Universidad Zaragoza
1542

Trabajo Fin de Grado

Optimización de una Infraestructura de Red Basada en Smart APs

Optimization of Smart APs Based Network Infraestructure

Autor

Juan Luis de la Cruz Cuevas

Director

Luis Sequeira Villarreal

Ponente

José Ruiz Mas

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2016

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. Juan Luis de la Cruz Cuevas,

con nº de DNI 73019749-Q 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) Optimización de una infraestructura de red basada en Smart APs,

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

Zaragoza, a 19 de Julio de 2016

Fdo: Juan Luis de la Cruz

Agradecimientos

*Agradezco a mi director **Luis** su paciencia y tiempo dedicado a ayudarme y orientarme en este trabajo, y le agradezco todos los conocimientos que he podido aprender de él. Sin él, este trabajo no sería posible.*

Agradezco también a la gente dentro del proyecto Wi-5 y que en algún momento me han prestado su ayuda y consejo y especialmente a José Saldaña, a Pepe y a Julián, a los que deseo mucha suerte y éxito en el proyecto y en futuras investigaciones.

A mi familia el ayudarme a llegar hasta aquí y que siempre han estado apoyándome en todo momento.

A mis amigos el interés mostrado en el trabajo y avances y el aguantarme todos estos años.

A los miembros de la AATUZ por todos esos momentos de convivencia, de preparación de talleres y charlas, y sobre todo de liberación de estrés en el despacho y por el apoyo mutuo en las épocas de estudio.

A mis hermanos del Grupo Scout Entaban con los que tantas aventuras he pasado, me han ayudado a formarme como persona y siempre han mostrado su apoyo.

A mis compañeros de clase sin los cuales todos los días que he pasado en la Escuela no habrían sido lo mismo, y tantas horas de estudio mucho más largas.

Al equipo de Rugby de Ingeniería por esos momentos de liberación de estrés a lo largo de la carrera y finalmente haber conseguido entre todos llegar a lo mas alto.

A los integrantes del Íbero CR, que en el último año me han brindado muchas horas de trabajo, alegrías e ilusión.

A los alumnos que hacen plantillas de LaTeX, y a tantos desarrolladores de herramientas libres.

• • •

Optimización de una infraestructura de red basada en Smart APs

Resumen

En empresas, centros comerciales y campus universitarios, el despliegue de una red Wi-Fi con un conjunto de puntos de acceso coordinados se está convirtiendo en una solución común para proveer de acceso a Internet a los usuarios. Los servicios que utilizan estos usuarios son muy variados, y además se deben suministrar tanto a usuarios estáticos como móviles con la calidad suficiente. El Proyecto Wi-5 propone una plataforma que, usando conceptos de SDN y puntos de acceso inteligentes, logre hacer un uso más eficiente de los recursos disponibles.

Dentro de este contexto, los objetivos de este trabajo han consistido en optimizar aspectos de funcionamiento de la plataforma aparte de añadir funcionalidades de monitorización y control del sistema. Para ello se ha revisado el código y realizado mejoras sustanciales e implementaciones de dichas funcionalidades. Para validar las mejoras se han efectuado un conjunto de experimentos variando parámetros de la configuración.

De estos experimentos se ha llegado a una solución válida para cualquier tipo de servicio, incluso aquellos servicios de tiempo real con determinados requerimientos de calidad, aplicable a una gran variedad de dispositivos (portátiles, móviles, PC…), sin perder el enfoque en la escalabilidad del sistema. En concreto, se ha conseguido un mecanismo de traspaso de clientes entre los puntos de acceso, de forma transparente al usuario. Además, queda patente que el tipo de servicios coexistentes en un punto de acceso debe ser tenido en cuenta a la hora de ejecutar algoritmos de gestión de recursos en la red, con el fin de proveer una mejor calidad de servicio. Estos avances podrán ser considerados para el diseño de redes Wi-Fi inalámbricas con múltiples puntos de acceso.

Optimization of Smart APs Based Network Infraestructure

Abstract

In large offices, commercial centers and campuses, the deployment of a set of coordinated Wi-Fi APs is becoming a common solution to provide Internet access for users. Network services used are diverse, besides both mobile and static users should be provided with sufficient quality. The Wi-5 Project proposes a platform that using SDN concepts and smart APs, succeeds in making a more efficient use of available resources.

In this context, the objectives of this study have consisted in optimizing aspects of operation of the platform, appart from adding monitor and control functionalities to the system. For that the source code has been revised and substantial improvements and implementations of such functionalities has been made. To validate the improvements a set of experiments has been completed, varying configuration parameters.

From these experiments, a valid solution for any kind of service, including real-time services with strict requirements has been reached, applicable to a large amount of devices, without losing the scalability approach of the system. Specifically, a seamless handoff mechanism have been achieved. Moreover, it is set clear that the type of coexisting services should be taken into account by radio resource management algorithms, in order to provide a better quality. This progress may be considered for Wi-Fi WLANs with multiple APs design.

Índice

CAPÍTULO 1

Introducción

El presente documento tiene como objetivo recoger toda la información relacionada con la realización del Trabajo de Fin de Grado (TFG) titulado: Optimización de una infraestructura de red basada en Smart APs.

Este trabajo pertenece a la titulación de Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación de la Universidad de Zaragoza, y ha sido realizado por el alumno Juan Luis de la Cruz Cuevas y dirigido y supervisado por el doctor Luis Sequeira Villarreal, investigador adscrito al *Communications Networks and Information Technologies for e-Health and Quality of Experience* (CeNITEQ) y al Grupo de Tecnología de las Comunicaciones (GTC) dentro de este, actuando como ponente el profesor José Ruiz Mas.

En el primer capítulo se incluyen el contexto académico y el tecnológico de este trabajo, los objetivos y motivaciones que han llevado a su realización y la estructura en la que se organiza este documento.

Este TFG se desarrolla como proyecto final del Grado de Ingeniería en Tecnologías y Servicios de Telecomunicación, dentro del CeNITEQ, cuya investigación se centra en la e-salud y calidad de servicio en redes de comunicaciones. Dicho grupo pertenece al Departamento de Ingeniería Electrónica y Comunicaciones (DIEC) de la Universidad de Zaragoza. El TFG se encuentra dentro del proyecto *What to do With the Wi-Fi Wild West* (Wi-5), que combina investigación y desarrollo para proponer una arquitectura basada en un conjunto de soluciones inteligentes, coordinadas e integradas entre si, capaz de reducir la interferencia entre *Acces Point* (AP)s vecinos eficientemente y proveer a servicios nuevos y emergentes de una conectividad óptima.

1.1. Motivación y Objetivos

En los últimos años, el número de dispositivos conectados a la red ha aumentado considerablemente, formando ya parte de nuestra vida cotidiana en la sociedad occidental. Gran parte de estos dispositivos hacen uso del conjunto de estándares relativos a tecnologías inalámbricas *Institute of Electrical and Electronics Engineers* (IEEE) 802.11 [1], creados y mantenidos por el grupo de trabajo IEEE 802. Los usuarios realizan numerosas actividades que requieren de estos dispositivos conectados, entre otras: la navegación web, descarga y subida de archivos, uso de redes sociales, realización llamadas por videoconferencia,

streaming de vídeos y juegos multijugador en red. Para todo ello, habitualmente se hace uso de los estándares descritos en 802.11a/b/g/n/ac, popularmente conocido como Wi-Fi.

Originalmente el estándar 802.11 hacía uso del protocolo de acceso CSMA/CA, usaba infrarrojos y especificaba una velocidad teórica de 1 y 2 Mbps. Posteriormente pasó a usar las bandas del espectro electromagnético sin licencia *Industrial, Scientific and Medical* (ISM) designadas por la *International Telecommunication Union* (ITU), en concreto la de 2,45 GHz para el estándar 802.11b y la de 5,8 GHz para el estándar 802.11a. Actualmente, los estándares más extendidos son el 802.11n que opera en ambas bandas, y usa modulación *Ortogonal Frequency Division Multiplexing* (OFDM), además de usar técnicas *Multile Input Multiple Output* (MIMO), con lo que consigue una velocidad máxima teórica de 600 Mbps, y el 802.11ac, que opera en la banda de 5,8 GHz, e introduce modulaciones con más bits por símbolo a las ya usadas por el estándar 802.11n, y la técnica *Multi User MIMO* (MU-MIMO), obteniendo velocidades teóricas por encima del gigabit por segundo.

Debido a la limitación física del espectro para este uso, es necesaria una utilización eficiente de los recursos disponibles, desde la modulación usada por el estándar 802.11 en sus diferentes versiones hasta la disposición de los elementos de la red, y las aplicaciones y usos que se le da a esta. En entornos donde la saturación Wi-Fi aumenta como centros de negocios, centros comerciales, campus universitarios o en grandes ciudades, la interferencia entre los diferentes APs afecta negativamente a la experiencia del usuario, por lo que se necesita una infraestructura específica basada en redes corporativas para solventar cuestiones de capacidad y donde la movilidad es un requerimiento.

El proyecto Wi-5[1] se encuentra dentro del programa europeo H2020 y se centra en 5 casos de uso diferentes en los que estas tecnologías pueden ser aplicadas aunque no son los únicos, basándose en casos de uso presentados por el grupo de trabajo IEEE 802.11 y la *Wi-Fi Alliance* como tendencias de uso inalámbrico para el futuro. La infraestructura de partida debe ser refinada y adaptada a las necesidades de los usuarios teniendo en cuenta el estado del arte y la capacidad de los operadores.

Algunos de los casos de uso descritos en [2] son estaciones de trenes y aeropuertos, bloques de apartamentos y despliegue en la calle con pico-celdas.

- En las estaciones de tren y aeropuertos, un número determinado de APs dan servicio a los usuarios que se encuentren en el área. Los usuarios se encuentran en las salas de espera casi todo el tiempo pero existe la posibilidad de que el usuario use un servicio mientras esta andando, por lo que se debe considerar el *handoff* entre APs. Este caso sería fácilmente extrapolable a centros comerciales y estadios deportivos.

- En los bloques de apartamentos existen multitud de propietarios que despliegan su propia red Wi-Fi de forma independiente, lo cual crea una situación de alta saturación e interferencias, dando como resultado una mala calidad de servicio para todos los usuarios, a pesar del uso de repetidores. Como solución los propietarios son libres de hacer acuerdos vecinales sobre el uso del espectro centralizado basándose en el controlador Wi-5. En este caso los usuarios demandan una amplia gama de aplicaciones

[1] http://www.wi5.eu

con altos requerimientos desde juegos *online* a vídeo de alta calidad por *streaming*, así como necesidades que requieran de movilidad al desplazarse por el edificio.

– Los despliegues a pie de calle se harán adicionalmente a los desarrollados por los operadores relativos a las redes 3G/4G actualmente, donde deberán ir integrados. De esta forma se ahorrarán costes y mejorarán el servicio a los usuarios, como una red auxiliar a las redes de telefonía móvil coexistentes. Mientras que los usuarios "nómadas"[2] y peatones podrán usar esta red auxiliar, los que vayan en transporte a más velocidad usarán la red de telefonía. Las necesidades de la red son las ya expuestas anteriormente.

Como se puede observar en los casos de uso existe una problemática con varios puntos en común. En este contexto, con el proyecto Wi-5 se busca una solución donde exista la posibilidad de tener *smart* APs coordinados de forma centralizadaa, con características avanzadas como puede ser balance de carga, planificación de frecuencias o control de potencia, mientras que se mantiene un hardware de bajo coste y software abierto. Algunos trabajos proponen el uso de *Software Defined Network* (SDN), sin embargo no solucionan directamente todos los problemas que aparecen en una red inalámbrica.

Uno de los problemas que surgen es que los dispositivos "cliente" o STA utilizan sus propios algoritmos para seleccionar el AP al que se quieren conectar, normalmente basándose en la potencia recibida. Esto hace que un cliente nunca cambia de AP respecto al que se conectó inicialmente. Otro de los problemas es que el *handoff* normal entre APs es perceptible por el usuario, al durar varios cientos de milisegundos [3] y además cortarse la conexión, lo que constituye una limitación crítica cuando se usan aplicaciones de tiempo real.

Una manera de solventar estos problemas es la introducción del concepto *Light Virtual Access Point* (LVAP) propuesto en [4], que consiste en que un AP utiliza varios LVAPs para comunicarse con cada STA. De este modo cada STA verá solamente un único *Service Set Identifier* (SSID) a pesar de que existan varios APs y la decisión de *roaming* no la toma la STA, sino que la puede tomar un controlador que coordina los APs. Esta es la solución propuesta por [5] en la infraestructura Odin.

La infraestructura Odin tiene la capacidad de que los *handoffs* entre APs son rápidos, y añade la posibilidad de que los *handoffs* no se lleven a cabo solo con criterio de potencia de señal sino por otros parámetros. Sin embargo, carece de ciertas capacidades como el poder realizar *seamless handoffs* entre puntos de acceso en distintos canales, una generación de *beacons* optimizada para los *seamless handoffs* o una monitorización por parte de los puntos de acceso para que el controlador pueda manejar los cambios entre APs de forma efectiva y centralizada.

La motivación de este TFG es la de analizar los problemas y características de la red que conlleva el uso de una infraestructura basada en Odin e implementar soluciones a estos problemas, en concreto implementar la movilidad *seamless* entre canales distintos, la optimización de parámetros de red y el aumento de medidas para la monitorización de los clientes desde el controlador.

[2] Los usuarios nómadas son los que se mueven cada cierta frecuencia.

Este TFG tiene como objetivo principal hacer un estudio de la situación actual y mejorar la eficiencia de la red, para aumentar la escalabilidad en un futuro. También, debido a la alta cantidad de usuarios en una red corporativa, es necesaria una monitorización de los usuarios de la red, para así realizar la gestión de la red eficaz. Como objetivos concretos se han propuesto los siguientes:

- Realizar un análisis exhaustivo de la infraestructura de red Odin, a nivel físico y a nivel de aplicación.
- Mejorar la configuración existente de la infraestructura Odin para obtener un rendimiento adecuado para aplicaciones que requieran buenas condiciones de retardo, *jitter* y pérdidas.
- Aumentar la cantidad de información disponible para que los agentes puedan enviar más información al controlador y de esta forma tenga capacidad de monitorización de los elementos de la red.

1.2. Contribuciones científicas

Las siguientes publicaciones científicas han derivado del trabajo realizado durante el proyecto. La metodología y los resultados presentados están basados en todas ellas. Todas las publicaciones han pasado por una revisión por pares.

Publicaciones en revistas internacionales (Indexadas en *Journal Citation Report*)

Luis Sequeira, Juan Luis de la Cruz, José Ruiz-Mas, José Saldana, Julián Fernandez-Navajas, José Almodovar. **"Building a SDN Enterprise WLAN Based On Virtual APs"**. *IEEE Communications Letters*. (*Minor Revision*, pendiente de publicación)

Publicaciones en congresos internacionales

José Saldana, Juan Luis de la Cruz, Luis Sequeira, Julián Fernandez-Navajas, José Ruiz-Mas. **"Can a Wi-Fi WLAN Support a First Person Shooter?"**, *NetGames 2015*, Zagreb, Croatia, December 3-4, 2015. ISBN 978-1-5090-0067-8.

Otras diseminaciones

Invitación a presentar una sección de demostración:

Luis Sequeira, Juan Luis de la Cruz, José Almodovar, José Saldana, Julián Fernandez-Navajas, José Ruiz-Mas. **"Demo: Can a Wi-Fi WLAN Support a Real-Time Service During Handoff?"**, *Ultra-fast Broadband Seminar 2016*, The Hague, Netherlands, June 27-30, 2016.

1.3. Estructura del documento

La memoria se ha organizado de la siguiente manera:

- En el Capítulo 1 se ha presentado una descripción del trabajo, motivación del mismo y objetivos, además de contribuciones científicas adicionales derivadas a las que se han realizado aportaciones.
- En el Capítulo 2 se ha presentado la plataforma Wi-5 en el que se desarrolla el trabajo así como las tecnologías relacionadas y las herramientas utilizadas.
- En el Capítulo 3 se detallan las pruebas a las que ha sido sometida la arquitectura, la descripción de los escenarios usados y una presentación de los resultados obtenidos.
- Por último, en el Capítulo 4 se han presentado las conclusiones finales del trabajo y posibles líneas futuras.

También se han incluido los siguientes anexos.

- Anexo A. Preparaciones previas a las pruebas
- Anexo B. Diagrama de secuencia del *handoff*
- Anexo C. *Script* csv_parser.py
- Anexo D. *Scripts* de Matlab
- Anexo E. Fotografías del despliegue

CAPÍTULO 2

Estado del Arte

2.1. Proyecto Wi-5 H2020

Durante los últimos años se ha dado un crecimiento sin precedentes del uso de las tecnologías inalámbricas, en especial *smartphones* y tabletas debido a su funcionalidad, su facilidad de uso y un precio asequible. La mayoría de estos dispositivos usa puntos de acceso Wi-Fi cuando es posible, además de las tecnologías 3G/4G, para conectarse a Internet.

En este contexto, el proyecto Wi-5 propone una arquitectura basada en un conjunto de soluciones inteligentes coordinadas e integradas entre si capaces de reducir la interferencia eficientemente entre los APs cercanos entre sí y proveer una conectividad optimizada para servicios nuevos. Se integrarán dentro del equipamiento Wi-Fi en diferentes capas de la pila de protocolos para soportar *seamless handoff* y de esta manera mejorar la experiencia del usuario, desarrollar nuevos modelos de negocio para optimizar el espectro disponible, e integrar nuevas funcionalidades inteligentes en los APs para solventar la congestión de los recursos radio y el actual uso ineficiente de estos. El proyecto tiene una duración para desarrollar y cumplir sus objetivos desde enero de 2015 hasta diciembre de 2017. El repositorio del proyecto se encuentra en `https://github.com/Wi5` y la página del proyecto es `http://www.wi5.eu`, donde se puede encontrar una información mas detallada acerca de este.

Para lograr esto, el proyecto debe afrontar varios desafíos. Por un lado, el diseño de las funcionalidades para la coordinación de los APs y optimización del uso de los recursos disponibles. También se buscan métodos de agrupamiento de paquetes, para así reducir el *overhead* y conseguir una mejor eficiencia. Se está trabajando en una plataforma de interacción vertical entre operadores, para que todos los proveedores de servicio, que tienen sus propios AP, puedan trabajar de forma conjunta y coordinada además de diseñar unos mecanismos de *handoff* entre las redes Wi-Fi y otras redes móviles o fijas para proveer una mejor calidad de red a los clientes. Todo esto estará integrado finalmente en una misma arquitectura de red, como se muestra en la figura 2.1 [2].

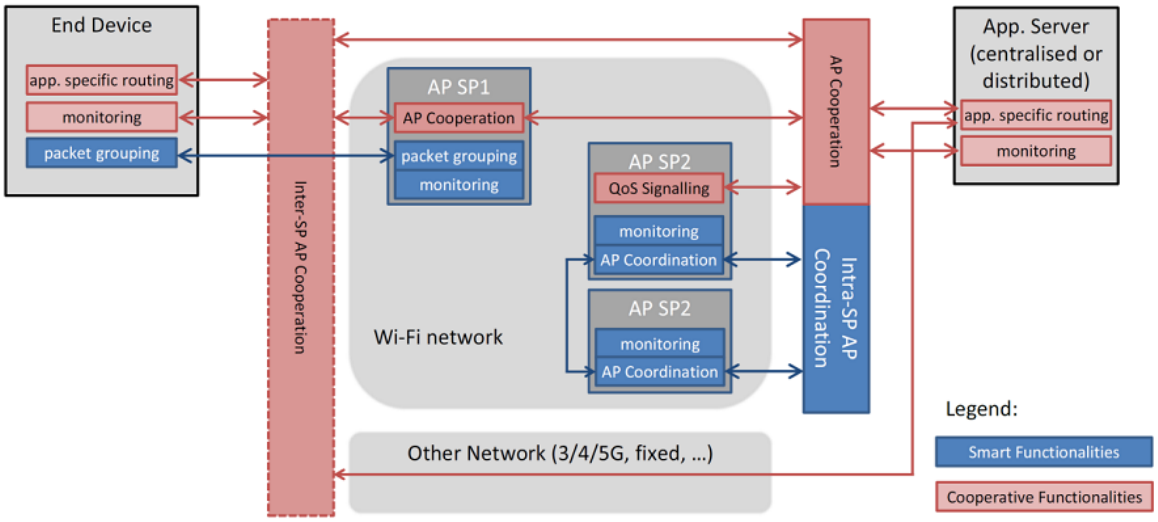


Figura 2.1: Borrador de la arquitectura Wi5 en alto nivel

2.2. Redes definidas por software

Hasta la existencia de las SDN [6] [7], la arquitectura de red tradicional se ha basado en dos planos: control y datos. En el plano de control se toman las medidas adecuadas para poder realizar las tablas de *forwarding*, que son usadas por los elementos de la red para redirigir los paquetes. El plano de datos, es el que precisamente consulta estas tablas para saber hacia donde tiene que dirigirse cada paquete que entra en el nodo de red. Cada nodo de red tiene su propia tabla de *forwarding* y gracias a protocolos como podrían ser *Router Information Protocol* (RIP) u *Open Shortest Path First* (OSPF) mantiene contacto con otros nodos.

