Universidad Zaragoza
1542

Trabajo Fin de Grado

Título del trabajo:
Agrupamiento de tramas en Wi-Fi: Mejora de la eficiencia en la red a través del mecanismo de agregación.

English tittle:
Wi-Fi Frame grouping: Improving network efficiency by using aggregation mechanisms

Autor/es

Cristian Hernández Fernández

Director/es

José María Saldaña Medina

Ponente

José Ruiz Más

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2015/2016

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. Cristian Hernández Fernández,

con nº de DNI 73027993-A 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) Agrupamiento de tramas en Wi-Fi: Mejora de la eficiencia en la red a través del mecanismo de agregación.

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

Zaragoza, 23 de Noviembre de 2016

Fdo: Cristian Hernández Fernández

Agradecimientos

A todas las personas que me han dedicado una parte de su tiempo para colaborar con la realización de este Trabajo Fin de Grado, sin vuestra ayuda hubiese resultado mucho más arduo.

En primer lugar, me gustaría agradecerles especialmente su ayuda a José María Saldaña y José Ruiz, mi director y ponente respectivamente, ya que durante el tiempo en el que se ha llevado a cabo este proyecto, siempre han estado dispuestos a resolver las dudas que iban apareciendo y me han dado valiosos consejos. Ha sido un placer trabajar con vosotros en un campo que me ha resultado muy interesante y del que gracias a este proyecto he podido investigar.

También me gustaría agradecer a mi familia, en especial a mis padres por el apoyo que me han brindado durante todos estos años.

Gracias también a todos los compañeros con los que he pasado incontables horas de estudio.

Muchas gracias.

Cristian Hernández Fernández

Resumen

Con la aparición de los *smartphone* y otros dispositivos portátiles que necesitan acceso a Internet, Wi-Fi (IEEE 802.11) se ha vuelto una tecnología muy popular. Pero su eficiencia es reducida: al transmitir las tramas en un medio compartido, se produce un retardo en cada acceso al medio. Por otra parte, 802.11 introduce un gran *overhead* que se añade al de IP, que estaba diseñado inicialmente para un entorno cableado.

En este TFG se estudian los efectos que se producen en una red Wi-Fi al implementar un mecanismo de agregación de tramas (A-MPDU, Aggregate MAC Protocol Data Unit). Al agregar tramas mejora la eficiencia, pues se introducen menos retardos debidos al acceso al medio, ya que se requieren menos tramas para enviar los mismos datos.

Este mecanismo fue propuesto por el IEEE en el estándar 802.11 versión *n* y consiste en unir varias tramas de nivel 2, añadiéndoles una cabecera de 4 bytes llamada MPDU *delimiter*. Este mecanismo es opcional en la versión *n*, pero en la siguiente versión (*ac*) es obligatorio su uso aunque únicamente se vaya a enviar una sola trama. Con esto se pretende mejorar la eficiencia de la red inalámbrica, ya que resulta muy baja especialmente para paquetes pequeños (pocas decenas de bytes).

Por otra parte, si usamos A-MPDU, también se reducirá el número de bytes que hace falta transmitir para confirmar los paquetes, ya que existe una trama (*Block* ACK) que permite confirmar hasta 64 subtramas de una sola vez. Por el contrario, si no se implementa este mecanismo, se envían tantos ACK como tramas de datos se hubiesen enviado. Esta reducción en los bytes es especialmente beneficiosa en los paquetes de pocas decenas de bytes, ya que puede llegar a ser de mayor tamaño el ACK que se necesita para confirmarlo (46 bytes), que los propios datos del paquete.

Otro beneficio que ofrece el uso de los mecanismos de agregación de tramas es que se reduce el número de cabeceras físicas (PLCP) que hay que enviar. Teniendo en cuenta que éstas se transmiten a una tasa muy baja (1 Mbps), la reducción del número de cabeceras físicas va a reducir significativamente el tiempo de transmisión de las tramas.

En este TFG se incluye en primer lugar un estudio teórico sobre los beneficios de la agregación de tramas A-MPDU, y se ha desarrollado también un programa que lo implementa en espacio de usuario. Usando dos máquinas Linux este programa permite ejecutar todo el proceso requerido para enviar A-MPDU reales, y así medir los ahorros obtenidos. Se ha comprobado que estos ahorros coinciden con los calculados teóricamente.

El presente TFG se enmarca dentro del proyecto Europeo H2020 Wi-5 (*What to do With the Wi-Fi Wild West*), en el que participa la Universidad de Zaragoza, junto con Telefónica I+D, TNO (Holanda), AirTies (Turquía) y la Universidad John Moores de Liverpool (coordinador).

Índice

1. Introducción

En los últimos años la tecnología IEEE 802.11 [1] (conocida popularmente como Wi-Fi) ha tenido una amplia difusión, llegando a muchos de los dispositivos que usamos habitualmente: ordenadores portátiles, *smartphone*, *tablet*. También muchas operadoras proporcionan a los usuarios un *router* para acceder a Internet, que lleva integrado un punto de acceso Wi-Fi. De esta manera, Wi-Fi se ha convertido en una de las tecnologías más usadas para acceder a Internet. Esto no se limita a las conexiones domésticas, sino que algunos operadores están realizando despliegues que integran un gran número de puntos Wi-Fi para cubrir un área (por ejemplo una estación, un aeropuerto, un centro comercial, o incluso una ciudad).

Wi-Fi está siendo actualmente una de las tecnologías de acceso a la web más utilizadas por los usuarios, pero su popularización hace que en algunas zonas la eficiencia baje, debido a varias razones:

- Wi-Fi se suele utilizar para transportar paquetes IP, una tecnología que no estaba pensada inicialmente para el uso inalámbrico ni para la movilidad. Por tanto, es necesario incluir mucha información extra en la cabecera.
- El medio inalámbrico puede producir un gran número de pérdidas de paquetes, dependiendo del entorno.
- El esquema de acceso al medio (CSMA-CA[1]) tiene una serie de tiempos de guarda, por utilizar un entorno compartido.

En este contexto, nuestro trabajo se centra en la mejora de la eficiencia de la red. En concreto, el objetivo de este proyecto es el de estudiar la mejora en la eficiencia que se consigue al agregar el tráfico a nivel 2, en concreto recurriendo al uso de A-MPDUs[2], una funcionalidad que se incluyó por primera vez en el estándar IEEE 802.11n, y sigue siendo utilizada en versiones posteriores como IEEE 802.11ac. Este mecanismo consiste en encapsular varias subtramas en una única trama.

El presente TFG se enmarca dentro del proyecto Europeo H2020 Wi-5 (*What to do With the Wi-Fi Wild West*)[3], en el que participa la Universidad de Zaragoza, junto con Telefónica I+D, TNO (Holanda), AirTies (Turquía) y la Universidad John Moores de Liverpool (coordinador). Dentro del Paquete de Trabajo número 3, la Tarea 3.3 está dedicada al estudio de la agregación de tramas como un método para optimizar el uso del espectro.

[1] CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance* o en español Acceso Múltiple con Escucha de Portadora y Prevención de Colisiones): es un protocolo de control de acceso a redes de bajo nivel, que permite que múltiples estaciones utilicen un mismo medio de transmisión.
[2] A-MPDU (*Aggregate MAC Protocol Data Unit)*: Trama que introduce el IEEE a partir de la versión *n* del protocolo 802.11, con el fin de reducir la carga de la red.
[3] Wi-5. What to do With the Wi-Fi Wild West, http://www.wi5.eu/, accedido Noviembre 2016.

1.1. Problemática y motivación

El diseño inicial de Internet no preveía la gestión de la cantidad de datos que actualmente envía y recibe un usuario. Además se diseñó para funcionar en entornos cableados, bien sea coaxial o par trenzado.

El paso del cable al espectro electromagnético en algunas partes de la red (red de acceso especialmente) hizo que aumentase el *overhead:* es decir, se tienen que emplear una gran cantidad de bits, que no forman parte de la información del usuario, para así reducir los errores que se producen al viajar toda la información por el medio inalámbrico.

También el acceso al medio es más complicado, por ser el espectro radioeléctrico un medio compartido, lo que conlleva unos retardos temporales necesarios para lograr el acceso al medio antes de transmitir cada trama.

Además, Wi-Fi utiliza las bandas de 2.4 GHz y 5 GHz, que son bandas libres, con lo que el espectro está bastante saturado en muchas zonas, como se puede ver en la Figura 1, obtenida en el laboratorio de Telemática mediante la aplicación Wifi Analyzer[4], o en la Figura 2 obtenida en un bloque de pisos de Zaragoza con la misma aplicación.

En ambas figuras se puede ver que incluso en un solo edificio pueden estar todos los canales disponibles ocupados.

Además, otros protocolos inalámbricos que no aparecen en la aplicación funcionan en esas mismas frecuencias (IEEE 802.15.1, *Bluetooth*), y también hay otros elementos que, aunque no relacionados con las comunicaciones, funcionan en esa frecuencia, como pueden ser los hornos microondas.

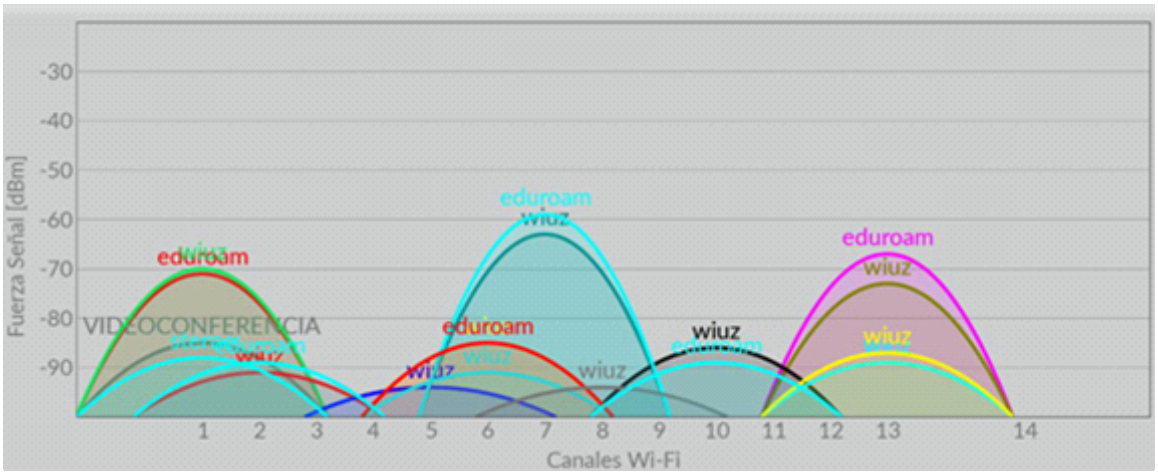


Figura 1: Ocupación de los canales Wi-Fi en 2.4 GHz. en el laboratorio de Telemática

[4] WiFi analyzer, https://play.google.com/store/apps/details?id=com.farproc.wifi.analyzer&hl=es, accedido Noviembre 2016.

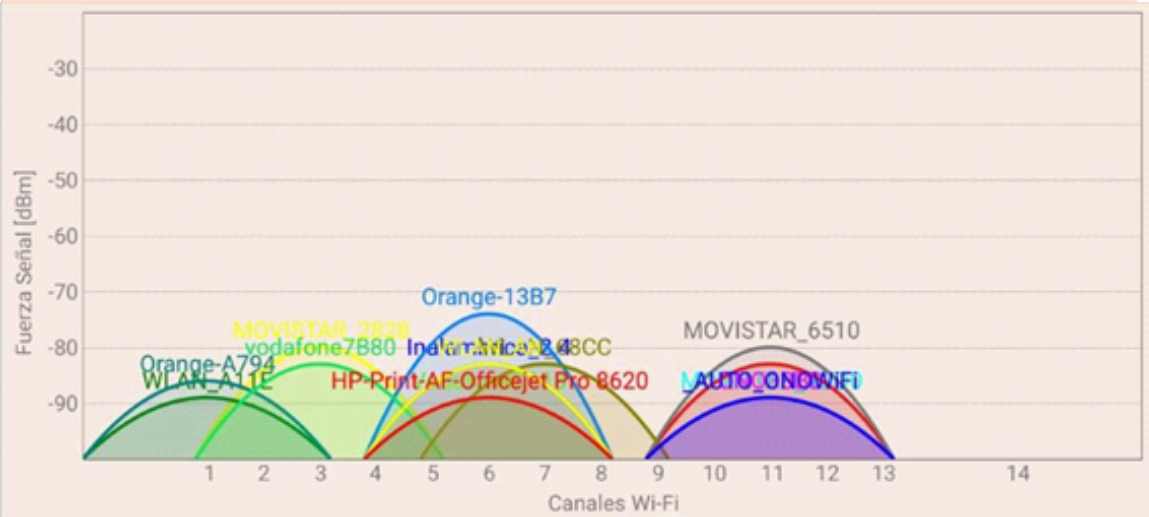


Figura 2: Ocupación de los canales Wi-Fi en 2.4 GHz. en un bloque de pisos de Zaragoza

Una de las maneras de mejorar la eficiencia de la red es mediante la reducción del número de tramas a enviar. Cada trama requiere una serie de mecanismos de acceso al medio, antes de ser enviada. Por tanto, si hay varios paquetes pequeños que se pueden transmitir conjuntamente, reduciremos el número de veces en que el mecanismo de acceso al medio se ejecuta.

Con este fin el IEEE lanzó unas modificaciones que se podían implementar en los equipos que usasen la versión *n* del 802.11, pero que no era obligatorio implementar. Estas modificaciones son los A-MPDU, A-MSDU[5], o ambas en conjunto (*Two Level Aggregation)*.

Posteriormente, en la versión 802.11ac del estándar, el uso de los A-MPDU es obligatorio: incluso cuando solo se envía un paquete, tiene que estar incluido en un A-MPDU.

El tamaño máximo de los paquetes que pueden circular por nuestra red es habitualmente de 1500 bytes, por ser el tamaño máximo de paquetes que permite Ethernet (MTU[6]).

Pero no son los paquetes del tamaño del MTU de la red los que van a sufrir una mejora usando los A-MPDUs, sino los paquetes más pequeños (pocas decenas de bytes), como los que se usan en aplicaciones de mensajería instantánea y juegos *online*, que cuentan con un gran número de usuarios en la actualidad.

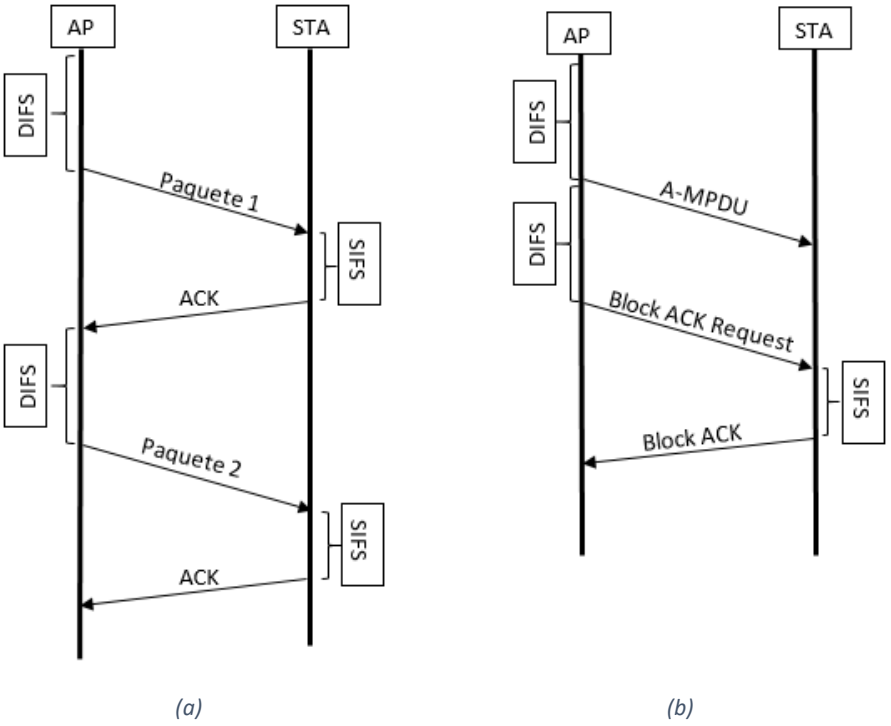
Esto se debe a que su *payload* es de un tamaño similar al de las cabeceras, por lo que resultan muy poco eficientes si se envían individualmente. Al agregarlos, la eficiencia puede aumentar significativamente.

Por otra parte, como Wi-Fi usa el espectro electromagnético, que es un medio compartido, debe implementar un mecanismo para el acceso al medio (CSMA-CA), que va a introducir un retardo por cada trama que se envíe, y uno menor por los ACK de nivel 2

[5] A-MSDU (*Aggregate MAC Servicel Data Unit)*: Otro tipo de trama que introduce el IEEE para reducir la carga en la red.
[6] MTU (*Maximum Transmission Unit*): Expresa el número máximo de bytes que puede tener un paquete para circular por la red, en los que se incluyen tanto los datos, como las cabeceras que tenga dicho paquete.

(cada trama requiere un paquete de reconocimiento, *acknowledgement* o ACK), lo que va reducir aún más la eficiencia de la red. En cambio, si implementamos un mecanismo de agregación, al reducir el número de tramas que es necesario enviar, el retardo debido al acceso al medio se reducirá en gran medida.

Esto se ilustra en la Figura 3, en la cual en el intercambio de paquetes del caso a) sería la manera en la que se accede el medio cuando no se usa un mecanismo de agregación de tramas: vemos que para enviar un paquete y recibir su confirmación va a haber un tiempo DIFS[7] + SIFS[8] más lo que tarden en viajar los paquetes por la red mientras que en b) el A-MPDU podría enviar dentro hasta 127 paquetes de los que aparecen en a), con lo que el retardo de acceso al medio es mucho menor.



(a) *(b)*

Figura 3: Esquema acceso al medio (a) sin y (b) con agregación

Por tanto, el uso de los A-MPDUs va a mejorar la eficiencia en la red, especialmente cuando se tengan que enviar muchos paquetes pequeños. Como contrapartida, se añadirá un retardo, ya que se deben almacenar los paquetes en un *buffer* hasta que se envíen juntos en una única trama. Esto produce que los primeros paquetes que entran en el *buffer* tengan un tiempo de servicio mayor que el que tendrían sin usar esta tecnología.

