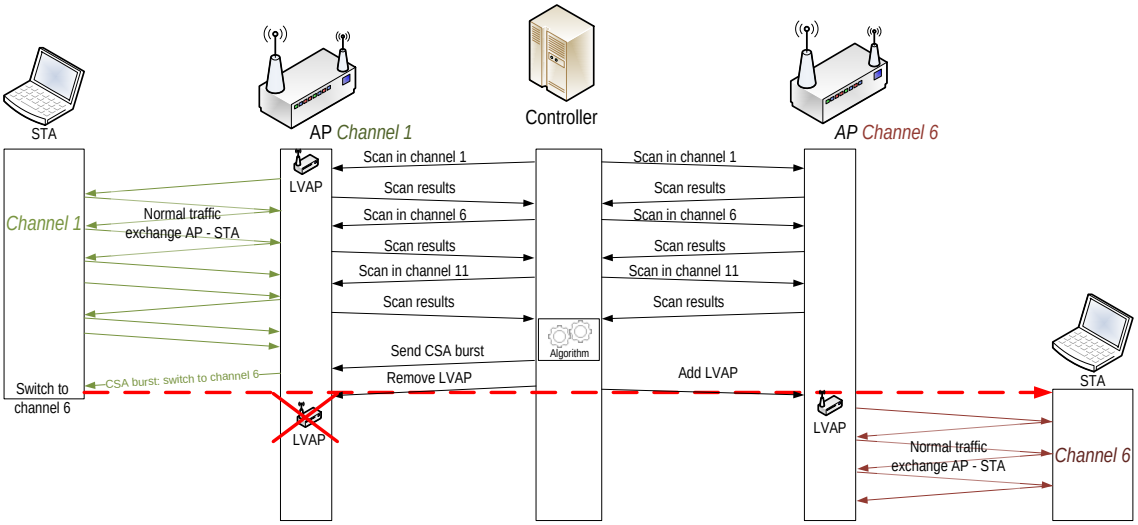


Figura 21. Esquema de funcionamiento SmartApSelection.

En el inicio del desarrollo, al trabajar con la gran cantidad de información obtenida de la monitorización externa, los tiempos de procesado se elevaban por encima de los 7 segundos, muy superiores a lo deseado para una aplicación proactiva. Tras el uso del nuevo *handler* y la optimización del procesado (que ronda el milisegundo), se consiguió un escaneo de 3 canales (200 ms en cada uno de ellos) de entre 2 y 3 segundos, perfecto para gestionar usuarios que se desplacen andando.

Podemos ver en la Figura 22 el interfaz de usuario que se ha programado para el controlador, en el que la información más actual está en todo momento disponible y nos informa con códigos de color del valor del RSSI para cada AP, y del estado y acción que realiza la aplicación (por ejemplo, indicando que se va a hacer un *handoff*)[33].

[33] Se puede ver un vídeo de la aplicación en funcionamiento en: https://www.youtube.com/watch?v=FWUsw8AhDQU Accedido noviembre 2017

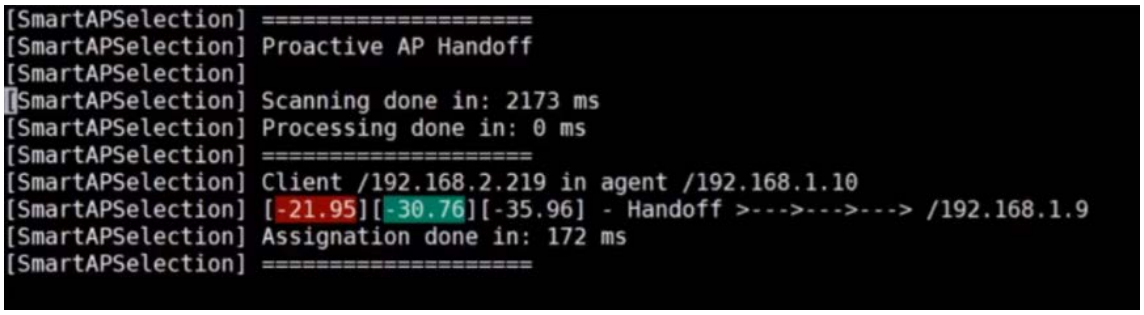


Figura 22. Interfaz de usuario SmartApSelection.

Comparando el comportamiento con la solución reactiva (Figura 23), podemos destacar las siguientes diferencias (por claridad, se ha reproducido la Figura 13 como Figura 23 a):

a) Escaneo (líneas discontinuas): En la versión reactiva (Figura 22 a), únicamente se escanea en busca del terminal en cuestión, y en el momento que se recibe el *publish* (por ejemplo, a partir de t=20). Sin embargo, en la aplicación proactiva (Figura 22 b) se escanea a todas las estaciones y constantemente en ciclos de 2-3 segundos.

b) Correlación: Utilizando la aplicación reactiva, al calcular la RSSI que se recibe de la estación a escanear, se hace uso de la media móvil acumulada. Sin embargo, para la aplicación proactiva se desea mantener un registro de los datos históricos. Para ello se ha definido un parámetro llamado *weighted* RSSI (*wRSSI*, RSSI ponderado). Su valor se va actualizando según se reciben los valores de cada intervalo de escaneo.

$RSSI_{i,j}$ es el valor medio del RSSI medido en la STA_i recibido por el AP_j.