Sin embargo, en SDN los elementos de la red se comportan como un solo elemento, dado que se separa el plano de control de los dispositivos y se introduce un controlador distribuido, como se puede observar en la figura 2.2. El controlador obtiene información de todos los nodos y a su vez puede hacer rápidamente los cambios necesarios desde el punto de vista de la infraestructura completa. Los aspectos clave de las redes definidas por software son la separación funcional, la virtualización de red y la automatización a través de la programación.

En el caso de una red inalámbrica como es el caso, el uso de SDN permite una utilización eficiente de los recursos inalámbricos, la implementación de mecanismos de *handoff* y operaciones de balanceo de carga entre puntos de acceso. Sin embargo, la implementación cableada de SDN no tiene en cuenta los problemas de las comunicaciones inalámbricas como las pérdidas de paquetes, el problema del terminal expuesto o el problema del terminal oculto. Para poder adaptar al medio inalámbrico las SDN deben satisfacer nuevas condiciones de diseño como movilidad, calidad de servicio en los enlaces y la localización de los usuarios.

2.2.1. OpenFlow

Como se mencionaba en la sección anterior, una infraestructura basada en SDN necesita de un controlador, y este a su vez necesita de una comunicación persistente con los elementos de la red. Para esto se usa el protocolo propuesto en [7], Openflow, que es uno de los

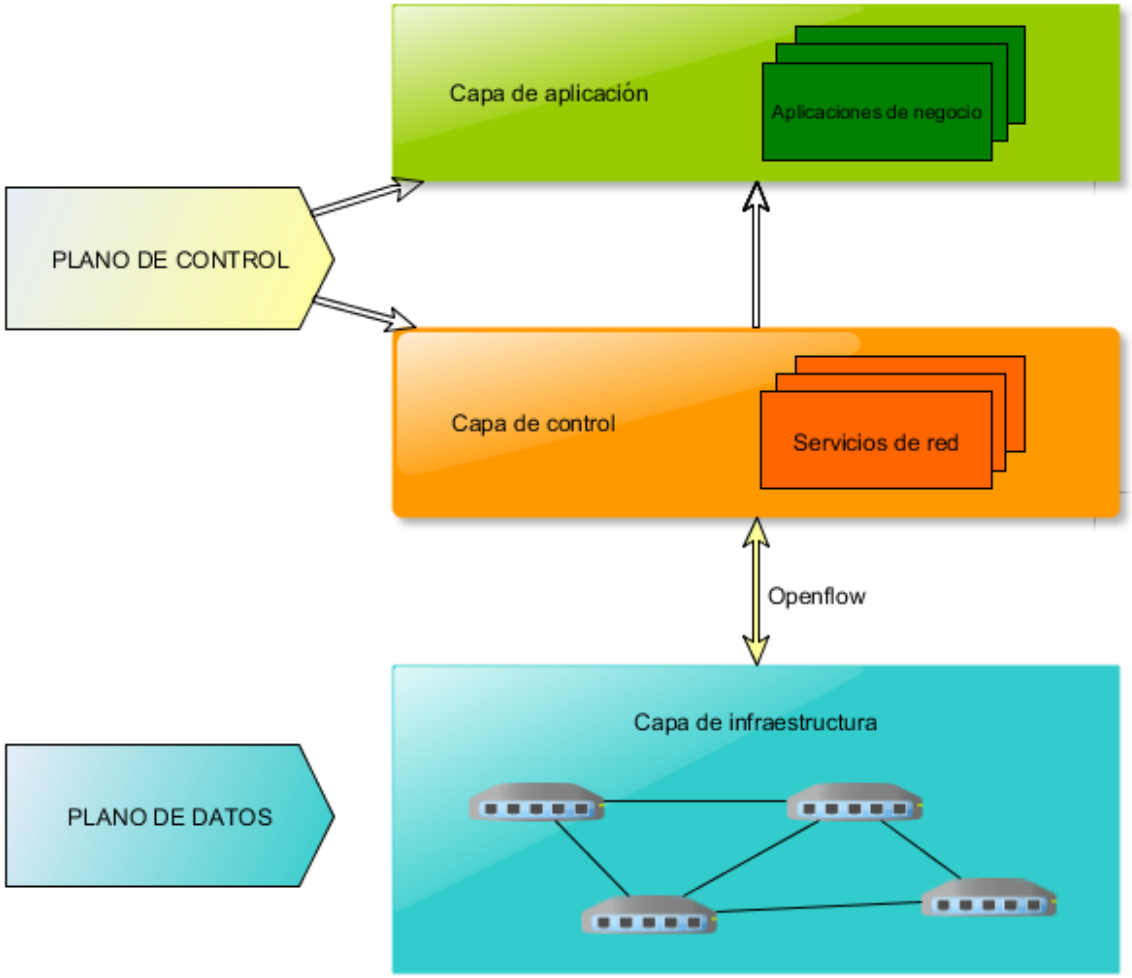


Figura 2.2: Concepto de SDN

protocolos usados para la comunicación entre agentes y controlador. Originalmente fue un proyecto desarrollado por la universidad de Stanford, sin embargo actualmente la *Open Networking Foundation*[1] es la entidad que lleva las riendas sobre el estándar, y aglomera tanto a muchas de las operadoras que dan servicio de internet a nivel mundial como a grandes empresas relacionadas con las infraestructuras de redes.

Existe un gran número de controladores OpenFlow[2]. En este caso se usa el controlador desarrollado por el proyecto Floodlight[3] [8], que tiene una licencia Apache y una comunidad abierta, por lo que es posible descargárselo y ver el código del controlador, y además existe bastante cantidad de información sobre el y ejemplos de implementación. El controlador Floodlight esta programado en Java, y por tanto las aplicaciones de red que lleva usan también el mismo lenguaje. Esto le da versatilidad a la hora de elegir una máquina para la implementación del controlador, siempre y cuando la máquina sea compatible con la *Java Virtual Machine* (JVM).

[1] https://www.opennetworking.org/index.php
[2] http://yuba.stanford.edu/~casado/of-sw.html
[3] http://www.projectfloodlight.org/floodlight/

Los agentes llevan integrado en el *kernel* el software *Open Virtual Switch* (OVS) [4], que actúa como un switch virtual multinivel. Está diseñado para poder automatizar en gran medida la gestión de una red, mediante la programación de normas, y es compatible con muchos protocolos e interfaces de administración, como pueden ser NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP y 802.1ag. Es compatible con el protocolo OpenFlow, por lo que permite que los agentes puedan comunicarse con el controlador, y direccionar los flujos que pasan por sus puertos de forma centralizada, a través de reglas que permiten al agente saber hacia donde dirigir cada flujo.

2.3. Click Modular Router

Una de las limitaciones de OpenFlow es la necesidad de dispositivos físicos de red compatibles con el protocolo. Existen numerosos dispositivos comerciales en el mercado, sin embargo, esto lo haría difícilmente implementable para sistemas de bajo coste. En el escenario que se plantea en el TFG se usan puntos de acceso comerciales que son compatibles con OpenWRT. En [9] [10] se propone el uso de Click[5] para transformar una máquina con un sistema operativo basado en Linux en un *switch* completamente programable y definido por software. En el caso de los APs usados, es la herramienta integrada dentro de la arquitectura Odin para hacer compatibles los agentes con OpenFlow.

En Click, cada módulo se define mediante código escrito en C o C++, lo que permite la modificación de los paramétros del *switch* y la adición de nuevos módulos a este diseñados de forma externa, en caso de que sea necesario. En nuestro caso, mediante el uso de click, se obtiene un *switch* con una interfaz inalámbrica y otra cableada a partir de un punto de acceso comercial, cuya estructura y comportamiento interno están definidas por software, lo que permite su optimización sin necesidad de cambiar de hardware y su adaptación a las necesidades de la red.

2.4. OpenWRT

OpenWRT[6] [11] es un sistema operativo basado en Linux, usado en los agentes de Odin.Está orientado a sistemas embebidos y es altamente personalizable, por lo que se ajusta muy bien a las necesidades del proyecto Wi-5. Sustituye a lo que sería el *firmware* de un router.

Al ser un proyecto abierto, está en constante actualización. Originalmente se diseño a partir del firmware diseñado por Linksys para el router WRT54G en 2004, y a partir de aquí el proyecto fue creciendo y haciéndose compatible con multitud de dispositivos. Cuando comenzó el proyecto Wi-5, la versión que se usaba en los APs era la *Barrier Breaker* 14.07, que salió en octubre de 2014. Esto permite modificar el software los puntos de acceso para probar diferentes configuraciones, además de la integración de Click, OVS y Odin. También habilita el uso de los puertos USB del dispositivo, a los que se pueden conectar otros periféricos compatibles, como por ejemplo impresoras, webcams o dispositivos de almacenamiento.

[4] http://openvswitch.org/
[5] http://www.read.cs.ucla.edu/click/
[6] https://openwrt.org/

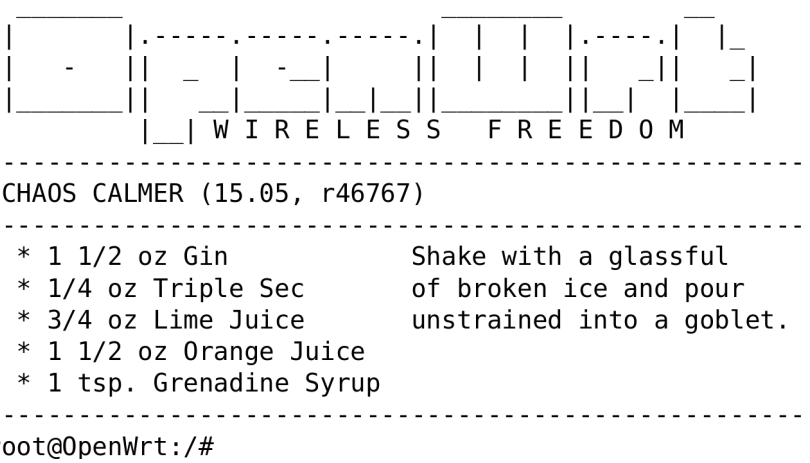


Figura 2.3: Pantalla de login de OpenWRT (v. Chaos Calmer 15.05)

La última versión es la *Chaos Calmer* 15.05.1, que salió a mediados de marzo de 2016, es incompatible con los dispositivos usados, debido a una limitación de memoria y espacio disponible. En las últimas pruebas del proyecto, se ha podido instalar en los últimos dispositivos que se han adquirido recientemente (TP-Link 1750 AC), sin embargo el estudio en estos dispositivos no ha sido posible por limitación de tiempo del TFG.

En la página web del proyecto OpenWRT existe una *wiki* con mucha información, donde se puede encontrar una extensa lista de dispositivos compatibles, guías para la instalación del *firmware*, iniciar por primera vez el sistema después de haber sido *flasheado*, y una configuración básica con la que poder usar el punto de acceso. Esto es posible gracias a una activa comunidad que hay detrás del proyecto y contribuye con guías e información útil.

2.5. Odin

Para permitir la comunicacíon entre el controlador y los diferentes agentes de la arquitectura del proyecto Wi-5, se usa Odin, descrito con detalle en [5] [12] [13], que es una implementación existente de una SDN basado en el controlador Floodlight y el software Click.

En Odin, se hace uso de una abstracción llamada LVAP, que consiste en la creación de un punto de acceso virtual para cada *Station* (STA). Este LVAP es a todos los efectos un AP, desde el punto de vista del cliente. Sin embargo, si el cliente se moviera de AP físico, es posible mover también el LVAP, por lo que desde el punto de vista de la STA la conexión se encontraría en el mismo AP. Esto tiene el potencial para permitir una movilidad transparente al cliente, sin embargo, es algo que no está completamente implementado en Odin, ya que Odin no tiene en cuenta la posibilidad de que los APs se encuentren en diferentes canales, limitando así la escalabilidad de la red y generando un entorno con mayor interferencia. Además en el caso de redes de alta densidad de usuarios, la saturación del entorno con *beacons* puede ser un problema debido al considerable número de clientes que se encuentran el mismo AP.

Por un lado, en el controlador se implementa un paquete de software llamado *Odin Master*, en el cual se centraliza el control de los dispositivos que componen la red. Las aplicaciones de red se ejecutan en una capa por encima del *Odin Master*, en las que se pueden definir diversas estrategias por parte de la administración de la red. En el TFG se realiza una

aplicación simplificada con un uso meramente académico, para la medida de parámetros y la optimización del cambio de canal.

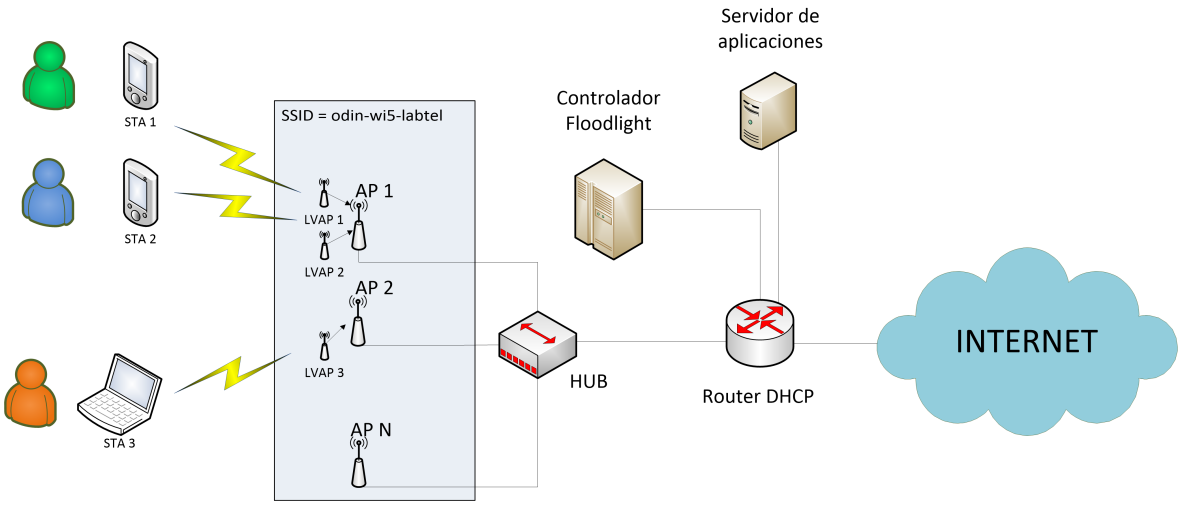


Figura 2.4: Diagrama de red Odin

Por otro lado, los agentes se implementan cómo un módulo de Click denominados *Odin Agent*, que se en. Estos son puntos de acceso inalámbrico que como ya se ha explicado anteriormente usan OpenWRT y Click para la adaptación a las necesidades de la red. La integración dentro de Click de dichos módulos se realiza mediante los ficheros: *Odinagent.cc* y *Odinagent.hh*. De esta manera, se hace compatible un *router* comercial con OpenFlow y se puede diseñar una red corporativa inalámbrica sea asequible y al alcance de mas gente. Una versión simplificada de la arquitectura Odin se puede observar en la figura 2.4, en ella se muestran los elementos básicos de la arquitectura que son el controlador y los agentes. Cada STA esta conectada a un LVAP propio, independientemente de a que punto de acceso físico estuvieran asociados. Si una de las STAs se moviera a otro AP, el LVAP asociado se movería con ella.

2.6. QoS en servicios de tiempo real

En el estudio realizado en el TFG se usan diferentes medidas relativas a la calidad de servicio en redes de comunicaciones. En [14] se apunta a que las medidas que más afectan a las transmisiónes en este ámbito son la pérdida de paquetes, el retardo y *jitter*. La mayoría de servicios en tiempo real como *Voice over IP* (VoIP), videoconferencias y juegos en red multijugador, son especialmente sensibles a estas medidas [15]. Sin embargo se han desarrollado a lo largo de los años mecanismos para paliar el efecto de las características del canal en la calidad de las comunicaciones, como pueden ser en el caso de VoIP técnicas de *Forward Error Correction (FEC)* o de entrelazado, para poder corregir errores añadiendo redundancia y mitigando el efecto de una ráfaga de errores en la comunicación de voz. Estas técnicas se han generalizado y se usan ahora en todo tipo de aplicaciones [16].

2.6.1. Pérdidas de paquetes

Las aplicaciones de VoIP, de *streaming* y la muchos de los juegos online, que tienen en común la necesidad de buena calidad de conexión, se basan en protocolos cuya capa inferior es *User Datagram Protocol* (UDP), por lo que son susceptibles a pérdidas de paquetes. Estas pérdidas podrían eliminarse enviando los paquetes a través de *Transmission Control Protocol* (TCP) [17], sin embargo el retardo aumentaría de forma considerable, y esto no es aceptable para un servicio de tiempo real. Además, el uso de TCP reduce de forma efectiva el ancho de banda efectivo en la comunicación, debido al control de congestión que tienen las diferentes implementaciones de TCP.

Aun así, la tolerancia a pérdidas depende de la aplicación. Según el estudio presentado en [18], en VoIP dependiendo del códec usado existe una tolerancia a errores de hasta el 5 % en algunos casos. En redes inalámbricas, se da una pérdida intrínseca a causa de las características de la comunicación, que pueden ser debidas a interferencia o ruido. No obstante, existe un mecanismo de retransmisión en Wi-Fi, por lo que estas pérdidas no son tan grandes como podrían serlo.

2.6.2. Retardo

El retardo [19] es la acumulación de los tiempos de transmisión, procesado y encolado en los *routers*; el tiempo de propagación en el canal de enlace y el tiempo de procesado final. Para aplicaciones conversacionales de tiempo real, como VoIP, un retardo aceptable debe ser menor que 450 ms [14], siendo imperceptible por el usuario un retardo menor que 150 ms. Los retardos por encima de 400 ms son altamente perjudiciales para una conversación fluida. Algunas aplicaciones descartan los paquetes cuyo retardo está por encima de un umbral, por lo que contarían como una pérdida a efectos de procesado.

En los juegos multijugador, depende del género del juego y del usuario la percepción del retardo, ya que habrá juegos y jugadores que requieran unos retardos menores mientras que en otros no será un parámetro tan crítico, como se apunta en [20]. En definitiva, un jugador de un *First Person Shooter* (FPS) percibirá el retardo de una forma mucho más determinante que un jugador de un juego de estrategia, así como también un jugador experto percibirá los altos niveles de retardo de una forma mucho mas acusada que un jugador casual.

2.6.3. Jitter

El *jitter* es la variación del retardo que experimenta un paquete dentro de un flujo. Se puede definir como la desviación estándar del retardo. Debido a esto, una emisión periódica de paquetes como puede ser un flujo VoIP deja de ser periódica, variando los tiempos de llegada de los paquetes e incluso causando la llegada de paquetes fuera de orden y afectando al rendimiento de la comunicación. El *jitter* es también un gran problema en los juegos multijugador, ya que afecta negativamene al rendimiento, y los paquetes son enviados con una frecuencia superior que una aplicación de voz. En [15] puede verse como el *jitter* afecta al rendimiento de los jugadores en gran medida.

2.6.4. Mean Opinion Score

Una forma de medir la calidad de un servicio es mediante el *Mean Opinion Score* (MOS). Es una puntuación de 1 a 5 que el usuario da a un servicio, y permite el análisis de este de forma sistemática, usando un modelo resultante de la regresión de las puntuaciones tomadas. Se considera que cuando el valor está alrededor de 2, o es menor, los jugadores se cambian de servidor por la mala experiencia de juego. En este TFG, se usa el modelo para *Quake IV*[7] propuesto en [21], que es el siguiente:

$$X = 0{,}104 * retardo_{medio} + jitter_{medio}$$

$$MOS_{model} = -0{,}00000587X^3 + 0{,}00139X^2 – 0{,}114X + 4{,}37$$

De este modo, se puede realizar una predicción del rendimiento de un sistema para unas condiciones de retardo y *jitter* medidas. Cabe destacar que el nivel de pérdidas no afecta a la calidad en este caso, debido a que la tolerancia de pérdidas es alrededor del 35 % [21].