Otra ventaja del uso de los A-MPDUs es que cuando no se incorpora esta mejora se enviaría un ACK de nivel 2 por cada paquete (con un tamaño de 46 bytes). Sin embargo,

[7] DIFS (*DCF Interframe Space*): Tiempo que se debe esperar para poder enviar un paquete. En la versión del estándar IEEE 802.11n puede variar entre 28 y 50 microsegundos.
[8] SIFS (*Short Interframe Space*): Es el tiempo en microsegundos que tarda el receptor de un paquete en procesarlo y poder enviar la respuesta. Para el IEEE 802.11n es de 10 microsegundos.

cuando se usan A-MPDUs, todos estos ACK se sustituyen por un único *Block* ACK que confirma todos los paquetes que se han enviado en una A-MPDU.

1.2. Objetivos

El objetivo principal de este TFG es estudiar el beneficio que se obtiene por la agregación de tramas, tanto en la eficiencia de la red como en la reducción del *overhead*.

Para alcanzar este objetivo hemos necesitado estudiar cuáles eran las opciones que podíamos usar para agregar, y los requisitos que precisaba cada una de ellas.

Finalmente, se ha implementado un programa que permite ejecutar el mecanismo de agregación elegido con tráfico real. El programa está hecho en lenguaje C y en espacio de usuario, para poder así realizar las pruebas con independencia de los *driver* concretos (inyectando los paquetes en la red), y de esta manera poder realizar las medidas del ahorro que se produce.

1.3. Estructura de la memoria

La memoria consta de los siguientes apartados:

- En el primero comentaremos de forma general los mecanismos de agregación de tramas que propone el IEEE en el estándar 802.11, las tramas asociadas para este uso, así como las condiciones que se tienen que dar para que puedan funcionar estos mecanismos en una red.
- En el siguiente estudiaremos de forma teórica cuáles son los beneficios de implementar estos mecanismos en una red, tanto por reducción del número de los bytes que tenemos que trasmitir por la red para enviar varias cantidades de bytes de datos, como en la mejora de la eficiencia del canal inalámbrico al tener que enviar un número menor de tramas, considerando una tasa nominal constante.
- En el siguiente apartado explicaremos qué opciones hemos escogido para realizar la implementación del mecanismo de agregación de tramas, así como la explicación de los campos que hemos tenido que rellenar para poder construirlas.
- En la siguiente sección realizamos un estudio practico de cuál es la reducción en el número de bytes que se transmiten por la red cuando se ha implementado el mecanismo de agregación. Los resultados se comparan con los obtenidos de forma teórica en un apartado anterior.
- Y en el último apartado se comentan las conclusiones que hemos podido extraer a partir de los resultados obtenidos tras la implementación del mecanismo de agregación, y se proponen nuevas líneas de investigación relacionadas con el proyecto.

2. Agregación en Wi-Fi

2.1. A-MPDUs. Descripción y Características

Como se ha explicado, los A-MPDUs son un tipo de tramas que introdujo el IEEE en el estándar 802.11n con el objetivo de mejorar la eficiencia de la red.

Los A-MPDUs son tramas de nivel 2 (*Data Link*) y están formados por varios MPDU[9], que son paquetes de cualquier protocolo que vaya sobre el nivel 2 de Wi-Fi (*IPv4*, *IPv6*, *ARP*, etc.). Su estructura es la de la Figura 4.

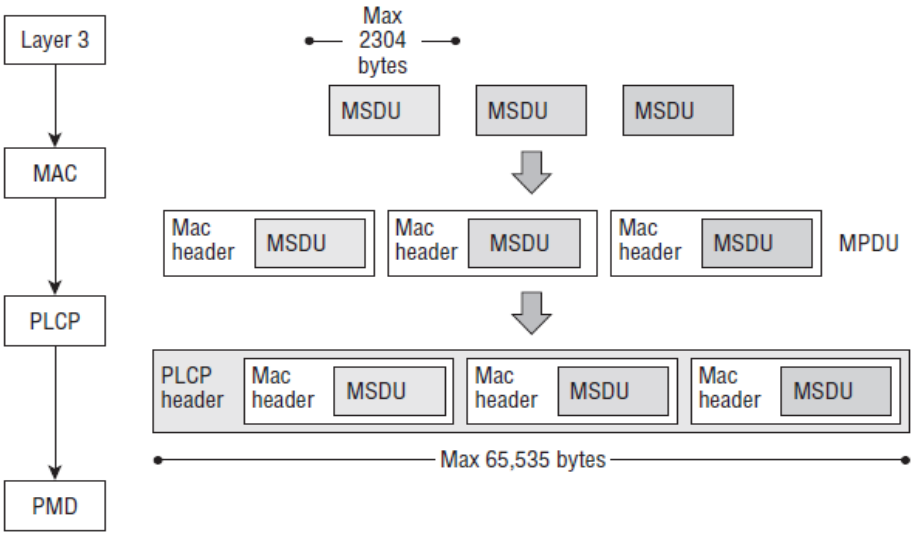


Figura 4: Estructura del A-MPDU en 802.11n[10]

En la Figura 4 se puede ver que los MPDU están formados por una cabecera MAC[11] y un MSDU, que corresponde al paquete que hemos comentado en el párrafo anterior

Las principales características de esta trama son [1] [2]:

- Si el cifrado está habilitado, cada MPDU está cifrada de forma individual
- Todos los MPDU pertenecientes a una A-MPDU deben tener la misma dirección de destino y el mismo tipo de *QoS*[12].
- El uso de A-MPDUs requiere el uso de *Block* ACK, que es la trama que se encarga de realizar la función de los ACK de nivel 2, con la diferencia de que con un *Block* ACK se pueden confirmar todos los MPDU que pertenecen a un A-MPDU.
- Los campos *duración/ID* de las cabeceras MAC de los MPDU llevan el mismo valor si pertenecen al mismo A-MPDU.

[9] MPDU (*MAC Protocol Data Unit*): Es un mensaje entre entidades MAC (de nivel 2) en un sistema de comunicaiones.
[10] Obtenida de https://mrncciew.com/2014/11/01/cwap-802-11-data-frame-aggregation/, accedido noviembre 2016.
[11] MAC (Media Access Control): Es una dirección única asociada a cada interfaz de red o bien puede estar referido a la cabecera que contiene estas direcciones cuando se envía un paquete.
[12] QoS (Quality of Service): Es el rendimiento que tiene una red. En este caso se refiere a un campo que va a determinar la prioridad que tiene esta trama para ordenarlas de esta manera.

- Todos los MPDU del A-MPDU que están protegidos tienen el mismo *Key ID*, que es un campo que se encarga del cifrado de las subtramas.

2.2. Diferentes opciones de agregación de tramas

En la Figura 5 podemos observar las dos opciones de agregación que nos ofrece el estándar Wi-Fi en la versión 802.11n. Podemos observar que la de A-MSDU está formada por una cabecera de nivel físico, una cabecera MAC, varias subtramas y el *FCS*[13], que sirve para comprobar que el A-MPDU que se recibe no tiene errores. Dentro de las subtramas se incluyen los campos *dirección destino*, *dirección origen*, *longitud*, el paquete que queramos encapsular (MSDU) y un *padding* para que sea múltiplo de 4 bits.

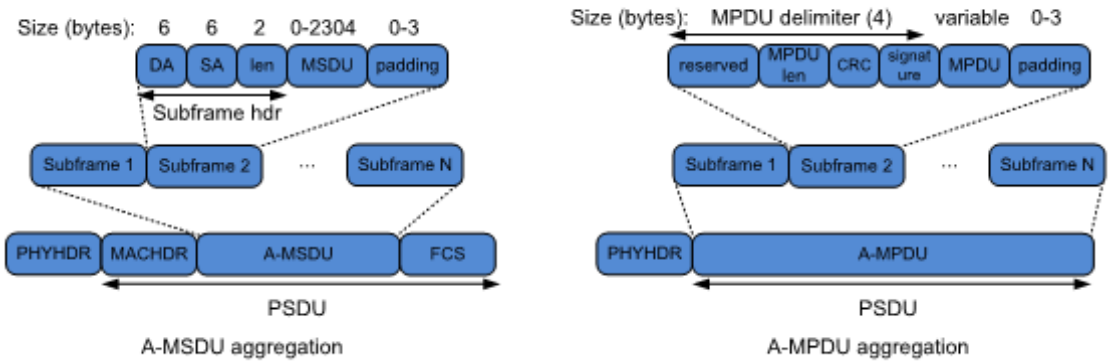


Figura 5: Opciones de agregación de tramas

El A-MPDU está formado únicamente por la cabecera de nivel físico y las subtramas, que contienen: un campo reservado; la longitud del paquete que estamos encapsulando; el *CRC*[14] del campo reservado y de la longitud; la firma de MPDU *Delimiter*, que es un valor fijo que aparece dado en el estándar para poder identificar las A-MPDUs; el paquete que vamos a incorporar al A-MPDU; y un relleno de ceros para que la longitud de la subtrama sea múltiplo de 4 bits.

Como conclusión de lo comentado en los párrafos anteriores acerca de la figura, observamos que el A-MSDU ofrece un mayor ahorro en el ancho de banda, ya que para el mismo número de paquetes a encapsular va a haber una única cabecera MAC, añadiendo cada MSDU únicamente las direcciones origen y destino. Por el contrario, en el A-MPDU cada paquete encapsulado deberá de incorporar su cabecera MAC.

Por otro lado, como el A-MSDU no tiene ningún tipo de CRC dentro de las subtramas, cuando se detecte un error no se podrá determinar en qué MSDU se ha producido, con lo que se tienen que descartar todos los paquetes que estaban agrupados en el mismo A-MSDU. En cambio, con el A-MPDU sólo se deberán retransmitir las tramas erróneas,

[13] FCS (*Frame Check Sequence):* Conjunto de bits para verificar la integridad de la información recibida.
[14] CRC (Verificación por redundancia cíclica): Es un código de detección de errores para detectar cambios accidentales en los datos.

ya que cada una tiene su CRC. Esto supone una gran ventaja, especialmente en entornos ruidosos con muchas pérdidas.

También existe una tercera opción de agregación que nos ofrece el estándar, conocida como *Two Level Aggregation* (Agregación a dos niveles), que se basa en encapsular A-MSDUs dentro de A-MPDUs.

Por este mismo motivo, las versiones posteriores del estándar han aumentado el soporte de las A-MPDU: en la versión 802.11ac es ya obligatorio el uso de este tipo de tramas, incluso si únicamente se va a enviar un paquete.

Por todas estas razones, en nuestro caso hemos decidido trabajar con A-MPDU, principalmente por la posibilidad que ofrecen de que cuando hay algún error en alguno de los paquetes encapsulados, sólo se deban descartar los erróneos.

2.3. La agregación de tramas en la literatura científica

En la literatura científica han aparecido diversos estudios y propuestas relacionadas con la agregación de tramas. Por ejemplo en [3], los autores presentan un modelo analítico en el cual estiman las mejoras de A-MPDU y A-MSDU. Se muestra un incremento significativo en la eficiencia de la red y unos mejores resultados para A-MPDU, especialmente cuando la tasa de error es alta.

En el artículo [4], los autores diseñaron un *scheduler* para A-MSDU y A-MPDU, que permite seleccionar los paquetes que van a ser agregados juntos, teniendo en cuenta que el estándar 802.11 únicamente especifica el formato de la trama, pero los mecanismos para decidir qué tramas viajan juntas dependen de lo que se decida en cada implementación. En el artículo los autores calcularon un tamaño óptimo de trama, y los paquetes eran agrupados intentando ajustarse a ese tamaño.

En el artículo [5], se optimiza el número de A-MSDUs, y se ajusta de forma dinámica de acuerdo al tamaño de la de la subtrama, llegando al nivel óptimo de *throughput* y mejorando así la eficiencia MAC.

Aunque no esté directamente relacionado con los resultados a estudiar en nuestro proyecto (la mejora que tiene el uso de estos mecanismos en la eficiencia de la red), parece interesante comentar que el uso de A-MPDUs puede tener algunas vulnerabilidades, como el *Beacon Injection*, como se explica en el artículo [6].

2.4. Intercambio de paquetes necesario para el uso de A-MPDUs

Para poder conectarnos a una red Wi-Fi, enviar A-MPDUs y que nos los puedan confirmar con *Block* ACK, es necesario que haya un intercambio de tramas concreto. Este tiene una parte común con la conexión a una red Wi-Fi sin agregación. La diferencia es que, dentro de las tramas, hay parámetros que dan información acerca de las capacidades para agregar

que tienen tanto el AP[15] (Punto de Acceso, Access Point) como la STA[16] (Estación, *Station*).

El esquema del intercambio de tramas que tiene que ocurrir para poder enviar A-MPDUs en un sentido consta de dos partes diferenciadas: una que es común con la situación habitual (Figura 6), que corresponde a la conexión a la red (descubrimiento, autenticación y asociación) y otra propia del envío de A-MPDUs, que veremos después (Figura 9).

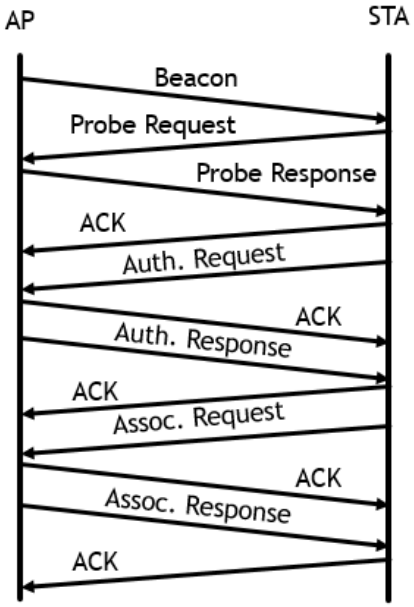


Figura 6: Esquema de la conexión a la red

Como se ve en el esquema de la Figura 6, para conseguir conectarse a una red, el cliente tiene que esperar a recibir un *Beacon*, que sirve para anunciar la red. Este tipo de paquetes se envían a la dirección *Broadcast* para que los dispositivos puedan saber en qué canal está, su *SSID*[17], la información acerca de la seguridad que usa y el resto de las capacidades que tiene la red. La estructura general de un *Beacon* puede verse en la Figura 7.

Radiotap Header	IEEE 802.11 MAC Header	IEEE 802.11 Management - Beacon

Figura 7: Estructura General de un Beacon

Tras el envío del *Beacon*, si una estación se quiere conectar a la red, deberá enviar un *Probe Request* a la dirección que ha obtenido del *Beacon* y con los parámetros que ha pasado (canal, *rate*, etc.).

[15] AP (*Access Point*): Es el dispositivo que nos permitiría conectarnos a una red o salir a Internet.
[16] STA: Son los dispositivos que se conectan a un AP para estar conectados a la red que el AP les proporcione acceso. En la literatura científica de este tema se suele emplear el término "STA" para referirse a los dispositivos que se conectan a un punto de acceso.
[17] SSID (*Service Set Identifier*): Nombre incluido en todos los paquetes de una red inalámbrica para identificarlos como parte de esa red.

Cuando el AP recibe el *Probe Request*, debe de crear un *Probe Response* donde le confirma que le escucha. Tras esto el cliente envía un *Authentication Request* en el que se proporciona la contraseña en el caso de que la red la requiera, y el AP le deberá contestar con un *Authentication Response* en el que confirmará si los datos que ha introducido en el *Request* son los correctos para conectarse. Tras esto el cliente se intenta conectar finalmente a la red mediante el envío de un *Association Request* y si el punto de acceso se lo permite le contestará con un *Association Response*.

Hasta aquí llega el proceso que tiene que seguir cualquier cliente para conectarse a una red Wi-Fi. Si entonces se quiere utilizar el mecanismo de agregación de tramas mediante A-MPDUs, necesitamos otras tramas (Figura 8). El emisor debe enviar al receptor el *AddBa Request*[18], que es una trama de tipo *Action* donde se negocia si el receptor puede recibir los A-MPDUs, y las características que estos van a tener (número máximo de tramas que van a tener agregadas, si pueden contener A-MSDUs, y otras opciones relacionadas).

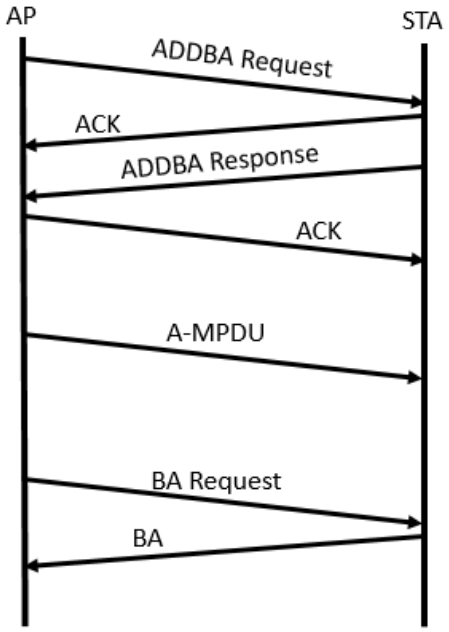


Figura 8: Esquema tramas para envió A-MPDU

Si el receptor tiene el hardware y software necesarios para decodificar este tipo de tramas, deberá de responder con una trama de *AddBa Response*[19], y a partir de este punto ya será posible el envío de A-MPDUs en un sentido. Si se quisiera que el intercambio fuese bidireccional, habría que hacer esto en los dos sentidos.

[18] *AddBa Request* (o *Add Block* ACK *Request*): Trama que sirve para la negociación de *Block* ACK y por tanto de A-MPDUs, en las que se comparten las características que se quieren/pueden usar con respecto a los *Block* ACK.

[19] *AddBa Response* (o *Add Block* ACK *Response*): Trama de respuesta al *AddBa Request* donde el receptor de este dice cuáles son sus capacidades con respecto a trabajar con *Block* ACK y si está de acuerdo con las que ha recibido se podrá a empezar a usar *Block* ACK en el sentido desde el que ha enviado el *Request* al que ha enviado el *Response*.

2.5. Estructura del A-MPDU

Los A-MPDUs tienen una estructura concreta que es la que permitirá que el receptor (en el caso de tenerlo implementado y haberlo negociado con el emisor siguiendo los pasos que hemos explicado anteriormente) detecte que es este tipo de trama.