Se define una matriz de los valores de RSSI ponderados:

$$\begin{pmatrix} wRSSI_{1,1} & wRSSI_{1,2} & wRSSI_{1,3} & \dots & wRSSI_{1,k} \\ wRSSI_{2,1} & wRSSI_{2,2} & wRSSI_{2,3} & \dots & wRSSI_{2,k} \\ wRSSI_{3,1} & wRSSI_{3,2} & wRSSI_{3,3} & \dots & wRSSI_{3,k} \\ \dots & \dots & \dots & \dots & \dots \\ wRSSI_{m,1} & wRSSI_{m,2} & wRSSI_{m,3} & \dots & wRSSI_{m,k} \end{pmatrix} \quad (1)$$

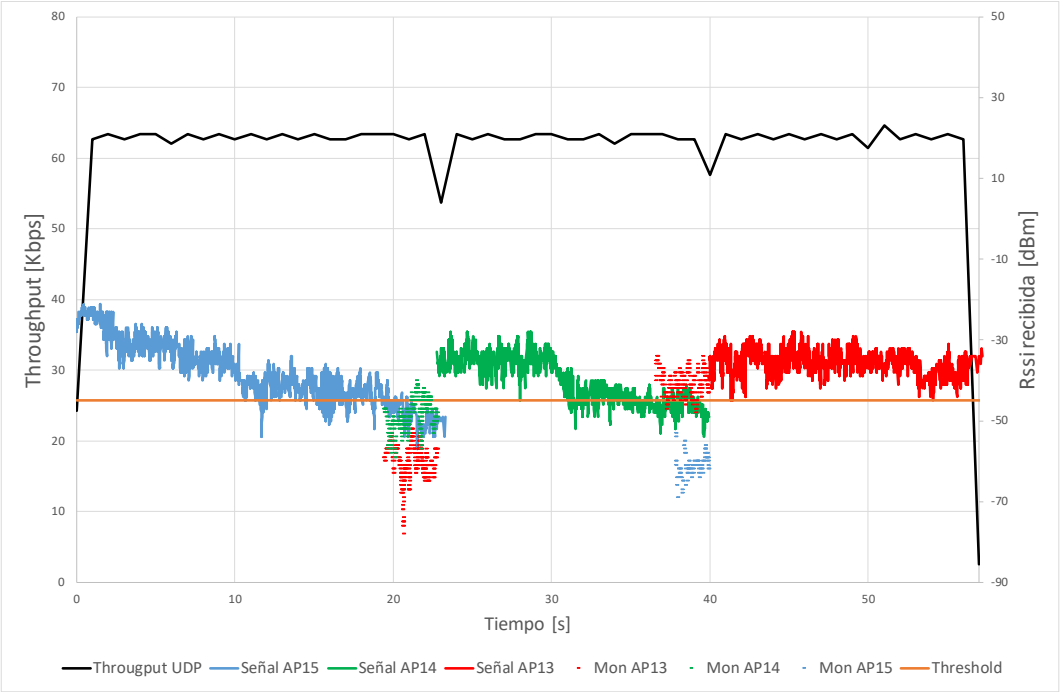
En cada iteración, el nuevo valor se obtiene así:

$$w\mathrm{RSSI}_{i,j}^{N+1} = \alpha \bullet w\mathrm{RSSI}_{i,j}^{N} + (1 - \alpha) \bullet \mathrm{RSSI}_{i,j}^{N} \quad (2)$$

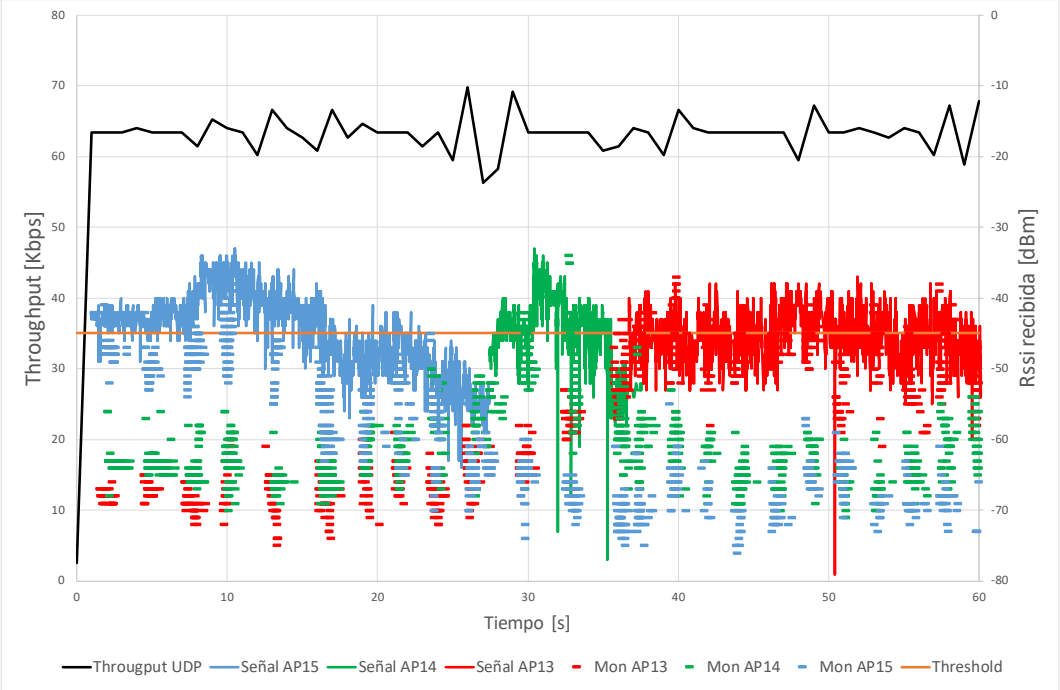
$RSSI_{i,j}^{N}$ es el valor medio obtenido en la última iteración (es decir, la media de la potencia con que el AP_i recibe las tramas de la STA_j en la iteración *N*), y $wRSSI_{i,j}^{N}$ representa el valor *N* del RSSI ponderado (*wRSSI*).

