

# Red WiFi para apps de tiempo real en un túnel

Sistemas Empotrados Ubicuos

Máster en Ingeniería Informática (2016-2017)

Escuela de Ingeniería y Arquitectura

Marcos Canales Mayo, 467716

## Tabla de contenido

Introducción .....	3
Contexto .....	4
Entorno.....	4
Objetivo.....	4
Problemas.....	4
Proceso de handoff .....	4
Tiempo de handoff.....	6
Finalidad del trabajo.....	7
Preparación de la infraestructura .....	8
Ambos APs por cable.....	11
Resultados .....	12
Wireless Distribution System (WDS) .....	18
Resultados .....	19
Resultados generales .....	21
Problemas encontrados .....	22
Trabajo futuro .....	23
Referencias.....	24
Enlaces de interés .....	25
Anexos .....	26
Anexo I (preparación de la infraestructura).....	26
Anexo II (csvToMat.m) .....	35
Anexo III (customhex2num.m).....	38
Anexo IV (SEU_Stats.m).....	39

## Introducción

Las aplicaciones de tiempo real tienen requerimientos muy estrictos en cuanto a latencias. Cuando esas aplicaciones están conectadas a una red WiFi, es necesario tener en consideración algunos aspectos que pueden degradar la calidad del servicio de la red y, en consecuencia, de la aplicación. Uno de esos aspectos es el proceso de ***roaming o handoff***, a través del cual la estación (en adelante STA) donde está corriendo la aplicación de tiempo real decide desconectarse/desasociarse de un punto de acceso (AP) y reasociarse a otro de la misma red.

El fin de este trabajo es preparar una red en un entorno concreto para proporcionar conectividad a una STA móvil que está ejecutando la aplicación en tiempo real. En este informe se presentan posibles variaciones en la infraestructura para lograr este objetivo, así como un análisis del impacto que tiene el proceso de *handoff* en el rendimiento de la red.

## Contexto

En esta sección se describen algunos factores a tener en cuenta que afectarán al desarrollo del trabajo.

## Entorno

La STA se moverá a lo largo de un tramo de un túnel de 7.7 km de largo, por lo que será necesario fijar dos puntos de acceso en el interior del túnel que le den cobertura WiFi en ese tramo. Además, es necesario que la latencia de los paquetes no supere los **150 ms**, para no degradar la calidad de la aplicación.

## Objetivo

El proceso de *handoff* supone un hito temporal crítico para la corrección de la aplicación. Por lo tanto, el objetivo principal que se persigue es que este proceso se mantenga siempre por debajo de los 150 ms requeridos.

## Problemas

Algunos de los problemas que se deben tener en mente son los siguientes:

- Es la STA la que decide cuándo se debe comenzar el proceso de *handoff*. En algunas ocasiones esto puede suponer un hándicap, ya que la STA no tiene información sobre el estado de la red que sí tienen los APs.
- El tiempo de *handoff* es dependiente del fabricante de la *Wireless Network Interface Controller* (WNIC) y del *software* usado.

## Proceso de *handoff*

El proceso de *handoff* se compone de 3 fases principales, aunque varía ligeramente entre implementaciones de diferentes fabricantes:

1. Una política determinada en la STA (e.g. no recibir ACKs en L2 o perder señal con el AP) dispara el proceso de *handoff*.
2. La STA busca posibles APs que se encuentren dentro de su rango en aquellos canales que se hayan especificado previamente.
3. Dada una política en la STA (e.g. mejor señal recibida) se selecciona uno de los APs de todos los encontrados.
4. La STA se reasocia con el AP escogido.

En la Figura 1 se puede observar el intercambio de tramas en detalle una vez se ha disparado el proceso de *handoff* [1]:

1. La STA envía una trama de sondeo (*probe request*) en forma de *broadcast*.
2. Los APs que reciban esa trama de sondeo responden (*probe response*) a la STA.

- La STA inicia el proceso de reasociación con el AP escogido, intercambiando tramas de desautenticación con el antiguo AP y de reasociación y autenticación con el nuevo AP (el orden de estas tramas es dependiente del fabricante).

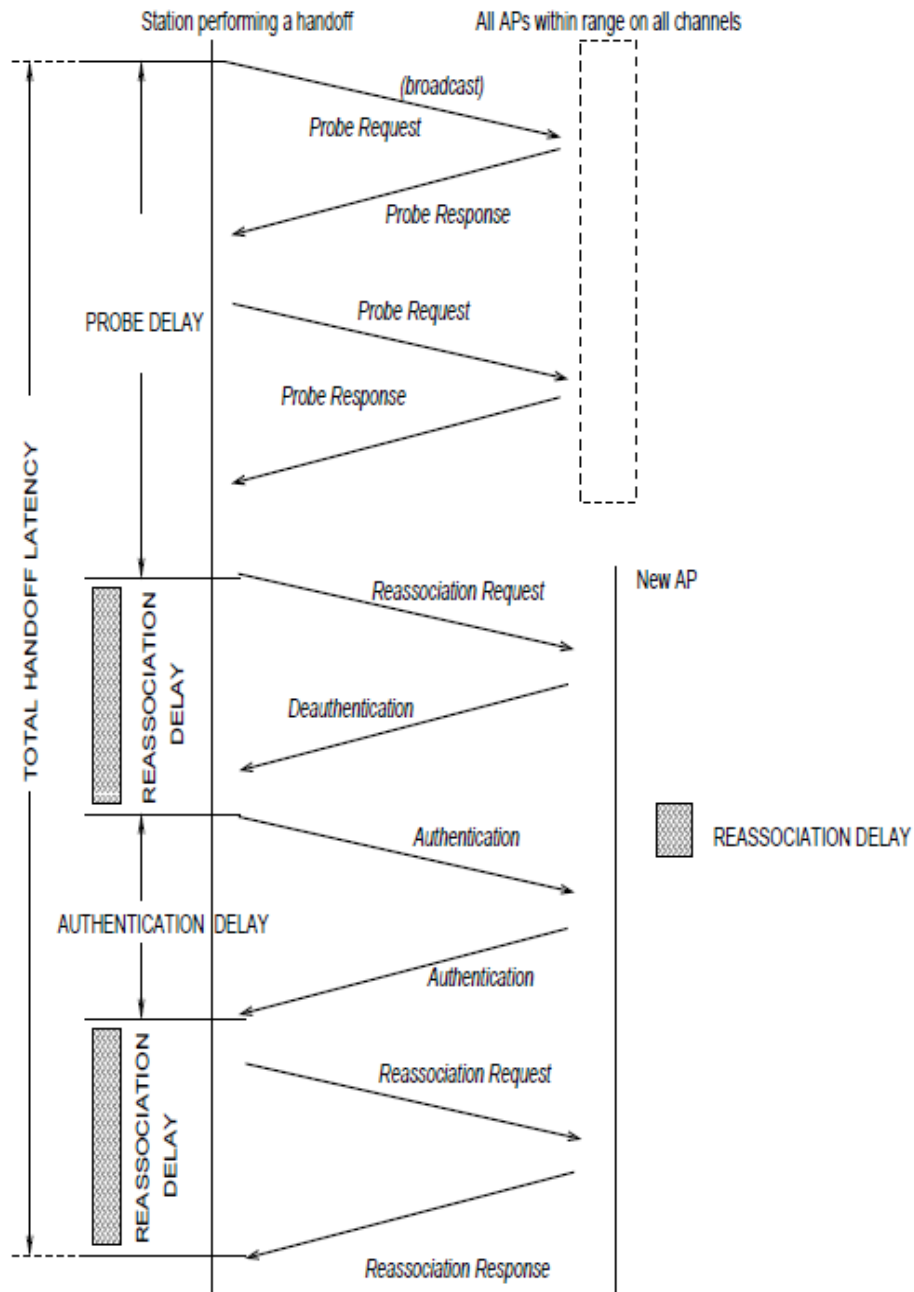


Figura 1. Proceso de handoff

## Tiempo de *handoff*

Como se ha comentado con anterioridad, el tiempo de *handoff* varía en función de la implementación y del *software*. En [1] se presentan los resultados de un estudio empírico de este tiempo, que se resumen en las gráficas mostradas en las Figuras 2 (media de tiempo), 3 (desviación estándar del tiempo) y 4 (diferencia entre la medida máxima y la mínima). Los distintos conjuntos delimitados por un cuadro se corresponden con el fabricante del AP, mientras que las distintas medidas obtenidas (valores en el eje Y) se corresponden cada una con un fabricante de la STA distinto.

Se puede apreciar que el tiempo de *handoff* varía mucho en función de la implementación. Además, para determinados fabricantes la desviación estándar y la diferencia entre máximo y mínimo es bastante alta, lo que hace que este tiempo sea bastante impredecible.