Por este motivo el A-MPDU tiene una estructura propia definida en el estándar del IEEE, que se muestra en la Figura 9:

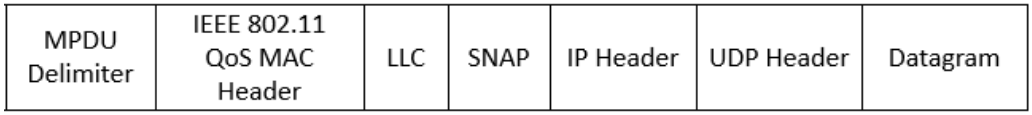


Figura 9: Estructura del A-MPDU

La estructura desde el MPDU *Delimiter* hasta el final se repetirá tantas veces como paquetes de datos queramos agregar en un único A-MPDU, teniendo en cuenta las limitaciones del MTU de la red en la que esté operando.

El MPDU *Delimiter* se compone de cuatro campos, y es el que va a permitir al receptor saber que se trata de un A-MPDU, así como separar cada una de las subtramas que están agregadas en el mismo A-MPDU:

- El primer campo es de 4 bits y está reservado, con lo que se pone todo a ceros.
- En el segundo campo se indica la longitud del MPDU en octetos y tiene un tamaño de 12 bits.
- El tercer campo está formado por 8 bits donde se especifica el *CRC* de los 16 bits anteriores, es decir de los 4 que están reservados y los 12 de la longitud.
- En el cuarto y último lugar hay un campo de 8 bytes que es el de *Delimiter Signature* que tiene un valor prefijado por el estándar que es el de una ene mayúscula 'N', que en valor hexadecimal corresponde con el valor '4E'.

Así que el receptor lo que hace para encontrar los MPDUs que componen cada A-MPDU es buscar el carácter 'N' y cuando lo ha encontrado mira el valor que toma lo que debería corresponder con la longitud y calcula su *CRC.* Si coincide con lo que hay en el campo de *CRC,* da por sentado que es un MPDU que está formando parte de un *A*-MPDU.

Después de haber enviado el MPDU *Delimiter*, todo el resto del MPDU es como un paquete de datos normal, con las siguientes diferencias:

- En primer lugar en una cabecera MAC que además en este caso va a soportar calidad de servicio, con lo que será igual que las que hemos visto anteriormente, únicamente cambiará:
  - El tipo que será *data*.
  - El subtipo que será *QoS.*
- Después de las direcciones MAC en vez de aparecer directamente el número de secuencia, se especificará el nivel de calidad de ese MPDU. Esto se debe de tener en cuenta antes de la creación de los A-MPDU, ya que al enviarse todos los

MPDUs que lo integran al mismo tiempo, tienen que tener la misma calidad de servicio.

- Finalmente, indicaremos que lo que viene a continuación es *IPv4* y rellenaremos la cabecera IP, la del protocolo de transporte que decidamos usar (UDP o TCP[20]) y los datos que queramos enviar.

2.6. Confirmación de los A-MPDU

Como hemos comentado antes, para confirmar los A-MPDUs se necesita una trama especial: el *Block* ACK, que puede ser *inmediate* o *delayed*. En el primer caso, se envía el *Block ACK* justo después de recibir el A-MPDU, mientras que en el segundo se puede enviar más tarde.

Esto permite reducir el ancho de banda consumido, ya que el *Block* ACK tiene un tamaño inferior al que tendrían los ACK que puede sustituir (hasta 64 ACK), esto en el caso de que haya pocos paquetes agrupados dentro del A-MPDU no producirá una gran reducción del ancho de banda que se requiere.

Cuando el *Block ACK Request* llega al receptor, este responde transmitiendo el *Block* ACK, donde indicará qué tramas a partir de la siguiente ha recibido correctamente. Este, mediante el campo del *BitMap*, nos indicará el resultado, marcando a 1 los bits correspondientes de los paquetes que haya recibido a partir del indicado en el *Block ACK Request*.

Por ejemplo si únicamente ha llegado el cuarto paquete, en el *BitMap* aparecerá un 1000000000000000, donde el 1 que es hexadecimal se correspondería con un 0001 en binario, que significa que no hemos recibido los tres primeros paquetes que esperábamos pero sí el cuarto.

En el caso que se va a ver en el círculo rojo de la Figura 10 está todo a ceros, que significa que ningún paquete ha llegado.

[20] TCP (*Transmission Control Protocol*): Protocolo que sirve para el envío de datos en el cual se tienen que confirmar los paquetes recibidos.

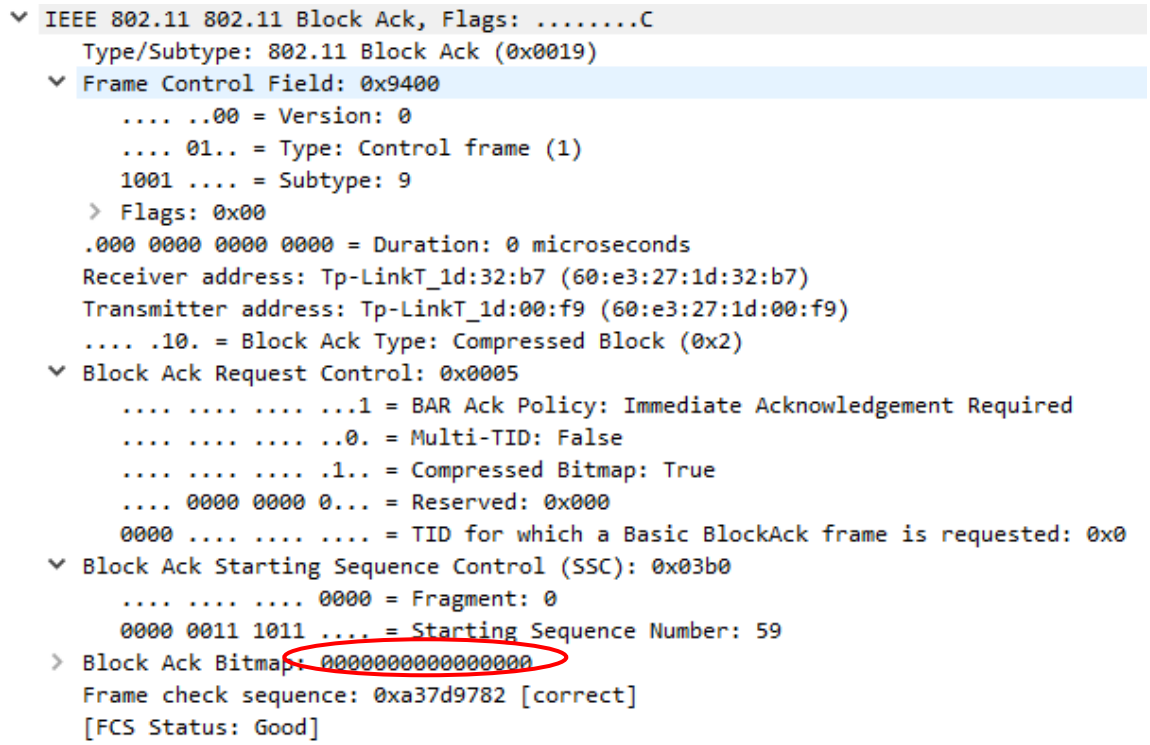


Figura 10: Block ACK

2.7. Uso de Radiotap para inyectar y recibir tráfico

Radiotap[21] es un estándar *de facto* para la inyección y recepción de tramas en 802.11 que funciona para distintos sistemas operativos (FreeBSD, Linux, NetBSD, OpenBSD y Windows con AirPcap).

La cabecera Radiotap permite obtener información adicional acerca de las tramas recibidas (canal, potencia, etc.) y especificar los parámetros de las tramas a enviar. Ofrece más flexibilidad que otras cabeceras de nivel físico como puede ser la cabecera *Prism*[22], ya que permite al desarrollador especificar un número de campos mediante una máscara de bits que se encuentra en la cabecera.

[21] Información obtenida en http://www.radiotap.org/, accedido noviembre 2016.
[22] LINKTYPE_IEEE802_11_PRISM: http://www.tcpdump.org/linktypes/LINKTYPE_IEEE802_11_PRISM.html, accedido noviembre 2016.

La estructura de la cabecera Radiotap es la que aparece en la Figura 11:

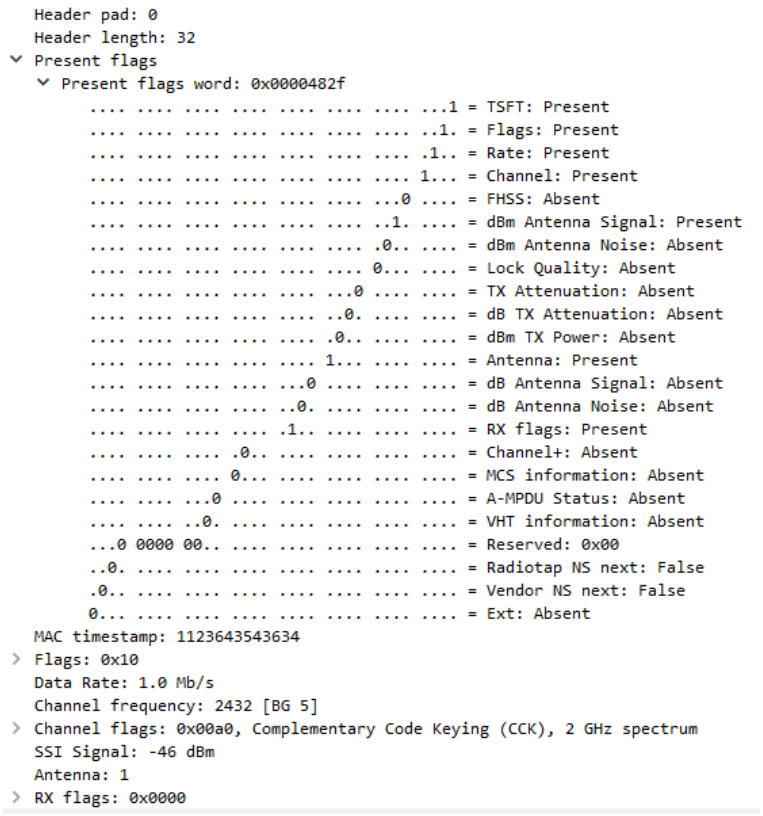


Figura 11 Estructura del Radiotap Header

3. Estudio teórico de los beneficios de la agregación

En esta sección se presentan dos estudios diferentes:

- En el primero se muestra cuál va a ser la reducción en la cantidad de información (medida en bytes) transmitida, cuando implementamos el mecanismo de agregación (A-MPDU), comparándolo con la solución sin agregación. Para ello se utilizarán paquetes de diferentes tamaños.
- En el segundo se estudia el aumento en la eficiencia del canal inalámbrico conforme decidamos agregar más subtramas. También se repetirá para distintos tamaños de paquetes.

Para ambos estudios se considera que la tasa nominal de la red va a ser constante y con valor de 130 Mbps

3.1. Estudio del número de bytes a transmitir por la red

En esta sección se presenta un estudio teórico de la reducción en bytes transmitidos. Se utilizarán paquetes de distinto tamaño, y se agregarán diferentes números de paquetes usando el mecanismo de agregación A-MPDU.

Para la realización de este estudio y de los posteriores hemos decidido que todos los paquetes que agregamos en un mismo A-MPDU sean del mismo tamaño, ya que así es más fácil ver la diferencia con respecto al modo nativo (no se usa el mecanismo de agregación). No obstante, esta tecnología permite que paquetes de distinto tamaño se integren en un mismo A-MPDU. Estos mismos tamaños los usaremos en la siguiente sección para las pruebas con tráfico real, donde podremos confirmar los resultados.

Como valor más pequeño elegimos un paquete IP de 50 bytes, que es tamaño que corresponde a una muestra de audio codificada con el códec G.729: 20 bytes de cabecera IP, 8 bytes de cabecera UDP y 22 de datos UDP, que corresponden a una cabecera RTP[23] (12 bytes) más los 10 bytes de una muestra de audio.

Como valor más grande elegimos 528 bytes, porque es un valor lo suficientemente grande como para poder agregar algunas tramas y llegar al MTU de la red, y así ver cuál es el efecto de la agregación de tramas en paquetes de mayor tamaño.

Para cada tamaño de paquete, haremos dos cálculos:

- Modo nativo (cada paquete viaja en una trama). Al tamaño del paquete (incluyendo el *payload* y las cabeceras UDP e IP) le añadiremos el de la cabecera MAC y con eso obtendremos la numero de bytes que se van a usar para enviar los datos en modo nativo. Por otro lado calcularemos los bytes necesarios para la confirmación de los datos, en este caso es únicamente el tamaño del ACK de nivel 2. Los valores calculados se multiplicaran por el número de paquetes y con eso

[23] RTP (*Real Time Protocol)*: Es un protocolo de nivel de sesión utilizado para la transmisión de información en tiempo real.

obtendremos el número de bytes que se usan para enviar los datos y cuántos para confirmarlos.

- A-MPDU (todos los paquetes viajan juntos). En este caso el número de bytes que se usan para enviar las tramas de datos, será el tamaño del paquete a nivel IP, al que hay que añadirle la cabecera MAC, un *padding* de ceros para que todo lo anterior sea múltiplo de 4 y el MPDU *delimiter*. Este valor se deberá multiplicar por el número de subtramas que queramos agregar en el A-MPDU. Sin embargo, el número de bytes que se emplean para la confirmación de los datos es constante, ya que únicamente estarán integrados por la suma de los bytes del *Block* ACK *Request* y del *Block* ACK, salvo en el siguiente caso: como el *Block* ACK puede confirmar hasta 64 subtramas, si el número de subtramas que tiene el A-MPDUs supera este número, se necesitaran varios. También se necesitaran varios *Block* ACK cuando haya que enviar varios A-MPDUs. Esto puede ocurrir cuando queramos agregar más de 127 subtramas; si el A-MPDU cuando vamos a agregar un paquete más supera 65.535 bytes que es el tamaño máximo que pueden alcanzar; o si el tamaño del A-MPDU supera el MTU de la red que, al ser un estudio teórico, consideraremos infinito.

Estudio con paquetes de 50 bytes

A partir de lo comentado anteriormente obtenemos las siguientes figuras: la Figura 12 muestra el número de bytes transmitidos por el emisor de los paquetes de 50 bytes. La Figura 13 da cuenta del número de bytes incluidos en los ACK de nivel 2. Y la Figura 14 muestra el número total de bytes requeridos (es decir, la suma de las dos anteriores).

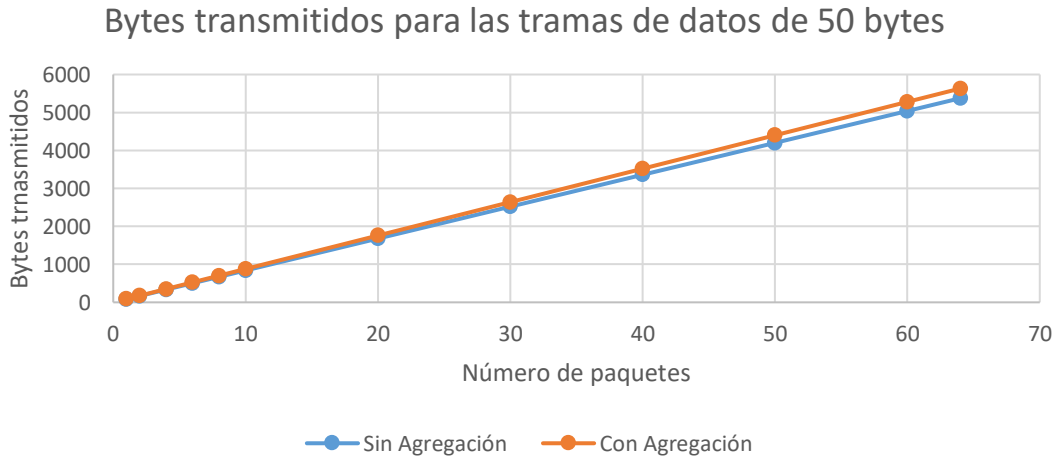


Figura 12: Bytes transmitidos con y sin agregación para distinto número de paquetes de datos

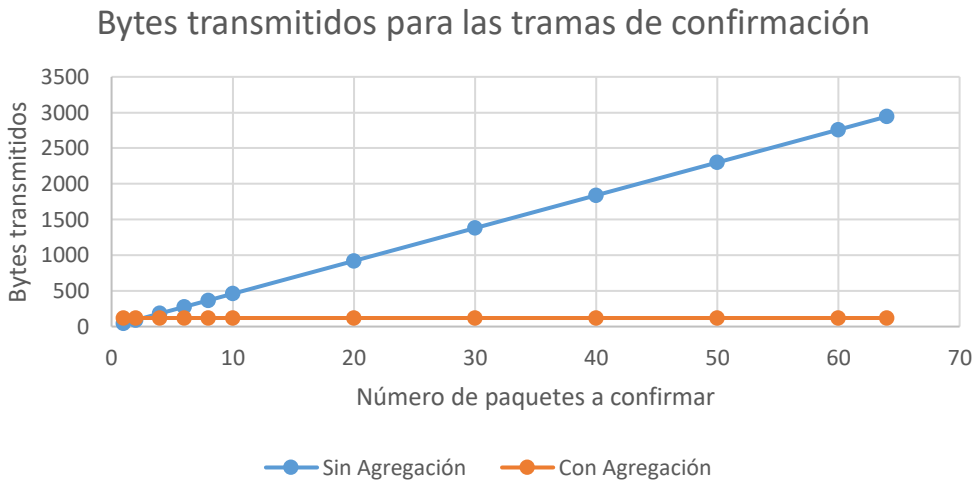


Figura 13: Bytes transmitidos con y sin agregación para los paquetes de confirmación

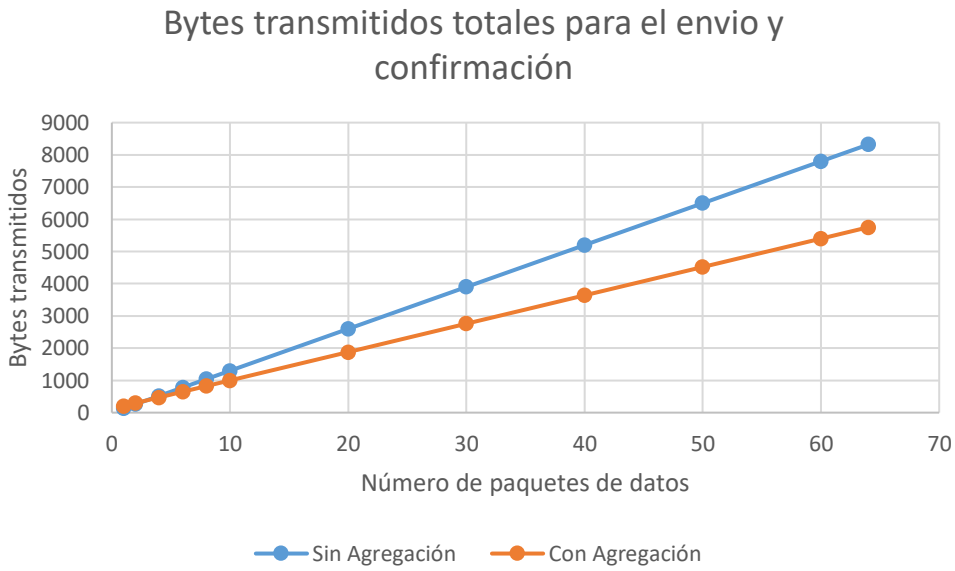


Figura 14: Bytes transmitidos con y sin agregación para todos los paquetes

En la Figura 12 se puede observar cuántos bytes se necesitan para enviar un número de paquetes (tramas en modo nativo o subtramas en A-MPDU) con o sin agregación. Como podemos ver, enviarlas con agregación conlleva un ligero aumento en los bytes que hay que enviar: 4 bytes por cada paquete debido al MPDU *delimiter*. Sin embargo en la Figura 13 podemos ver que, en el caso de las tramas de confirmación, a partir de un cierto número de paquetes, 3 en este caso, el número de bytes necesarios para confirmar las tramas usando A-MPDUs es inferior al modo nativo. Esto se debe a que, mientras que en el modo nativo se tiene que enviar un ACK por cada paquete, en el A-MPDU se pueden confirmar todos con el uso del *Block* ACK.