El parámetro α se utiliza para dar un peso al valor histórico, y otro peso al último valor obtenido. De esta forma se puede hacer un balance entre la información histórica y la última muestra, consiguiendo un sistema flexible.

En la Figura 23 b podemos ver cómo se ha sido conservador en el cambio entre AP15 y AP14 (t=27s), mientras que en el siguiente se ha realizado un cambio más ágil (t=36s) debido a que el valor histórico era menor.



(a) Comportamiento reactivo



(b) Comportamiento proactivo

Figura 23. Comparativa soluciones reactiva y proactiva.

c) Trafico en el plano de control: Por último, la versión proactiva genera mucho más tráfico dirigido al controlador que la reactiva. El hecho de comunicar las estadísticas constantemente nos hace prácticamente triplicar el ancho de banda utilizado (pasamos de 21 a 57kbps para 3 APs en diferentes canales). De todas formas, esto no supone un problema porque se trata de enlaces cableados entre el AP y el controlador.

6. Conclusiones y líneas futuras

6.1 Conclusiones

En este trabajo se ha analizado el funcionamiento de una solución de código abierto que permite coordinar una red inalámbrica mediante tecnologías SDN. Las tareas realizadas, dentro del proyecto europeo Wi-5, han contribuido principalmente en los WP3 (*Smart AP Solutions*) y WP5 (*Integration and Field Trials*).

Se ha modificado, ajustado y estudiado el rendimiento del sistema en un entorno distribuido, adquiriendo conocimientos sobre los diversos protocolos utilizados, así como de la arquitectura y lenguajes de programación en los que se basa la solución.

Se ha integrado el trabajo de otros socios del proyecto Wi-5 en el despliegue SDN, en concreto, un algoritmo de asignación de canales, demostrando la flexibilidad del sistema y su capacidad de recopilar información y utilizarla para tomar decisiones que mejoren el rendimiento global.

Se han desarrollado aplicaciones que cumplen con los objetivos iniciales: mejorar los traspasos cuando un usuario se mueve en un entorno con varios AP, siguiendo una aproximación reactiva y otra proactiva. De esta forma se puede proporcionar un servicio similar al que brindan soluciones propietarias, pero con un coste significativamente menor.

Las soluciones integran las propuestas de mejora hechas por el resto de socios del proyecto, a los que también se ha dado soporte, por ejemplo, de cara a las pruebas de validación. Asimismo, se ha realizado una demostración ante el *Operator Board* del proyecto (La Haya, Países Bajos, noviembre 2017) en la que se ha mostrado el rendimiento del sistema. Los resultados obtenidos formarán parte del informe final del proyecto ante la Comisión Europea, así como de varias publicaciones científicas[34].

6.2 Líneas futuras

Con los objetivos principales alcanzados, podríamos continuar el estudio de esta manera:

- Realización de baterías de pruebas para modelar cuantitativamente los traspasos, realizando una comparativa según los fabricantes de las tarjetas inalámbricas.

- Comprobar la mejora introducida por el algoritmo de asignación de canales en relación a una selección inicial con solapamiento espectral.

Por otro lado, podemos destacar una serie de mejoras a incorporar en la solución:

- Coordinación de aplicaciones: Actualmente la ejecución simultánea de varias aplicaciones de gestión en Odin no está contemplada. Se propone estudiar la opción de coordinarlas para evitar contradicciones en las decisiones de cada una de ellas.

[34] Actualmente se está preparando una publicación que se enviará a la revista IEEE Access, que recogerá las pruebas presentadas, y los resultados obtenidos por otros socios en la validación.

- Implementación eficiente del controlador: En nuestro caso, al tratarse de una prueba de concepto, se ha utilizado un controlador en Java debido a su flexibilidad. El siguiente paso puede ser optar por un controlador embebido que necesite menos recursos y procese los datos más ágilmente, utilizando como dispositivo una Raspberry o similar.

- Desarrollo de interfaz de usuario: Implementación de un interfaz de usuario amigable para gestionar el controlador. Se sugiere almacenar toda la información en una base de datos SQL y utilizar una interfaz Web para acceder a ellos.

- Mejora del protocolo Odin: Uno de los puntos donde se mejoraría la eficiencia del sistema es modificar el protocolo Odin. Enviando los datos con formato, en vez de en texto plano como se realiza ahora, permitiría un acceso más rápido.

- Uso de controladores locales: Para mejorar la escalabilidad, se puede estudiar la opción de incluir una jerarquía de controladores coordinados, para evitar tener un número elevado de puntos de acceso bajo su mando. De esa manera se evitarían retardos y tiempos de procesado, delegando funciones a los miembros de escalafones inferiores.

6.3 Planificación del trabajo

Por último se incluye la Figura 24, donde se expone la planificación del trabajo realizado dividido en las diferentes tareas que se han desarrollado.

Id.	Nombre de tarea	Comienzo	Fin	Duración
1	Analisis y documentación	17/04/2017	26/05/2017	6s
2	Montaje escenario de pruebas	29/05/2017	02/06/2017	1s
3	Pruebas y mejoras MobilityManager	05/06/2017	23/06/2017	3s
4	Pruebas y mejoras Monitorización	26/06/2017	13/07/2017	2,8s
5	Captura y análisis de resultados	17/07/2017	20/11/2017	18,2s
6	Integración Asignación de canales	07/08/2017	25/08/2017	3s
7	Desarrollo ChannelAssignment	28/08/2017	15/09/2017	3s
8	Desarrollo SmartApSelection	09/10/2017	27/10/2017	3s
9	Demostración en La Haya (Países Bajos)	06/11/2017	09/11/2017	,8s
10	Memoria	09/10/2017	24/11/2017	7s

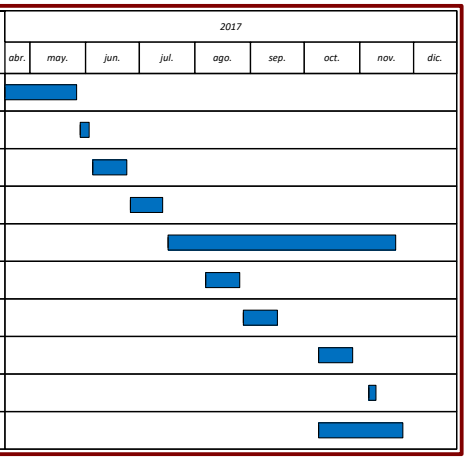


Figura 24. Diagrama de Gantt del trabajo.