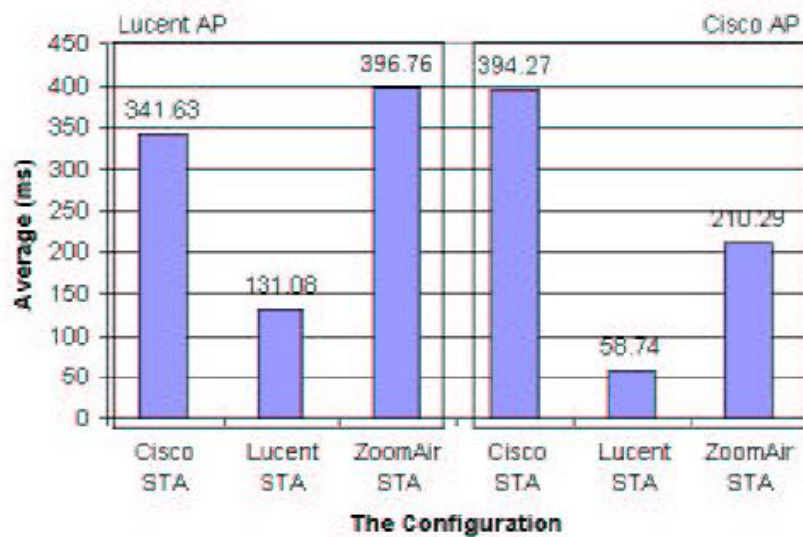


Figura 2. Media de tiempo de *handoff*

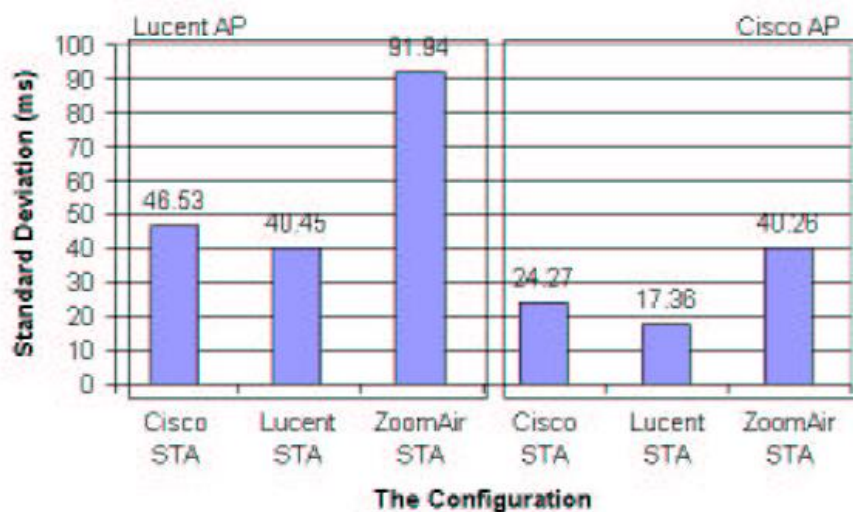


Figura 3. Desviación estándar de tiempo de *handoff*

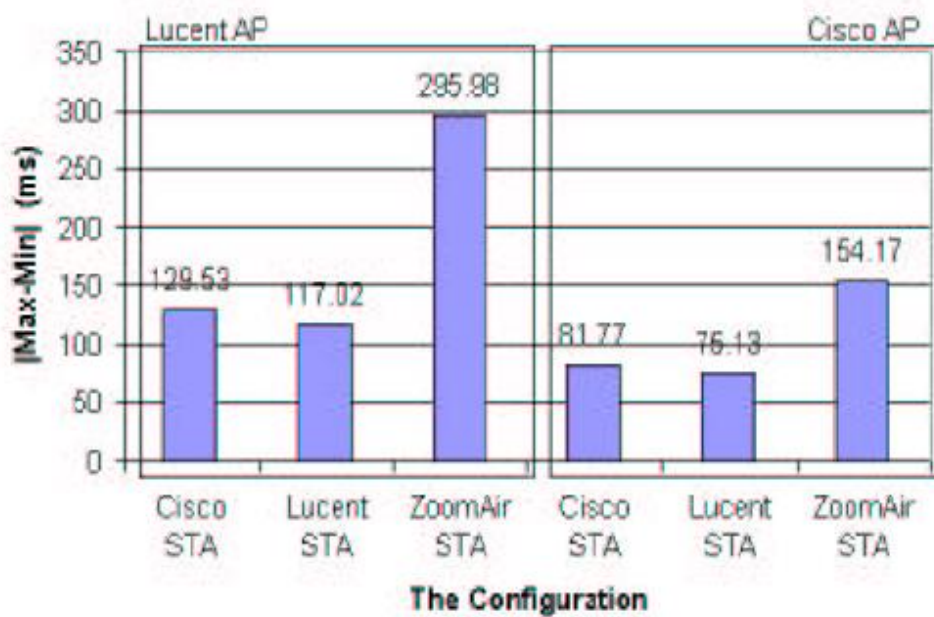


Figura 4. Diferencia entre la medida máxima y la mínima del tiempo de handoff

El experimento fue realizado en la Universidad de Maryland (EE.UU.) y se caracteriza por los siguientes puntos:

- Existen 6 APs por planta del fabricante Lucent dando cobertura WiFi de una red.
- Existen 8 APs por planta del fabricante Cisco dando cobertura WiFi de otra red.
- Se usan los canales 1, 6 y 11 y APs adyacentes están operando en distintos canales, para evitar interferencias.
- Para obtener las medidas anteriormente mostradas en las gráficas, la STA envía de forma periódica (no se especifica el periodo) paquetes ICMP a la red.
- Para capturar el tráfico, se dispone de dos *sniffers* externos pero cercanos a la STA mientras se mueve. Uno de los *sniffers* dispone de una WNIC y el otro dispone de dos WNICs, de manera que cada WNIC está en modo monitor capturando el tráfico de uno de los canales anteriormente comentados.
- Los *sniffers* se sincronizan por el protocolo NTP con una precisión siempre menor de 80  $\mu$ s.
- Por cada combinación de STA – AP, la STA realiza siempre una ruta idéntica de 30 minutos, llegando a realizar entre 15 y 27 *handoffs* (dependiendo de la combinación).

### Finalidad del trabajo

Dados todos estos problemas conocidos a priori que pueden condicionar en gran medida el despliegue de la infraestructura en el entorno real, el trabajo desarrollado debe cumplir los siguientes objetivos:

- Ser capaz de reproducir el problema en un entorno controlado.

- Permitir variar ciertos parámetros (como el umbral de señal para disparar el proceso de *handoff*, canales en los que transmitir, potencia a la que transmitir...) para analizar cómo afectan al rendimiento.
- Analizar el rendimiento en múltiples escenarios:
  - Red con APs conectados por cable.
  - Red con APs en forma de *Wireless Distribution System* (WDS).

## Preparación de la infraestructura

Los distintos elementos que se han decidido usar para montar la infraestructura de red son los siguientes:

- Raspberry Pi 2 Modelo B para los APs, ya que son dispositivos baratos y con suficiente capacidad computacional para resolver el problema que nos ocupa.
  - 900MHz quad-core ARM Cortex-A7 CPU
  - 1GB RAM
- Tarjeta Alfa AWUS051NH v2 como WNIC para los APs y la STA.
  - 802.11 a/b/g/n
- OpenWRT 15.05.01 como sistema operativo en los APs.
  - Distro de Linux muy ligera, enfocada a sistemas empujados para comunicaciones inalámbricas.
  - Soporte para el *hardware* usado, que es Raspberry Pi 2 (otros sistemas operativos no disponen de este soporte).
  - Fácil de personalizar gran variedad de parámetros de comunicaciones inalámbricas (de nuevo, otros sistemas operativos no son tan fáciles de usar y no dejan tanta flexibilidad a la hora de configurar ciertas cosas).
- hostapd
  - Necesario para activar las comunicaciones WiFi.

En la Figura 5 se puede observar el *hardware* usado:

- (Parte izquierda) Tarjeta  $\mu$ SD como almacenamiento persistente del sistema.
- (Parte izquierda) Adaptador SD –  $\mu$ SD para flashear la tarjeta  $\mu$ SD desde un PC e introducirle el sistema operativo inicialmente.
- (Parte central) Raspberry Pi.
- (Parte inferior) Alimentación a la Raspberry Pi por  $\mu$ USB.
- (Parte derecha) Tarjeta Alfa conectada por USB.
- (Parte derecha) Conexión al router a través de Ethernet.

Se incluye en el anexo I la serie de pasos que es necesario realizar para preparar toda la infraestructura.





Figura 5. Hardware usado

El proceso para realizar los experimentos y el consiguiente análisis es el siguiente:

1. La STA que simulará el movimiento a través del túnel mientras recibe tráfico comienza a capturar el tráfico (a través de Wireshark o TCPDump ) que recibe a través de la tarjeta de red con la que está conectada a la red WiFi. A esta STA se le denominará a partir de ahora STA1.
2. La STA que simula el envío de tráfico en tiempo real a la STA1 comienza a enviar de forma periódica, cada 10 ms, paquetes UDP con un identificador único por cada paquete. A esta STA se le denominará a partir de ahora STA2.
3. Tras un tiempo determinado, la STA1 se mueve progresivamente y se produce el *handoff*.
4. Tras otro espacio de tiempo, se para el envío del tráfico desde la STA2 y la captura del tráfico en la STA1.
5. Se exporta el tráfico como CSV (mediante Wireshark o TShark).
6. Se limpian y formatean los datos del fichero CSV y se guardan como un archivo de MATLAB por si se necesita usar en sucesivas ocasiones (script *csvToMat.m*, anexo II; función auxiliar *customhex2num*, anexo III).
7. Se procesan los datos ya formateados y se obtienen datos estadísticos y gráficas sobre el experimento realizado (script *SEU\_Stats.m*, anexo IV).

Como se puede observar, este proceso difiere en algunos aspectos del experimento realizado en la Universidad de Maryland. Por ejemplo:

- La STA móvil recibe tráfico UDP en vez de enviar tráfico ICMP.
- La captura del tráfico se realiza en la misma WNIC que se conecta a la red, en vez de realizarse con *sniffers* externos.

Además, dadas las características del lugar donde se realiza el trabajo (laboratorio 2.08 del edificio Ada Byron, Escuela de Ingeniería y Arquitectura, Zaragoza), también se aprecian las siguientes diferencias:

- Existen bastantes redes WiFi adicionales que pueden causar interferencias. En la Figura 6 se puede ver la saturación de los distintos canales WiFi (la red SEUtest es la propia para realizar el experimento). También es diferente respecto al túnel, pues ahí no habrá más redes WiFi.
- Hay bastantes obstáculos (paredes, mesas, ordenadores, pantallas...). Esto también difiere de las condiciones en el entorno del túnel, ya que ahí no habrá ningún obstáculo.
- La distancia entre los APs es muy corta (alrededor de 3 metros), mientras que en el túnel están separados por una distancia más larga (alrededor de 250 metros).

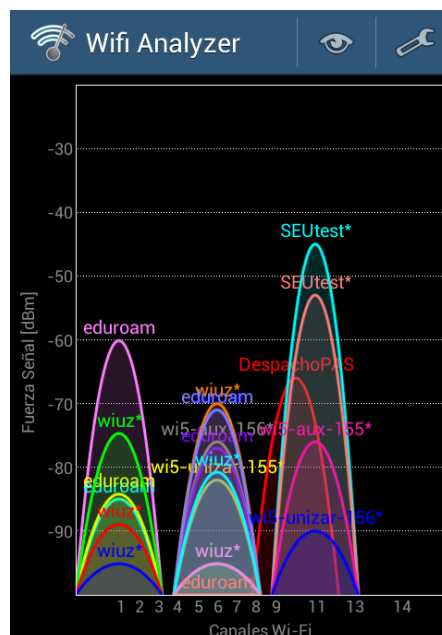


Figura 6. Redes emitiendo en Wifi en cada uno de los canales

Los campos capturados por la STA1 deben ser los mostrados en la Figura 7. Cabe destacar los siguientes:

- MAC fuente y destino, que indican qué dispositivo es el emisor y cuál el receptor
- MAC transmisor y receptor, que indican qué dispositivo es el emisor y cuál el receptor. La Figura 8 ilustra la diferencia que hay entre estos campos en diferentes situaciones.
- *Retry*, para saber si se trata de la retransmisión de una trama anterior
- *Rate*, que muestra la velocidad a la que se está transmitiendo
- *Received Signal Strength Indicator (RSSI)*, es decir, el nivel de señal que se recibe en el punto donde se está capturando el tráfico respecto al dispositivo que envía la trama.

Displayed	Title	Type	Fields	Field Occurrence
<input checked="" type="checkbox"/>	No.	Number		
<input checked="" type="checkbox"/>	MAC src	Custom	wlan.sa	0
<input checked="" type="checkbox"/>	MAC dst	Custom	wlan.da	0
<input checked="" type="checkbox"/>	MAC transmitter	Custom	wlan.ta	0
<input checked="" type="checkbox"/>	MAC receiver	Custom	wlan.ra	0
<input checked="" type="checkbox"/>	BSSID	Custom	wlan.bssid	0
<input checked="" type="checkbox"/>	Time	Time (format as specified)		
<input checked="" type="checkbox"/>	Source	Source address		
<input checked="" type="checkbox"/>	Destination	Destination address		
<input checked="" type="checkbox"/>	Protocol	Protocol		
<input type="checkbox"/>	Length	Packet length (bytes)		
<input type="checkbox"/>	Info	Information		
<input checked="" type="checkbox"/>	Retry	Custom	wlan.fc.retry	0
<input checked="" type="checkbox"/>	Rate	Custom	radiotap.datarate	0
<input checked="" type="checkbox"/>	RSSI	IEEE 802.11 RSSI		
<input checked="" type="checkbox"/>	Data	Custom	data.data	0

Figura 7. Campos a capturar en la STA1

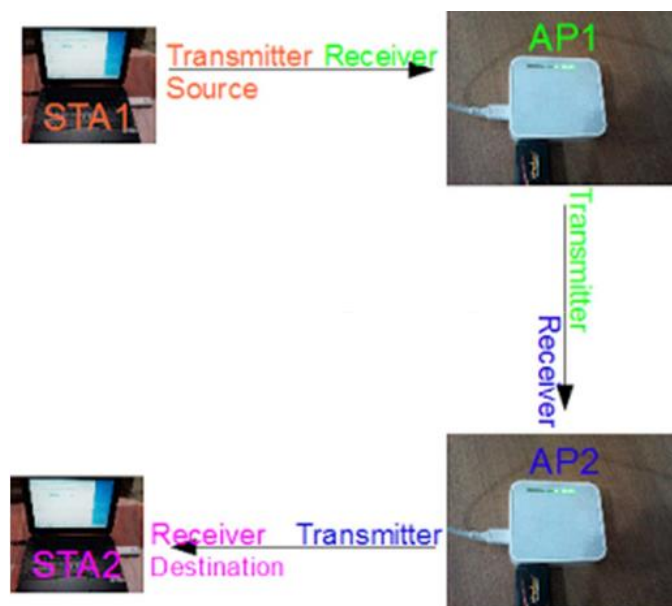


Figura 8. Diferencia entre MAC fuente, destino, transmisor y receptor

### Ambos APs por cable

La configuración más sencilla (y primera que se ha probado) es la mostrada en la Figura 9, en la que se conectan ambos APs por cable Ethernet al router. La STA2 enviará, a través de su conexión Ethernet al router, tráfico a la STA1, que está conectada por WiFi a uno de los dos puntos de acceso.