2.7. Otras herramientas utilizadas

Para el desarrollo de este TFG se han usado las siguientes herramientas, la mayoría de estas son software libre y se encuentran disponibles para su descarga y uso:

Linux Para el TFG se han usado diferentes distribuciones basadas en Linux. Linux es el *kernel* de numerosos sistemas operativos, siguiendo la filosofía de código abierto y siendo desarrollado actualmente por *The Linux Kernel Organization*. Originalmente el objetivo de sus desarrolladores fue crear un clon del sistema operativo UNIX, que sigue el estándar *Portable Operating System Interface* (POSIX). Debian[8] es la familia de distribuciones que se han usado para el desarrollo y puesta en marcha de parte de la infraestructura, la justificación de su uso es el ser más estable que la mayoría de distribuciones Linux, y el tener un archivo de distribuciones mas antiguas para asuntos de compatibilidad. En la máquina usada para capturar en la red, se ha usado la distribución Kali[9] Linux, que aunque es mas inestable, tiene numerosas herramientas de *pentesting* y seguridad, y también es software libre.

tcpdump Tcpdump[10] es un software de línea de comandos para la captura de tráfico de red y posterior análisis, que usa la librería *libpcap* escrita en C/C++. En este TFG se ha usado para capturar tráfico Ethernet y también Wi-Fi.

wireshark/tshark Wireshark[11] es un software de código abierto para la captura y análisis de tráfico de red, que tiene interfaz gráfica de usuario y herramientas de análisis muy potentes. Usa *libpcap* en sistemas Linux y *winPcap* en sistemas Windows, por lo que es una apropiado para capturas realizadas con tcpdump o con software basado en libpcap.

[7] http://store.steampowered.com/app/2210/
[8] https://www.debian.org/index.es.html
[9] https://www.kali.org/
[10] http://www.tcpdump.org/
[11] https://www.wireshark.org/

kismet Kismet[12] es un software de captura y detección de intrusos para redes inálambricas 802.11. Se ha usado para captura en la interfaz inalámbrica y análisis de diferentes canales al mismo tiempo. Gracias a este análisis, se pudo elegir los canales con mejores condiciones de congestión para las pruebas, debido a que había cierta variabilidad de estos.

MATLAB® MATLAB®[13] es un software propietario de cálculo matemático desarrollado por *MathWorks*. Se ha usado para el análisis de los datos de la captura para el cálculo de estadísticas, y se ha elegido este programa por ser familiar al haberlo usado durante el Grado.

D-ITG *Distributed Internet Traffic Generator* (D-ITG)[14] [22] es un software de generación de tráfico que réplica de forma precisa varias aplicaciones de red, como Quake3[15] o una aplicación de VoIP entre otros, y además permite un análisis posterior de medidas de rendimiento como retardo o *jitter*. Se ha usado en el TFG para generar el tráfico similar al que genera un cliente de Quake3, del tipo UDP, y sirve como ejemplo de aplicación multimedia de tiempo real.

iPerf iPerf [16] es una herramienta para la medida de capacidades de red, al contrario que D-ITG el tráfico generado es constante, aunque da opción a usar un fichero como origen del tráfico. Se ha usado para hacer análisis más rápidos y con tráfico constante a la infraestructura de red, pero no de forma tan extensiva como D-ITG.

VMware vSphere VMware vSphere[17] es un hipervisor que permite la virtualización de máquinas virtuales. En este TFG se ha usado el cliente, para acceder a varias máquinas virtuales, dado que varios componentes de la arquitectura están virtualizados como se detallará más adelante en la apartado 3.3.1.

C/C++ C y C++ son dos lenguajes de programación de propósito general, uno en su versión imperativa y otro en su versión orientada a objetos, siendo ambos lenguajes muy comunes en numerosas aplicaciones. Su uso en el TFG es dentro de los agentes de Odin, ya que sus capas de software están escritos ya sea en C o en C++. Ambos lenguajes permiten una programación a bajo nivel muy potente.

Java Java es un lenguaje de programación orientada a objetos, que usa JVM para dar un soporte multiplataforma sin tener problemas de compatibilidad con el sistema operativo. El controlador de Odin esta implementado en Java así como sus aplicaciones de red.

Python Python es un lenguaje de alto nivel multiparadigma, cuya sintaxis permite hacer tareas complejas en pocas líneas de código respecto a otros lenguajes. Además, es

[12] https://www.kismetwireless.net/
[13] http://es.mathworks.com/products/matlab/
[14] http://traffic.comics.unina.it/software/ITG/
[15] Quake3 es un popular juego multijugador
[16] https://iperf.fr/
[17] http://www.vmware.com/es/products/vsphere.html

posible usarlo sin tener que compilar, dado que es un lenguaje interpretado. Se ha usado para realizar scripts para el tratado previo de datos, por así decirlo, un módulo entre los datos exportados de wireshark y los importados en MATLAB.

Bash Bash es un intérprete de comandos, además de un lenguaje para este intérprete, que se ha usado para realizar la automatización de ciertas tareas a la hora de realizar las pruebas.

TEXMaker/LATEX LATEX[18] es un sistema de composición de textos, orientado a la generación de documentos de alta calidad tipográfica y ampliamente usado. TEXMaker[19] es un entorno de trabajo para hacer mas sencillo el uso de LATEX. Ambos son software libre. Además, se han usado multitud de paquetes y herramientas, que han permitido el diseño y estructuración de esta memoria.

GitHub GitHub[20] es una plataforma de desarrollo colaborativo para alojar proyectos de software que usa el control de versiones Git. En esta plataforma está alojado el software del proyecto Wi-5 y permite crear ramas para modificar el código, y luego hacer peticiones para introducirlo en la rama principal. También permite abrir hilos de problemas, para solucionarlos de forma colaborativa.

[18] https://www.latex-project.org/
[19] http://www.xm1math.net/texmaker/
[20] https://github.com/

CAPÍTULO 3

Análisis y optimización de la plataforma Odin-Wi5

En este capítulo se detalla el proceso de las pruebas así como su posterior análisis y el procedimiento tras la obtención de resultados. Se ha decidido no detallar la totalidad de las pruebas, debido a que muchas de las pruebas realizadas son inválidas o se han tenido que repetir por la imposibilidad de un análisis conciso debido a que los datos obtenidos habían sido erróneos. Por ello, se ha resumido el proceso en dos pruebas más notables, y se han obviado otras pruebas menos determinantes aunque debe constar que sí se han aplicado aspectos derivados de estas.

3.1. Análisis preliminar de la red Odin

3.1.1. Consideraciones iniciales

Lo primero que se ha hecho es simplificar el escenario lo máximo posible, como se muestra en la figura 3.1, en la que solo coexisten en el escenario los siguientes elementos: una STA, dos APs, el controlador Floodlight que a su vez es el servidor *Dynamic Host Configuration Protocol* (DHCP) aunque en la imagen se muestre por separado, un hub y una máquina de captura externa. El controlador tiene configurado realizar *handoffs* cada poco tiempo, de forma que se pueden ver qué paquetes intervienen en el proceso. La máquina capturadora también captura en el interfaz radio en modo monitor, para así poder ver como se comporta la plataforma en esta interfaz. Este análisis previo se ha hecho con el motivo de habituarse al entorno de Odin,y observar sus características especiales.

Después de haber realizado una serie de pruebas iniciales y haber analizado los datos obtenidos, es conveniente señalar los aspectos más relevantes que se han observado:

- Los paquetes de administración de la red inalámbrica como los *beacon frames*, que normalmente son *broadcast* en este caso se tratan de paquetes *unicast*. Esto es debido a que cada LVAP se comporta de forma virtualmente independiente, a pesar de estar en el mismo AP. Por otro lado, esta característica permite hacer el *handoff* entre APs.

- Existe tráfico de control en el interfaz ethernet entre el AP y el controlador, debido a que el controlador debe proveer la información de destino para los paquetes que el agente no sabe dónde deben ir. Esto se realiza con los paquetes *packet IN* y *packet OUT* de OpenFlow. Este tráfico añadido es un problema si no se controla bien, dado que la red

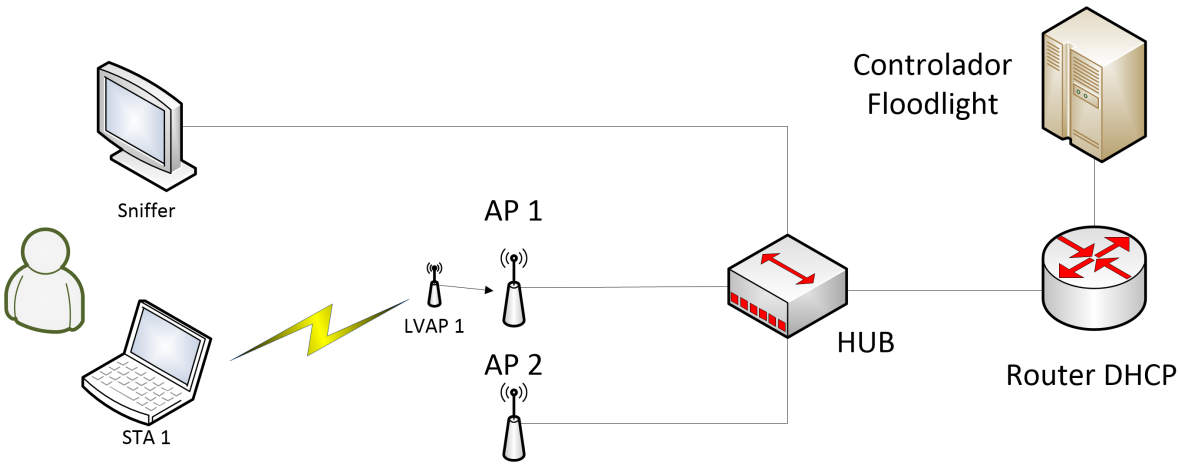


Figura 3.1: Diagrama básico de Odin

si no esta bien configurado tanto el agente como el controlador puede llegar a saturarse, y esto puede conllevar a la desconexión de los agentes.

– La red Odin puede funcionar tanto con SSID público como con SSID oculto, para las primeras pruebas el SSID era a todos los efectos público, sin embargo posteriormente se configuró para que no fueran respondidos los paquetes *probe request* que iban en *broadcast*, sino solo los dirigidos a la MAC del LVAP o los que tuvieran en el campo de SSID especificado el nombre de la red Odin. De esta forma se evita además una posible contaminación de las pruebas por parte de gente que se encuentre por la zona donde se efectúa la prueba y tenga activo algún dispositivo Wi-Fi.

– En la versión 14.07 de OpenWRT, existe una limitación con el tráfico de bajada, dado que solo se pueden alcanzar velocidades de hasta 6 Mbps, esto es debido a una limitación del *driver* y en la versión actual del sistema si es posible aumentar la tasa hasta 54 Mbps. Esto es un inconveniente a la hora de realizar pruebas de tráfico de bajada, por lo que para medir la capacidad del sistema se harán pruebas con tráfico de subida.

– Aun no se ha implementado ninguna capa de seguridad para la red Odin, por lo que los datos se encuentran sin encriptación ni existe un control de acceso a la red.

3.1.2. Automatización de procesos

Para facilitar las tareas de configuración, puesta en marcha y análisis de los ensayos realizados, ha sido conveniente el uso de *scripts* para la simplificación y automatización de dichas tareas.

En el Anexo capítulo C se presenta el *script* usado para dar formato adecuado a los datos exportados desde Wireshark. Para ello se hace uso de *Regular Expressions*.

En este Anexo capítulo D se presentan los *scripts* usados para procesar la información obtenida capturando. El fichero *main.m* se encarga de automatizar el proceso para las pruebas con los diferentes parámetros, los cuales se encuentran guardados de forma estructurada en carpetas separadas. Por otro lado, *ScriptQuake_v4.m* es el fichero que realiza la mayoría de cálculos necesarios. *ScriptQuake_v4.m* usa alguna función personalizada que

realiza operaciones sencillas y muy utilizadas en el proceso, y se puede ejecutar de forma independiente a *main.m*.

Además se han creado otros scripts para ejecutar tareas sencillas. A continuación se presentan algunos ejemplos. El Código 3.1 se usa para capturar en dos canales inalámbricos y guardar con un nombre indexado la captura. El Código 3.2 se usa para iniciar el servidor DHCP y configurar el *routing*. El **??** se usa para simplificar la ejecución de D-ITG, pasando de parámetros la duración y el tipo de tráfico.

Código 3.1: captura_wifi.sh

```
#!/bin/bash

# Start capturing in both interfaces.
# Monitor mode should be enabled.
tcpdump -i wlan0 ...
        -w quake_cap_$(date +-%j-%H%M%S)_$1_if1.pcapng &
tcpdump -i wlan1 ...
        -w quake_cap_$(date +-%j-%H%M%S)_$1_if2.pcapng &
```

Código 3.2: initiate_for_odin.sh

```
#!/bin/bash

# Activamos el forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# Configuramos para que haga NAT
iptables -t nat -A POSTROUTING -o wlp0s29f7u3 -j MASQUERADE

# Iniciamos el servidor dhcp
/etc/init.d/isc-dhcp-server start
```

Código 3.3: launch_ITGSend.sh

```
#!/bin/bash

# Launch traffic to server 192.168.1.131
D-ITG-2.8.1-r1023/bin/ITGSend -a 192.168.1.131 ...
                                        -i eth0 -t $1 $2
```

3.1.3. Generación de *beacons*

El *beacon interval* determina el período de emisión de *beacons* por parte de un agente a un cliente, en las redes Wi-Fi domésticas suele tener un valor de entre 50 y 150 milisegundos. Sin embargo, debido a la tipología de la infraestructura usada, esto es inadmisible, ya que el número de *beacons* aumenta de forma proporcional al número de usuarios. Además, por la implementación de la generación de *beacons* no se realizan de forma sincronizada, sino que cada uno se genera con una temporalización propia, por lo que podría resultar en la ocupación temporal del canal, por llevando a la congestión de este.

Por este motivo en este trabajo se han estudiado varios intervalos diferentes de *beacon* para los diferentes conjuntos de pruebas, con la finalidad de encontrar el valor más adecuado de forma empírica: 5, 10, 20, 30, 40, 50, 120 y 500 milisegundos. Se ha llegado a la conclusión de que para valores pequeños de intervalo, el rendimiento de la red se ve afectado. Conforme aumenta el *beacon interval*, las pérdidas asociadas a interferencia disminuyen. En un entorno real, con multitud de STAs, este *beacon interval* debe ser alto, puesto que la capacidad máxima de usuarios del sistema depende en gran medida de este valor.

Hay que señalar que para algunos adaptadores usados, un tiempo demasiado alto entre *beacons* llevaba a una desconexión de la STA, por lo que debe adaptarse a cada dispositivo, en caso de ser necesario o bien seleccionar un valor que permita el buen funcionamiento en diversos dispositivos sin saturar demasiado el canal radio.

3.1.4. Esquema de *handoff*

Para cumplir con el estándar se hace uso de un proceso de cambio de canal especificado en la normalización IEEE 802.11n, llamada *Channel Switch Announcement* (CSA). Se puede ver el diagrama del proceso en la figura 3.2, asi como un diagrama de secuencia más detallado en el Anexo capítulo B. Dicho procedimiento está pensado para que un AP anuncie a todas las STAs conectadas por motivos de interferencia a otras redes o tecnologías como rádar que se encuentran en la banda de 5 GHz, se va a cambiar de canal. Entonces, al final del proceso tanto el AP como todas las STAs cambian al canal destino, anunciado en los paquetes CSA.

En el presente trabajo este mecanismo se ha usado para implementar el *seamless handoff*. Es un procedimiento reactivo, basado en el especificado en el estándar, sin embargo en este caso el AP que comenzó el proceso no cambia de canal, solo la STA a la que se le envía, al ser los *beacons* unicast. Existen una serie de pasos que se deben dar para activar el mecanismo de handoff. Cuando la STA se aleja del AP1, este detecta que la señal está por debajo de un umbral arbitrario y envía un mensaje *PUBLISH* al controlador (2). El controlador envía una petición de *scan* (3) a los APs adyacentes a AP1. Estos cambian su interfaz auxiliar al canal A e intentan escuchar paquetes de STA. En caso de poderlo escuchar, se envía una respuesta (4) al controlador incluyendo la potencia recibida desde la STA. Este toma la decisión (5) de a que AP irá destinado la STA basándose en la potencia recibida desde cada AP, por lo que resulta elegido el AP2 que se encuentra en el canal B. Desde el controlador, se inicia el proceso de CSA enviando una directiva *SEND_CSA*(6) al AP2.

Una vez termina de enviar los mensajes de CSA(7) a la STA, consistente en cinco *beacons* en este caso, aunque es un parámetro configurable, el controlador envía de forma simultánea al AP 1 y al AP 2 los mensajes *REMOVE_LVAP* y *ADD_LVAP*(8) respectivamente, por lo que el LVAP pasa de un agente a otro. El cliente, si cumple con el estándar, cambiará de canal tras el último *beacon* del agente origen, y esperará mensajes del LVAP. Durante este proceso, el AP 2 ya ha comenzado a emitir tramas *beacon*, con la dirección MAC del LVAP, y tras las pruebas se ha observado que tras tres *beacons* con éxito normalmente se reanuda la comunicación, aunque esto puede variar, por pérdidas de algún *beacon* u otros motivos como implementación del driver del STA.

Para conseguir esto, se ha tenido que conformar el paquete, modificando la generación de *beacons* que ya estaba implementada en los agentes. El paquete de CSA (ver **??**) consta de los

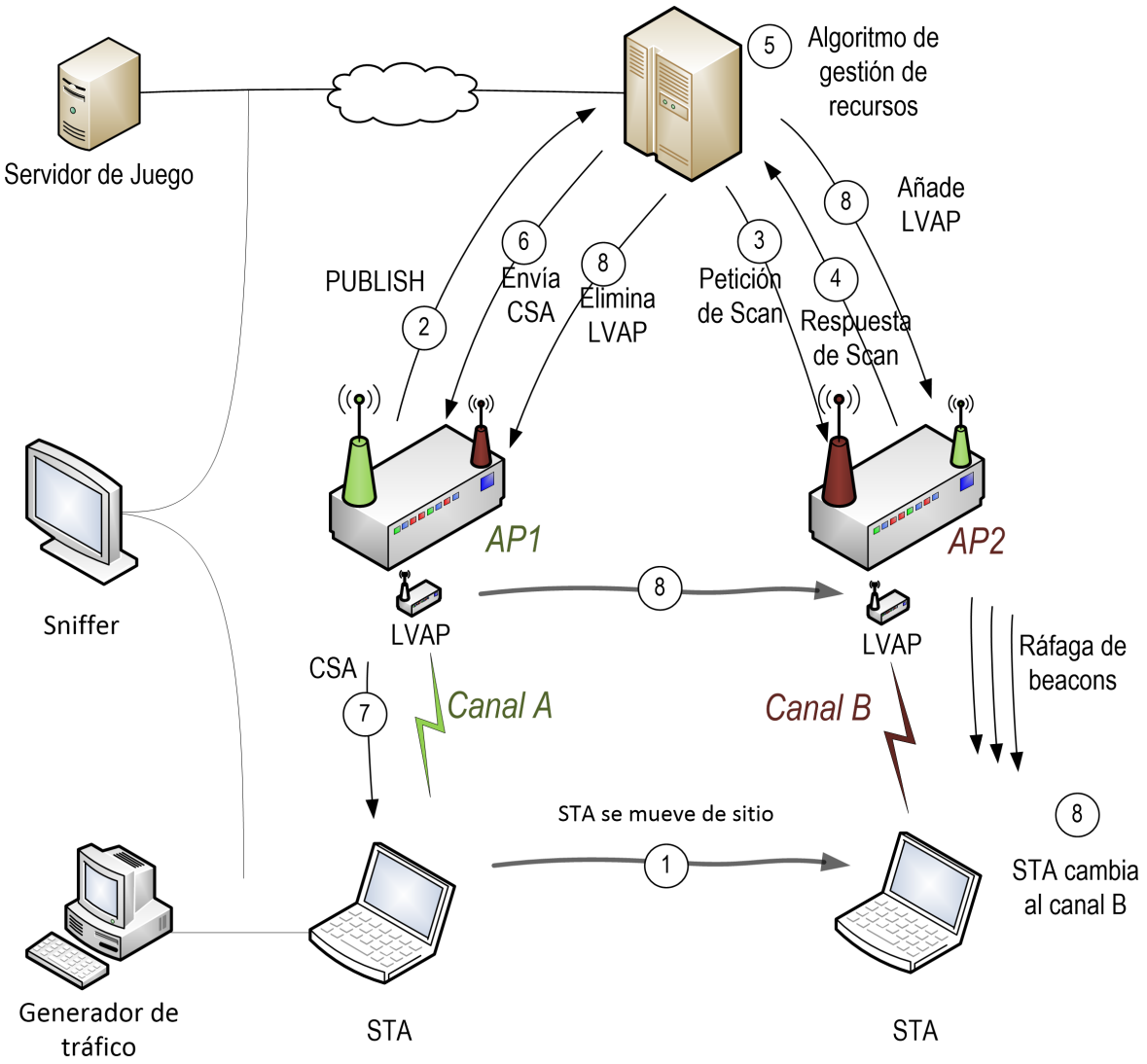


Figura 3.2: Diagrama ilustrativo del mecanismo de *CSA*

mismos campos que un *beacon* periódico normal, al que se añade un campo *Tag* de valor 37, o lo que es lo mismo, del tipo *Channel Switch Announcement*, que incluye tres campos:

- El campo *Mode* indica si existe alguna restricción de transmisión hasta que se efectúe el cambio de canal. En nuestra implementación tiene valor 0, por lo que no hay restricciones.
- El segundo campo, *Number*, indica el canal al que se tiene que cambiar la STA, en este caso es el canal 9.
- El tercer campo, *Count*, se trata de un contador decreciente. En este caso, indica que quedan 4 beacons de este tipo para que la STA efectúe el cambio de canal.