Referencias

[1] IEEE 802.11 group. “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”, IEEE Std 802.11v-2011, pp. 1-433, Feb. 2011

[2] F. den Hartog, J. de Nijs, “The role of regulation in preventing Wi-Fi over-congestion in densely populated areas,” Australian Journal of Telecommunications and the Digital Economy, vol. 5, no. 2, Jun. 2017.
Disponible en http://telsoc.org/ajtde/index.php/ajtde/article/view/93

[3] A. Mishra, M. Shin, W. A. Arbaugh, “An empirical analysis of the IEEE 802.11 MAC layer handoff process”, Computer Communication Review, pp. 93-102

[4] H2020 Wi-5 Project (What to do With the Wi-Fi Wild West) – www.wi5.eu [Accedido nov 2017].

[5] E.Haleplidis, K. Pentikousis, S. Denazis, J.H. Salim, D. Meyer, O. Koufopavlou. “Software-defined networking (SDN): Layers and architecture terminology”. No. RFC 7426. 2015

[6] Nick McKeown y col. “OpenFlow: Enabling Innovation in Campus Networks”. SIGCOMM Comput. Commun. Rev. 38.2, pp. 69-74, Mar. 2008

[7] Emerson Network Power and Ponemon Institute, "Cost of Data Center Outages", Ene. 2016

[8] Mahalingam, Mallik, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Lawrence Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks. No. RFC 7348. 2014

[9] Y. Grunenberger, F. Rousseau, “Virtual Access Points for Transparent Mobility in Wireless LANs”, WCNC, IEEE, p. 16.

[10] J. Schulz-Zander, P. L. Suresh, N. Sarrar, A. Feldmann, T. Hhn, R. Merz, “Programmatic Orchestration of WiFi Networks”. G. Gibson, N. Zel- dovich (Eds.), USENIX Annual Technical Conference, USENIX Association, pp. 347-358, 2014

[11] Navajas, J. F., Villarreal, L. S., Mas, J. R., & Medina, J. M. S., “Mejora de la Calidad en Redes WLAN Coordinadas a través de SDWN”, JITEL 2017

[12] Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F. “The Click modular router”. ACM Transactions on Computer Systems (TOCS), 18(3), pp.263-297, 2000.

[13] Alessio Botta, Alberto Dainotti, Antonio Pescapé. “A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios”, Comput. Netw 56.15 pp.3531-3547, Oct. 2012.

[14] Luis Sequeira, Juan Luis de la Cruz, Jose Ruiz-Mas, Jose Saldana, Julian Fernandez-Navajas, Jose Luis Almodovar, "Building a SDN Enterprise WLAN Based On Virtual APs", IEEE Communications Letters, vol. 21, no. 2, pp. 374-377, Feb. 2017.

[15] Seyedebrahimi, M., Bouhafs, F., Raschellà, A., Mackay, M., & Shi, Q. (2016, June). “SDN-based channel assignment algorithm for interference management in dense Wi-Fi networks”. Networks and Communications (EuCNC), 2016 European Conference on (pp. 128-132). IEEE.

Acrónimos

AP – *Access Point*

BSSID – *Basic Service Set Identifier*

CSA – *Channel Switch Announcement*

DHCP – *Dynamic Host Configuration Protocol*

D-ITG – *Distributed Internet Traffic Generator*

ESS – *Extended Service Set*

IEEE – *Institute of Electrical and Electronics Engineers*

IP – *Internet Protocol*

IRTF – *Internet Research Task Force*

LVAP – *Light Virtual Access Point*

MAC – *Media Access Control*

MCR – MATLAB *Compiler Runtime*

QoS – *Quality of Service*

RSSI – *Received Signal Strength Indicator*

SDN – *Software Defined Networking*

SDNRG – *Software Defined Networking Research Group*

SDWN – *Software Defined Wireless Network*

SOHO – *Small Office Home Office*

SSID – *Service Set Identifier*

STA – *(Wireless) Station*

TCP – *Transmission Control Protocol*

UDP – *User Datagram Protocol*

VLAN – *Virtual Local Area Network*

VoIP – *Voice over Internet Protocol*

VXLAN – *Virtual Extensible Local Area Network*

WI-5 – *What to do With the Wi-Fi Wild West*

WLAN – *Wireless Local Area Network*

WP – *Work Package*

wRSSI – *Weighted Received Signal Strength Indicator*

Lista de figuras

Anexo A:
Instalación ESXi en Intel Nuc, alojamiento de máquinas virtuales

A la hora de instalar el sistema operativo encargado de dar el servicio de virtualización, podemos encontrarnos con problemas de incompatibilidad hardware. En nuestro caso, el problema residía en la controladora de almacenamiento del dispositivo.