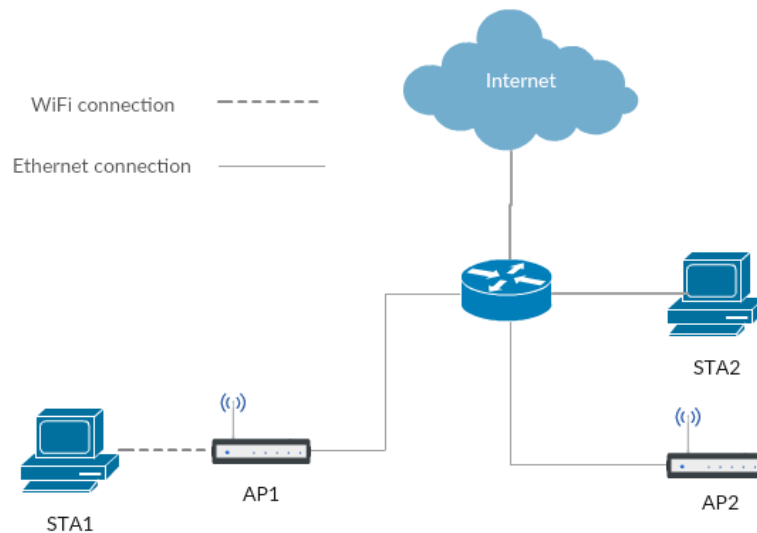


Figura 9. Esquema de infraestructura con ambos APs por Ethernet

La vida de cada uno de los paquetes enviados es la siguiente:

1. El paquete con destino STA1 sale de STA2 a través de la interfaz Ethernet hacia el router.
2. El router redirige el paquete a través de la interfaz Ethernet con el AP al que está conectado STA1.
3. Este AP envía por WiFi el paquete a STA1.

## Resultados

Para ver cómo se comporta esta configuración, se han realizado diversos experimentos y analizado los datos estadísticos obtenidos tras procesar las capturas. En este subapartado se muestran los datos obtenidos más relevantes.

El primer experimento realizado con esta configuración consiste en que los dos APs están operando en canales distintos y la STA se mueve de forma que se aleja de ambos APs. Los datos y gráficas obtenidos son los siguientes:

- La Figura 10 muestra el RSSI del AP al que está conectado durante el transcurso del experimento. Se puede ver que, cuando la STA1 baja de cierto umbral de RSSI, realiza el proceso de *handoff*.
- La Figura 11 contiene la evolución del *Packet Delivery Ratio* (PDR) en una ventana temporal. Para este mostrar este parámetro se ha empleado una ventana de 100 paquetes, por lo que en cada medida de la gráfica se indica cuántos paquetes han conseguido entregarse a su destinatario en una ventana de 100 paquetes. En el proceso de *handoff* existe un pico negativo que muestra la pérdida de paquetes.
- En la Figura 12 se puede observar en azul el *Inter Arrival Time* (IAT), es decir, el tiempo de llegada entre paquetes sucesivos. En el momento en el que se produce el *handoff* se produce una degradación importante. Se puede observar que el IAT aumenta hasta 2819,20 ms, que es lo que tarda la STA1 en reasociarse con el nuevo AP. Además, se

puede ver en rojo el acumulado de paquetes perdidos durante todo el experimento, que sufre un pico importante durante el *handoff*.

- En la Figura 13 se puede apreciar la distribución que siguen los paquetes perdidos mediante un diagrama de caja. En este diagrama se muestra el Q1 (cuartil 1) y Q3 (cuartil 3) como partes inferiores y superiores de la caja, mientras que la mediana se indica con una línea roja entre estos dos cuartiles. Además, los "bigotes" (las líneas discontinuas que acaban con una línea continua que las corta) indican las medidas mínima y máxima de la distribución. Las medidas que se consideran aisladas, es decir, fuera de la distribución, se muestran con el signo "+" en rojo. Estas medidas aisladas suelen suponer una cantidad ínfima respecto al total. En este caso el total son 675 paquetes perdidos. En este diagrama se puede observar que la gran mayoría de paquetes que se han perdido han sido durante el proceso de *handoff* (en el eje Y está la evolución temporal).
- Por último, de **6772 paquetes** que se enviaron se han perdido **675 (un 9,97%)**, la mayoría de ellos durante el *handoff*, que duró **2819,20 ms**.

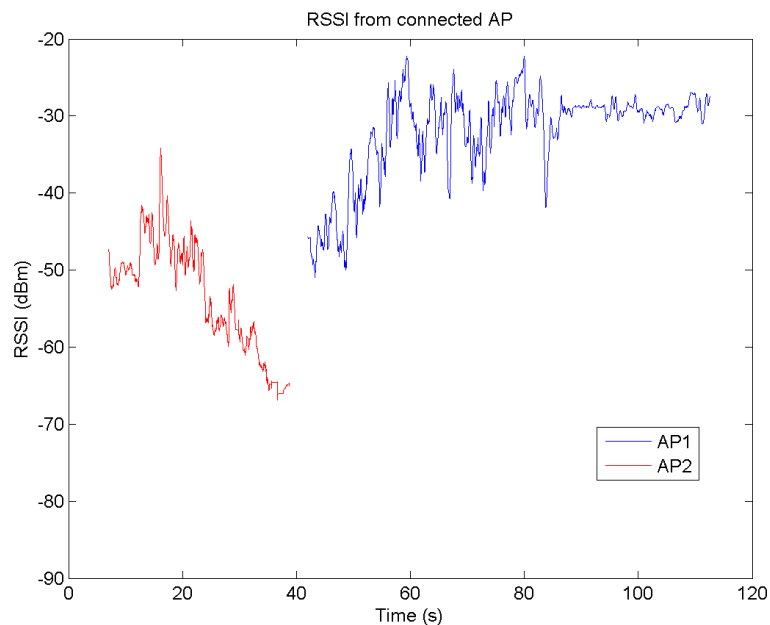


Figura 10. RSSI del AP al que está conectado la STA1



Figura 11. Packet Delivery Ratio

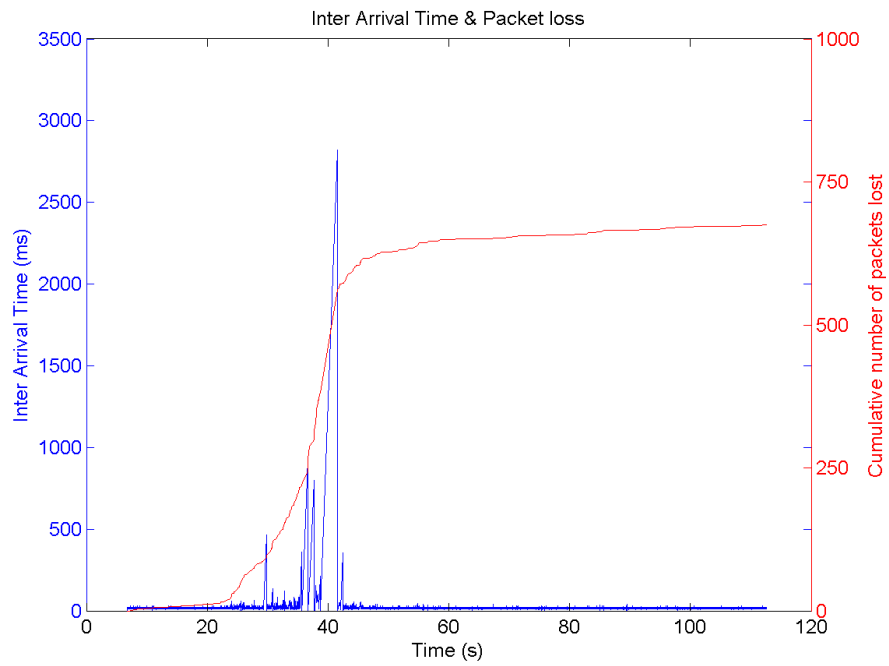
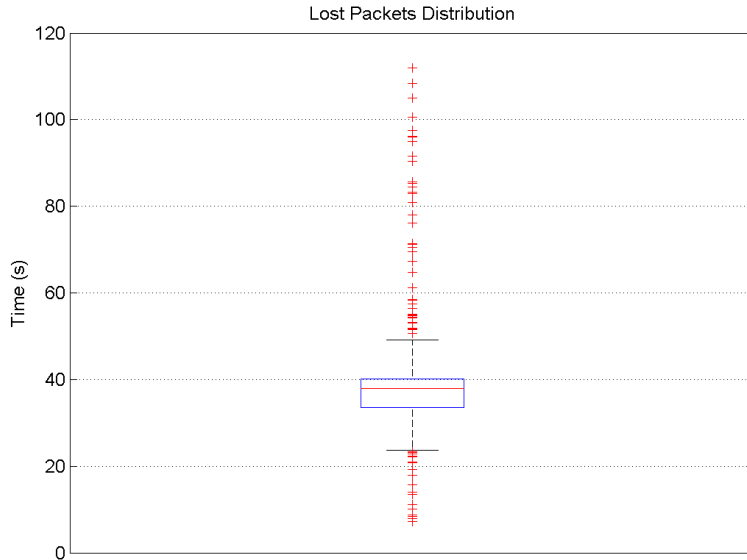


Figura 12. Inter Arrival Time y acumulado de paquetes perdidos



*Figura 13. Distribución de los paquetes perdidos*

Se ha realizado otro experimento en el que los APs están operando en el mismo canal y la STA1 se mueve de forma que se aleja progresivamente de un AP y se va acercando a otro AP:

- En este caso, se puede deducir el movimiento a partir de la evolución de la señal respecto a los dos APs (posible de obtener gracias a estar los dos APs en el mismo canal y poder capturar los beacons de los dos) en la Figura 14. En el caso de la Figura 15 sólo se muestra la evolución de la señal respecto al AP con el que se está conectado.
- El PDR disminuye muy ligeramente hasta completarse el proceso de *handoff* (ver Figura 16).
- El IAT aumenta durante el tiempo que dura este proceso, así como la cantidad de paquetes perdidos, que no se ve incrementada una vez se completa el proceso. Esto se puede apreciar en la Figura 17.
- Finalmente, de **5960** paquetes enviados se perdieron **190 (3,19 %)** y el tiempo de reasociación duró **46,26 ms**, lo cual es aceptable teniendo en cuenta los requisitos de la aplicación de tiempo real.

