Universidad Zaragoza
1542

Trabajo Fin de Máster

Propuesta y desarrollo de mecanismos para RAN Slicing en redes WLAN 5G en entorno real experimental

Proposal and Development of Mechanisms for RAN Slicing in 5G WLAN Networks in a Real Experimental Environment

Autor/es

Sara Ibáñez Alloza

Director/es

Ángela Hernández Solana

Titulación del autor

Máster Universitario en Ingeniería de Telecomunicación

ESCUELA DE INGENIERÍA Y ARQUITECTURA
AÑO 2025

RESUMEN

El slicing en redes permite asignar recursos específicos de la red a diferentes tipos de tráfico, dependiendo de sus necesidades, lo que permite garantizar un uso eficiente y adaptado de los recursos de la red.

Este trabajo fin de máster se centra en la implementación de mecanismos para RAN Slicing en redes WLAN, contando en todo momento con las limitaciones y dificultades que supone realizarlo en un entorno real. Con esta finalidad, se crea un sistema de colas que almacena los paquetes que deben ser transmitidos a las estaciones (STAs) y se aplica sobre ellos un esquema de slicing basado en estructura compartida (ISS) y basado en calidad de servicio (QoS).

Para ello, el sistema de colas tiene dos niveles jerárquicos. En primer lugar, el tráfico se divide por tipo de cliente y se almacena en los slices. En segundo lugar, el tráfico en un slice se clasifica a su vez en colas por la clase de servicio a la que pertenece. Estos permiten el aislamiento y la priorización, mientras se realizan las segmentaciones adecuadas de los recursos.

El algoritmo usado para el reparto de recursos entre ellos está basado en los principios de Airtime Deficit Weighted Round Robin (ADWRR), donde, a través de la configuración de un valor de quantum, se establece el porcentaje de los recursos que puede usar cada partición. A él se añade la posibilidad de adaptar la reasignación natural del esquema, que se produce cuando alguna de las partes no hace uso total de la porción asignada. De este modo, se presentan propuestas para que esta redistribución siga unos requisitos específicos, como pueden ser la priorización de unos slices frente a otros o el intento de que su satisfacción en cuanto a tasa sea igual en todos ellos.

Por último, se realizan pruebas diseñadas para mostrar el comportamiento de la implementación ante diferentes situaciones, como pueden ser los flujos de tráfico variantes o las retransmisiones. Estas pruebas ayudan a validar la efectividad de la solución propuesta.

ÍNDICE

1. INTRODUCCIÓN **3**
1.1. Contexto y objetivos del proyecto 4
1.2. Herramientas 5
1.2.1. C 5
1.2.2. Iperf 5
1.2.3. Matlab 5
1.3. Estructura de la memoria 5

2. ESTADO DEL ARTE **7**

3. ARQUITECTURA **9**
3.1. Sistema de colas en C 11
3.2. Clasificación de tráfico 13

4. ALGORITMOS **14**
4.1. Inter/Intra-slice scheduling 14
4.1.1. ADRR 14
4.1.2. Pruebas y resultados 21
4.1.2.1. Configuración de las pruebas 21
4.1.2.2. Procesado de los resultados 22
4.1.2.3. Evaluación y resultados 22
4.2. Redistribución del quantum sobrante 30
4.2.1. Reparto para igualar tasas de satisfacción 36
4.2.2. Reparto por prioridad 37
4.2.3. Redistribución Inter-Slice 37
4.2.4. Pruebas y resultados 40
4.2.4.1. Procesado de los resultados 40
4.2.4.2. Evaluación y resultados 40

5. CONCLUSIÓN **51**

BIBLIOGRAFÍA **52**

ÍNDICE DE FIGURAS **53**

ÍNDICE DE TABLAS **54**

LISTA DE ACRÓNIMOS **55**

ANEXOS **56**

I. RR **57**

II. Reparto proporcional a los grados de satisfacción **60**

1 INTRODUCCIÓN

El término de slicing consiste, como su nombre indica, en segmentar. En el ámbito de las telecomunicaciones, este concepto se extiende a lo conocido como "Network Slicing", es decir, slicing en redes. Este surge de la necesidad de reparto de los mismos recursos físicos entre diferentes redes lógicas. De este modo, esta técnica permite dividir de forma eficiente los recursos radio de una infraestructura WLAN coordinada en múltiples redes virtuales independientes, llamadas *slices*. Cada slice puede estar optimizado para diferentes perfiles de clientes, tipos de tráficos y perfiles de aplicación, con requisitos específicos, proporcionando un aislamiento en términos de capacidad y calidad de servicio.

A pesar de la importancia del slicing y de los avances significativos en la gestión y orquestación de slices, las propuestas para el slicing en la Red de Acceso por Radio (RAN) son limitadas debido a la variabilidad de los entornos radioeléctricos. A diferencia de otras redes donde los entornos son estables, RAN se enfrenta a cambios dinámicos como la variación de la intensidad de la señal, las interferencias, la movilidad de los usuarios y la variabilidad en las demandas de tráfico. Todo ello resulta en un entorno impredecible que dificulta la asignación y el aislamiento de los recursos de forma que se cumplan unos requisitos.

Esto es lo que ocurre en las redes WLAN. Además, los estándares IEEE 802.11 operan en espectro no licenciado y dependen del mecanismo CSMA/CA (Acceso Múltiple con Detección de Portadora y Prevención de Colisiones). En comparación con las redes móviles 4G/5G, esto limita el control granular sobre el acceso al canal. Esta limitación supone una dificultad añadida para garantizar la calidad del servicio y deja la red más vulnerable a las interferencias externas y la congestión.

El slicing en estas redes suele plantearse como la asignación del airtime, es decir, el tiempo en que una transmisión entre un punto de acceso (AP) y una estación (STA) ocupa el medio inalámbrico compartido. Según [1] existen dos enfoques diferentes a la hora de plantear el slicing.

La **segmentación basada en compartición de estructura** (**ISS**, por sus siglas en inglés *Infrastructure-Sharing Slicing*). Esta corresponde al concepto tradicional de segmentación de la infraestructura en redes virtuales. Consiste en asignar un subconjunto de los recursos de la red/RAN a cada segmento.

La **segmentación basada en calidad de servicio** (**QoS**, por sus siglas en inglés *Quality-of-Service Slicing*). En esta segmentación, el objetivo es segmentar en función de clases de servicios diferentes. La asignación realizada a cada slice debe ser dinámica para cumplir los requisitos de QoS, gestionándose adecuadamente para impedir la colisión entre sus necesidades.

En este trabajo se propone un modelo de slicing en red RAN siguiendo el esquema ISS, pero integrando un nivel secundario de slicing QoS.

1.1 Contexto y objetivos del proyecto

Este trabajo fin de master se enmarca en el proyecto con número 171539 denominado PID2022-136476OB-I00: *Optimización de redes WLAN coordinadas de última generación basadas en arquitecturas programables y virtualizadas (NeWLAN)*, desarrollado por el grupo de investigación CeNIT (*Communications Networks and Information Technologies*) y financiado por MICIN y FEDER, y del que he formado parte desde enero de 2024.

El objetivo principal de este TFM es la extensión del concepto de slicing, explicado en el apartado anterior, a redes WLAN de quinta generación, su implementación y validación en el entorno experimental desarrollado en el Proyecto *NeWLAN*. En las tecnologías IEEE 802.11, se pretende distribuir el airtime (tiempo radio) a través de la definición y especificación de mecanismos de gestión y scheduling de colas en los puntos de acceso (APs), entre los diferentes slices.

Para lograr este propósito, se plantean los siguientes objetivos específicos:

Revisión del estado del arte y especificación del problema. Analizar la bibliografía existente para identificar los desafíos y oportunidades relacionados con el concepto de slicing en redes IEEE 802.11.

Desarrollo de un sistema de colas en C. Implementar un sistema de slices subdivido a su vez en un sistema de colas. Este sistema será usado posteriormente para poder garantizar el aislamiento y el trato diferenciado de los paquetes almacenados en cada una de ellas.

Desarrollo y programación de algoritmos específicos. Programar diferentes algoritmos destinados a la gestión y el reparto de los recursos entre los diferentes slices, y sus colas, del sistema.

Validación del proyecto en un testbed WiFi real. Implementar el proyecto en un entorno de prueba real. En este testbed, dentro del proyecto *NeWLAN*, se desarrolla toda la funcionalidad inteligente de los puntos de acceso y solo se usan las capacidades básicas del protocolo de acceso radio de tarjetas comerciales en las que es preciso inyectar el tráfico de la manera adecuada.

Definición y puesta en marcha de los escenarios de pruebas. Diseñar pruebas experimentales que permitan evaluar de manera exhaustiva el comportamiento de los algoritmos en distintas condiciones operativas.

Procesado y análisis crítico de los resultados. Crear un programa de MATLAB para el procesado de los datos experimentales de las pruebas recogidos en un fichero log. Así como la posterior representación gráfica de los parámetros relevantes para la validación del proyecto implementado. A partir de estos, realizar un análisis crítico de las soluciones propuestas.

1.2 Herramientas

1.2.1 C

El lenguaje de programación empleado para el desarrollo del proyecto es C. Es el lenguaje utilizado en la programación del punto de acceso y, en su momento, se eligió debido a su capacidad para minimizar la latencia y maximizar el rendimiento, requerimientos fundamentales para el propósito del proyecto.

1.2.2 Iperf

Iperf es una herramienta para mediciones activas de ancho de banda máximo alcanzable en redes IP [2]. Ha sido usado como generador de tráfico para las pruebas, modificando parámetros como el tamaño de los paquetes, la tasa de transmisión y el tiempo, e indicando la dirección IP y el puerto de destino.

1.2.3 Matlab

MATLAB es un entorno de programación y un lenguaje desarrollado por *MathWorks*. Se ha usado para crear un programa capaz de procesar los logs generados durante las pruebas. Este, a partir de los datos obtenidos del procesado, genera las gráficas de los parámetros correspondientes para permitir el análisis de la prueba, como pueden ser el porcentaje de airtime usado, el throughput (tasa de transferencia efectiva) o el número de paquetes transmitidos por segundo.

1.3 Estructura de la memoria

La memoria se estructura de la siguiente manera. En el apartado actual se ha introducido el TFM, hablando del contexto en el que se encuentra, los objetivos que pretende lograr y las herramientas usadas para ello.

En el apartado 2 se revisan algunas de las propuestas sobre RAN Slicing en WLAN, analizando sus ventajas, limitaciones y aplicaciones. Este análisis proporciona las bases para justificar y contextualizar las contribuciones del trabajo propuesto.

A continuación, en el apartado 3, se describe la arquitectura del proyecto. En él se explica, además, la creación del sistema de colas necesario y cómo se realiza la clasificación de tráfico para almacenarlo en ellas correctamente.

Después, en el apartado 4, se describen detalladamente los algoritmos implementados, tanto los generales para el reparto entre slices y colas, como las propuestas para la redistribución dinámica de los quantums/pesos. Así mismo, para todos ellos, se presentan los escenarios de

pruebas elegidos y se analizan de manera crítica los resultados obtenidos en condiciones reales.

Por último, en el apartado 5, se hace un repaso del trabajo realizado y los objetivos alcanzados, incluyendo una reflexión acerca de las posibles líneas futuras de investigación.

2 ESTADO DEL ARTE

Una de las referencias más citadas en cuanto al slicing extremo a extremo (E2E, por sus siglas en inglés *End-to-End*) en redes WLAN en entorno real es Lasagna, que se presenta en [3]. En ella se propone una gestión centralizada de la red mediante aplicaciones basadas en Python que se ejecutan en un controlador. En cada interfaz Wi-Fi del AP, los paquetes son clasificados en diferentes colas según sus slices, basados en reglas de tráfico como pueden ser las de OpenFlow. En cuanto al proceso de desencolado, se aplica un esquema ADWRR entre los slices, mientras que entre sus colas se usa Round Robin.

En el esquema ADWRR cada cola tiene asociado un contador llamado contador de déficit (DC, por sus siglas en inglés), este contador indica la cantidad de tiempo de transmisión (en μs) que puede utilizar en la ronda. Las colas se visitan siguiendo un esquema rotativo (*round robin*) y en cada una de ellas el contador DC incrementa en una cantidad fija Q (quantum, valor independiente en cada cola y determinado en configuración). Cuando es el turno de una cola, esta transmite mientras le queden paquetes y el tiempo de transmisión del primer paquete encolado sea menor que su DC. Si esto se cumple, la cola transmite el paquete y su DC se reduce en esa cantidad de tiempo. En el momento que el valor de DC es insuficiente o su cola queda vacía (momento en que se reiniciará el valor de su DC a 0), se pasa el turno a la siguiente cola.

Esta propuesta emplea una asignación de quantum estática, lo que se alinea con el enfoque ISS definido en [1]. En ella se muestra que el esquema ADWRR garantiza correctamente el aislamiento de recursos entre slices. No obstante, la evaluación para comprobar que la asignación a cada uno se adecúe con la proporción designada con el quantum se realiza en función de las tasas de transmisión logradas. Es decir, se compara la de las STAs de un slice con la suma de las tasas en todos los slices. Esta medida no es adecuada cuando tratamos con un sistema real donde las STAs transmiten paquetes con diferentes tamaños, empleando distintas MCSs y con diferentes tasas de error.

En el trabajo [4] se extiende la implementación de ADWRR al slicing QoS [1]. En él se crean slices para hacer diferenciación de servicio y se les asignan quantums distintos que permiten priorizar unos sobre otros. Como la asignación se hace según la relación entre el quantum de un slice y la suma del de todos, un quantum mayor implica un mayor acceso a los recursos radio. Sin embargo, existen variabilidades en el entorno que pueden hacer que la selección del quantum no pueda garantizar de manera efectiva el control de QoS. Es por ello que en este trabajo se propone la implementación de un algoritmo heurístico cuya implementación se basa en verificar si se cumplen los requisitos QoS de la aplicación, vinculados al slice, y, en función de ello, modificar dinámicamente su *quantum*. El problema es que, al adaptar el *quantum* de cada slice de manera independiente, utilizando factores de incremento o decremento, se pierde el control sobre la partición de la asignación de recursos entre slice y los efectos globales sobre la QoS del resto de slices. De hecho, esta es una de las principales limitaciones de esta propuesta

y de muchas otras que sugieren la adaptación dinámica del *quantum* para la segmentación basada en QoS.

Por último, en [1] se propone un enfoque alternativo de slicing ISS llamado Adaptive Time-Excess Round Robin (ATERR). En este esquema, se garantiza a cada slice un porcentaje del airtime total en un AP. Si alguna porción de airtime no se utiliza, esta se reparte proporcionalmente entre el resto de los slices. Esto constituye una adaptación de ADWRR, pero se usa el exceso de tiempo en lugar del déficit. Por otro lado, dentro de cada slice existen varias colas, cada una asociada a un flujo de tráfico de usuario distinto. El quantum de cada slice se distribuye equitativamente entre las que se encuentren activas (aquellas que contienen paquetes). Se establecen relaciones matemáticas entre el quantum del slice y el de cada cola. El modelo matemático propuesto, calcula dinámicamente los valores correspondiente cada vez que una cola se activa o desactiva. Sin embargo, según se ha visto en este trabajo, los mismos objetivos pueden alcanzarse con una implementación más simple y, además, extenderse para gestionar cuotas de tiempo de transmisión diferenciadas entre las STAs.

Ante todo esto, el trabajo presente propone un enfoque híbrido ISS-QoS para el componente local del slicing, que, a través de una programación jerárquica basada en los principios de ADWRR/ADRR, garantiza aislamiento y reparto de recursos entre categorías de clientes. A su vez, ofrece diferenciación del tráfico, priorización y aislamiento de recursos para las clases de servicio dentro de los slices. La propuesta incluye adaptaciones locales del quantum a ambos niveles para mejorar los requisitos específicos de QoS, modificando la reasignación natural del esquema ADWRR.

Además, se enfatiza en la importancia de abordar los problemas prácticos de la implementación en un entorno real, para lo cual se propone:

- **Distribución sencilla y eficaz del quantum**. Basándose en un esquema ADWRR/ADRR el procedimiento propuesto reparte el quantum entre las colas de un slice sin emplear técnicas de optimización matemática.

- **Adaptación del quantum basada en redistribuciones controladas**. Este enfoque permite anticipar y gestionar de manera precisa los efectos sobre otros slice, garantizando una redistribución efectiva de los recursos y manteniendo la equidad.

- **Estimación de airtime para ADWRR con correcciones por retransmisiones**. Se estima el airtime en función de la tasa física real utilizada y el déficit se corrige tras conocer el número real de retransmisiones que se han producido.

- **Compatibilidad entre dispositivos de diferentes fabricantes**. El diseño considera las particularidades del hardware y software de IEEE 802.11, pero sin depender de implementaciones específicas.

- **Pruebas bajo condiciones reales**. Los métodos propuestos se someten a pruebas en condiciones reales, con tráfico variante y cambios en las condiciones del canal.

3 ARQUITECTURA

El prototipo experimental desarrollado en este trabajo, que implementa las funcionalidades propuestas de RAN slicing, se basa en el framework de código abierto descrito en [5] y derivado de [6]. Su arquitectura de alto nivel se muestra en la Figura 1, donde se observa que está formado por un controlador central y agentes distribuidos (APs).

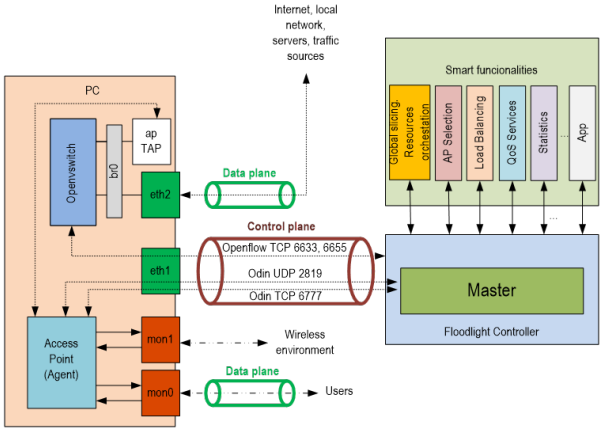


Figura 1: Arquitectura implementada a alto nivel.

El controlador recopila información de los APs y permite que las aplicaciones de red tomen decisiones inteligentes con respecto a la asignación de clientes a los APs, el balanceo de carga, y la coordinación y gestión de recursos entre los APs. De manera distribuida, el software implementado en los agentes locales, que están vinculados a los APs, gestiona funcionalidades MAC no críticas en tiempo, siempre dentro de los ajustes y políticas seleccionados por el controlador central. Esta configuración ayuda a optimizar QoS y el uso eficiente de los recursos a nivel local. Por el contrario, los aspectos críticos del protocolo MAC de Wi-Fi, como la Función de Coordinación Distribuida (DCF), los tiempos y los reconocimientos, son gestionados por el núcleo mac802.11, independientemente del controlador específico utilizado.

En esta arquitectura, la solución de RAN slicing para Wi-Fi basada en SDN sigue un enfoque jerárquico que combina componentes centralizados y distribuidos para gestionar la segmentación en despliegues Wi-Fi densos. El elemento principal es el componente de slicing local, implementado en el agente local. Este es el encargado de distribuir el tiempo de transmisión entre los flujos de los clientes Wi-Fi asociados a un determinado AP, siguiendo las políticas implementadas que se basan en una versión revisada del conocido algoritmo ADWRR o ADRR. De este modo, la porción de tiempo recibida depende del slice al que pertenezca, así como de la clase de servicio.