Para resolver el problema se debe hacer uso de la herramienta ESXi-Customizer[35]:

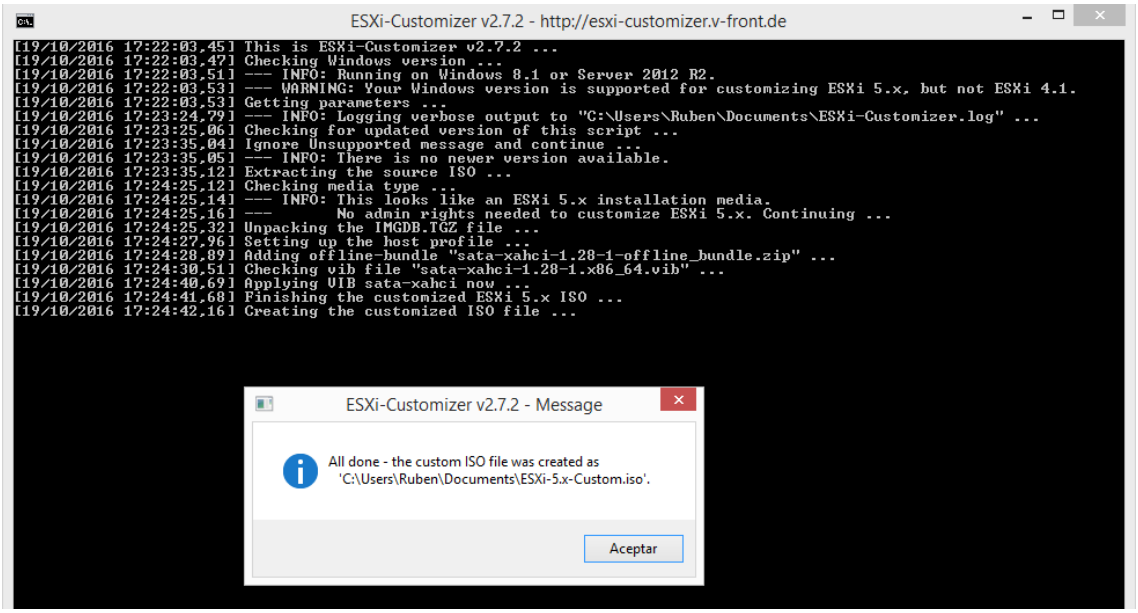


Figura A. Creación de imagen ISO para la instalación correcta de ESXi.

Con la imagen de disco parcheada el sistema reconoce el disco SSD y se completa la instalación sin errores.

[35] ESXi Customizer: http://www.virten.net/2015/03/esxi-6-0-image-for-intel-nuc/ Accedido noviembre 2017

Anexo B:
Configuración de red vSphere

En el entorno de virtualización VMware ESXi se encuentran las siguientes máquinas Debian:

- Controlador: Encargada de gestionar la red inalámbrica, acceso a plano de control.
- DHCP: Con acceso al plano de datos, realiza la tarea de servidor DHCP y Router de dicho plano.
- Detector: Agente encargado de la detección de flujos en el plano de datos. Tambien tiene acceso al plano de control para informar al controlador.
- Game-Server: Servidor de juegos utilizado para medir el comportamiento generado por diferentes aplicaciones de ocio.
- Router: Sin uso ya que su rol lo realiza la máquina DHCP.

Para lograr el acceso a los diferentes planos utilizados y gestionar las maquinas desde una red independiente a estos, se utilizan dos conmutadores virtuales (vSwitch). Podemos observar en la Figura B como el vSwitch0 permite utilizar el interfaz físico del Nuc para dar acceso de gestión a las diferentes máquinas. Respecto a el *vSwitch2*, nos ayuda a crear el enlace entre el Servidor DHCP y el Detector, que facilita el acceso al plano de datos a este último.

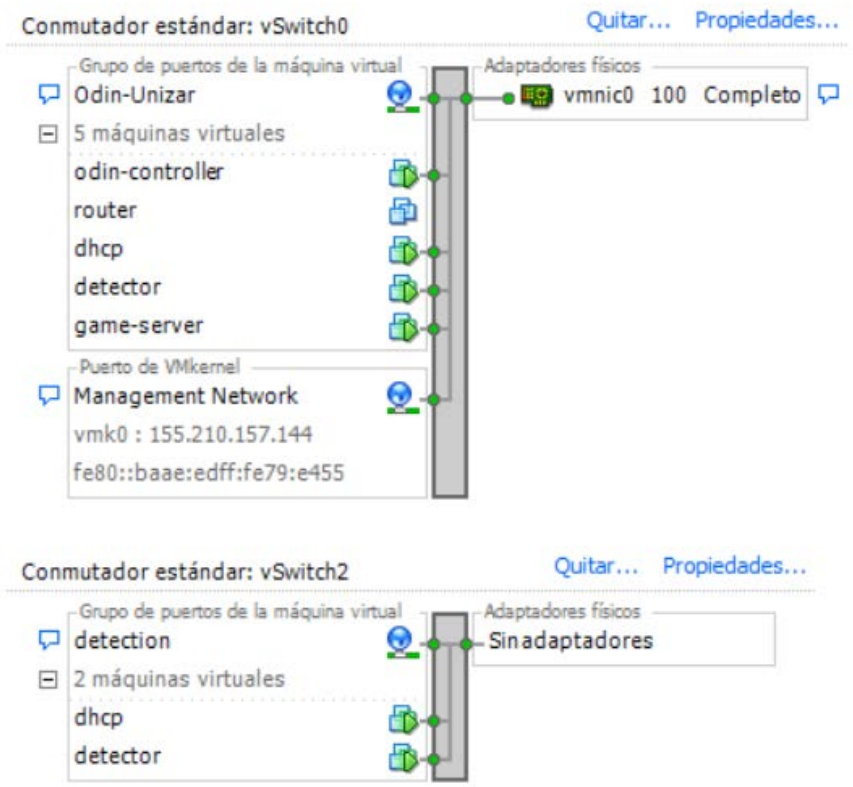


Figura B. Configuración vSwitch con vSphere.