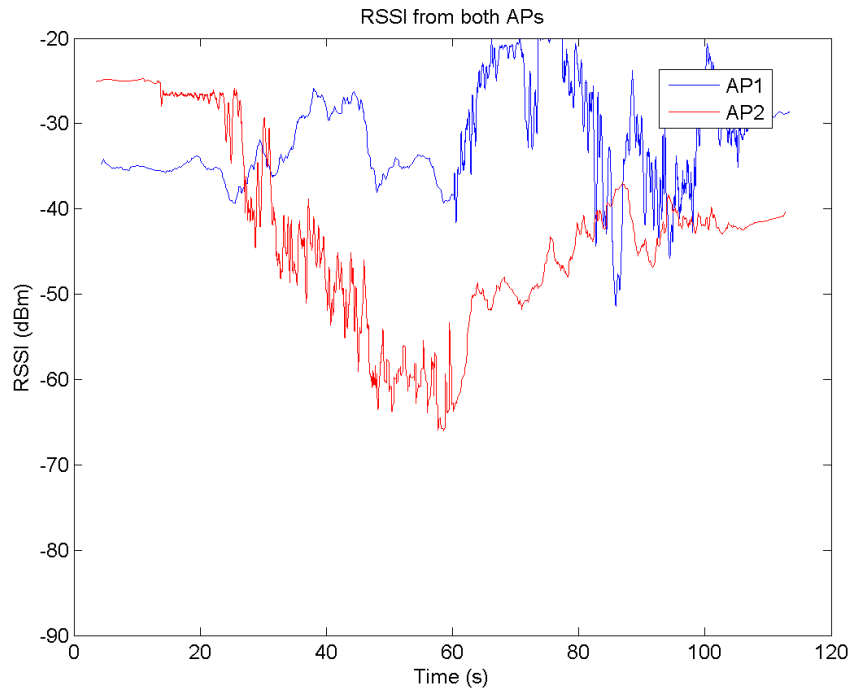


Figura 14. RSSI de los dos APs

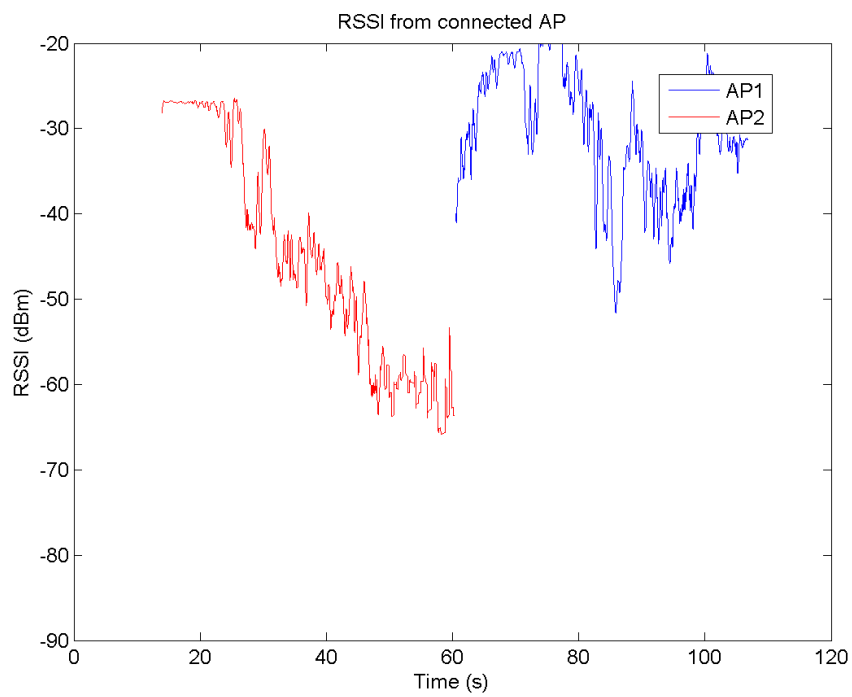


Figura 15. RSSI del AP al que está conectado la STA1



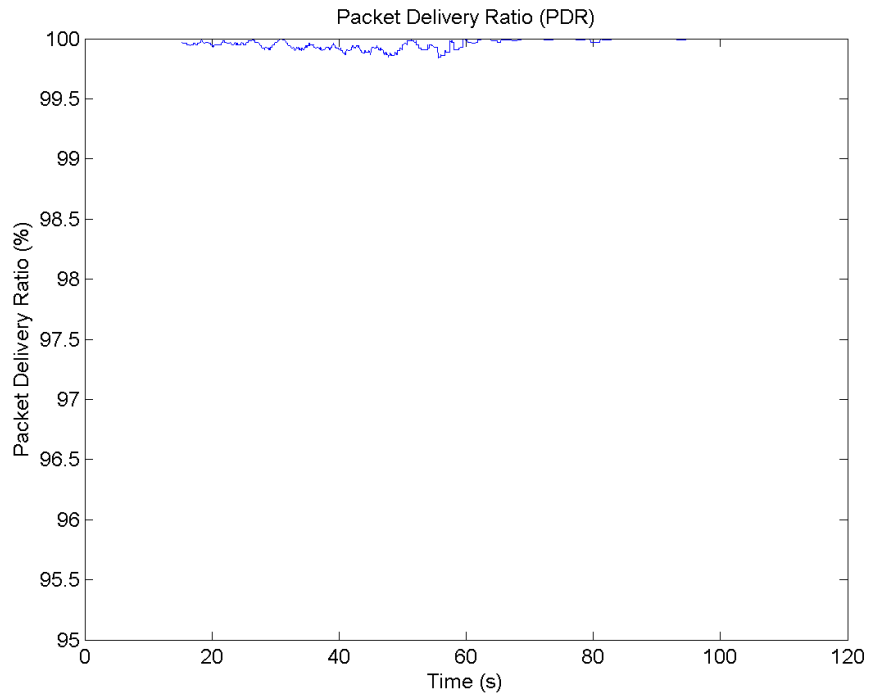


Figura 16. Packet Delivery ratio

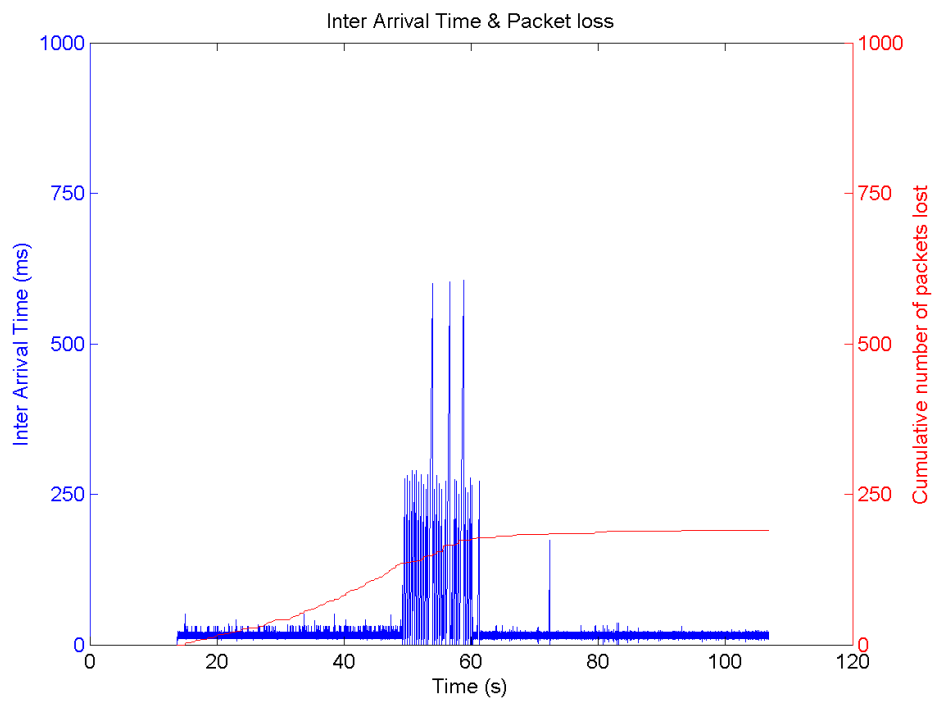


Figura 17. Inter Arrival Time y acumulado de paquetes perdidos

## Wireless Distribution System (WDS)

Esta configuración implica que uno de los APs está conectado por cable mientras que el otro hace de repetidor de forma inalámbrica, tal y como se muestra en la Figura 18, lo que tiene una serie de implicaciones a tener en cuenta:

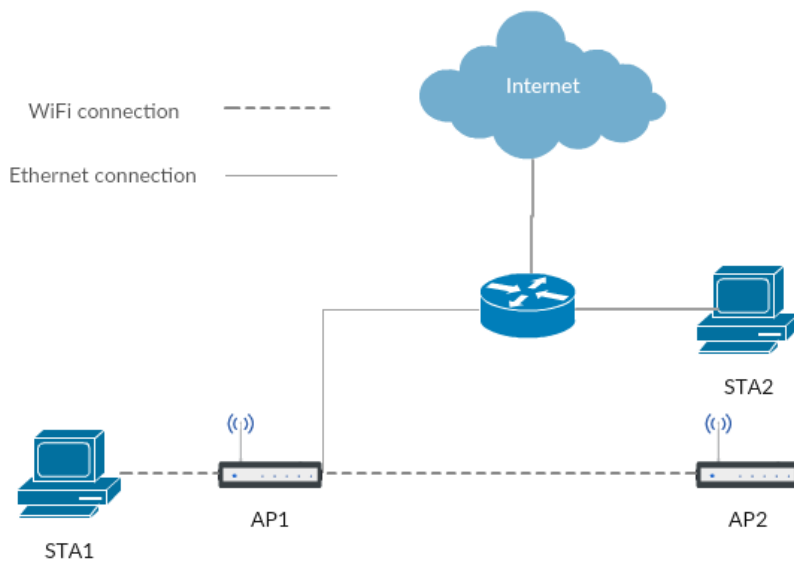
- Todo el tráfico, tanto entrante como saliente, que pasa por AP2 tiene que pasar también por AP1, por lo que el rendimiento del AP2 se divide por la mitad. Esto puede resolverse si se usa un canal para la comunicación AP-to-AP y otro para la comunicación AP-to-STA y STA-to-AP. Para los experimentos realizados no se ha resuelto este problema, por lo que todas las comunicaciones se realizan en el mismo canal.
- Las WNICs de los APs deben ser del mismo fabricante, ya que la configuración WDS no es un estándar y la implementación depende del fabricante. Incluso es habitual que la implementación entre diferentes modelos de un mismo fabricante también varíe.

La vida un paquete puede variar según el AP al que esté conectado la STA1. Si está conectado al AP1 entonces:

1. El paquete con destino STA1 sale de STA2 a través de la interfaz Ethernet hacia el router.
2. El router redirige el paquete a través de la interfaz Ethernet con el AP al que está conectado STA1.
3. Este AP envía por WiFi el paquete a STA1.

Si por el contrario, la STA1 está conetada al AP2 entonces:

1. El paquete con destino STA1 sale de STA2 a través de la interfaz Ethernet hacia el router.
2. El router redirige el paquete a través de la interfaz Ethernet al AP1.
3. El AP1 envía este paquete al AP2 a través de WiFi.
4. El paquete llega a la STA1 a través de WiFi desde el AP2.

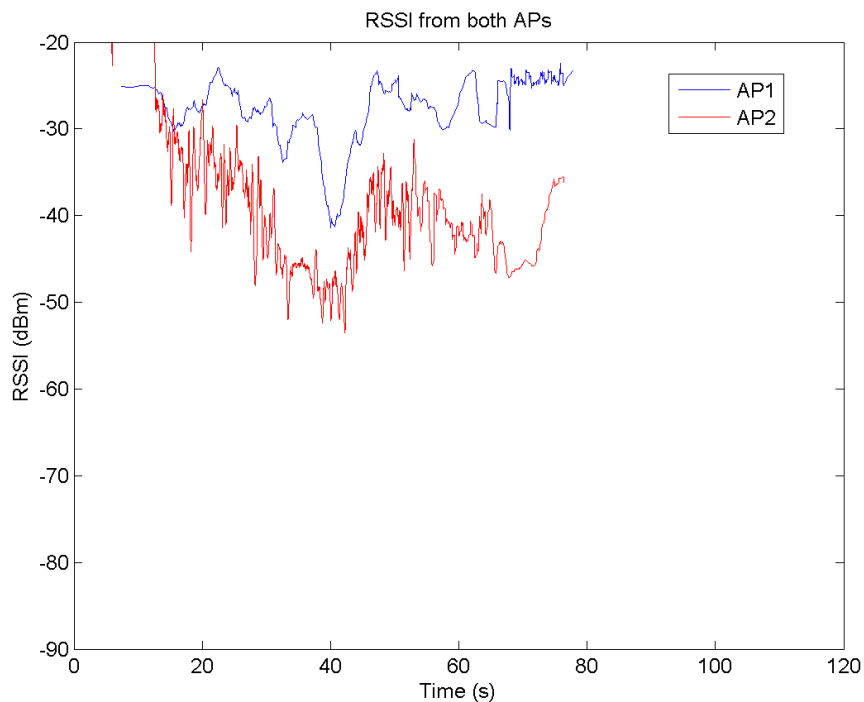


*Figura 18. Esquema de infraestructura con WDS*

## Resultados

En este subapartado se muestran los resultados de un experimento en el que, como se ha mencionado anteriormente, ambos APs están operando en el mismo canal, y la STA1 se aleja progresivamente de un AP mientras se acerca al otro:

- Como se puede observar en la Figura 19, la señal es bastante inestable, aunque aún se puede deducir el movimiento de la STA1 en ciertos momentos. En la Figura 20 está la misma gráfica pero sólo mostrando la RSSI del AP al que está conectada la STA1.
- En la Figura 21 se muestra el PDR, que sufre un pico negativo en el momento de la reasociación.
- La Figura 22 contiene el IAT y el acumulado de paquetes perdidos. En el caso de los paquetes perdidos, se puede apreciar que va aumentando en la misma medida siempre hasta el momento de la reasociación, que experimenta un pico notable, aunque tras ese espacio de tiempo vuelve a la misma tendencia.
- De **4541** paquetes enviados se perdieron **431 (9,49 %)** y el tiempo de reasociación fue de **134,53 ms**. De nuevo, entra dentro de los límites establecidos de 150 ms.