Código 3.4: Paquete *beacon* del tipo *CSA*

```
Frame 13204: 103 bytes on wire (824 bits),
             103 bytes captured (824 bits)
Radiotap Header v0, Length 26
IEEE 802.11 Beacon frame, Flags: ........C (24 bytes)
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x000000000aec8b82
        Beacon Interval: 5.120000 [Seconds]
        Capabilities Information: 0x0001
    Tagged parameters (37 bytes)
        Tag: SSID parameter set: Odin-wi5-labtel
        Tag: Supported Rates 12, 18, 24, 54, [Mbit/sec]
        Tag: DS Parameter set: Current Channel: 4
        Tag: Traffic Indication Map (TIM): DTIM 0 of 0 bitmap
        Tag: Channel Switch Announcement Mode: 0,...
                ...Number: 9 , Count: 4
```

El tiempo entre *beacons* o *beacon interval* es un parámetro crítico en el CSA, dado a que de este depende el tiempo de interrupción de la comunicación. Uno de los cometidos ha sido minimizar este tiempo de interrupción, sin afectar a la calidad de las comunicaciones. Sin embargo, como se presenta anteriormente, el *beacon interval* es causante de una ocupación del canal para valores de este muy pequeños, por lo que no se puede disminuir sin perjudicar el rendimiento de todos los clientes. La solución propuesta es introducir un *beacon interval* variable, para que mientras no haya necesidad de CSA, se mantenga un *beacon interval* alto, sin embargo, llegado el momento del cambio de canal, se aumenta este tiempo, enviando una ráfaga de una duración corta, pero lo suficiente como para que el STA haya realizado la operación con éxito.

3.1.5. Gestión del tráfico de control

Otra mejora que se realizó fue la reducción del tráfico de control. En el escenario descrito con detalle en la apartado 3.3.1, una vez se optimizaron otros parámetros que afectaban al rendimiento, se pudo observar un tráfico TCP entre el controlador y los agentes, que

consumía un ancho de banda muy elevado. Para aislar el problema, se usó D-ITG con la opción de *Quake3*, y también de forma accesoria tráfico con *iperf* e *icmp*. En la figura 3.3 puede observarse la diferencia entre el tráfico detectado, de color gris, y el tráfico útil de la aplicación D-ITG, de color verde. De color marrón y apenas visible, se encuentra el tráfico UDP de control entre el controlador y los agentes.

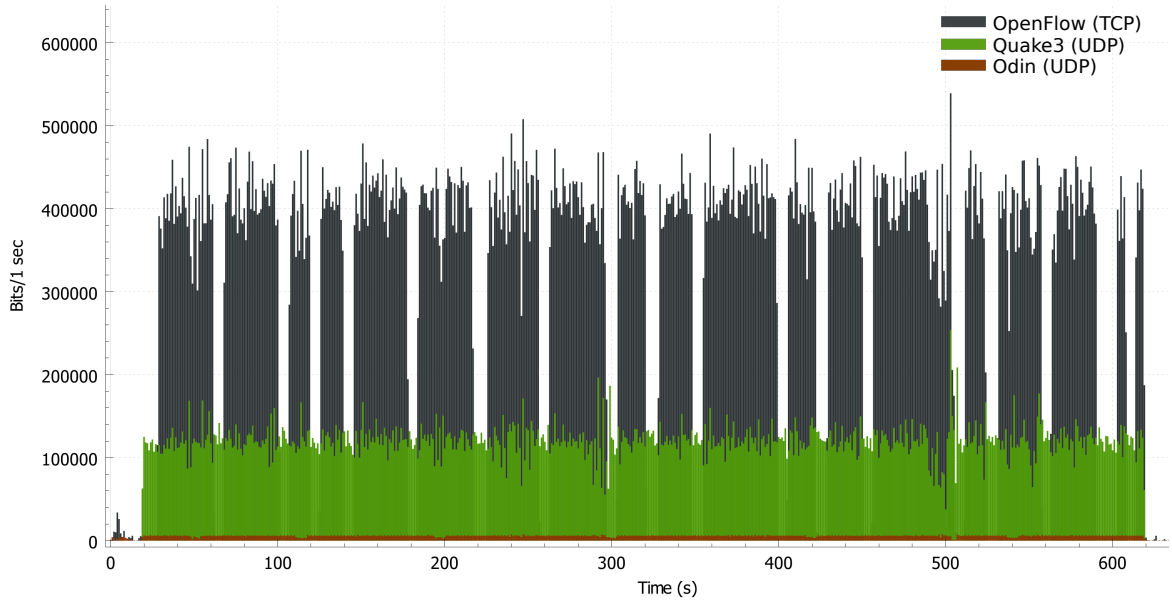


Figura 3.3: Tráfico en el hub

Este tráfico se trata de flujo *OpenFlow*, y se genera desde los agentes hacia el controlador, cuando reciben un paquete del cual no saben el destino. Se generan tres paquetes, *packetIn*, *packetOut* y *flowMod*. El *packetIn* va desde el agente hacia el controlador, indicando que le ha llegado un paquete de un flujo desconocido, por lo que no sabe como direccionarlo. Los paquetes *packetOut* y *flowMod* son generados por el controlador como respuesta a este, indicándole al agente que debe liberar el paquete, y hacia donde debe enviarlo. Esto se debe a una configuración indebida de las reglas de *OVS* en los agentes.

Para solucionar esta problemática, se incluyeron en la configuración de los agentes las siguientes líneas al fichero *init_cli.sh* (Ver Código 3.5). De esta forma se añade a la interfaz *br0* del AP las reglas para que cuando lleguen mensajes desde el puerto 1 y tienen como origen la dirección física del interfaz Wi-Fi de la STA, deben dirigirse al puerto 2 del switch virtual, que es el puerto conectado a la red cableada. También hay que agregar la regla inversa, la que tiene como destino la STA.

Código 3.5: Reglas de OVS (caso concreto)

```
ovs-ofctl add-flow br0 in_port=1,dl_type=0x0800,...
...dl_src=60:e3:27:1d:00:f9,actions=output:2
ovs-ofctl add-flow br0 in_port=2,dl_type=0x0800,...
...dl_dst=60:e3:27:1d:00:f9,actions=output:1
```

Este problema causaba una congestión en el plano de datos, llegando incluso la desconexión de los agentes. Sin embargo, esta solución no es definitiva, por lo que se ha de

modificar el código del controlador para que se añadan automáticamente estas reglas cada vez que se agrega un LVAP a un AP.

3.1.6. Reducción de retransmisiones Wi-Fi

Al igual que en la apartado 3.1.5, gracias a las pruebas en base al escenario propuesto en la apartado 3.2.1 se pudo observar capturando en el interfaz radio que existían un gran número de paquetes repetidos (retransmisiones). Estas pueden deberse por dos motivos principalmente: la pérdida del paquete original, o la pérdida del *ACK* de la capa Wi-Fi que debe enviar el destinatario del primer paquete. Para reducir el ancho de banda usado por cada STA, se propuso la reducción del número de retransmisiones que se realizan por el agente, a 1 o 2.

Esto es posible en Linux a través del comando *iw* (ver Código 3.6), que permite la configuración de las interfaces inalámbricas del sistema. Por defecto, el número de retransmisiones Wi-Fi en los dispositivos estudiados es de 8, en caso de no recibir el *ACK* correspondiente. Esto suele suceder cuando hay un alto nivel de interferencia, aunque es habitual la observación de pérdidas, que gracias a esta capacidad de Wi-Fi se solventan de forma fácil, a costa de uso de recursos.

Código 3.6: Reducción de retransmisiones

```
# Para paquetes Wi-Fi cuya longitud
# es menor que el umbral RTS/CTS
iw phy <device> set retry short <maxRetriesShort>
# Para paquetes Wi-Fi cuya longitud
# es mayor que el umbral RTS/CTS
iw phy <device> set retry long <maxRetriesLong>
```

Por desgracia, este comando no funciona con el firmware usado en los APs (OpenWRT 14.07) es posible que en futuras versiones si funcione. Se intentó tanto en "frío" como en "caliente" con el mismo resultado sin éxito. Se desconoce si el problema se debe al sistema operativo usado, el cual tiene ciertas limitaciones al ser un sistema ligero, o puede ser debido a alguno de los paquetes de software usados internamente como Click u OVS. En las STAs, si se tratan de un sistema Linux, si que es posible modificar el parámetro, sin embargo dado el enfoque de uso general de la red no es algo deseable para el usuario tener que modificar este parámetro.

3.1.7. Evaluación de aplicaciones

En SDN existen aplicaciones de red corriendo en el controlador para realizar diferentes tareas de gestión de la red de forma automatizada. Las hay de dos tipos: *northbound* y *southbound*. Las aplicaciones *northbound* se encargan de proveer un correcto servicio a las aplicaciones de más alto nivel que corren los clientes, como por ejemplo podría ser en el caso de la integración en la SDN de un almacenamiento *cloud*. Las aplicaciones *southbound* son las encargadas de la comunicación entre el controlador y los elementos de más bajo nivel, a través de una interfaz (en este caso OpenFlow), como podrían ser los elementos físicos que

componen la red. La correcta implementación de estas aplicaciones es crítica en las SDN sin las cuales nada podría funcionar.

En el controlador de la plataforma Odin, las aplicaciones *southbound* pueden ser reactivas o proactivas. Las aplicaciones reactivas, como su propio nombre indica, reaccionan a un evento dado en la red. Por el contrario, las aplicaciones proactivas no esperan a un evento sino que realizan operaciones de forma periódica. Inicialmente Odin cuenta con dos aplicaciones: SimpleLoadBalancer (proactiva) y OdinMobilityManager (reactiva). La aplicación SimpleLoadBalancer reasigna cada cierto tiempo todas las STA conectadas a los APs de la forma más equitativa posible, asignando las STA una a una a cada AP. Por otro lado la aplicación OdinMobilityManager funciona con un sistema de subscripciones, en el que los AP "subscriben" a cada STA a Odin y van emitiendo mensajes de forma periódica al controlador con información de las STAs activas. En caso de recibir un mensaje *PUBLISH* de un AP, se inician una serie de comparaciones para determinar la situación de la STA que viene referida en el mensaje, y se determina si es necesario hacer *handoff*.

Una de las aplicaciones más usadas en este trabajo es HandoverMultichannel, la cual deriva de OdinMobilityManager. Al igual que su antecesora se trata de una aplicación reactiva porque sigue el sistema de subscripciones. Esta se encarga de avisar a los agentes desde el controlador de cuando deben realizar un cambio de canal para un cliente concreto. Este cambio de canal se basa en la potencia medida del cliente objetivo por parte del agente. Si está por debajo de un umbral, se realiza el *handoff* al cabo de un tiempo determinado. Este tiempo es un parámetro configurable, y aunque no es el caso, se podrían usar otros parámetros (por ejemplo, ancho de banda disponible u homogeneidad de los servicios en un AP) para decidir el *handoff*. Está diseñada para funcionar con dos APs por lo que se trata de una delimitación de OdinMobilityManager, no obstante se comporta ligeramente diferente y además permite *handoffs* entre canales.

Para lograr el correcto funcionamiento del *handoff* entre canales, en el código de Odin se han modificado métodos existentes o añadido nuevos si ha sido necesario. Por ejemplo, en el código encargado de la gestión de clientes de Odin, se ha añadido la funcionalidad de cambiar de canal a frecuencia, y viceversa. Se ha modificado también parte del código correspondiente en los agentes (APs)) puesto que los paquetes en el controlador y los agentes son complementarios.

Para obtener información de las redes del entorno, así como de los clientes, se ha propuesto una función de *scanning* en los agentes. Esta aplicación se basa en la posibilidad de implementar en los APs un segundo dispositivo Wi-Fi auxiliar por medio de una interfaz USB. De esta manera, les es más fácil monitorear diferentes canales, y también da la posibilidad de dar una lista de los clientes a la vista y la potencia recibida al controlador, con lo que podría hacer un mapa de clientes espaciados por potencia, para poder tomar decisiones de traspaso. También, en caso de que hubiera mucha interferencia en el canal actual, el agente podría cambiar de canal o el controlador podría modificar la distribución de canales.

Sin embargo, con el hardware y el software utilizado en este trabajo, no se ha podido añadir la funcionalidad de un adaptador inalámbrico adicional a los AP. En la versión 15.05 de OpenWRT si que es posible esta opción, sin embargo es necesario hardware con una memoria de almacenamiento más grande. Esto se ha podido conseguir en nuevos APs (TP-Link 1750

AC), pero por la limitación de tiempo que tienen los TFGs no ha sido posible probar y analizar esta aplicación. Sin embargo, se sugiere una evaluación detallada de dicha aplicación cómo línea futura.

Para que el controlador tenga la información necesaria y sea capaz de tomar decisiones acerca de la gestión de recursos, de cambiar a los clientes de punto de acceso y en general tomar decisiones acerca del estado de la arquitectura, es necesaria una monitorización continua de los elementos conectados a la red. Esto es posible gracias a la comunicación periódica entre los APs y el controlador, y a una interfaz Wi-Fi auxiliar, ya que por ésta se realiza la monitorización.

Por un lado, es posible modificar los módulos de Click para aumentar la cantidad de estadísticas disponibles para el agente, y que este pueda enviarlos al controlador. Sin embargo, esto es algo que puede afectar al rendimiento en caso de que se dé mucho tráfico y a causa de la limitada capacidad de las máquinas usadas como agente. En estos módulos es posible aumentar los campos de la cabecera *radiotap* almacenados por paquete recibido. En esta cabecera se encuentran parámetros radio como la tasa de datos, la señal recibida o el ruido. Se podrían añadir la frecuencia en la que se ha recibido el paquete y el *timestamp* del paquete.

Además, es interesante la diferenciación de servicios en diferentes APs físicos, puesto que lleva a un aumento del rendimiento, y permite gestionar de forma mas eficiente los recursos. Esto se podría hacer usando el módulo de Click que se encarga de leer el paquete, y sacando al información del protocolo que se encuentra encapsulado. No obstante, por motivos de tiempo y aspectos técnicos relacionados al correcto funcionamiento del segundo dispositivo Wi-Fi no se ha podido terminar el trabajo en esta línea y se recomienda como línea futura el realizar una aplicación para la monitorización automatizada.

Por último, se ha añadido a la arquitectura un modo *debug* en los APs, útil para el proceso de desarrollo y para la ejecución de las pruebas. Se han dispuesto varios niveles diferentes de *debug* regular la cantidad de información, que se configuran en el fichero a_agent.cli del AP.

Para más detalles acerca del código, consultar el repositorio del proyecto Wi-5: `https://github.com/Wi5`. En él, están listados los diferentes paquetes del programa. En *odin-wi5*, se encuentra la wiki del proyecto así como ficheros de consulta generales. En *odin-wi5-controller* está el código fuente del controlador, escrito en Java e integrado en el controlador Floodlight. En *odin-wi5-agent* se encuentra el módulo escrito en C++ que se integra en Click y que ejecutan los APs.

3.2. *Seamless Handoff* con *APs* en el mismo canal

Como ya se ha mencionado, uno de los objetivos de este TFG es conseguir dar soporte a servicios de tiempo real con suficiente calidad en la plataforma Odin-Wi5. Esto incluye tener en cuenta la coexistencia de estas aplicaciones con otros servicios y algoritmos de gestión de recursos. Además, los *seamless handoffs* entre APs no son solo necesarios para soportar la movilidad de STAs nómadas sino también en el hipotético caso de que una STA estática sea asignada a otro AP, si fuera necesario para optimizar los recursos de la red. En [23] se refiere a los FPS como uno de los servicios de tiempo real más estrictos, motivo por el cuál se ha

elegido este tipo de juegos como caso concreto de estudio para esta prueba. Dicho esto, se plantean dos cuestiones:

– ¿Es posible realizar un *seamless handoff* entre APs manteniendo unas condiciones de servicio adecuadas?

– ¿Debe considerarse el hecho de tener un jugador en un AP a la hora de decidir un *handoff*?

3.2.1. Escenario para las pruebas

El proyecto Wi-5 se encontraba ya en fase de desarrollo cuando se empezó a trabajar en el TFG, por lo que existía un escenario ya puesto en marcha desde el principio. En la figura 3.4 se presenta dicho escenario. Existen varias subredes privadas y una pública, usando el rango de direcciones que tiene la universidad, dentro del escenario, separadas por hubs.

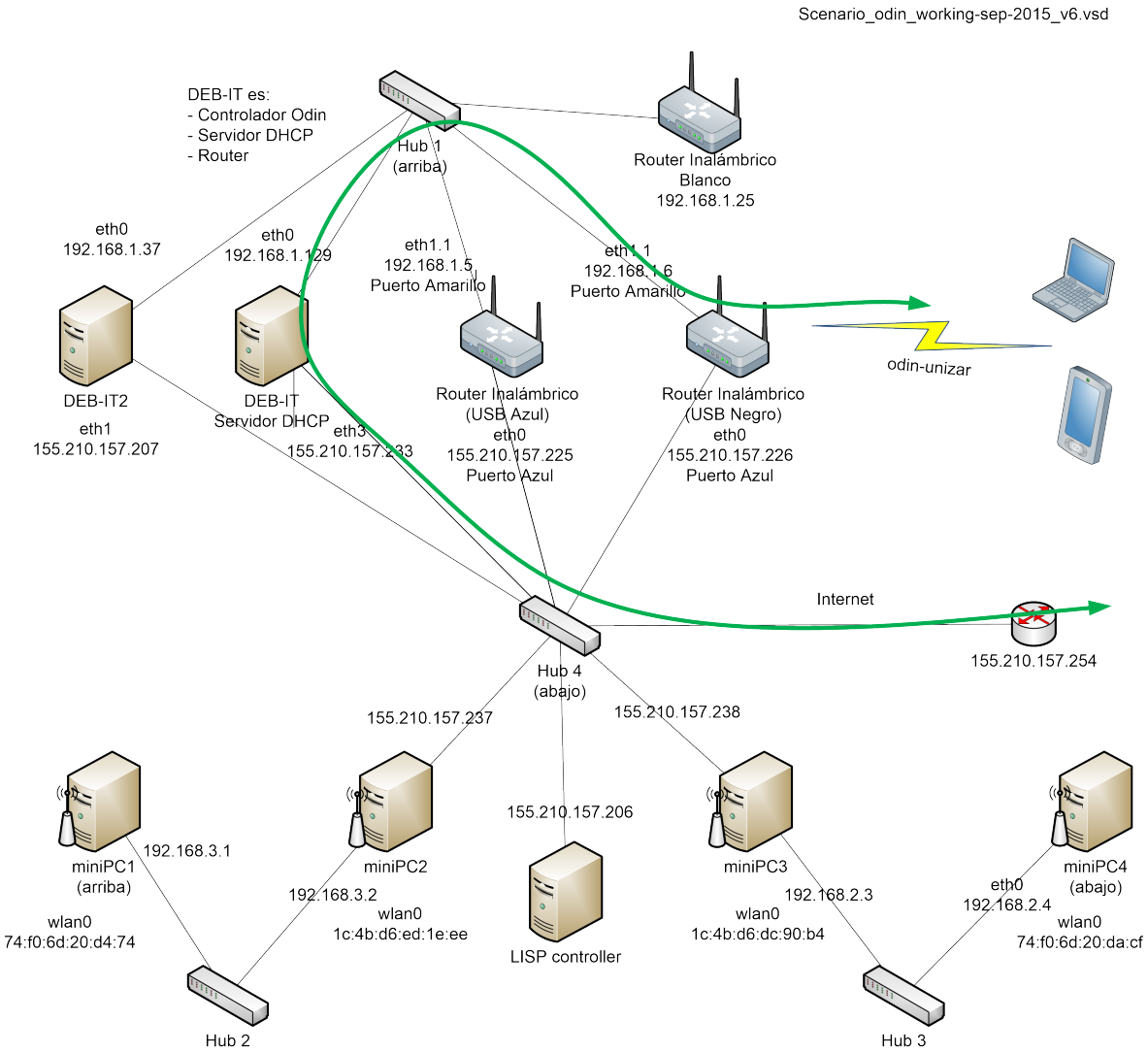


Figura 3.4: Escenario Inicial (Septiembre 2015)

Una primera subred privada consiste en tres puntos de acceso inalámbrico, de los cuales se usaban uno o dos para las pruebas iniciales. Estos son el Router Inalámbrico (USB Azul)

y el Router Inalámbrico (USB Negro). El controlador estaba situado en la máquina *DEB-IT*, donde además también corría el servidor DHCP, para dar un servicio centralizado. Existía una segunda máquina de reserva, el *DEB-IT2*, por si fallaba el otro controlador o existiera cualquier otro problema. Además está el Router Inalámbrico Blanco, que se usó en una versión previa del sistema pero para este trabajo no se llegó a usar. Esta primera subred era la 192.168.1.0/24, que coincide con el rango de direcciones al que provee servicio el servidor DHCP, es decir, el rango de direcciones de las STA en Odin. Los puntos de acceso útiles se dispusieron de diferente forma a lo largo del análisis, dejando uno fijo cerca y otro a diferentes distancias. Todos los elementos de esta subred salvo el Router Inalámbrico Blanco tienen otra interfaz pública, para habilitar el acceso remoto desde redes externas, usando por ejemplo *Secure Shell* (SSH), lo que permite su manejo desde el exterior.