El agente local está vinculado al módulo mac80211 y es compatible con cualquier chipset Wi-Fi que funcione en Linux y admita la inyección de tramas en modo monitor, lo que lo convierte en una opción ideal para pruebas experimentales en entornos de redes Wi-Fi.

En la Figura 2 se muestra en detalle el funcionamiento del plano de usuario, centrándose en la implementación del componente de slicing local dentro de los APs.

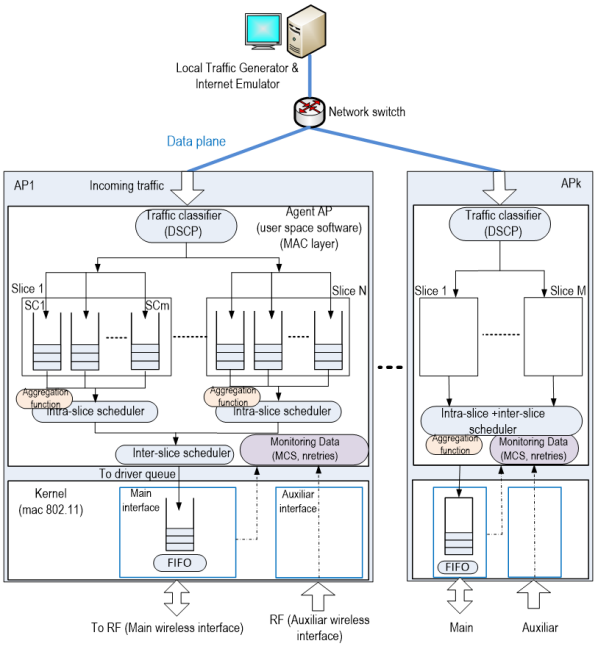


Figura 2: Implementación local del AP (plano de datos)

En ella se observa que el AP tiene tres interfaces: la fija, la principal y la auxiliar. La fija permite al AP comunicarse con el resto de la infraestructura de red. La principal, configurada para soportar la inyección de tramas en modo monitor, permite capturar y transmitir todas las tramas en un canal específico. La inyección de tramas según Radiotap [7] permite la transmisión activa de tramas creadas o modificadas en el medio inalámbrico, lo cual es esencial para escenarios de prueba como el RAN Slicing. La interfaz inalámbrica auxiliar, también configurada en modo monitor, se usa para la monitorización de todas las tramas transmitidas y recibidas por la principal.

Siguiendo esta estructura, el tráfico que llega al AP a través de la interfaz principal es clasificado (ver el apartado 3.2) para ser tratado en el slice y cola correspondiente.

Llamamos slice al primer nivel jerárquico en la división de los recursos de la red. Dentro de cada slice, existe un subnivel de colas que permite, a su vez, diferenciar el tráfico por clase de servicio (la implementación de este sistema se explica en el apartado 3.1).

Mientras el agente del AP revisa la llegada de tráfico y lo almacena en la cola correspondiente, un hilo asíncrono, en paralelo, se encarga de desencolar estos paquetes y transmitirlos a las STAs destino. Es decir, este hilo se encarga de gestionar la planificación de cada cola siguiendo los esquemas definidos, tanto a nivel intra-slice como a nivel inter-slice. Dichos esquemas pueden ser RR o ADWRR. En caso del segundo, la asignación se realiza en términos de fracciones de airtime reservadas para cada slice en el AP, determinadas por los valores de quantum configurados. Seleccionado el paquete a enviar (uno o varios si se aplica agregación), a través

de estos esquemas, este se desencola, se construye la trama y queda almacenada en el buffer del driver hasta su transmisión por la interfaz principal.

Por otro lado, el AP también recibe paquetes a través de la interfaz auxiliar utilizada para monitorización. Las tramas capturadas en ella permiten obtener información en tiempo real necesaria para la implementación de los mecanismos de control de tasa y scheduling. Lo ideal sería que esta información se obtuviera directamente del driver de la interfaz principal. Sin embargo, ante la imposibilidad de obtener la información del driver, la interfaz auxiliar proporciona una solución viable a nivel de testbed que no resta validez a los desarrollos realizados. En concreto, se accede a información disponible en sus cabeceras Radiotap, como la tasa (MCS), la cabecera 802.11 (incluyendo dirección de origen y destino, número de secuencia, longitud del paquete, si es un retransmisión, etc.) o la cabecera IP (Differentiated Services Code Point, DSCP; longitud del paquete IP). A partir de los datos monitorizados, se obtiene la información necesaria para controlar la ocupación física del buffer del driver y contabilizar el número de retransmisiones de los paquetes. Esto último sirve para la implementación del control de tasa y para el correcto cálculo del airtime. Por último, se obtienen otros datos estadísticos que nos permiten evaluar el sistema y definir ajustes dinámicos del quantum en caso de emplear el esquema ADWRR.

3.1 Sistema de colas en C

Como se ha explicado anteriormente, los recursos de la red son divididos en porciones denominadas slices. Cada una de estas, a su vez, tiene asociadas una serie de colas entre las que se reparte esa porción. En este apartado se pretende ilustrar brevemente la implementación de este sistema en C.

Una cola es una estructura con una serie de variables asociadas que nos permiten hacer la clasificación de tráfico, el almacenamiento de los paquetes y su liberación, tal y como se expone en este trabajo. Las variables que almacena son:

- Un puntero al paquete en cabeza (q_{head}), es decir, aquel que lleva más tiempo encolado y debe ser el siguiente en liberarse.

- Un puntero al paquete último en la cola (q_{tail}), es decir, aquel que es más nuevo y determina el final de la cola.

- El tamaño de la cola o, lo que es lo mismo, el número de paquetes que tiene almacenados (q_{size}). Esto nos permite conocer si la cola está vacía o tiene paquetes acumulados (está activa).

- El identificador (ver Apartado 3.2). Este valor identifica la cola y el slice al que pertenece.

- Valores asociados con la agregación: si está habilitada en la cola y, en caso de estarlo, cuál es el tamaño máximo permitido.

- Valores asociados a tasas. Se recogen estadísticas sobre la tasa demandada y la tasa servida,

así como parámetros asociados al acuerdo de nivel de servicio (SLA, por sus sigla en inglés *Service Level Agreement*) como la tasa máxima de bit (MBR, por sus siglas en inglés *Maximum Bit Rate*).

- Valores asociados a ADWRR. Si este es el esquema intra-slice se asocian a la cola los datos necesarios para ello, como son el peso y el déficit.

A esto se le añade un mutex, ya que es el programa principal el que encola los paquetes, pero es un hilo en paralelo el que se encarga de la gestión y la liberación.

Cabe destacar que lo que se está denominando paquete es en realidad una estructura, que a partir de ahora llamaré nodo, que, además del propio paquete en sí, guarda su longitud IP, todos los datos asociados a la STA destino (como pueden ser la MCS, la MAC, ...) y un puntero hacia el siguiente nodo en la cola.

A partir de esta estructura es posible generar funciones encargadas de la creación y destrucción de las colas, así como de su encolado y desencolado.

El proceso de encolado (Algoritmo 1) consiste en almacenar el nuevo paquete al final de la cola. Para ello se guarda en la estructura de un nodo toda la información descrita anteriormente que está relacionada con el paquete (línea 1 del Alg.1) . Al ser este un nuevo paquete, debe colocarse al final, es decir, el puntero que apuntaba a la cola ahora va a apuntar a este nuevo nodo, que a su vez se marca como siguiente del que antes era el último (líneas 5-7 del Alg.1). En caso de que la cola estuviera vacía, este nuevo nodo se marcaría a la vez como la cabeza y el final de la cola (líneas 2-4 del Alg.1). Por supuesto, finalizado este proceso, se añade uno al valor del tamaño de la cola (línea 9 del Alg.1).

Algoritmo 1 Proceso de encolado

▷ q_{head}: paquete en la cabeza de la cola
▷ q_{tail}: paquete al final de la cola
▷ q_{size}: tamaño de la cola

1: Se guardan todos los valores necesarios relativos al paquete en la variable $nodo$
2: **if** q_{head} == NULL **then** ▷ Si la cola está vacía el nuevo paquete es el único
3: $q_{head} = nodo$
4: $q_{tail} = nodo$
5: **else** ▷ Si ya había paquetes, se añade al final de la cola
6: $q_{tail} -> next = nodo$
7: $q_{tail} = nodo$
8: **end if**
9: q_{size} += 1

En cuanto al proceso de desencolado (Algoritmo 2), este consiste en recuperar el paquete que más tiempo lleva almacenado en la cola. Para ello se devuelve el puntero que marcaba la cabeza de la cola (línea 1 del Alg.2) y se marca como nueva cabeza la que tenía como siguiente (línea 6 del Alg.2). En caso de que fuera el único nodo en la cola (la posición a la que apuntan cabeza y final es la misma), estas pasan a NULL (líneas 2-4 del Alg.2). De nuevo, finalizado el proceso se actualiza el tamaño disminuyendo en uno (línea 8 del Alg.2).

Una vez se tienen las colas, cada slice se plantea como una estructura que tiene asociada, a parte de las variables equivalentes a las de la cola, un vector de colas. De este modo, se consigue tener

una estructura inicial que a su vez contenga el sistema de las colas que almacenan y liberan paquetes.

Algoritmo 2 Proceso de desencolado

▷ q_{head}: paquete en la cabeza de la cola
▷ q_{tail}: paquete al final de la cola
▷ q_{size}: tamaño de la cola
▷ $nodo$: paquete desencolado

1: $nodo = q_{head}$ ▷ Se guarda el paquete a desencolar en una variable
2: **if** $q_{head} == q_{tail}$ **then** ▷ Si solo hay un paquete en la cola, se vacía
3: q_{head} = NULL
4: q_{tail} = NULL
5: **else** ▷ Si ya hay más de un paquete, actualiza el que está en cabeza
6: $q_{head} = q_{head} -> next$
7: **end if**
8: q_{size} -= 1
9: **return** $nodo$

Los esquemas a partir de los cuales se selecciona el slice y su cola concreta que desencola el paquete serán explicados en el Apartado 4.

3.2 Clasificación de tráfico

La clasificación del tráfico que llega al AP se basa en el campo DSCP (*Differentiated Services Code Point*) que forma parte del campo TOS (tipo de servicio, por sus siglas en inglés *Type of Service*), el segundo byte en la cabecera de los paquetes IP. A través de este se clasifican los paquetes y se gestiona el tráfico en la red de acuerdo con las políticas de QoS.

El campo DSCP tiene una longitud de 6 bits, los cuales se usan para señalar el slice y cola al que pertenece ese paquete.

Como se muestra en la Figura 3, se usan los bits más significativos para indicar el slice en el que debe almacenarse la trama y los menos significativos para mostrar la cola concreta dentro de este slice. De este modo se puede mapear adecuadamente el tráfico.

Al usarse 3 bits para cada uno de estos identificadores, se pueden emplear un máximo de 8 slices de hasta 8 colas cada uno.

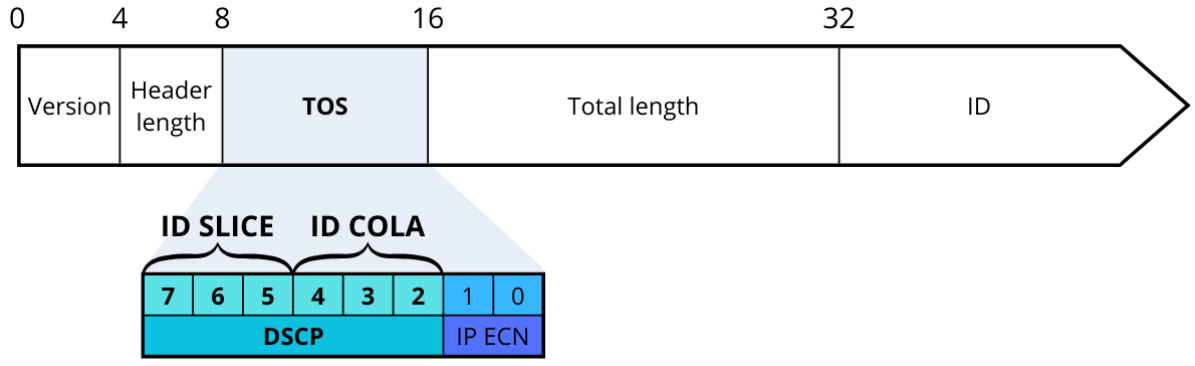


Figura 3: Estructura trama IP y uso del campo DSCP.

4 ALGORITMOS

En este apartado se presenta la propuesta e implementación de algoritmos de scheduling, así como escenarios de prueba que van a permitir llevar a cabo un análisis crítico de los mismos.

Esta sección se divide en dos grandes bloques, por un lado, la presentación de algoritmos de scheduling, y, por otro, la propuesta de añadidos para perfeccionar la asignación de recursos.

4.1 Inter/Intra-slice scheduling

Llamamos algoritmo inter-slice a aquel encargado de realizar el reparto de recursos entre los slices activos (tienen al menos una cola con paquetes), e intra-slice al que distribuye los recursos entre las colas activas (tienen paquetes) que pertenecen a un mismo slice.

En este TFM se han implementado dos algoritmos, el esquema de ADWRR (*Airtime Deficit Weighted Round Robin*), que a lo largo de este TFM se nombrará simplemente como ADRR, y RR (*Round Robin*). El RR, más sencillo, se implementó como paso previo a la implementación del ADRR y se recoge en el Anexo I.

El uso de cualquiera de ellos como algoritmos inter/intra-slices es independiente, es decir, el reparto entre los slices puede seguir un esquema ADRR mientras que las colas dentro de cada uno de ellos siguen el esquema ADRR o RR para el reparto de recursos.

4.1.1 ADRR

Como se ha comentado anteriormente, en el esquema ADRR cada cola tiene asociado un contador llamado contador de déficit (DC, por sus siglas en inglés), este contador indica la cantidad de tiempo de transmisión (en μs) que puede utilizar en la ronda. Las colas se visitan siguiendo un esquema rotativo (*round robin*) y en cada una de ellas el contador DC incrementa en una cantidad fija Q (quantum, valor independiente en cada cola y determinado en configuración).

Cuando es el turno de una cola, esta transmite mientras le queden paquetes y el tiempo de transmisión del primer paquete encolado sea menor que su DC. Si esto se cumple, la cola transmite el paquete y su DC se reduce en esa cantidad de tiempo. En el momento que el valor de DC es insuficiente o su cola queda vacía (momento en que se reiniciará el valor de su DC a 0), se pasa el turno a la siguiente cola.

A continuación, en los algoritmos 3 y 4, se muestra la implementación del esquema ADRR como algoritmos de scheduling inter e intra slice, respectivamente, cuya notación se explica en la Tabla 1. En el apartado 4.2, se mostrarán mejoras para perfeccionar la asignación de recursos. Para ello, se realizarán adaptaciones locales del quantum con el objetivo de cumplir

con requisitos específicos de QoS y modificaciones en la reasignación natural de los recursos no utilizados de los slices y las colas.

Variable	Descripción
$airtime[s,i]$	Airtime (tiempo de vuelo) estimado del primer paquete de la cola i del slice s (sin retransmisiones)
$Q[s]$	Quantum del slice s (configurado por el controlador)
$W[s,i]$	Peso de la cola i del slice s (W_{nom} configurado por el controlador)
$Deficit[s]$	Déficit del slice s
$Deficit[s,i]$	Déficit de la cola i del slice s
$Deficit_{retries}[s]$	Déficit acumulado del slice s debido a las retransmisiones
$Deficit_{retries}[s,i]$	Déficit acumulado de la cola i del slice s debido a las retransmisiones
N_{slices}	Número de slices
$N_{queue}[s]$	Número de colas del slice s

Tabla 1: Variables ADRR

$Q[s]$ garantiza que la proporción de tiempo de transmisión del AP asignada al slice s cumpla con los requisitos de rendimiento y recursos. De este modo, cuando el slice tiene tráfico pendiente en alguna de sus colas, se le garantiza, como mínimo, la proporción establecida en la Ecuación 1, donde $\sum_{j=0}^{N_{slices}} Q\,[j]$ es la suma de los quantum de todos los slices del AP.

$$\frac{Q\,[s]}{\sum_{j=0}^{N_{slices}} Q\,[j]} \tag{1}$$

Esta cantidad puede aumentar si alguno de los otros slices no tiene tráfico y, por lo tanto, no utiliza su asignación de tiempo de transmisión, o no tiene tanto como para requerir usar su asignación al completo.

Algoritmo 3 Inter-slice *s*cheduling ADRR

```
1:  s = 0
2:  loop every
3:      Mientras la ocupación física del buffer del driver sea el umbral, espera

        #Actualiza el déficit del slice s si tiene paquetes que enviar
4:      if Slice[s].empty == false then    ▷ Las colas de s tienen paquetes
            ▷ Actualiza el déficit con el quantum y el déficit del slice s debido a las retransmisiones anteriores. Deficit_retries siempre es ≥ 0
5:          Deficit[s] += Q[s] - Deficit_retries[s]
6:          Deficit_retries[s] = 0

            #Envía los paquetes del slice mientras Deficit[s]>0 y haya paquetes con airtime más pequeño que su Deficit[s]
7:          ALGORITMO INTRA-SLICE para servir las colas (Alg. 4 ó 8)
8:      end if

        #Continúa con el siguiente slice
9:      if s < (N_slices - 1) then s++
10:     else s = 0
11: end loop
```

Algoritmo 4 Algoritmo Intra-Slice para servir las colas del slice. Opción ADRR

#Suma el déficit de todas las colas del slice *s*
$w_{QueueNoEmpty}[s] = \sum_{\forall i_{empty} \in s} W[s,i]$

for cada cola i no vacía del slice s **do**
  $Q[s,i] = (W[s,i]/w_{QueueNoEmpty}) * Q[s]$
  $Deficit[s,i]$ += $Q[s,i] - Deficit_{retries}[s,i]$
  $Deficit_{retries}[s,i] = 0$
end for

#Proceso de desencolado
$q_{ini} = Slice[s].Queue_index$ ▷ Guarda el índice inicial de la cola para continuar la ronda
$i = q_{ini}$
while $Deficit[s] > 0$ **do**
  while $Deficit[s,i] > 0$ and $Queue[s,i]$.empty == false **do**
    Mientras la ocupación física del buffer del driver sea el umbral, espera

    #Opción de no agregación de paquetes
    ▷ Calcula el airtime del primer paquete de *Queue[s,i]*
    $airtime[s,i] = Queue[s,i]$.airtime(head packet)
    $npAggr = 1$

    #Opción de agregación de paquetes
    ▷ Calcula el número de paquetes que pueden ser agregados (*npAggr)* basado en el *Deficit[s,i]*, el máximo número de la agregación y el número de paquetes de la cola (A-MSDU)
    $npAggr = Queue[s,i]$.MaxAggregationAllowed($Deficit[s,i]$)
    $airtime[s,i] = Queue[s,i]$.airtime($npAggr$)

    if $airtime[s,i] \leq Deficit[s,i]$ **then**
      ▷ Desencola el paquete de la cabeza (A-MSDU) de la cola i (uno o más si pueden ser agregados) del slice s
      $Queue[s,i]$.dequeue($npAggr$)
      Construye la trama agregada con los paquetes desencolados
      Send(frame)
      $Deficit[s]$ -= $airtime[s,i]$
      $Deficit[s,i]$ -= $airtime[s,i]$

      if $Queue[s,i]$.empty == true **then**
        ▷ El último paquete de la cola *i* ha sido enviado. Su déficit es redistribuido proporcionalmente entre las colas activas basado en su quantum
        $w_{QueueNoEmpty}[s] = \sum_{\forall i \in s} W[s,i]$
        for cada cola j no vacía del slice s **do**
          $Deficit[s,j]$+=$(W[s,j]/w_{QueueNoEmpty}[s])$*$Deficit[s,i]$
        end for
        $Deficit[s,i] = 0$
        break ▷ Siguiente cola
      end if
    else break ▷ Siguiente cola
    end if
  end while

  #Continúa con la siguiente cola del slice *s*
  if $i <(N_{queues}[s]$ - 1) **then** i++
  else $i = 0$
  end if
  if $i == q_{ini}$ **then**
    $Slice[s].Queue_index = i$
    break ▷ Siguiente slice
  end if
  if $Slice[s]$.empty == true **then** $Deficit[s] = 0$ ▷ Siguiente slice
end while

En cuanto al segundo nivel de scheduling, la proporción de tiempo de transmisión del AP asignado al slice s debe distribuirse entre todas las colas activas dentro de dicho slice. Para garantizar que la proporción asignada a cada cola del slice s cumpla con los requisitos de rendimiento, el controlador asigna valores de ponderación adimensional $W[s,i]$ (siendo s el

slice e i la cola), a los que denominamos pesos. Así, se asegura que la cola i del slice s reciba una proporción mínima respecto al tiempo de transmisión total del slice (Ecuación 2).

$$\frac{W\left[s,i\right]}{\sum_{j=0}^{N_{queue}[s]} W\left[s,j\right]} \tag{2}$$

Con base en esta asignación, el quantum asociado a la cola i del slice s, denotado como $Q[s,i]$ (µs), se calcula en la línea 3 del Algoritmo 4 con la Ecuación 3. Como se observa en esta, la asignación puede ser superior si alguna de las colas del slice no está activa:

$$Q[s,i] = \left(\frac{W[s,i]}{\sum_{j \text{ no vacía} \in s} W[s,j]} \right) \cdot Q[s] \tag{3}$$

Este cálculo garantiza que cada cola activa reciba un quantum proporcional a su ponderación relativa dentro del slice, ajustándose dinámicamente a la demanda de tráfico. Este dinamismo es lo que diferencia a $Q[s,i]$ de $W[s,i]$, ya que este último es un valor fijo asignado por el controlador.