*Figura 19. RSSI de los dos APs*

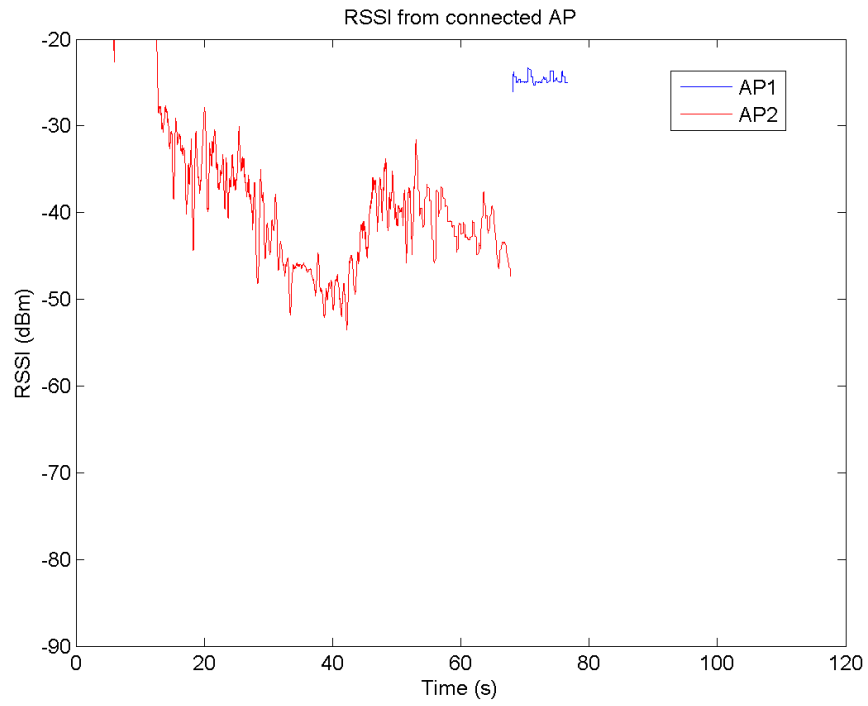


Figura 20. RSSI del AP al que está conectado la STA1

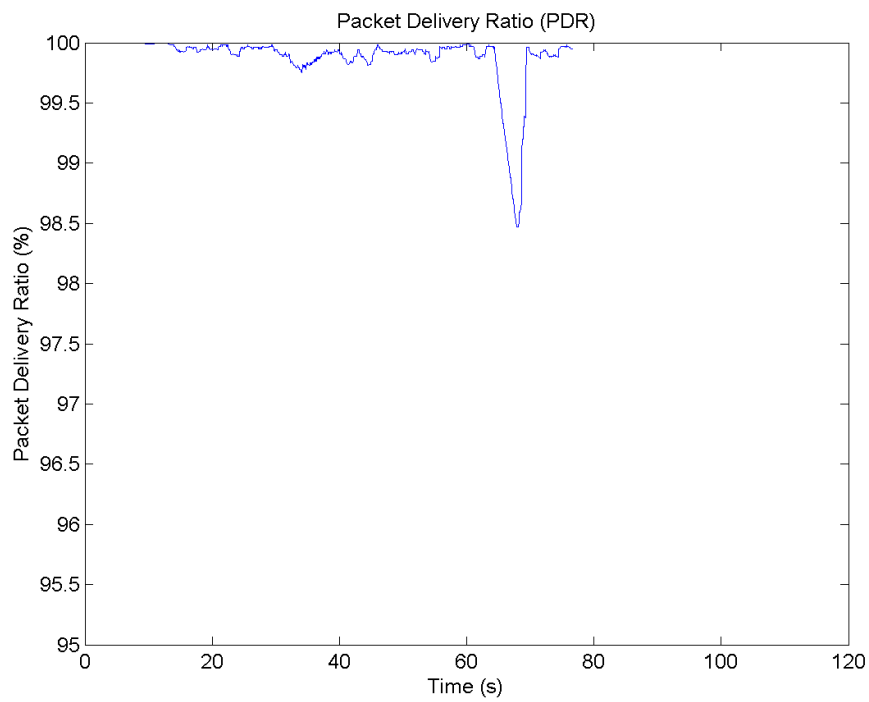


Figura 21. Packet Delivery Ratio

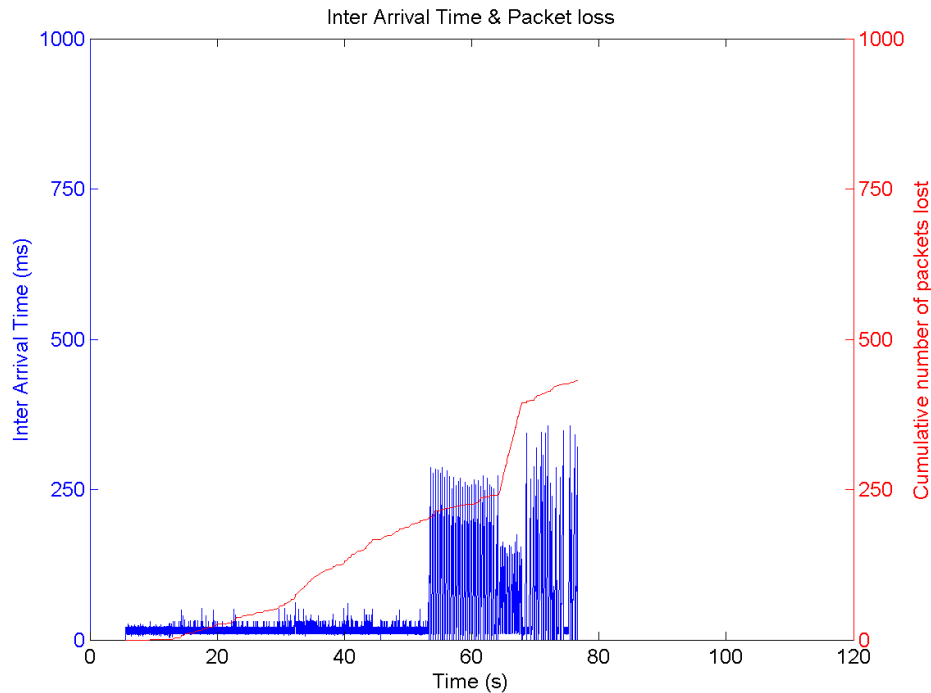


Figura 22. Inter Arrival Time y acumulado de paquetes perdidos

## Resultados generales

Además de los experimentos anteriores, se han realizado otros experimentos adicionales para ver el comportamiento de la red, por ejemplo, si la STA1 se mantiene en el mismo lugar físico sin moverse. La siguiente tabla muestra los resultados obtenidos de todos los experimentos realizados:

Experiment setup				Results			
WDS		Channel	Movement policy	Number of packets sent	Number of packets lost	Percentage of packets lost	AP change reassociation time (ms)
Status	STA Connected to						
Disabled	N/A	Same	Not moving	6998	47	0,67%	
			Move away from both APs	7070	783	11,07%	3.090,80
			Move away from one AP, get closer to the other one	5960	190	3,19%	46,26
		Different	Not moving	6996	7	0,10%	
			Move away from both APs	6772	675	9,97%	2.819,20
Enabled	Master AP	Same	Not moving	6695	285	4,26%	
	Slave AP			6584	980	14,88%	
	N/A		Move away from one AP, get closer to the other one	4541	431	9,49%	134,53

Cabe destacar el tiempo de reasociación en los experimentos en los que la STA1 se aleja de ambos APs, que aproximadamente es de **3 segundos**. Sin embargo, si la STA1 se aleja de un AP pero se acerca al otro, el tiempo de reasociación es aceptable, ya cumple con los requerimientos previamente establecidos. Por último, en la configuración WDS, si la STA1 se asocia al AP que hace de repetidor, el resultado es una gran pérdida de paquetes de casi un **15%, un 10% más** que si estuviera asociada al AP conectado por cable.

## Problemas encontrados

En este apartado se describen algunos de los problemas encontrados durante el desarrollo de este trabajo:

- A pesar de que las tarjetas Alfa sean el mismo modelo del mismo fabricante, con algunas no es posible habilitar el WiFi en ninguna de las dos placas. Si una tarjeta falla, falla en las dos placas con el mismo error. Sin embargo, en base a la experiencia con varias tarjetas alfa, si varias tarjetas fallan es posible que el error que da cada tarjeta sea distinto, independientemente de que sean el mismo modelo del mismo fabricante.
- Hay mucha documentación para OpenWRT, pero algunos apartados no están lo suficientemente organizados y explicados. Además, en ocasiones la documentación no se encuentra actualizada a la última versión del sistema operativo (por ejemplo, prácticamente todos los ejemplos de configuración a través de la interfaz web del sistema que se explican son con la interfaz web antigua).
- En algunas ocasiones los logs del sistema no son lo suficientemente claros para identificar qué es lo que ha fallado.

## Trabajo futuro

Para completar este trabajo se plantean una serie de tareas que se podrían realizar con el fin de mejorar todo lo desarrollado anteriormente:

- Tal y como ya se ha mencionado, sería interesante probar la configuración WDS con dos WNICs en cada AP, de manera que la comunicación AP-to-AP se haga en una frecuencia determinada a través de una WNIC y las comunicaciones AP-to-STA y STA-to-AP en otra frecuencia mediante la otra WNIC, evitando así interferencias.
- Ajustar algunos parámetros como la RSSI umbral a la que se inicia el proceso de *handoff*, los canales en los que buscar APs en la fase de *probing*, el intervalo entre beacons de los APs, etc. De esta manera, se podría analizar el impacto que tienen estos parámetros y poder ajustarlos a las características del entorno y de las necesidades de la aplicación.
- Probarlo en el túnel o un ambiente que se asemeje a él ya que, como ya se ha comentado, el entorno en el que se han realizado los experimentos difiere en algunos aspectos del entorno real.
- Probar otras soluciones como SDN + AP virtual [2] para observar las diferencias con la solución ya desarrollada y saber qué solución conviene adoptar dadas las necesidades de la aplicación. La Figura 23 ilustra una configuración de este tipo. Lo que caracteriza a esta solución es que la STA móvil siempre piensa que está conectada a un mismo AP, que en realidad es virtual. De esta forma, es la red, a través de un controlador de una SDN, quien decide mover ese AP virtual desde un AP físico a otro. Una de las ventajas de esta solución es que esta decisión de mover un AP virtual se basa en el estado de la red, es decir, en información que la STA no tiene. Por ejemplo, podría tenerse en cuenta el tráfico que está soportando cada AP físico, de manera que el controlador puede hacer equilibrar la carga entre los APs físicos que componen la red.

