Universidad Zaragoza

Trabajo Fin de Máster

Propuesta y evaluación de un sistema centralizado de control de los parámetros de QoS en una red IEEE 802.11.

Proposal and evaluation of a centralized system for tuning QoS parameters in an IEEE 802.11 network.

Autor

Miguel Ángel López Lafuente

Director

Jorge Ortín Gracia

Ponente

María Canales Compés

Escuela de Ingeniería y Arquitectura

Máster en Ingeniería de Telecomunicación

Diciembre 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. MIGUEL ÁNGEL LÓPEZ LAFUENTE,

con nº de DNI 73093413-B 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) MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN (Título del Trabajo) PROPUESTA Y EVALUACIÓN DE UN SISTEMA CENTRALIZADO DE CONTROL DE LOS PARÁMETROS DE QOS EN UNA RED IEEE 802.11,

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

Zaragoza, 24 DE NOVIEMBRE 2016

Fdo: MIGUEL ÁNGEL LÓPEZ LAFUENTE

Propuesta y evaluación de un sistema centralizado de control de los parámetros de QoS en una red IEEE 802.11.

RESUMEN

Este Trabajo Fin de Máster propone un algoritmo para mejorar la priorización de flujos que realiza el estándar IEEE 802.11e por defecto mediante su método EDCA.

A diferencia de la mayoría de soluciones propuestas por otros autores, que se basan en una solución distribuida en la que cada punto de acceso actúa de modo independiente al resto, en este Trabajo Fin de Máster se propone un algoritmo centralizado, capaz de controlar todos los puntos de acceso de la red simultáneamente. Esta centralización permite proponer soluciones más avanzadas que consiguen mejorar los resultados obtenidos con los algoritmos distribuidos analizados en el estado del arte.

El algoritmo propuesto se ha programado en ns-3, el cual es un simulador de red de código libre ampliamente utilizado en el ámbito académico.

Dicho algoritmo se pone a prueba en diferentes escenarios también programados en ns-3, donde se compara su rendimiento en diferentes situaciones de tráfico con respecto al método EDCA y otras soluciones encontradas en la bibliografía.

Proposal and evaluation of a centralized system for tuning QoS parameters in an IEEE 802.11 network.

ABSTRACT

This Final Master Project proposes an algorithm to improve the flow prioritization the standard IEEE 802.11 carries out by default with its EDCA method.

Unlike most of solutions of other authors, which are based on distributed solutions in which each access point works independently, in this Final Master Project we propose a centralized algorithm, which is able to control every access point of the network at the same time. This centralization could propose most advanced solution, which improves the results obtained with the distributed algorithms analysed at the state-of-the-art.

The proposed algorithm has been programed in ns-3, an open source network simulator widely used in the academic environment.

This algorithm is tested in different scenarios also programmed in ns-3, where we compare the performance at different traffic situations in relation to the default EDCA standard and other state-of-the-art solutions.

Índice general

Índice de Figuras

Índice de Tablas

Acrónimos

- **AC**: Access Categories
- **AC_BE**: Access Category Best-Effort
- **AC_BK**: Access Category Background
- **AC_VI**: Access Category Video
- **AC_VO**: Access Category Voice
- **AIFS**: Arbitration Inter-Frame Spacing
- **AIFSN**: Arbitration Inter-Frame Space Number
- **AP**: Access Point
- **CCA**: Clear Channel Assessment
- **CFP**: Content Free Period
- **CP**: Content Period
- **CSMA/CA**: Carrier Sense Multiple Access / Collision Avoidance
- $\boldsymbol{CW}$: Contention Window
- $\boldsymbol{CW_{max}}$: Maximum Contention Window

- $\boldsymbol{CW_{min}}$: Minimum Contention Window
- **DCF**: Distributed Coordination Function
- **EDCA**: Enhanced Distributed Channel Access
- **FTP**: File Transfer Protocol
- **IEEE**: Institute of Electrical and Electronics Engineers
- **MAC**: Media Access Control
- **MPDU**: MAC Protocol Data Unit
- **PCF**: Point Coordination Function
- **QoS**: Quality of Service
- **RTS/CTS**: Ready to Send / Clear to Send
- **SIFS**: Short Inter-Frame Space
- **STA**: Station
- **TCP**: Transmission Control Protocol
- **TPC**: Transmission Power Control
- **TXOP**: Transmission Oportunity
- **UDP**: User Datagram Protocol
- **VoIP**: Voice over IP

Capítulo 1

Introducción

En los últimos años, el despliegue de las tecnologías inalámbricas en el sector de las telecomunicaciones ha crecido de forma exponencial. La evolución actual de la sociedad hacia la sociedad de la información es una de las causas de dicho crecimiento. Entre las nuevas tecnologías de comunicación masivamente desplegadas, sobresalen las comunicaciones mediante dispositivos como smartphones u ordenadores portátiles que por su naturaleza requieren utilizar el medio inalámbrico.

Por otro lado, se ha producido un notable aumento en la demanda de aplicaciones multimedia tales como voz sobre IP (Voice over IP - VoIP), conferencias de audio o de vídeo. Estas aplicaciones necesitan que la red les ofrezca garantías de calidad de servicio (Quality of Service - QoS) en cuanto al ancho de banda, retardo de extremo a extremo, variación del retardo (jitter) o tasa de error. En concreto, la mayoría de las aplicaciones multimedia son sensibles a la variabilidad en la tasa de transmisión, al retardo y al jitter. Además de este tipo de aplicaciones, la red tiene que seguir dando servicio a aplicaciones clásicas (envío de e-mails, descarga de servidores FTP) más flexibles al retardo y a la disponibilidad de ancho de banda que las aplicaciones multimedia.

Las redes WiFi, basadas en el estándar 802.11, son una de las tecnologías inalámbricas más comunes hoy en día. En una red WiFi típica, tanto las aplicaciones multimedia como las clásicas compiten en igualdad de condiciones por el medio de transmisión inalámbrico, lo que dificulta el cumplimiento de las restricciones de QoS para las aplicaciones multimedia. Aunque en el propio estándar 802.11 se propone un mecanismo, denominado EDCA (Enhanced Distributed Channel Access), para dar prioridad a las aplicaciones con requerimientos de QoS más elevados, en muchos escenarios no es efectivo debido a la ausencia de coordinación entre los puntos de acceso (Access Points – AP) de la red.

Este trabajo propone un algoritmo centralizado para mejorar la QoS de los servicios que exigen menor latencia (servicios multimedia) en una red WiFi. El algoritmo ajusta de modo centralizado los principales parámetros que regulan el acceso al medio de los AP, con lo que se consigue priorizar los diferentes tipos de tráfico que aparecen en nuestra red de modo más efectivo que con la solución definida en el estándar. La validez de la solución propuesta se prueba mediante simulaciones, cuyos resultados muestran que el algoritmo propuesto mejora los resultados del mecanismo definido por el estándar, así como los de otras soluciones propuestas en el estado del arte.

1.1 Motivación y objetivos

La realización de este Trabajo Fin de Máster (TFM) se enmarca dentro del proyecto europeo *WI-5* [**1**], englobado dentro del programa de investigación *"Horizon 2020"* de la Unión Europea [**2**]. Este proyecto investiga nuevos métodos para mejorar las prestaciones de las redes Wi-Fi, especialmente el funcionamiento de servicios en tiempo real, como videojuegos online o llamadas de VoIP. El enfoque seguido en el proyecto es el de una solución virtualizada, en la que parte de las funcionalidades típicas de los AP se realizan de modo centralizado para mejorar su coordinación.

El objetivo de este proyecto es centralizar y mejorar el mecanismo EDCA propuesto en el estándar IEEE 802.11 [**3**] para priorizar los tipos de tráfico existentes en una red (FTP, VoIP, P2P), de modo que se satisfagan las demandas de QoS requeridas por las aplicaciones. La implementación se lleva a cabo en ns-3, simulador de licencia gratuita y de código abierto, ampliamente utilizado para simulación de redes.

Posteriormente, se comprueba su rendimiento en diferentes escenarios con tipos de tráfico variados para ver la mejora que consigue con respecto al EDCA, especialmente en escenarios con alta demanda de tráfico multimedia. Para ello, se simula una red WiFi con varias estaciones y AP cuyos parámetros de transmisión están fijados por el algoritmo propuesto. Los resultados obtenidos se comparan con los que se consiguen con el algoritmo que utiliza EDCA (solución por defecto) y con los de una solución distribuida.

1.2 Materiales y herramientas utilizadas

Las herramientas empleadas en este Trabajo Fin de Máster son las siguientes:

- **ns-3:** Simulador de red basado en eventos discretos de código libre. Permite simular redes WiFi con varios AP y estaciones que se envían tráfico de diversos tipos (UDP, TCP). Está escrito en el lenguaje de programación C++. En el Anexo A se encuentra una descripción detallada de su estructura, así como de sus características principales.

- **Matlab:** Herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje propio de programación. Se ha utilizado para la creación y diseño de las gráficas del Capítulo 4, así como para procesar todos los datos obtenidos de las simulaciones de dicho capítulo.

- **HERMES [4]:** Clúster de supercomputación utilizado para realizar las simulaciones del Capítulo 4, debido a su capacidad para ejecutar estas simulaciones en paralelo y a su gran velocidad de procesado.

1.3 Organización de la memoria

El contenido de la memoria se estructura de la siguiente forma:

- En el **Capítulo 1** se ha realizado una breve introducción del Trabajo fin de Máster y del objetivo final que se quiere alcanzar.

- En el **Capítulo 2** se define el contexto de nuestro trabajo, incluyendo una explicación del estándar utilizado, sus limitaciones y el estado del arte referido a los algoritmos que intentan solventar estas limitaciones.

- En el **Capítulo 3** se propone el algoritmo centralizado, desarrollando su estructura, analizando los pasos que sigue y explicando cómo se ha programado en ns-3.

- En el **Capítulo 4** se muestran los resultados obtenidos en los diversos escenarios propuestos para comprobar el funcionamiento del algoritmo, así como los cálculos y datos necesarios para la realización de dichos escenarios.

- En el **Capítulo 5** se comentan las conclusiones a las que hemos llegado con este trabajo, así como las posibles líneas futuras a seguir.

En la parte final se encuentra el **Anexo A,** en el cual se explica en detalle el funcionamiento del simulador ns-3. Este simulador ha sido empleado para programar la solución propuesta. En el Anexo también se muestra el código necesario para la creación de los escenarios usados en las pruebas del Capítulo 4.

Capítulo 2

Estado del arte

En este Capítulo vamos a analizar el estado del arte respecto a la provisión de QoS en redes 802.11. Para ello, realizaremos una revisión de los mecanismos propuestos por el propio IEEE para la incorporación de QoS en 802.11. De igual forma, presentaremos las principales contribuciones de la literatura científica que mejoran dichos mecanismos.

Comenzamos introduciendo el concepto de Calidad de Servicio y las diferentes formas en las que ésta se puede proveer dentro de una red 802.11. Posteriormente, nos centramos en la presentación del estándar IEEE 802.11e-EDCA [**3**], profundizando en el funcionamiento de la capa MAC y en la manera en que provee calidad de servicio. Seguidamente, comentamos las principales limitaciones del método EDCA para conseguir los niveles de calidad de servicio apropiados para finalmente analizar los métodos y algoritmos propuestos por la comunidad científica para tratar de paliar estas limitaciones.

2.1 Calidad de servicio en 802.11

Las redes basadas en IEEE 802.11 son redes inalámbricas en las que el canal se comparte entre todos los usuarios de la red. Esto hace necesario que incorporen un mecanismo de control de acceso al medio para evitar colisiones si más de un usuario quiere transmitir en el mismo intervalo de tiempo.

Los principales mecanismos propuestos en el estándar son CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) y RTS/CTS (Ready to Send / Clear to Send), los cuales se engloban dentro de DCF (Distributed Coordination Function). Sin embargo, este mecanismo de control de acceso al medio no distingue entre los distintos tipos de tráfico y no evita que un cliente pueda monopolizar el medio en mayor medida que el resto durante un intervalo no despreciable de tiempo, afectando negativamente a aplicaciones sensibles al retardo y al jitter, como son la voz sobre IP y las videoconferencias.

Una primera aproximación a la resolución a este problema consiste en un sistema de control llamado PCF (Point Coordination Function). Dicho mecanismo solo funciona en redes tipo infraestructura, debido a que es el AP el que controla el acceso al medio, por lo que no funciona en las redes ad-hoc. Cuando se activa el PCF, el tiempo que existe entre dos tramas "beacon"[1] enviadas por el punto de acceso, se divide en dos periodos denominados CFP (Content Free Period) y CP (Content Period). Durante el periodo CP el funcionamiento de la red es el normal (acceso al medio mediante CSMA/CA), pero durante el CFP los clientes sólo emiten cuando les llega un paquete enviado por el AP. El cliente aprovechará la oportunidad para emitir o para enviar un paquete indicando que no tiene nada que transmitir.

[1] Las tramas "beacon" son tramas que manda el AP periódicamente para anunciarse e informar de las características de la red

Con este método se evita que un cliente monopolice el medio de transmisión y se permite un reparto equitativo del tráfico entre todos los usuarios. Sin embargo, este sistema no es capaz de diferenciar los tipos de tráfico (solo diferencia a los clientes) y tratará igual tanto a un cliente que deba transmitir video, como al que transmita datos o voz. Además, muy pocos sistemas lo han implementado.

Como hemos visto en la introducción, la existencia de diferentes tipos de tráfico hace necesario diferenciarlos y aplicarles un tratamiento individual dependiendo de sus requisitos de QoS (ancho de banda, retardo, jitter). Este tratamiento va a implicar fundamentalmente la priorización de los tráficos que tengan unos requisitos de QoS más estrictos. Por ejemplo, en el caso de que tuviésemos simultáneamente una descarga de un archivo mediante FTP (tráfico TCP de background) y una llamada de VoIP, es necesario priorizar la llamada sobre la descarga. Esto va a hacer necesario que el AP o el controlador de la red Wi-Fi tenga mecanismos para decidir qué paquetes se han de transmitir en cada momento, de tal manera que la latencia y el jitter de la llamada tengan valores adecuados.

Para conseguir este objetivo, en un primer momento empresas como Cisco [**5**] desarrollaron protocolos propietarios. Sin embargo, en un entorno como el de las redes Wi-Fi, donde solo es posible tener control sobre los AP y no sobre los clientes, no resultaban funcionales ni obtenían los resultados deseados. Fue con la llegada del protocolo 802.11e y del mecanismo EDCA cuando la QoS entró en IEEE 802.11.

2.2 QoS en IEEE 802.11e-EDCA

La QoS que proporciona 802.11e-EDCA se basa en dar mayor o menor prioridad a las distintas tramas MAC o MPDUs (MAC Protocol Data Units) en función de la clase de tráfico a la que pertenezcan. Todas las tramas que pertenecen a la misma clase de tráfico acceden al medio físico en igualdad de condiciones de acuerdo a la categoría de acceso (Access Categories - AC) que se le asigne de entre cuatro posibles. Así, las tramas pertenecientes a la categoría con mayor prioridad esperan en media menos tiempo para acceder al medio. Además, una vez que ganan el acceso al medio, pueden mantenerlo ocupado un mayor periodo de tiempo. A continuación, se explica con mayor detalle cómo se consigue esta diferenciación de servicio.

EDCA utiliza cuatro AC para manejar las distintas prioridades, lo que permite diferenciar entre cuatro clases diferentes de tráfico. Las cuatro AC que define el estándar son, de mayor a menor prioridad: $\boldsymbol{AC\_VO}$ (voz), $\boldsymbol{AC\_VI}$ (video), $\boldsymbol{AC\_BE}$ (best-effort) y $\boldsymbol{AC\_BK}$ (background).

Tal y como se muestra en la Figura 2.1, cada i-ésima categoría de acceso tiene su propia cola de transmisión, caracterizada por los siguientes parámetros:

- $AIFSN_i$ (Arbitration Inter-Frame Space Number): Intervalo de tiempo mínimo para que la AC i considere que el canal está libre.
- $CW_{min,i}$ (Minimum Contention Window): Tamaño mínimo de la ventana de contienda.
- $CW_{max,i}$ (Maximum Contention Window): Tamaño máximo de la ventana de contienda.
- $TXOP_i$ (Transmision Oportunity): intervalo de tiempo durante el cual una estación que ha ganado el acceso al medio puede transmitir de modo continuo una serie de tramas de la AC i.

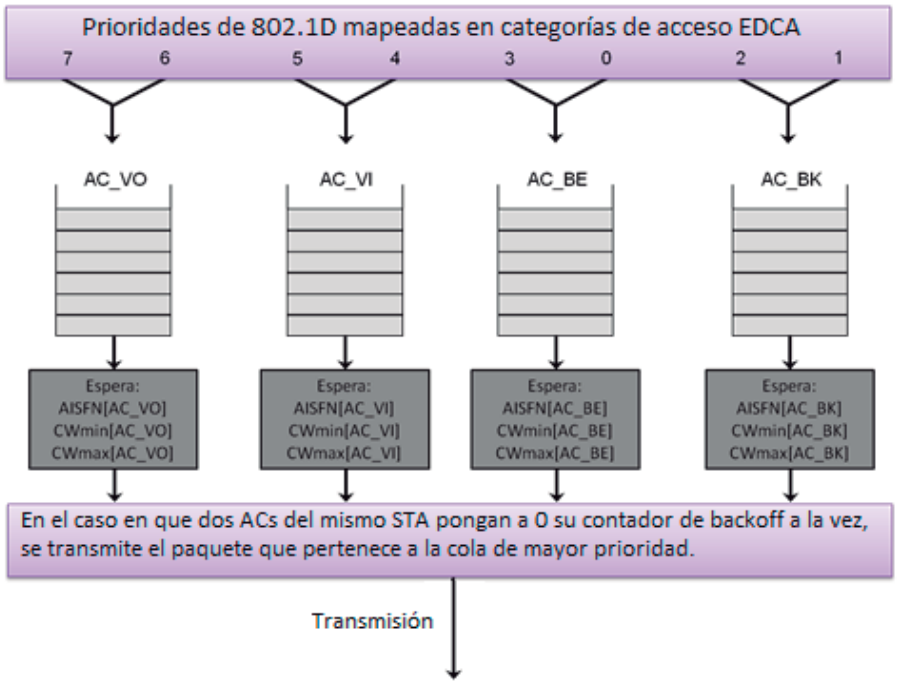


Figura 2.1: Esquema de EDCA [6]

Una estación que quiere transmitir una trama perteneciente a la i-ésimo AC en primer lugar monitoriza el canal de transmisión. Si el canal está libre durante un intervalo de tiempo igual al parámetro $AIFS_i = SIFS + AIFSN_i \cdot \sigma$, entonces transmite. En la expresión anterior, σ es el tamaño de slot (slot time) en el proceso de backoff y SIFS es un valor fijo denominado Short Inter-Frame Space, que es el tiempo mínimo requerido por cada estación para comenzar a transmitir cualquier tipo de trama.