Existen otras dos subredes privadas, la 192.168.3.0/24 y la 192.168.2.0/24, cada una de dos elementos, miniPC1 y miniPC2 están en la primera subred mientras que miniPC3 y miniPC4 están en la segunda. Todos estos miniPC son máquinas usadas como clientes de Odin, dado que además de ethernet tienen interfaz inalámbrico integrado. Son máquinas que corren diferentes versiones de Debian. De estos, solo miniPC2 y miniPC3 tienen dirección pública, por lo que se puede acceder directa o indirectamente a todas las máquinas por acceso remoto.

El dominio de rango direcciones públicas está dentro de la subred 155.210.157.0/24 lo cual permite el acceso remoto mediante SSH, además de acceso a internet desde Odin a través del *DEB-IT*, que actúa como puerta de enlace. Los *miniPCs* 2 y 3 tienen también acceso a internet de forma autónoma, aunque durante las pruebas se deshabilitaba siempre que fuera posible, en caso de estar usando dicha máquina.

En resumen, las características de las máquinas usadas son las siguientes:

- Dos puntos de acceso TP-Link1043NDv2[1], usando el sistema OpenWRT ver. 14.07.
- El controlador es un PC con un sistema operativo Linux Debian 6.
- El tráfico es generado por una máquina Linux conectada a la red Wi-Fi, usando el generador de tráfico D-ITG [22], que incluye una opción para generar tráfico emulando un cliente de *Quake3*.
- El tráfico es recibido por un servidor D-ITG en el controlador.
- Adicionalmente, para una variante de las pruebas "básicas" se ha usado otra máquina Linux conectada a uno de los APs. Esta máquina esta realizando una descarga de un fichero desde la web, en concreto una imagen ISO de Debian.

3.2.2. Procedimiento utilizado

Inicialmente, se realizaron capturas en diferentes puntos clave de la infraestructura: en el *DEB-IT*, en la interfaz inalámbrica del punto o puntos de acceso que se estuvieran usando en la prueba, y en la interfaz inalámbrica del cliente o clientes que estuvieran activos. Asimismo se usaba un portátil o un *miniPC* inactivo para tener monitorizado el canal Wi-Fi que estuviera

[1] http://www.tp-link.com/en/download/TL-WR1043ND_V2.html

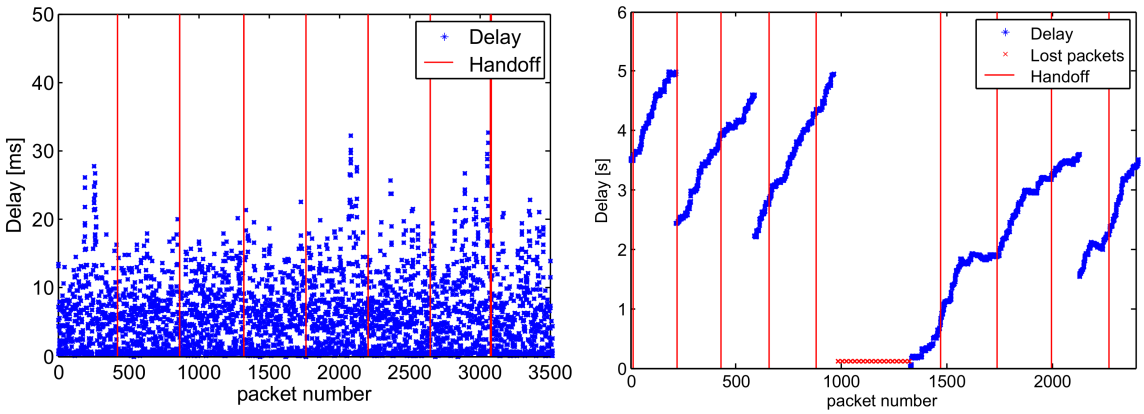
ocupado en ese momento por el cliente. Este canal no ha sido siempre el mismo, variando debido a motivos de interferencia con otras redes inalámbricas existentes en el entorno, para que las medidas realizadas estuvieran hechas independientemente del escenario.

En el primer experimento, el controlador Odin ejecuta la aplicación OdinMobilityManager (ver apartado 3.1.7) y una STA genera tráfico usando D-ITG de tipo *Quake3*. La STA cambia de AP de forma periódica. En el segundo experimento se usaron los mismos elementos que en el primero, añadiendo una STA estática que se descarga un archivo grande de un servidor, usando TCP. Cada experimento tiene una duración total de 600 segundos.

Una vez ha terminado la transmisión, se usan los ficheros generados por el generador de tráfico D-ITG para obtener los resultados, y se usan las capturas de Wireshark para confirmar los datos. Los ficheros se exportan a formato CSV, y se importan en Matlab para su procesado. Se ha usado un *script* llamado ScriptTrafQuake (ver Código D). Sin embargo, para estas pruebas se usó una versión anterior a la incluida con menos funcionalidades, aunque válida para estos cálculos.

3.2.3. Análisis de los resultados

El *handoff* se probó con varias configuraciones diferentes, aunque en el caso presentado la aplicación de movilidad está configurada para realizar el *handoff* entre APs cada 3 segundos.



(a) Retardo, sin tráfico de fondo

(b) Retardo, con tráfico de fondo

Figura 3.5: Retardos para los dos experimentos

En el primer experimento se obtuvieron buenos valores de retardo, como se muestra en la figura 3.5a, que es una gráfica del retardo de cada paquete durante 24 segundos (8 *handoffs*). Se puede observar que no hay un aumento considerable del retardo a causa de los *handoffs* y que este se encuentra por debajo de los 15 ms. Además, la tasa de pérdidas es del 3,25 % y el *jitter* es 5,5 ms.

Para estimar la calidad del usuario se ha usado un estimador de calidad subjetiva [21], que tomando como parámetros en el retardo y el jitter, se ha usado para obtener los resultados presentados en la tabla 3.1. Se han añadido valores típicos de *Round Trip Time*, considerando que un servidor al que quiera acceder un jugador pudiera estar en localizaciones

remotas, considerando que el escenario las máquinas están en la misma LAN. El MOS (ver apartado 2.6.4) está por encima de 3 para todos los casos, por lo que se trata de un valor decente para jugar.

Tabla 3.1: Estimación de la calidad dependiendo del retardo en el primer experimento.

Escenario	**Estimador G-Model**		
	retardo [ms]	*jitter* [ms]	**MOS**
LAN	5	5,5	3,73
Intra-region	20	5,5	3,58
Inter-region	80	5,5	3,04

A continuación, podemos observar en la figura 3.5b que en las codiciones del segundo experimento la aplicación se hace injugable. Se observa que el retardo aumenta considerablemente en esta prueba, y además si existe una cierta correlación de las pérdidas con los instantes de *handoff*. Además, hay una ráfaga de pérdidas de más de 400 paquetes, lo cual haría que desde el punto de vista del jugador, la aplicación sufriera de un jitter elevado.

En resumen, se concluye con estas pruebas que es posible solucionar el problema de los *seamless handoffs* usando LVAPs. El ha demostrado que estos no influyen al retardo de forma considerable. Respondiendo a la segunda pregunta, queda claro que un servicio TCP afecta de forma considerable a las condiciones del servicio de tiempo real si se encuentran en el mismo AP, por consiguiente se debe considerar el hecho de separar los servicios en diferentes APs.

3.3. *Seamless handoff* con *APs* en canales diferentes

En el primer conjunto de experimentos se implementó el mecanismo de *seamless handoff* entre APs y se analizó su comportamiento en la plataforma Odin. Sin embargo, todos los APs se encuentran en el mismo canal, por lo que existen problemas de escalabilidad en esta solución a causa principalmente de la interferencia entre los propios AP solapados, y de las STAs conectadas. Además, se ha llegado a la conclusión de que la decisión de cambiar una STA de AP por parte del controlador no es únicamente por movilidad, sino que puede deberse a balance de carga u otros parámetros. Además de esto, debido al uso de LVAPs como solución trae consigo un problema, dado que ahora los paquetes de difusión que usan los APs en las comunicaciones Wi-Fi y en condiciones normales llegan a todas las STAs conectadas, ahora son *unicast*, por lo que se añade otro problema de escalabilidad añadida.

En consecuencia a los problemas planteados, este experimento:

- La solución usada debe ser general para cualquier STA sin hacer modificaciones en ella.
- Se debe implementar un mecanismo de *seamless handoff* con APs en diferentes canales, para solucionar el problema de escalabilidad.
- La generación de *beacons* debe estar limitada.

Una vez implementado el *seamless handoff* entre canales diferentes (ver apartado 3.1.4), se pudo plantear un escenario para la consecución de los objetivos propuestos.

3.3.1. Escenario para las pruebas

Inicialmente, se planteó un escenario en el cual se necesitaba una máquina con dos tarjetas de red o dos adaptadores de red USB que fueran iguales. Se probó primero con un PC del laboratorio equipado con dos tarjetas PCI, sin embargo no capturaban la cabecera *radiotap* ni los paquetes de datos, cuando estaban en modo monitor. Se optó por usar dos adaptadores de red USB, que eran compatibles con los *miniPCs* donde se realizaban las capturas. Aunque ya se sabía la posible problemática al capturar en la interfaz radio debida a las pérdidas de paquetes en dicha interfaz por diversas causas (interferencia electromagnética o baja potencia de recepción), esta añade cierto error de medida en las pruebas. Hay que tener en cuenta que al capturar en modo monitor, la recepción de paquetes es muy ineficiente. Debido esto, se hace inviable el uso de una única máquina capturando la interfaz radio para las pruebas y hace plantearse una alternativa.

En el escenario propuesto finalmente, solo existen dos capturas útiles, una en origen y otra en destino, de esta manera se eliminaría la incertidumbre introducida por el Wi-Fi medido, aunque por contra no se podría medir la potencia o las tramas de nivel de enlace 802.11 como son por ejemplo las tramas *beacon* o las tramas *Request To Send* y *Clear To Send* de control de flujo. De todas formas, se siguió usando una máquina para este cometido, aunque no se pudiera medir todo el tráfico y no se usaran las capturas de esta interfaz de forma extensiva, simplemente a modo de comprobación

Se realizaron algunos cambios respecto al escenario en la prueba anterior, como se muestra en la figura 3.6. En una máquina se origina el tráfico, conectada a un *hub* ethernet. A este *hub* se conecta la máquina capturadora o *sniffer*, con un interfaz sin dirección *Internet Protocol* (IP), para evitar cualquier interacción indeseada con la red. A otro puerto del *hub* se conecta DebianMini2, que hace deSTA, y está equipada con tres interfaces de red. Las otras dos interfaces son una inalámbrica y la otra ethernet. Esta configuración evita la degradación de los datos capturasdos y permite probar diferentes adaptadores inalámbricos realizando pocos cambios.

Se usó una nueva máquina como hypervisor, y dentro de ella varias maquinas virtuales:

- El controlador de Odin.
- Un servidor DHCP, que además hace de puerta de enlace al exterior.
- Un servidor de juegos y multimedia, usado como destinatario de tráfico durante las pruebas.

La red cableada conectada a los agentes (APs) es igual a la que se presentaba en el segundo escenario, con el añadido de que al *switch* donde están conectados todos los elementos se conecta también el *sniffer*, de la misma manera que se había conectado al *hub* de origen. De esta forma el *sniffer* captura ambas interfaces con los relojes de las dos capturas sincronizados, por lo que cualquier paquete que no llega a su destino se puede asegurar que se ha perdido o bien por interferencias en el interfaz radio o bien por errores en algún agente, contando con el buen funcionamiento del resto de elementos del escenario.

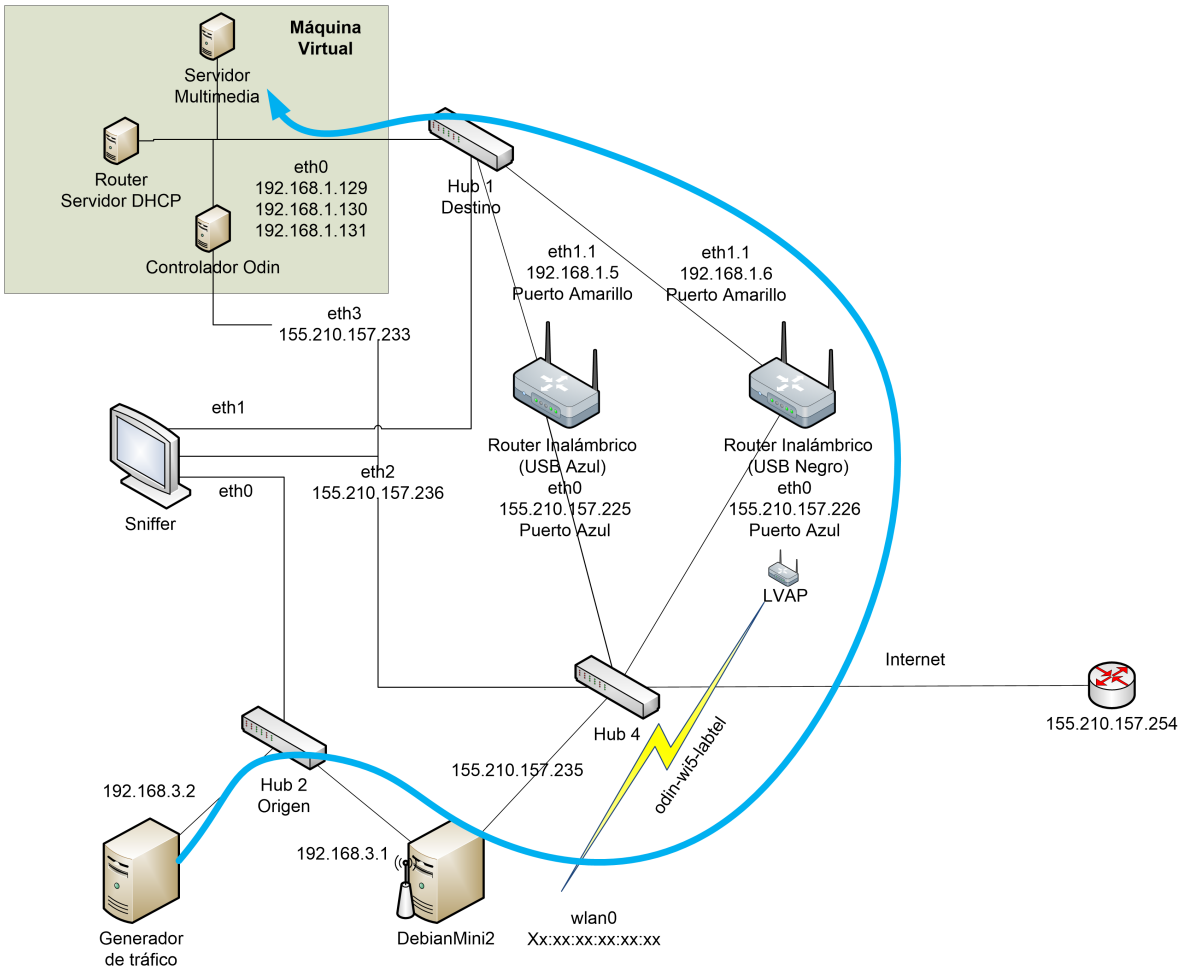


Figura 3.6: Escenario Final (Julio 2016)

(a) Linksys WUSB54GC (b) WiPi (c) TL-WN722N

Figura 3.7: Adaptadores inalámbricos usados

Para las pruebas se usaron tres adaptadores inalámbricos de diferentes fabricantes, cuyos modelos se muestran en la figura 3.7 (Linksys WUSB54GC[2], WiPi WLAN USB b/g/n[3], y TP-LINK TL-WN722N[4]). Para cada adaptador se plantearon ocho experimentos diferentes, habiendo 18 experimentos en total.

3.3.2. Procedimiento utilizado

Para la realización de las pruebas, se semiautomatizó el proceso, puesto que la secuencia de pruebas podía llegar a las 3 horas continuadas sin contar la adaptación de los datos obtenidos para importarlos en Matlab, de forma que aunque había que estar pendiente por si hubiera algún tipo de problema, casi no hacía falta interacción. Para ello se realizaron varios scripts (ver Apartado 3.1.2 y Anexos capítulo A, capítulo C, capítulo D). Además, se añadió una funcionalidad de *logging* al controlador Odin, para que pudiera comprobar con facilidad si la prueba había sido válida.

En estas pruebas se intentó minimizar cualquier tipo de interferencia para poder aislar la influencia de los tiempos de *beacon*, por lo que se acercaron los puntos de acceso dejando aproximadamente un metro de distancia entre sí y se minimizó la pérdida de canal acercando el adaptador al punto medio entre los puntos de acceso. Se eligió en cada momento el canal óptimo para la captura, por lo que previamente, usando *kismet* (ver apartado 2.7), se comprueba el estado de los canales. Esto era necesario debido a que las redes del entorno (por ejemplo, wiuz o eduroam) no estaban siempre ocupando los mismos canales.

Para cada experimento, se varía el valor de *beacon interval* (BI). Los valores usados son **10,20,30,40,50,120,500**[ms], teniendo además dos versiones de 120 y 500 milisegundos con ráfagas de *beacons* de $BI = 10$ ms. El controlador comienza el mecanismo de *handoff* cada 30 segundos. Como se comenta en apartado 3.1.4, de esta forma se mejoran las condiciones de *handoff* y se reduce el tiempo de este. El tiempo de *handoff* se estima desde e punto de vista de la aplicación, dado que durante los *handoffs* hay ráfagas de paquetes perdidos. Dicho tiempo se calcula como el instante entre el último paquete observado en la traza de transmisión y el primer paquete observado en la traza de recepción, existiendo una ráfaga de paquetes perdidos entre ambos paquetes.

Para generar el tráfico se usa D-ITG configurado para replicar una traza de Quake3, y la transmisión tiene una duración de 600 segundos, habiendo 20 *handoffs* durante el proceso.

3.3.3. Análisis de los resultados

En la figura 3.8 se muestra el retardo obtenido para cada adaptador inalámbrico, para un $BI = 10ms$. Se han añadido lineas verticales en los momentos en los que se ha detectado *handoff* siguiendo el procedimiento descrito previamente. Podemos comprobar que para el adaptador TP-Link se detectan los 20 *handoffs* mientras que para WiPi y Linksys *handoffs* que se hacen indetectables, debido a que no se puede distinguir entre paquetes perdidos a causa del *handoff* de forma directa o los que se pierden de forma habitual en el canal. La no

[2] http://www.linksys.com/mx/support-product?pid=01t80000003KPGGAA4

[3] http://es.farnell.com/element14/wipi/dongle-wifi-usb-for-raspberry/dp/2133900

[4] http://www.tp-link.es/products/details/cat-11_TL-WN722N.html

detección de estos *handoffs* es síntoma de que la comunicación es de buena calidad. Además, se observa que los instantes cambios de canal no conllevan un aumento brusco del retardo, sino que el retardo depende del canal en el que se encuentra. Esto se observa de forma muy clara en el caso de estudio del Linksys.