Continuando con la implementación de los algoritmos, el funcionamiento es el que se explica a continuación.

Siguiendo un esquema rotativo, los slices reciben su turno. El esquema rotativo, o ciclo circular, es aquel en el que los elementos, en este caso los slices, se recorren de forma cíclica: una vez que se llega al último, se regresa al primero y el proceso se repite indefinidamente (líneas 9-10 del Alg.3).

Cuando es el turno de un slice, si este no está vacío, es decir, alguna de sus colas tiene paquetes acumulados, actualiza su contador de déficit ($Deficit[s]$) incrementando su valor $Q[s]$ (línea 5 del Alg.3), así como los contadores de sus colas activas ($Deficit[s,i]$) aumentándolos un valor $Q[s,i]$ (líneas 2-6 del Alg.4). En ese momento, se activa el algoritmo inter-slice (línea 7 del Alg.3), en el que las colas del slice transmiten paquetes mientras el contador de déficit del slice ($Deficit[s]$) sea mayor que 0 y queden paquetes para transmitir con un airtime inferior al valor de este (líneas 9, 10 y 16 del Alg.4). Si no pasará el turno al siguiente slice (líneas 9-10 del Alg.3).

Una vez empieza el proceso de desencolado, el ciclo comienza desde la cola cuyo turno quedó pendiente en la última ronda (líneas 7, 8 del Alg.4). Al igual que los slices, el turno de las colas también se da siguiendo un esquema rotativo. La cola con el turno envía paquetes (líneas 17-19 del Alg.4) mientras no esté vacía (líneas 10 del Alg.4) y su contador de déficit ($Deficit[s,i]$) sea mayor al airtime del paquete a enviar (línea 16 del Alg.4). Cuando se envía un paquete, se actualiza el valor de los contadores de déficit de la cola ($Deficit[s,i]$, línea 21 del Alg.4) y de su slice ($Deficit[s]$, línea 20 del Alg.4) restándoles el airtime que se calcula que va a emplear ese paquete (obtenido en la línea 12 o 14, dependiendo de la opción elegida).

La trama que se ha enviado puede ser agregada o no, es una característica opcional que puede

aplicarse de manera selectiva e independiente a cada una de las colas. Existen dos opciones, si se ha configurado para que no se realice agregación (líneas 12-13 del Alg.4) se desencolará el primer paquete de la cola (línea 17 del Alg.4). En caso de que se haya configurado para que haya agregación (líneas 14-15 del Alg.4) se calculará el máximo número de paquetes que pueden agregarse. Los paquetes agregados tienen como destino la misma STA que el paquete a la cabeza de la cola y se permite la búsqueda de paquetes a lo largo de la cola para intentar que se agregue lo máximo posible. Este número depende del contador de déficit de la cola ($Deficit[s, i]$), del número de paquete que tenga y del máximo que se haya establecido en configuración. Obtenido el número, se desencolan los paquetes oportunos (línea 17 del Alg.4) y se construye la trama a enviar (línea 18 del Alg.4).

En el momento en que la cola queda vacía o no tenga déficit suficiente para transmitir, se pasa el turno a la siguiente cola (líneas 28, 30 y 33-35 del Alg.4). Pero si todos las colas han recibido ya su turno, se pasa al siguiente slice (líneas 36-39 del Alg.4, 9-10 del Alg.3).

Si un slice no tiene paquetes para transmitir en ninguna de sus colas, pero mantiene un valor positivo en su contador de déficit ($Deficit[s, i]$), se reinicia su contador a 0 (línea 40 del Alg.4) y se pasa el turno al siguiente slice (líneas 9-10 del Alg.3). De igual manera, si quedan paquetes, pero el déficit no es suficiente para transmitirlos, se pasa el turno al siguiente slice (líneas 9-10 del Alg.3).

Cabe aclarar que los estándares 802.11n y 802.11ac permiten dos tipos de agregación de tramas: A-MPDU (por sus siglas en ingles *Aggregated MAC Protocol Data Unit*) y A-MSDU (por sus siglas en inglés *Aggregated MAC Service Data Unit*). La que se emplea en este proyecto es A-MSDU, los MSDU de varios paquetes se agregan en un solo MPDU, añadiendo una única cabecera MAC y FCS y, luego, la cabecera correspondiente de la trama.

No obstante, se han aplicado correcciones y añadidos para garantizar que el comportamiento sea el esperado en condiciones del mundo real.

En primer lugar, cuando una cola se vacía porque acaba de transmitir el último paquete que tenía almacenado, el valor de su contador de déficit ($Deficit[s, i]$) es redistribuido entre el resto de las colas activas del slice s antes de reiniciarse a 0 (líneas 22-29 del Alg.4). Esta redistribución se hace proporcionalmente al peso, siguiendo la Ecuación 2. De este modo, el contador de déficit de cada cola activa se incrementa en función de la proporción existente entre su peso y el de toda las colas activas del slice s ($w_{QueueNoEmpty}[s]$). En caso de no realizarse esta redistribución, el slice s no garantizaría su proporción asignada de los recursos, lo que comprometería las garantías inter-slice. Además, esta implementación permite que cuando una cola no necesita toda la asignación de recursos que se le garantiza, la porción no empleada pueda ser usada por otras colas del slice s de forma proporcional a sus pesos.

Por otro lado, el proceso de desencolado en el scheduling es un proceso regulado. Antes de realizar cualquier acción con los paquetes, se garantiza que la ocupación física del buffer del driver sea inferior a un umbral predefinido, en nuestro caso de 10 paquetes. En caso de que no

se cumpla, se espera antes de continuar (línea 3 del Alg.3 y línea 11 del Alg.4). Esta acción permite prevenir el desbordamiento del buffer del driver, asegurar un funcionamiento correcto del control de tasa (que en este testbed se implementa fuera del driver) y asegurar el cumplimiento de los requisitos de aislamiento entre slices y QoS. En este proyecto, no se puede acceder directamente a la ocupación del driver, por lo que se emplea la interfaz auxiliar como mecanismo alternativo de monitorización. Al comparar el número de tramas, y sus números de secuencia, entre los paquetes enviados por la interfaz principal y los recibidos en la interfaz auxiliar, podemos obtener el número de paquetes que se encuentran almacenados en el driver. Esta comparativa también es usada para detectar las pérdidas y las retransmisiones.

Por último, existen consideraciones respecto a la estimación del tiempo de transmisión (airtime). Habitualmente, en esquemas ADRR, el airtime requerido para enviar un paquete unicast a través de la interfaz aérea y recibir el reconocimiento correspondiente, teniendo en cuenta todas las retransmisiones necesarias, es calculado según la Ecuación 4.

$$Airtime = \overline{n_{Tx}} \cdot \left(\overline{T_{Backoff}} + T_{DIFS} + T_{DATA} + T_{SIFS} + T_{ACK}\right) \tag{4}$$

En esta, al propio tiempo de transmisión de los datos ($T_{DATA} = 1/R$, siendo R la tasa de transmisión de datos de la STA correspondiente), se le suman otros debido al protocolo CSMA/CA. Entre ellos se encuentra el tiempo medio de back-off antes de la transmisión ($\overline{T_{Backoff}}$), aunque no se tiene en cuenta en todos los trabajos que estiman con esta ecuación. En nuestro caso $\overline{T_{Backoff}} = (CW_{min}/2) \cdot Tslot = 15/2 \cdot 9s = 67'5s$, siendo CW_{min} el tamaño mínimo de la ventana de contención y T_{slot} el intervalo de tiempo que se espera antes de retransmitir, ya que los valores son los de las normas usadas en el proyecto, 802.11g/n/ac/a. También están el tiempo de espaciado distribuido entre tramas (T_{DIFS}, por sus siglas en inglés *Distributed Interframe Space*, 28µs para canales de 2.4GHz y 34µs para 5GHz), el tiempo de espaciado corto entre tramas (T_{SIFS}, por sus siglas en inglés *Short Interframe Space*, 10µs para canales de 2.4GHz y 16µs para 5GHz) y el tiempo de transmisión de la trama de control ACK (T_{ACK} (24Mbps)=28 µs).

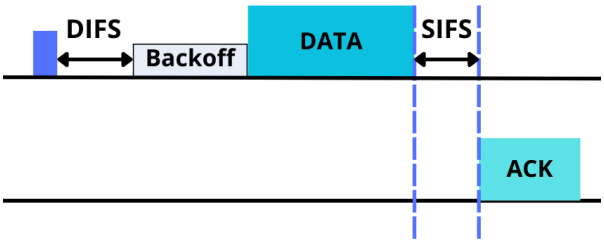


Figura 4: Esquema de transmisión de un paquete.

Para tener en cuenta todas las retransmisiones, en esta ecuación el número medio de transmisiones (la original más las retransmisiones), representado como $\overline{n_{Tx}}$, se obtiene como el inverso de la probabilidad de recepción del ACK ($p_{ACK}(R)$) cuando los paquetes se han transmitido utilizando una tasa física R, asociada a un MCS específico, ($\overline{n_{Tx}} = 1/p_{ACK}(R)$).

El problema es que tanto R como $p_{ACK}\,(R)$ provienen de estimaciones estadísticas basadas en la tasa que proporciona el mayor rendimiento efectivo y omiten datos relevantes referentes a cómo se han obtenido dichas estimaciones. Es por eso que, en este trabajo, en vez de usar la estadística, la asignación de tiempo de transmisión se determina en función del valor real de transmisión (que depende del índice MCS/tasa) de cada paquete.

De esta manera, se calcula el airtime requerido para enviar un paquete unicast a través de la interfaz aérea y recibir el reconocimiento correspondiente, sin tener en cuenta las retransmisiones, según la Ecuación 5 (es la Ecuación 4 con $\overline{n_{Tx}} = 1$, ya que solo cuenta la transmisión original).

$$Airtime = \overline{T_{Backoff}} + T_{DIFS} + T_{DATA} + T_{SIFS} + T_{ACK} \tag{5}$$

No se tienen en cuenta las retransmisiones porque en el momento en el que se estima el airtime de un paquete para ver si puede transmitirse se desconoce si va a ser retransmitido o cuántas veces va a serlo. Es por eso por lo que se calcula el tiempo necesario para la transmisión original, y una vez se haya enviado, se estimará el airtime real conociendo las retransmisiones que se han producido.

Para saber el número de retransmisiones, se emplea la interfaz auxiliar. Al capturar los paquetes en ella también nos permite conocer la tasa con la que han sido retransmitidos, ya que esta ha podido variar respecto a la del paquete original si se ha empleado el algoritmo de control de tasa.

El airtime extra se calcula multiplicando el obtenido en la Ecuación 5 por el número de retransmisiones que han ocurrido. Este tiempo se almacena en las variables $Deficit_{retries}[s]$ (donde se acumula el debido a las retransmisiones en todas las colas del slice s) y $Deficit_{retries}[s,i]$. Como se explicó en el apartado 3.2, se emplea el campo DSCP de los paquetes para señalar el slice y la cola a la que pertenecen. Accediendo a este campo en los paquetes retransmitidos conocemos cuales son el slice s y la cola i a las que debe acumularse este tiempo extra.

Estas variables se usan para actualizar los valores del contador de déficit del slice ($Deficit[s]$, línea 5 del Alg.3) y las colas ($Deficit[s,i]$, línea 4 del Alg.4) cuando comienza el turno del slice. Una vez son restados al déficit, se reinician a 0 (líneas 6 del Alg.3 y 5 del Alg.4) para continuar con el proceso de acumulación.

El sistema se adapta rápidamente a estas correcciones, debido a la regulación que se lleva a cabo con el control de la ocupación del buffer del driver. De este modo, se asegura que los slices y las colas reciban sus porciones de recursos garantizadas.

En cuanto al tiempo de transmisión de los datos (en µs), este se calcula mediante la expresión para los estándares basados en OFDM (Ecuación 6).

$$T_{DATA} = T_{PHY_{OH}} + T_{MPDU} = T_{PHY_{OH}} + T_{symbol}\left\lceil \frac{8 \cdot (MAC_{OH} + L)}{T_{symbol} \cdot R\,(MCS)} \right\rceil \tag{6}$$

Donde $T_{PHY_{OH}}$ es el tiempo de las cabeceras físicas y T_{MPDU} el tiempo asociado a la transmi-

sión de un MPDU. T_{MPDU} depende del número de símbolos OFDM necesarios para la transmisión total de los bits a la tasa física dada (T_{symbol} es el tiempo de símbolo, 4 µs), así como de la cabecera MAC ($MAC_{OH} = L_{service}\,(2B) + L_{HMAC}\,(24B) + L_{QoS}\,(2B) + L_{FCS}\,(4B)$) y la longitud de los datos recibidos desde el nivel LLC ($L = l_{LLC_{OH}}(8B) + l_{data}$, siendo $l_{LLC_{OH}}$ la longitud de la cabecera LLC y $l_{data}\,(IP\,(20B) + UDP\,(9B) + DATOS)$ la del paquete IP).

Finalmente, respecto a los valores del quantum y su orden de magnitud, estos pueden variar entre decenas de microsegundos y decenas de milisegundos. En caso de no realizar agregación, como el quantum se acumula en el déficit, no se observan diferencias significativas, aunque es cierto que valores más pequeños proporcionan mayor granularidad y equidad en la asignación de recursos. Sin embargo, cuando se agrega es importante que los valores de quantum sean suficientes para permitir el tiempo de transmisión de tramas agregadas. Es por ello que el orden de magnitud del quantum será elegido por la granularidad, así como las características del tráfico y los requisitos de calidad de servicio.

4.1.2 Pruebas y resultados

Las pruebas se han llevado a cabo en un escenario real con equipos en la banda de 5 GHz y usando el protocolo IEEE 802.11n.

A continuación, se presenta la configuración de las pruebas, las condiciones de evaluación y los resultados obtenidos.

4.1.2.1 Configuración de las pruebas

El escenario está compuesto por diversos componentes integrados en la red de gestión, donde se puede monitorizar el tráfico generado. Estos son el punto de acceso (AP), unos switches Ethernet, para construir las redes correspondientes al plano de datos y al plano de gestión, un mini PC, para la generación y clasificación de tráfico, y un conjunto de estaciones (STAs).

El punto de acceso (AP) opera en el espacio de usuario sobre un mini PC (*Asus NUC RNUC13A-NKI5, i5-1340P*) con Ubuntu 22 (*kernel 5.19.9*). Este AP emplea dos tarjetas inalámbricas USB *Alfa AWUS036ACH* y/o *Alfa AWUS036ACM*, equipadas con los chipsets *RTL8812au* y *MT7612u*, que admiten la inyección de tramas en modo monitor en la banda de 5 GHz. Las antenas de las tarjetas se encuentran separadas con un reflector para minimizar la correlación de canal.

El mini PC (*ASRock Core 100HT*) con Debian (*kernel 4.19.28*) se utiliza para la generación de tráfico y la clasificación de flujos IP. Para ello se emplea la herramienta iperf, complementada con la manipulación de la caché ARP y el uso de reglas iptables para clasificar el tráfico y enviarlo a las STAs según lo requerido. A través de las reglas de iptables, se asigna un valor específico al campo DSCP de los datagramas salientes hacia un puerto determinado, lo que

permite clasificar el tráfico en diferentes slices y colas directamente durante su generación (en iperf, el puerto se indica con el puerto con -p).

Por ejemplo, con la siguiente regla iptables se indica que el tráfico saliente del mini PC con puerto destino 5005 tendrá asignado el valor 3 (000011 en binario) en su campo DSCP. Luego, cuando este paquete sea recibido, se leerá este campo (000011), que indica que pertenece a la cola 3 (011) del slice 0 (000), como se explica en el apartado 3.2.

iptables -t mangle -A OUTPUT -p udp –dport 5005 -j DSCP –set-dscp 3

La tabla ARP se utiliza para vincular direcciones IP con las direcciones MAC de las STAs, ya que iperf requiere que el tráfico se genere hacia una dirección IP destino (con el parámetro -d).

Por último, las STAs también utilizan tarjetas inalámbricas USB *Alfa AWUS036ACH* y/o *Alfa AWUS036ACM*, conectadas a un mini PC (*Asus NUC RNUC13ANKI5, i5-1340P*) que ejecuta Ubuntu 22 (*kernel 5.19.9*). Para variar las condiciones de las STAs, algunas se encuentran directamente conectadas al mini PC, otras emplean un cable USB de 20 metros y otras tienen en sus antenas atenuadores fijos o programables variables para emular la movilidad en el sistema.

4.1.2.2 Procesado de los resultados

Los datos medidos en cada prueba son almacenados en ficheros que, posteriormente, son procesados en un programa de Matlab diseñado para obtener los parámetros que se quieren analizar.

En concreto, cada vez que llega un paquete se escribe en un fichero el tiempo en el que ha llegado, hacia que MAC (STA) va dirigido, cuál es su número de secuencia, si es una retransmisión, la MCS empleada, cuántos paquetes se están agregando (1 si no hay agregación), el tamaño AMSDU e IP, así como el slice y cola donde se va a almacenar. A partir de toda esta información se realiza el procesado y las operaciones pertinentes para obtener las gráficas de distribución del airtime, throughput, tasa de AMSDUs/MSDUs por segundos (total, sin retransmisiones o solo retransmisiones) y evolución de las MCS.