En la Figura 14 se muestra la suma de los bytes usados para enviar las tramas de datos y para las tramas de confirmación, y se puede observar que usar que el uso de A-MPDU va a conllevar un ahorro en el número de bytes que es necesario transmitir. Este ahorro supone un 30,86% de bytes si enviamos un único A-MPDU, además de la ventaja de no

habrá apenas retardo debido al acceso al medio, aunque aparecerá el retardo debido a esperar a que todos los paquetes estén listos para poder encapsularlos en el A-MPDU.

Debido a que la cabecera física (PLCP[24]) se transmite a una tasa (*rate*) mucho menor (1Mbps) que el que hemos supuesto en el escenario (130Mbps), es muy interesante reducir el número de tramas a enviar ya que, como se puede ver en la Figura 15, el tiempo de transmisión de las tramas es mucho menor cuando se usa agregación. La figura se ha calculado de la siguiente manera:

- Modo nativo:

$$\frac{Número\ de\ paquetes\ de\ datos \times L_{PLCP}}{Tasa_{PLCP}} + \frac{Número\ de\ paquetes\ de\ datos \times L_{datos}}{Tasa_{Datos}}$$

- A-MPDU:

$$\frac{L_{PLCP}}{Tasa_{PLCP}} + \frac{Número\ de\ paquetes\ de\ datos \times L_{datos}}{Tasa_{Datos}}$$

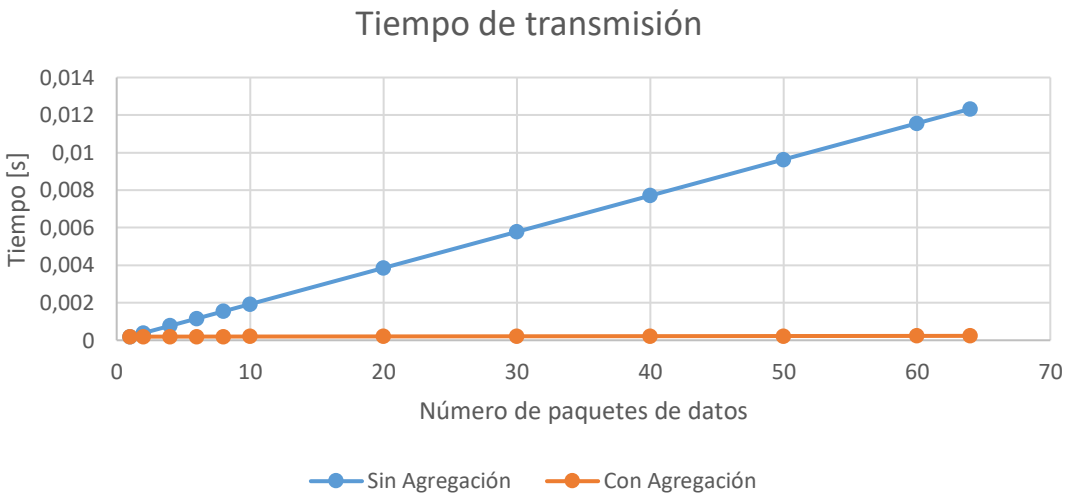


Figura 15: Tiempo de transmisión de las tramas de datos con y sin agregación

Estudio con paquetes de 78 bytes

Una vez visto el ahorro obtenido con paquetes de 50 bytes, pasamos a realizar los cálculos con paquetes con 78 bytes a nivel IP (50 bytes de datos UDP), en los que obtenemos lo que aparece en la Figura 16:

[24] PLCP (*Physical Layer Convergence Procedure*): Protocolo de la capa física que se usa en las transmisiones de datos.

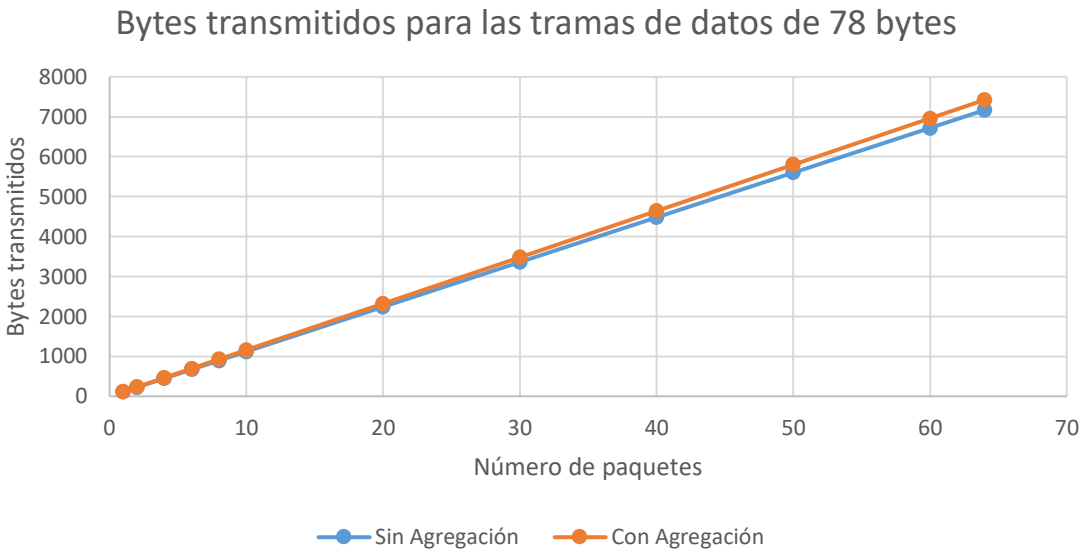


Figura 16: Bytes transmitidos con y sin agregación para distinto número de paquetes de datos

Los bytes que se usan para las tramas de confirmación son los mismos que para el caso anterior (Figura14). Sumando los bytes que se usan para las tramas de datos con los bytes que su usan para las tramas de confirmación obtenemos una reducción en los bytes a transmitir del 30,87% cuando su usa A-MPDU.

Estudio con paquetes de 228 bytes

Hacemos de nuevo las cuentas para paquetes de 228 bytes (200 bytes de datos UDP), obteniendo los resultados de la Figura 17:

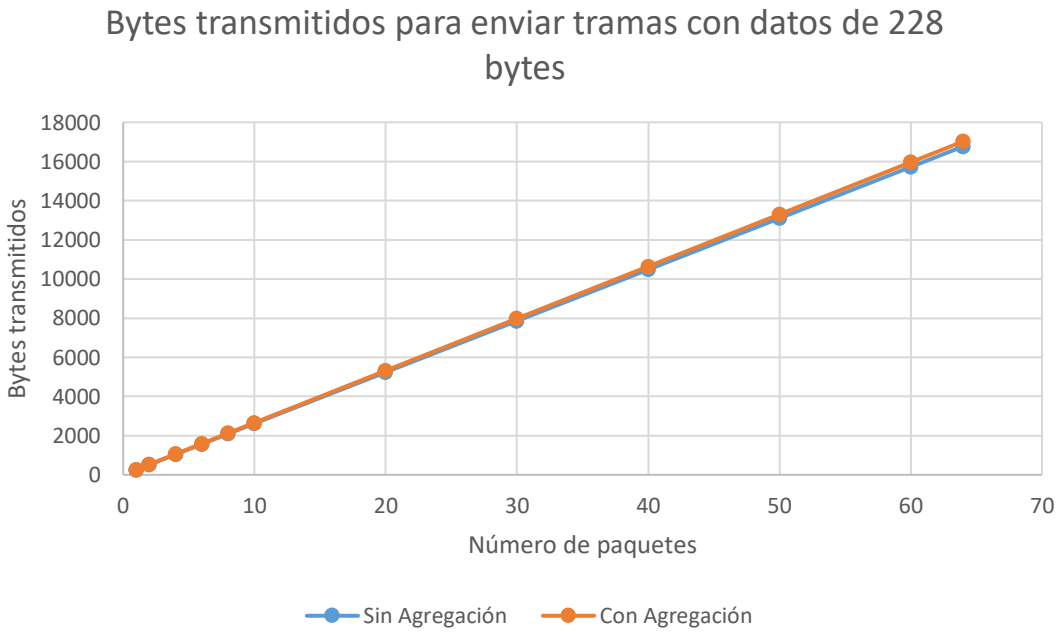


Figura 17: Bytes transmitidos con y sin agregación para distinto número de paquetes de datos

En este caso el número de bytes para las tramas de confirmación coincide con los anteriores (Figura 14), con lo que al igual que en los casos anteriores, si sumamos los

bytes que se necesitan transmitir para enviar las tramas de datos y confirmación con y sin agregación, la reducción máxima que se obtiene usando A-MPDU es del 13,03%.

Estudio con paquetes de 528 bytes

Por último calculamos lo mismo con paquetes de 528 bytes (500 bytes de datos UDP), y obtenemos lo que aparece en las Figuras 18 y 19:

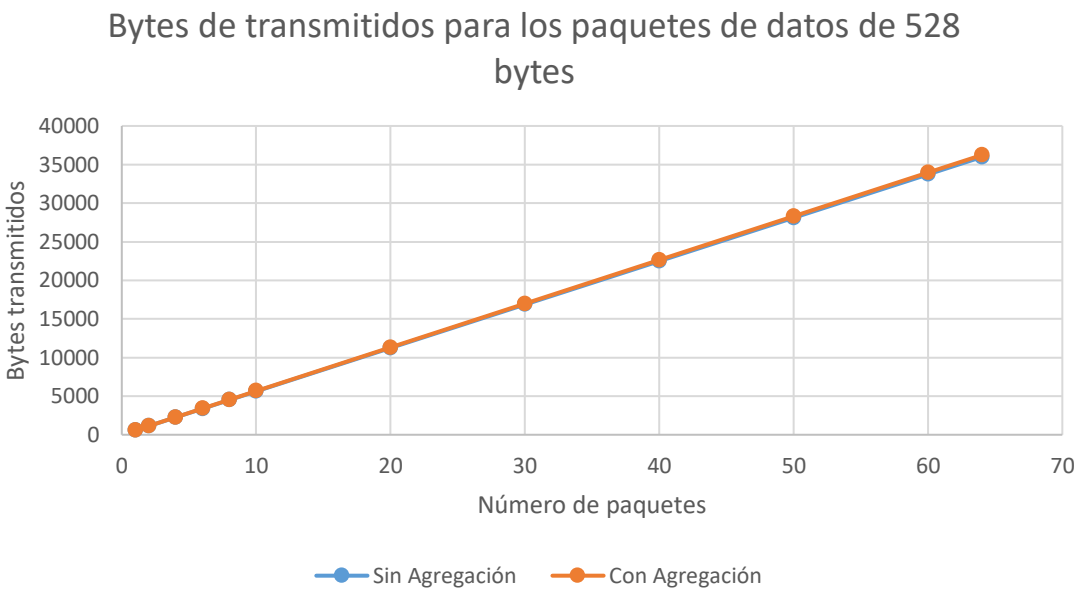


Figura 18: Bytes transmitidos con y sin agregación para distinto número de paquetes de datos

De nuevo el número de bytes que se usan para las tramas de confirmación es el mismo que para los casos anteriores (Figura 14). Por lo que este caso la suma de todos los bytes que se transmiten es la que aparece en la Figura 19.

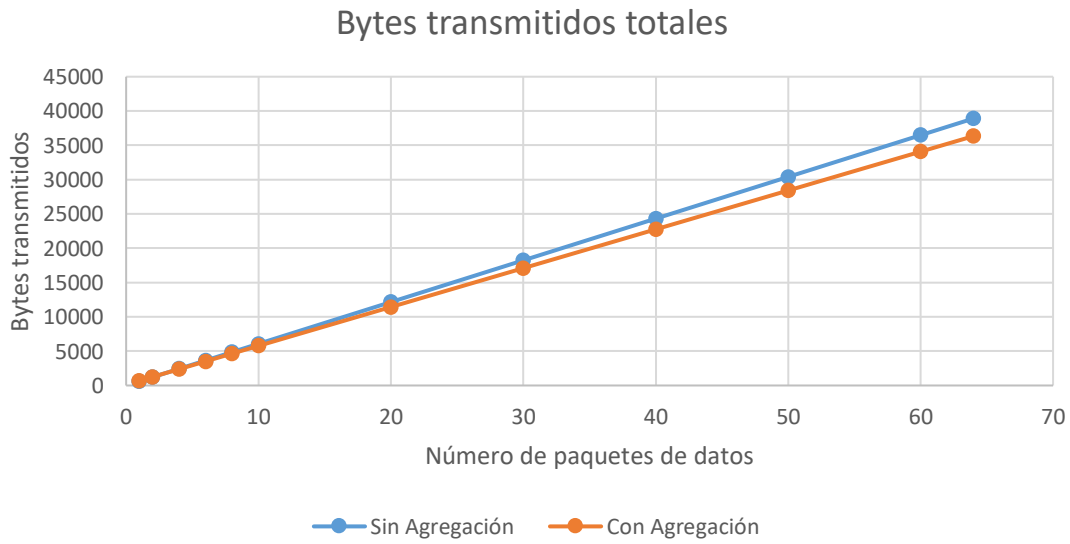


Figura 19: Bytes transmitidos totales para enviar paquetes de 528 de datos con y sin agregación

En este caso el ahorro de bytes a transmitir que obtenemos es del 6,6%.

Por lo cual se puede concluir que cuanto más pequeños son los paquetes que vamos a agregar, más se consigue ahorrar con este mecanismo. Esto se debe a que el *overhead* que se introduce en un paquete no depende del tamaño de los datos del mismo, con lo que a los pequeños les afecta más.

El tiempo de transmisión que se obtiene para paquetes de este tamaño se puede ver en la Figura 20. Como podemos observar, si usamos A-MPDU, aunque con este tamaño la reducción de bytes es muy inferior a la que obteníamos con los paquetes de 50 bytes, el tiempo de transmisión sigue siendo muy inferior al que se obtiene mediante el método nativo, ya que lo que más le está afectando es la diferencia de tasas entre la usada para los datos y la de la cabecera física.

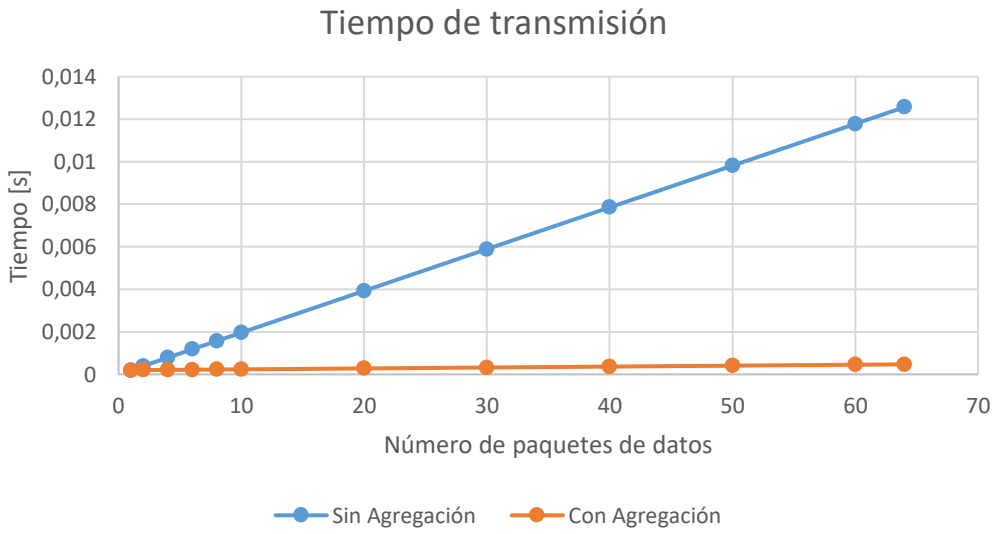


Figura 20: Tiempo necesario para transmitir con y sin agregación paquetes de 528 bytes de datos

3.2. Estudio de la eficiencia del canal inalámbrico

Hasta ahora se ha calculado la reducción en función del número de bytes a nivel MAC, y del tiempo de transmisión de esos bytes. Pero hay que tener en cuenta que 802.11 es un protocolo inalámbrico con un mecanismo de acceso al medio CSMA-CA. Por tanto, para calcular la eficiencia no debemos limitarnos al número de bytes a enviar, sino que hay que tener en cuenta las tasas, los tiempos de guarda, preámbulos, etc.

Por tanto, en esta sección, basándonos en el artículo [3], procedemos a calcular la eficiencia del canal inalámbrico, es decir, la relación que hay entre el *data rate* (tasa a nivel IP) y el *rate* a nivel físico (la tasa nominal), que para nuestro estudio teórico suponemos de 130 Mbps como se ha comentado previamente. Se tienen en cuenta los valores reales de los tiempos de guarda (DIFS, SIFS), la tasa a la que se envía cada parte de la trama, etc.

Estudiaremos cuál va a ser la eficiencia del canal inalámbrico en cuatro opciones de agregación de tramas (A-MPDU con TCP, A-MPDU con UDP, A-MSDU con TCP y A-MSDU con UDP), pero centrándonos especialmente en los A-MPDU.