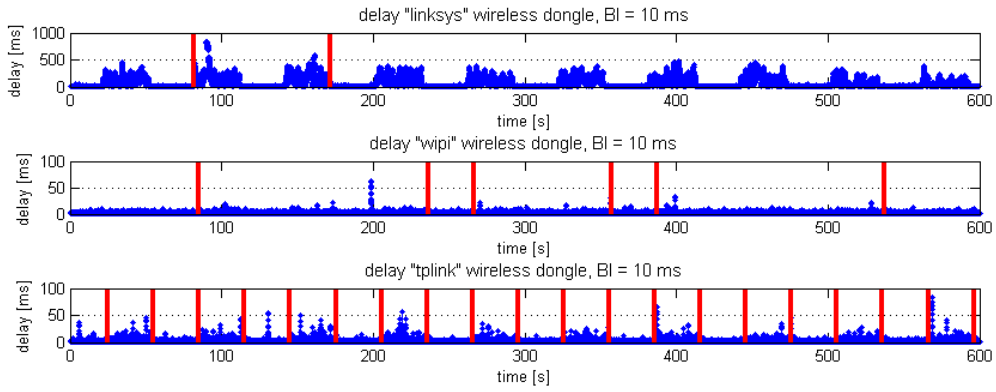


Figura 3.8: Retardo y detección de los cambios de canal, $BI = 10$ ms

Tabla 3.2: Resultados relativos a pérdidas y handoff, Adaptador Linksys

BI [ms]	*detección*	*duplicados*	*perdidos*	$perdidos_{handoff}$	*enviados*
5	10 %	8	9,10 %	0,31 %	86351
10	10 %	6	5,64 %	0,06 %	86221
20	25 %	8	4,11 %	1,22 %	85997
30	30 %	3	2,61 %	0,49 %	86534
40	65 %	1	1,43 %	3,39 %	86541
50	90 %	0	0,08 %	49,32 %	86039
120	100 %	0	0,83 %	23,11 %	86237
500	100 %	0	2,62 %	79,61 %	86321
120-10	0 %	5	1,21 %	-	86306
500-10	20 %	8	0,55 %	0,84 %	86207

En las tablas 3.2 a 3.4 se muestran los paquetes perdidos en total y las ráfagas que se han detectado en el *handoff* para cada experimento. Cada adaptador se comporta de manera diferente dependiendo del *beacon interval* y de si se usa o no la técnica de ráfaga: los adaptadores Linksys y Wipi (ver tablas 3.2 y 3.3) se observa como el porcentaje de pérdidas atribuidas al *handoff* es muy pequeña respecto al porcentaje total de pérdidas, y que la detección es inversamente proporcional al *beacon interval*. En cambio, en el adaptador TP-Link (ver tabla 3.4 casi todas las pérdidas son atribuidas al proceso de *handoff* y se detectan todos ellos. Se debe señalar la existencia de paquetes duplicados en este caso, que aunque no afectan al rendimiento, si se trata de tráfico innecesario. No se ha logrado solucionar este hecho aunque es un aspecto que se debe tratar en el futuro.

En las tablas 3.5 a 3.7 se muestran tanto las medidas de retardo y *jitter* para cada experimento como el mínimo tiempo de *handoff* detectado para cada caso. En la tabla 3.5 se observa que los tiempos de retardo para el adaptador Linksys son inversamente

Tabla 3.3: Resultados relativos a pérdidas y handoff, Adaptador WiPi

BI [ms]	$detección$	$duplicados$	$perdidos$	$perdidos_{handoff}$	$enviados$
5	20 %	500	1,24 %	0,84 %	86368
10	30 %	458	1,09 %	0,95 %	86304
20	35 %	674	1,39 %	3,69 %	86104
30	50 %	458	0,80 %	8,85 %	86485
40	65 %	802	0,16 %	86,86 %	86158
50	90 %	801	0,30 %	87,01 %	86071
120	100 %	824	1,53 %	61,17 %	86062
500	100 %	551	5,35 %	90,36 %	86088
120-10	25 %	691	0,63 %	2,41 %	86039
500-10	45 %	558	0,55 %	3,39 %	85850

Tabla 3.4: Resultados relativos a pérdidas y handoff, Adaptador TP-Link

BI [ms]	$detección$	$duplicados$	$perdidos$	$perdidos_{handoff}$	$enviados$
5	100 %	0	0,31 %	99,63 %	86214
10	100 %	0	0,33 %	100,00 %	86173
20	100 %	0	0,36 %	100,00 %	85586
30	100 %	0	0,40 %	100,00 %	85803
40	100 %	0	0,48 %	99,19 %	85894
50	100 %	0	0,60 %	100,00 %	85834
120-10	100 %	0	0,31 %	99,63 %	86474
500-10	100 %	0	0,31 %	100,00 %	85938

proporcionales al *beacon interval* hasta 50 ms, mientras que luego se mantiene estable. El jitter alcanza su punto óptimo en $BI = 50$ ms. Podemos observar como en los experimentos híbridos mantienen un buen valor de retardo y jitter.

Tabla 3.5: Resultados relativos a retardo y *jitter*, Adaptador Linksys

BI [ms]	$detección$	$retardo_{medio}$ [ms]	$jitter$ [ms]	$t_{handoff_{min}}$ [ms]
5	10 %	88,756	119,103	223,891
10	10 %	55,702	88,610	73,800
20	25 %	49,744	79,722	53,129
30	30 %	27,828	67,142	59,004
40	65 %	19,329	51,977	46,258
50	90 %	8,532	17,255	66,800
120	100 %	15,203	34,266	251,138
500	100 %	12,414	73,813	1434,230
120-10	0 %	11,659	29,771	-
500-10	20 %	12,308	21,422	38,094

Tabla 3.6: Resultados relativos a retardo y *jitter*, Adaptador WiPi

BI [ms]	$detección$	$retardo_{medio}$ [ms]	$jitter$ [ms]	$t_{handoff_{min}}$ [ms]
5	20 %	1,880	1,448	28,555
10	30 %	1,736	1,090	22,290
20	35 %	1,793	1,641	35,444
30	50 %	1,588	0,647	30,958
40	65 %	1,559	0,776	43,274
50	90 %	1,508	0,644	61,796
120	100 %	1,583	0,708	202,558
500	100 %	1,644	1,792	1425,471
120-10	25 %	1,602	0,784	20,293
500-10	45 %	1,635	1,121	23,541

El adaptador WiPi presenta muy buenos valores, como se muestra en la tabla 3.6, aunque los tiempos de *handoff* son dependientes del *beacon interval*. En la solución híbrida obtenemos unos buenos valores de *handoff* sin perjudicar el resto de valores.

En la tabla 3.7, correspondiente al adaptador TP-Link, se presentan unos valores de retardo y *jitter* buenos, de la mismo modo que en el adaptador WiPi. Sin embargo este adaptador tiene la peculiaridad de mantener un valor mínimo de *handoff* casi independiente del *beacon interval*. Además, los experimentos para los *beacon interval* configurados a 120 y 500 ms no funcionan correctamente, debido a la desconexión de la STA.

La figura 3.9 muestra una comparación entre los histogramas del tiempo entre paquetes correspondientes a los experimentos realizados con el adaptador Linksys. Puede observarse como en este caso, los histogramas de recepción son muy diferentes a la transmisión. Esto se debe, probablemente, a la existencia de un *buffer* interno en el adaptador, que produce un "aplanamiento" del histograma, aunque el *buffer* no se ha estudiado en detalle. La diferencia

Tabla 3.7: Resultados relativos a retardo y *jitter*, Adaptador TP-Link

BI [ms]	$detección$	$retardo_{medio}$ [ms]	$jitter$ [ms]	$t_{handoff_{min}}$ [ms]
5	100 %	2,176	2,581	89,860
10	100 %	1,867	1,730	87,010
20	100 %	1,598	0,868	89,980
30	100 %	1,583	0,875	92,580
40	100 %	2,062	4,224	93,100
50	100 %	1,552	0,791	109,100
120-10	100 %	1,755	1,809	89,280
500-10	100 %	2,039	6,157	88,160

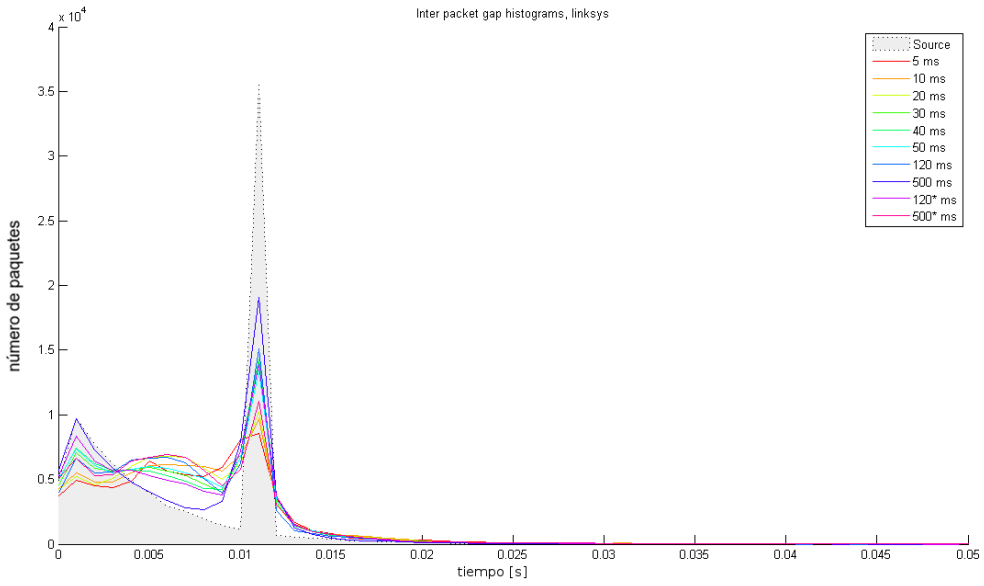


Figura 3.9: Histogramas del tiempo entre paquetes, adaptador Linksys WUSB54GC

entre los histogramas es síntoma de una mala calidad en las comunicaciones, al estar correlado con el jitter y el retardo.

Sin embargo, las figuras 3.10 y 3.11 correspondientes a los histogramas de los adaptadores WiPi y TP-Link respectivamente muestran una morfología similar para los diferentes *beacon intervals*. Para el caso del adaptador WiPi los histogramas son similares en el caso de 120 ms y 120 ms con ráfaga (120*), así como también los de 500 y 500*. Con estos histogramas se puede observar que el patrón de recepción apenas varía respecto al patrón de transmisión, lo cual es signo de unas buenas condiciones de red.

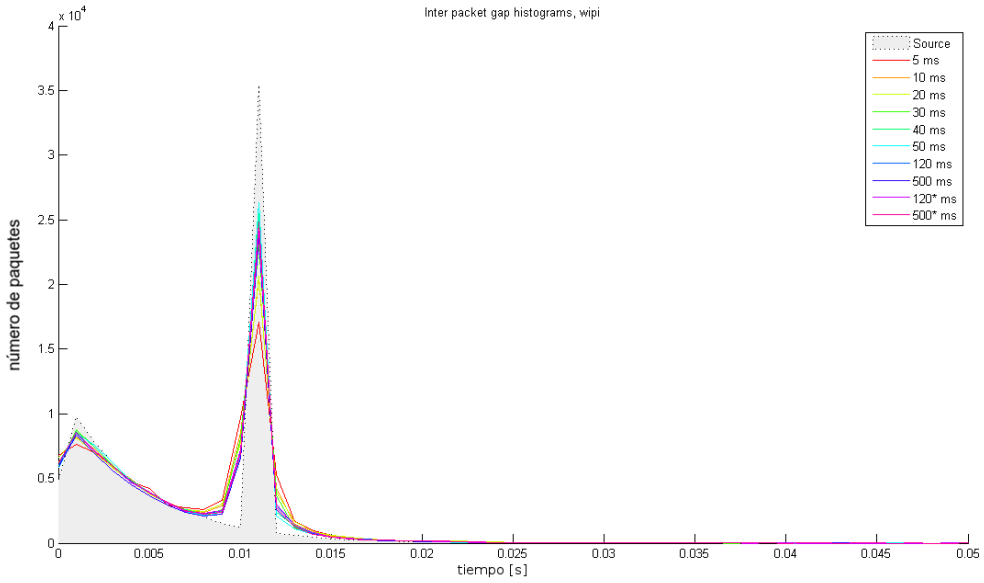


Figura 3.10: Histogramas del tiempo entre paquetes, adaptador WiPi

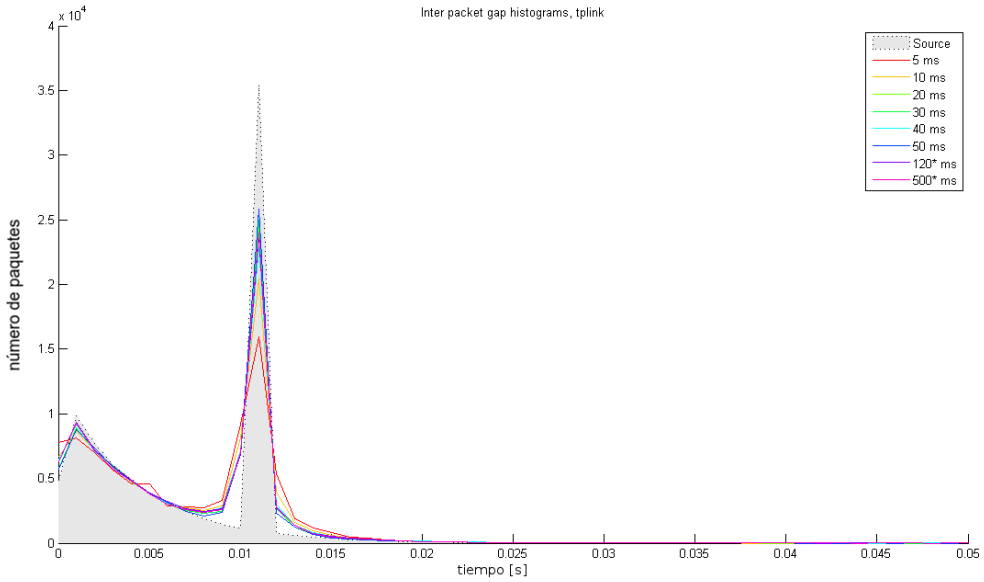


Figura 3.11: Histogramas del tiempo entre paquetes, adaptador TL-WN722N

En la tabla 3.8 podemos ver el MOS estimado para cada experimento. Queda patente el hecho de que el adaptador Linksys no podría dar una buena calidad, salvo en el caso de $BI = 50$ [ms], y usando el mecanismo de *handoff* con ráfaga, superando en estos casos el valor de 2 para el MOS. En los adaptadores WiPi y TP-Link se alcanzan y superan los valores de 4 en la mayoría de casos, lo cual serían valores más que suficiente para una experiencia de juego fluida.

Tabla 3.8: Estimación de la calidad dependiendo del retardo en el segundo experimento.

BI [ms]	**Estimador G-Model**		
	Linksys	WiPi	TP-Link
5	0,27	4,19	4,06
10	1,10	4,23	4,16
20	1,15	4,17	4,25
30	1,21	4,28	4,25
40	1,37	4,26	3,89
50	2,72	4,28	4,26
120	1,80	4,27	-
500	1,74	4,15	-
120-10	2,07	4,26	4,14
500-10	2,42	4,23	3,70

Tras realizar los experimentos, se concluye que esta solución de *seamless handoff* funciona de forma notable para los adaptadores WiPi y TP-Link, mientras que el adaptador Linksys tiene un comportamiento pobre para servicios de tiempo real, aunque no para otro tipo de servicios. Se ha comprobado que el mecanismo de *seamless handoff* con ráfaga funciona correctamente, y la limitación de *beacon interval* no afecta de forma perceptible a los adaptadores inalámbricos probados.

CAPÍTULO 4

Conclusiones y líneas futuras

4.1. Conclusiones

En este trabajo, se ha estudiado el funcionamiento de una plataforma que da servicio de red inalámbrico y que esta siendo desarrollada por un consorcio europeo. Esta se basa en SDN, una tecnología que se encuentra en crecimiento. La plataforma podrá como red auxiliar de las redes móviles tradicionales, como red comunitaria o como una red de tipo corporativo. Los servicios de tiempo real serán unos de los mas utilizados y se tratará de un entorno en el que características como la movilidad, la monitorización y el uso eficiente de los recursos son cruciales.

Se ha estudiado el comportamiento del protocolo OpenFlow así como el protocolo interno que usa la red Odin, adquiriendo conocimientos sobre estos y analizando su funcionamiento dentro del código. Se ha generado una aplicación a modo de ejemplificar el concepto de aplicación de red, demostrando el potencial que este concepto tiene de cara a la automatización de tareas de administración de la red.

Además se ha analizado el funcionamiento de la abstracción LVAP con detalle, minimizando los efectos nocivos que puede tener en la red el uso de esta abstracción de cara a la escalabilidad del sistema, en caso de no estar correctamente configurada. Se ha llegado a la conclusión de que es una elección válida para lograr *seamless handoffs* de forma satisfactoria.

Para la consecución de los objetivos se han realizado modificaciones en los *scripts* ya existentes de configuración, compilación y ejecución de la plataforma para introducir mejoras y funcionalidades. Además, se han creado nuevos *scripts* y se ha desarrollado una estructura para el análisis sistemático de los datos obtenidos.

Se ha minimizado el tiempo de *handoff* mencionado previamente, implementando una ráfaga de *beacons* y manteniendo el resto del tiempo un intervalo entre *beacons* alto, para mejorar la escalabilidad de la plataforma. Además se han estudiado varios adaptadores de red diferentes para evitar posibles errores de interpretación a causa de una visión concreta del problema con lo que se habría perdido una visión global. Los resultados obtenidos han sido satisfactorios puesto que los valores de retardo, *jitter*, pérdidas han sido bajos, por lo que los resultados del MOS han sido en última instancia altos para configuraciones escalables como la de $BI = 120ms, BI_{burst} = 10ms$, superiores a 4 en el caso de dos de los tres adaptadores analizados, y valores de 4,262 en el caso de WiPi y 4,148 en el caso de TP-Link, ambos con

unas pérdidas inferiores al 1 %, mientras que el adaptador Linksys obtenía un MOS de 2,072 y pérdidas del 0.549 %. Esto podría deberse a la existencia de un *buffer* y de su antigüedad (data de 2008). En el mejor de los casos, se han conseguido tiempos de *handoff* de alrededor de 20 ms. Todo esto es consecuencia del buen funcionamiento del sistema debido a su optimización y a las características de cada adaptador. Esta ha consistido en la adecuación de parámetros y la reducción de la carga de computación de los APs por el exceso de tráfico de control.

Todas estas conclusiones serán tomadas en cuenta para el desarrollo del proyecto Wi-5 y el esfuerzo realizado en este TFG será asimilado por dicho proyecto por sus integrantes. Además, se han usado los resultados de este TFG para la publicación de varios documentos científicos.

4.2. Recomendaciones y Trabajo Futuro

Todos los objetivos planteados inicialmente se han cumplido, sin embargo existen ciertas líneas de trabajo que pueden ser interesantes continuando con algunas de las empezadas en este trabajo:

- Para el trabajo futuro, se propone implementar la **reducción de las retransmisiones** de forma satisfactoria y estudiar su impacto en la red inalámbrica.
- Para facilitar las tareas de desarrollo se plantea **aumentar las funcionalidades del modo debug** de la plataforma
- En la versión actual del firmware se pueden desarrollar más funcionalidades en la **monitorización y generar una base de datos** a partir de todas las estadísticas recolectadas en la red.
- Asimismo, es posible probar la **función de *scanning*** que ya se encuentra en fase de desarrollo.
- Una vez implementadas y estudiadas, las funciones de **monitorización y *scanning* podrían integrarse** en un solo método, en la que cada AP envíe de forma periódica información al controlador del estado de la red.
- Para hacer más sencilla la **administración de la plataforma**, se plantea el desarrollo de una **aplicación global** para la gestionarla de forma eficaz y permitir entradas manuales en tiempo real para, por ejemplo, la gestión de movilidad y la distribución de los recursos en los APs.
- **Ampliar el estudio a más dispositivos** inalámbricos de diferentes fábricantes a los ya usados, y aumentar el espectro de modelos usados. Para poder medir correctamente los datos lo optimo sería el diseño de un *testbed* en el que se pudiera **capturar en el interfaz radio con cierta fiabilidad**, usando adaptadores de mejor calidad, y/o reduciendo la interferencia.
- Solucionar el problema de los **paquetes duplicados** observado en alguno de los experimentos.