4.1.2.3 Evaluación y resultados

El escenario de esta prueba (cuyas variables se encuentran resumidas en la Tabla 2) se basa en 8 flujos de tráfico UDP con origen el mini PC y destino 8 estaciones (STAs). Los flujos de tráfico generados son uniformes y con 50 segundos de duración, pero con parámetros que cambian a lo largo del experimento. También es preciso resaltar que, aunque la tasa especificada en iperf sea la misma, al tratarse de paquetes con diferentes tamaños, los requisitos de tasa de paquetes también lo son, dando lugar a patrones de tráfico variados. Por otro lado, esta prueba se ha llevado a cabo en un entorno de laboratorio, por lo que las interferencia externas son mínimas, pero no nulas.

Los algoritmos usados tanto para el reparto inter-slice como intra-slice $s$on ADRR y las opciones de agregación no han sido habilitadas.

	Slice 0			**Slice 1**		**Slice 2**		
Quantum (Q_{max}) [µs]	3500			2500		4000		
	Cola 0	**Cola 1**		**Cola 0**	**Cola 1**	**Cola 0**	**Cola 1**	**Cola 2**
Peso (W_{cola})	50	50	50	30	70	50	30	20
STA	*STA 0*	*STA 1*	*STA 2*	*STA 3*	*STA 4*	*STA 5*	*STA 6*	*STA 7*
MCS	3	6	1	Variable	1	2	4	6
Tasa [Mbps]	1.4 (0-50 s)	1.4 (0-50 s)	1.4 (0-50 s)	1.4 (0-50 s)	1.4 (0-50 s)	1.4 (0-10 s) 0.5 (10-20 s) 1.4 (20-30 s) 0.5 (30-50 s)	1.4 (0-30 s) 0.5 (30-50 s)	1.4 (0-30 s) 0.5 (30-50 s)
Tamaño UDP (bytes)	250 B	1250 B	650 B	500 B	250 B	250 B	250 B	400 B
Paquetes/seg (*iperf*)	700 (0-50 s)	140 (0-50 s)	270 (0-50 s)	350 (0-50 s)	700 (0-50 s)	700 (0-10 s) 250 (10-20 s) 700 (20-30 s) 250 (30-50 s)	700 (0-30 s) 250 (30-50 s)	437 (0-30 s) 156 (30-50 s)
Airtime (µs)	281.5 µs	345.5 µs	625.5 µs	-	377.5 µs	313.5 µs	249.5 µs	241.5 µs

Tabla 2: Escenario de pruebas

Las STAs se encuentran repartidas en 3 slices con hasta 3 colas (clases de servicio) cada uno. Cada una de estas colas recibe tráfico dirigido a una STA distinta, exceptuando la cola 1 del slice 0 que recibe tráfico de las STAs 1 y 2. Así podemos verificar el rendimiento con múltiples flujos de datos de diferentes STAs asociados a una misma cola.

Además, todas las STAs se encuentran próximas al mini PC pero, para observar el impacto de las retransmisiones, las STA 1 y STA 3 se encuentran fuera del laboratorio, conectadas con un cable de 20 metros. Por otro lado, al estar configurado para operar en 802.11n/HT en la banda de 5 GHz, los valores de MCS admitidos varían de 0 a 15. Estos valores son distintos y fijos para todas las STAs, a excepción de la STA 3 que tiene el control de tasa (ONOE) activo. Esta se mantendrá en movimiento para observar el efecto de las variabilidad de las condiciones del entorno.

Antes de pasar a analizar los resultados, se calcula las asignaciones garantizadas para cada slice y cola. Recordemos que los valores de quantum para cada slice ($Q[s]$) son 3500, 2500 y 4000 (ver Tabla 2), por lo que siguiendo la Ecuación 1 la proporción mínima garantizada para cada slice es:

$\sum_{j=0}^{N_{slices}} Q\,[j] = 3500 + 2500 + 4000 = 10000$

Para el slice 0: $Q\,[0]$ / $\sum_{j=0}^{N_{slices}} Q\,[j] = 3500/10000 \rightarrow 35\,\%$

Para el slice 1: $Q\,[1]$ / $\sum_{j=0}^{N_{slices}} Q\,[j] = 2500/10000 \rightarrow 25\,\%$

Para el slice 2: $Q\,[2]$ / $\sum_{j=0}^{N_{slices}} Q\,[j] = 4000/10000 \rightarrow 40\,\%$

Mientras que, para las colas, siguiendo la Ecuación 2, el porcentaje de asignación mínimo respecto al total de su slice es:

Para el slice 0: $\sum_{j=0}^{N_{queue}[0]} W\,[0,j] = 50 + 50 = 100$

Cola 0: $50/100 \rightarrow 50\,\%$; Cola 1: $50/100 \rightarrow 50\,\%$

Para el slice 1: $\sum_{j=0}^{N_{queue}[1]} W\,[1,j] = 30 + 70 = 100$

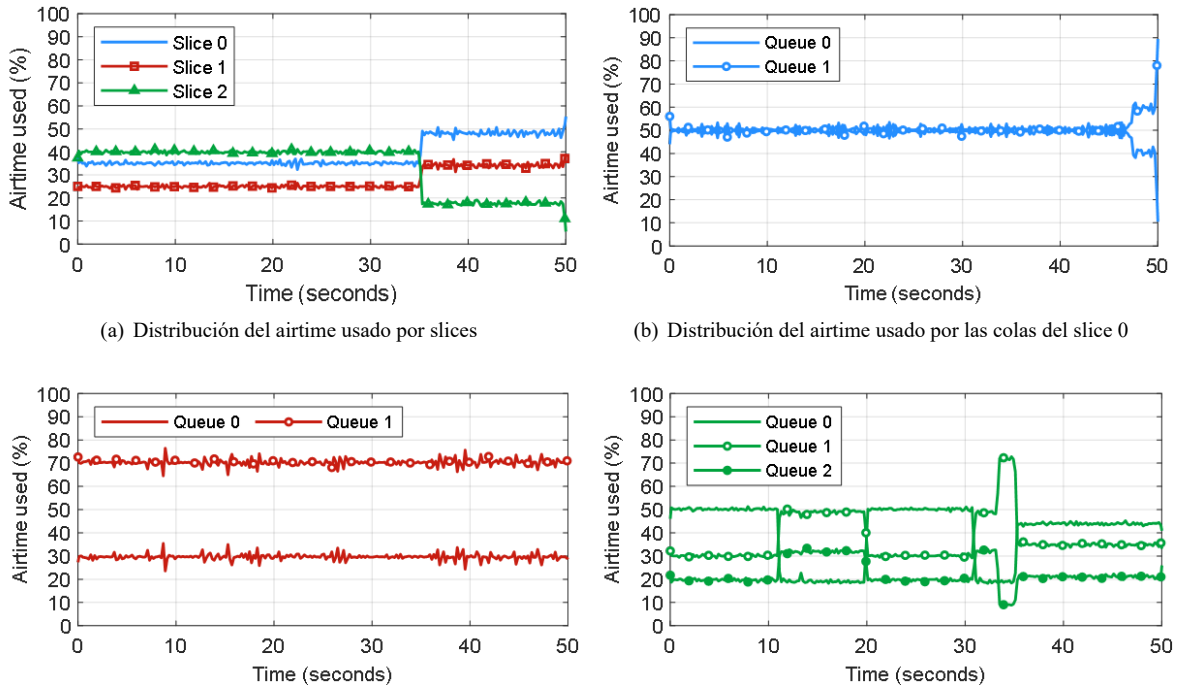
Cola 0: $30/100 \rightarrow 30\,\%$; Cola 1: $70/100 \rightarrow 70\,\%$

Para el slice 2: $\sum_{j=0}^{N_{queue}[2]} W\,[2,j] = 50 + 30 + 20 = 100$

Cola 0: $50/100 \rightarrow 50\,\%$; Cola 1: $30/100 \rightarrow 30\,\%$; Cola 2: $20/100 \rightarrow 20\,\%$

Conocidos estos datos, se muestran los resultados promediados en intervalos de 200ms del porcentaje de airtime ocupado por cada slices (Figura 5a) y, dentro de cada slice, por cada cola (Figura 5b, c, d), así como la tasa efectiva de trasmisión de paquetes (Figura 6) y el throughput (Figura 7). Además, se incluyen los paquetes retransmitidos por segundo (Figura 8) y la variación de la MCS de la STA con control de tasa activo (Figura 9), para poder ver el efecto de ambos. Con estos se pretende mostrar el correcto funcionamiento del algoritmo.

A continuación, se explican los resultados a través del conjunto de datos expuestos en las figuras.



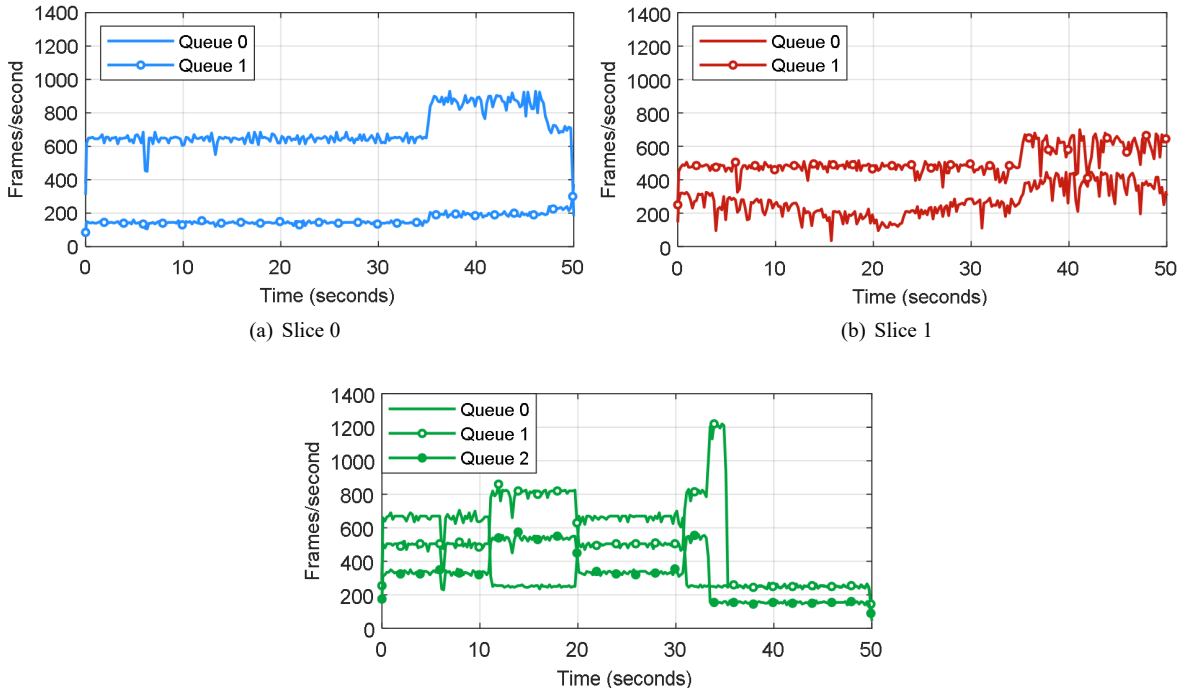
(a) Distribución del airtime usado por slices

(b) Distribución del airtime usado por las colas del slice 0

(c) Distribución del airtime usado por las colas del slice 1

(d) Distribución del airtime usado por las colas del slice 2

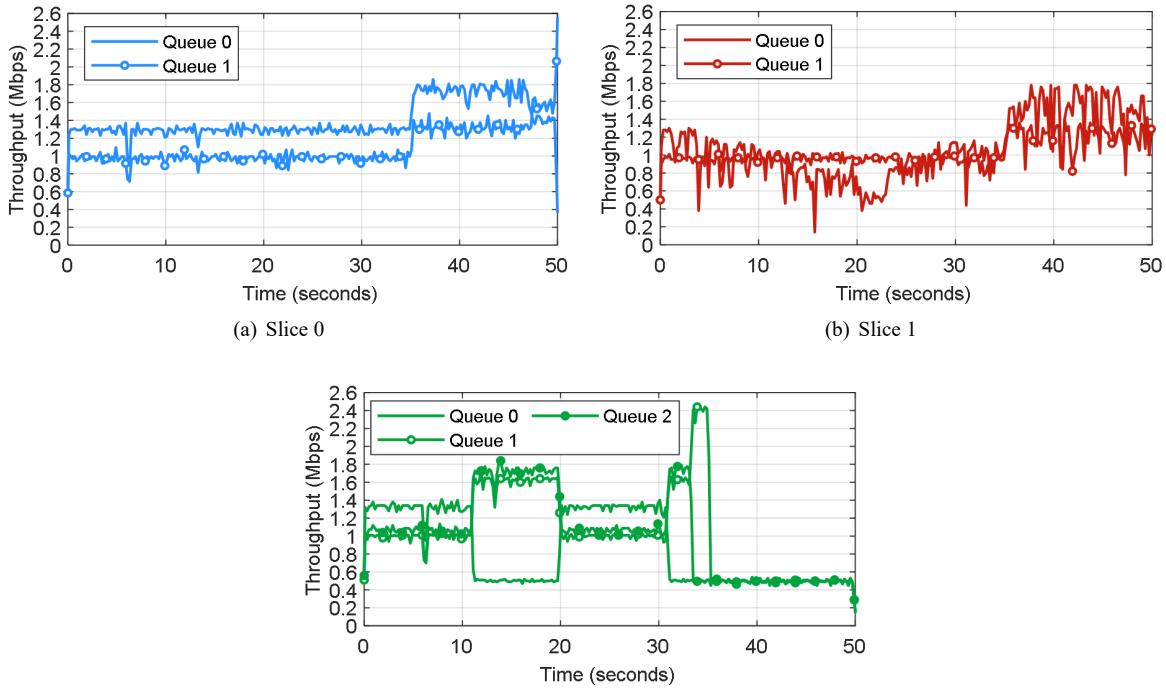
Figura 5: Distribución del airtime entre los slices y las colas.



(a) Slice 0 (b) Slice 1

(c) Slice 2

Figura 6: Paquetes por segundo transmitidos correctamente en cada las colas de cada slice.



(a) Slice 0 (b) Slice 1

(c) Slice 2

Figura 7: Throughtput por colas de cada slice.

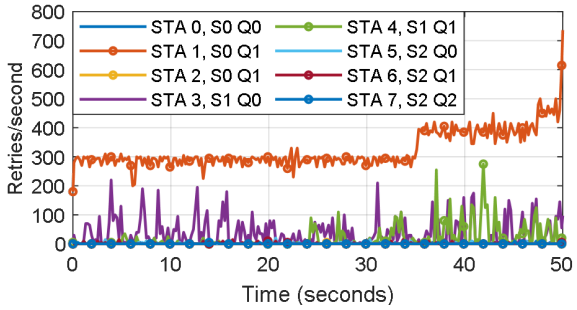


Figura 8: Tramas retransmitidas de cada cola de los slices 1, 2 y 3.

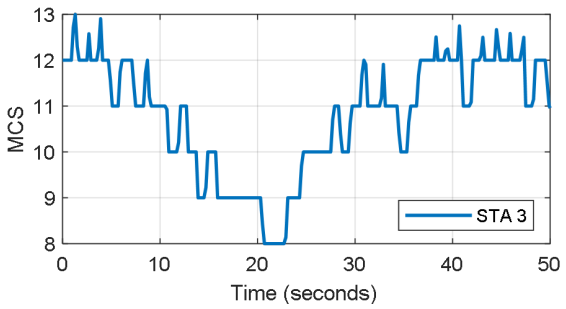


Figura 9: Variaciones de MCS en la STA 3.

En el primer intervalo de tiempo (0 a 10 segundos) hay una situación de congestión, es decir, las demandas de tráfico superan los recursos de airtime disponibles. Si observamos la Figura 5 vemos que la distribución del airtime es exactamente la garantizada, tanto para los slices (35 %/25 %/40 %, Fig. 5a) como para sus colas (50 %/50 %, Fig. 5b; , 30 %/70 %, Fig. 5c; 50 %/30 %/20 %, Fig. 5d). Esto muestra que, en estas condiciones, se garantiza de manera efectiva el aislamiento.

Cabe destacar que esta proporción se está dando correctamente a pesar de las cuantiosas retransmisiones que se están produciendo en las STA 1 y STA 3 (Figura 8). Estas retransmisiones son unas 300 por segundo en el caso de la STA 1 (una de las STAs de la cola 1 del slice 0) y con picos que alcanzan las 200 por segundo en el caso de la STA 3.

En cuanto a los paquetes transmitidos correctamente (Figura 6), vemos la situación de congestión comentada. Muchas de las colas están transmitiendo por debajo de las tasas de generación de paquetes (ver Tabla 2), por lo que estos se están acumulando. Por ejemplo, la cola 0 del slice 0 (Fig.5a) transmite 650 paq/s, lo que se aproxima a la tasa de generación (700 paq/s), pero sigue siendo inferior. Esto se observa todavía más en la cola 1, donde su tasa de generación es 410 paq/s (140 paq/s de la STA1 y 270 paq/s de la STA2), pero transmite menos de 200 paq/s, más las retransmisiones comentadas de la STA1. También se ve en el slice 1, donde las tasas de transmisión son en torno a 300 paq/s y 490 paq/s (Fig.5b), mientras que las tasas de generación son 350 paq/s y 700 paq/s, respectivamente. Al igual que en el slice 2, donde sus colas tienen tasas de transmisión de 670 paq/s, 470 paq/s y 325 paq/s, frente a las tasas de generación de 700 paq/s, 700 paq/s y 437 paq/s, respectivamente. Lo mismo puede observarse en la Figura 7, ya que en este intervalo todas las tasas de generación son de 1’4Mbps, pero vemos que ninguna de las colas tiene esa tasa de transmisión.

En el siguiente intervalo (10 a 20 segundos), es relevante observar el comportamiento del slice 2. La tasa de generación de la cola 0 decae a 0’5Mbps o, lo que es lo mismo, 250 paq/s. Como esta cola había acumulado paquetes en el periodo anterior, pero no demasiados, ya que la tasa de transmisión efectiva era poco inferior a la de generación, vemos que la caída de esta tasa se refleja un segundo después (Figura 6d, Figura 7d). En el segundo 11, la cola 0 ha podido vaciar su cola y ahora transmite todos los paquetes que le llegan. Esto hace que pueda ceder parte de

su asignación de recursos.

En la Figura 5d, vemos que la cola 0 usa el $19\,\%$ del airtime, aunque su asignación era del $50\,\%$. El $31\,\%$ que no usa, debe ser repartido entre las otras colas activas del slice.

De este modo, teóricamente la cola 0 cede $18'6\,\%(31\,\%\cdot\frac{30}{50})$ a la cola 1, haciendo que su asignación durante este periodo sea del $48'6\,\%(30\,\% + 18'6\,\%)$, y $12'4\,\%(31\,\%\cdot\frac{20}{50})$ a la cola 2, por lo que usa un $32'4\,\%(20\,\% + 12'4\,\%)$. Volviendo de nuevo a las gráfica, vemos que estos porcentajes se cumplen.