Por el contrario, si el canal se encuentra ocupado, la estación deberá empezar un proceso de backoff. Para ello, calcula un número aleatorio entero uniformemente distribuido en el intervalo $[0, CW_i - 1]$, inicializando un contador a ese valor. El parámetro CW_i se denomina ventana de contienda y depende de la AC de la trama y del número de colisiones que ha sufrido la trama con anterioridad. En el primer intento de transmisión, se establece $CW_i = CW_{min,i}$.

Cuando el canal vuelve a estar libre, el contador se decrementa en una unidad cada σ segundos. Si se detecta una transmisión, el contador se congela y no se reactiva hasta que se detecta el canal otra vez libre durante un intervalo de tiempo igual a $AIFS_i$.

Cuando el contador alcanza el valor 0, la estación transmite y espera a recibir un ACK del destino, que confirmará la trama si le ha llegado correctamente.

Si el ACK llega antes de un tiempo definido como ACK_Timeout, la transmisión se considera exitosa. En caso contrario se retransmite la trama siguiendo el mismo mecanismo, pero doblando el valor de CW_i (hasta un valor máximo dado por el parámetro $CW_{max,i}$). Este proceso se repite tantas veces como indique el límite máximo de retransmisiones.

Una colisión tiene lugar cuando dos o más estaciones transmiten simultáneamente. Si dos AC diferentes intentan transmitir simultáneamente en la misma estación, se produce una colisión interna que se resuelve permitiendo que la clase más prioritaria transmita, mientras que la otra es tratada como si ocurriera una colisión real.

Finalmente, si la transmisión es exitosa, la estación puede continuar enviando tramas de ese AC durante un límite de tiempo acotado por el parámetro $TXOP_i$. Si $TXOP_i = 0$, la estación únicamente podrá transmitir una trama cuando gane el acceso al canal.

La Tabla 2.1 muestra los valores por defecto de los diferentes parámetros, en función de la categoría de acceso.

Tabla 2.1: Parámetros EDCA por defecto del estándar IEEE 802.11e ***[3].***

AC	$\boldsymbol{CW_{min}}$	$\boldsymbol{CW_{max}}$	***AIFSN***	***Max TXOP***
Background (*AC_BK*)	aCW_{min}	aCW_{max}	7	0
Best Effort (*AC_BE*)	aCW_{min}	aCW_{max}	3	0
Video (*AC_VI*)	$(aCW_{min}+1)/2 -1$	aCW_{min}	2	3.008ms
Voice (*AC_VO*)	$(aCW_{min}+1)/4 -1$	$(aCW_{min}+1)/2 -1$	2	1.504ms

Para el caso típico de $aCW_{min} = 15$ y $aCW_{max} = 1023$, los valores resultantes son los que se muestran en la Tabla 2.2:

Tabla 2.2: Parámetros EDCA por defecto del estándar IEEE 802.11e con aCWmin = 15 y aCWmax = 1023.

AC	$\boldsymbol{CW_{min}}$	$\boldsymbol{CW_{max}}$	***AIFSN***	***Max TXOP***
Background (*AC_BK*)	15	1023	7	0
Best Effort (*AC_BE*)	15	1023	3	0
Video (*AC_VI*)	7	15	2	3.008ms
Voice (*AC_VO*)	3	7	2	1.504ms

En la Figura 2.2 podemos ver una representación del valor mínimo que tendrá que esperar una trama de cada tipo de servicio si encuentra la cola correspondiente a su AC vacía y (i) encuentra el canal libre (en verde en la figura), (ii) encuentra el canal ocupado (en azul), teniendo en cuenta las fórmulas expuestas anteriormente y utilizando los valores por defecto que se muestran en la Tabla 2.2. Esto permite hacernos una idea de la diferencia de tiempos de espera para transmitir de acuerdo a las diferentes categorías de acceso:

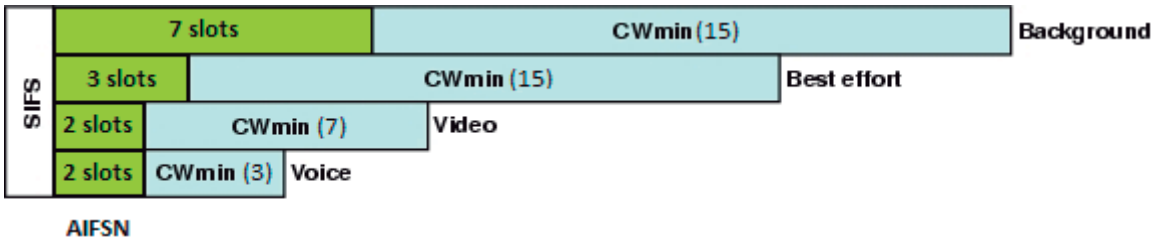


Figura 2.2: Representación de la suma de AIFS y CW_{min} por cada cola [6]

Finalmente, es importante señalar que la última versión del estándar IEEE 802.11 (estándar 802.11-2012 [**3**]) incluye todas las mejoras propuestas en el estándar 802.11e, por lo que el EDCA también está incluido. Este mecanismo será el que utilizaremos en las simulaciones del Capítulo 4 como base para comparar los resultados obtenidos con la solución propuesta en este TFM.

2.3 Limitaciones de EDCA

Existen algunas características en el diseño de EDCA que, bajo ciertas condiciones, hacen que este mecanismo no sea efectivo para proveer la QoS necesaria a las aplicaciones y servicios multimedia.

Estas limitaciones nacen del hecho de que en EDCA, la clasificación de los paquetes por AC es sólo para los flujos de tráfico internos, por lo que no se puede gestionar la prioridad entre las transmisiones provenientes de diferentes estaciones dentro del mismo dominio de colisión. Este inconveniente puede conducir a los siguientes problemas:

- **Parámetros por defecto de EDCA:** Los parámetros por defecto de EDCA, en especial CW_{min}, no ofrecen la suficiente diferenciación de tráfico. Un número elevado de tráficos de baja prioridad en estaciones diferentes haría que un único tráfico de alta prioridad no funcionase correctamente.

- **Asignación de CW en tráficos menos prioritarios:** Si una red se encuentra muy cargada de tráfico, las actualizaciones constantes de CW_i (al enviar correctamente un paquete, CW_i pasa a ser $CW_{min,i}$, mientras que, por cada paquete colisionado, CW_i se va duplicando) en los tipos de AC menos prioritarios (***AC_BK***, ***AC_BE***) afectan negativamente al funcionamiento de los tráficos más prioritarios.

- **Diferenciación intra-AC (no inter-AC):** EDCA no proporciona mecanismos para que cada AP pueda saber el estado de los otros AP de la red. Por lo tanto, la probabilidad de que varios AP que tengan tráfico de la misma prioridad (mismo AC) elijan los mismos valores de backoff aumenta conforme crece el número de AP, lo que conduce a un incremento en el número de colisiones.

Una posible solución para los dos primeros problemas sería editar manualmente el valor de $CW_{min,i}$ para imponer un CW mínimo mayor a los tráficos menos prioritarios. Esto haría que funcionasen mejor los más prioritarios, obteniendo una priorización más eficiente.

Respecto al tercer problema, la mejor manera de resolverlo es emplear un algoritmo centralizado que tenga en cuenta el estado de todos los AP de la red para fijar sus parámetros de acceso al medio.

2.4 Algoritmos propuestos en la literatura

Para tratar de corregir las limitaciones que presenta EDCA, se han presentado multitud de trabajos con propuestas de soluciones. En este apartado vamos a analizar las más importantes a nuestro parecer.

Como se ha visto en la sección anterior, los parámetros que permiten modificar el funcionamiento por defecto de EDCA son cuatro: $AIFSN_i, CW_{min,i}, CW_{max,i}$ y $TXOP_i$. Estos parámetros tienen establecido un valor por defecto para cada una de las categorías de acceso, tal y como se ha mostrado en la Tabla 2.1, e influyen directamente en la QoS de las tramas enviadas, afectando a su ancho de banda, retardo y jitter.

El mecanismo básico para priorizar el acceso al canal que considera la especificación EDCA es la variación de los parámetros CW_{min} y CW_{max}, tal y como se menciona en [**8**]. Estos parámetros poseen la suficiente versatilidad para gestionar la priorización de los flujos y se pueden configurar dinámicamente en cada categoría de acceso de cada AP. En cuanto a los otros dos parámetros mencionados, $AIFSN$ y $TXOP$ no son muy utilizados en los algoritmos encontrados en la literatura.

Según se indica en [**9**] (donde se presenta una comparativa de varios algoritmos) el parámetro CW_{min} es fundamental para la priorización de las colas de transmisión. Valores muy bajos de CW_{min} implican altas probabilidades de colisión, por lo que el rendimiento sería bajo. Por otro lado, valores elevados de CW_{min} evitarían las colisiones a costa de esperar un número excesivo de ranuras de tiempo, lo que también influiría negativamente en el rendimiento. Por regla general, el valor óptimo del parámetro CW_{min} depende del número de nodos activos en la red y se debe fijar de tal modo que se minimice el retardo que sufren las tramas y se maximice el caudal efectivo cursado por la red.

Los algoritmos que existen en la literatura se pueden clasificar de diversos modos. Una primera clasificación se haría en función de si son estáticos o adaptativos. Los algoritmos estáticos (como el que utiliza EDCA por defecto) son ineficientes al no poder adaptarse a los cambios en la red. Los adaptativos, en cambio, permiten seleccionar los parámetros adecuados para mejorar la calidad de servicio de la red en cualquier momento.

Por otro lado, también se puede distinguir entre algoritmos iterativos o no iterativos. Los algoritmos iterativos permiten encontrar la solución óptima, pero en la mayoría de casos no pueden trabajar en tiempo real ya que el tiempo necesario para obtener dicha solución es muy elevado. A pesar de ello, existen aproximaciones heurísticas que mitigan este problema. Los algoritmos no iterativos obtienen la solución con respecto a métricas aplicadas a la red, como número de flujos o retardo medio. Estos pueden trabajar en tiempo real, aunque la solución no es tan exacta como la que se consigue con los iterativos.

A continuación, se explican con más detalle los algoritmos más interesantes encontrados en el estado del arte y que han servido de base para el algoritmo propuesto.

El primer algoritmo que se ha analizado ha sido el que se propone en [**10**]. Este algoritmo se basa en una configuración dinámica de los parámetros CW_{min} y CW_{max} en función del número de estaciones conectadas al AP. La configuración propuesta es muy conservadora y otorga una gran prioridad a los paquetes de VoIP. Los principales cambios con respecto a los parámetros por defecto (los que aparecen en la Tabla 2.2) para el ***AC_VO*** son:

- $TXOP_v$ = 4,512 ms en lugar de $TXOP_v$ = 1,504 ms
- $CWmin_{v,d} = CWmax_{v,d} = 32$ slots en lugar de $CWmin_{v,d} = 7$ slots y $CWmax_{v,d} = 15$ slots
- $CWmin_{v,u} = CWmax_{v,u} = 64$ slots en lugar de $CWmin_{v,u} = 7$ slots y $CWmax_{v,u} = 15$ slots

Por otro lado, para el tráfico elástico (***AC_BK***) propone los cambios indicados a continuación:

- $AIFSN_{e,d} = 15$ slots, en lugar de $AIFSN_{e,d} = 2$ slots.
- $AIFSN_{e,u} = 15$ slots, en lugar de $AIFSN_{e,u} = 2$ slots.

En los parámetros anteriores, se ha seguido la siguiente notación:

Subíndice v : cola AC_VO
Subíndice e : cola AC_BK (tráfico elástico)

Subíndice d : downlink (enlace descendente)
Subíndice u : uplink (enlace ascendente)

El algoritmo realiza los siguientes pasos:

Paso 1:
Inicializa los parámetros MAC con los valores mencionados anteriormente.

Paso 2:
Bucle. Espera a una nueva petición de flujo.

Paso 3:
Almacena $CW_{min,e}$ y $CW_{max,e}$

Paso 4:
Establece:

a. $CW_{min,e}$ = min (1024, número de STA con tráfico elástico * 16)
b. $CW_{max,e}$ = min (1024, 32 * $CW_{min,e}$)

Paso 5:
Estima el rendimiento de la red en términos de ancho de banda.

Paso 6:
Si el rendimiento es aceptable en ese momento, se acepta el flujo entrante. Si no es aceptable, rechaza el flujo entrante y restaura los valores guardados en el paso 3 de $CW_{min,e}$ y $CW_{max,e}$. Fin del bucle.

Como podemos ver, este algoritmo es adaptativo y no iterativo, ya que trabaja con los flujos de tráfico elástico que existen en la red y no se ejecuta cada cierto tiempo de modo periódico.

El segundo algoritmo analizado es el propuesto en **[11].** En este trabajo se propone un algoritmo local que se ejecuta en cada AP de la red y que ajusta de modo adaptativo el valor de CW_{min} dependiendo de los flujos de datos existentes en el AP. La función a maximizar es el throughput de cada AP teniendo en cuenta la prioridad del tráfico más sensible (en este caso voz) y la presencia de tráfico de los tipos $\boldsymbol{AC\_BK}$ y $\boldsymbol{AC\_VO}$.

El algoritmo fija todos los parámetros a los valores por defecto de EDCA vistos en la Tabla 2.2, excepto los parámetros CW_{min} y CW_{max} de la cola $\boldsymbol{AC\_BK}$, que son variables y dependen del número de estaciones que transmiten ese tipo de tráfico. La adaptación del parámetro CW_{min} se basa en un incremento adaptativo del mismo dependiendo del estado de la red y de su rendimiento en ese momento.

Los pasos que realiza el algoritmo son los siguientes:

Paso 0:
Inicializa los parámetros MAC y los parámetros del algoritmo:
W = 320; $\Delta SF = 0.1$; SF = ΔSF; a=0.95

Paso 1.
El factor de escalado (SF) se establece en un valor inicial $SF1 = \Delta SF$, y se mide el throughput agregado $R1$ en el AP cada Tm segundos.
Editamos el valor de CW_{min} del siguiente modo: $CW_{min,e} = W*SF$

Paso 2.
$SF2 = SF1 + \Delta SF$, y medimos el throughput agregado $R2$ en el AP.

Paso 3.
Si $R2 > R1$, *SF se incrementa* un paso ΔSF mientras el throughput agregado aumente
Si $R2 < R1$, *SF disminuye un paso* ΔSF mientras el throughput agregado aumente

Paso 4.
$SF*$ es el factor de escalado óptimo, y $R*$ el throughput cuando acaba el Paso 3.
Hacemos $SF1 = SF*$
Para el mismo factor de escalado, el AP continuamente mide el throughput agregado $R1$. Si $R1 < aR*$, donde $a \in (0, 1)$ entonces volvemos al Paso 2.

El algoritmo se ejecuta hasta que los requisitos de QoS para el tráfico sensible (voz) se cumplen, fijando el valor de los parámetros a los que hay en ese momento. Si el tráfico elástico aumenta, el cambio se detectaría por el AP y los parámetros se recalcularían automáticamente.

Este algoritmo es iterativo, ya que el algoritmo se ejecuta cada T_m segundos y tiene en cuenta el throughput agregado para modificar o no los parámetros.

Además, es adaptativo, ya que varía los parámetros CW_{min} y CW_{max} si se cumple una determinada condición.

Tras analizar estos algoritmos, decidimos tomar como base el segundo de los dos, debido a que mantiene fijos la mayoría de los parámetros por defecto de EDCA y sólo modifica los valores de CW_{min} y CW_{max} de la cola $\boldsymbol{AC\_BK}$. Además, según la comparativa realizada en **[9],** los resultados que obtiene el segundo algoritmo son mejores que los que consigue el primero en los escenarios analizados.

Capítulo 3

Algoritmo propuesto

En este capítulo vamos a desarrollar el algoritmo que hemos diseñado en este Trabajo fin de Máster.

En primer lugar, se van a analizar los diversos problemas que tienen los algoritmos del apartado anterior que hemos tomado como base para la realización del que proponemos en este capítulo. Posteriormente, se explicará en detalle el algoritmo propuesto, explicando todos los pasos que sigue durante su ejecución y por qué se ha decidido que siga ese modelo de funcionamiento. Finalmente, se explicará cómo ha sido implementado en ns-3, el entorno de desarrollo que vamos a utilizar para las pruebas del siguiente capítulo.

3.1 Limitaciones de los algoritmos presentados en el estado del arte

Los algoritmos analizados en el estado del arte del capítulo anterior tienen los siguientes problemas o limitaciones:

- Se ejecutan en cada AP por separado (son locales) y no utilizan la información de los AP vecinos que podrían estar influyendo negativamente en el AP analizado. Un algoritmo centralizado permitiría modificar los parámetros en varios AP simultáneamente teniendo en cuenta la interrelación que hay entre ellos.

- Aumentan el valor de CW_{min} de forma aritmética (sumando una cierta cantidad cada vez a CW_{min}), lo que puede provocar que tarden demasiado en priorizar el tráfico más sensible (por ejemplo, una llamada de voz sobre IP) con respecto al tráfico elástico. Un aumento de tipo geométrico (multiplicar por una cantidad fija el valor de CW_{min}) permitiría un ajuste más rápido.

- Modifican los parámetros de EDCA en función del throughput de la red y no del retardo. En el caso de los tráficos sensibles (VoIP, videojuegos online, etc.) tan importante o más que el throughput es mantener el retardo en unos límites aceptables, ya que una llamada de voz con un retardo muy elevado tendrá una experiencia de uso muy deficiente, aunque su throughput sea el necesario para realizar la llamada.