Estudio con paquetes de 50 bytes

Comenzamos con un valor de 50 bytes (Cabecera IP + Datos IP), que se corresponde con el primer tamaño usado en el estudio anterior.

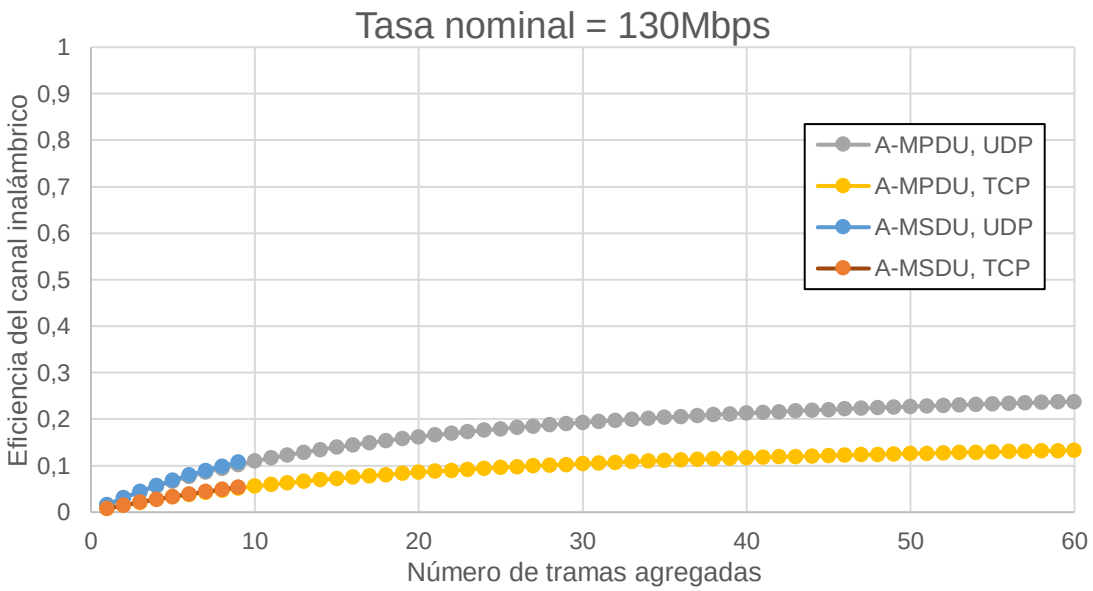


Figura 21: Eficiencia del canal inalámbrico con paquetes de 50 bytes

En la Figura 21 podemos observar que si se usa A-MPDU para agregar 60 paquetes UDP, la eficiencia aumenta del 1,7% a un 23,7%. Si se usa TCP la eficiencia aumenta del 0,75% a un 13,27%.

Si se usa A-MSDU ya no nos permite agregar tantas subtramas en una única trama, como se puede ver en la figura: únicamente podemos llegar a agregar 9 tramas, y usando UDP como protocolo de transporte obtenemos una eficiencia de 10,75% mientras que usando TCP del 5,35%

Estudio con paquetes de 78 bytes

Pasamos a calcular ahora la eficiencia si los MPDU tienen un tamaño de 78 bytes, que equivale a 50 bytes de datos UDP.

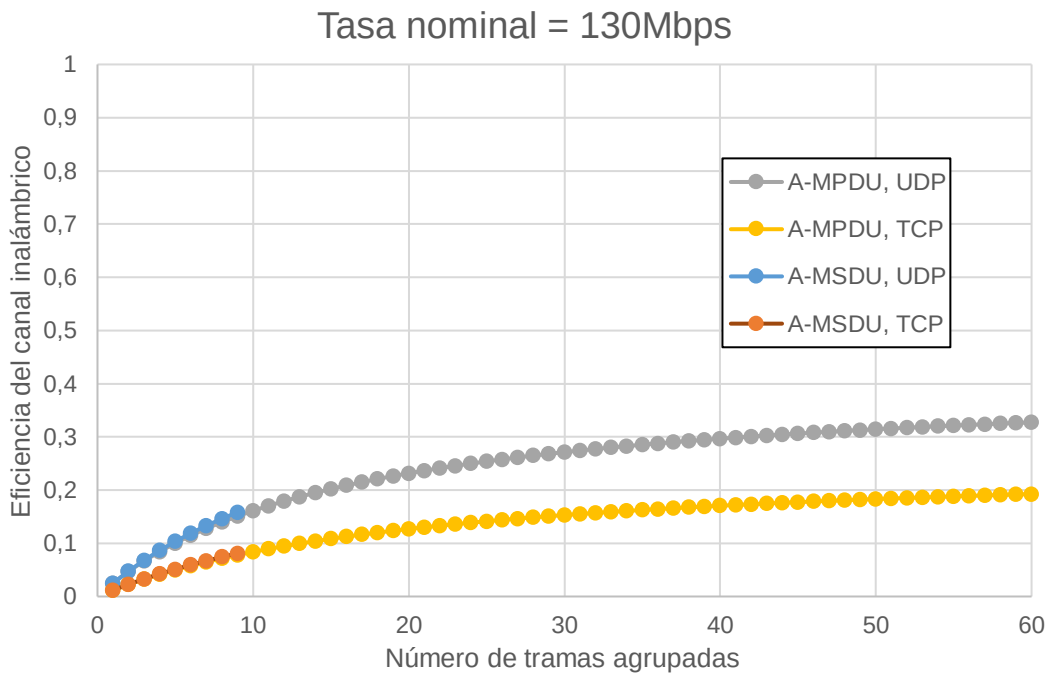


Figura 22: Eficiencia inalámbrica con tramas de 78 bytes

Como se puede observar en la Figura 22, centrándonos en los A-MPDUs en este caso obtenemos una eficiencia del 32,7% usando UDP y un 19,25% usando TCP cuando agregamos 60 tramas, frente al 2,5% y 1,17% que obtendríamos con los anteriores protocolos de transporte respectivamente sin usar agregación.

Estudio con paquetes de 228 bytes

Hacemos de nuevo un estudio de la eficiencia del canal inalámbrico, esta vez con el tamaño de bytes necesario para que haya 200 bytes de datos UDP, para lo que se usarán paquetes IP de 228 bytes.

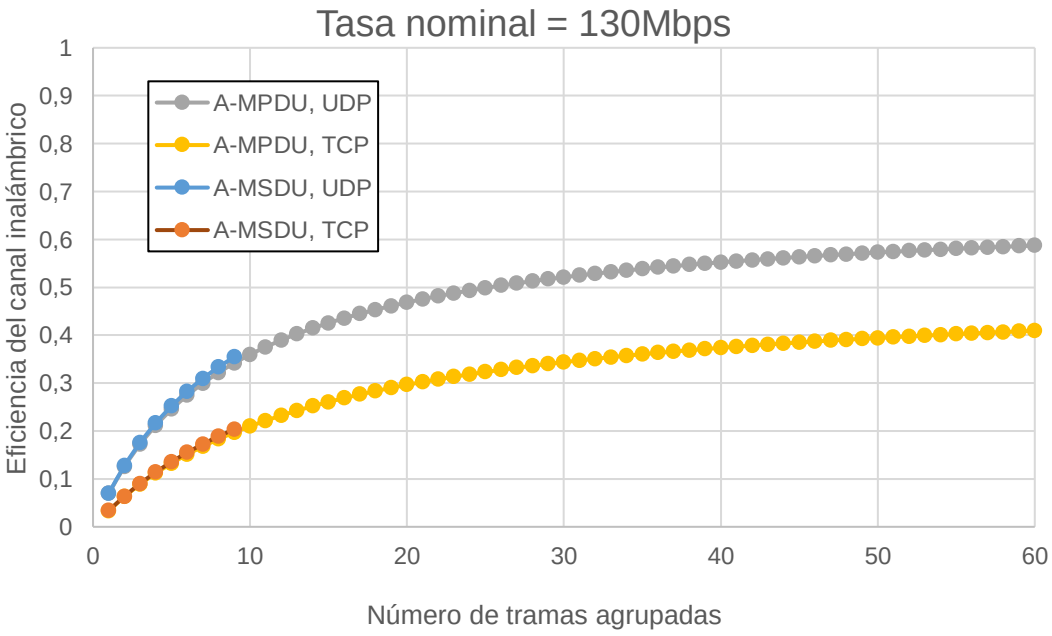


Figura 23: Eficiencia del canal inalámbrico agregando paquetes de 228 bytes

En este caso (Figura 23) la eficiencia que obtenemos es del 58,7% en UDP y del 40,92% en TCP mientras que en el caso de que no se implemente un mecanismo de agregación de tramas son en UDP el 6,9% y en TCP el 3,3%.

Estudio con paquetes de 528 bytes

Por último realizaremos el estudio de eficiencia para el tamaño de datos UDP más grande con el que hemos calculado la reducción de bytes en el apartado anterior, es decir, 500 bytes de UDP, con lo que usaremos subtramas de 528 bytes.

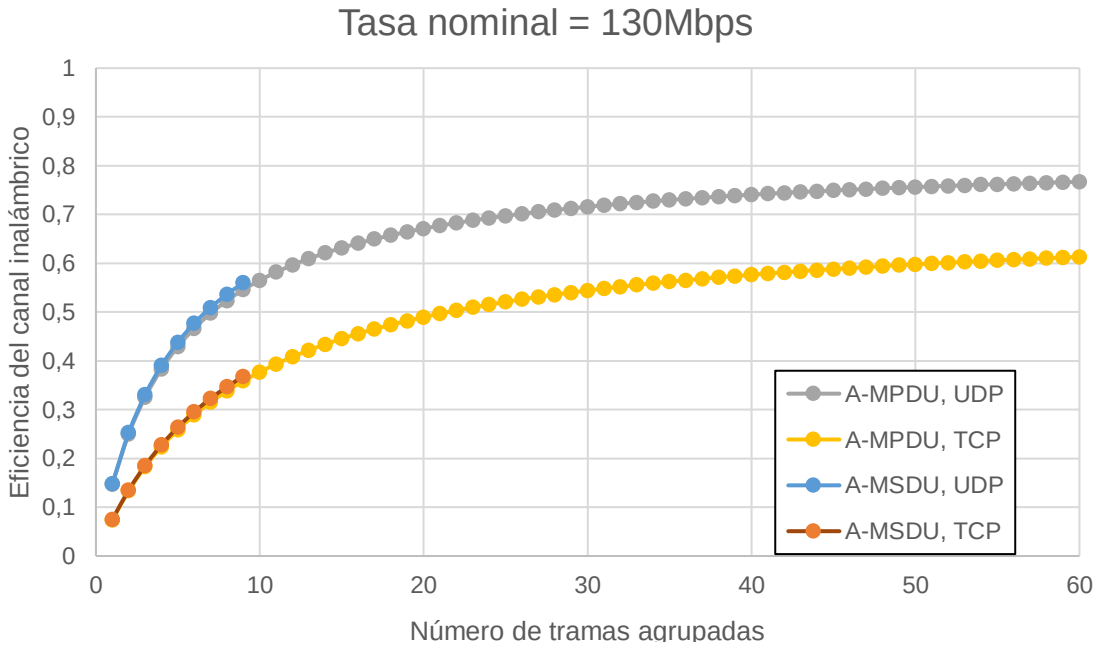


Figura 24: Eficiencia del canal inalámbrico agregando subtramas de 578 bytes

En la Figura 24 podemos ver que la eficiencia del canal en este caso alcanza un valor en UDP del 76,7% y en TCP del 61,3% cuando agregamos 60 subtramas en un único A-MPDU mientras que si la red no incorpora ningún mecanismo de agregación, la eficiencia tiene unos valores del 14,69% en UDP y del 7,4% en el TCP.

Como conclusión, podemos precisar que en todos los casos la eficiencia del canal inalámbrico mejora si incorporamos el mecanismo de agregación de tramas, y especialmente en el caso de que estemos usando A-MPDU, con paquetes UDP. Además se puede ver que cuanto mayor es el paquete, mayor es la eficiencia de la red. El acceso al medio requiere menos tiempo en proporción, y por tanto podemos usar más el canal.

4. Implementación y pruebas con tráfico Wi-Fi real

Uno de los objetivos de este TFG era la implementación de un programa que nos permitiese el envío de tramas reales 802.11 agregadas, mediante alguno de los mecanismos que propone el estándar. En esta sección explicaremos cómo se ha realizado.

Para esto nos vamos a valer de un ordenador que cuenta con una distribución Linux Debian 6, en el que ejecutaremos el programa que hemos creado a partir de otro de código abierto[25]. El objetivo es que este ordenador realice las funciones de un AP (*fake AP*, AP *falso)* gracias a una tarjeta de red TP-LINK que permite la inyección de paquetes mediante el uso de Radiotap.

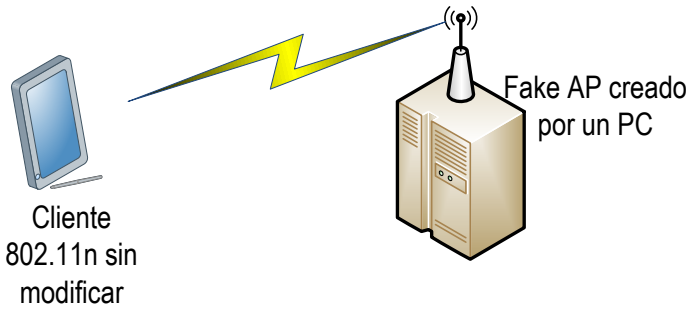


Figura 25: Esquema de la red que implementamos

El programa está escrito en lenguaje C. Utiliza las librerías *sys/socket.h* para poder abrir un *raw socket*[26] que nos permita inyectar tráfico y las librerías *ieee80211.h* e *ieee80211_radiotap.h*[27], ya que incorporan algunas de las estructuras de las cabeceras de nivel físico y de enlace que necesitamos.

Inicialmente el programa se limitaba a inyectar *Beacon* en la red y recibir los *Probe Request* de los clientes, por lo que no permitía que estos se asociaran. Por tanto, le hemos añadido todas las funcionalidades para permitir que los clientes se conecten a la red Wi-Fi que crea, y la inyección de tráfico, con y sin mecanismos de agregación, hacia los clientes.

El código del programa se ha compartido en el repositorio del proyecto Wi-5 en GitHub[28].

Por otra parte contaremos también con otro ordenador Linux que va a cumplir el papel del cliente (STA) que se conecta, recibe los paquetes y confirma la correcta recepción de los mismos con el uso del *Block* ACK.

---

[25] Evan Jones, Fake Access Points. http://www.evanjones.ca/software/fakeaps.html, accedido noviembre 2016

26 *Raw Socket*: Es un tipo de socket Linux, que permite enviar y recibir paquetes sin añadir ningún protocolo. Permite crear tramas IPv4 en espacio de usuario.

27 Se pueden obtener en http://ieee80211.sourceforge.net/#downloads, accedido noviembre 2016

[28] Wi-5 Aggregation. https://github.com/Wi5/wi5-aggregation, accedido noviembre 2016

4.1. Implementación del programa para agregar

El objetivo de este programa es actuar como un AP, al que se pueda asociar una STA, para así comenzar luego el mecanismo A-MPDU y poder de esta manera medir los ahorros. Nos interesaba hacerlo en espacio de usuario para poder así inyectar las tramas construidas por nosotros mismos, y así controlar todos los parámetros (sin dejar que el sistema operativo lo haga por nosotros), y poder así realizar las pruebas deseadas.

Por lo tanto el programa tiene que implementar todo el intercambio de tramas que se ha explicado en los apartados 2.4, 2.5 y 2.6, y que se va a recordar mediante la Figura 26

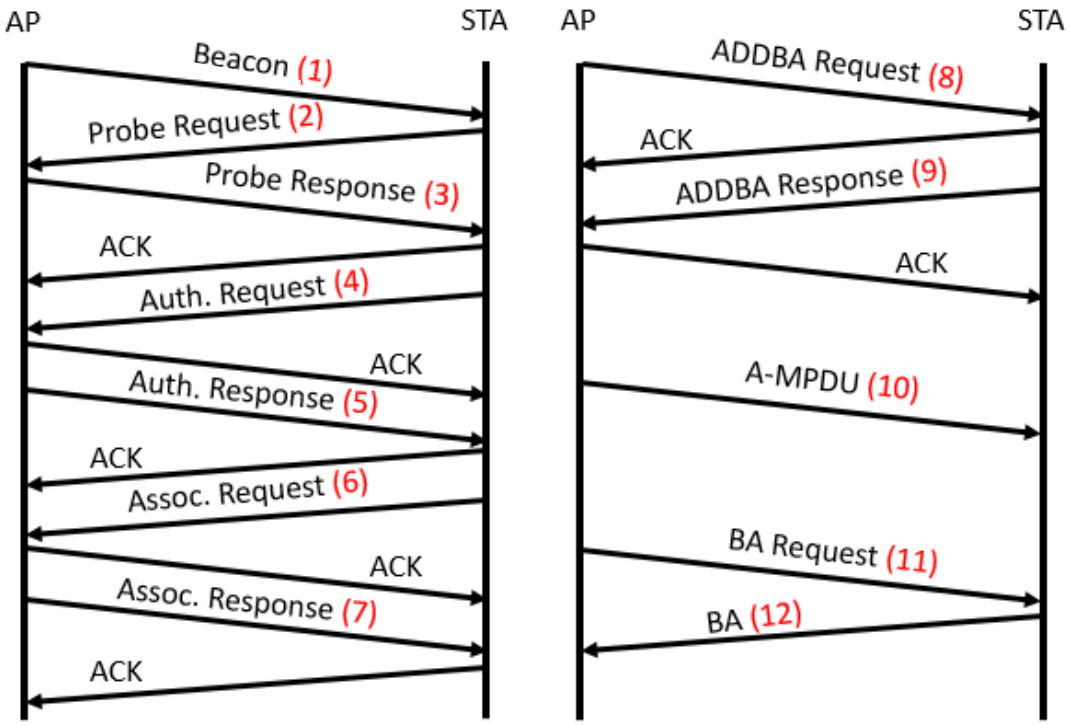


Figura 26: Intercambio de tramas que se tiene que implementar

Funcionalidad implementada previamente

Como se ha comentado anteriormente, el código del que partíamos sólo implementaba la creación y envío del *Beacon* (Figura 26 (1)) y se preparaba para la recepción de los *Probe Request* (Figura 26 (2)). La estructura del *Beacon*, como hemos visto previamente, es la mostrada en la Figura 7.