Bibliografía

[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». En: *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)* (feb. de 2011), págs. 1-433. DOI: 10.1109/IEEESTD.2011.5716530 (cita en pág. 1).

[2] Wi-5 integrants. *Wi-5 use cases and requirements*. Dic. de 2015. URL: http://www.wi5.eu/wp-content/uploads/2015/02/D2_3-Use-cases-and-requirements-final.pdf (visitado 19-09-2016) (citas en págs. 2, 6).

[3] Arunesh Mishra, Minho Shin y William Arbaugh. «An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process». En: *SIGCOMM Comput. Commun. Rev.* 33.2 (abr. de 2003), págs. 93-102. ISSN: 0146-4833. DOI: 10.1145/956981.956990. URL: http://doi.acm.org/10.1145/956981.956990 (cita en pág. 3).

[4] Y. Grunenberger y F. Rousseau. «Virtual Access Points for Transparent Mobility in Wireless LANs». En: *2010 IEEE Wireless Communication and Networking Conference*. Abr. de 2010, págs. 1-6. DOI: 10.1109/WCNC.2010.5506709 (cita en pág. 3).

[5] Julius Schulz-Zander y col. «Programmatic Orchestration of WiFi Networks». En: *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, jun. de 2014, págs. 347-358. ISBN: 978-1-931971-10-2. URL: https://www.usenix.org/conference/atc14/technical-sessions/presentation/schulz-zandery (citas en págs. 3, 10).

[6] Martin Casado y col. «Rethinking Packet Forwarding Hardware.» En: *HotNets*. Citeseer. 2008, págs. 1-6 (cita en pág. 7).

[7] Nick McKeown y col. «OpenFlow: Enabling Innovation in Campus Networks». En: *SIGCOMM Comput. Commun. Rev.* 38.2 (mar. de 2008), págs. 69-74. ISSN: 0146-4833. DOI: 10.1145/1355734.1355746. URL: http://doi.acm.org/10.1145/1355734.1355746 (cita en pág. 7).

[8] *Floodlight Documentation - Floodlight Controller - Project Floodlight*. URL: https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+Documentation (visitado 21-09-2016) (cita en pág. 8).

[9] Eddie Kohler y col. «The Click Modular Router». En: *ACM Trans. Comput. Syst.* 18.3 (ago. de 2000), págs. 263-297. ISSN: 0734-2071. DOI: 10.1145/354871.354874. URL: http://doi.acm.org/10.1145/354871.354874 (visitado 19-09-2016) (cita en pág. 9).

[10] Eddie Kohler. «The Click Modular Router». Tesis doct. Massachusetts Institute of Technology, 2000 (cita en pág. 9).

[11] Alan Holt y Chi-Yu Huang. «OpenWRT». En: *Embedded Operating Systems*. Springer Science & Business Media, 2014, págs. 161-181. DOI: 10.1007/978-1-4471-6603-0_8. URL: http://dx.doi.org/10.1007/978-1-4471-6603-0_8 (cita en pág. 9).

[12] Lalith Suresh y col. «Towards Programmable Enterprise WLANS with Odin». En: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. HotSDN '12. Helsinki, Finland: ACM, 2012, págs. 115-120. ISBN: 978-1-4503-1477-0. DOI: 10.1145/2342441.2342465. URL: http://doi.acm.org/10.1145/2342441.2342465 (cita en pág. 10).

[13] Lalith Suresh Puthalath. «Programming the enterprise WLAN: an SDN approach». Tesis doct. Instituto Superior Técnico, 2012 (cita en pág. 10).

[14] JF Kurose y KW Ross. *Computer Networking. A Top-Down Approach; Harlow (UK)*. Pearson, 2013 (citas en págs. 11, 12).

[15] Sebastian Zander y Grenville Armitage. «Empirically measuring the QoS sensitivity of interactive online game players». En: *Proc. ATNAC*. 2004, págs. 511-518 (citas en págs. 11, 12).

[16] A. Nafaa, T. Taleb y L. Murphy. «Forward error correction strategies for media streaming over wireless networks». En: *IEEE Communications Magazine* 46.1 (ene. de 2008), págs. 72-79. ISSN: 0163-6804. DOI: 10.1109/MCOM.2008.4427233 (cita en pág. 11).

[17] Jon Postel. *Transmission control protocol*. RFC 793. RFC Editor, 1981. DOI: 10.17487/RFC0793. URL: http://www.rfc-base.org/txt/rfc-793.txt (cita en pág. 12).

[18] H. Oouch y col. «Study on appropriate voice data length of IP packets for VoIP network adjustment». En: *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*. Vol. 2. Nov. de 2002, 1618-1622 vol.2. DOI: 10.1109/GLOCOM.2002.1188471 (cita en pág. 12).

[19] Kun I Park. *QoS in packet networks*. Vol. 779. Springer Science & Business Media, 2004 (cita en pág. 12).

[20] J. Saldana y col. «The Effect of TCP Variants on the Coexistence of MMORPG and Best-Effort Traffic». En: *2012 21st International Conference on Computer Communications and Networks (ICCCN)*. Jul. de 2012, págs. 1-5. DOI: 10.1109/ICCCN.2012.6289245 (cita en pág. 12).

[21] A. F. Wattimena y col. «Predicting the Perceived Quality of a First Person Shooter: The Quake IV G-model». En: *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games*. NetGames '06. New York, NY, USA: ACM, 2006. ISBN: 978-1-59593-589-2. DOI: 10.1145/1230040.1230052. URL: http://doi.acm.org/10.1145/1230040.1230052 (visitado 19-09-2016) (citas en págs. 13, 28).

[22] Alessio Botta, Alberto Dainotti y Antonio Pescapé. «A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios». En: *Comput. Netw.* 56.15 (oct. de 2012), págs. 3531-3547. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2012.02.019. URL: http://dx.doi.org/10.1016/j.comnet.2012.02.019 (visitado 19-09-2016) (citas en págs. 14, 27).

[23] A. Kaiser y col. «On the Objective Evaluation of Real-Time Networked Games». En: *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. Nov. de 2009, págs. 1-5. DOI: 10.1109/GLOCOM.2009.5426032 (cita en pág. 25).

Acrónimos

AP	*Acces Point*. 1–3, 6, 9–11, 16, 19, 22–30, 40, 47
CeNITEQ	*Communications Networks and Information Technologies for e-Health and Quality of Experience*. 1
CSA	*Channel Switch Announcement*. 19, 21, 47
CSV	*Comma-separated Values*. 46
DHCP	*Dynamic Host Configuration Protocol*. 16, 18, 27, 30, 46
DIEC	Departamento de Ingeniería Electrónica y Comunicaciones. 1
D-ITG	*Distributed Internet Traffic Generator*. 14, 18, 22, 27, 28, 32, 45, 46
FPS	*First Person Shooter*. 12, 25
GTC	Grupo de Tecnología de las Comunicaciones. 1
IEEE	*Institute of Electrical and Electronics Engineers*. 1, 2
IP	*Internet Protocol*. 30
ISM	*Industrial, Scientific and Medical*. 2
ITU	*International Telecommunication Union*. 2
JVM	*Java Virtual Machine*. 9, 15
LVAP	*Light Virtual Access Point*. 3, 10, 11, 16, 17, 19, 23, 29, 39, 46
MIMO	*Multile Input Multiple Output*. 2
MOS	*Mean Opinion Score*. 13, 29, 38–40
MU-MIMO	*Multi User MIMO*. 2
OFDM	*Ortogonal Frequency Division Multiplexing*. 2
OSPF	*Open Shortest Path First* . 7
OVS	*Open Virtual Switch*. 9, 23

POSIX	*Portable Operating System Interface*. 13
RIP	*Router Information Protocol*. 7
SDN	*Software Defined Network*. 3, 7, 8, 10, 23, 24, 39
SSH	*Secure Shell*. 27
SSID	*Service Set Identifier*. 3, 17
STA	*Station*. 10, 11, 16, 19, 21–25, 27–30, 35, 47
TCP	*Transmission Control Protocol*. 12, 21, 28, 29
TFG	Trabajo de Fin de Grado. 1, 3, 4, 9–11, 13–15, 25, 26, 40
UDP	*User Datagram Protocol*. 12, 14, 22
VoIP	*Voice over IP*. 12–14, 46
Wi-5	*What to do With the Wi-Fi Wild West*. 1–3, 5, 6, 9, 10, 15, 25, 26, 40

Lista de Figuras

Lista de Tablas

ANEXO A

Preparaciones previas a las pruebas

Antes de efectuar las pruebas, se ha de realizar una preparación previa que detallo a continuación. Lo primero que se ha de hacer es compilar el código fuente para los diferentes elementos del sistema. En el caso del controlador esto no es nada complejo, ya que se usa la herramienta Ant[1], que es similar a *Make* solo que está escrito en Java. Esta herramienta nos facilita las tareas de compilación, solo hace falta lanzar dos instrucciones (ver Código A.1) para compilar todo el controlador.

Código A.1: Compilado del paquete *OdinController* con Ant

```
#!/bin/bash
sudo ant clean
sudo ant
```

Por otro lado, habría que compilar los archivos binarios que harán funcionar el agente, en caso de ser necesario. Para esto, hace falta realizar compilación cruzada o *cross-compile*, debido a que los agentes llevan un sistema limitado que no puede soportar un compilador de C, debido a la limitación de espacio y a sus prestaciones. Esta es una técnica necesaria normalmente para sistemas embebidos. Para la compilación cruzada se usa el *miniPC2*, en el que se compila el software click. La mayoría de las veces solo ha habido que modificar *odinagent.cc* y *odinagent.hh*, que forman el módulo de Click para integrar al software dentro de la infraestructura Odin. Se usa el siguiente script, para facilitar la repetición de la tarea.

Lo siguiente que se debe hacer es iniciar el software de captura en los nodos de la red donde se haya decidido sondear el tráfico de la red.

- Si la interfaz que se captura es cableada, lo único que habrá que tener en cuenta es es posible que haya picos de alta cantidad de tráfico, por lo que para mejorar el rendimiento del software de captura se limita el tamaño del *buffer* a 50 *bytes*, que es el tamaño mínimo requerido para poder obtener el número de secuencia incluida por la aplicación D-ITG. De esta forma se minimiza la posibilidades de errores de medida.

- Si la interfaz que se captura es inalámbrica habrá que configurar la interfaz primero en modo monitor, cambiar el canal al que se desee sondear, y comprobar que la interfaz está activa. Después el software debería funcionar bien, aunque existen problemas de saturación que a pesar de disminuir el tamaño del *buffer* no se solventan, y otra cosa que

[1] http://ant.apache.org/index.html

sucede es que si se esta capturando desde la misma maquina por dos canales distintos, usando dos adaptadores, se pierden prestaciones respecto a usar solo uno.

Una vez se han puesto en marcha los puntos de captura, se puede iniciar el sistema Odin. Primero es recomendable iniciar el controlador, y acto seguido se deben iniciar los agentes. En las aplicaciones de red que se han usado existía un tiempo de *setup* en el que después haber iniciado el controlador daba un margen para iniciar todos los agentes, en el caso de las últimas versiones era de 30 segundos, para que diera tiempo a iniciar el resto de elementos. También se ha de iniciar el servidor DHCP si no estaba en marcha y el servidor auxiliar.

En los agentes se tiene que haber introducido previamente la MAC de la máquina que se va a usar para la prueba, así como en el controlador, esto es necesario debido a que se debe crear el LVAP previamente, para evitar posibles intrusiones en el sistema. Cualquier MAC que no esté introducida no funciona, y en el caso de las últimas pruebas solo se usaba una MAC a la vez. Esta característica es opcional, y configurable desde el código, aunque por el momento no se puede cambiar una vez lanzado el controlador.

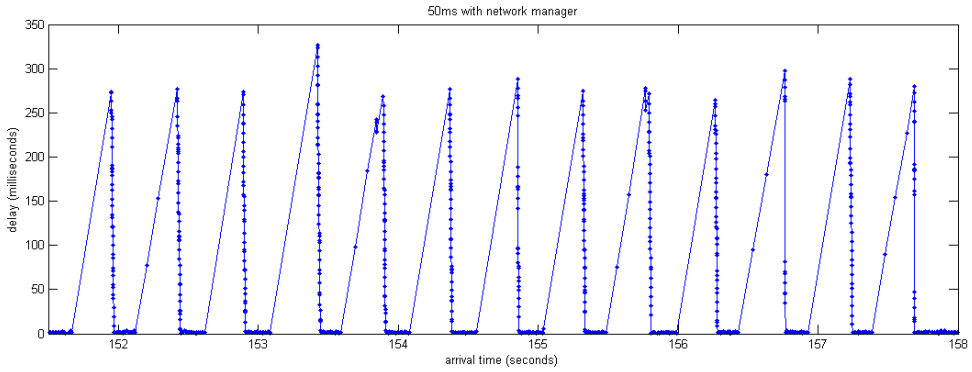


Figura A.1: Detalle del retardo: Picos causados por el escaneo activo del cliente

Una vez hecho esto, se conecta el móvil. Para obtener mejores resultados, es conveniente inhabilitar el *network manager*, dado que sino periódicamente puede darse el patrón de retardo que se observa en la Figura A.1, derivada de hacer escaneo activo por parte del móvil. Una vez se ha obtenido satisfactoriamente una dirección IP, se puede empezar a generar tráfico. Se han usado varias aplicaciones de generado de tráfico, en concreto *Ping*, D-ITG e Iperf. D-ITG permite generar tráfico más específico que *Ping* o Iperf, dado que tiene opción de simular tráfico por ejemplo de Quake3 o VoIP con diferentes parámetros de forma precisa. Además, de forma meramente cualitativa, se ha probado con aplicaciones reales, pero sin hacer mediciones cuantitativas de las estadísticas en este caso.

Cuando se desee, se desconecta el móvil de Odin, y se reinician tanto el controlador como los agentes. Una vez hecho esto, se cortan los procesos de captura, y ya se puede abrir el archivo de salida para su análisis, para lo que se ha utilizado *Wireshark*. En este programa es posible crear gráficas con diferentes parámetros de entrada, que sirven para tener una visión rápida de los resultados, sin embargo para realizar un análisis exhaustivo se debe proceder a la exportación de los datos filtrados. En este caso, normalmente se usan los datos producidos por la aplicación de generado de tráfico, por lo que primero se filtran y se exportan a formato *Comma-separated Values* (CSV). Una vez hecho esto, usaremos el *script* de *MATLAB* (ver Anexo D usado para procesar los datos y generar las estadísticas derivadas de las medidas.

ANEXO B

Diagrama de secuencia del *handoff*

La Figura B.1 es un diagrama de secuencia del procedimiento de *handoff* entre APs de forma esquemática. No se han incluido posibles pérdidas de paquetes, en cuyo caso existirían retransmisiones. En el caso de perder un *beacon* CSA y todos las retransmisiones no pasa nada, sino que el AP continúa con el siguiente. El único problema viene si se pierde el último CSA.El AP 2 empieza a enviar beacons en cuanto recibe la orden del controlador, aunque el STA continúe asociado al otro AP por lo que se asegura su correcta recepción en el canal.

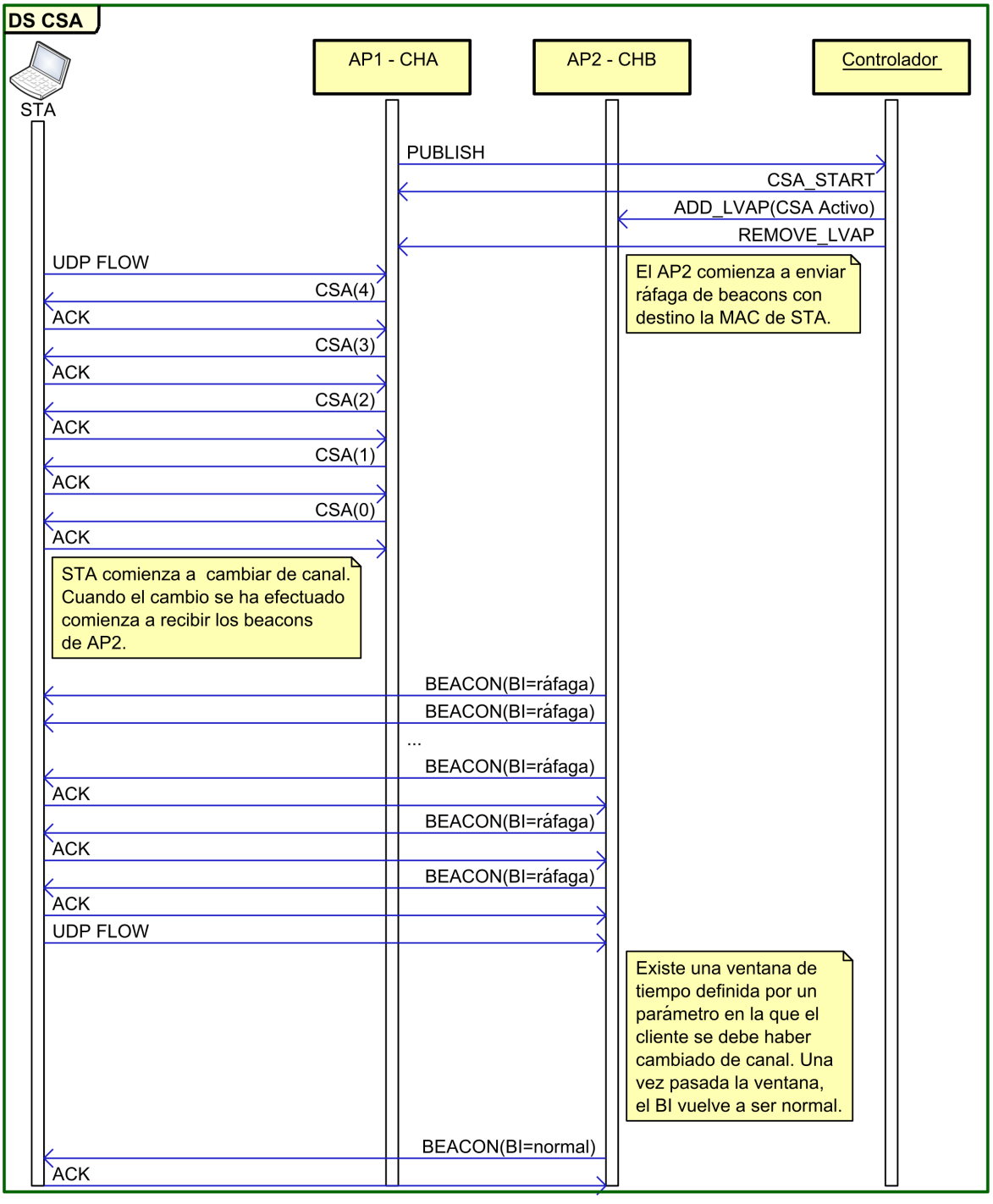


Figura B.1: Diagrama de secuencia del cambio de canal

ANEXO C

Script csv_parser.py

csv_parser.py

```
#prefix: 0{7}10{3} (for 5 hex of seq number, ensure there is no more)
#suffix: .{32}\.\.\. or filter capture to 50 Bytes (Ethernet)
import re
import sys
from os import remove,  close
from tempfile import mkstemp
from shutil import move

fh, abs_path = mkstemp()
file_path = sys.argv[1]
prefix = re.compile(r"(0{7}10{3})")
#suffix = re.compile(r"(.{32}\.\.\.)")
quotation = re.compile(r'"')
#decimal = re.compile(r'\d,\d')
#test = '"10216",
#                        "121.925291","192.168.1.216",
#                        "192.168.1.130","UDP","109",
#                        "39635  >  8999  Len=67","",
#                        "","","1466613771.482540000",
#                        "","","0000000100000001"'
#print(test)
with open(file_path,  'r') as old_file:
    with open(abs_path, 'w') as new_file:
        for line in old_file:
            line=re.sub(prefix,'',  line)
            line=re.sub(quotation,'',  line)
            new_file.write(line)
close(fh)
remove(file_path)
move(abs_path, file_path)
#print(a)
#print(b)
```