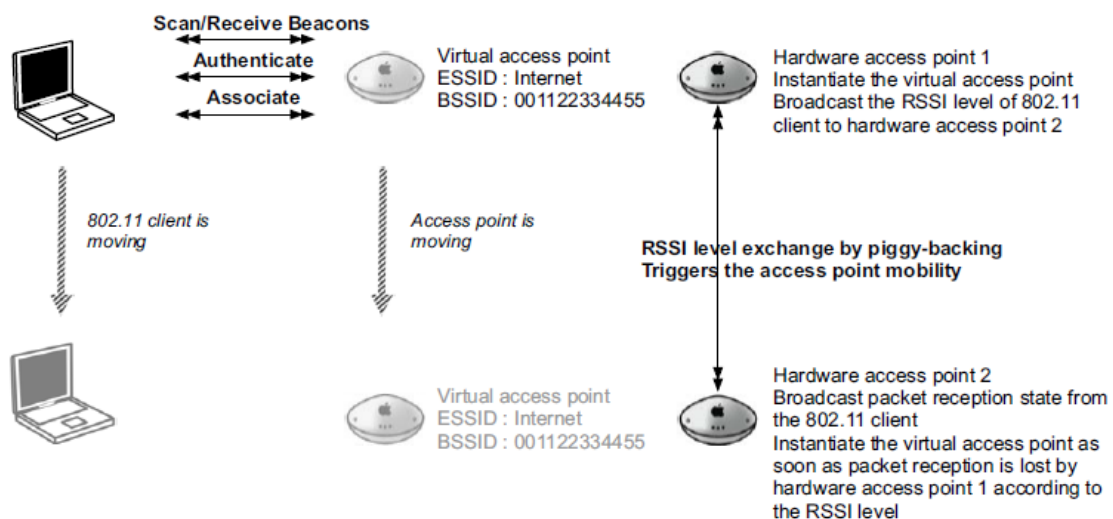


Figura 23. Solución de APs virtuales + SDN

## Referencias

- [1] A. Mishra, M. Shin and W. Arbaugh, *An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process (2002)*
- [2] Yan Grunenberger and Franck Rousseau, *Virtual Access Points for Transparent Mobility in Wireless LANs (2010)*



## Enlaces de interés

OpenWRT:

- (Último acceso: 18/06/2017) <https://wiki.openwrt.org/doc/uci/wireless>
- (Último acceso: 18/06/2017) <https://wiki.openwrt.org/doc/howto/clientmode>
- (Último acceso: 18/06/2017) <https://wiki.openwrt.org/doc/howto/wide.area.wifi>

hostapd (aunque la configuración que se hace por ficheros de hostapd se puede hacer también con los métodos descritos en el anexo I):

- (Último acceso : 18/06/2017)  
<http://linuxwireless.org/en/users/Documentation/hostapd/>
- (Último acceso: 18/06/2017)  
<https://wireless.wiki.kernel.org/en/users/documentation/hostapd>
- (Último acceso: 18/06/2017) <https://w1.fi/cgit/hostap/plain/hostapd/hostapd.conf>

Código fuente de los scripts de MATLAB:

- (Último acceso: 18/06/2017) <https://github.com/MarcosCM/wifi-traffic-analytics/tree/master/scripts>

## Anexos

### Anexo I (preparación de la infraestructura)

Instalar OpenWRT:

1. Descargar OpenWRT 15.05.1 Chaos Calmer para Raspberry Pi 2: [https://downloads.openwrt.org/chaos\\_calmer/15.05.1/brcm2708/bcm2709/openwrt-15.05.1-brcm2708-bcm2709-sdcard-vfat-ext4.img](https://downloads.openwrt.org/chaos_calmer/15.05.1/brcm2708/bcm2709/openwrt-15.05.1-brcm2708-bcm2709-sdcard-vfat-ext4.img)
2. Insertar la tarjeta µSD y flashearla (en un PC de Windows se puede hacer con la utilidad Win32DiskImager: <https://sourceforge.net/projects/win32diskimager/>).
3. Expulsar con seguridad la tarjeta.

Configurar acceso la Raspberry Pi 2:

1. Insertar la tarjeta µSD en la placa.
2. Conectar la alimentación eléctrica a la placa.
3. Conectar el PC a la placa (por ejemplo, a través de Ethernet montando una red ad-hoc).
4. Abrir navegador web y conectarse a la interfaz web accediendo a la dirección 192.168.1.1 (user: root; password en blanco).
5. Configurar una nueva contraseña para acceder a la placa tanto por la interfaz web como por SSH: System → Administration → (Introducir nueva contraseña) → Save & Apply.

Configurar red:

1. Acceder a la Raspberry por SSH.
2. Actualizar IP, máscara, gateway y DNS de la interfaz para que coincida con los requerimientos de la red a la que se va a conectar y deshabilitar DHCP y DHCPv6. Esto se puede hacer a través de los ficheros de configuración del sistema.

(/etc/config/network)

```
config interface 'lan'
```

```
option ifname 'eth0'
```

```
option type 'bridge'
```

```
option proto 'static'
```

```
option ipaddr '155.210.155.183'
```

```
option netmask '255.255.255.0'
```

```
option gateway '155.210.155.254'
```

```
option ip6assign '60'
```

```
list dns '8.8.8.8'
```

```
(/etc/config/dhcp)
```

```
config dhcp 'lan'
```

```
option interface 'lan'
```

```
option ignore '1'
```

Instalar drivers y configurar WiFi:

1. Reiniciar la placa (comando 'reboot').
2. Conectar la placa al router via Ethernet.
3. Conectar el PC al router via Ethernet.
4. Conectarse a la placa mediante SSH con la nueva IP que se le ha fijado.
5. Ejecutar 'opkg update'.
6. Ejecutar 'opkg install kmod-rt2800-lib kmod-rt2800-usb kmod-rt2x00-lib kmod-rt2x00-usb' para instalar los drivers de la tarjeta Alfa.
7. Ejecutar 'opkg install kmod-usb-core kmod-usb-ohci kmod-usb-uhci kmod-usb2 usbutils' para instalar los drivers de los puertos USB.
8. Ejecutar 'opkg install iwinfo' para instalar una utilidad que permite obtener información legible sobre las interfaces.
9. Ejecutar 'opkg install hostapd wpa-suplicant' para instalar el software que permitirá activar y gestionar las conexiones WiFi.
10. Reiniciar la placa (comando 'reboot').
11. Conectar la Alfa a la Raspberry via USB.
12. Ejecutar 'wifi detect > /etc/config/wireless' para establecer las opciones por defecto.
13. Configurar las opciones WiFi: 0 dBm (1 mW), canal, etc y habilitar WiFi (para modificar estas opciones a través de la interfaz de web tiene que existir el fichero '/etc/config/wireless', creado al ejecutar 'wifi detect > /etc/config/wireless').

```
(/etc/config/wireless)
```

```
config wifi-device 'radio0'
```

```
option type 'mac80211'
```

```
option channel '11'
```

```
option hwmode '11g'
```

```
option path 'platform/bcm2708_usb/usb1/1-1/1-1.3/1-1.3:1.0'
```

```
option txpower '0'
```

```
option country 'ES'
```

```
option htmode 'HT20'
```

config wifi-iface

option device 'radio0'

option network 'lan'

option mode 'ap'

option encryption 'none'

option ssid 'SEUtest'

**\*\*NOTA\*\***: la línea que contiene 'option path' depende del puerto en el que se conecta la tarjeta Alfa, así que se recomienda que no se conecte a otro puerto. Sino, es necesario cambiar esta opción (la forma más rápida es hacer copia de seguridad renombrando el archivo '/etc/config/wireless' y ejecutar 'wifi detect' para ver qué 'option path' es el que indica.

Habilitar WiFi:

1. Ejecutar 'wifi up'.
2. Si no se ha activado en 5 segundos entonces ejecutar 'wifi up' de nuevo. Si esto pasa, puede pasar una o dos veces, por fallos al leer '/var/run/wifi-phyX.pid' o '/var/run/hostapd-phyX.conf' (puede comprobarse en el log del sistema a través de la interfaz web en la pestaña 'Status'). Sino, alguno de los pasos anteriores no se ha hecho correctamente.

**\*\*NOTA\*\***: No hay una opción a través de la interfaz web para hacer que la antena comience a dar cobertura WiFi, esto tiene que hacerse a través del comando anteriormente mencionado ('wifi up').

**\*\*NOTA\*\***: Es posible que la corriente del adaptador en la interfaz web no se actualice hasta que el adaptador un dispositivo se conecte a su red WiFi o que el propio adaptador se conecte a otra red WiFi. Para verificar la corriente del adaptador probar los comandos 'iwinfo wlan0 info' y 'iwlist wlan0 txpower'. Si la corriente del adaptador no se ha actualizado en estas condiciones entonces es posible que tenga activado el ahorro automático de energía (que no permite controlar la corriente a suministrarle). Para deshabilitar este ahorro automático de energía ejecutar 'iw dev wlanX set power\_save off'.

Conectar y capturar tramas en Wireshark desde la STA1:

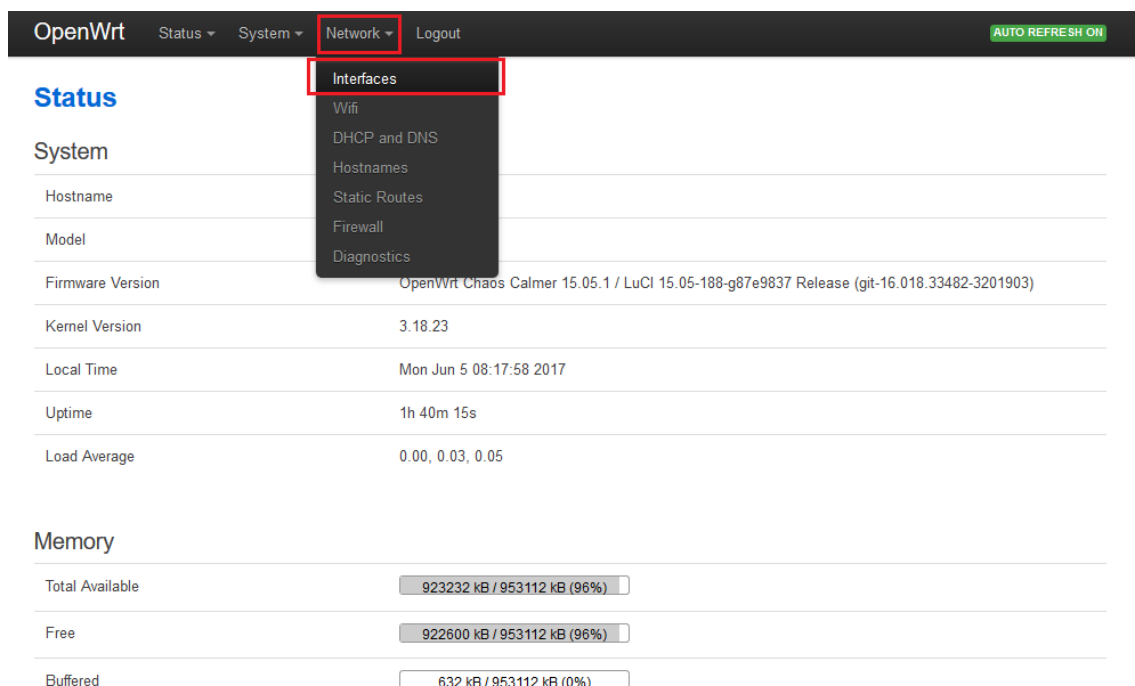
1. Conectar la tarjeta Alfa al PC.
2. Ejecutar 'sudo iw dev wlanX interface add monX type monitor' para añadir una interfaz virtual en modo monitor que sea capaz de capturar las tramas con todos los campos requeridos.
3. Ejecutar 'sudo ifconfig mon1 up'.