Se han añadido las siguientes modificaciones: en el Radiotap *Header* hemos rellenado el campo *version* con un '0', la longitud (*length*) con un '9', que es la suma del propio Radiotap *Header* más el *data rate* (velocidad) con el que va a ser transmitido.

El campo *present flags* de Radiotap funciona de la siguiente manera: si el desarrollador decide que es necesario especificar alguno de los parámetros que permite la cabecera, marca a 1 el bit que corresponda y entonces a continuación el driver del receptor sabrá que tiene que leer el número de campos que estén a 1 en el receptor.

También se marca en los *present flags* que se va a especificar el *data rate* (Figura 27 A). A continuación se añade el *data rate*, que en este caso hemos establecido a 1 Mbps (Figura

27 B). Posteriormente, en la recepción de esta trama, el *driver* completa algunos datos cambiando la longitud y quedando como se muestra en la Figura 27.

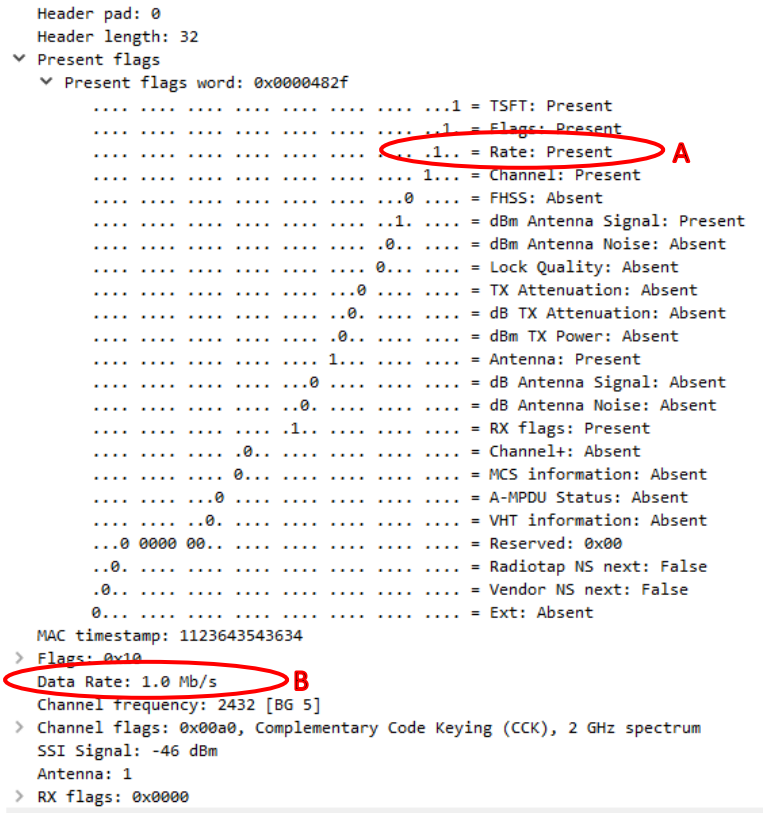


Figura 27: Radiotap Header del Beacon

Modificaciones en el *beacon* necesarias para la agregación

Para poder realizar la agregación hay que rellenar adecuadamente los campos del *Beacon* (Figura 26 (1)).

El campo MAC *Header* se rellenará con el *type* y *subtype*, donde se indicará que es de tipo *Management* y de subtipo *Beacon*. También se indica cuáles son las direcciones de destino, origen y *BSS*[29], la de destino es la *Broadcast* (FF:FF:FF:FF:FF:FF) y las otras dos iguales y son las MAC del equipo en el que estemos ejecutando el programa. También es necesario rellenar el número de secuencia. En la recepción, el driver de la STA rellenará de forma automática la duración quedando así totalmente completa la cabecera MAC.

En el campo *Management Beacon* completamos los campos: *Timestamp*, el intervalo entre *beacon* que fijamos en 102.4 milisegundos; en las *capabilities* (capacidades) marcamos *ESS* para indicar que el transmisor es un AP, que soporta el *Inmediate Block ACK* necesario para la agregación de tramas mediante el uso de A-MPDUs y el de *QoS*.

Posteriormente se añaden los *tags* (etiquetas) que aportan información extra acerca de la red, en este caso son: *SSID* que especifica el nombre de la red; las velocidades de

[29] BSS (Conjunto de servicio básico): Grupo de STA que se comunican entre ellas.

transmisión que soporta; el canal en el que está ubicado el AP; e información acerca de la transmisión a alta velocidad, en la cual se da información acerca de los A-MPDU.

Después de haber formado el paquete, se transmite y se espera a que llegue un *Probe Request* (Figura 26 (2)) o se agote el tiempo entre *Beacon* para enviar otro nuevo.

Formación del *Probe Response*

En el caso de que recibamos un *Probe Request*, deberemos responder con un *Probe Response* (Figura 26 (3)), en el que daremos más información y confirmaremos las características de la red.

Posteriormente, esperaremos a recibir un *Authentication Request* (Figura 26 (4)) en el que mediante otra trama de tipo *Management* (Figura 28 A) con subtipo *Authentication* (Figura 28 B) la STA dirá que se quiere conectar a un sistema abierto (campo *Algorithm* sin seguridad, Figura 28 C). Mediante el campo *SEQ* expresa que es una petición, al estar el valor a '1', como se puede ver en la Figura 28 D.

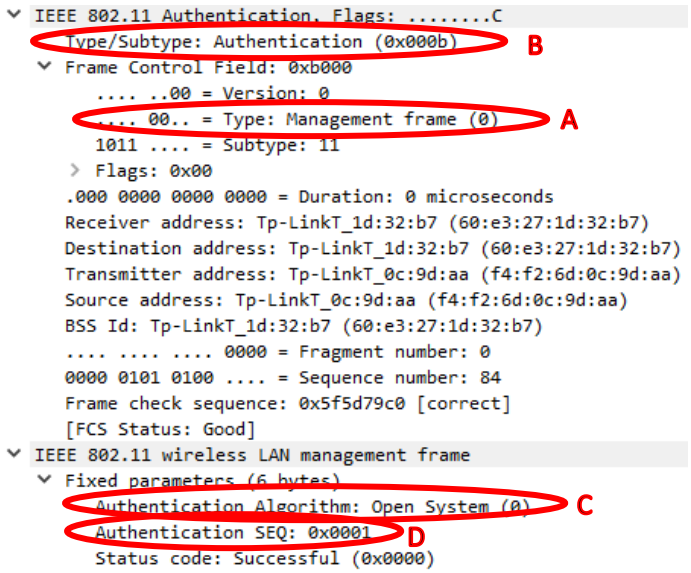


Figura 28: Auth. Request

Formación del *Authentication Response*

Una vez el AP recibe la petición, procede a contestar con un *Authentication Response* (Figura 26 (5)) que es similar al anterior, pero cambiando las direcciones MAC de la cabecera: la que era antes la de destino pasa a ser la de origen y *BSS*, la de origen anterior pasa a ser la de origen. Por otra parte, en el campo de *SEQ*, en el que en el *Authentication Request* había un '1' para indicar que era una petición, ahora se pone un '2' para indicar que es una trama de respuesta.

Si la STA que se está intentando conectar recibe la trama de *Authentication Response* con el campo de *status* con valor '0', esto implica que la autenticación ha sido realizada con éxito, con lo que procederá a intentar asociarse con el AP.

Para esto la STA envía un *Association Request* (Figura 26 (6)) al AP dentro de una trama de tipo *Management* y subtipo *Association Request*. Ahí se envían las *capabilities* que puede soportar la STA, así como información de la red a la que se está intentando asociar:

el *SSID*, las distintas velocidades de transmisión y las capacidades que tiene acerca de la transmisión a alta velocidad.

Formación del *Association Response*

Al recibir esto, el AP forma un *Association Response* (Figura 26 (7)) donde incluye sus *capabilities*, entre las que aparece *ESS* (Figura 29 A), indicando que es un AP que soporta el *Inmediate Block ACK* (Figura 29 B).

También transmite información acerca de las condiciones que se van a poder usar en el enlace, como el *SSID* (Figura 29 C), el canal en el que se encuentra (Figura 29 D) y las distintas velocidades de transmisión (Figura 29 E).

Podemos ver cómo es un *Association Response* en la Figura 29.

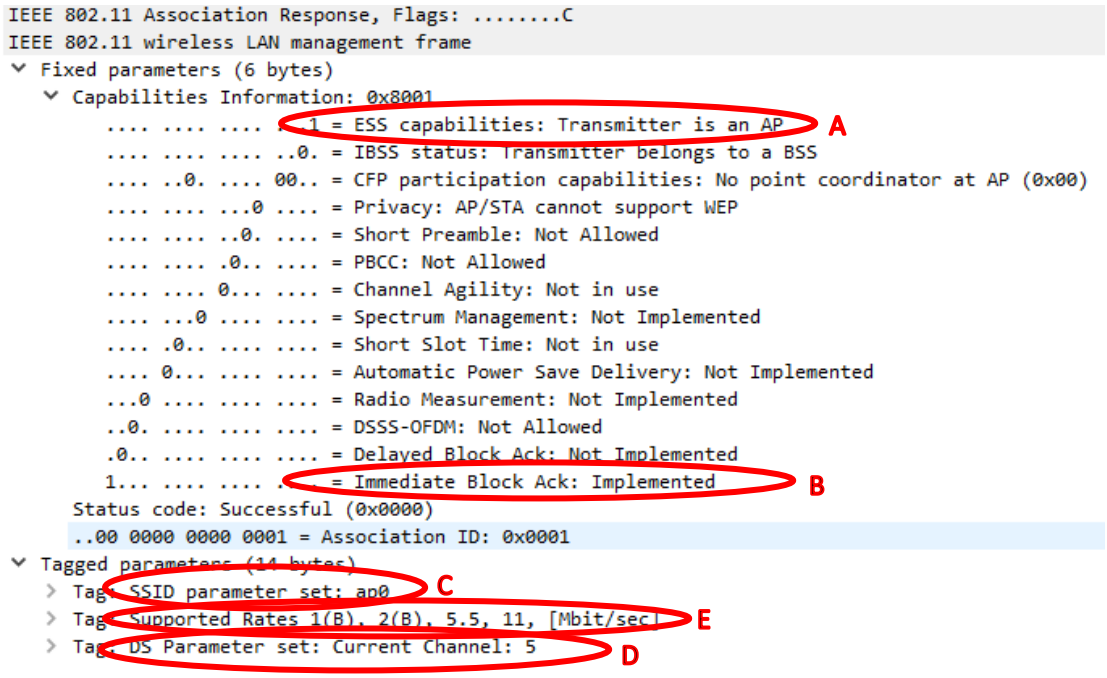


Figura 29: Association Response

Una vez que se ha producido el intercambio de tramas que hemos comentado hasta este momento, ya es posible que el AP o los clientes envíen datos por la red, pero todavía no hemos realizado la negociación necesaria para el uso de A-MPDUs, con lo que deberemos seguir implementando el *Add Block ACK Request* (Figura 26 (8)) como veremos a continuación.

Formación del *ADD Block ACK Request*

Esta trama es la que en la Figura 26 aparecía como *ADDBA Request*. En este caso los campos que tenemos que rellenar son tipo *Management*, subtipo *Action* (Figura 30 A), y dentro de las tramas de acción deberemos escoger la categoría 3 (Figura 30 B), que es la relacionada con los *Block ACK*; en el tipo de acción (Figura 30 C) debemos elegir *Add Block Ack Request*, y deberemos asignarle un *token* (Figura 30 D), que es un número que nos servirá para poder asociar la contestación en el caso de que se enviasen varios tipos de tramas de acción.

Completaremos también la *Block Ack Policy* (Figura 30 E), diciendo que no va a haber transmisión de A-MSDUs y que conforme vayan llegando los A-MPDUs vaya transmitiendo los *Block ACKs*. También rellenamos el *Timeout* (Figura 30 F) dándole el valor '0' para que en ningún momento de la sesión se dejen de usar los *Block ACKs.* Podemos ver cómo es este paquete en la Figura 30.

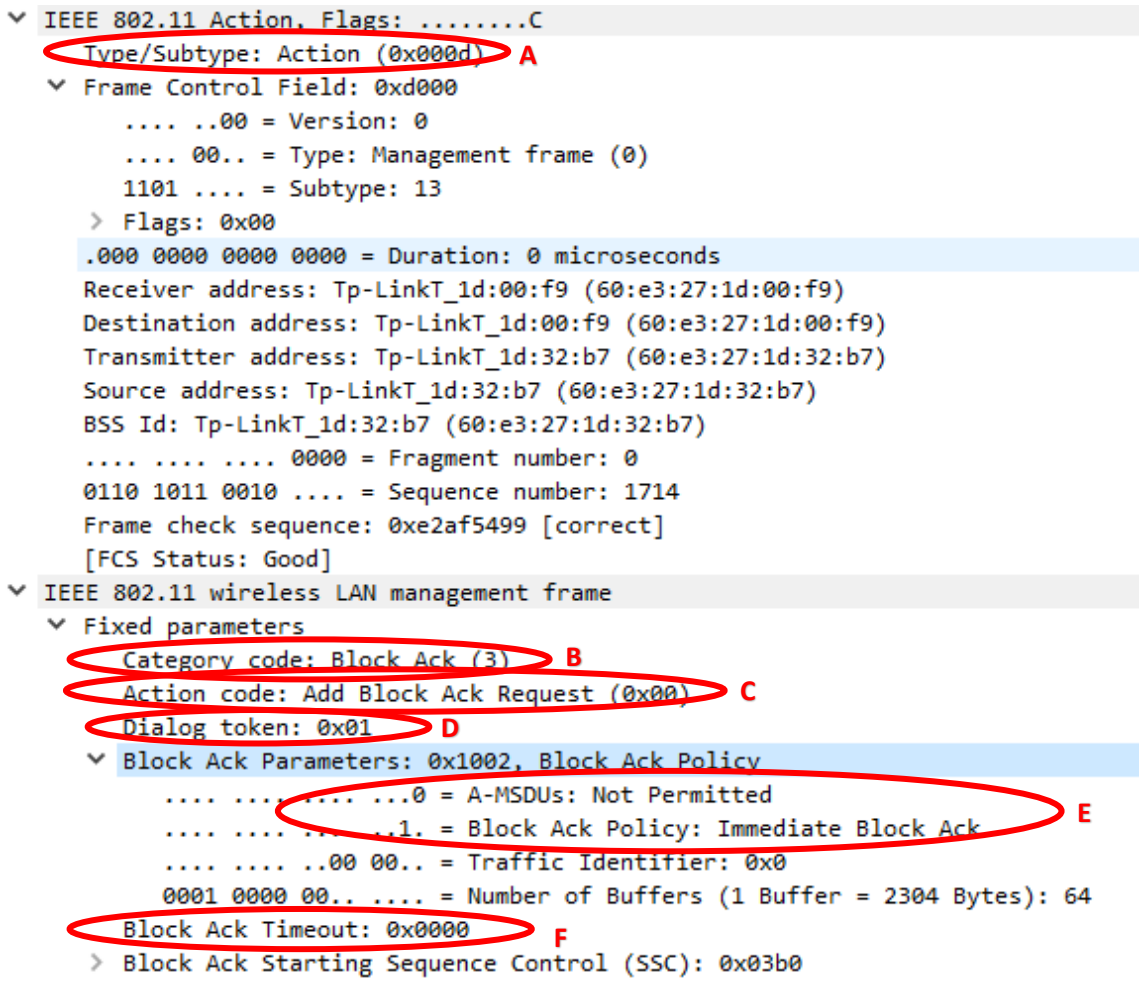


Figura 30: Add Block Ack Request

A esta trama, la STA le deberá responder con otra trama de acción, en este caso será un *Add Block ACK Response* (Figura 26 (9)) que únicamente se diferenciará del anterior, salvando el cambio en las direcciones MAC, en que su código de acción es el de *Add Block Ack Response* (Figura 31 A). Además, aparece un nuevo campo que es el de *status* (Figura 31 B), donde deberá aparecer un '0' en el caso de que la STA soporte el uso de *Block Ack* y quiera usarlo en el AP, ya que de este modo el AP sabrá que se ha acordado con éxito el uso de *Block Ack* durante el resto de la sesión.

Podemos ver la trama en la Figura 31:

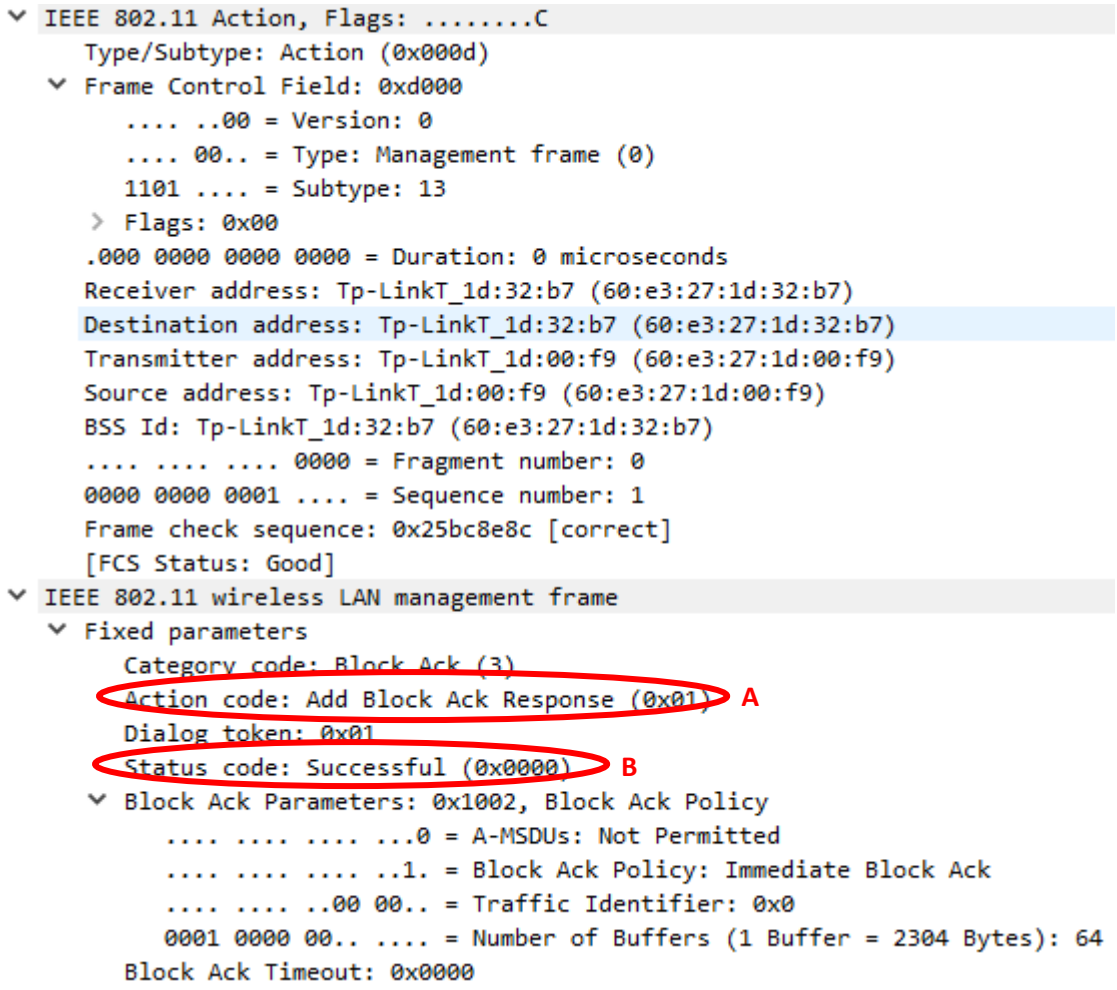


Figura 31: Add Block Ack Response

A partir de este momento el AP podrá enviar A-MPDUs (Figura 21 (10)) a la STA con la que se ha producido el intercambio de paquetes.