Para el resto de conexiones se utilizan adaptadores USB-Ethernet asignados a cada máquina virtual.

Anexo C:
Configuraciones OpenWrt

Configuración nativa Routed AP:

```
config wifi-device  radio0
        option type     mac80211
        option channel  1
        option hwmode   11g
        option path     'platform/ar934x_wmac'
        option htmode   HT20
        option txpower '10'
        option country 'US'
        option disabled 0

config wifi-iface
       option device   radio0
       option network  wifi
       option mode     ap
       option ssid     wi5-demo
       option encryption none

(…)
```

Figura C.1. Modificaciones en /etc/config/wireless

```
(…)

config interface 'wifi'
       option proto 'static'
       option netmask '255.255.255.0'
       option ipaddr '192.168.4.1'

(…)
```

Figura C.2. Modificaciones en /etc/config/network

```
(…)

config zone
        option name     wifi
        list network    'wifi'
        option input    ACCEPT
        option output   ACCEPT
        option forward  REJECT

config 'forwarding'
        option src      'wifi'
        option dest     'wan'

(…)
```

Figura C.3. Modificaciones en /etc/config/firewall

Creando un interface wifi, asignándole una dirección IP y permitiendo que los paquetes se reenvíen hacia la red WAN, el AP se comporta de forma nativa.

Configuración Wi-5 con puerto trunk:

A la hora de conectar los puntos de acceso con los diferentes planos (gestión, control y datos), se optó en primer lugar por utilizar los diferentes puertos del *switch* incluido en el AP. En un despliegue real, el uso de tres cables para cada dispositivo es inviable, por lo tanto, se configuró el puerto 5 como *trunk*, enlace que transporta diferentes VLANs.

```
(…)
config switch
        option name 'switch0'
        option reset '1'
        option enable_vlan '1'
        option enable_learning '0'

config switch_vlan
        option vlan '1'
        option vid  '1'
        option ports '2 5t 0t'
        option device 'switch0'

config switch_vlan
        option vlan '2'
        option vid  '2'
        option ports '3 5t 0t'
        option device 'switch0'

config switch_vlan
        option vlan '3'
        option ports '4 5t 0t'
        option device 'switch0'

config switch_vlan
        option vlan '4'
        option ports '5t 0t'
        option device 'switch0'

config switch_vlan
        option vlan '5'
        option vid  '5'
        option ports '1 5t 6'
        option device 'switch0'

(…)
```

Figura C.4. Modificaciones en /etc/config/network

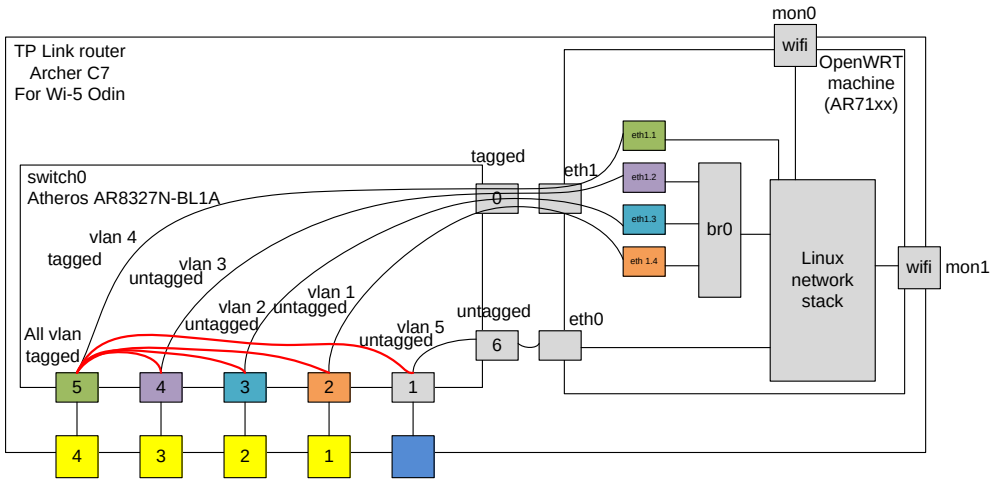


Figura C.5. Conexión interna del switch tras configuración *trunk*.

En la figura C.5 podemos ver como todas las VLAN pueden encaminarse hacia el puerto 5 (correspondiente al puerto físico 4) con su respectivo TAG. La condición para que funcione el despliegue es utilizar un switch que soporte VLAN como núcleo de las comunicaciones de la solución.

Anexo D:
Configuración del Switch con Trunking

Debido a la configuración *trunk* explicada en el Anexo C, se utiliza como núcleo del despliegue un switch capaz de trabajar con VLANs. Se configuran los tres planos (Gestión, Control y Datos) añadiendo los puertos 13, 14, 15, 23 y 24 como miembros etiquetados de las respectivas VLAN.