ANEXO D

Scripts de Matlab

main.m

```
%% main.m
% This script is used to generate the charts

clearvars;
close all;

%% Launcher - statistics accumulator
%Initialize vars
stat_noplot = false;

stat_size = 8; %Basic BI, 6+2 for tplink and 8+2 for linksys and wipi
stat_special_size = 2; %Improved BI
stat_colors = hsv(stat_size);
stat_markers = {'+','o','*','.','x','s','d','^','v','>','<','p','h'};


set(0, 'DefaultAxesFontName', 'Arial')
figure(100);
hold on;
figure(101);
hold on;

stat_lost_packets=zeros(stat_size,1);
stat_handoff_time=zeros(stat_size,1); %ms
stat_error_lost_packets=zeros(stat_size,2);
stat_error_handoff_time=zeros(stat_size,2);
stat_confidence_percentage = 0.95;
stat_maxY = 400;
stat_cat = { '5ms' '10ms' '20ms' '30ms' '40ms' '50ms'...
             '120to10ms' '500to10ms'};

stat_addr = ['.\ScriptQuake_v4.m'];

stat_ipg_nolost = zeros(stat_size,1);

%Output data
% stat_index = {};
stat_total_lost = zeros(stat_size,1);
stat_total_dups = stat_total_lost;
stat_mean_delay = stat_total_lost;
```

```
stat_jitter = stat_total_lost;
stat_MOS = stat_total_lost;
stat_P=1;

% Handoff detection Thresholds
stat_th_losses = 1;
% stat_th_time = (5/(3*1000))*[5 10 20 30 40 50 120 500 10 10];
stat_th_time = ones(stat_size,1)*0.08; %tplink
% stat_th_time = zeros(stat_size,1);

%loop
for stat_i=1:stat_size
    addrT = ['..\' stat_cat{stat_i} '\client_sent.csv'];
    dataT=importNparse(addrT);
    addrR = ['..\' stat_cat{stat_i} '\server_received.csv'];
    dataR=importNparse(addrR);
%     addr = ['..\' stat_cat{stat_i} '\ScriptQuake_v3.m'];
    run(stat_addr)

    stat_lost_packets(stat_i,1)=mean(lost_pk(:,2));
    stat_error_lost_packets(stat_i,:)=confidenceIntervals(lost_pk(:,2),...
        stat_confidence_percentage);

    stat_handoff_time(stat_i,1)=1000*mean(handoff_time);
    stat_error_handoff_time(stat_i,:)=confidenceIntervals(1000*handoff_time,...
        stat_confidence_percentage);

    if (stat_i==2 || stat_i==6 || stat_i > stat_size-(stat_special_size+1))
        v=genvarname(['stat_' stat_cat{stat_i} '_time_delay']);
        eval([v '= time_delay;'])

        v=genvarname(['stat_' stat_cat{stat_i} '_delay_nolost']);
        eval([v '= delay_nolost;'])

        v=genvarname(['stat_' stat_cat{stat_i} '_src_time']);
        eval([v '= normTime(:,1);'])
        v=genvarname(['stat_' stat_cat{stat_i} '_delay']);
        eval([v '= delay;'])

        v=genvarname(['stat_' stat_cat{stat_i} '_handoff_time']);
        eval([v '= handoff_time;'])
        v=genvarname(['stat_' stat_cat{stat_i} '_lost_pk']);
        eval([v '= lost_pk;'])
    end
    figure; stem(lost_pk(:,1),lost_pk(:,2));
    title(['lost packets, BI=' stat_cat{stat_i}]);
    figure; stem(lost_pk(:,1),1000*handoff_time);
    title(['handoff time, BI=' stat_cat{stat_i}]);
%     figure;hist(handoff_time,40);
%     title(['handoff time histogram, BI=' stat_cat{stat_i}]);
%     %Twenty largest ipg values

    %Inter packet gap histograms
    figure;
    stat_ipg_nolost(stat_P) = max(diff(normTime(:,2)));
    [N1,X1]=hist(diff(normTime(:,1)),[0.001:1e-3:0.53]);
    [N2,X2]=hist(diff(normSortTime),[0.001:1e-3:0.53]);
    [N3,X3]=hist(handoff_time,[0.001:1e-3:0.53]);
```

```
99      max_seq=dataT(end,3);
100     area(X1,N1);
101     hold on;
102     plot(X2,N2,'Color','g','LineWidth',0.8);
103     plot(X3,N3,'Color','r','LineStyle','--','LineWidth',0.8);
104     title(['inter packet gap histogram, BI=' stat_cat{stat_i}]);
105     hold off;
106
107     %Normalized histogram of IPG
108     figure(100)
109     if (stat_P==1)
110         area(X1,N1./max_seq);
111     end
112     plot(X2,N2./max_seq,'Color',stat_colors(stat_P,:),...
113         'LineStyle','-','Marker',stat_markers{stat_P});
114     title('inter packet gap histogram, normalized');
115
116     % Histogram of IPG
117     figure(101)
118     if (stat_P==1)
119         area(X1,N1);
120     end
121     plot(X2,N2,'Color',stat_colors(stat_P,:),...
122         'LineStyle','-','Marker',stat_markers{stat_P});
123     title('inter packet gap histogram, absolute');
124 %     figure(101);
125 %     hist(diff(normSortTime),[0:1e-3:0.05 0.1]);
126
127 %     stat_index(stat_P) = stat_cat{stat_i};
128     stat_total_lost(stat_P) = packets_lost;
129     stat_total_dups(stat_P) = duplicate;
130     stat_mean_delay(stat_P) = mean_delay;
131     stat_jitter(stat_P) = jitter;
132     stat_MOS(stat_P) = MOS_model;
133     stat_P = stat_P + 1;
134
135 %     pause()
136     clearvars -except stat*
137 %     close(1:3);
138 end
139
140 cd ..\summary
141
142 %% Compare mean packets lost on handoff and mean handoff time
143 %Packets
144 figure;
145 xdata = 1:stat_size;
146 bar(xdata,stat_lost_packets,'g');
147 set(gca, 'Xtick', xdata, 'XtickLabel', stat_cat);
148 hold on
149 errorbar(xdata,stat_lost_packets,stat_lost_packets -...
150     stat_error_lost_packets(:,1),stat_error_lost_packets(:,2)...
151     -stat_lost_packets,'k','LineStyle','none');
152 set(gca,'Ygrid','on');
153 xlabel 'Beacon Interval [ms]'
154 ylabel 'Lost packets'
155 title('Average packets lost in the handoff');
156 legend('Number of packets');
```

```
157 hold off
158
159 %Time
160 figure;
161 bar(xdata,stat_handoff_time,'g');
162 set(gca, 'Xtick', xdata, 'XtickLabel', stat_cat(1:6));
163 hold on
164 errorbar(xdata,stat_handoff_time,stat_handoff_time -...
165     stat_error_handoff_time(:,1),stat_error_handoff_time(:,2)...
166     -stat_handoff_time,'k','LineStyle','none');
167 set(gca,'Ygrid','on');
168 xlabel 'Beacon Interval [ms]'
169 ylabel 'Handoff Time [ms]'
170 title('Average handoff time');
171 legend('Handoff time');
172 hold off
173
174 %% Compare handoff time
175
176 % 10 ms, constant beacon interval
177 figure;
178 % subplot(3,1,1);
179
180 stem(stat_10ms_src_time,stat_10ms_delay,'-b.');
181
182 % axis([353 383.3 0 20]);
183 xlabel 'time [s]'
184 ylabel 'delay [ms]'
185 title('End to end delay, 2 handoffs close-up (BI=10ms)');
186 legend('delay');
187 set(gca,'Layer','Top')
188
189 %Break The Axes
190 % h = breakxaxis([353.3 383]);
191 % unbreakxaxis(h)
192
193 % beacon interval being 50 ms
194 % subplot(3,1,2);
195 figure;
196
197 stem(stat_50ms_src_time,stat_50ms_delay,'-b.');
198
199 % axis([425.55 455.85 0 20]);
200 xlabel 'time [s]'
201 ylabel 'delay [ms]'
202 title('End to end delay, 2 handoffs close-up (BI=50ms)');
203 legend('delay');
204 set(gca,'Layer','Top')
205
206 % h = breakxaxis([425.85 455.55]);
207
208 % beacon interval being 120 ms to 10 ms when CSA
209
210 % subplot(3,1,3);
211 figure;
212
213 stem(stat_120to10ms_src_time,stat_120to10ms_delay,'-b.');
214
```

```
% axis([358.3 388.95 0 20]);
xlabel 'time [s]'
ylabel 'delay [ms]'
title('End to end delay , 2 handoffs close-up (BI=120ms, BI_C_S_A=10ms)');
legend('delay');
set(gca,'Layer','Top')

% h = breakxaxis([358.6 388.65]);

%% 8-periode snapshot, numbers

% 10 ms, constant beacon interval
figure;
stem(stat_10ms_src_time,stat_10ms_delay,'-b.');

hold on
stem(stat_10ms_lost_pk(:,1),...
    stat_maxY*ones(size(stat_10ms_lost_pk,1),1),...
    'Marker','none','Color','red');
a = num2str(stat_10ms_lost_pk(:,2));
b = cellstr(a);
c = strtrim(b);
h = text(stat_10ms_lost_pk(:,1),...
    stat_maxY*0.75*ones(size(stat_10ms_lost_pk(:,:),1),1),c);
set(h, 'BackgroundColor', 'white') %Cambia el bg a blanco
set(h, 'Color', 'red');
set(h, 'FontSize', 16);
hold off

% axis([stat_10ms_lost_pk(4,1)-1 stat_10ms_lost_pk(11,1)+30 0 stat_maxY]);
xlabel 'time [s]'
ylabel 'delay [ms]'
title('End to end delay , BI=10');
legend('delay','handoff');
set(gca,'Layer','Top')
```

ScriptQuake_v4.m

```
%% Calculate delay and jitter from Quake3 traffic over wlan
% clear all
% close all

%% Import captured data from wireshark

% dataT=importNparse('./client_sent.csv');
% dataR=importNparse('./server_received.csv');

% dataT_full=importNparse('./client_sent.csv');
% dataR_full=importNparse('./server_received.csv');
%
% dataT=dataT_full(3000:7999,:);
% dataR=dataR_full(3000:7999,:);

t0=dataT(1,2);%*ones(size(dataT,1),1);
%refDst=dataR(1,2)*ones(size(dataT,1),1);

%% Pair the contents of both arrays

srcP=1;
dstP=1;
tempP=0;
results=NaN(size(dataT,1),4);

dataR_sort=sortrows(dataR,3);
duplicate=0;

% for i=2:data


while (srcP<=size(dataT,1)&&dstP<=size(dataR_sort,1))
    if (dataT(srcP,3)==dataR_sort(dstP,3))
        results(dataT(srcP,3)-dataT(1,3)+1,:)=[dataT(srcP,3),...
            dataT(srcP,2),dataR_sort(dstP,2),dataT(srcP,4)];
        %|SeqNumber|EpochSrc|EpochDst|Lenght|
        srcP=srcP+1;
        dstP=dstP+1;
%         sprintf('%d:%d',srcP,dstP)


    elseif (dataT(srcP,3)<dataR_sort(dstP,3))
        while (not(dataT(srcP,3)==dataR_sort(dstP,3)))
            results(dataT(srcP,3)-dataT(1,3)+1,:)=[dataT(srcP,3),...
                dataT(srcP,2),NaN,dataT(srcP,4)];
            %|SeqNumber|EpochSrc|NaN|Lenght| in case of loss
            srcP=srcP+1;
        end
    elseif (dataT(srcP,3)>dataR_sort(dstP,3))
        duplicate=duplicate+1;
        dataR_sort(dstP,:)=[];
    end
end

%% Statistics calculation

```

```
delay_0=0;
j=1;
k=0; %burst counter pointer
reset_burst=true; %losses burst detection
lost_pk=[];
handoff_time=[];
% th_losses = 0; %GLOBAL (main.m)
th_time = 0;
lost_burst_discarded = 0;
for i=1:size(results,1)
    if not(isnan(results(i,3)))
        normTime(i,:)=[results(i,2)-t0,results(i,3)-t0];
        delay(i)=1000*(normTime(i,2)+delay_0-normTime(i,1)); %ms
        delay_nolost(j)=delay(i);
        j=j+1;
        if ~reset_burst
            handoff_time(k)=[normTime(i,2)-last_pk_heard];
        end
        reset_burst=true;
    else
        normTime(i,:)=[results(i,2)-t0,results(i,3)];
        delay(i)=inf;
        if reset_burst
            %descarto las prdidas de "wifi"
            if ((k>0) && ((lost_pk(k,2)<stat_th_losses) || ...
                (handoff_time(k)<stat_th_time(stat_i))))
                lost_pk(k,:)=[0 0];
                lost_burst_discarded = lost_burst_discarded + 1;
                last_pk_heard = normTime(i-1,1);
            else
                k=k+1;
                lost_pk=[lost_pk; 0 0];
                last_pk_heard = normTime(i-1,1);
                % time | number_of_packets
            end
            lost_pk(k,1)=normTime(i,1);
            reset_burst=false;
        end
        lost_pk(k,2)= lost_pk(k,2)+1;

    end
end

%Last iteration
if ((k>0) && ((lost_pk(k,2)<stat_th_losses) || ...
    (handoff_time(k)<stat_th_time(stat_i)))
                lost_pk(k,:)=[0 0];
                lost_burst_discarded = lost_burst_discarded + 1;
                last_pk_heard = normTime(i-1,1);
end

if (lost_pk(end,2)<stat_th_losses)
    lost_pk(end,:)=[];
    handoff_time(end)=[];
end

threshold=500;
indices = find(delay_nolost>threshold);
```

```
115 delay_nolost(indices) = [];
116
117 lim_inf_graf=1;
118 lim_sup_graf=length(delay_nolost);
119 tBase_graf=0;
120
121 mean_delay=mean(delay_nolost(lim_inf_graf:lim_sup_graf)); %splice
122 jitter = sqrt(var(delay_nolost(lim_inf_graf:lim_sup_graf)));
123
124
125 %Percentage lost
126
127 packets_lost=(1-(size(dataR_sort,1)/size(dataT,1)))*100;
128 % Overall lost packets are calculated with the total length of the dataT
129 % matrix, because those are the packets really sent. It should be the
130 % equal to the last sequence number too, if there are no capture
131 % or ethernet transmission problems.
132
133 %Bandwidth and Throughput
134
135 meanBW=sum(dataT(:,4))*8/normTime(end,1);
136 meanTP=sum(dataR_sort(:,4))*8/normTime(end,2);
137
138 normSortTime=sort(normTime(:,2));
139 normSortTime(isnan(normSortTime))=[];
140 time_delay=normSortTime;
141 time_delay(indices) = [];
142
143 %% Data rate
144 intervaloR=0.001; %segundos
145
146 [R,T]=computeRate(intervaloR,normSortTime,dataR_sort(:,4));
147
148 %MA filter - TODO-function
149 n=500; %MA filter length
150 B = 1/n*ones(n,1);
151 R_m= filter(B,1,R);
152
153 %% Mean Opinion Score Estimation
154
155 X = 0.104*mean_delay + jitter;
156 MOS_model = -0.00000587*X^3 + 0.00139*X^2 - 0.114*X + 4.37;
157
158 %% Plotting
159
160 %COMMENT THIS IF NOT USING main.m
161 if ~stat_noplot
162
163     maxY=300;
164     figure; plot(time_delay(lim_inf_graf:lim_sup_graf)-tBase_graf,...
165         delay_nolost(lim_inf_graf:lim_sup_graf),'-b.');
166     figure; plot(normTime(:,1),delay,'-b.');
167     hold on
168     % plot(threshold.*isinf(delay),'r*')
169     stem(lost_pk(:,1),maxY*ones(size(lost_pk,1),1),...
170         'Marker','none',...
171         'Color','red');
172     % I_lost=find(isinf(delay));
```

```
173     % y = [0.1  0.2  -0.1  4.1  -2  1.5  -0.1];
174     % x = 1:1:length(y);
175     % figure; plot(x,y)
176     a = num2str(lost_pk(:,2));
177     b = cellstr(a);
178     c = strtrim(b);
179     h = text(lost_pk(:,1),maxY*0.75*ones(size(lost_pk,1),1),c);
180     set(h, 'BackgroundColor', 'white') %Cambia el bg a blanco
181     set(h, 'Color', 'red');
182     set(h, 'FontSize', 16);
183
184     hold off
185     figure; plot(T,R,'.')
186     % figure; plot(R2);
187 end
188
189 %Final Report
190 sprintf('*******************************\n');
191 disp(ans);
192
193 sprintf('%s',pwd);
194 disp(ans);
195
196 sprintf('Packets sent = %d',size(dataT,1));
197 disp(ans);
198
199 sprintf('%f %%perdidas',packets_lost);
200 disp(ans);
201
202 sprintf('%f %%perdidas handoff',...
203     (sum(lost_pk(:,2))/size(dataT,1))/(packets_lost/100)*100);
204 disp(ans);
205
206 sprintf('Retardo medio = %f ms\n Jitter = %f ms',mean_delay,jitter);
207 disp(ans);
208
209 sprintf('ThroughputIn   = %f kbps\nThroughputOut = %f kbps\n',...
210     meanBW/1000,meanTP/1000);
211 disp(ans);
212
213 sprintf('Mean Opinion Score %f',MOS_model);
214 disp(ans);
215
216 sprintf('*******************************\n');
217 disp(ans);
```

ANEXO E

Fotografías del despliegue

Figura E.1: La arquitectura lista para su uso en la Demo *Ultra-fast Broadband Seminar 2016*, La Haya, Países Bajos

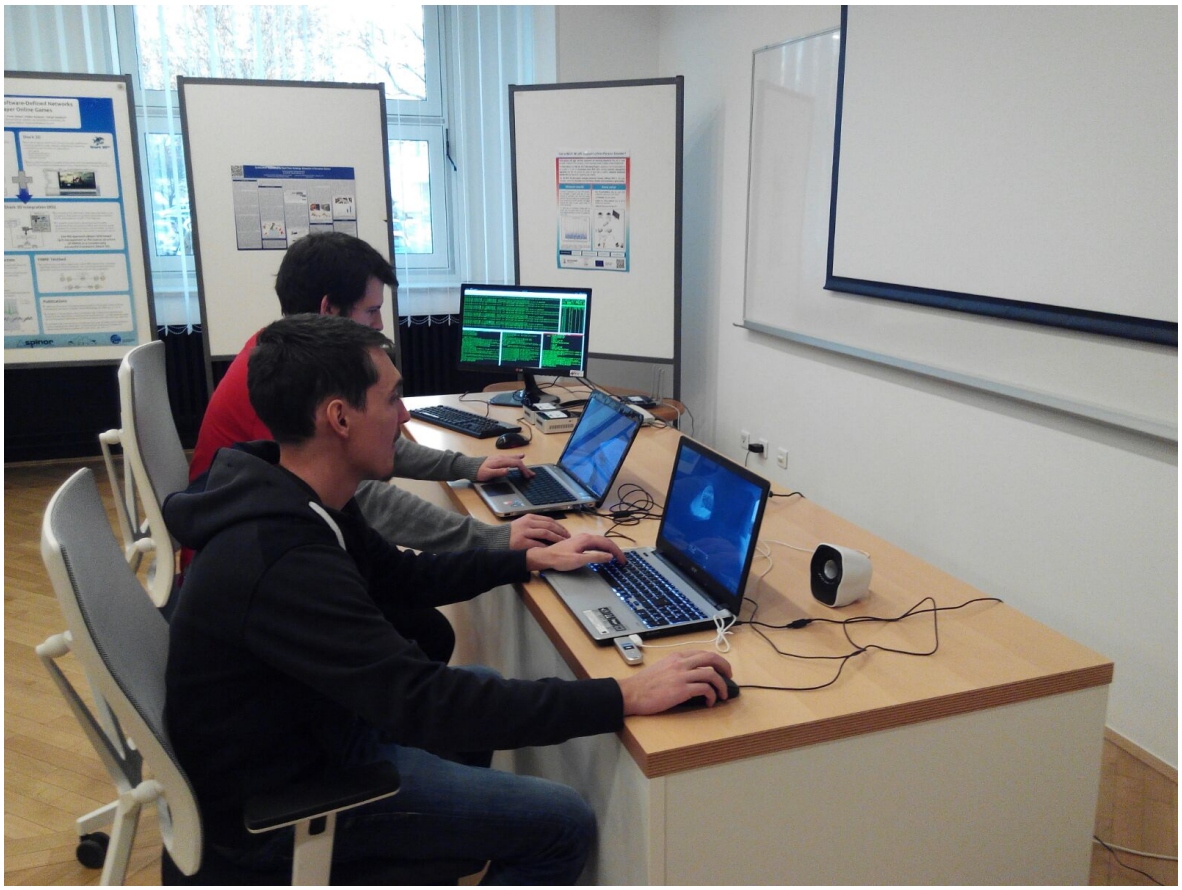


Figura E.2: Demostración durante la *NetGames 2015*, Zagreb, Croacia

Figura E.3: Entorno de trabajo en el laboratorio