El envío de A-MPDUs se hace siguiendo el estándar, tal y como se ha explicado en el apartado 2.5. Solamente queremos resaltar que, una vez se ha formado la cabecera MAC, hay que rellenar la información *LLC*[30] y *SNAP*[31], para indicar que lo que va a aparecer a continuación es un paquete de tipo de IPv4, donde tendremos que rellenar la cabecera IP, posteriormente la cabecera UDP, y luego se añadirían los datos a enviar. Finalmente, si no fuesen múltiplo de 4, habría que añadir un *padding* de ceros.

Para solicitar el envío del *Block Ack* (Figura 26 (12)), el emisor del A-MPDU, en nuestro caso el AP, envía un *Block ACK Request* (Figura 26 (11)), que le indica a partir de qué número de secuencia está esperando confirmación (Figura 32 A). En nuestro caso hemos implementado el *Inmediate Block ACK*, como se ha podido ver en el campo *Policy* de la trama *Add Block Ack Request* y se puede ver en la Figura 32 B.

[30] LLC (*Logical Link Control*): Es la subcapa más alta del *data link* que ha definido el IEEE y se encarga del control de errores y de flujo

[31] SNAP (*Subnetwork Access Protocol*): Extensión del protocolo LLC para distinguir un mayor número de protocolos de la capa superior

Se puede ver cómo es el *Block ACK Request* que hemos implementado en la Figura 32:

```
IEEE 802.11 802.11 Block Ack Req, Flags: ........C
   Type/Subtype: 802.11 Block Ack Req (0x0018)
 ▾ Frame Control Field: 0x8400
      .... ..00 = Version: 0
      .... 01.. = Type: Control frame (1)
      1000 .... = Subtype: 8
    › Flags: 0x00
   .000 0000 0000 0000 = Duration: 0 microseconds
   Receiver address: Tp-LinkT_1d:00:f9 (60:e3:27:1d:00:f9)
   Transmitter address: Tp-LinkT_1d:32:b7 (60:e3:27:1d:32:b7)
   .... .00. = Block Ack Request Type: Basic Block Ack Request (0x0)
 ▾ Block Ack Request (BAR) Control: 0x0001
      .... .... .... .1.. = BAR Ack Policy: Immediate Acknowledgement Required
      .... .... .... ..0. = Multi-TID: False
      .... .... .... .0.. = Compressed Bitmap: False
      .... 0000 0000 0... = Reserved: 0x000
      0000 .... .... .... = TID for which a Basic BlockAck frame is requested: 0x0
 ▾ Block Ack Starting Sequence Control (SSC): 0x03b0
      .... .... .... 0000 = Fragment: 0
      0000 0011 1011 .... = Starting Sequence Number: 59
   Frame check sequence: 0xae83ead1 [correct]
   [FCS Status: Good]
```

B

A

Figura 32: Estructura del Block ACK Request implementado

Tras esto recibiremos el *Block ACK*, tal y como hemos explicado en el apartado 2.6.

4.2. Elección del hardware: tarjetas de red y drivers

Detallaremos a continuación algunos detalles sobre el hardware y los drivers que se han usado para ejecutar el programa. Inicialmente empezamos a trabajar con unas tarjetas de red USB de la marca TP-LINK (Archer T2U), que soportaban Wi-Fi hasta la versión 802.11ac. Pero la cabecera que usaban a nivel físico no era Radiotap sino Prism, por lo que tuvimos que cambiar a otras.

Finalmente optamos por unas tarjetas de red externas (USB) TP-LINK TL-WN722N (Figura 33), que sí usan *Radiotap*, permitiendo más flexibilidad para los desarrolladores, y siendo muy usadas y con más información disponible. Usamos estas tarjetas tanto en el equipo que hace de AP con en el que hace de STA.

Figura 33: TP-LINK TL-WN722N

Una vez que ya estuvimos trabajando con las tarjetas que soportaban la cabecera Radiotap se implementó una conexión en la que no se usase el mecanismo de agregación para poder realizar las pruebas de comparación y comprobar que esto funcionaba de forma correcta.

Después se pasó a implementar el mecanismo de agregación que, como hemos comentado antes difiere del normal, porque tiene que enviarse el *AddBARequest* y obtener la respuesta, luego enviarse los A-MPDUs y finalmente enviar un *BARequest* y obtener el *Block ACK*.

Todo lo implementado funcionó correctamente (*Add Block* ACK *Request*, *Block* ACK *Request*) a excepción de la trama A-MPDU que por algún motivo que desconocemos, el driver con el que funcionábamos el *ath9k*[32] no detectaba el MPDU *delimiter* con lo que cambiaba las direcciones MAC de origen y destino. Como hemos visto previamente lo antecede y los MPDUs contenidos dentro del A-MPDU no llegan al destino. Por tanto, decidimos actualizar el driver de los equipos a la versión *ath9k_htc*[33], una versión con más funcionalidades.

Pero nos siguió dando el mismo problema, por lo que nuevamente probamos a revisar la formación de los A-MPDUs, con el mismo resultado.

Al ver que lo con lo anterior no habíamos conseguido la correcta interpretación de los A-MPDUs en el destino, vimos que el driver *ath10k*[34] sí permitía hacerlo. Pero el único dispositivo disponible que admitía este driver era un *router* TP-LINK Archer c7 funcionando con *OpenWrt*[35]. Por tanto preparamos el *router* para funcionar con el *ath10k* y seguimos las instrucciones que da *OpenWrt* para que el *router* actúe como cliente[36]. Pero tras seguir esos pasos, no conseguimos que el *router* actuase como cliente frente al ordenador que estaba haciendo las veces de AP.

Finalmente, nos decidimos a hacer el estudio del ancho de banda y tiempo, utilizando un PC trabajando como AP, con las tarjetas de red de la marca TP-Link y los driver *ath9k_htc*, ya que, aunque el receptor no interpreta de forma correcta los A-MPDU, sí responde al *Block ACK Request* que le enviamos, con lo que tanto el número de bytes como el tiempo que tarda en recibir los paquetes se pueden determinar de forma correcta.

[32] *Ath9k: https://wireless.wiki.kernel.org/en/users/drivers/ath9k*, accedido noviembre 2016
[33] *Ath9k_htc*: https://wireless.wiki.kernel.org/en/users/drivers/ath9k_htc, accedido noviembre 2016
[34] *Ath10k*: https://wireless.wiki.kernel.org/en/users/drivers/ath10k, accedido noviembre 2016
[35] *OpenWrt Wireless Freedom*: https://openwrt.org/, accedido noviembre 2016
[36] *OpenWrt Client Mode Wireless*: https://wiki.openwrt.org/doc/howto/clientmode, accedido noviembre 2016

5. Estudio práctico

Una vez que disponemos de un hardware y de un programa que transmite tramas A-MPDU del tamaño que nosotros decidamos, y agrupadas en el número que se desee, realizamos las medidas. El objetivo es comprobar que los resultados coinciden con lo que hemos calculado de forma teórica en el estudio sobre el ahorro del número de bits.

Los cálculos presentados en el estudio de la eficiencia del canal no se pueden repetir en este caso, porque el canal varía y por tanto la tasa de error no es constante. Se necesitaría un entorno aislado (por ejemplo una cámara anecoica) del que no disponemos.

Para las pruebas utilizamos el programa desarrollado, usando distintos tamaños de paquetes (distintos tamaños de datos UDP) y para distinto número de paquetes agregados, teniendo en cuenta que el MTU que permitía nuestra red es de 1500. Para obtener los resultados realizamos capturas con *Wireshark.*

Estudio con paquetes de 50 bytes

En primer lugar, transmitiremos paquetes IP de 50 bytes. Debido al problema que hemos comentado anteriormente con respecto al MTU de la red, en nuestras medidas el tamaño máximo de tramas que caben en un A-MPDU es de 10, ya que la siguiente medida que tomamos es con 20 subtramas que serían 1000 bytes a nivel IP, pero añadiendo los MPDU *Delimiter* y las 20 cabeceras MAC se supera el MTU de la red. Por tanto, las medidas obtenidas a partir de este valor se obtienen mediante el uso de varios A-MPDUs. Esto conlleva una reducción del máximo ahorro al que se podría llegar, ya que cada vez que se tiene que enviar un A-MPDU, se necesita confirmarlo.

En las siguientes figuras (Figura 34, Figura 35 y Figura36) se puede ver la reducción del *overhead* que se puede conseguir con este mecanismo, ya que para el mismo tamaño de datos, va a variar el número de bits que circulan por la red, dependiendo si lo enviamos en paquetes UDP (sin agregar) o lo enviamos como subtramas dentro de uno o varios A-MPDUs. Además esta segunda opción ofrece una gran ventaja, como hemos comentado antes: como se reduce el número de tramas que hay que enviar y estamos trabajando en un medio inalámbrico, es decir un medio compartido, se va a reducir el tiempo de acceso al medio.

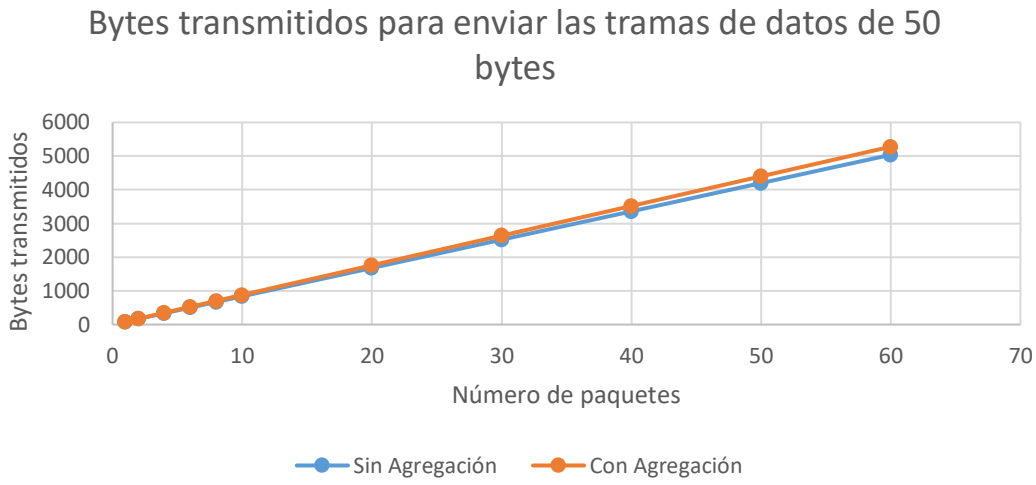


Figura 34: Bytes enviados para enviar paquetes de datos de 50 bytes a nivel IP

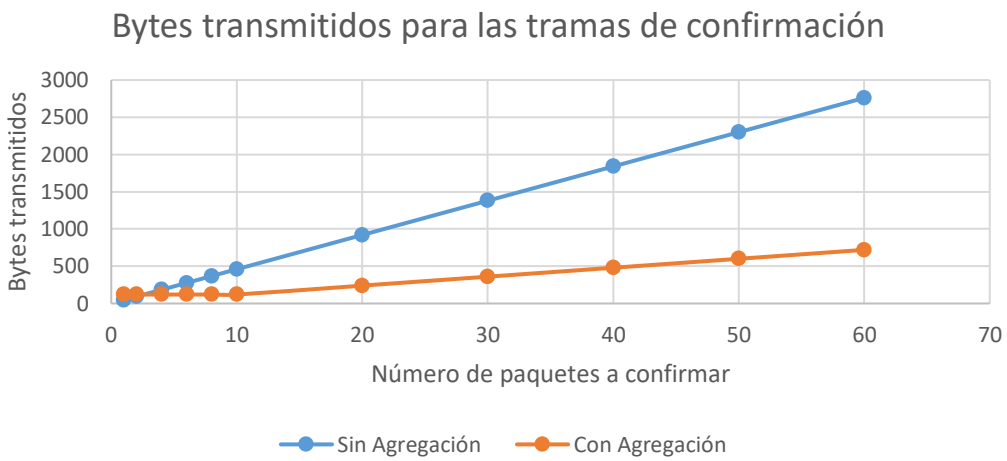


Figura 35: Bytes transmitidos para confirmar las tramas de datos con y sin agregación

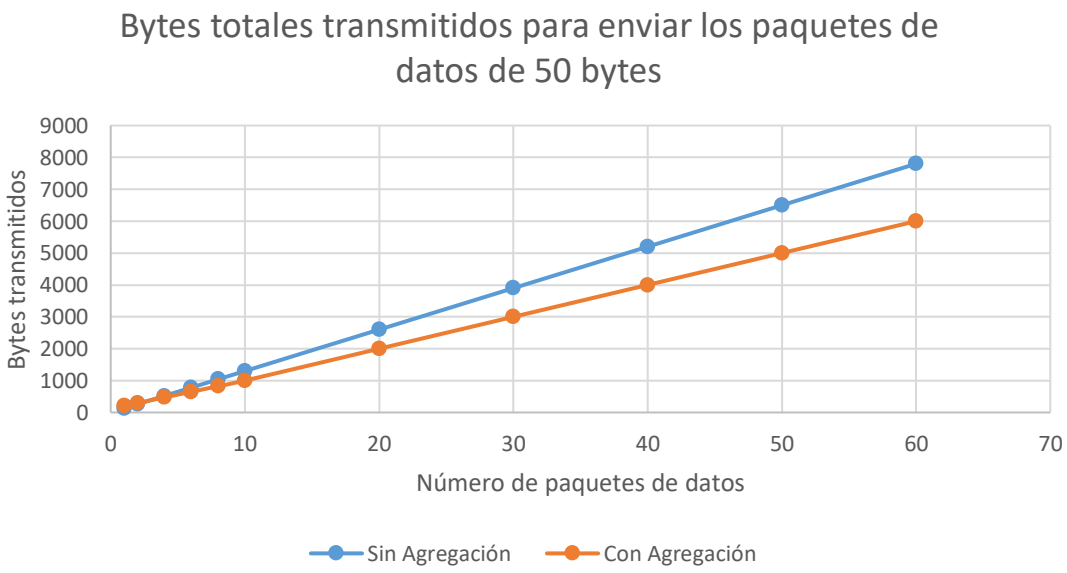


Figura 36: Bytes transmitidos con y sin agregación para el envío de paquetes de datos

En la Figura 34 podemos ver con tráfico real, lo que obtuvimos teóricamente en el apartado 3.1: el número de bytes que es necesario transmitir para enviar cierto número de tramas de datos, en una red que implemente o no el mecanismo de agregación de tramas A-MPDU.

En la Figura 35, podemos observar los bytes que se precisan para confirmar las tramas de datos que se han enviado. En este caso difiere de lo obtenido en el estudio teórico, ya que el MTU de la red es un efecto limitante que hace que se tengan que enviar varias tramas de A-MPDU, con lo que hará falta enviar varios *Block* ACK, lo que conlleva un aumento en los bytes que se necesitan transmitir por la red para confirmar las tramas.

En la Figura 36 (suma de los valores obtenidos en las figuras 34 y 35) se puede ver que se tienen que transmitir muchos menos bytes si se usan mecanismos de agregación cuando el tamaño de los paquetes es pequeño. En este caso se alcanza un ahorro de hasta el 23,07%, que se mantiene constante desde el momento en que tenemos que enviar varios A-MPDUs, para no sobrepasar el MTU que la red nos impone.

Estudio con paquetes de 78 bytes

Tras realizar el estudio del ahorro con 50 bytes pasamos a hacer las mismas capturas pero enviando ahora paquetes de 78 bytes (50 bytes de datos UDP) que sigue siendo un tamaño bastante pequeño.

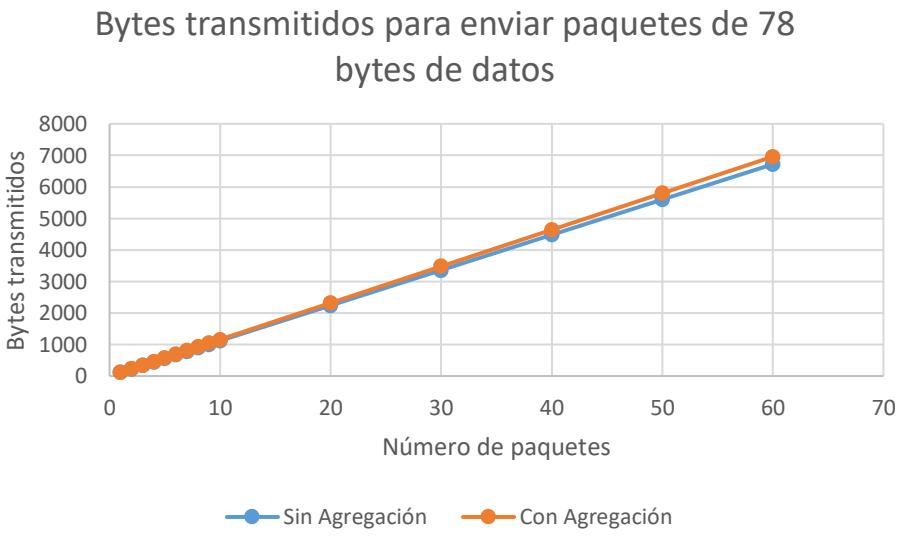


Figura 37: Bytes que se transmiten para enviar los paquetes de datos de 78 bytes

Igualmente a lo que pasaba en el caso anterior este se queda fijo en este valor a partir de que se envían 10 paquetes agregados en el A-MPDU, ya que en nuestra siguiente medida (con 20 paquetes), si se agregan en una A-MPDU esta sobrepasa el MTU de la red. Con lo que el número de bytes que se usan para confirmar los paquetes de datos coinciden con los del caso anterior (Figura 35). Con lo que en este caso el ahorro llega a ser del 18,99%.