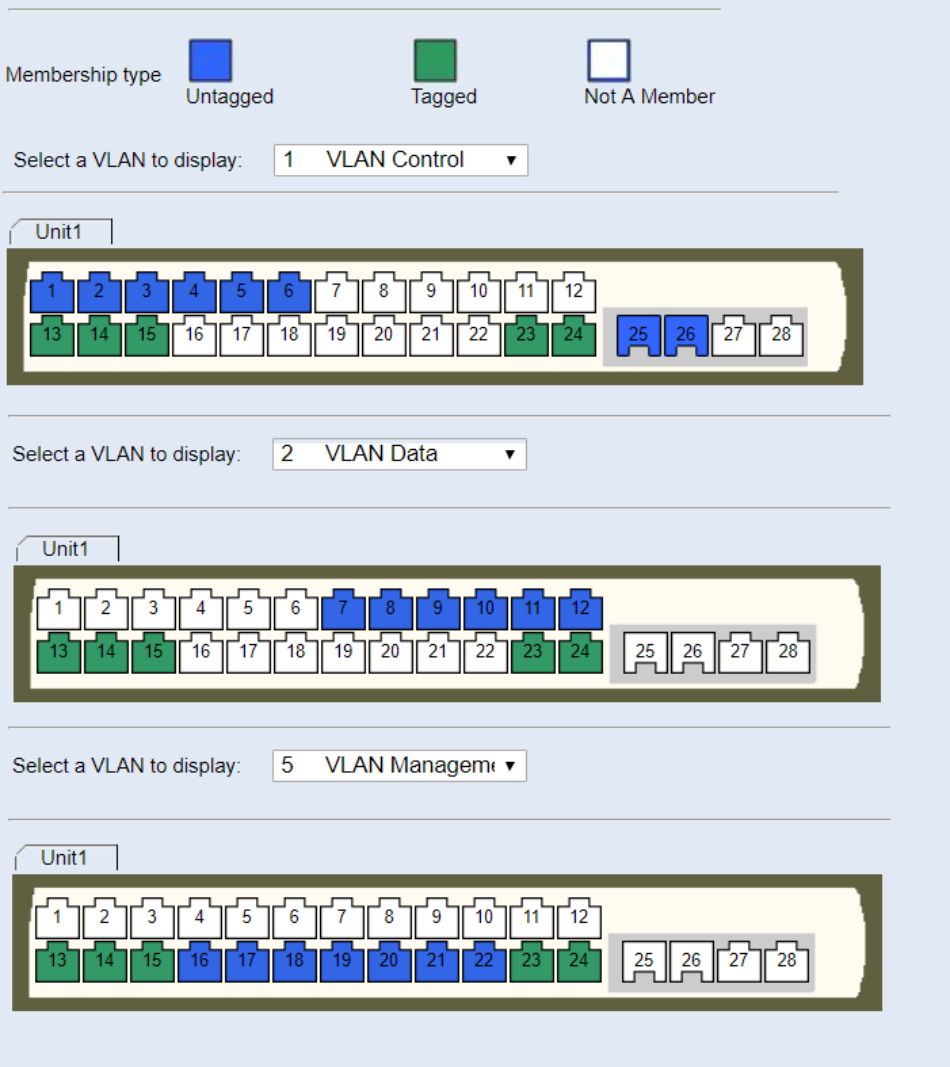


Figura D. Configuración VLAN en Switch con puertos trunk.

Anexo E: Esquema Click Modular Router

El agente de Odin se encuentra integrado como un módulo de *Click Modular Router*. En la Figura E podemos observar la estructura completa del programa.

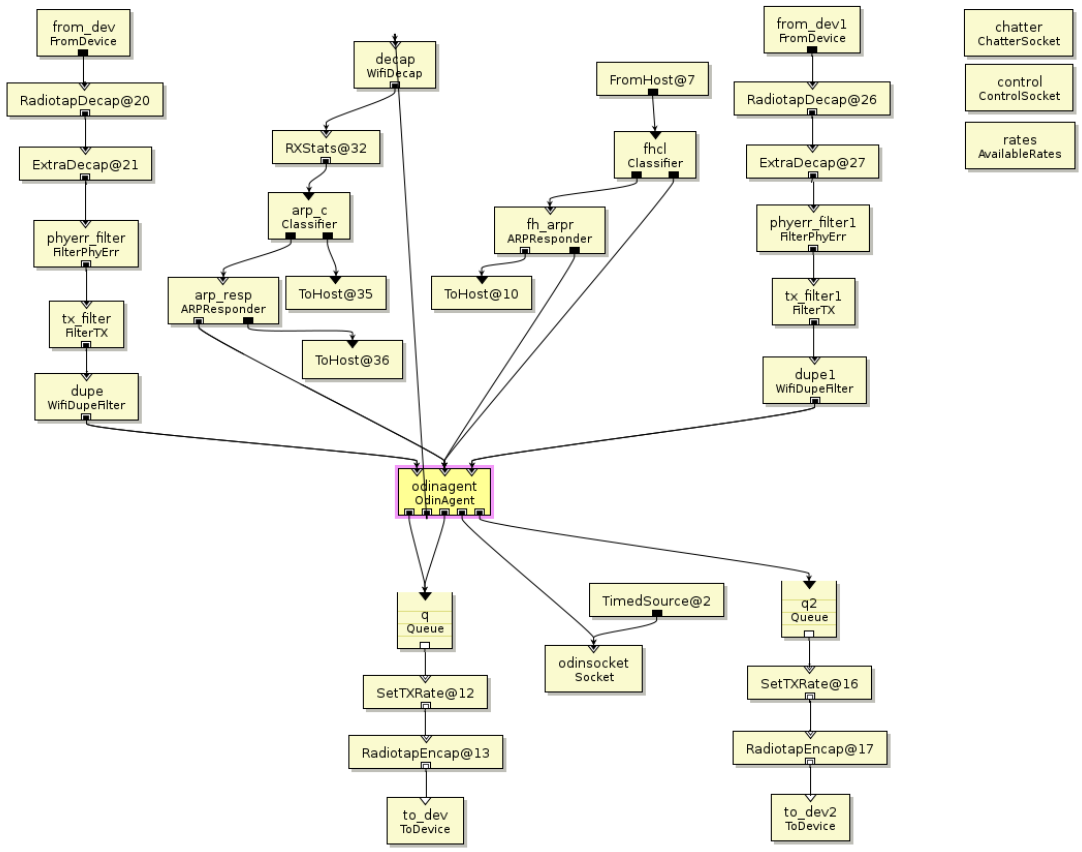


Figura E. Esquema de *Click Modular Router* utilizado.