En el periodo de tiempo entre 20 y 30 segundos, la tasa de generación de la cola 0 del slice 2 vuelve a subir a 1'4Mbps, lo que hace vuelva a necesitar toda su asignación de recursos y no pueda ceder. Observando la Figura 5d se verifica que la recuperación de los recursos es inmediata, volviendo a tener el mismo escenario que en el periodo de 0 a 10 segundos.

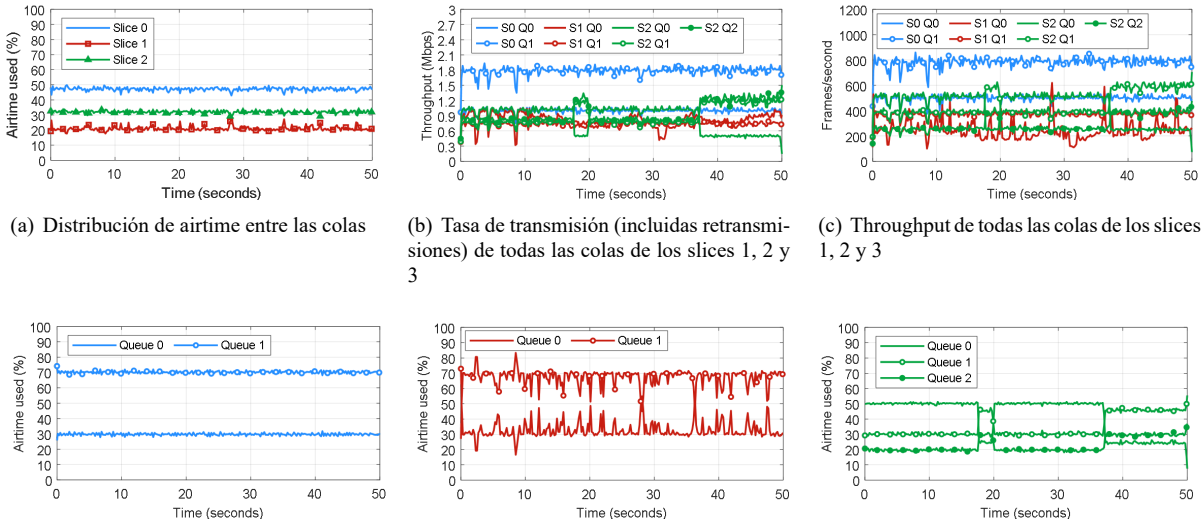
El último periodo en el hay cambios en la tasa de generación es de 30 a 50 segundos, donde esta pasa de 1'4Mbps a 0'5Mbps para las tres colas del slice 0. Al igual que en la anterior bajada de tasa, la cola 0 tenía algunos paquetes acumulados, pero tras vaciar su cola, en el segundo 31 es capaz de transmitir todo lo que recibe con un uso del airtime del $19\,\%$. En ese momento, se cede el restante a las colas 1 y 2 que, como ocurría en el intervalo de 10 a 20 segundos, emplean un $48'6\,\%$ y un $32'4\,\%$, respectivamente. Sin embargo, como estas colas también han bajado su tasa de generación, el número de paquetes por segundo que están recibiendo es inferior, y en el segundo 33 la cola 1 consigue vaciar su cola y transmitir todos los paquetes que le van llegando. En ese momento, ya solo necesita usar el $9\,\%$ del airtime, por lo que el $11\,\%$ sobrante, respecto a su $20\,\%$ garantizado, es cedido a la cola 2. Es decir, la cola 2 va a usar su $30\,\%$ garantizado, el $31\,\%$ cedido por la cola 0 y el $9\,\%$ cedido por la cola 1, lo que hace un total del $70\,\%$ del airtime del slice 2. Al llegar al segundo 35, se observa que la cola 2 también ha conseguido vaciarse, ya que la tasa de transmisión pasa a ser de 0'5Mbps. De este modo, el slice 2 ya no necesita el $40\,\%$ del airtime que tiene garantizado y cede el restante ($40\,\% - 18\,\% = 22\,\%$) a los otros dos slices.

Ahora el slice 0 emplea el $47'8\,\%(35\,\% + 22\,\%\cdot(\frac{3500}{3500+2500})$ y el slice 1 el $34'2\,\%(25\,\% + 22\,\%\cdot\frac{2500}{3500+2500})$. Esto no supone una nueva asignación de recursos entre las colas dentro de los slices, ya las colas de los slices 0 y 1 necesitan la porción de asignación garantizada en su totalidad. Esto es así hasta el segundo 47, donde la cola 0 del slice 0 vacía su cola y transmite a 1'4Mbps, empleando el $40\,\%$ de los recursos, es decir, cediendo el $10\,\%$ restante a la otra cola del slice.

Un aspecto importante que destacar de esta prueba es que muestra que el esquema estima de manera efectiva el consumo de airtime para cada cola, incluso cuando varía la tasa o hay un número significativo de retransmisiones. En esos momentos, se sigue apreciando que la asignación se mantiene estable. Además, recordemos que en una de las colas (cola 1 del slice 1) había tráfico dirigido a dos STAs, con uso de MCS y tamaños de paquetes diferentes y unas condiciones de canal muy distintas. Sin embargo, la estimación de consumo de la cola es

precisa, las correcciones de retransmisión funcionan de manera eficiente y siempre se garantiza el aislamiento entre colas.

Ahora se repite el experimento, pero sin aplicar la corrección por retransmisiones para poder ver el efecto que produce el error en la estimación del airtime (Figura 10).



(a) Distribución de airtime entre las colas

(b) Tasa de transmisión (incluidas retransmisiones) de todas las colas de los slices 1, 2 y 3

(c) Throughput de todas las colas de los slices 1, 2 y 3

(d) Distribución de airtime entre las colas del slice 0

(e) Distribución de airtime entre las colas del slice 1

(f) Distribución de airtime entre las colas del slice 2

Figura 10: Resultado del experimento de la Tabla 2 si no se aplican las correcciones por retransmisiones.

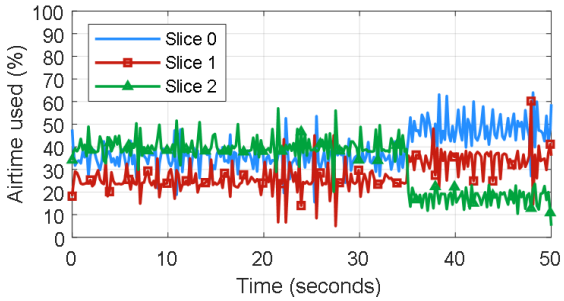
En la Figura 10a se ve que, efectivamente, al estar haciendo la estimación del airtime como si no hubiera retransmisiones, cuando sí que las hay, el cálculo del airtime no es correcto y las colas y slices con retransmisiones están consumiendo más de lo que les tocaría. En el slice 0, el reparto debería ser 50/50 pero al tener la cola 1 muchas retransmisiones (recordemos que tenía la STA1 con MCS alta y fuera del laboratorio) y estas no estar reflejadas, hace uso del 70 % de los recursos, "robando" 20 % a la otra cola de su asignación garantizada. Observando las tasas de transmisión (Figura 10b,c) vemos que no se trata de una cesión, porque la cola 0 transmite cerca de 500 paq/s y la tasa de generación es de 700 paq/s, por lo que sí que necesita los recursos. En el slice 2 vemos que el reparto no es constante. Esto se debe a que la cola 1 tiene la STA3, aquella que se encuentra fuera del laboratorio y en movimiento, por lo que tiene un gran número de retransmisiones que no son contempladas.

En el reparto inter-slice también se ve reflejado, ya que el slice 0 emplea el 48 % de los recursos, cuando solo le corresponde el 35 %. El slice 2 debería usar el 40 % del airtime, pero en cambio solo emplea el 31 %, aunque se ve que tiene acumulación de paquetes porque la tasa de transmisión en todas sus colas es muy inferior a la de generación.

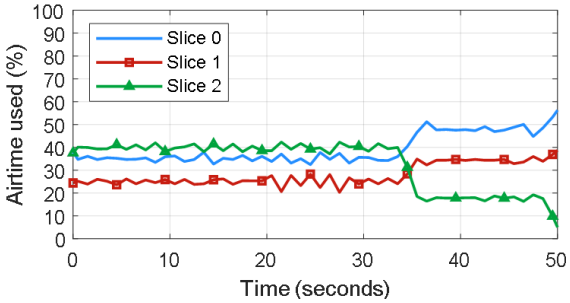
Esto muestra la importancia de tener en cuenta las retransmisiones a la hora de estimar el airtime, ya que es crítico para hacer un reparto justo y eficiente. También demuestra que la forma en la que se tienen en cuenta en este trabajo funciona correctamente.

Por último, para mostrar el efecto del orden de magnitud del quantum, se repite el experimento con quantums superiores e inferiores.

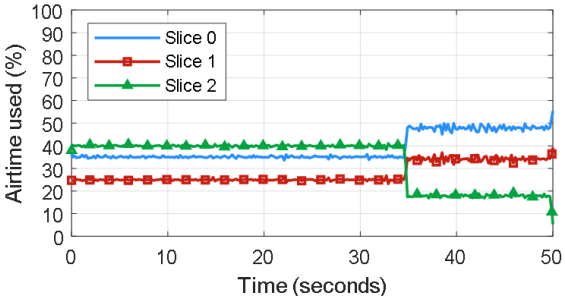
Recordemos que el experimento se ha realizado con quantums del orden de 1000µs (3500, 2500, 4000), en la Figura 11 se observa el efecto de subir o bajar en uno este orden.



(a) Distribución del airtime por slice con quantums del orden de 10ms. Promediado temporal de 200ms.



(b) Distribución del airtime por slice con quantums del orden de 10ms. Promediado temporal de 1s.



(c) Distribución del airtime por slice con quantums del orden de 100µs. Promediado temporal de 200ms.

Figura 11: Resultado del experimento de la Tabla 2 con órdenes de quantum mayores y menores.

En la Figura 11a se ha subido el orden de magnitud de los quantums a 10ms (35000 µs, 25000 µs, 40000 µs). Esto ha supuesto la pérdida de la granularidad. En media si podemos ver que el porcentaje de ocupación es 35 %, 25 % y 40 %, por lo que el reparto de recursos es correcto, pero incluso si aumentamos el promediado a 1s (Figura 11b) se sigue viendo que esta ocupación no es tan constante como en el experimento original.

Por el contrario, en la Figura 11c, observamos unos valores constantes en la ocupación del airtime por parte de los slices, pero bastante similar al resultado obtenido para los quantum del orden de 100 µs (Figura 5a). Esto muestra que un orden menor del quantum supone una mayor granularidad, por lo que, en caso de no realizar agregación, se considera mejor usar quantum menores a 1ms.

4.2 Redistribución del quantum sobrante

En este apartado se presentan propuestas de algoritmos para la redistribución del quantum sobrante cuando se usa ADRR. En el apartado 4.1.1 se mostró que, en este esquema, cuando una cola no emplea toda su asignación de recursos, y hay otras en situación de saturación, se realiza un reparto de esos recursos sobrantes proporcional a los quantum/pesos de las necesitadas. Con estos algoritmos, se pretende controlar ese reparto para que cumplan, por ejemplo, un trato de prioridad en alguna de las colas o trate de igualar sus satisfacciones en cuanto a tasa (relación entre la tasa conseguida y la demandada).

Por supuesto, esto solo tiene sentido cuando una cola no necesita todos sus recursos asignados y hay otras que necesitan más de los garantizados. De igual forma, para que se produzca esta redistribución es necesario que el número de colas sea superior a dos, ya que, en caso contrario, todo lo que no necesite una se dará a la otra de forma natural.

Para llevar a cabo esto no se modifican los valores nominales establecidos en configuración ($Q_{nom}\,[s]$, $W_{nom}\,[s,i]$), no se realizan aumentos o disminuciones del quantum mediante factores de ajuste o modificaciones por pasos, ni se emplea un criterio de optimización matemática. Lo que se hace son incrementos específicos en el quantum de un slice, que corresponde a su vez a la disminución de los quantum, en su misma cantidad, de los slice que ceden recursos, haciendo que la suma de los quantum siempre se mantenga constante. Estas cesiones se hacen en intervalos de tiempo configurables, que en nuestra caso es 1 segundo. Durante este tiempo, se recogen medidas que permiten controlar la cesión, adaptándola a los objetivos requeridos.

Este esquema de distribución se establece a nivel intra-slice e inter-slice, y se determinan los objetivos de la redistribución para conseguir igualar los grados de satisfacción de las colas insatisfechas y establecer un reparto prioritario de unas colas respecto a otras. Es importante destacar que estos algoritmos funcionan de manera independiente, por lo que el esquema que se siga a nivel intra-slice puede ser distinto al inter-slice. Además, estos modifican los valores de $W[s,i]$ y $Q[s]$, que permanecen constantes durante el proceso de slicing en cada intervalo T.

En el Algoritmo 5 se muestra la implementación de la redistribución a nivel intra-slice, cuyas variables quedan explicadas en la Tabla 3, mientras que, más adelante, en el Algoritmo 6 y la Tabla 4 mostrarán la redistribución a de nivel inter-slice.

En ambos casos, hay que decir que el grado de satisfacción se especifica considerando como límite para la demanda de tasa, el parámetro MBR (Maximum Bit Rate).

Así pues, en el Algoritmo 5, $MBR\,[s,i]$ determina la tasa máxima que el sistema garantiza a la cola i del slice s. Este valor viene determinado por configuración, conforme a un SLA (*Service Level Agreement*). Sin embargo, puede haber ocasiones donde la tasa que realmente esté demandando el flujo, $R_{demanded}\,[s,i]$, sea superior a este límite. En ese caso, la demanda

quedará limitada al $MBR\,[s,i]$, y solo se actuará en función del $R_{demanded}\,[s,i]$ si el resto de las colas están satisfechas, con la finalidad de emplear al máximo el canal.

Variable	Descripción
T	Intervalo de actualización
$airtime_T[s,i]$	Airtime estimada de todos los paquetes servidos por la cola i del slice s en el intervalo T, donde $n_{retries}$ es el número de retransmisiones. $\sum_{\forall k \in T} \left(1 + n_{retries}[s,i,k] \cdot airtime[s,i,k]\right)$
$airtime_T[s]$	Airtime estimado de todas las colas del slice s. $\sum_{\forall i \in s} airtime_T[s,i]$
$airtime_T$	Airtime estimado para todas los slices. $\sum_{\forall s} airtime_T[s]$
$MBR[s,i]$	Tasa Máxima de Bits (Maximum Bit Rate) de la cola i del slice s.
$R_{demanded}[s,i]$	Tasa de datos en bps a transmitir de la cola i.
$R_{achieved}[s,i]$	Tasa de datos efectiva en bps (nivel IP) de la cola i.
$S_{unsatisfied}[s]$	Grupo de colas insatisfechas del slice s.
$S_{underutilized}[s]$	Grupo de colas del slice s que tienen pesos para ceder.
$W_{nom}[s,i]$	Peso nominal de la cola i del slice s (configurado por el controlador).
$W[s,i]$	Peso usado por la cola i del slice s en el intervalo T. $W_{nom}[s,i]$ por defecto. $W_{nom}[s,i]$ - $W_{ceded}[s,i]$ si la cola i da peso a otras colas insatisfechas del mismo slice $s$lice s. $W_{nom}[s,i]$ + $W_{added}[s,i]$ si la cola i recibe peso de colas satisfechas del mismo slice s.
$W_{total}[s]$	Suma de los pesos de las colas del slice s. Esto se mantiene constante en todos los intervalos de tiempo. $\sum_{\forall i \in s} W_{nom}[s,i]$
$W_{ceded}[s,i,j]$	Peso cedido por la cola i a la cola j del slice s. Tiene el mismo valor que $W_{added}[s,j,i]$
$W_{ceded}[s,i]$	Peso total cedido por la cola i del slice s. $\sum_{\forall j \in S_{unsatisfied[s]}} W_{ceded}[s,i,j]$
$W_{added}[s,i,j]$	Peso recibido por la cola i de la cola j del slice s. Mismo valor que $W_{ceded}[s,j,i]$
$W_{added}[s,i]$	Peso total recibido por la cola i del slice s. $\sum_{\forall j \in S_{underutilized[s]}} W_{added}[s,i,j]$
$W^{exp}_{ceded}[s,i]$	Peso extra esperado de la cola i para ser cedido.
$W^{need}_{added}[s,i]$	Peso requerido por la cola i del slice s para estar satisfecha.
$\frac{W[s,i]}{W_{total}[s]}$	Porcentaje de airtime esperado (en el AP) usado por la cola i del slice s respecto a la asignación de su slice.
$DS[s,i]$	Grado de satisfacción (Degree of Satisfaction) de la cola i del slice s (0=totalmente insatisfecho, 1=totalmente satisfecho). $R_{achieved}[s,i]/R_{demanded}[s,i]$ si $R_{demanded}[s,i] < MBR[s,i]$ $R_{achieved}[s,i]/MBR[s,i]$ si $R_{demanded}[s,i] > MBR[s,i]$
$IDS^{exp}[s,i]$	El incremento de $DS[s,i]$ de la cola i del slice s esperado para el incremento de una unidad de peso.

Tabla 3: Variables de la redistribución Intra-Slice

Decimos que una cola está satisfecha cuando su tasa de transmisión, $R_{archived}\,[s,i]$ es igual a la tasa requerida, el menor entre $MBR\,[s,i]$ o $R_{demanded}\,[s,i]$. Es decir, transmite todo lo que

quiere.

De este modo, se calcula el grado de satisfacción con la Ecuación 7.

$$DS\left[s,i\right] = \begin{cases} \frac{R_{archived}[s,i]}{R_{demanded}[s,i]}, & \text{si } R_{demanded}\left[s,i\right] < MBR\left[s,i\right] \\ \\ \frac{R_{archived}[s,i]}{MBR[s,i]}, & \text{si } R_{demanded}\left[s,i\right] \geq MBR\left[s,i\right] \end{cases} \tag{7}$$

Solo en caso de que todas las colas estés satisfechas, es decir, $DS\left[s,i\right] = 1$ para todas las colas del slice s, se recalcularán los $DS\left[s,i\right]$ con los valores reales demandados si estos son superiores al límite (Ecuación 8).

$$DS\left[s,i\right] = \frac{R_{archived}\left[s,i\right]}{R_{demanded}\left[s,i\right]} \tag{8}$$

Con la finalidad de realizar correctamente el algoritmo, en función del parámetro descrito se establecen dos grupos de colas.

$S_{unsatisfied}\left[s\right]$ recoge todas las colas insatisfechas del slice s, $DS\left[s,i\right] < 1$, mientras que $S_{underutilized}\left[s\right]$ agrupa a todas sus colas infrautilizadas. Decimos que una cola está infrautilizada cuando está satisfecha, $DS\left[s\right] = 1$, y el porcentaje de tiempo de transmisión que ocupa, con respecto al que le corresponde, es tal que es posible ceder parte de sus recursos a otras colas del mismo slice s, manteniendo un umbral α. Esto sucede cuando la demanda de los recursos es inferior a la garantizada o la cola está inactiva.

Recordemos (Ecuación 3) que los recursos son repartidos de forma proporcional a la relación entre $W\left[s,i\right]$ y la suma total de los pesos de todas las colas activas del slice s, $W_{total}\left[s\right]$. De este modo, al modificar el valor de $W\left[s,i\right]$, estamos cambiando el quantum asociado a la cola, $Q\left[s,i\right]$.