3.2 Algoritmo propuesto

El algoritmo que proponemos está basado en el segundo analizado en el capítulo 2. Hemos incorporado las siguientes variaciones al mismo para incorporar las ventajas que ofrecen un sistema centralizado y resolver los problemas detectados en el apartado anterior:

- Realizamos una actualización geométrica (en vez de lineal) del parámetro CW_{min} (duplicando su valor) cada vez que no se satisfacen los requisitos de la red. Esto permite una mayor y más rápida priorización de los tráficos más sensibles. Además, dividimos por dos el parámetro CW_{min} si durante un intervalo de tiempo mayor se satisfacen los requisitos de la red de una forma más estricta.

- La reducción a la mitad de CW_{min} se produce cuando ha pasado un número configurable de segundos en los que el tráfico sensible cumple los requerimientos marcados, en lugar de hacerlo directamente en la siguiente iteración, ya que el throughput global de la red no es tan importante en nuestro caso como lo es el garantizar el retardo medio de los paquetes prioritarios. Esto se realizar para garantizar cierta estabilidad en el algoritmo y no reaccionar en exceso, teniendo en cuenta que la prioridad no es el throughput global, por lo que el tráfico TCP podría funcionar un poco peor durante dicho número de segundos para garantizar el buen funcionamiento del tráfico sensible.

- El algoritmo propuesto es centralizado y adapta el valor de CW_{min} en función de las condiciones de toda la red y no solo de un único AP. Con esto solventamos una de las mayores limitaciones que aparecían en los artículos analizados, que era el hecho de que sólo podían controlar a un único AP.

El principio de funcionamiento del algoritmo se basa en las siguientes reglas:

1. Cada $\boldsymbol{T}_{update}^{(1)}$ se mide el retardo medio del tráfico perteneciente a la categoría de acceso $\boldsymbol{AC\_VO}$ de cada uno de los AP de la red.

2. Si alguno de dichos retardos medios es mayor que $\boldsymbol{D_{max}}$, realizamos las siguientes modificaciones en las colas $\boldsymbol{AC\_BK}$ de los AP que han superado el retardo y también en las colas $\boldsymbol{AC\_BK}$ de los AP que les interfieren:

   a. $CW_{min} = 2^* CW_{min}$ (hasta llegar a $CW_{min} = 1023$)
   b. $CW_{max} = 1023$

   De este modo damos mayor prioridad al tráfico de voz que está sufriendo un retardo demasiado alto ya que accede al medio con más prioridad (más exactamente, el resto de tráfico accede con menos prioridad). Para evitar repetir modificaciones en el caso en el que el mismo AP interfiera o sea interferido varias veces, sólo se modifica como máximo una vez cada AP. Es decir, en cada iteración, una vez que se detecta que un AP debe modificar su cola $\boldsymbol{AC\_BK}$ (por sí mismo o por interferir a otros), esta modificación es única.

3. Por otro lado, cada $\boldsymbol{T}_{update}^{(2)}$ segundos se comprueba si la media del retardo en cada AP y en los AP que le interfieren es menor de $\boldsymbol{D_{min}}$. En caso afirmativo, realizamos las siguientes modificaciones en la cola $\boldsymbol{AC\_BK}$ de todos ellos:
   a. $CW_{min} = CW_{min}$ /2 (hasta llegar a $CW_{min} = 15$)
   b. $CW_{max} = 1023$

El objetivo de esta actualización es que, si todo el tráfico prioritario está siendo servido de modo satisfactorio, el tráfico no prioritario tenga que esperar menos para transmitir y de este modo aumentar el throughput total de la red.

Este caso es especialmente importante cuando el tráfico de la cola de mayor prioridad disminuye o desaparece. En ese caso, si el algoritmo no hiciese nada para evitarlo, el tráfico de la cola ***AC_BK*** seguiría con un valor elevado de CW_{min} a pesar de que ya no necesitaríamos dicho valor tan elevado porque el tráfico de la cola ***AC_VO*** habría disminuido.

Es importante que $\boldsymbol{D_{max}}$ tenga un valor bastante superior a $\boldsymbol{D_{min}}$ para introducir una histéresis que evite comportamientos inestables del algoritmo. Además, el intervalo elegido para disminuir CW_{min} ($\boldsymbol{T}_{update}^{(2)}$) es superior al que se emplear para aumentarlo ($\boldsymbol{T}_{update}^{(1)}$), lo que aun refuerza más la estabilidad del algoritmo.

En la Figura 3.1 se muestra el diagrama de flujo del algoritmo.

A continuación, indicamos los valores numéricos que se han escogido para los parámetros del algoritmo:

- $\boldsymbol{D_{max}}$ **(Retardo máximo de paquete)**: 100 ms
- $\boldsymbol{T}_{update}^{(1)}$ **(Intervalo de actualización del algoritmo):** 1 s
- $\boldsymbol{D_{min}}$ **(Retardo mínimo para adaptación):** 20 ms
- $\boldsymbol{T}_{update}^{(2)}$ **(Intervalo de comprobación adaptativo):** 10 s

El retardo máximo de paquete lo fijamos conforme a la ITU-T G-114 [**7**], que establece que a partir de los 100 ms dicho retardo puede afectar a aplicaciones interactivas, como por ejemplo aplicaciones vocales, de videoconferencia o de datos interactivos.

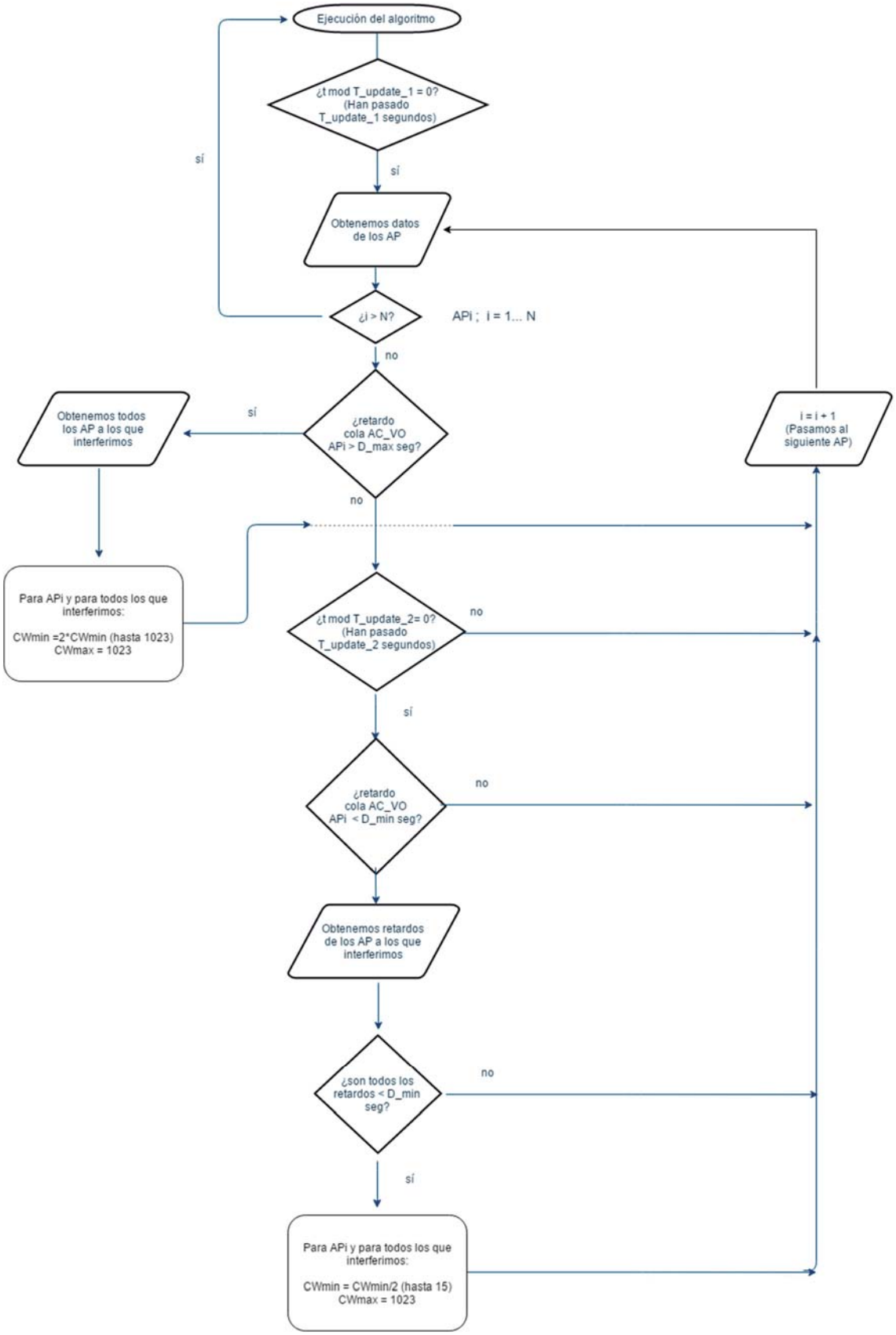


Figura 3.1. Diagrama de flujo del algoritmo.

En cuanto al intervalo de actualización del algoritmo, se ha fijado tras una serie de pruebas realizadas que aparecen en siguiente capítulo.

Respecto al intervalo de comprobación adaptativo (recordamos que se utiliza para comprobar si el tráfico sensible funciona correctamente para, en caso afirmativo, dividir CW_{min} por dos), se ha escogido un valor 10 veces superior al intervalo de actualización (10 segundos), para evitar posibles inestabilidades en el valor de CW_{min} y por tanto en el retardo.

El retardo mínimo de paquete para que se ejecute esta adaptación es de 20 milisegundos, de tal modo que solo mejora el tráfico no sensible (TCP de background) cuando se pueda permitir aumentar un poco el retardo medio del tráfico sensible. En el capítulo siguiente se analiza el impacto de estos dos parámetros mediante una serie de simulaciones.

En resumen, gracias al conocimiento que tiene la red de todos los AP que interfieren entre sí y del tráfico que hay en cada uno de ellos, el algoritmo centralizado garantiza que el tráfico de baja prioridad de un AP no interfiera con el tráfico de alta prioridad de un AP adyacente, por lo que se mejora el funcionamiento global de la red. Esta es la principal ventaja de la solución propuesta cuando se compara con las soluciones que se han propuesto con anterioridad.

3.3 Implementación del algoritmo en el sistema.

La implementación del algoritmo propuesto supone el empleo de una arquitectura centralizada como la de la Figura 3.2. en la que los parámetros de acceso al medio de los AP se establecen remotamente por un controlador central.

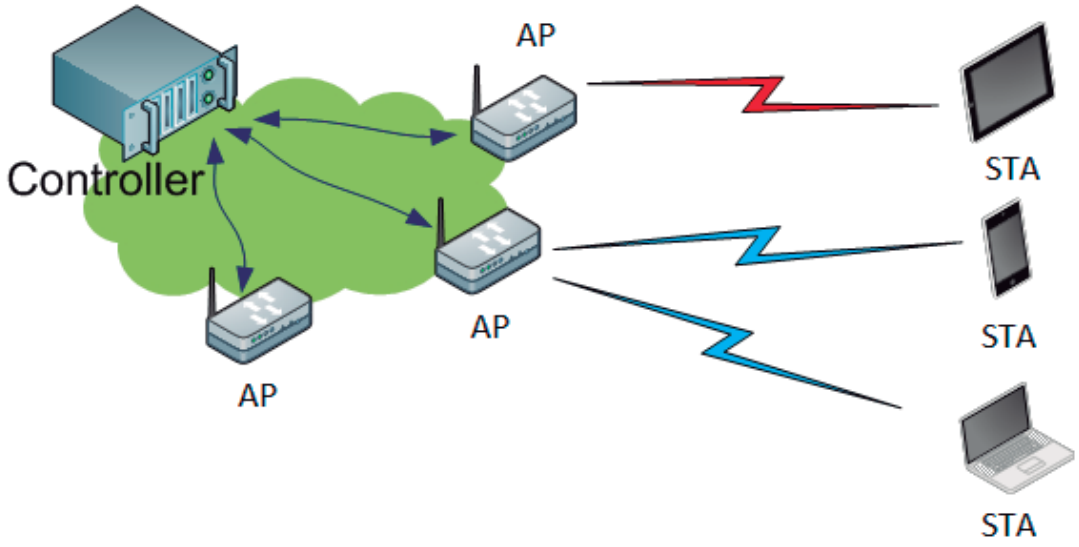


Figura 3.2. Arquitectura centralizada.

El empleo de una solución centralizada implica la necesidad de incorporar un mecanismo de señalización al sistema para que los AP envíen periódicamente el retardo que sufren los paquetes en la cola de la categoría de acceso para voz (***AC_VO***). Con esta información, el controlador central ejecuta el algoritmo y devuelve a los AP el parámetro de CW_{min} que han de emplear en su cola correspondiente al tráfico elástico (***AC_BK***).

Por otra parte, la correcta ejecución del algoritmo propuesto implica que el controlador conoce cómo se interfieren los AP entre sí. Esta información puede ser conocida a priori por el controlador si la fase de diseño de la red se ha realizado empleando alguna herramienta de planificación en la que se ha calculado las zonas de cobertura y de interferencia de cada AP. Por otro lado, esta información también se puede obtener una vez que la red está en funcionamiento mediante la monitorización por parte de los AP de otros AP que estén transmitiendo en el mismo canal. El propio estándar 802.11 define mecanismos específicos para realizar este tipo de medidas a través del envío de las tramas TPC (Transmission Power Control) Request / Report o la propia monitorización de las tramas de beacon enviadas por las estaciones.

En resumen, la solución propuesta es viable desde un punto vista técnico y la señalización necesaria para la comunicación entre los AP y el controlador central no es muy elevada al necesitarse enviar únicamente tres tipos de datos: retardo en la cola ***AC_VO*** (de los AP al controlador), AP interferentes (de los AP al controlador) y valor del parámetro CW_{min} (del controlador a los AP).

3.4 Implementación del algoritmo en ns-3

El siguiente paso a realizar consiste en programar el algoritmo propuesto en este capítulo en el simulador de red ns-3. Este simulador, como se ha comentado en el Capítulo 1, es un simulador de red de código libre basado en eventos discretos.

En el Anexo A se ha realizado un completo análisis del simulador y del modo en que se han generado los escenarios en los que se ha probado el algoritmo. Por el contrario, en este apartado nos centramos en la programación del algoritmo propuesto en el simulador ns-3.

En cuanto al algoritmo en sí, su programación se realizó en los siguientes pasos:

En primer lugar, nos encargamos de calcular el retardo medio y el throughput de los paquetes que pasan por la cola ***AC_VO*** de cada AP, así como el porcentaje de paquetes perdidos (estos cálculos se muestran con detalle en el Anexo A), ya que nos serán útiles para analizar el funcionamiento de la red en cada uno de los escenarios.

Obteniendo el dato del retardo medio de la cola ***AC_VO*** de todos los AP, ya podemos comenzar a ejecutar el algoritmo:

Para cada AP, se realizan los pasos ya explicados en el apartado anterior y cuyo código es el siguiente:

```
if(delayMedioUDPDownlink[i]/numeroSTA_UDP>D_max){
   if(matrizInterferencia[i][j] > 0){
      if(vectorYaActivado[matrizInterferencia[i][j]] == false){
            vectorYaActivado[matrizInterferencia[i][j]] = true;

            if ((parametros[matrizInterferencia[i][j]][0]*2) <= 1023){
                  parametros[matrizInterferencia[i][j]][0]=2*parametros
                  [matrizInterferencia[i][j]][0]+1;//CWminBK
            }

            else{
                  parametros[matrizInterferencia[i][j]][0] = 1023;
            }
            parametros[gruposInterferencia[i][j]-1][4]=1023;//CWmaxBK
            SetCwMin(apNodes[matrizInterferencia[i][j]].Get(0),
            parametros[matrizInterferencia[i][j]]);
      }
   }
}
```

Esto es, en primer lugar, comprobamos si el retardo medio de los paquetes de la cola $\boldsymbol{AC\_VO}$ del AP_i es mayor a $\boldsymbol{D_{max}}$ **(línea 1)**. Si es así, obtenemos de la matriz de interferencias los AP que interfieren al que estamos analizando (línea 2). La variable `vectorYaActivado` se encarga de obligar a que sólo se ajusten una vez los parámetros de cada AP (línea 3).

La matriz de interferencias es una matriz que indica los AP que interfieren a cada AP. Por ejemplo, para el caso de tener 6 AP tenemos:

```
int matrizInterferencia[6][6] = {   { 1,2,6,0,0,0 },
                                    { 1,2,3,0,0,0 },
                                    { 2,3,4,0,0,0 },
                                    { 3,4,5,0,0,0 },
                                    { 4,5,6,0,0,0 },
                                    { 1,5,6,0,0,0 } };
```

En este ejemplo, esta matriz indica que el AP_1 (fila 1) es interferido e interfiere al AP_2 y al AP_6, y así sucesivamente.

La variable `parametros` es una matriz que contiene los parámetros que queremos modificar de todos los AP del escenario, ordenados por colas.

Cada fila de la matriz corresponde a un AP y contiene los siguientes parámetros para dicho AP:

$$[CW_{minBK}, CW_{maxBK}, AIFSN_{BK}, CW_{minBE}, CW_{maxBE}, AIFSN_{BE},$$

$$CW_{minVI}, CW_{maxVI}, AIFSN_{VI}, CW_{minVO}, CW_{maxVO}, AIFSN_{VO}]$$

Un ejemplo de esta variable para el caso de que existan 6 AP es el siguiente:

```
parametros[6][12] =  {  {15, 15, 7, 3, 1023, 1023, 15, 7, 7, 3, 2, 2},
                        {15, 15, 7, 3, 1023, 1023, 15, 7, 7, 3, 2, 2},
                        {15, 15, 7, 3, 1023, 1023, 15, 7, 7, 3, 2, 2},
                        {15, 15, 7, 3, 1023, 1023, 15, 7, 7, 3, 2, 2},
                        {15, 15, 7, 3, 1023, 1023, 15, 7, 7, 3, 2, 2},
                        {15, 15, 7, 3, 1023, 1023, 15, 7, 7, 3, 2, 2}};
```

Aunque nuestro algoritmo sólo modifica los valores de CW_{minBK}, se podría modificar algún otro parámetro en el futuro si fuera necesario.

La función `SetCwMin` asigna el valor modificado del parámetro al AP escogido. Su funcionamiento se detalla en el Anexo A.