Estudio con paquetes de 228 bytes

Posteriormente pasamos a tomar datos con paquetes de 228 bytes con lo que, de nuevo debido al MTU, a partir de 5 paquetes estarán formados por varios A-MPDUs que agregan 5 paquetes y obtenemos lo que aparece en las siguientes gráficas.

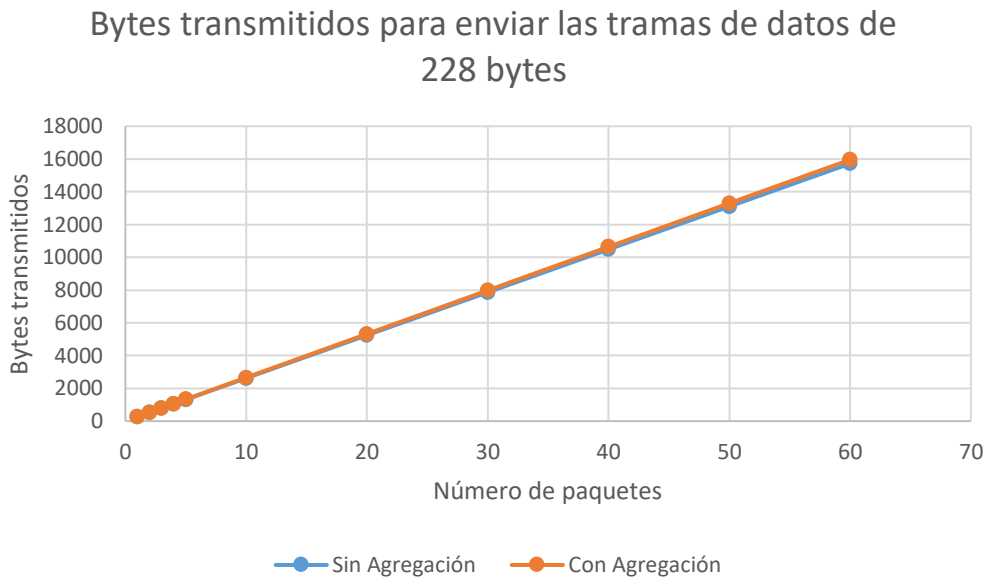


Figura 38: Número de bytes transmitidos con y sin agregación para enviar los paquetes de 228 bytes

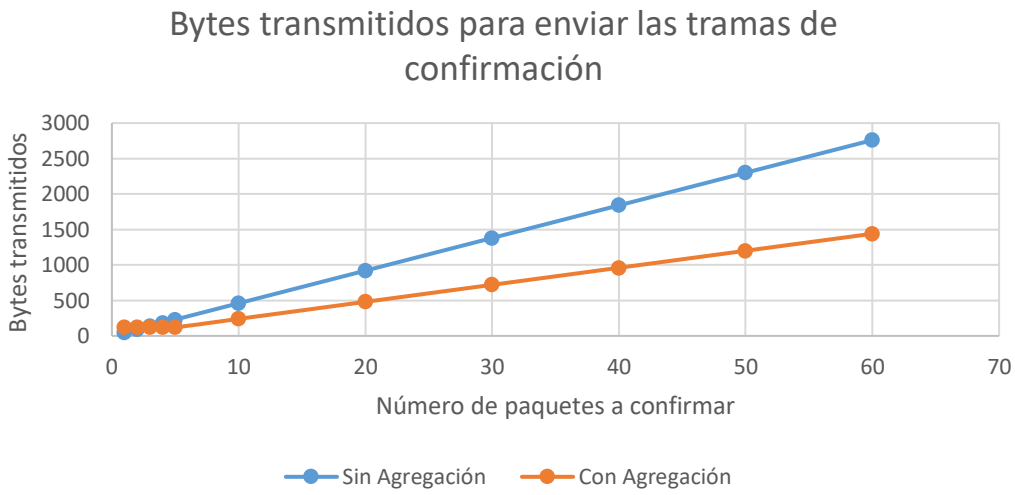


Figura 39: Bytes transmitidos para confirmar las tramas de datos con y sin agregación

Basándonos en los resultados mostrados en las figuras 38 y 39 podemos observar que el ahorro que se produce ahora es menor (5,84%), ya que este disminuye conforme aumenta el tamaño de los paquetes. Además, el problema añadido del MTU nos limita el tamaño del A-MPDU, impidiéndonos juntar paquetes hasta llegar al tamaño máximo que permiten los A-MPDU según el estándar.

Estudio con paquetes de 528 bytes

Por último realizamos la misma prueba con paquetes con un tamaño de datos UDP de 528 bytes, con lo que el número máximo de paquetes que podemos agregar en un A-MPDUs es de 2, ya que con tres únicamente con los datos UDP alcanzaríamos el MTU y a esto hay que añadirle todas las cabeceras, con lo que se sobrepasa.

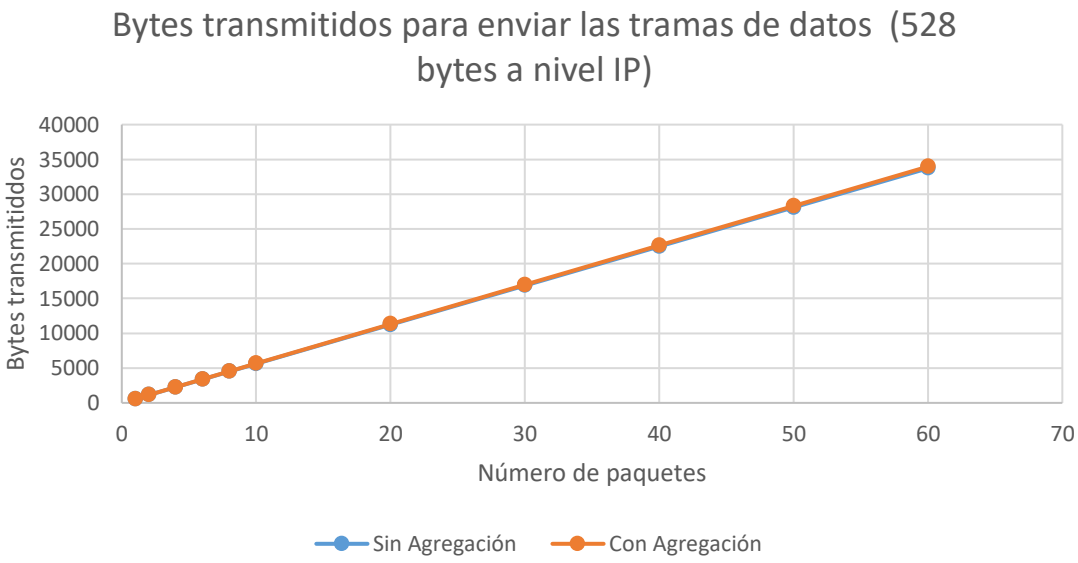


Figura 40: Bytes transmitidos con y sin agregación para enviar las tramas de datos (528 bytes a nivel IP)

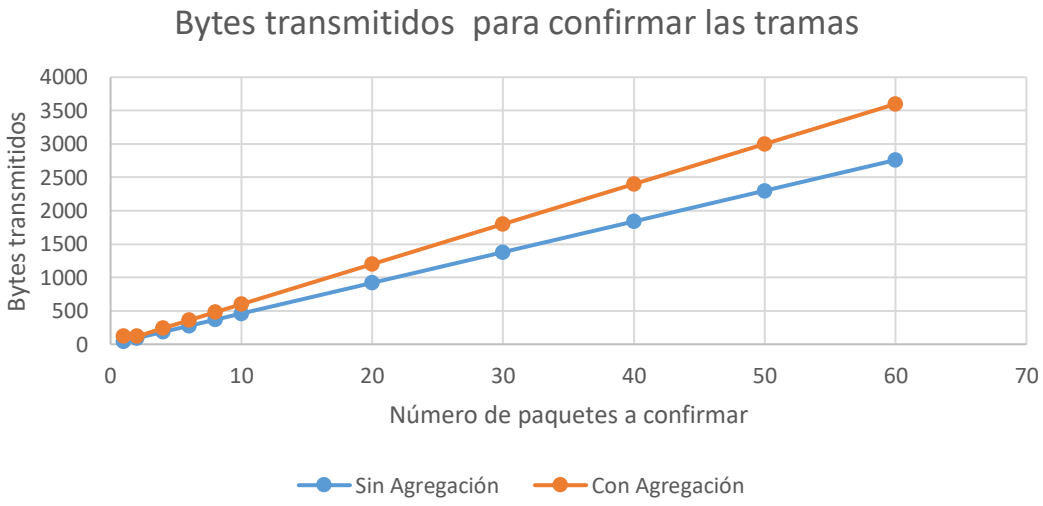


Figura 41: Bytes transmitidos con y sin agregación para confirmar las tramas de datos

Sumando los datos de las figuras anteriores (Figura 40 y Figura 41) obtenemos los bytes totales que se envían para transmitir y confirmar los paquetes de datos, tanto en modo nativo como con el uso de A-MPDUs.

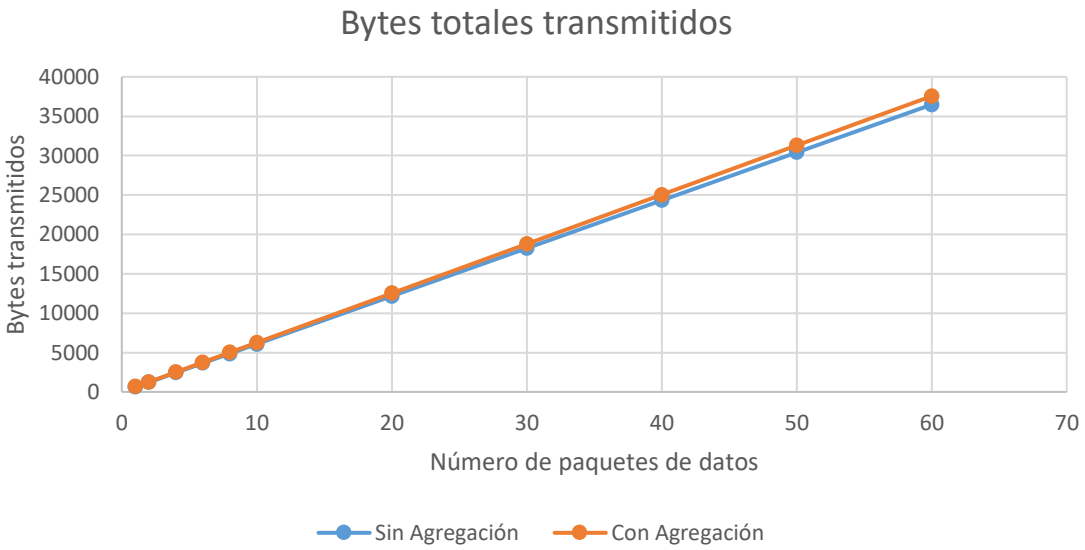


Figura 42: Bytes tranmitidos para enviar las tramas de datos (528 bytes) y las de confirmacion

En la Figura 42 podemos ver que en este caso debido al límite que nos pone el MTU en nuestra red el uso de A-MPDUs supone que tengamos que enviar 2,96% más bytes que si no lo implementamos.

Los resultados obtenidos coinciden con los presentados en el estudio teórico. Las únicas diferencias se deben al MTU de la red en la que hemos realizado las medidas, que ha hecho necesario enviar más de un A-MPDU, aumentando así el número de bytes que es necesario enviar para confirmar las tramas y datos y por tanto el número de bytes totales, reduciendo el ahorro en el número de bytes.

6. Conclusiones y Líneas futuras

6.1. Conclusiones

Tras presentar en detalle lo que se ha realizado en este TFG, se pueden extraer las siguientes conclusiones y resultados.

Se ha visto que el número de bytes a enviar se reduce al usar el método de agregación de tramas. Por otra parte, una sola cabecera de nivel físico se comparte entre un número de tramas, lo cual es otra ventaja, pues se transmite a una tasa menor.

La eficiencia del canal inalámbrico se incrementa en todos los casos, al agregar tramas. Debido al mecanismo de agregación, disminuye el número de tramas que se tienen que enviar por la red, y esto representa una gran ventaja: al usarse un medio inalámbrico para transmitir la información, es necesario ejecutar una serie de mecanismos de acceso al medio con lo que al reducir el número de tramas que se envían a la red se va a reducir el tiempo en el que el dispositivo espera a que le toque enviar cada trama. Por ejemplo, la eficiencia puede llegar hasta un 62% al usar este mecanismo de agregación, al enviar tramas UDP de 500 bytes agregadas mediante A-MPDUs, suponiendo una tasa nominal de 130 Mbps.

Se ha realizado un programa de código abierto, para permitir el estudio práctico. El programa permite inyectar tramas agregadas y compararlas con el modo nativo. Este programa se ha puesto en un repositorio a disposición de la comunidad científica, al ser parte de un proyecto financiado por la UE, y también por ser la evolución de otro programa de código abierto. Los resultados obtenidos con este programa coinciden con los esperados por el estudio teórico.

Se ha comprobado que este mecanismo de agregación produce ahorros significativos para servicios que utilicen muchos paquetes de pequeño tamaño. A la vez, se debe evitar introducir retardos que perjudiquen la experiencia de usuario en servicios con requisitos de tiempo real. Esto se deberá tener en cuenta a la hora de implementar este mecanismo. Por ejemplo, será conveniente establecer una cota superior para el retardo en el buffer antes de enviar una trama agregada.

6.2. Líneas futuras

Una posible línea de investigación futura es el estudio comparativo de la agregación de tramas mediante el uso de los otros dos mecanismos que propone el estándar Wi-Fi en la versión 802.11n:

- La agregación mediante A-MSDUs, en la cual se puede usar la conexión a la red Wi-Fi que se ha implementado, variando únicamente los campos en los cuales se tenga que indicar que se van a enviar A-MSDUs.
- La agregación de dos niveles, la cual sería la combinación del mecanismo que se implementa en este proyecto y de los A-MSDUs, ya que consiste en encapsular

los A-MSDUs en A-MPDUs. Habría que modificar el *Add Block ACK Request*, para señalar que dentro de los A-MPDUs hay A-MSDUs.

Otra posible línea de investigación sería comparar los mecanismos de agregación de tramas (a nivel 2) con otros protocolos que combinen paquetes (a nivel 3) como puede ser Simplemux, un protocolo para reducir el *overhead* que ha surgido en la Universidad de Zaragoza [7]. Esto se podría hacer con distintos tipos de tráfico, para estudiar el ahorro de ancho de banda que se consigue con cada uno de los protocolos, en cada situación.

Y finalmente se podría diseñar un *scheduler* que, basándose en el tipo de tráfico que está transmitiendo, use un algoritmo que le permita reducir al máximo el número de bits que circulan por la red. Esto permitiría reducir la carga de la red, lo que puede resultar especialmente interesante en las redes domésticas, en las cuales no se dispone de mucho ancho de banda. De este modo se permitirá que los usuarios puedan usar más servicios simultáneamente con una buena *QoE* (*Quality of Expirience*, Calidad de la Experiencia).

6.3. Planificación del trabajo

En la Figura 43 se puede ver el diagrama de Gantt con las etapas del trabajo y su duración.

Actividad	Inicio	Duración	Final
Estudio previo	15-feb.-16	77	2-may.-16
Decisión mecanismo	3-mar.-16	4	7-mar.-16
Estudios teóricos	14-mar.-16	16	30-mar.-16
Elección de equipos	4-abr.-16	17	21-abr.-16
Implementación	3-may.-16	81	23-jul.-16
Pruebas otro tipo de implementaciones	24-jul.-16	32	25-ago.-16
Estudio práctico	26-ago.-16	10	5-sep.-16
Elaboración de la memoria	12-sep.-16	46	28-oct.-16

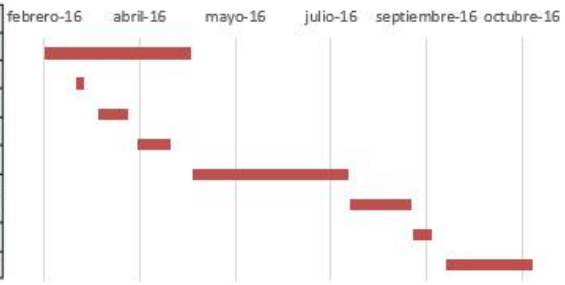


Figura 43: Diagrama de Gantt

7. Bibliografía

[1] IEEE 802.11, IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 1: Radio Resource Measurement of Wireless LANs, IEEE Std 802.11k-2008, 2008.

[2] IEEE 802.11ac, IEEE Draft Standard for IT - Telecommunications and Information Exchange Between Systems - LAN/MAN - Specific Requirements - Part 11: Wireless LAN Medium Access Control and Physical Layer Specifications - Amd 4: Enhancements for Very High Throughput for operation in bands below 6GHz, IEEE P802.11ac/D3.0, June 2012.

[3] Ginzburg B., Kesselman A., "Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11n," Sarnoff Symposium, 2007 IEEE, vol., no., pp.1,5, April 30 2007-May 2 2007.

[4] T. Selvam, S. Srikanth, "A frame aggregation scheduler for IEEE 802.11n," Communications (NCC), 2010 National Conference on, vol., no., pp.1, 5, 29-31 Jan. 2010.

[5] J. Liu, M. Yao, Z. Qiu, "Enhanced Two-Level Frame Aggregation with Optimized Aggregation Level for IEEE 802.11n WLANs" in Communications Letters, IEEE, vol.19, no.12, pp.2254-2257, Dec. 2015

[6] Robyns, P., Quax, P., Lamotte, W. (2015, June). Injection attacks on 802.11 n MAC frame aggregation. In Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (p. 13). ACM.

[7] J. Saldana, I. Forcen, J. Fernández-Navajas, and J. Ruiz-Mas, "Improving Network Efficiency with Simplemux," presented at the IEEE CIT 2015, International Conference on Computer and Information Technology, Liverpool, 2015, pp. 446–453, http://diec.unizar.es/~jsaldana/personal/chicago_CIT2015_in_proc.pdf, [Ultima visita: Noviembre 2016].

Índice de Figuras