El objetivo del algoritmo va a ser modificar $W\left[s,i\right]$ para aumentar o disminuir su peso respecto al nominal, $W_{nom}\left[s,i\right]$, que es su valor por defecto.

Una cola i insatisfecha puede aumentar su $W\left[s,i\right]$ siguiendo la Ecuación 9. En ella, el peso que ha recibido, $W_{added}\left[s,i\right]$, es la suma del que ha recibido de todas las colas con recursos infrautilizados, $S_{underutilized}\left[s\right]$. Mientras que el peso que le ha sido cedido únicamente por la cola j se almacena en la variable $W_{added}\left[s,i,j\right]$.

$$W\left[s,i\right] = W_{nom}\left[s,i\right] + W_{added}\left[s,i\right] \tag{9}$$

Por otro lado, el peso de una cola i después de ceder disminuye según la Ecuación 10. $W_{ceded}\left[s,i\right]$ es el peso total que ha cedido a todas las colas insatisfechas, $S_{unsatisfied}\left[s\right]$, y el peso que ha

cedido únicamente a la cola j queda guardado en $W_{ceded}\,[s,i,j]$.

$$W\,[s,i] = W_{nom}\,[s,i] - W_{ceded}\,[s,i] \tag{10}$$

La cantidad de peso que la cola i cede a la cola j es la misma que la cantidad que la cola j recibe de la cola i, por lo que $W_{added}\,[s,i,j] = W_{ceded}\,[s,j,i]$. Es decir, todos los incrementos de peso recibidos provienen de cesiones, lo que implica que la suma de toda los $W\,[s,i]$ siempre se mantiene constante.

Según lo descrito, puede verse que una cola no puede ceder y recibir peso a la vez, ya que es imposible que una cola esté insatisfecha e infrautilizada al mismo tiempo, en caso de necesitar recursos directamente no cedería. No obstante, es posible definir el valor de $W\,[s,i]$ de forma general con la Ecuación 11, siempre teniendo en cuenta que no es posible la coexistencia de $W_{ceded}\,[s,i] > 0$ y $W_{added}\,[s,i] > 0$.

$$W\,[s,i] = W_{nom}\,[s,i] + W_{added}\,[s,i] - W_{ceded}\,[s,i] \tag{11}$$

Por último, se necesita conocer el aumento que supone en el grado de satisfacción de la cola i el incremento en una unidad de su peso ($IDS^{exp}\,[s,i]$). Este se estima a partir del grado de satisfacción medido en el intervalo de tiempo anterior, $DS\,[s,i]$, al usar el peso $W\,[s,i]$ (Ecuación 12).

$$IDS^{exp}\,[s,i] = \frac{DS\,[s,i]}{W\,[s,i]} \tag{12}$$

A partir de todo lo comentado, el algoritmo de redistribución se divide en 4 fases: identificación inicial de las colas insatisfechas, estimación del grado de satisfacción para el valor nominal del peso de la cola, identificación de las colas infrautilizadas y redistribución de los pesos siguiendo alguna de las dos opciones configuradas.

Algoritmo 5 Redistribución Intra-Slice del quantum no usado

#Identificar colas insatisfechas del slice *s*

▷ $DS[s,i]$: Grado de satisfacción de la cola i del slice s

▷ $IDS^{exp}[s,i]$: El aumento de *DS* esperado para el incremento de una unidad de peso

for cada cola $i \in$ slice s **do**
  $DS[s,i] = \max\left(\frac{R_{achieved}[s,i]}{R_{demanded}[s,i]}, \frac{R_{achieved}[s,i]}{MBR[s,i]}\right)$
  if $DS[s,i] < 1$ **then**
    Añade i al grupo de $S_{unsatisfied}[s]$ y guarda el par ($W[s,i]$, $DS[s,i]$)
  end if
  $IDS^{exp}[s,i] = DS[s,i]/W[s,i]$
  $ds[s,i] = DS[s,i]$ ▷ Variable que guarda la tasa de satisfacción esperada
end for

#Estima *DS* si el peso fuera el nominal

▷ $S_{ceded}[s,i]$: grupo de colas que han cedido recursos a la cola $i \in s$

▷ $W_{ceded}[s,i,j] = W_{added}[s,j,i]$: recursos cedidos por la cola i a la $j \in s$

▷ $W_{added}[s,i] = \sum_{\forall j \in S_{ceded}[s,i]} W_{ceded}[s,j,i]$

▷ $W[s,i] = W_{nom}[s,i] + W_{added}[s,i] - W_{ceded}[s,i]$

for cada cola $i \in$ slice s cuyo $W[s,i] > W_{nom}[s,i]$ **do**
  for cada cola $j \in S_{ceded}[s,i]$ **do**
    $W[s,j]$ += $W_{ceded}[s,j,i]$
    $ds[s,j]$ = min($ds[s,j] + IDS^{exp}[s,j]$*$W_{ceded}[s,j,i]$, 1)
    $W[s,i]$ -= $W_{added}[s,i,j]$
    $ds[s,i]$ = max($ds[s,i] - IDS^{exp}[s,i]$*$W_{added}[s,i,j]$, 0)
  end for
  if $ds[s,i] < 1$ **then** Añade la cola i al grupo $S_{unsatisfied}[s]$
  else Elimina a la cola i del grupo $S_{unsatisfied}[s]$
end for

▷ Los pesos han sido reseteados, por lo que *W[s,i]* = W_{nom}*[s,i]* $\forall i \in s$
  W_{ceded}*[s,i,j]*=W_{added}*[s,i,j]*=W_{ceded}*[s,i]*=W_{added}*[s,i]* = 0 $\forall i, j \in s$

#Identificar colas infrautilizadas del slice *s*

▷ $excess[s,i]$: porcentaje de la porción de recursos que teóricamente corresponden a la cola i y no están ocupados

for cada cola $i \notin S_{unsatisfied}[s]$ **do**
  $excess[s,i] = \left(\frac{W[s,i]}{W_{total}[s]} - \frac{airtime_T[s,i]}{airtime_T[s]}\right) / \left(\frac{W[s,i]}{W_{total}[s]}\right)$
  if $excess[s,i] > \alpha$ **then**
    Añade a la cola i al grupo $S_{underutilized}[s]$
    $W^{exp}_{ceded}[s,i]$ = ($excess[s,i]$-α)*$W[s,i]$ ▷ Peso extra para ceder
  end if
end for

#Opción Redistribuir pesos para conseguir una tasa de satisfacción similar

$w_{ceded} = \sum_{\forall i \in S_{underutilized}[s]} W^{exp}_{ceded}[s,i]$

▷ Distribuir w_{ceded} entre las colas $\in S_{unsatisfied}[s]$

while $w_{ceded} > 0$ **do**
  Encuentra la cola $j \in S_{unsatisfied}$[s] con menor $ds[s,j]$
  Encuentra la cola $i \in S_{underutilized}[s]$ con mayor $W^{exp}_{ceded}[s,i]$
  if $\lceil W_{nom}[s,i] * \beta \rceil < w_{ceded}$ **then** $step = \lceil W_{nom}[s,i] * \beta \rceil$
  else $step = w_{ceded}$
  w_{ceded} -= $step$
  $W^{exp}_{ceded}[s,i]$ -= $step$
  $W_{ceded}[s,i]$ += $step$ and $W_{ceded}[s,i,j]$ += $step$
  $W_{added}[s,j]$+= $step$ and $W_{added}[s,j,i]$ += $step$
  $W[s,i]$ -= $step$
  $W[s,j]$ += $step$
  $ds[s,j]$ += $IDS^{exp}[s,j]$ * step
  if $ds[s,j] \geq 1$ **then** Elimina la cola j del grupo $S_{unsatisfied}[s]$
  if $S_{unsatisfied}[s] = \emptyset$ **then**
    if $R_{demanded}[s,j] > MBR[s,j]$ para alguna j **then**
      Redefine $S_{unsatisfied}[s]$ con $R_{demanded}$ real (*repite líneas 1-7*)
      if $S_{unsatisfied}[s]$ sigue vacío **then** Salir de *while*
    end if
  end if
end while

```
#Opción Redistribuye el peso siguiendo una prioridad
47:  Calcula el W requerido por cada cola j para estar satisfecha W^need_added[s, j]
48: for cada cola i ∈ S_underutilized[s] ordenada de forma descendiente según su W^exp_ceded[s, i] do
49:     while W^exp_ceded[s, i] > 0 do
50:         Iterar las colas del grupo S_unsatisfied[s] en orden de prioridad
51:         if W^need_added[s, j] > W^exp_ceded[s, i] then
52:             W^need_added[s, j] -= W^exp_ceded[s, i]
53:             w = W^exp_ceded[s, i]
54:         else
55:             W^need_added[s, j] = 0
56:             w = W^need_added[s, j]
57:         end if
58:         W^exp_ceded[s, i] -= w
59:         W_ceded[s, i] += w and W_ceded[s, i, j] = w
60:         W_added[s, j] += w and W_added[s, j, i] = w
61:         W[s, i] -= w
62:         W[s, j] += w
63:         ds[s, j] += IDS^exp[s, j] * w
64:         if ds[s, j] ≥ 1 then Elimina la cola j del grupo S_unsatisfied[s]
65:         if S_unsatisfied[s] = ∅ then
66:             if R_demanded[s, j] > MBR[s, j] para alguna j then
67:                 Redefine S_unsatisfied[s] con el valor real R_demanded (repite líneas 1-7)
68:                 if S_unsatisfied[s] sigue vacío then Salir de while
69:             end if
70:         end if
71:     end while
72: end for
```

En la fase 1 (líneas 1-8 del Alg.5), se calcula el grado de satisfacción de todas las colas, $DS\,[s,i]$, basándose en las estadísticas recopiladas (por la interfaz auxiliar) durante el último intervalo T, siguiendo la definición de la Ecuación 7, y se almacena en la variable temporal $ds\,[s,i]$. A partir de este valor se identifica a las colas insatisfechas, añadiéndolas al grupo $S_{unsatisfied}\,[s]$, y se obtiene el valor esperado de $IDS^{exp}\,[s,i]$.

En la fase 2 (líneas 9-18 del Alg.5), para tratar que la redistribución del sobrante sea lo más justa posible, se reinician los valores de los pesos, $W\,[s,i]$, a los nominales. En caso de no hacerlo, los repartos iniciales determinarían el resto de las redistribuciones. Esto supondría que solo se redistribuirían los pesos que no se usan y que no han sido cedidos en la ronda anterior, en vez de redistribuir la totalidad de los pesos no empleados. Un ejemplo de este problema se podría ilustrar en un caso en el que hay 3 colas en un slice, la cola 0 está infrautilizada, la cola 1 está insatisfecha y la cola 2 está satisfecha pero no tiene recursos suficientes para ceder, por lo que la cesión se realiza completamente de la cola 0 a la cola 1. Si llega un momento en que la cola 2 pasa a estar insatisfecha, pero la cola 1 sigue necesitando todo el peso que le cedía la cola 0, no recibirá nada, ya que la cola 0 no tiene nada que ceder, puesto que ese peso forma parte de la cola 1.

De este modo, en la fase 2, el algoritmo estima de forma intermedia que todas las colas devuelven lo recibido a las colas que se lo han cedido (líneas 11, 13 del Alg.5) y recalculan su grado de satisfacción en función de su grado en el intervalo anterior y el incremento/decremento que se estima que supone el aumento/disminución del peso. Es decir, la cola que había cedido, como

recupera peso, aumenta en función de su IDS^{exp} (líneas 12 del Alg.5), mientras que la cola que había recibido, como pierde lo que le habían cedido, disminuye según su IDS^{exp} (líneas 14 del Alg.5). Se recuerda que el grado de satisfacción está acotado entre los valores 0 y 1, es por ello, que al disminuir el grado este no puede ser inferior a 0 y al aumentarlo no puede ser superior a 1. Al haber realizado este proceso es posible que alguna de las colas haya pasado a estar insatisfecha, por lo que se añade al grupo $S_{unsatisfied}[s]$, o haya dejado de estarlo, en cuyo caso se elimina del grupo.

En este punto del algoritmo ya se han identificado las colas insatisfechas, pero falta encontrar si hay otras colas que puedan cederles recursos.

En la fase 3 (líneas 19-25 del Alg.5), se identifican las colas infrautilizadas. Con esta finalidad, se recorren las colas satisfechas, aquellas que no pertenecen a $S_{unsatisfied}[s]$, calculando el porcentaje de recursos que teóricamente corresponden a la cola i, pero no han sido usados en el último periodo ($excess[s,i]$). Si este valor supera un umbral β predefinido, por ejemplo $20\,\%$, se considera que la cola tiene recursos suficientes para ceder, por lo que se trata de una cola infrautilizada y se añade al grupo $S_{underutilized}[s]$. Este margen trata de evitar que una cola infrautilizada pueda volverse insatisfecha en el siguiente intervalo.

A partir de este exceso, manteniendo el margen ($(excess[s,i]-\alpha)\cdot W[s,i]$, línea 23 del Alg.5), se calcula el peso que puede ser cedido por la cola i del slice s a otras colas, $W^{exp}_{ceded}[s,i]$.

Por último, la fase 4 es la propia redistribución. Existen tres opciones, redistribuir con el objetivo de que las colas insatisfechas tengan un grado de satisfacción similar (líneas 26-46 del Alg.5), siguiendo una prioridad (líneas 47-72 del Alg.5) o de forma proporcional al grado de satisfacción de las colas insatisfechas (Anexo II). Esta última finalmente fue descarta por resultar en un reparto sin un objetivo tan claro y justificado como los las otras dos propuestas.

4.2.1 Reparto para igualar tasas de satisfacción

En esta propuesta se busca que en cada reparto sea la cola infrautilizada con más peso (línea 29 del Alg.5) la que ceda a la cola más insatisfecha del slice (línea 28 del Alg.5), con la finalidad de que las colas insatisfechas alcancen un grado de satisfacción similar. Este proceso se realiza mientras quede peso por repartir (q_{ceded} es la suma del peso sobrante de todas las colas infrautilizadas del slice) y quede al menos una cola insatisfecha. La cesión se realiza por pasos (valor entero 1 o superior) definidos como un porcentaje β del valor nominal de la cola que cede. Se decide establecer un porcentaje en función del peso en vez de con un número fijo porque, por ejemplo, un paso unidad puede ser adecuado para el reparto de órdenes de 100-1000 unidades de peso, pero puede suponer un retardo importante con valores más elevados, ya que la cantidad a repartir sería la misma que el número de veces que se tiene que realizar este proceso. En nuestro caso se determina $\beta = 1\,\%$, lo que permite un equilibrio entre la granularidad y el retardo.

Si durante este proceso alguna de las colas deja de estar insatisfecha, ya no recibirá más cesiones, eliminándose así del grupo $S_{unsatisfied}$ (línea 39 del Alg.5). En caso de que sean todas las colas las que hayan dejado de estarlo ($S_{unsatisfied}$ vacío), se puede decidir no hacer nada más y que el tiempo de transmisión se redistribuya según los pesos alcanzados, o recalcular el grado de satisfacción de las colas en función de su $R_{demanded}$ si este es superior a su $MBR[j]$, con la finalidad de emplear al máximo el canal (línea 40-42 del Alg.5). En el segundo caso, el proceso terminaría si sigue sin haber ninguna insatisfecha tras su recálculo (línea 43 del Alg.5).

4.2.2 Reparto por prioridad

En esta propuesta se busca que las colas infrautilizadas cedan su peso sobrante a las colas insatisfechas ordenadas por prioridad. La prioridad se determina según el índice de la cola, contra menor sea el índice, más prioritaria se considera.

Después de cada cesión se estima su grado de satisfacción (DS), si se considera que la cola está satisfecha, se elimina del grupo $S_{unsatisfied}$ y se pasa a la siguiente cola. Es decir, cuando la cola 0 haya pasado a estar satisfecha se cederá a la cola 1 y así consecutivamente.

En cuanto a las colas que ceden, estas se ordenan por su peso sobrante, es decir, la primera en ceder será aquella con más recursos infrautilizados (línea 64 del Alg.5) y lo que va a ceder a una cola es lo que esta necesita (W^{need}_{added}) o, si no tiene suficiente, el total de su sobrante (líneas 51-57 del Alg.5).

Si tras alguna cesión todas colas han pasado a ser satisfechas, como en la propuesta anterior, se puede decidir no hacer nada más y que el tiempo de transmisión se redistribuya según los pesos alcanzados, o recalcular el grado de satisfacción de las colas en función de su $R_{demanded}$ si este es superior a su MBR. En el segundo caso, el proceso terminaría si sigue sin haber ninguna insatisfecha tras su recálculo (línea 65-69 del Alg.5).

4.2.3 Redistribución Inter-Slice

La redistribución entre los slices es equivalente a la explicada para las colas de un slice. La deducción es directa aplicando los cambios en las variables correspondientes, cuya notación se recoge en la Tabla 4.

En cuando al algoritmo (Algoritmo 6), este incorpora la diferencia de que las actualizaciones se realizan directamente sobre los valores de quantum ($Q[s]$). En consecuencia, todas las variables asociadas a los pesos (W_{ceded}, W_{added}, ...) se remplazan por sus equivalentes en quantum (Q_{ceded}, Q_{added}, ...).

El resto de las variables y grupos pasan a ser los asociadas a los slices (por ejemplo, $DS[s]$ en vez de $DS[s,i]$ y $S_{unsatisfied}$ en vez de $S_{unsatisfied}$[s]). Además, el MBR de un slice ($MBR[s]$)

se define como la suma del MBR de todas sus colas.