La programación del mecanismo de adaptación que se produce cada $\boldsymbol{T}_{update}^{(2)}$ en caso de que los paquetes de la cola $\boldsymbol{AC\_VO}$ estén cumpliendo de modo holgado sus requisitos de QoS (retardo por debajo de D_{min}) es la siguiente:

```
if(modulo_segundo == 0){
  for(int indice = 0; indice < 6; indice++){
    if(matrizInterferencia[i][indice] >= 0){
      if(( (DelayAcum, (matrizInterferencia[i][indice])/T_u_2) >
D_min){
            bajarCW = false;
      }
    }
  }

  if(bajarCW == true){
    if(vectorYaActivado[i] == false){
      vectorYaActivado[i] = true;
      if ((parametros[i][0]-1)/2 > 14){
            parametros[i][0]=(parametros[i][0]-1)/2;//CWminBK
      }
      else{
            parametros[i][0] = 15;
      }

      parametros[i][4]=1023;//CWmaxBK
      SetCwMin(apNodes[i].Get(0), parametros[i]);
  }
  }
  else{
      bajarCW = true;
  }
}
```

En primer lugar, comprobamos si han pasado $\boldsymbol{T}_{update}^{(2)}$ segundos desde la última vez que se ejecutó este código. Continuamos comprobando si en todos los AP que interfieren al AP en el que se encuentra el bucle (incluyendo el propio AP) el retardo medio de los paquetes de la cola $\boldsymbol{AC\_VO}$ es menor a $\boldsymbol{D_{min}}$ en los últimos $\boldsymbol{T}_{update}^{(2)}$ segundos. Si en algún caso esto no ocurre, se establece la variable booleana `bajarCW` a false y no se ejecuta el mecanismo. Si, por el contrario, todos los AP analizados cumplen esta condición, se modifica el parámetro CW_{min} del AP en el que se encuentra el bucle.

Capítulo 4

Evaluación del algoritmo

En este capítulo vamos a comprobar el funcionamiento del algoritmo propuesto en diferentes escenarios de simulación, cada uno con características distintas.

En primer lugar, se presentan y motivan los parámetros utilizados en las simulaciones. Posteriormente se representan y analizan los resultados obtenidos en cada uno de los escenarios considerados. Cada escenario analiza el impacto de una característica concreta o un parámetro del algoritmo.

4.1 Parámetros de simulación

Antes de analizar cada escenario por separado, se explican los parámetros que se fijan en la herramienta ns-3 para realizar las simulaciones. Dichos parámetros son los siguientes:

Parámetros generales de la red WiFi:

Tabla 4.1: Datos de la red WiFi utilizada en las simulaciones.

Tasa de transmisión	6 Mbps
Modelo de transmisión	Log-distance path loss model [**12**]
Estándar utilizado	IEEE 802.11-2012

Parámetros de transmisión de los AP y estaciones [13]:

Tabla 4.2: Datos de los AP y STA utilizados en las simulaciones.

Ganancia de transmisión (AP y STA)	1 dB
Ganancia de recepción (AP y STA)	1 dB
Potencia de transmisión (AP y STA)	16.0206 dBm
CCA Detection Threshold	-99 dBm
Energy Detection Threshold	-96 dBm
Duración máxima del paquete en cola	0.5 segundos
Tamaño de las colas de cada categoría de acceso	400 paquetes

Parámetros de los modelos de tráfico empleados:

Tabla 4.3: Datos del tráfico utilizado en las simulaciones.

Tráfico VoIP (uplink y downlink)	
Tasa de transmisión	64 Kbps
Tamaño de paquete	160 bytes
Paquetes generados por segundo	50 paquetes

Tráfico TCP de background (uplink y downlink)	
Tamaño de paquete	1500 bytes

Estos parámetros de los modelos de tráfico se mantienen para todas las simulaciones con el fin de homogeneizar los resultados obtenidos.

Otro parámetro necesario para construir los escenarios es la **distancia máxima de interferencia**, es decir, la distancia a partir de la cual dejan de producirse interferencias entre los distintos AP de la red. Para calcularla, hemos seguido los siguientes pasos:

En primer lugar, se han consultado los detalles del modelo de propagación empleado en la documentación de ns-3 [**12**]. Como se ha mencionado, este modelo es el "log distance", el cual emplea la siguiente expresión para las pérdidas de propagación:

$$L = L_0 + 10\, n\, log_{10}(\frac{d}{d_0})$$

con:

$$n = 3$$

$$L_0 = 46{,}6777\ dB$$

$$d_0 = 1\ m$$

A partir de estos valores y los parámetros de transmisión de los AP y las estaciones de la Tabla 4.2 se puede calcular la distancia máxima de interferencia. La expresión que relaciona las potencias de transmisión y recepción es:

$$TXPower + TXGain - L + RXGain \geq CCATh$$

Con lo que el umbral de las pérdidas de propagación para que haya interferencia es:

$$16{,}0286\ dBm + 1\ dB - L + 1\ dB \ \geq\ -99\ dBm$$
$$L\ \leq 117{,}0286\ dB$$

Si ahora sustituimos en la expresión de las pérdidas:

$$117{,}0286 = 46{,}6777 + 10 * 3 * \log_{10}(d)$$

$$\log_{10}(d) = \frac{117{,}0286 - 46{,}6777}{30} = 2{,}34503$$

$$d = 10^{2.34503} = 221{,}324\ m$$

Por lo tanto, obtenemos una distancia máxima de interferencia de 221.324 metros. Una vez obtenido este valor, se comprobó mediante simulación que era correcto, verificándolo mediante pruebas con diferentes distancias.

Para la realización de las simulaciones se ha escogido un valor fijo de distancia entre AP. Dicho valor es inferior a la distancia máxima de interferencia, pero lo suficientemente grande como para permitir que unos AP interfieran con otros o no dependiendo de dónde sean colocados.

Tabla 4.4: Distancia máxima de interferencia y distancia fija entre AP.

Distancia máxima de interferencia	221,324 metros
Distancia fija entre AP	175 metros

4.2 Análisis de resultados

4.2.1. Resultados con un único AP

En primer lugar, se va a analizar el funcionamiento del algoritmo propuesto en una red que consta de un único AP. Los resultados se van a comparar tanto con el mecanismo EDCA como con una versión adaptada del algoritmo descrito en [**11**] y que ha servido de base para el algoritmo propuesto. La modificación introducida en ese algoritmo ha sido que el ajuste de CW_{min} se realiza en función del retardo de las aplicaciones multimedia y no del throughput de la red. La razón de introducir esta modificación es que en nuestro caso el factor más importante es cumplir con los requisitos de estas aplicaciones y no maximizar la capacidad total de la red. De ahora en adelante, a este algoritmo lo denominaremos algoritmo local.

El escenario considerado se representa en la Figura 4.1. Como se ha comentado en el párrafo anterior, consta de un único AP que envía y recibe flujos de tráfico. Este AP da servicio a 17 estaciones con tráfico VoIP tanto ascendente como descendente, generado de acuerdo a los parámetros de la Tabla 4.3. Además, hay 21 estaciones con tráfico TCP descendente. La duración de la simulación es de 100 segundos.

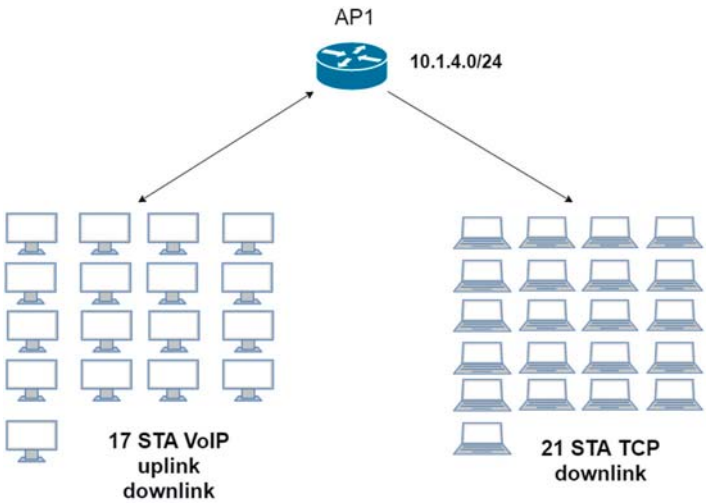


Figura 4.1. Escenario con 1 AP.

En este escenario, se va a monitorizar el retardo de los paquetes de voz (los que ocupan la cola ***AC_VO***) para el mecanismo EDCA, para el algoritmo local y para el algoritmo propuesto. Los resultados aparecen representados en la figura 4.2.

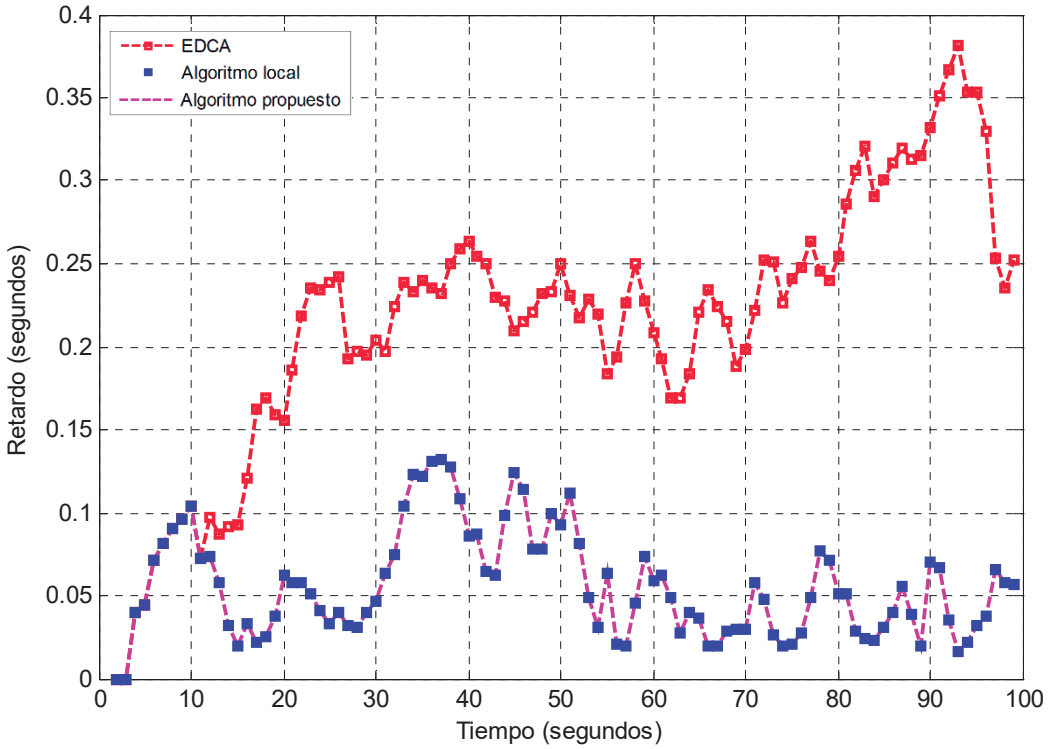


Figura 4.2. Retardo de los paquetes de la cola AC_VO en el escenario con 1 AP.

Como se puede apreciar, en cuanto el retardo medio de los paquetes de la cola ***AC_VO*** supera los 100 milisegundos, el algoritmo propuesto incrementa el valor de CW_{min} para la cola de tráfico de background y disminuye su prioridad. De este modo, se consigue que el retardo se mantenga estable en un intervalo entre 30 y 80 milisegundos. Por el contrario, el mecanismo EDCA no realiza este ajuste y es incapaz de suministrar a las aplicaciones multimedia un retardo inferior a 100 milisegundos, alcanzándose valores que no son admisibles en una conversación de voz. Finalmente, el resultado obtenido con el algoritmo local es el mismo que el obtenido con el algoritmo propuesto.

Esto se debe a que, al considerar un único AP, no aparecen los beneficios asociados a la coordinación de AP que suministra el algoritmo propuesto.

A continuación, en la figura 4.3 se representa el porcentaje de paquetes perdidos correspondientes a las conversaciones de VoIP. En este caso, las pérdidas son similares en todos los algoritmos, ya que la red no llega a saturarse (pese a que el retardo medio es alto cuando se emplea el algoritmo EDCA). Estas pérdidas las mostramos en porcentaje debido a que si empleásemos la escala logarítmica habría valores no representables.

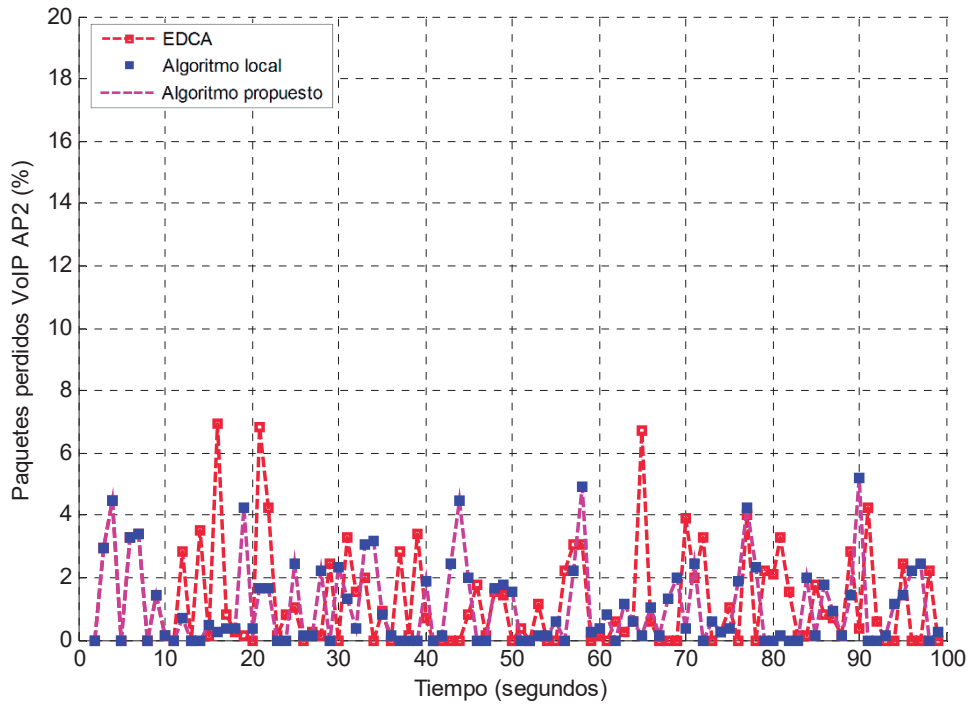


Figura 4.3. Porcentaje de paquetes perdidos por intervalo de actualización de la cola AC_VO en el escenario con 1 AP.

4.2.2 Escenario estático con 3 AP

A continuación, se va a comprobar el funcionamiento del algoritmo propuesto en una red compuesta por tres AP como la de la figura 4.4. La ubicación de los AP es tal que dos de los AP no se interfieren entre sí (AP1 y AP3 en la figura 4.4), pero sí interfieren simultáneamente a un tercer AP (el AP2 colocado en el centro de la figura). Para cumplir con esta condición, la separación entre AP cumple con la distancia de interferencia calculada en el apartado 4.1. Además, los tres AP usan el mismo canal para forzar que se interfieran entre sí. Cada simulación dura 100 segundos y el tráfico ofrecido en cada AP es el siguiente:

AP1:

- **1** estación con tráfico VoIP tanto descendente como ascendente según los parámetros de la tabla 4.3.
- **5** estaciones con tráfico TCP descendente.
- **1** estación con tráfico TCP ascendente.

AP2:

- Un número variable de estaciones (entre 0 y 14) con tráfico VoIP tanto descendente como ascendente. El objetivo es ver el comportamiento del algoritmo al variar este número de estaciones.
- **5** estaciones con tráfico TCP descendente.
- **1** estación con tráfico TCP ascendente.

AP3:

- **1** estación con tráfico VoIP tanto descendente como ascendente según los parámetros de la tabla 4.3.
- **5** estaciones con tráfico TCP descendente.
- **1** estación con tráfico TCP ascendente.

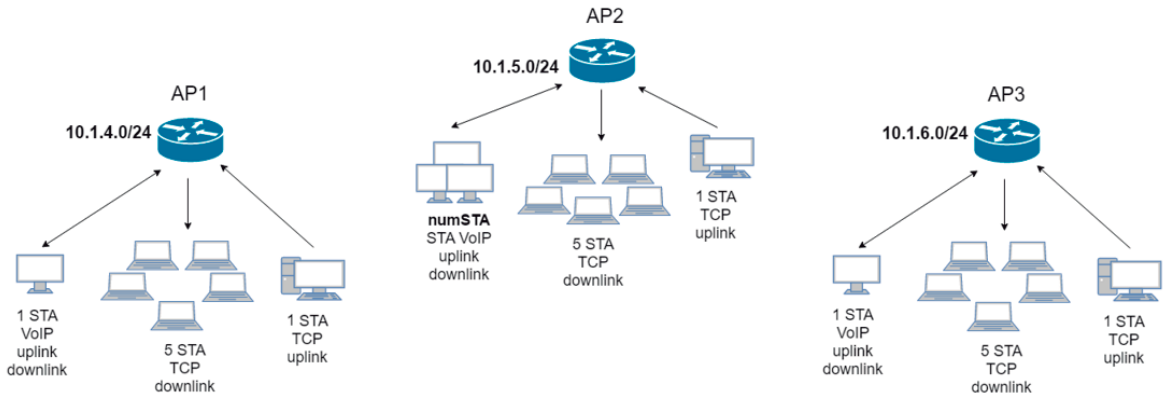


Figura 4.4. Escenario estático con 3 AP

La elección del número de estaciones y el tráfico de cada una de ellas (TCP y VoIP, ascendente y descendente) se ha realizado para intentar simular un escenario heterogéneo con tráfico diverso. Hay que tener en cuenta que el algoritmo propuesto únicamente controla los AP, por lo que todo el tráfico ascendente que generan las estaciones no se puede controlar.

En primer lugar, se representa en la Figura 4.5. el retardo medio de los paquetes en la cola ***AC_VO*** del AP2 (correspondientes a conversaciones de VoIP) en función del número de estaciones con tráfico VoIP en dicho AP. Se representan los resultados en este AP ya que es el más interferido y, por tanto, el que peores resultados va a obtener.