4. Conectarse a la red WiFi con la interfaz correspondiente a la tarjeta (y establecer una IP estática, ya que se ha deshabilitado DHCP en los APs).
5. Ejecutar 'sudo wireshark' para abrir Wireshark.
6. Configurar la interfaz monX en Wireshark para que capture el tráfico en modo promíscuo.

Analizar la captura:

1. Abrir la captura desde Wireshark.
2. Establecer el filtro 'radiotap.dbm\_antsignal' para mostrar sólo los paquetes WiFi.
3. Exportar a un fichero CSV mediante 'File -> Export Packet Dissections -> as "CSV" -> All packets (Displayed)'.
4. Abrir MATLAB y el fichero 'csvToMat.m' y cambiar los índices de las columnas para que correspondan con las columnas del fichero CSV exportado.
5. Cambiar los nombres de los ficheros a analizar y ejecutar el script.
6. Abrir el fichero 'SEU\_Stats.m' y modificar los ficheros a analizar y los parámetros de las gráficas como se desee y ejecutar el script (se guardarán las gráficas automáticamente como imágenes con formato '.png' en el directorio).

En la siguiente secuencia de imágenes se muestra cómo configurar las interfaces de red a través de la interfaz web que proporciona OpenWRT.








The screenshot shows the OpenWRT web interface. At the top, there is a navigation bar with 'OpenWrt', 'Status', 'System', 'Network', and 'Logout'. The 'Network' menu is open, showing options like 'Interfaces', 'Wifi', 'DHCP and DNS', 'Hostnames', 'Static Routes', 'Firewall', and 'Diagnostics'. The 'Interfaces' option is highlighted. Below the navigation bar, the 'Status' section is visible, showing system information like 'Hostname', 'Model', 'Firmware Version', 'Kernel Version', 'Local Time', 'Uptime', and 'Load Average'. The 'Memory' section is also visible, showing 'Total Available', 'Free', and 'Buffered' memory usage.

Section	Item	Value
Status	Hostname	
	Model	
	Firmware Version	OpenWrt Chaos Calmer 15.05.1 / LuCI 15.05-188-g87e9837 Release (git-16.018.33482-3201903)
	Kernel Version	3.18.23
	Local Time	Mon Jun 5 08:17:58 2017
	Uptime	1h 40m 15s
	Load Average	0.00, 0.03, 0.05
Memory	Total Available	923232 kB / 953112 kB (96%)
	Free	922600 kB / 953112 kB (96%)
	Buffered	632 kB / 953112 kB (0%)

## Interfaces

### Interface Overview

Network	Status	Actions
<div>LAN</div> <div>br-lan</div>	<div>Uptime: 1h 40m 35s</div> <div>MAC-Address: B8:27:EB:28:B6:0B</div> <div>RX: 1.78 GB (2695292 Pkts.)</div> <div>TX: 1.80 GB (2663047 Pkts.)</div> <div>IPv4: 155.210.155.183/24</div> <div>IPv6: fd06:6b68:4280::1/60</div>	<div> Connect</div> <div> Stop</div> <div> Edit</div> <div> Delete</div>

 Add new interface...


### Global network options

IPv6 ULA-Prefix


Save & Apply Save Reset

Powered by LuCI 15.05-188-g87e9837 Release (git-16.018.33482-3201903) / OpenWrt Chaos Calmer 15.05.1

General Setup Advanced Settings Physical Settings Firewall Settings

Status br-lan


Uptime: 1h 41m 56s  
MAC-Address: B8:27:EB:28:B6:0B  
RX: 1.78 GB (2701370 Pkts.)  
TX: 1.81 GB (2668685 Pkts.)  
IPv4: 155.210.155.183/24  
IPv6: fd06:6b68:4280::1/60

Protocol	Static address ▾
IPv4 address	<input type="text" value="155.210.155.183"/>
IPv4 netmask	255.255.255.0 ▾
IPv4 gateway	<input type="text" value="155.210.155.254"/>
IPv4 broadcast	<input type="text"/>
Use custom DNS servers	<input type="text" value="8.8.8.8"/> 

IPv6 assignment length

 Assign a part of given length of every public IPv6-prefix to this interface

IPv6 assignment hint

 Assign prefix parts using this hexadecimal subprefix ID for this interface.

## DHCP Server

General Setup

IPv6 Settings

Ignore interface ☒ [Disable DHCP](#) for this interface.

[Back to Overview](#)

Save & Apply

Save

Reset

## DHCP Server

General Setup

IPv6 Settings

Router Advertisement-Service disabled

DHCPv6-Service disabled

NDP-Proxy disabled

Announced DNS servers

Announced DNS domains

[Back to Overview](#)

Save & Apply

Save

Reset

En la siguiente secuencia de imágenes se muestra cómo configurar el WiFi mediante la interfaz web.

OpenWrt

Status ▾System ▾Network ▾Logout

AUTO REFRESH ON

Status

System

Hostname

Model

Firmware Version

Kernel Version

Local Time

Uptime

Load Average

Interfaces

Wifi

DHCP and DNS

Hostnames

Static Routes

Firewall

Diagnostics

OpenWrt Chaos Calmer 15.05.1 / LuCI 15.05-188-g87e9837 Release (git-16.018.33482-3201903)

3.18.23

Mon Jun 5 08:24:13 2017

1h 46m 8s

0.00, 0.01, 0.05

Memory

Total Available

Free


Buffered

925236 kB / 953112 kB (97%)

924604 kB / 953112 kB (97%)

632 kB / 953112 kB (0%)

### Wireless Overview

 **Generic MAC80211 802.11abgn (radio0)**

ScanAdd

SSID: SEUtest | Mode: Master  
0% Wireless is disabled or not associated

DisableEditRemove

### Associated Stations

SSID	MAC-Address	IPv4-Address	Signal	Noise	RX Rate	TX Rate
------	-------------	--------------	--------	-------	---------	---------

No information available

Powered by LuCI 15.05-188-g87e9837 Release (git-16.018.33482-3201903) / OpenWrt Chaos Calmer 15.05.1

### Device Configuration

- General Setup
- Advanced Settings

Status

SSID: SEUtest | Mode: Master  
0% Wireless is disabled or not associated

Wireless network is enabled

Disable

Operating frequency

ModeBandChannelWidth

N2.4 GHz11 (2462 MHz)20 MHz

Transmit Power

0 dBm (1 mW)

dBm

### Device Configuration

- General Setup
- Advanced Settings

Country Code

ES - Spain

Use ISO/IEC 3166 alpha2 country codes.

Distance Optimization

Distance to farthest network member in meters.

Fragmentation Threshold

RTS/CTS Threshold



### Interface Configuration

General SetupWireless SecurityMAC-Filter


ESSID

SEUtest

Mode

Access Point

Network

☒ lan: 

☐ create:

Choose the network(s) you want to attach to this wireless interface or fill out the *create* field to define a new network.

Hide ESSID

☐

WMM Mode

☒

Back to Overview


Save & ApplySaveReset

Powered by LuCI 15.05-188-g87e9837 Release (git-16.018.33482-3201903) / OpenWrt Chaos Calmer 15.05.1


Después de ejecutar el comando WiFi up debería aparecer la siguiente imagen:

OpenWrtStatus ▾System ▾Network ▾LogoutAUTO REFRESH ON

Wireless Overview



**Generic MAC80211 802.11abgn (radio0)**  
Channel: 11 (2.462 GHz) | Bitrate: ? Mbit/s

 SSID: SEUtest | Mode: Master  
0% BSSID: 00:C0:CA:90:79:29 | Encryption: None

ScanAdd

DisableEditRemove

Associated Stations

SSID	MAC-Address	IPv4-Address	Signal	Noise	RX Rate	TX Rate
No information available						

Powered by LuCI 15.05-188-g87e9837 Release (git-16.018.33482-3201903) / OpenWrt Chaos Calmer 15.05.1


Después de que alguna estación se haya conectado a la WiFi debería aparecer la siguiente imagen:

OpenWrt

Status ▾System ▾Network ▾Logout


AUTO REFRESH ON

### Wireless Overview



Generic MAC80211 802.11abgn (radio0)

Channel: 11 (2.462 GHz) | Bitrate: 6.5 Mbit/s

 SSID: SEUtest | Mode: Master

47% BSSID: 00:C0:CA:90:79:29 | Encryption: None

Scan


Add

Disable

Edit

Remove

### Associated Stations

	SSID	MAC-Address	IPv4-Address	Signal	Noise	RX Rate	TX Rate
	SEUtest	00:C0:CA:90:79:3C	155.210.155.134	-77 dBm	0 dBm	1.0 Mbit/s, MCS 0, 20MHz	6.5 Mbit/s, MCS 0, 20MHz

Powered by LuCI 15.05-188-g87e9837 Release (git-16.018.33482-3201903) / OpenWrt Chaos Calmer 15.05.1

## Anexo II (*csvToMat.m*)

```
close all;

% filenames = {'diff_ch_no_moving.csv', 'diff_ch_moving.csv',...
%             'same_ch_no_moving.csv', 'same_ch_moving.csv',...
%             'wds_same_ch_no_moving_masterAP.csv',
%             'wds_same_ch_no_moving_slaveAP.csv',...
%             'wds_same_ch_moving.csv',
%             'realistic_same_ch_moving.csv'};

for idx = 1:numel(filenames)
    clearvars -except filenames idx;

    curr_filename = filenames{:,idx};

    % avoid first row, which contains column tags
    fileID = fopen(curr_filename, 'r');

    %
    no,wlan.sa,wlan.da,wlan.ta,wlan.ra,wlan.bssid,Time,ip.src,ip.dst,proto
    col,length,info,wlan.fc.retry,.radiotap.datarate,wlan.rssi,data.data

    packets = textscan(fileID, '%q %q %q %q %q %q %q %q %q %q %q %q %q
    %q %q %q %q', 'Delimiter', ',', 'HeaderLines', 1);

    fclose(fileID);

    No = packets{:, 1};
    MacSource = packets{:, 2};
    MacDestination = packets{:, 3};
    MacTransmitter = packets{:, 4};
    MacReceiver = packets{:, 5};
    BSSID = packets{:, 6};
    Time = packets{:, 7};
    IPSrc = packets{:, 8};
    IPDst = packets{:, 9};
```

```

Protocol = packets(:, 10);
Length = packets(:, 11);
Info = packets(:, 12);
Retry = packets(:, 13);
Rate = packets(:, 14);
RSSI = packets(:, 15);
Channel = packets(:, 16);
Data = packets(:, 17);

% String to int
No = [cellfun(@str2num, No)];

% String to double
% first Time measure will be 0, the remaining Times will be
relative to
% that one
Time = [cellfun(@str2num, Time)];
Time = Time - min(Time);

% String to int
Length = [cellfun(@str2num, Length)];

% String to boolean
Retry = strrep(Retry, 'Frame is not being retransmitted', '0');
Retry = strrep(Retry, 'Frame is being retransmitted', '1');
Retry = [cellfun(@str2num, Retry)];

% Decimals with dot instead of comma
Rate = strrep(Rate, ',', '.');
Rate = [cellfun(@str2num, Rate)];

% String to int

```

```
RSSI = strrep(RSSI, ' dBm', '');  
RSSI = [cellfun(@str2num, RSSI)];  
  
% String to double  
Data = [cellfun(@customhex2num, Data)];  
  
clearvars ans packets fileID;  
% Save variables into .mat file  
save(strrep(curr_filename, '.csv', '.mat'));  
end
```

### Anexo III (customhex2num.m)

```
function [ num ] = customhex2num( hex )
    if ~strcmp(hex, '') && strcmp(hex, strep(hex, '...', ''))
        num = double(hex2dec(hex));
    else
        num = NaN(1);
    end
end
```