Variable	**Descripción**
$MBR[s]$	Máxima Tasa de Bit (Maximum Bit Rate) del slice s
$R_{demanded}[s]$	Tasa de datos en bps a transmitir del slice s
$R_{achieved}[s]$	Tasa de datos efectiva en bps (nivel IP) del slice s
$S_{unsatisfied}$	Grupo de slices insatisfechos
$S_{underutilized}$	Grupo de slices que tienen pesos para ceder
$Q[s]$	Quantum usado por el slice s en el intervalo T, $Q_{nom}[s]$ por defecto. $Q_{nom}[s]$ - $Q_{ceded}[s]$ si el slice s reduce su quantum para dárselo a otros slices que están insatisfechos. $Q_{nom}[s]$ + $Q_{added}[s]$ si el slice s recibe quantum cedido por una o más slices satisfechas.
$Q_{nom}[s]$	Quantum nominal del slice s (configurado por el controlador)
Q_{total}	Suma el quantum de los slices. El valor se mantiene constante en todos los intervalos de tiempo $\sum_{\forall s} Q_{nom}[s]$
$Q_{ceded}[s,j]$	Quantum cedido por el slice s al slice j. Mismo valor que $Q_{added}[j,s]$
$Q_{ceded}[s]$	Quantum total cedido por el slice i $\sum_{\forall j \in S_{unsatisfied}} Q_{ceded}[s,j]$
$Q^{exp}_{ceded}[s]$	Quantum extra esperado del slice $s$para ser cedido
$Q_{added}[s,j]$	Quantum recibido por el slice s del slice j. Mismo valor que $Q_{ceded}[j,s]$
$Q_{added}[s]$	Quantum total que ha recibido el slice s $\sum_{\forall j \in S_{underutilized}} Q_{added}[s,j]$
$Q^{need}_{added}[s]$	Quantum requerido por el slice s para estar satisfecho
$\frac{Q[s]}{Q_{total}}$	Porcentaje de airtime (en el AP) que se espera que use el slice s respecto al uso total
$DS[s]$	Grado de satisfacción del slice s (0=totalmente insatisfecho, 1=totalmente satisfecho) $R_{achieved}[s]/R_{demanded}[s]$ if $R_{demanded}[s] < MBR[s]$ $R_{achieved}[s]/MBR[s]$ if $R_{demanded}[s] > MBR[s]$
$IDS^{exp}[s]$	Incremento de $DS[s]$ esperado para un incremento en una unidad de quantum

Tabla 4: Variables de la redistribución Inter-Slice

Algoritmo 6 Inter-Slice Redistribución del quantum no usado

#Identificar slices insatisfechos

▷ $DS[s]$: Grado de satisfacción del slice s

▷ $IDS^{exp}[s]$: El incremento del *DS* esperado por el incremento de una unidad de quantum

for cada slice s **do**

  $DS[s] = \max\left(\frac{R_{achieved}[s]}{R_{demanded}[s]}, \frac{R_{achieved}[s]}{MBR[s]}\right)$

  if $DS[s] < 1$ **then**

    Añadir slice s al grupo $S_{unsatisfied}$ y guardar par ($Q[s]$, $DS[s]$)

  end if

  $IDS^{exp}[s] = DS[s]/Q[s]$

  $ds[s] = DS[s]$ ▷ Variable para guardar la tasa de satisfacción esperada

end for

#Estimar *DS* si el quantum es el nominal

▷ $S_{ceded}[s]$: grupo de slices que han cedido recursos al slice s

▷ $Q_{ceded}[s, j] = Q_{added}[j, s]$: recursos cedidos por el slice s to j

▷ $Q_{added}[s] = \sum_{\forall j \in S_{ceded}[s]} Q_{ceded}[j, s]$

▷ $Q[s] = Q_{nom}[s] + Q_{added}[s] - Q_{ceded}[s]$

for cada slice s cuyo $Q[s] > Q_{nom}[s]$ **do**

  for cada slice $j \in S_{ceded}[s]$ **do**

    $Q[j]$ += $Q_{ceded}[j, s]$

    $ds[j] = \min(ds[j] + IDS^{exp}[j] * Q_{ceded}[j, s], 1)$

    $Q[s]$ -= $Q_{added}[s, j]$

    $ds[s] = \max(ds[s] - IDS^{exp}[s] * Q_{added}[s, j], 0)$

  end for

  if $ds[s] < 1$ **then** Añade el slice s al grupo $S_{unsatisfied}$

  else Elimina al slice s del grupo $S_{unsatisfied}$

end for

▷ Los valores de quantum han sido reseteados, $Q[s] = Q_{nom}[s]\ \forall s$

$Q_{ceded}[s, j] = Q_{added}[s, j] = Q_{ceded}[s] = Q_{added}[s] = 0\ \forall s, j$

#Identificar slices infrautilizados

▷ $excess[s]$: porcentaje de la porción de recursos que teóricamente corresponden al slice s y no están ocupados

for each slice $s \notin S_{unsatisfied}$ **do**

  $excess[s] = \left(\frac{Q[s]}{Q_{total}} - \frac{airtime_T[s]}{airtime_T}\right) / \left(\frac{Q[s]}{Q_{total}}\right)$

  if $excess[s] > \alpha$ **then**

    Añade el slice s al grupo $S_{underutilized}$

    $Q^{exp}_{ceded}[s] = (excess[s]\text{-}\alpha) * Q[s]$ ▷ Peso extra para ceder

  end if

end for

#Opción Resdistribuir quantums para conseguir una tasa de satisfacción similar

$q_{ceded} = \sum_{\forall s \in S_{underutilized}} Q^{exp}_{ceded}[s]$

▷ Distribute q_{ceded} among slices $\in S_{unsatisfied}$

while $q_{ceded} > 0$ **do**

  Find slice $j \in S_{unsatisfied}$ with lower $ds[j]$

  Find $s \in S_{underutilized}$ with the highest $Q^{exp}_{ceded}[s]$

  if $\lceil Q_{nom}[s] * \beta \rceil < q_{ceded}$ **then** $step = \lceil Q_{nom}[s] * \beta \rceil$

  else $step = q_{ceded}$

  q_{ceded} -= $step$

  $Q^{exp}_{ceded}[s]$ -= $step$

  $Q_{ceded}[s]$ += $step$ and $Q_{ceded}[s, j]$ += $step$

  $Q_{added}[j]$+= $step$ and $Q_{added}[j, s]$ += $step$

  $Q[s]$ -= $step$

  $Q[j]$ += $step$

  $ds[j]$ += $IDS^{exp}[j]$ * step

  if $ds[j] \geq 1$ **then** Remove j to $S_{unsatisfied}$ group

  if $S_{unsatisfied} = \emptyset$ **then**

    if $R_{demanded}[j] > MBR[j]$ para alguna j **then**

      Redefine $S_{unsatisfied}$ with real $R_{demanded}$ (*repite líneas 1-7*)

      if $S_{unsatisfied}$ still empty **then** Exit *while*

    end if

  end if

end while

```
#Opción Redistribuye quantum siguiendo una prioridad
47: Calcula el Q requerido por cada slice j para estar satisfecho Q^need_added[j]
48: for cada slice s ∈ S_underutilized ordenado de forma descendiente según su Q^exp_ceded[s] do
49:     while Q^exp_ceded[s] > 0 do
50:         Itera los slices del grupo S_unsatisfied en orden de prioridad
51:         if Q^need_added[j] > Q^exp_ceded[s] then
52:             Q^need_added[j] -= Q^exp_ceded[s]
53:             q = Q^exp_ceded[s]
54:         else
55:             Q^need_added[j] = 0
56:             q = Q^need_added[j]
57:         end if
58:         Q^exp_ceded[s] -= q
59:         Q_ceded[s] += q and Q_ceded[s, j] = q
60:         Q_added[j] += q and Q_added[j, s] = q
61:         Q[s] -= q
62:         Q[j] += q
63:         ds[j] += IDS^exp[s, j] * q
64:         if ds[j] ≥ 1 then Elimina j del grupo S_unsatisfied
65:         if S_unsatisfied = ∅ then
66:             if R_demanded[j] > MBR[j] para alguna j then
67:                 Redefine S_unsatisfied con el R_demanded real (repite líneas 1-7)
68:                 if S_unsatisfied sigue vacío then Salir de while
69:             end if
70:         end if
71:     end while
72: end for
```

4.2.4 Pruebas y resultados

4.2.4.1 Procesado de los resultados

En estas pruebas se requiere analizar más parámetros, es por eso que, al fichero descrito en el apartado 4.1.2.2 se le añade uno nuevo.

En este caso se necesita recoger en cada ventana temporal los quantum de los slices y los pesos de todas sus colas, así como los datos relativos al grado de satisfacción (DS), tanto del slice ($R_{demanded}[s]$, $R_{archived}[s]$, $MBR[s]$), como de sus colas ($R_{demanded}[s,i]$, $R_{archived}[s,i]$, $MBR[s,i]$).

A partir de estos, se crea un programa en Matlab que devuelva la evolución de los quantum y de DS.

4.2.4.2 Evaluación y resultados

En este apartado se muestran y evalúan los resultados obtenidos tras experimentar con los métodos de adaptación propuestos. En concreto, se exponen dos experimentos con escenarios distintos, para analizar y medir el rendimiento. El primero de ellos está centrado en mostrar

el impacto aislado de la adaptación intra-slice, excluyendo el uso de adaptaciones inter-slice. El segundo aplica simultaneamente adaptaciones a nivel inter-slice e intra-slice para evaluar el rendimiento global del sistema.

Las condiciones de laboratorio siguen siendo las mismas que las mostradas en el apartado 4.1.2. Además, en la Tabla 5 se recogen las condiciones para ambas pruebas, incluyendo los valores nominales de quantum (Q_{nom}) y peso (W_{nom}) configurados para los slices y las colas, los tamaños de los datos de los paquetes para cada flujo, los valores de MBR (tasa de datos neta, para facilitar la visualización en el análisis de los resultados) y las tasas de generación que van variando a lo largo de los 70 segundos de duración de la prueba. En ellas se realiza la inyección en 802.11n/HT en la banda 5GHz y las STAs están dentro de un radio de 2 metros del AP. Se generan nueve flujos UDP dirigidos hacia las respectivas STAs, cada una asociada a una única cola y slice. En total van dirigidos a 3 slices de hasta 4 colas. Cada una de estas colas contiene flujo de una única STA, a excepción de la cola 0 del slice 1, que contiene tráfico dirigido a las STA 2 y STA 3, para poder verificar el rendimiento cuando múltiples flujos de datos pertenecientes a diferentes STAs están asociados con la misma cola. Además, las MCS son variadas pero fijas, para facilitar la interpretación de los resultados obtenidos.

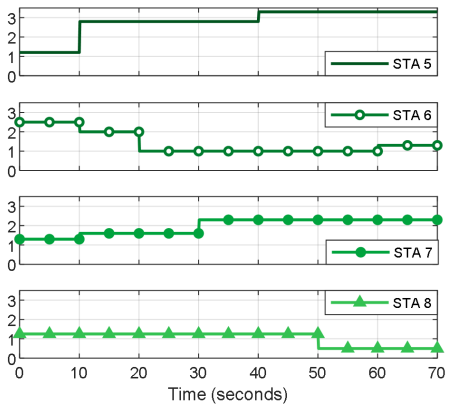
Por otro lado, para estas pruebas se ha habilitado la agregación de tramas, cuyos tamaños máximos están reflejados en la Tabla 5. Esta configuración permite alcanzar mayores niveles de rendimiento al generar tamaños variables de AMSDU y, consecuentemente, demandas variables de airtime.

		Slice 0		**Slice 1**			**Slice 2**			
Quantum (Q_{nom}) [µs]		3000		2000			5000			
		Cola 0	**Cola 1**	**Cola 0**		**Cola 1**	**Cola 0**	**Cola 1**	**Cola 2**	**Cola 3**
Peso (W_{nom})		120	80	140		60	70	60	40	30
MBR [Mbps]		2.5	2	2.5		2	3	2.5	2.5	2
STA		*STA 0*	*STA 1*	*STA 2*	*STA 3*	*STA 4*	*STA 5*	*STA 6*	*STA 7*	*STA 8*
MCS		2	4	3	2	6	2	3	3	6
Tasa [Mbps]	**Prueba 1**	2'5 (0-70 s)	2 (0-70 s)	1 (0-70 s)	0'8 (0-70 s)	1'2 (0-70 s)	1'2 (0-10 s) 2'8 (10-40 s) 3'3 (40-70 s)	2'5 (0-10 s) 2 (10-20 s) 1 (20-60 s) 1'3 (60-70 s)	1'3 (0-10 s) 1'6 (10-30 s) 2'3 (30-70 s)	1'25 (0-50 s) 0'5 (50-70 s)
	Prueba 2	2'5 (0-10 s) 1 (10-60 s) 2 (60-70 s)	2 (0-20 s) 3 (20-30 s) 1 (30-70 s)	1'6 (0-70 s)	1'2 (0-70 s)	1'2 (0-20 s) 2 (20-70 s)				
Tamaño UDP (bytes)		1250B	250B	200B	350B	700B	350B	350B	200B	250B
Max tamaño agregación		1200 B	1200 B	1200 B		1200 B	1200 B	1200 B	1200 B	1200 B

Tabla 5: Escenario de pruebas para la redistribución dinámica de quantum

Primero, se va a mostrar y explicar los resultados obtenidos del primer experimento. En este se considera el algoritmo de redistribución intra-slice (con un intervalo de actualización de T=1s), mientras que se mantiene el esquema ADRR con quantums fijos entre los slices. Se comparan las opciones de redistribución de pesos propuestas con el esquema ADRR que mantiene los pesos fijos. El objetivo es mostrar cómo los algoritmos garantizan una redistribución efectiva de los recursos según los criterios seleccionados, asegurando a su vez, en caso de ser necesario, el tiempo de transmisión garantizado basado en valores nominales y sin degradar el rendimiento

de las colas que ceden recursos.

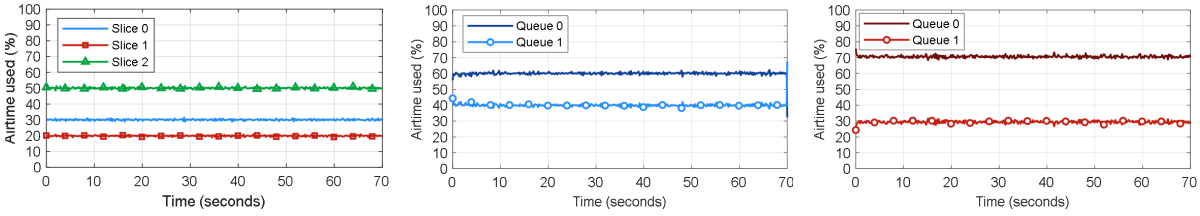


Antes de comenzar con el análisis, en la Figura 12 se muestra la generación de tráfico con destino las STAs del slice 2 (datos recogidos en la Tabla 5). A modo de nota, destacar que los colores y marcadores utilizados para ilustrar las STAs en esta figura corresponden a la representación de la cola y el slice que transmiten su flujo en las figuras posteriores, con el propósito de facilitar la lectura e interpretación conjunta de los resultados.

Figura 12: Experimento 1. Tráfico hacia el slice 2

Recordemos que las propuestas de redistribuciones de peso solo tienen sentido en caso de que haya más de dos colas, puesto que, en caso contrario, la que puede ceder dará todo de forma natural a la otra cola del slice. Es por ello que se diseñó la prueba para que el análisis de la redistribución de pesos se centre en el comportamiento del slice 2, que tiene 4 colas.

Con este objetivo se establecieron demandas de tráfico en los slices 0 y 1 constantes y lo suficientemente altas como para evitar la redistribución de airtime entre los slices. Lo mismo para sus colas. Si nos fijamos en la Figura 13, vemos que la distribución del airtime se mantiene constante en la ocupación garantizada por los valores nominales dados en la Tabla 5, tanto para los slices (30 %, 20 % y 50 %, Fig.13a), como para las colas del slice 0 (60 % y 40 %, Fig.13b) y del slice 1 (70 % y 30 %, Fig.13c). Esta figura corresponde a los resultados del escenario en el que se aplica el criterio de igualación del grado de satisfacción, pero las otras opciones tienen resultados similares.



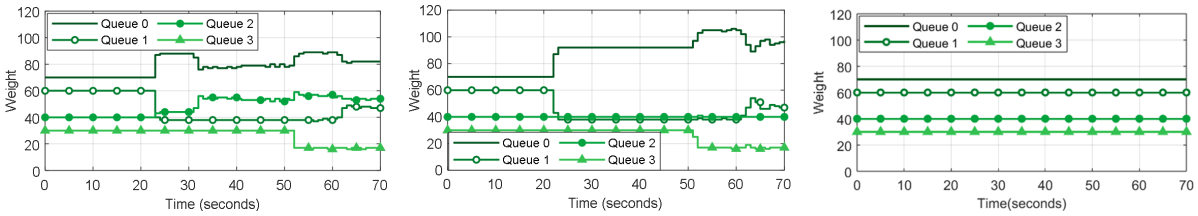
(a) Distribución del airtime entre los slices

(b) Distribución del airtime de las colas en el slice 0

(c) Distribución del airtime de las colas en el slice 1

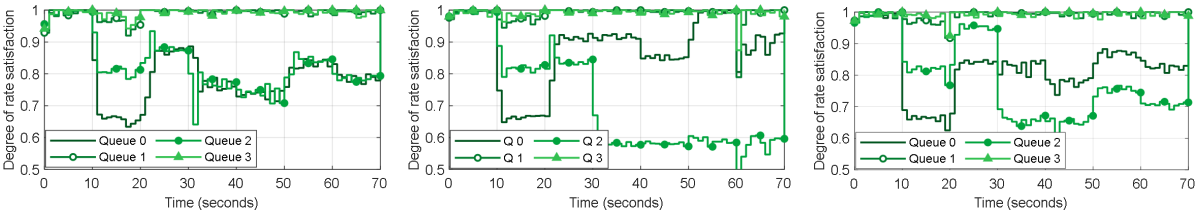
Figura 13: Experimento 1. Distribución del airtime entre los slices y entre las colas de los slices 0 y 1.

Ahora pasamos analizar el comportamiento de las colas del slice 2 a través de las Figuras 14-17, donde los resultados (a) corresponden con la opción de intentar igualar el grado de satisfacción, los (b) con la opción de ceder por prioridad y los (c) con el esquema ADRR manteniendo los pesos de las colas fijos.



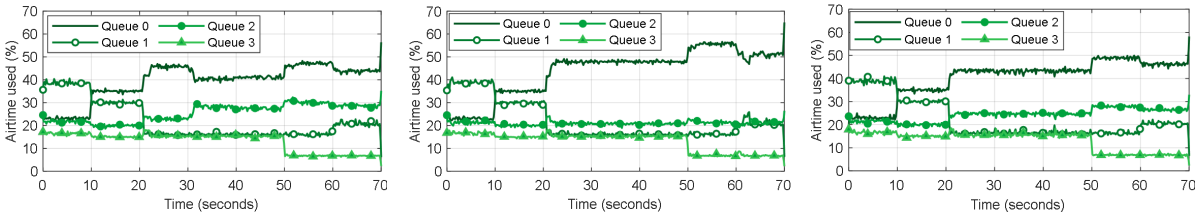
(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 14: Experimento 1. Evolución del peso de las colas del slice 2



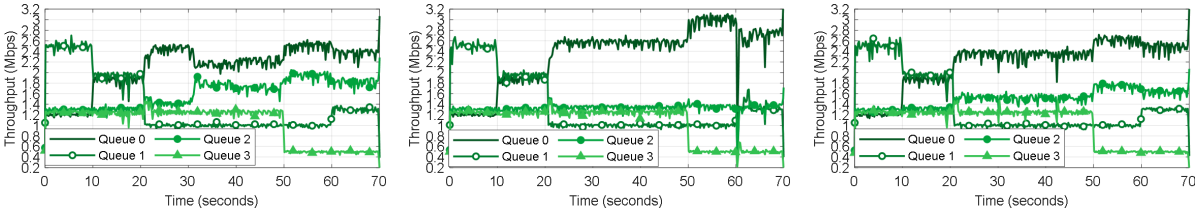
(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 15: Experimento 1. Grado de satisfacción de las colas del slice 2



(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 16: Experimento 1. Distribución del airtime entre las colas del slice 2



(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 17: Experimento 1. Throughput de las colas del slice 2

En el primer periodo, de 0 a 10 segundos, observamos que el canal del AP no está saturado, por lo que la estrategia ADRR redistribuye naturalmente los recursos no utilizados de las colas infrautilizadas (cola 0, en la Figura 16 se muestra que su ocupación del airtime es bastante inferior a su garantizada, 35 %) a otros que requieren recursos adicionales. En consecuencia, la adaptación de peso/quantum no se activa, por lo que el comportamiento observado es el mismo en las tres opciones. No es un periodo de saturación porque no se están encolando paquetes en

ninguna de la colas. Esto lo vemos en la Figura 17, ya que todas están transmitiendo a la misma tasa que la de generación y, además, en la Figura 15 todas aparecen satisfechas (DS=1).