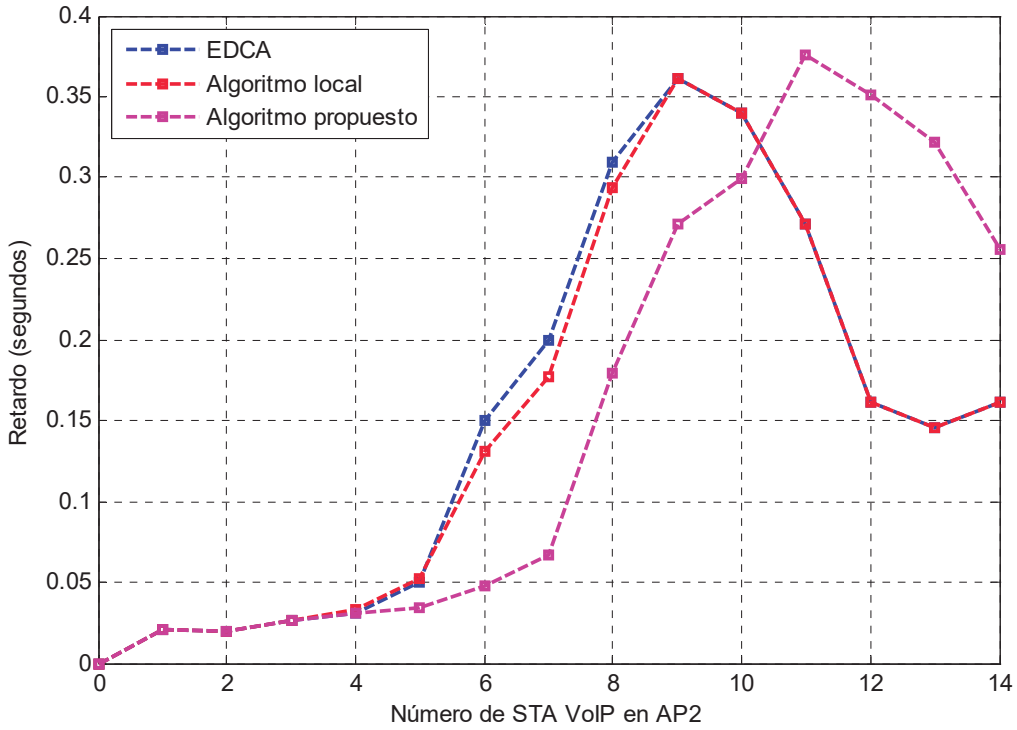


Figura 4.5. Retardo medio de los paquetes de la cola AC_VO del AP2 en el escenario estático con 3 AP.

Como se puede observar, el algoritmo propuesto consigue una mejora apreciable en el retardo medio de los paquetes correspondientes a conversaciones de VoIP. De hecho, si consideramos que el retardo máximo tolerable es de 100 ms, la mejora obtenida por el algoritmo propuesto es de casi un 50% (se pasan de 5 a 7 conversaciones) en comparación con el EDCA o el algoritmo local. Por otro lado, los resultados obtenidos por el algoritmo local son muy similares a los del EDCA. Esto se debe a que este algoritmo únicamente ajusta el valor de CW_{min} para la cola de tráfico de background de cada AP en función del retardo que sufren los paquetes de VoIP del mismo AP, por lo que el retardo de los paquetes de VoIP del AP2 no se tienen en cuenta cuando se ajusta CW_{min} en los otros dos AP.

Por el contrario, el algoritmo propuesto permite modificar los valores de CW_{min} de la cola ***AC_BK*** de todos los AP que interfieren entre sí. De este modo, al detectar que el tráfico de VoIP del AP2 tiene un retardo excesivo, los valores de CW_{min} de las colas del AP1 y del AP3 también se modifican. Esto permite bajar la prioridad de su tráfico TCP, por lo que el tráfico de VoIP del AP2 puede ganar el acceso al medio con más facilidad.

Finalmente, se presenta para el mismo escenario el porcentaje de paquetes perdidos de conversaciones de VoIP en el AP2. Igual que en el caso anterior, el algoritmo propuesto mejora con claridad los resultados del EDCA o del algoritmo local. Si fijamos por ejemplo que el porcentaje de paquetes perdidos para una conversación de VoIP no puede superar el 10%, la mejora es del 33% (de 6 a 8 conversaciones).

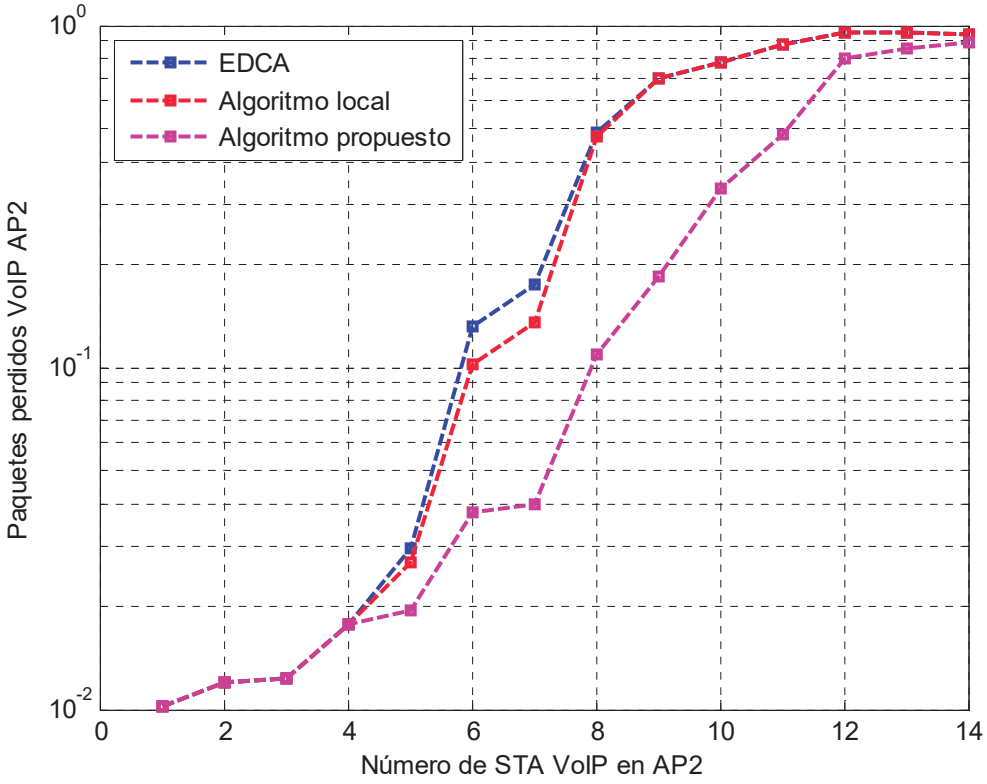


Figura 4.6. Paquetes perdidos de la cola AC_VO del AP2 en el escenario estático con 3 AP en escala logarítmica.

4.2.3 Escenario dinámico con 3 AP

En este escenario se va a mostrar el comportamiento dinámico del algoritmo. Esto es, cómo reacciona frente a cambios en el número de flujos que hay en la red. El objetivo es comprobar el procedimiento por el cual el algoritmo incrementa rápidamente la prioridad del tráfico multimedia una vez que el retardo que sufre supera el valor del parámetro $\boldsymbol{D_{max}}$ y como la desciende cuando ese retardo está por debajo del parámetro $\boldsymbol{D_{min}}$. Esta reducción permite aumentar la cantidad de tráfico elástico transmitido en la red, mejorando su tasa agregada (y siempre manteniendo los requisitos del tráfico multimedia).

El escenario considerado se muestra en la figura 4.7. La ubicación de los AP es idéntica a la del anterior escenario, por lo que los AP de los extremos no se interfieren entre sí pero sí lo hacen al AP central.

La simulación dura 100 segundos y el tráfico ofrecido en cada AP es el siguiente:

AP1:

- **8** estaciones con tráfico VoIP ascendente y descendente
- **5** estaciones con tráfico TCP descendente

AP2:

- **8** estaciones con tráfico VoIP ascendente y descendente. Estas estaciones dejan de transmitir / recibir a los 55 segundos de simulación.
- **5** estaciones con tráfico TCP descendente

AP3:

- **8** estaciones con tráfico VoIP ascendente y descendente
- **5** estaciones con tráfico TCP descendente

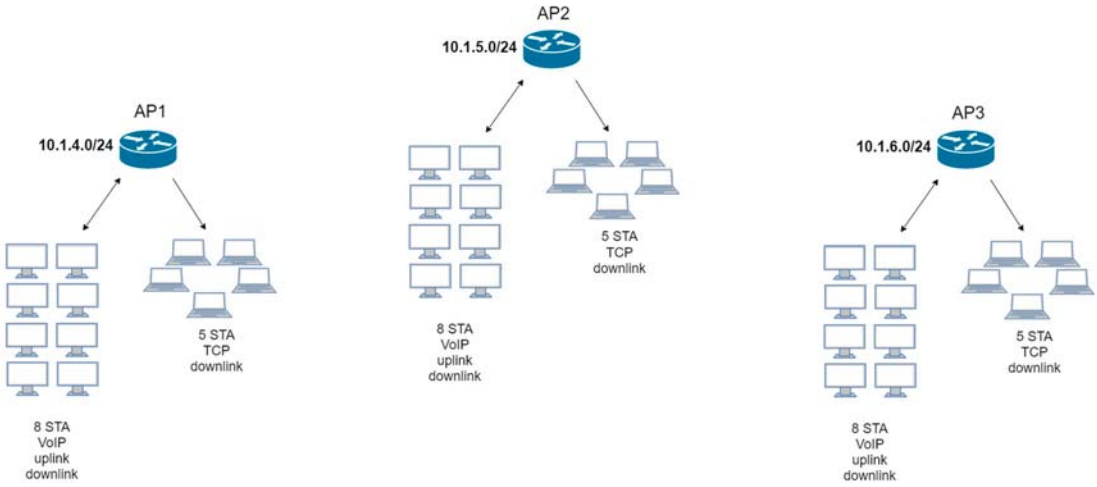


Figura 4.7. Escenario con tráfico dinámico.

Por tanto, en el instante T = 55 s se va a producir una disminución del tráfico ofrecido a la red ya que van a desaparecer todos los flujos de VoIP del segundo AP. A continuación, se muestra en la figura 4.8 la evolución temporal de la ventana de contienda del tráfico elástico en el segundo AP.

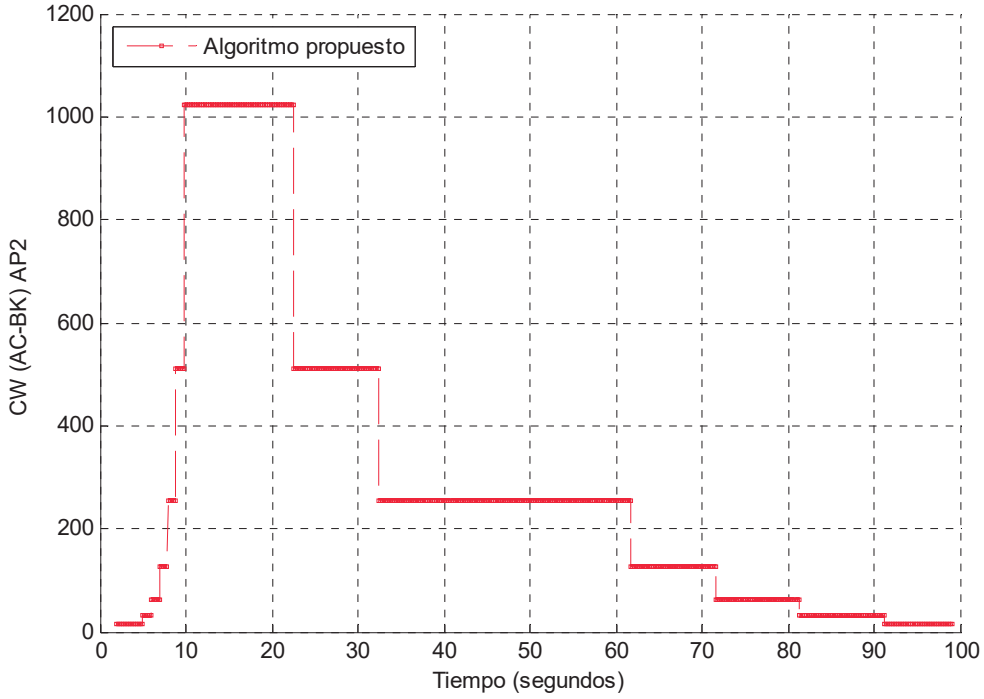


Figura 4.8. CW de la cola AC_BK del AP2 en el escenario con tráfico dinámico.

La evolución de la ventana de contienda se puede dividir en tres fases:

- En primer lugar, se produce un aumento rápido de la ventana de contienda debido a que el algoritmo duplica el valor de CW_{min} cada segundo ($\boldsymbol{T}_{update}^{(1)}$) mientras el retardo medio ($\boldsymbol{D_{max}}$) de los paquetes de la cola $\boldsymbol{AC\_VO}$ (los paquetes VoIP) es superior a 100 milisegundos. Esto ocurre durante los primeros segundos de la simulación.

- Posteriormente y una vez que alcanza su valor máximo (1023), CW_{min} empieza a disminuir ya que el retardo del tráfico VoIP es inferior a $\boldsymbol{D_{min}}$ (20 milisegundos). Como el periodo para realizar esta comprobación ($\boldsymbol{T}_{update}^{(2)}$) es superior a $\boldsymbol{T}_{update}^{(1)}$ (10 segundos frente a 1 segundo), el ajuste a la baja de CW_{min} es mucho más lento que cuando crece al comienzo de la simulación. Esta disminución de CW_{min} se realiza en dos ocasiones, estabilizándose en un tamaño de 255 hasta que desaparece el tráfico VoIP del segundo AP.

- Finalmente, al apagar todos los flujos de VoIP en el AP2, el algoritmo detecta la ausencia de tráfico multimedia en dicho AP y disminuye nuevamente el valor de CW_{min} hasta llegar a su valor mínimo ($CW = 15$) en los instantes finales de la simulación.

En la figura 4.9 se muestra el retardo medio que sufren los paquetes de VoIP (paquetes de la cola $\boldsymbol{AC\_VO}$) en el AP2 a lo largo de la simulación. Como se puede observar, el retardo sube al comienzo de la simulación hasta que el algoritmo ajusta el parámetro CW_{min} del tráfico elástico a un valor lo suficientemente alto como para que el retardo del tráfico multimedia sea aceptable. De hecho, este valor llega a ser inferior a 20 milisegundos, lo que hace que CW_{min} disminuya desde 1023 hasta 255. Esta disminución no afecta al retardo, que sigue estando por debajo de 20 milisegundos. Finalmente, cuando desaparece el tráfico de voz, el retardo cae a cero al no haber tráfico que mandar.

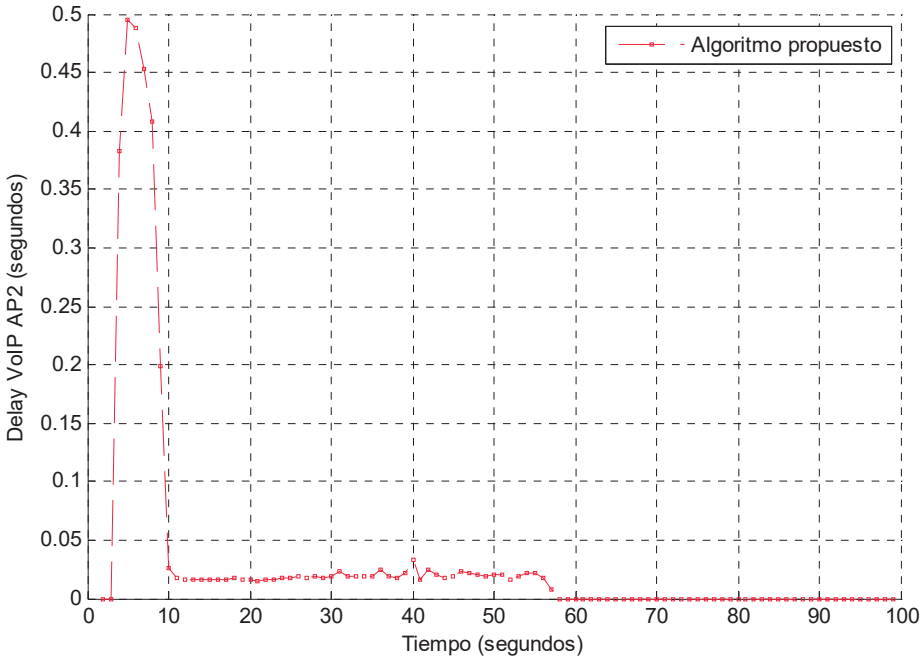


Figura 4.9. Retardo medio de los paquetes de la cola AC_VO del AP2 en el escenario con tráfico dinámico.

Finalmente, en la Figura 4.10 se muestra el throughput agregado de los flujos TCP del AP1 (los del AP3 son similares) a lo largo del tiempo. Como se puede ver, el throughput tiene un pico al comienzo de la simulación mientras el algoritmo ajusta el valor del parámetro CW_{min}. Una vez que el algoritmo garantiza la prioridad necesaria al tráfico de VoIP, el tráfico TCP se ajusta para ocupar el ancho de banda sobrante. Por otro lado, cuando deja de haber tráfico de VoIP en el AP2, el valor de CW_{min} de la cola $\boldsymbol{AC\_BK}$ va disminuyendo cada $\boldsymbol{T}_{update}^{(2)}$ segundos, lo que se aprecia también en la gráfica ya que las mejoras de throughput corresponden aproximadamente a los cambios en el valor de CW_{min}.