#### Anexo IV (*SEU\_Stats.m*)

```
clearvars;

filename_no_ext = 'diff_ch_moving';
figs_ext = 'png';

% RSSI graphs
y_axis_rssi_lower = -90;
y_axis_rssi_upper = -20;
% PDR graphs
y_axis_pdr_lower = 95;
y_axis_pdr_upper = 100;
% IAT graphs
y_axis_iat_lower = 0;
y_axis_iat_interval = 250;
y_axis_iat_upper = 1000;
% Packet Loss graphs
y_axis_packet_loss_lower = 0;
y_axis_packet_loss_interval = 250;
y_axis_packet_loss_upper = 1000;
% Generic
x_axis_lower = 0;
x_axis_upper = 120;

load(strcat(filename_no_ext, '.mat'));

ExpMacSource = '2c:56:dc:26:17:77';
ExpProtocol = 'UDP';
% BSSID in regular AP
ExpBSSIDAP1 = '00:c0:ca:90:79:29';
% BSSID in WDS slave AP
```

```
%ExpBSSIDAP1 = '00:c0:ca:90:79:28';
```

```
ExpBSSIDAP2 = '00:c0:ca:90:79:3e';
```

```
% Get WiFi UDP packets
```

```
pRSSI = RSSI(strcmp(Protocol, ExpProtocol) & ~isnan(Data) &  
strcmp(MacSource, ExpMacSource));
```

```
pNo = No(strcmp(Protocol, ExpProtocol) & ~isnan(Data) &  
strcmp(MacSource, ExpMacSource));
```

```
pTime = Time(strcmp(Protocol, ExpProtocol) & ~isnan(Data) &  
strcmp(MacSource, ExpMacSource));
```

```
pData = Data(strcmp(Protocol, ExpProtocol) & ~isnan(Data) &  
strcmp(MacSource, ExpMacSource));
```

```
pBSSID = BSSID(strcmp(Protocol, ExpProtocol) & ~isnan(Data) &  
strcmp(MacSource, ExpMacSource));
```

```
pAp(strcmp(pBSSID, ExpBSSIDAP1)) = 1;
```

```
pAp(strcmp(pBSSID, ExpBSSIDAP2)) = 2;
```

```
pAp = pAp';
```

```
pData = pData + 1 - min(pData);
```

```
packets = [pNo pTime pAp pRSSI pData];
```

```
% HINT: UDP packets may be received out of order, so they need to be  
sorted
```

```
for i = 1:length(packets)
```

```
    packetsRx(packets(i, 5), :) = packets(i, :);
```

```
end
```

```
% Sort packets by data (UDP ID) and then packet ID to break ties  
(although
```

```
% there shouldn't be duplicated packets at this point)
```

```
packetsRxSort = sortrows(packetsRx, [5, 1]);
```

```
% Remove blank rows
```



```

packetsRxSort(packetsRxSort(:, 5)<1, :) = [];

% For each packet, get how many packets between the previous one and
this
% one were lost (works out because packets are sorted by UDP ID)
% Example: packetsRxSort(:, 5) = [1 1 2 3 5 8 13 21];
%
%       y = diff(packetsRxSort(:, 5));
%       y = [0 1 1 2 3 5 8]
difData = [0; diff(packetsRxSort(:, 5)) - 1];

% Since packets can be send more than once, the STA may receive duplicate
% packets. Here, diff could result in -1, which doesn't make sense.
% Therefore, negative numbers should be zeroed.
difData(difData<0) = 0;

% Packet Delivery Ratio (PDR): number of packets received vs number of
packets
% sent out by the sender, in percentage
pdr = 100*(1-tsmovavg(difData, 's', 100, 1)/100);

% Inter Arrival Time (IAT): Time between each packet arrival and the
next
% one
iat = [0; 1000*abs(diff(packetsRxSort(:, 2)))];

% Packets with different AP transmitter than the previous one
packetsLostSum = cumsum(difData);

% Get percentage of packets lost vs packets sent (assuming last packet
% received is last packet sent)
packetsLostPctg    =    100*packetsLostSum(size(packetsLostSum,    1),
1)/packetsRxSort(size(packetsRxSort, 1), 5);
packetsTxSortIds = 1:packetsRxSort(size(packetsRxSort, 1), 5);
packetsTxSortIds = packetsTxSortIds';

% 1st column: packet UDP id
packetsLost(:, 1) = setdiff(packetsTxSortIds, packetsRxSort(:, 5));

% 2nd column: estimated times
for i=1:size(packetsLost, 1)
    i_lower    = find(packetsRxSort(:, 5) <= packetsLost(i, 1), 1,
'last');

```

```

        i_upper    = find(packetsRxSort(:, 5) >= packetsLost(i, 1), 1,
'first');
        time_lower = packetsRxSort(i_lower, 2);
        time_upper = packetsRxSort(i_upper, 2);
        % Estimated time will be mean of upper and lower packets time
        % difference
        packetsLost(i, 2) = time_lower + (time_upper - time_lower) / 2;
end

```

% Get beacons

```

bSSID      =      BSSID(strcmp(MacTransmitter,      ExpBSSIDAP1)      |
strcmp(MacTransmitter, ExpBSSIDAP2));

bRSSI      =      RSSI(strcmp(MacTransmitter,      ExpBSSIDAP1)      |
strcmp(MacTransmitter, ExpBSSIDAP2));

bNo = No(strcmp(MacTransmitter, ExpBSSIDAP1) | strcmp(MacTransmitter,
ExpBSSIDAP2));

bTime      =      Time(strcmp(MacTransmitter,      ExpBSSIDAP1)      |
strcmp(MacTransmitter, ExpBSSIDAP2));

bAp = zeros(length(bSSID), 1);
bAp(strcmp(bSSID, ExpBSSIDAP1)) = 1;
bAp(strcmp(bSSID, ExpBSSIDAP2)) = 2;
beacons = [bNo bTime bAp bRSSI];

% get beacons of both APs, excluding the rest
beacons_ap1 = beacons(beacons(:, 3) == 1, :);
beacons_ap2 = beacons(beacons(:, 3) == 2, :);

```

% Calculate time in reassociating with other AP

% 1st column: packet no. before AP change

```

apChange = find(diff(pAp));
if ~isempty(apChange)
    for i=1:size(apChange, 1)
        i_lower = apChange(i, 1);
        i_upper = i_lower + 1;
        time_lower = pTime(i_lower);

```

```

        time_upper = pTime(i_upper);
        % 2nd column: start of reassociation
        apChange(i, 2) = time_lower;
        % 3rd column: reassociation time
        apChange(i, 3) = time_upper - time_lower;
    end
end

window_size = 20;
figure(20)
% AP1 is the only one emitting
% Capturing a few beacons from AP1 means the STA performed the roaming
and
% scanned multiple channels. AP1 was emitting in one of those channels
if isempty(beacons_ap2) || size(beacons_ap2, 1) < window_size
    plot(beacons_ap1(:, 2), tsmovavg(beacons_ap1(:, 4), 's',
window_size, 1), 'b');
    leg = legend('AP1');
% AP2 is the only one emitting
elseif isempty(beacons_ap1) || size(beacons_ap1, 1) < window_size
    plot(beacons_ap2(:, 2), tsmovavg(beacons_ap2(:, 4), 's',
window_size, 1), 'r');
    leg = legend('AP2');
% both APs emitting
else
    plot(beacons_ap1(:, 2), tsmovavg(beacons_ap1(:, 4), 's',
window_size, 1), 'b', beacons_ap2(:, 2), tsmovavg(beacons_ap2(:, 4),
's', window_size, 1), 'r');
    leg = legend('AP1', 'AP2');
end
set(gca, 'FontName', 'Arial');
set(gca, 'FontSize', 12);
title('RSSI from both APs');
set(leg, 'location', 'best');

```

```

ylim([y_axis_rssi_lower y_axis_rssi_upper]);
xlim([x_axis_lower x_axis_upper]);
xlabel('Time (s)');
ylabel('RSSI (dBm)');
fig = gcf;
saveas(fig, strcat(filename_no_ext, '_rssi_both'), figs_ext);

packets_ap1 = packets(packets(:, 3) == 1, :);
packets_ap2 = packets(packets(:, 3) == 2, :);
figure(21)
% full time connected to AP1
if isempty(packets_ap2)
    plot(packets_ap1(:, 2), tsmovavg(packets_ap1(:, 4), 's',
window_size, 1), 'b');
    leg = legend('AP1');
% full time connected to AP2
elseif isempty(packets_ap1)
    plot(packets_ap2(:, 2), tsmovavg(packets_ap2(:, 4), 's',
window_size, 1), 'r');
    leg = legend('AP2');
% connected to both APs
else
    plot(packets_ap1(:, 2), tsmovavg(packets_ap1(:, 4), 's',
window_size, 1), 'b', packets_ap2(:, 2), tsmovavg(packets_ap2(:, 4),
's', window_size, 1), 'r');
    leg = legend('AP1', 'AP2');
end
set(gca, 'FontName', 'Arial');
set(gca, 'FontSize', 12);
title('RSSI from connected AP');
set(leg, 'location', 'best');
ylim([y_axis_rssi_lower y_axis_rssi_upper]);
xlim([x_axis_lower x_axis_upper]);

```

```

xlabel('Time (s)');
ylabel('RSSI (dBm)');
fig = gcf;
saveas(fig, strcat(filename_no_ext, '_rssi_connected'), figs_ext);

```

```

figure(22)
plot(packetsRxSort(:, 2), pdr(:, 1), 'b');
set(gca, 'FontName', 'Arial');
set(gca, 'FontSize', 12);
title('Packet Delivery Ratio (PDR)');
ylim([y_axis_pdr_lower y_axis_pdr_upper]);
xlim([x_axis_lower x_axis_upper]);
xlabel('Time (s)');
ylabel('Packet Delivery Ratio (%)');
fig = gcf;
saveas(fig, strcat(filename_no_ext, '_pdr'), figs_ext);

```

```

figure(23)
[hAx, hLine1, hLine2] = plotyy(packetsRxSort(:, 2), iat(:, 1),
packetsRxSort(:, 2), packetsLostSum(:, 1));
set(hAx(1), 'FontName', 'Arial');
set(hAx(2), 'FontName', 'Arial');
set(hAx(1), 'FontSize', 12);
set(hAx(2), 'FontSize', 12);
title('Inter Arrival Time & Packet loss');
set(hAx(1), 'ylim', [y_axis_iat_lower y_axis_iat_upper]);
set(hAx(2), 'ylim', [y_axis_packet_loss_lower
y_axis_packet_loss_upper]);
set(hAx(1), 'xlim', [x_axis_lower x_axis_upper]);
set(hAx(2), 'xlim', [x_axis_lower x_axis_upper]);
set(hAx(1), 'YTick',
[y_axis_iat_lower:y_axis_iat_interval:y_axis_iat_upper]);

```

```

set(hAx(2), 'YTick',
[y_axis_packet_loss_lower:y_axis_packet_loss_interval:y_axis_packet_lo
ss_upper]);
set(hLine1, 'Color', 'b');
set(hLine2, 'Color', 'r');
set(hAx(1), 'ycolor', 'b');
set(hAx(2), 'ycolor', 'r');
xlabel(hAx(1), 'Time (s)');
ylabel(hAx(1), 'Inter Arrival Time (ms)');
ylabel(hAx(2), 'Cumulative number of packets lost');
fig = gcf;
saveas(fig, strcat(filename_no_ext, '_iat_and_packet_loss'), figs_ext);

```

```

figure(24)
boxplot(packetsLost(:, 2));
grid on;
set(gca, 'FontName', 'Arial');
set(gca, 'FontSize', 12);
set(gca, 'xticklabel', {})
title('Lost Packets Distribution');
ylim([x_axis_lower x_axis_upper]);
ylabel('Time (s)');
fig = gcf;
saveas(fig, strcat(filename_no_ext, '_lost_packets_distribution'),
figs_ext);

```

```

disp(['Percentage of packets lost: ' num2str(packetsLostPctg) '% ('
num2str(size(packetsLost, 1)) ' out of ' num2str(size(packetsTxSortIds,
1)) ')']);

```

```

for i=1:size(apChange, 1)
    disp(['AP change #', num2str(i), ' started at time ',
num2str(apChange(i, 2)), ' s and took ', num2str(1000*apChange(i, 3)),
' ms']);

```

```

end

```