Se trata de un módulo que tiene como entradas tres ramas. Por una parte, los paquetes con origen el interfaz inalámbrico principal. Esos paquetes son tratados extrayendo las cabeceras de *Radiotap* y filtrando errores y duplicidades. Ocurre lo mismo con los que obtenemos del interfaz auxiliar. La ultima entrada recibe los paquetes del interfaz ethernet del plano de datos.

Como salidas tenemos dos ramas para cada interfaz inalámbrico, que preparan los paquetes para transmitirlos vía radio, un enlace interno de Odin y la salida hacia el plano de datos por Ethernet.

Anexo F:
Archivo de configuración “poolfile”

Se trata del archivo donde se especifican los datos que utiliza el controlador para inicializarse. Podemos ver como se asignan una serie de agentes al sistema (NODES) y se establece el nombre de la red inalámbrica.

Posteriormente se encuentran las aplicaciones que se ejecutan en el controlador, cada una de ellas con sus parámetros concretos.

Archivo poolfile:

```
# Pool-1
NAME pool-1
NODES 192.168.1.13 192.168.1.14 192.168.1.15
NETWORKS wi5-demo

####### Now applications and its parameters are defined ######

####### FlowDetectionManager params
####### DETECTION IpAddressOfDetector
# APPLICATION net.floodlightcontroller.odin.applications.FlowDetectionManager
# DETECTION 192.168.1.200

####### ShowStatistics
# APPLICATION net.floodlightcontroller.odin.applications.ShowStatistics

####### MobilityManager params
####### MOBILITY TimeToStart(sec) IdleClient(sec) Hysteresis(sec)
SignalThreshold(dBm) ScanningTime(sec) NumberOfTriggers TimerResetTriggers(sec)
# APPLICATION net.floodlightcontroller.odin.applications.MobilityManager
# MOBILITY 30 180 10 -45 1 5 1

####### ShowScannedStationsStatistics params - Optional filename to save statistics
####### INTERFERENCES TimeToStart(sec) ReportingPeriod(sec) ScanningInterval(sec)
AddedTime(sec) Filename
# APPLICATION
net.floodlightcontroller.odin.applications.ShowScannedStationsStatistics
# INTERFERENCES 30 30 5 1 ScannedStationsStatistics.txt

####### ShowMatrixOfDistancedBs params
####### MATRIX TimeToStart(sec) ReportingPeriod(sec) ScanningInterval(sec)
AddedTime(sec) Channel
# APPLICATION net.floodlightcontroller.odin.applications.ShowMatrixOfDistancedBs
# MATRIX 30 30 2 1 6

####### ChannelAssignment params
#######
#######Start
#######  |       Scan      Scan      Scan & Algorithm           Restart
#######  |         |         |    ...     |                          |
#######  |_________V_________V____________V__________________________V___
#######  <--------><--------><-------->...<--------------------------->    t
#######     Pause     Pause     Pause             IdleTime
#######
####### CHANNEL TimeToStart(sec) PauseBetweenScans(sec) ScanningInterval(sec)
AddedTime(sec) NumberOfScans IdleTime(sec) Channel Method
####### Method = 1 : Wi5, 2 : RANDOM, 3 : LCC  ;
# APPLICATION net.floodlightcontroller.odin.applications.ChannelAssignment
# CHANNEL 30 5 3 1 3 20 6 1
```

```
####### Interactive (prompt-based) application, which allows the user to introduce a
channel number for each AP throught the keyboard
# APPLICATION net.floodlightcontroller.odin.applications.ChannelPrompt

####### Continuous loop (10 sec) between the 11 channels of 2.4 GHz. The application
moves an AP between all the channels
# APPLICATION net.floodlightcontroller.odin.applications.ChannelLoop

####### Smart Ap Selection params
####### SMARTAPSELECTION TimeToStart(sec) ScanningInterval(msec) AddedTime(msec)
SignalThreshold(dBm) Hysteresis(sec) Alpha(0-1) Pause(sec)
# APPLICATION net.floodlightcontroller.odin.applications.SmartApSelection
# SMARTAPSELECTION 50 200 0 -45 5 0.2 0
```

Anexo G:
Códigos en GitHub

Ante el elevado número de líneas de código utilizadas en el proyecto, se opta por hacer un breve resumen de las contribuciones realizadas en el presente trabajo, agrupándolas en dos elementos:

1. Agente Odin[36]:

Desarrollado en C++, módulo parte de *Clik Modular Router* encargado de implementar los *Handlers* que ofrecen al controlador las diversas funcionalidades. Se han creado nuevos y modificado existentes, asi como realizado externalizacion de parametros para conseguir flexibilidad a la hora de hacer baterias de pruebas.

2. Controlador Odin[37]:

Desarrollado en Java, nucleo pricipal del sistema SDN. Encargado de ofrecer las funcionalidades a las aplicaciones y ordenar modificaciones en el comportamiento de los agentes. Se han creado y modificado funciones internas y aplicaciones.

Para la realización de pruebas de validación, se han lanzado diferentes versiones del código (*Releases*), coincidiendo con reuniones o el desarrollo de aplicaciones.

[36] Odin Agent: https://github.com/Wi5/odin-wi5-agent/graphs/contributors?from=2017-06-04&to=2017-11-16&type=c Accedido noviembre 2017
[37] Odin Controller: https://github.com/Wi5/odin-wi5-controller/graphs/contributors?from=2017-05-24&to=2017-11-16&type=c Accedido noviembre 2017

Anexo H:
Fotografías La Haya 2017

Figura H.1. Reunión de los socios del proyecto Wi-5.

Figura H.2. Intel Nuc y switch durante la demostración.

Figura H.3. Despliegue de puntos de acceso.