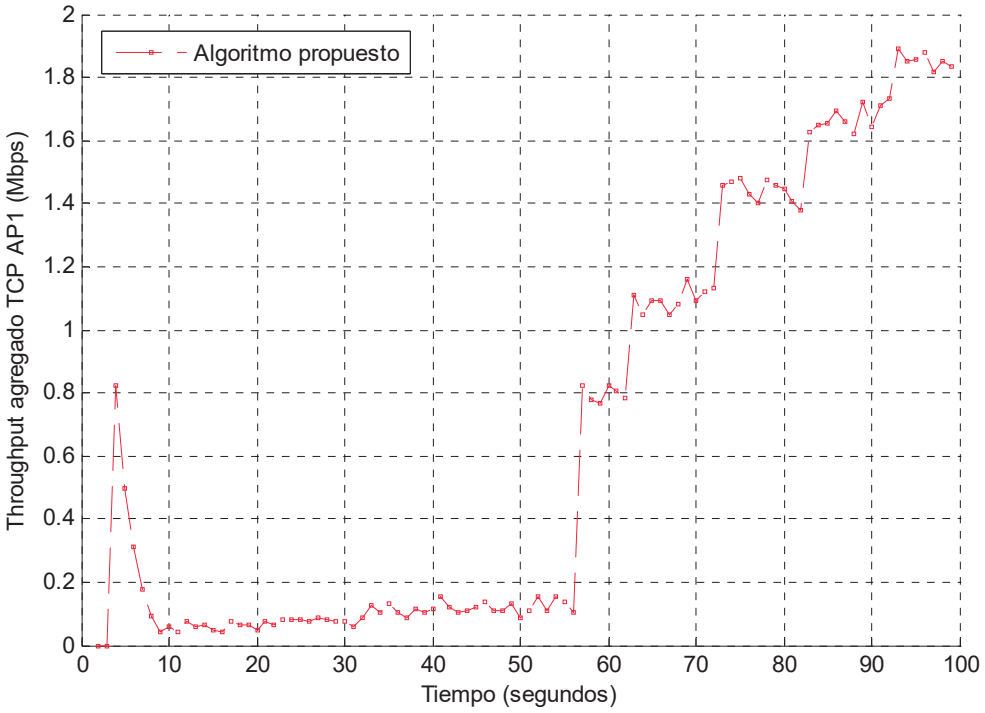


Figura 4.10. Throughput agregado del tráfico TCP del AP1 en el escenario con tráfico dinámico.

4.2.4. Impacto de los parámetros del algoritmo en el comportamiento

A continuación, se va a analizar el impacto de los parámetros característicos del algoritmo: $\boldsymbol{T_{update}^{(1)}}$, $\boldsymbol{T_{update}^{(2)}}$ *y* $\boldsymbol{D_{min}}$ en el sistema.

4.2.4.1 Impacto de $\boldsymbol{T_{update}^{(1)}}$

En primer lugar, se va a analizar el impacto del parámetro $\boldsymbol{T_{update}^{(1)}}$ en el comportamiento del algoritmo. En concreto, el objetivo es analizar cómo influye $\boldsymbol{T_{update}^{(1)}}$ en el retardo del tráfico VoIP y en la estabilidad del parámetro CW_{min} de la cola $\boldsymbol{AC\_BK}$.

El escenario considerado para realizar el análisis se muestra en la figura 4.11. La ubicación de los AP es idéntica a la del escenario del apartado 4.2.2, por lo que los AP de los extremos no se interfieren entre sí pero sí lo hacen al AP central. La simulación dura 100 segundos y el tráfico ofrecido en cada AP es el siguiente:

AP1:

- **8** estaciones con tráfico VoIP ascendente y descendente
- **5** estaciones con tráfico TCP descendente

AP2:

- **8** estaciones con tráfico VoIP ascendente y descendente.
- **5** estaciones con tráfico TCP descendente

AP3:

- **8** estaciones con tráfico VoIP ascendente y descendente
- **5** estaciones con tráfico TCP descendente

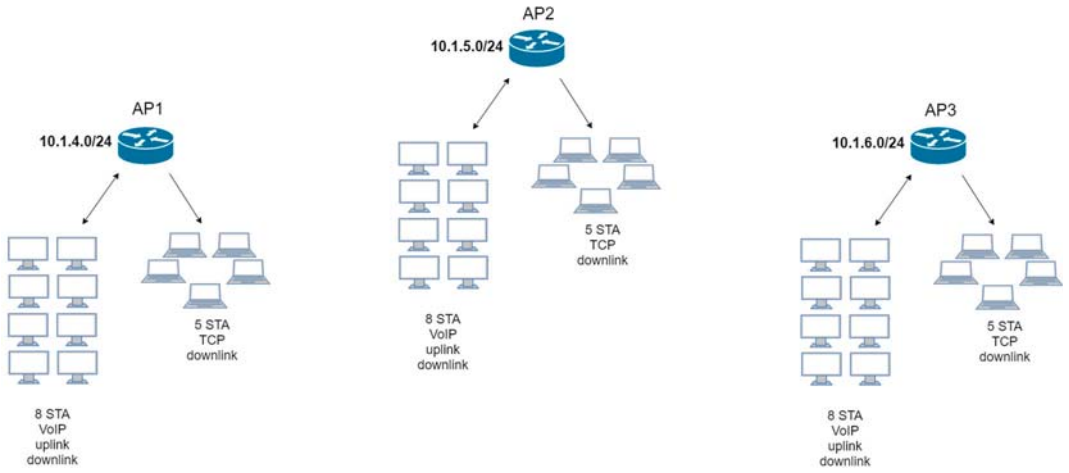
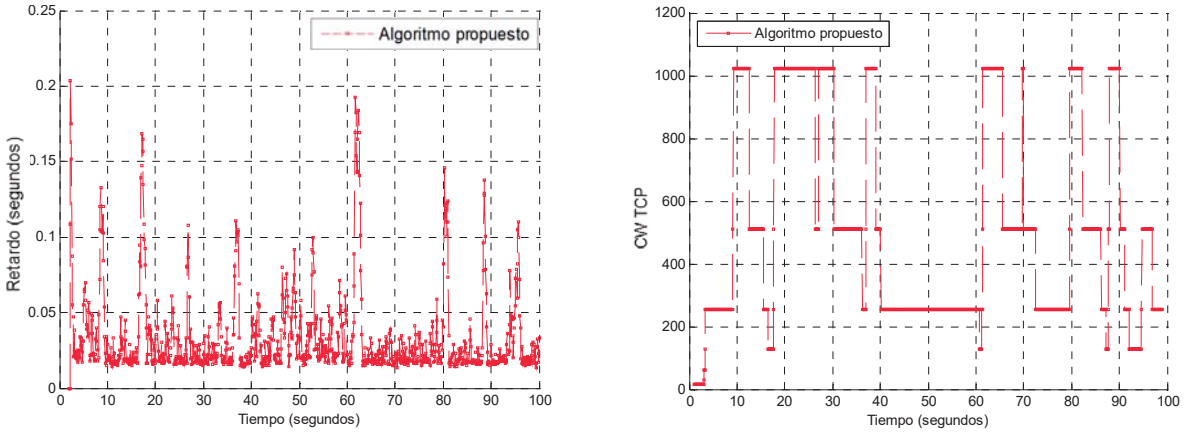


Figura 4.11. Escenario utilizado para el análisis de $\boldsymbol{T}_{update}^{(1)}$.

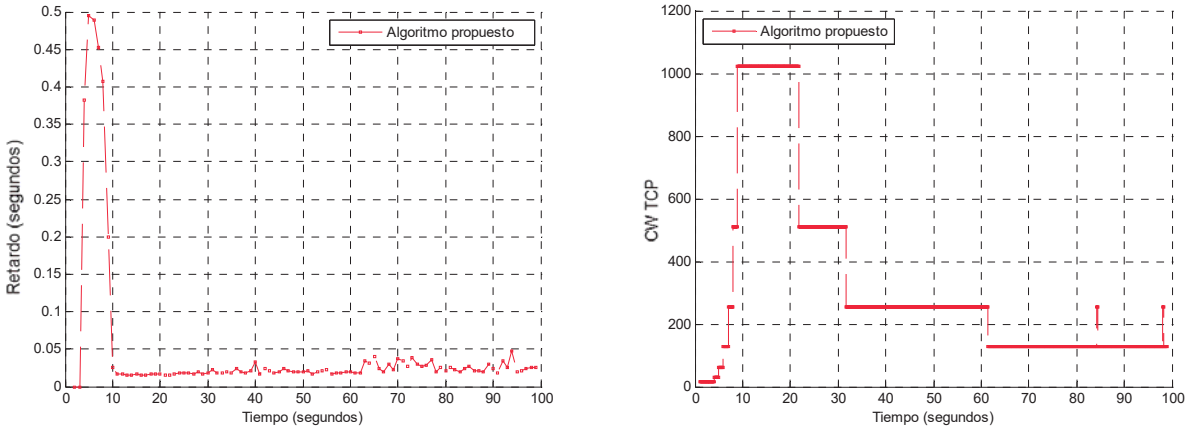
Las figuras 4.12, 4.13 y 4.14 muestran los resultados obtenidos (retardo de los paquetes de VoIP en la izquierda y valor de CW_{min} en la derecha) para valores del parámetro $\boldsymbol{T}_{update}^{(1)}$ iguales a 0.1, 1 y 10 segundos. En todos los casos, el valor del parámetro $\boldsymbol{T}_{update}^{(2)}$ se ha dejado fijo a 10 veces el valor de

$T_{update}^{(1)}$, siendo los valores de D_{max} y D_{min} iguales a 100 y a 20 ms respectivamente.



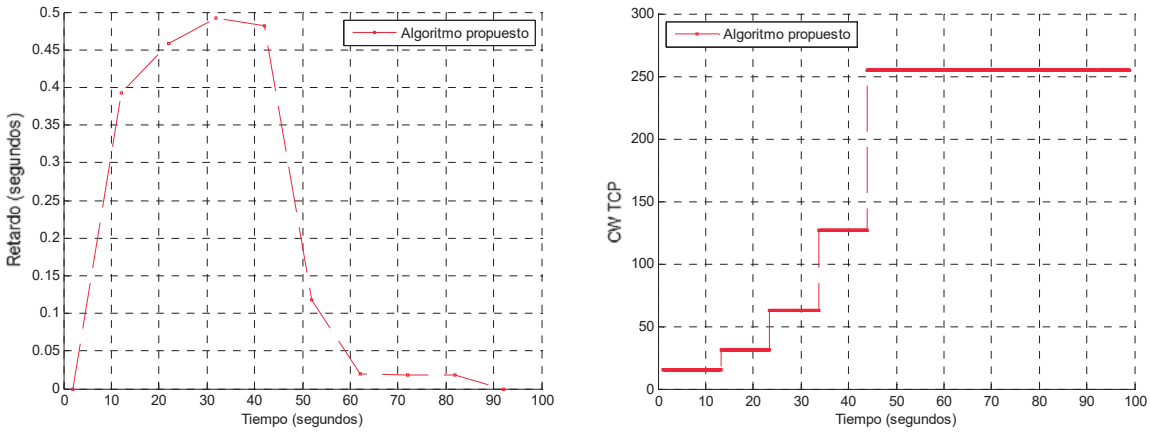
a) Retardo medio VoIP *b) CW para TCP*

Figura 4.12. Retardo de los paquetes de VoIP (a) y valor de CW_{min} para TCP (b) en función del tiempo con ($T_{update}^{(1)} = 0.1$ s) en el AP2



b) Retardo medio VoIP *b) CW para TCP*

Figura 4.13. Retardo de los paquetes de VoIP (a) y valor de CW_{min} para TCP (b) en función del tiempo con ($T_{update}^{(1)} = 1$ s) en el AP2



a) Retardo medio VoIP *b) CW para TCP*

Figura 4.14. Retardo de los paquetes de VoIP (a) y valor de CW_{min} para TCP (b) en función del tiempo con ($T_{update}^{(1)}$= 10 s) en el AP2

Como se puede comprobar en las figuras, aparece un compromiso claro entre la estabilidad del algoritmo (en términos tanto de la variación de CW_{min} como de la variabilidad del retardo) por un lado, y la velocidad de adaptación cuando el retardo del tráfico de VoIP aumenta por encima de D_{max} por otro. Como se puede apreciar en Figura 4.15, utilizar un valor de $T_{update}^{(1)}$ elevado implica que el tiempo necesario para conseguir que el retardo de la voz sea adecuado es muy elevado (en este caso por encima de 50 segundos), lo cual es inasumible para una llamada de voz.

En cuanto al caso $T_{update}^{(1)} = 0.1$ segundos, se observa en la Figura 4.14 que dicho valor provoca una elevada variabilidad en el retardo y en CW_{min}. Aunque durante buena parte de la simulación el retardo se mantiene en límites aceptables, se aprecian picos de retardo que podrían afectar a la llamada de voz.

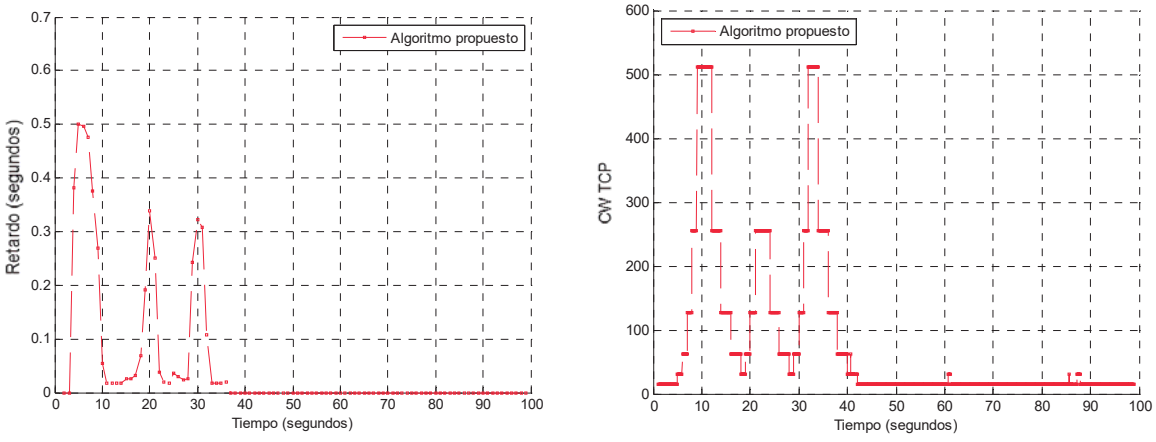
Por el contrario, para el caso de $T_{update}^{(1)} = 1$ segundo, el compromiso entre estabilidad y velocidad de adaptación es adecuado. Según se aprecia en la gráfica, en menos de 10 segundos se consigue priorizar el tráfico VoIP lo suficiente como para mantener el retardo por debajo de D_{max}. Además, en cuanto lo hace se mantiene estable durante el resto de simulación.

4.2.4.2 Impacto de $T_{update}^{(2)}$

En segundo lugar, se va a analizar el impacto del parámetro $T_{update}^{(2)}$ en el retardo del tráfico VoIP y en la estabilidad del parámetro CW_{min} de la cola AC_BK, así como en el throughput de un AP vecino.

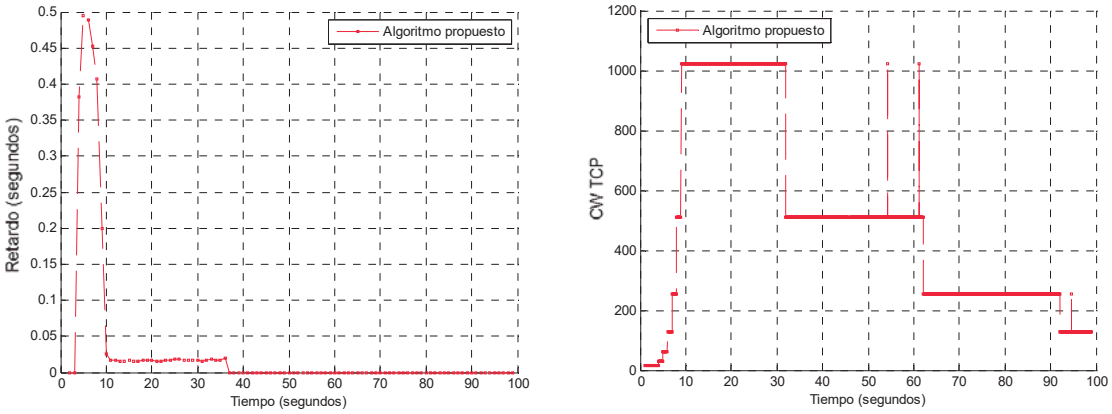
El escenario utilizado para este análisis es idéntico al empleado para analizar $T_{update}^{(1)}$, con la única diferencia de que se apagan los flujos de tráfico UDP de AP2 en t = 35 segundos. Esto se hace para ver mejor cómo se comporta la adaptación del algoritmo al apagar dichos flujos.

Las figuras 4.15, 4.16 y 4.17 muestran los resultados obtenidos (retardo de los paquetes de VoIP en la izquierda y valor de CW_{min} en la derecha) para valores del parámetro $T_{update}^{(2)}$ iguales a 2, 10 y 30 segundos. En todos los casos, el valor del parámetro $T_{update}^{(1)}$ se ha dejado fijo a 1 segundo, siendo los valores de D_{max} y D_{min} iguales a 100 y a 20 ms respectivamente.



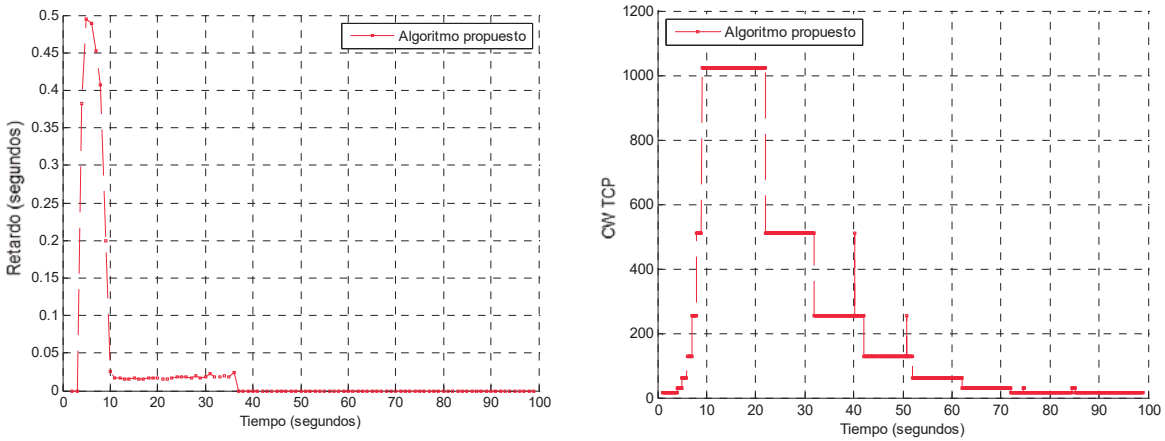
a) Retardo medio VoIP *b) CW para TCP*

Figura 4.15. Retardo de los paquetes de VoIP (a) y valor de CW_{min} para TCP (b) en función del tiempo con ($T_{update}^{(2)}$ = 2 s) en el AP2



a) Retardo medio VoIP *b) CW para TCP*

Figura 4.16. Retardo de los paquetes de VoIP (a) y valor de CW_{min} para TCP (b) en función del tiempo con ($T^{(2)}_{update}$= 10 s) en el AP2



a) Retardo medio VoIP *b) CW para TCP*

Figura 4.17. Retardo de los paquetes de VoIP (a) y valor de CW_{min} para TCP (b) en función del tiempo con ($T^{(2)}_{update}$= 30 s) en el AP2

Además de estos datos, vamos a mostrar en las figuras 4.18, 4.19 y 4.20 el throughput del tráfico TCP en un AP vecino (en este caso, AP1) para ver cómo afecta la adaptación al principal beneficiado por la misma, el tráfico TCP.

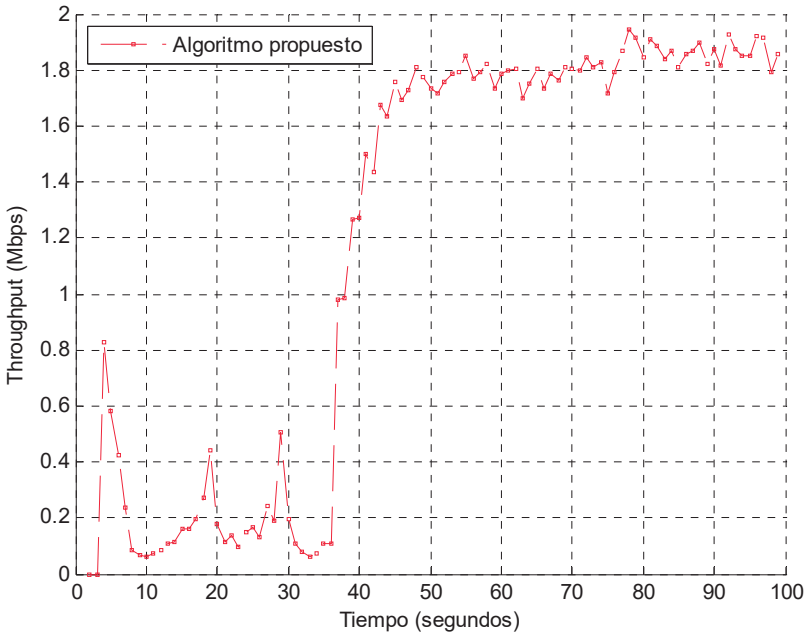


Figura 4.18. Throughput agregado de los paquetes TCP en el AP1 en función del tiempo con ($\boldsymbol{T}_{update}^{(2)}$ = 2 s)

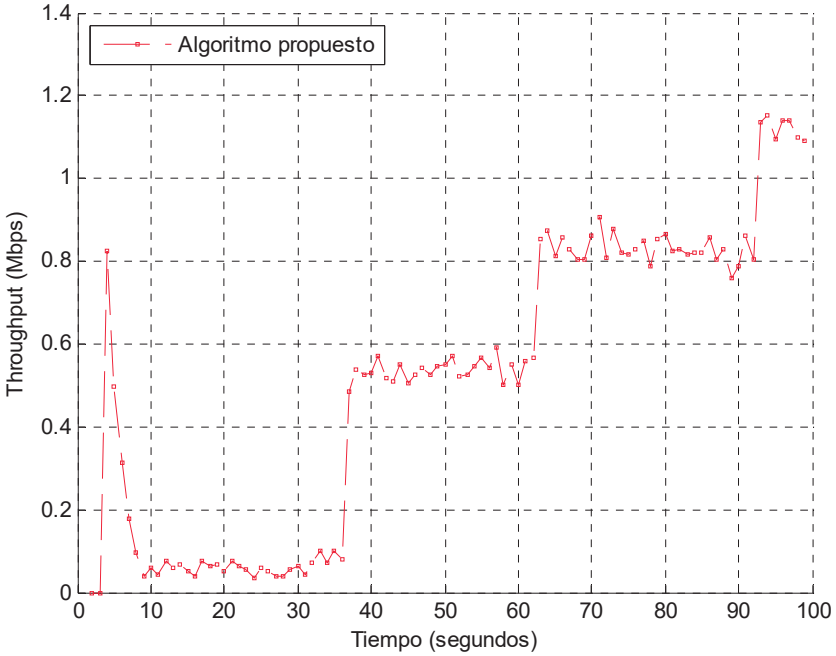


Figura 4.19. Throughput agregado de los paquetes TCP en el AP1 en función del tiempo con ($\boldsymbol{T}_{update}^{(2)}$=10 s)

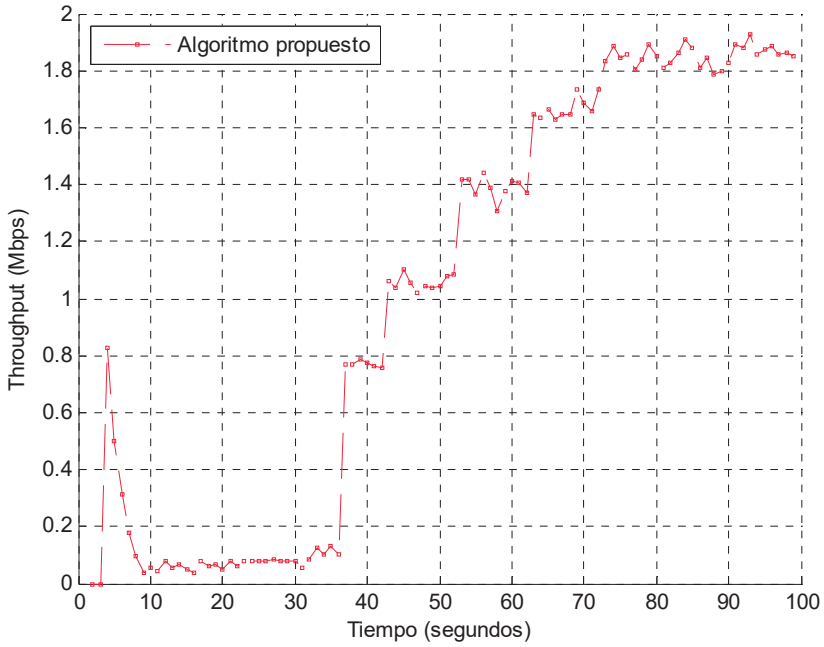


Figura 4.20. Throughput agregado de los paquetes TCP en el AP1 en función del tiempo con ($T_{update}^{(2)} = 30$ s)

Como podemos observar en la figura 4.15, cuando $T_{update}^{(2)} = 2$ *s* se producen demasiadas variaciones en el CW_{min} de la cola ***AC_VO***, lo que provoca la presencia de picos de retardo cuando el parámetro CW_{min} disminuye demasiado. Aunque cuando desaparece el tráfico de VoIP el aumento del throughput del tráfico TCP del AP vecino es el más rápido (figura 4.18), este hecho no compensa el mal comportamiento en el retardo del tráfico de VoIP.

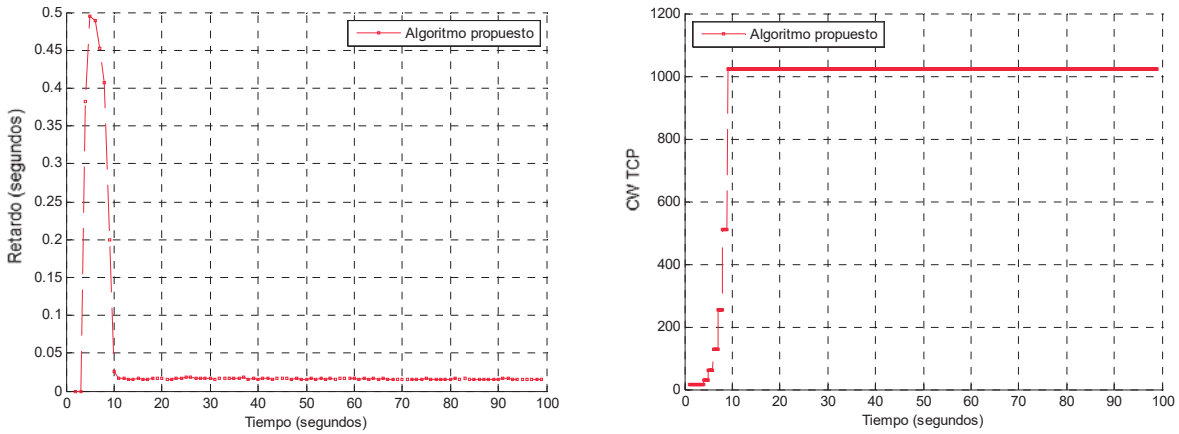
En cuanto al caso de $T_{update}^{(2)} = 30$ *s*, podemos apreciar que, si bien el retardo de la VoIP siempre cumple las restricciones, la recuperación del tráfico TCP es muy lenta, lo que afecta de forma evidente al throughput del tráfico TCP del AP1. La elección por tanto de un valor de 30 segundos es demasiado conservadora.

Por el contrario, para el caso de $T_{update}^{(2)} = 10$ segundos, el compromiso entre estabilidad y velocidad de adaptación es adecuado. Como se aprecia en la Figura 4.19, el tiempo en que el tráfico TCP vuelve a su máximo valor de throughput no es excesivo, y la estabilidad del retardo es evidente (Figura 4.16).

4.2.4.3 Impacto de D_{min}

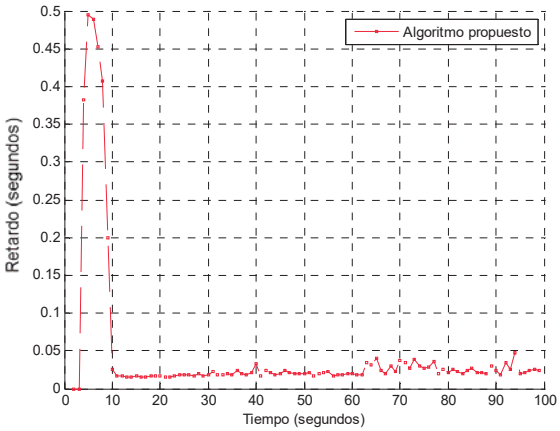
Finalmente, se va a analizar el impacto del parámetro D_{min} en el sistema. Igual que en los casos anteriores, empleamos el mismo escenario que en el caso del análisis de $T^{(1)}_{update}$ (Figura 4.11).

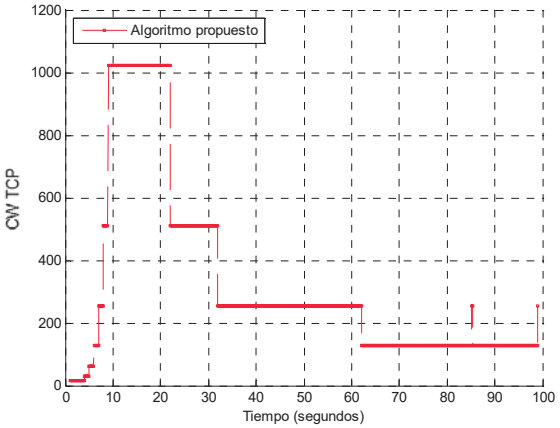
Las figuras 4.21, 4.22 y 4.23 muestran los resultados obtenidos (retardo de los paquetes de VoIP en la izquierda y valor de CW_{min} en la derecha) para valores del parámetro D_{min} iguales a 5, 20 y 90 milisegundos. En todos los casos, el valor del parámetro $T^{(1)}_{update}$ se ha dejado fijo a 1 segundo, siendo el valor de $T^{(2)}_{update}$ igual a 10 segundos y el valor de D_{max} igual a 100 ms.



a) Retardo medio VoIP *b) CW para TCP*

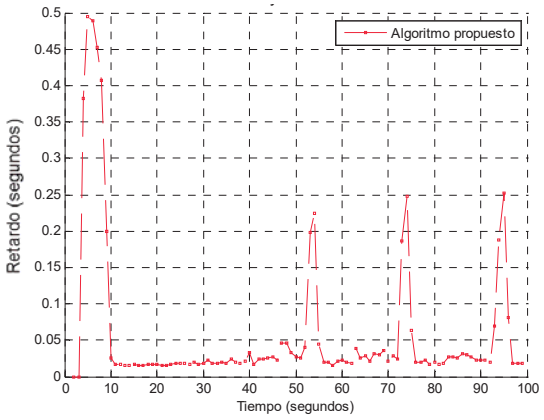
Figura 4.21. Retardo de los paquetes de VoIP (a) y valor de CW_{min} para TCP (b) en función del tiempo con (D_{min} = 5 ms) en el AP2

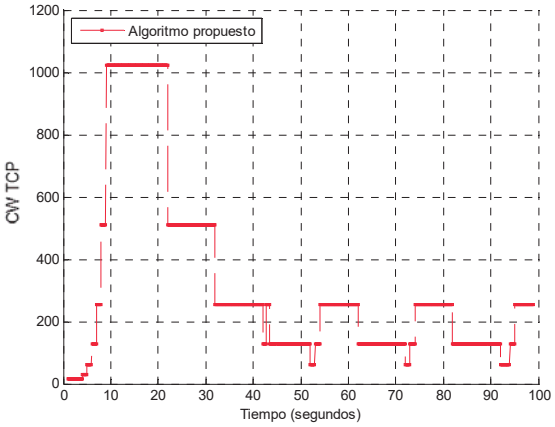




a) Retardo medio VoIP *b) CW para TCP*

Figura 4.22. Retardo de los paquetes de VoIP (a) y valor de $\boldsymbol{CW_{min}}$ *(b) para TCP en función del tiempo con (*$\boldsymbol{D_{min}}$ *= 20 ms) en el AP2*





a) Retardo medio VoIP *b) CW para TCP*

Figura 4.23. Retardo de los paquetes de VoIP (a) y valor de $\boldsymbol{CW_{min}}$ *(b) para TCP en función del tiempo con (*$\boldsymbol{D_{min}}$ *= 90 ms) en el AP2*

Además de estos resultados, vamos a mostrar en las figuras 4.24, 4.25 y 4.26 el throughput del tráfico TCP en AP1 para ver cómo afecta la adaptación al tráfico TCP, de forma análoga a como se ha realizado al analizar el impacto de $\boldsymbol{T_{update}^{(2)}}$.

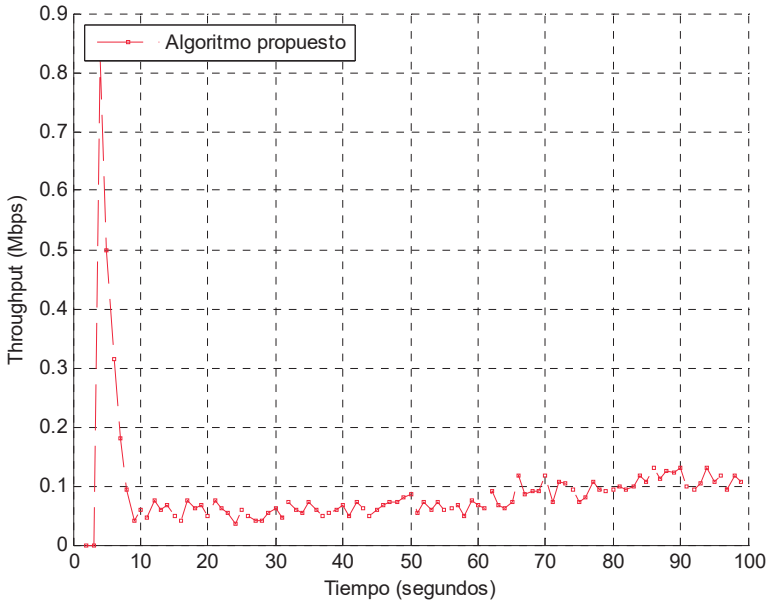


Figura 4.24. Throughput agregado de los paquetes TCP en el AP1 en función del tiempo con ($\boldsymbol{D_{min}}$ = 5 ms)

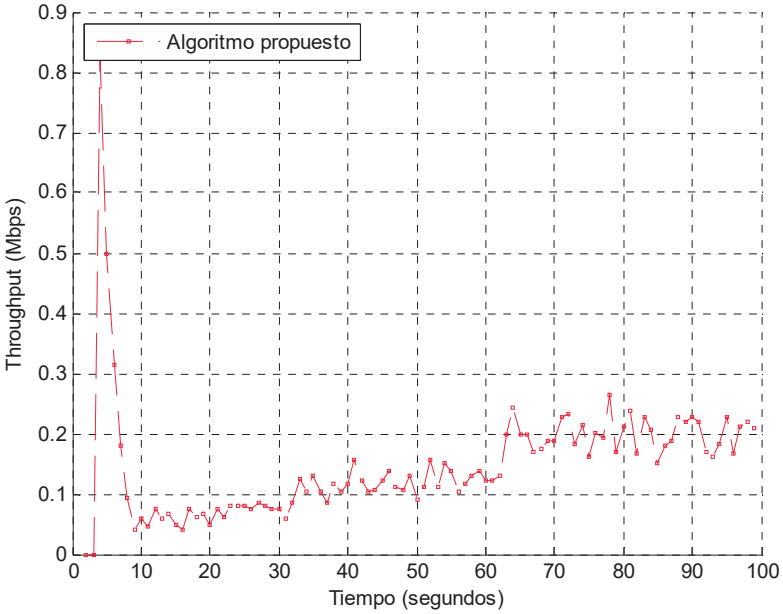


Figura 4.25. Throughput agregado de los paquetes TCP en el AP1 en función del tiempo con ($\boldsymbol{D_{min}}$ = 20 ms)

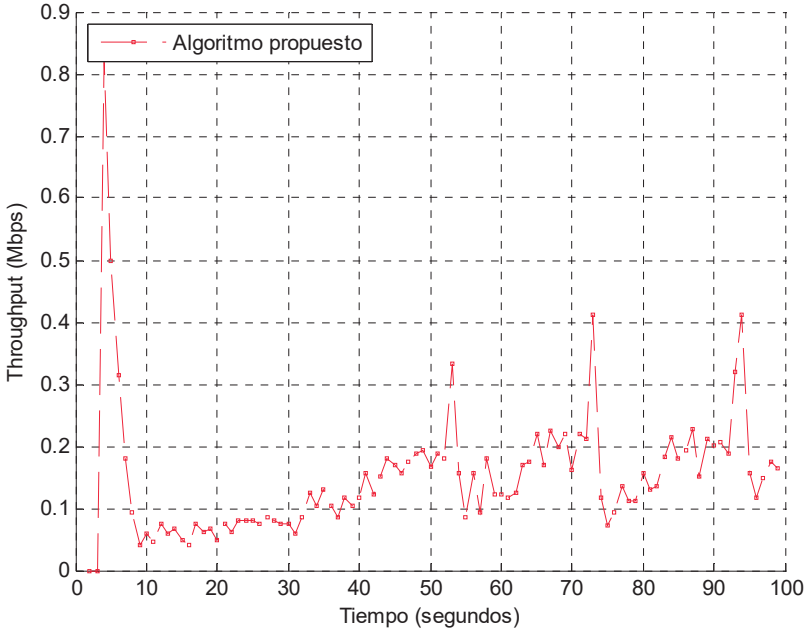


Figura 4.26. Throughput agregado de los paquetes TCP en el AP1 en función del tiempo con ($\boldsymbol{D_{min}}$ = 90 ms)

Como podemos observar en las figuras obtenidas, con un $\boldsymbol{D_{min}} = 5\ ms$ nunca se llega a ejecutar el mecanismo de adaptación, por lo que el throughput del tráfico TCP de los AP vecinos no mejora conforme pasa el tiempo de una forma visible.

En cuanto al caso de $\boldsymbol{D_{min}} = 90\ ms$, podemos apreciar que existe una gran inestabilidad en el sistema, ya que al estar el valor tan cercano a $\boldsymbol{D_{max}}$ se producen variaciones continuas en el valor de CW_{min} que provocan picos en el retardo del tráfico VoIP inasumibles para el buen funcionamiento del sistema.

Por el contrario, para el caso de $\boldsymbol{D_{min}} = 20\ ms$, el compromiso entre estabilidad y velocidad de adaptación es adecuado. Como se aprecia en la Figura 4.25, se produce una recuperación mayor del tráfico TCP que en el caso de $\boldsymbol{D_{min}} = 5\ ms$ y el retardo del tráfico VoIP no se ve afectado en exceso.

4.2.5 Escenario con múltiples AP

En este apartado vamos a comprobar el funcionamiento de nuestro algoritmo ante un escenario más complejo y variado. Consiste en un escenario hexagonal donde existen 19 AP, distribuidos de la forma siguiente:

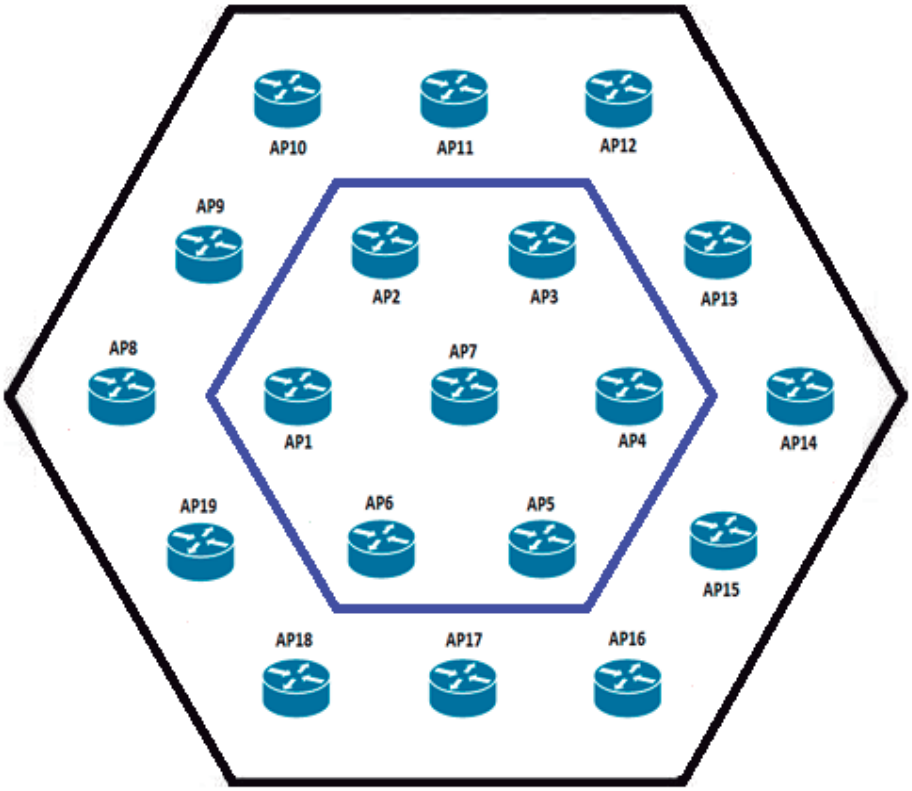


Figura 4.27. Escenario con múltiples AP.

Con esta topología se consigue que los siete AP centrales (numerados del 1 al 7, y marcados en el dibujo dentro del anillo azul) sean interferidos por el mismo número de AP (seis), mientras que los AP del anillo exterior (los que se encuentran entre el anillo azul y el anillo negro) no, ya que su número varía entre tres y cuatro. Es por esto que tomamos como datos útiles para los resultados los del anillo central.

La elección de esta topología se debe a que se utiliza una similar en los grupos de trabajo del estándar 802.11 **[14]**, por lo que es un escenario generalista adecuado para evaluar la solución propuesta.

En la topología anterior asumimos que hay un número variable de estaciones (de 10 a 100) con tráfico VoIP y repartidas entre todos los AP de forma aleatoria (aunque con unos valores máximo y mínimo por AP). En concreto, se van a utilizar los siguientes casos:

Tabla 4.5 Simulaciones del escenario con múltiples AP.

Número total de STA VoIP	STA VoIP por AP	Número total de STA VoIP	STA VoIP por AP
10	[0 - 1]	60	[2 - 4]
20	[0 - 2]	70	[3 - 4]
30	[1 - 2]	80	[3 - 5]
40	[1 - 3]	90	[4 - 5]
50	[2 - 3]	100	[4 - 6]

Además, cada AP cuenta con 5 estaciones con tráfico TCP descendente, y se reparten 7 estaciones con tráfico TCP ascendente al azar entre los 19 AP.

Los resultados obtenidos en la simulación se muestran en las figuras 4.28-4.30. Como se puede observar y tal y como ha ocurrido en los escenarios anteriores, el algoritmo propuesto mejora enormemente las prestaciones del EDCA y del algoritmo local.

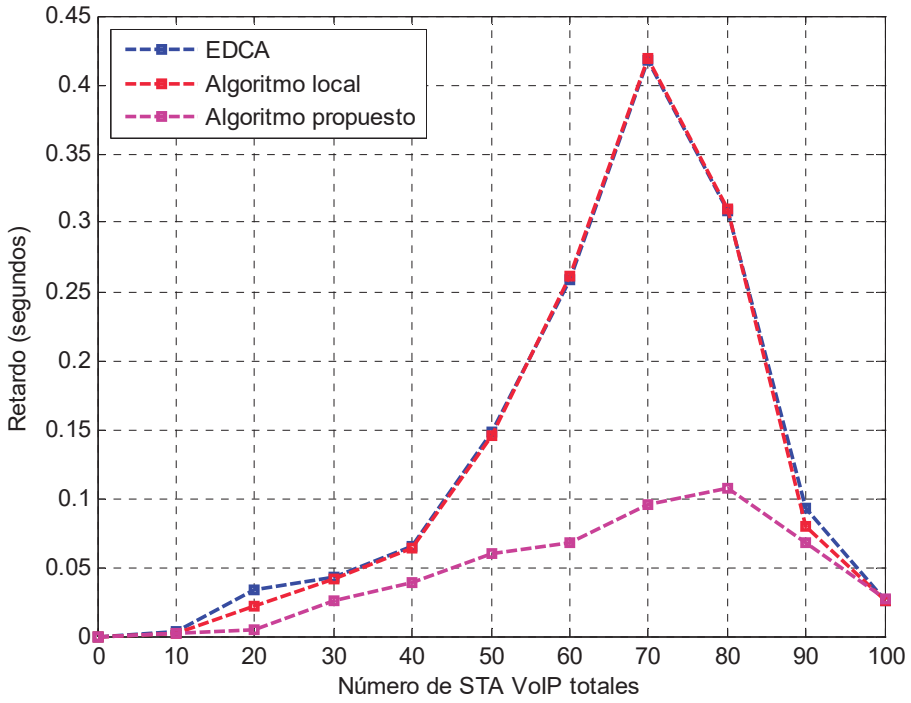


Figura 4.28. Retardo medio de los paquetes de la cola AC_VO del anillo central (AP1-7) en el escenario con múltiples AP.

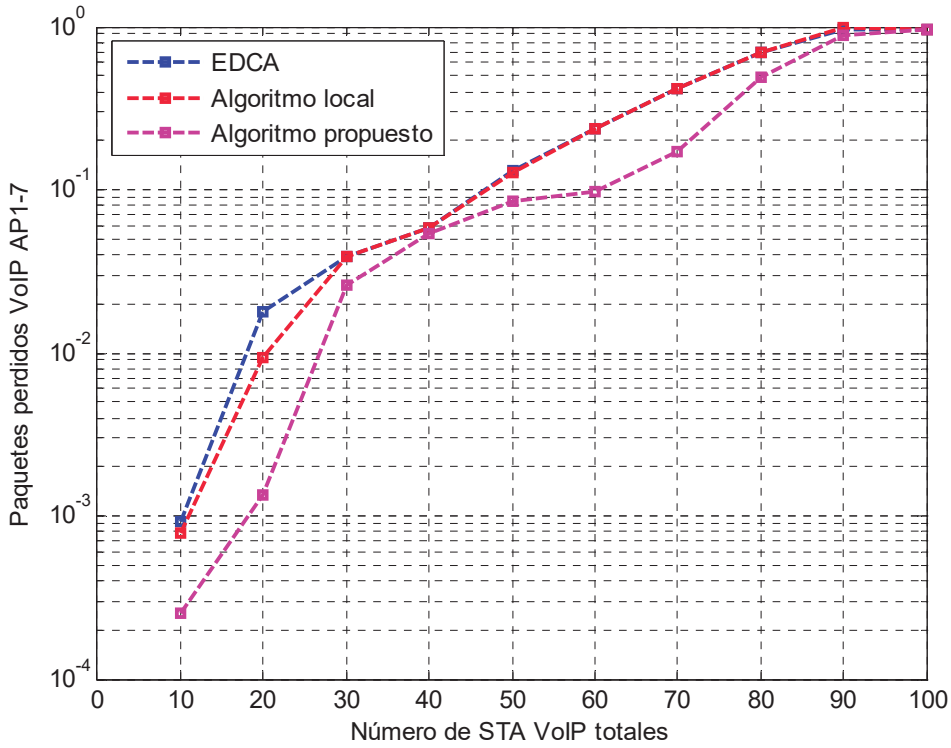


Figura 4.29 Paquetes perdidos de la cola AC_VO en el anillo central (AP1-7) en el escenario con múltiples AP en escala logarítmica.

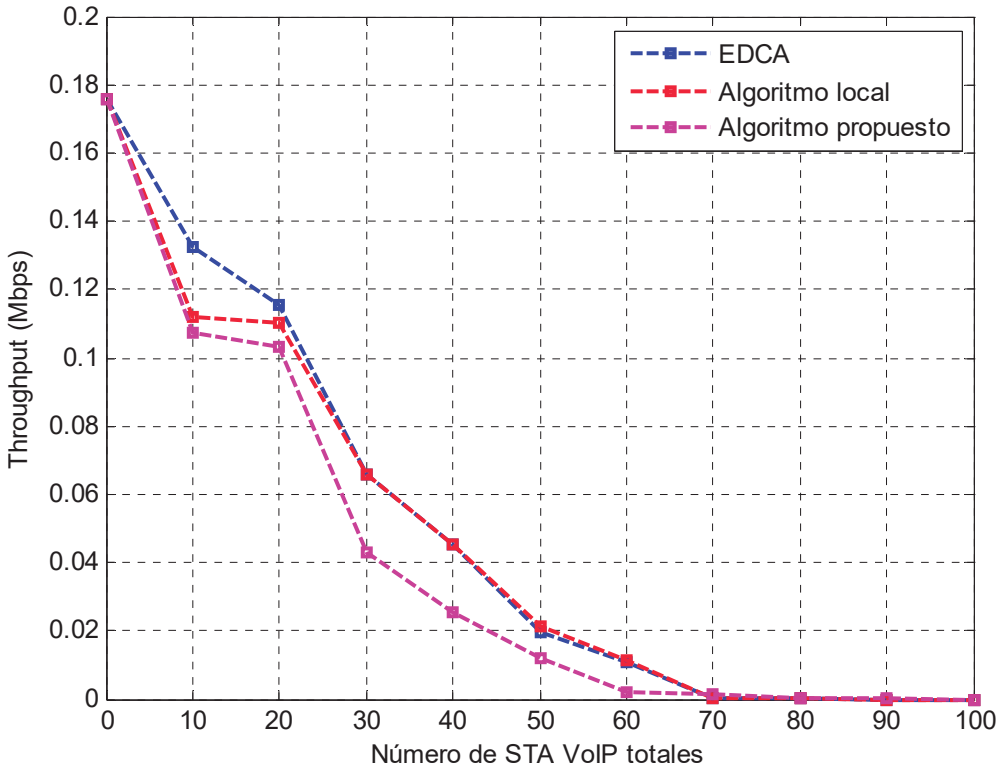


Figura 4.30. Throughput agregado medio del tráfico TCP en el anillo central (AP1-7) en el escenario con múltiples AP.

La figura 4.30 muestra claramente cómo el algoritmo propuesto consigue bajar el retardo del tráfico VoIP a costa de darle más prioridad, lo que implica que en este caso el throughput del tráfico TCP (cola ***AC_BK***) disminuya cuando se compara con el de EDCA o el algoritmo local.

Capítulo 5

Conclusiones y líneas futuras de trabajo

5.1 Conclusiones

En este trabajo se ha propuesto un algoritmo centralizado para proveer QoS en una red inalámbrica basada en el estándar 802.11. El algoritmo propuesto mejora el mecanismo EDCA, el cual es la solución adoptada en el propio estándar para suministrar QoS.

El algoritmo propuesto se basa en un sistema centralizado, el cual ajusta los parámetros de los AP que conforman la red para intentar cumplir con los requisitos de QoS del tráfico más prioritario. En nuestro caso concreto, se ajustan dichos parámetros para que el retardo de aplicaciones sensibles al mismo, como la VoIP, esté por debajo de un umbral predefinido.

El algoritmo propuesto se ha simulado utilizando la herramienta ns-3, y sus resultados se han evaluado en varios escenarios donde se ha comprobado el mejor comportamiento del mismo frente a EDCA y una solución no centralizada en la que cada AP actúa de modo independiente al resto.

5.2 Líneas futuras de trabajo

Partiendo de este trabajo, se proponen las siguientes líneas futuras de trabajo:

- Empleo del resto de parámetros de transmisión ajustables, como CW_{max}, $AIFSN$ y $TXOP$ para intentar mejorar las prestaciones de la solución propuesta.

- Análisis del impacto e introducción de mecanismos de coordinación cuando se emplean otras opciones disponibles en el estándar 802.11 como la agrupación de tramas.

- Optimización conjunta de los parámetros del mecanismo de acceso al medio junto con la selección del canal y las potencias de transmisión.

- Implementación del sistema en dispositivos reales, programando e incorporando el algoritmo en los routers y en un controlador central, de tal modo que se pueda evaluar el impacto asociado a la carga de tráfico o los retardos que implican la señalización entre los AP y el controlador. Este trabajo lleva asociado el diseño del protocolo de señalización entre los AP y el controlador.

Bibliografía

[**1**] Enlace a la página Web del proyecto Wi-5: http://www.wi5.eu/ [Fecha de consulta: 23/11/16]

[**2**] Enlace a la página Web del programa de investigación "Horizon 2020": https://ec.europa.eu/programmes/horizon2020/ [Fecha de consulta: 23/11/16]

[**3**] IEEE 802.11-2012: https://standards.ieee.org/findstds/standard/802.11-2012.html [Fecha de consulta: 23/11/16]

[**4**] Enlace a la página Web de Hermes: https://i3a.unizar.es/es/laboratorio/cluster-de-supercomputacion-hermes [Fecha de consulta: 23/11/16]

[**5**] Enlace a la página Web de Cisco: http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Mobility/vowlan/41dg/vowlan41dg-book/vowlan_ch2.html [Fecha de consulta: 23/11/16]

[**6**] Enlace a la página Web de referencia: http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob30dg/WANQoS.html [Fecha de consulta: 23/11/16]

[**7**] ITU-T G-114: https://www.itu.int/rec/T-REC-G.114/es [Fecha de consulta: 23/11/16]

[**8**] Optimal Configuration of 802.11e EDCA for Real-Time and Data Traffic. IEEE Transactions on Vehicular Technology (Volume: 59, Issue: 5, Jun 2010)

[**9**] Tuning the EDCA parameters in WLANs with heterogeneous traffic: A flow-level analysis. Computer Networks: The International Journal of Computer and Telecommunications Networking archive (Volume 54, Issue 13, September, 2010)

[**10**] P. Serrano, Estrategias de Configuración de Redes WLAN IEEE 802.11e EDCA, Tesis Doctoral, Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, September 2006.

[**11**] V.A. Siris, M. Kavouridou, Achieving service differentiation and high utilization in IEEE 802.11, Lecture Notes in Computer Science (2003) 128–137.

[**12**] Enlace al modelo de propagación por defecto utilizado por ns3: https://www.nsnam.org/docs/release/3.3/doxygen/classns3_1_1_log_distance_propagation_loss_model.html. [Fecha de consulta: 23/11/16]

[**13**] Enlace a los datos por defecto de ns3 sobre los STA y los AP: http://www.nsnam.org/doxygen/wifi-phy_8cc_source.html#l00475 [Fecha de consulta: 23/11/16]

[**14**] Enlace al documento con escenarios de simulación de IEEE 802.11: https://mentor.ieee.org/802.11/dcn/14/11-14-0980-16-00ax-simulation-scenarios.docx [Fecha de consulta: 23/11/16]

[**15**] Enlace a la página Web donde se describen los módulos existentes en ns3: https://www.nsnam.org/docs/release/3.13/manual/singlehtml/index.html [Fecha de consulta: 23/11/16]

[**16**] Enlace al esquema de WifiNetDevice: https://www.nsnam.org/docs/release/3.12/models/html/wifi.html [Fecha de consulta: 23/11/16]