En el siguiente periodo, de 10 a 20 segundos, la cola que estaba infrautilizada pasa de recibir 1'2Mbps a 2'8Mbps, lo que hace que ya no pueda redistribuir entre el resto y recupere su ocupación garantizada del airtime. Como el resto siguen sin estar infrautilizados, esto se transforma en una situación de saturación, donde la distribución del airtime pasa a ser la garantizada por sus valores de peso nominales (35 %, 30 %, 20 % y 15 %, respectivamente).

En el segundo 20, la tasa de generación de la cola 1 baja, lo que hace que en t=23s en la Fig.14 y t=22s en la Fig.15 comience el proceso de redistribución del peso. La razón de que esto no se haya producido instantáneamente en t=20s se debe a que la cola tenía paquetes pendientes encolados, por lo que a partir de ese momento es cuando empieza a vaciarse, tiempo en el que no se tienen recursos suficientes para iniciar la redistribución.

Cuando esta comienza nos permite ver que el rendimiento, en términos de distribución de airtime, throughput y grado de satisfacción, difiere significativamente entre los algoritmos propuestos y comparado con la configuración de peso estático. Se observa que la cola 3, en todos los casos, emplea la totalidad de su airtime garantizado, además de tener un alto grado de satisfacción, mientras que la 0 y la 2 están bastante insatisfechas. Es por eso que las colas 0 y 2 son las candidatas para recibir peso cedido por parte de la cola 1.

En el escenario donde se aplica el algoritmo para tratar de igualar los grados de satisfacción (a) se ve, efectivamente, que en el periodo de 20 a 50 segundos, la cola 1 está cediendo peso a las colas 0 y 2 (Fig.14a). Esto se traduce en que, independientemente de las variaciones en las demandas de tráficos en estas colas, su grado de satisfacción (DS) alcanza valores muy similares (Fig.15a). Cabe señalar que cuando la cola 2 aumenta las demandas de tráfico, dentro de la ventana de monitorización del grado de satisfacción de tasa t=30s-31s, la adaptación de quantum se lleva a cabo en dos pasos en los siguientes 2s. Dado que el aumento en el tráfico no se compensa con una disminución en otros tráficos, el DS de la cola 2 experimenta momentáneamente valores bajos. Por último, en el segundo 50, el tráfico recibido en la cola 3 disminuye (la generación pasa de 1'25Mbps a 0'5Mbps) lo que hace que pase a ser una cola infrautilizada y ceda también sus pesos sobrantes a las colas 0 y 2.

El comportamiento es bastante distinto en el escenario donde se redistribuye el peso sobrante siguiendo una prioridad (b). Al igual que antes, en el periodo de 20 a 50 segundos, se observa que las colas 0 y 2 están insatisfechas y la cola 1 infrautilizada. No obstante, si nos fijamos en la Fig.14b la cola 2 no recibe peso. Esto se debe, precisamente, a esa prioridad, ya que mientras la cola 0 necesite recursos, ninguna otra los recibirá. El cumplimiento de esta prioridad se evidencia también en la Fig.15a donde el valor de DS para la cola 0 es bastante superior al caso anterior, rondando el 0.9, mientras que la cola 2, al no recibir cesiones, se queda en 0.6. En el segundo 50, al pasar la cola 3 a estar infrautilizada, cede también a la cola prioritaria que consigue estar satisfecha (importante resalta que la cola 0 (Fig.17b) está limitado a su MBR, 3Mbps frente a 3.3Mbps de tráfico entrante). No obstante, en el segundo 60 hay una fuerte

interferencia externa (17a) que hace que las colas que habían cedido tengan que recuperar parte de su peso hasta volver a cederlo al final. En todos los casos, incluso cuando una cola no recibe peso cedido, su airtime garantizado está asegurado.

Con todo esto puede concluirse que cuando se aplica la adaptación de peso, en ambos casos, la redistribución de peso se implementa de manera efectiva sin ninguna degradación en el rendimiento de los colas que ceden (colas 1 y 3). El throughput sigue siendo el mismo que si hubieran mantenido sus pesos nominales (Fig.17a,b,c). De este modo, se garantizan ajustes predecibles y controlados con el objetivo de cumplir requerimientos QoS sin afectar a las colas que ceden.

Complementariamente, se muestra en las Figuras 18-19 que en este escenario, concretamente corresponde con el de igualar grados de satisfacción (a), al permitir la agregación se continúa estimando correctamente el consumo del airtime (Fig.16a).

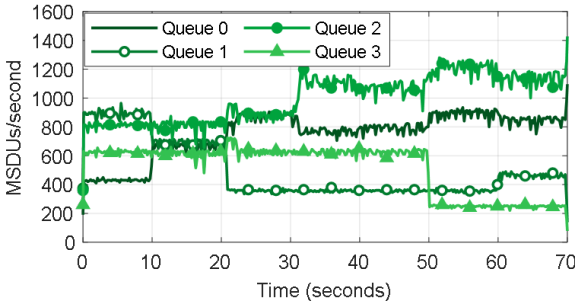


Figura 18: Experimento 1 con redistribución para alcanzar DS similares. MSDUs/s colas del slice 2.

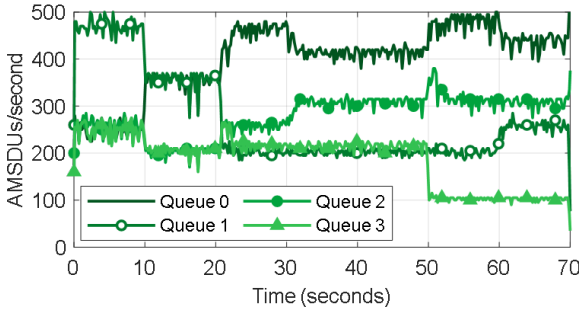


Figura 19: Experimento 1 con redistribución para alcanzar DS similares. AMSDUs/s colas del slice 2.

En cuanto al segundo experimento, se diseñó para permitir un análisis simultáneo de los efectos de la interacción entre los algoritmos de redistribución de quantum($Q[s]$)/peso($W[s,i]$) a nivel inter-slice e intra-slice. Para ello, a nivel inter-slice se realiza una comparación entre las opciones de redistribución del quantum para lograr un grado de satisfacción similar y de redistribución siguiendo prioridad con el esquema ADRR de pesos fijos. En cuanto al nivel intra-slice, se emplea la redistribución para alcanzar un grado de satisfacción similar entre las colas, por ser esta la opción donde los resultados son más evidentes y significativos.

Las demandas de tráfico del slice 2 son las mismas a las del anterior experimento (Figura 12), pero las de los slices 0 y 1 han cambiado. Sus valores para esta prueba quedan reflejados en la parte *Prueba 2* de la Tabla 5 y se ilustran en la Figura 20 para que sea más sencillo interpretar los resultados.

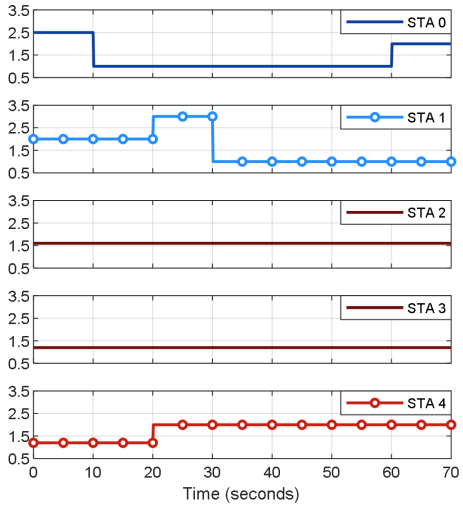
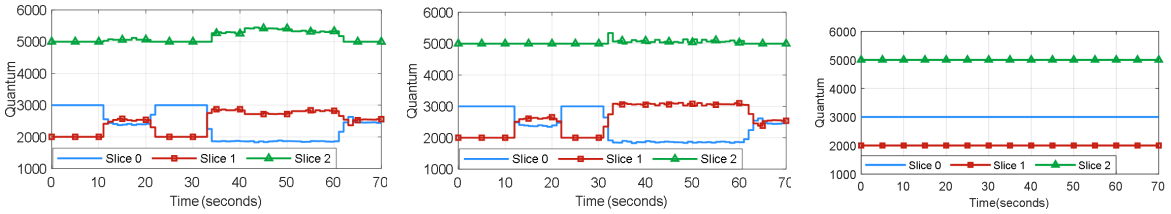


Figura 20: Experimento 2. Tráfico hacia las colas de los slices 0 (azul) y 1 (rojo)

A modo de nota, destacar que los colores y marcadores utilizados para ilustrar las STAs en esta figura corresponden a la representación de la cola y el slice que transmiten su flujo en las figuras posteriores, con el propósito de facilitar la lectura e interpretación conjunta de los resultados.

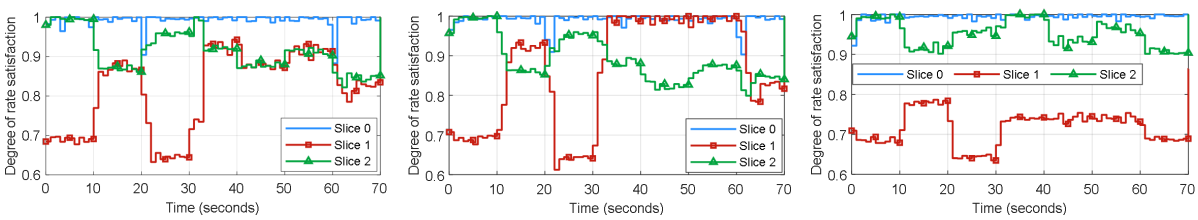
Estos flujos mantienen al AP en estado de saturación durante toda la prueba de 70 segundos, ya que, en caso contrario, no se llevaría a cabo la redistribución que queremos valorar. Recordemos que en ausencia de saturación ADRR redistribuye los recursos de manera natural.

El análisis del comportamiento de los slices se va a realizar a través de las figuras que muestran la evolución del quantum (Fig.21), los grados de satisfacción (Fig.22), la distribución del airtime (Fig.23) y el throughput (Fig.24). En cuanto al comportamiento intra-slice, se visualiza la evolución de los pesos en las colas del slice 2 (Fig.25) y sus grados de satisfacción (Fig.26). En ambos casos, los resultados (a) corresponden con la opción de intentar igualar el grado de satisfacción, los (b) con la opción de ceder por prioridad y los (c) con el esquema ADRR manteniendo los quantums/pesos fijos, siempre promediados en intervalos de 200ms.



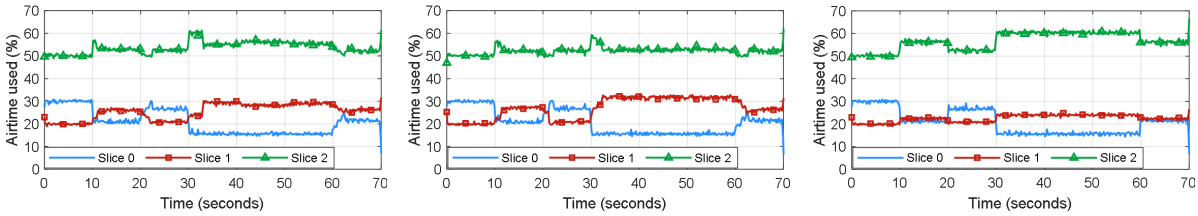
(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 21: Experimento 2. Evolución del quantum de los slices



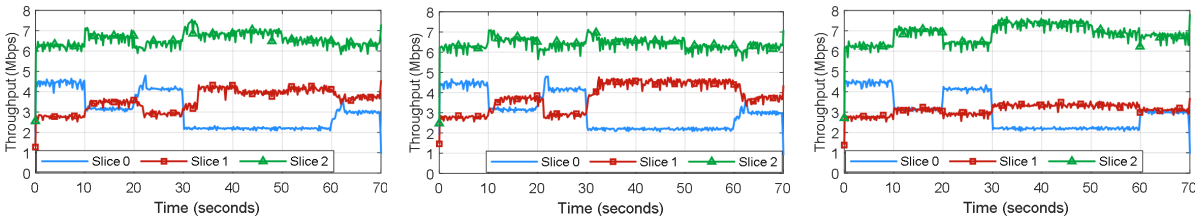
(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 22: Experimento 2. Grado de satisfacción de los slices



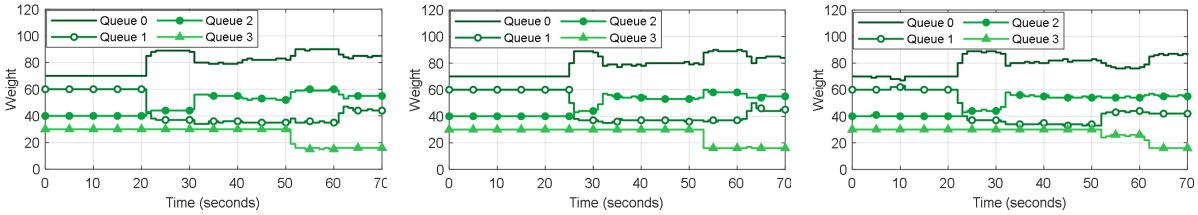
(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 23: Experimento 2. Distribución del airtime entre los slices



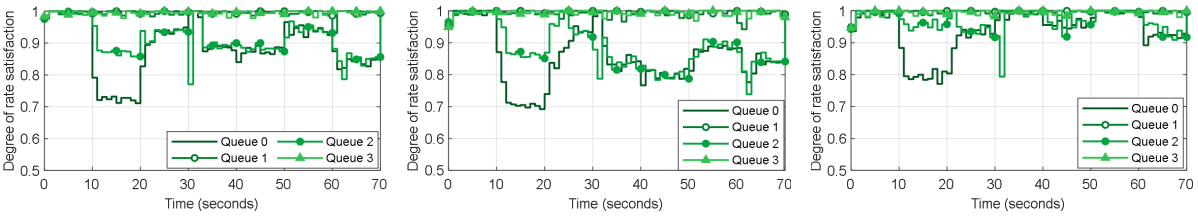
(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 24: Experimento 2. Throughput de los slices



(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 25: Experimento 2. Evolución del peso de las colas del slice 2 siguiendo una redistribución para igualar sus DS



(a) Opción de igualar grado de satisfacción (b) Opción de prioridad (c) Opción peso estático

Figura 26: Experimento 2. Grado de satisfacción de las colas del slice 2 siguiendo una redistribución para igualar sus DS

En el primer intervalo, de 0 a 10 segundos, ningún slice tiene recursos que ceder, por lo que la distribución del airtime sigue los valores garantizados (30 %, 20 % y 50 %, respectivamente, Fig.23), definidos por los quantums nominales (Fig.21). Si nos fijamos en el throughput (Fig.24), vemos que las tasas para los slices 0 y 2 son muy similares a las demandadas (las

tasas de generación son 4’5Mbps para el slice 0 y 6’25Mbps para el slice 2), mientras que el slice 1 está lejos de satisfacer su demanda (la tasa de generación en el iperf es de 4Mbps). Esto se ve reflejado en valores de DS prácticamente satisfechos para los slices 0 y 2, pero significativamente más bajo, 0.7, para el slice 1 (Fig.26).

En el segundo 10, la demanda de tráfico del slice 0 desciende considerablemente debido a que la tasa de generación de su cola 0 pasa de 2’5Mbps a 1Mbps (Fig.20). Esto supone que se convierte en un slice totalmente satisfecho con quantum para ceder, es decir, un slice del grupo de infrautilizados, iniciando así el proceso de redistribución. Este momento también coincide con un aumento en la demanda por parte del slice 2 (Fig.12), que pasa a estar bastante insatisfecho, por lo que el slice 1 y 2 están a la espera de estos recursos sobrantes.

En el caso de la redistribución para alcanzar grados de satisfacción similares (a), se observa que el slice 0 cede quantum a los slices 1 y 2 (más al 2 por estar más insatisfecho) (Fig.21a), consiguiendo que sus grados se igualen (Fig.22a). Por otro lado, en la redistribución por prioridad (b), el slice 0 cede todo su quantum sobrante al slice 1 (Fig.21b), ya que este es más prioritario, consiguiendo que su DS se reduzca considerablemente, siendo inferior a la del slice 2 (Fig.22b).

Sin embargo, en el instante 20, la demanda del slice 0 aumenta por un incremento de tráfico en su cola 1 (Fig.20), lo que hace que deje de estar infrautilizada y necesite recuperar lo que había cedido. De este modo, se vuelve a la distribución de airtime con los valores garantizados según los quantums nominales (Fig.23).

Esta situación se mantiene hasta el segundo 30, en el que la demanda del slice 0 disminuye de nuevo considerablemente (Fig.20). No obstante, el proceso de redistribución del quantum se retrasa al segundo 33 debido a que los mecanismos de ajuste del sistema requieren tiempo adicional para reasignar recursos y estabilizar las mediciones del grado de satisfacción de la tasa.

Esto supone que el slice 0 vuelve a tener exceso de recursos que va a poder ceder al resto de los slices del AP que se encuentran insatisfechos.

En el caso de la redistribución para alcanzar grados de satisfacción similares (a), de nuevo vemos como el slice 0 cede quantum al resto de los slices (Fig.21a), haciendo que alcancen grados de satisfacción similares hasta el segundo 60 (Fig.22a). En cuanto al reparto por prioridad, se ve que el aumento de demanda del slice 1 ha hecho que esté todavía más insatisfecho, pero gracias a la cesión casi total del slice 0 consigue estar satisfecha (Fig.22b). Como el slice 1 está satisfecho con lo cedido (vemos que se calcula la satisfacción en función de 4500 Mbps, su MBR, en lugar de los 4800 Mbps asociados con la tasa requerida) y el slice 0 aún tiene quantum para ceder, cede el que le queda al slice 2 (Fig.21b).

En el segundo 60, un aumento de nuevo en la demanda del slice 0 hace que tenga que recuperar parte de los recursos cedidos. Sin embargo, sigue siendo un slice con exceso, aunque

menor, que lo redistribuye. En cuanto al reparto para igualar grados de satisfacción, el slice 0 cede únicamente al slice 1 en un intento de igualar el mayor nivel de satisfacción del slice 2, aunque este esfuerzo resulta insuficiente. Durante este mismo intervalo, el margen de quantum reservado por el slice 0 (un porcentaje α de recursos excedentes) se redistribuye entre las slices 1 y 2 según los pesos actualizados. Esta redistribución explica por qué el airtime del slice 2 supera los valores correspondientes a su quantum nominal. Esto es similar a lo que ocurre en el algoritmo de prioridad para este intervalo.

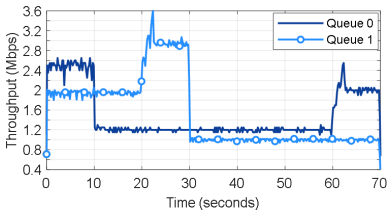
Finalmente, cuando se aplica quantum estático a los slices, la redistribución natural del airtime permitida por ADRR aumenta el airtime de las slices 1 y 2 en proporción a su quantum nominal. En consecuencia, el slice 2 alcanza un mayor airtime en comparación con las opciones previas de adaptación de quantum (a,b). Durante los intervalos de 32 a 40 segundos y de 51 a 60 segundos, las colas del slice 2 alcanzan un grado de satisfacción igual a 1 (Fig.26c), correspondiente a la suma de las tasas requeridas de sus colas, restringida por los requisitos de MBR (aproximadamente 7'45 Mbps y 6'8 Mbps, respectivamente, en lugar de 7'1 Mbps en el segundo intervalo).

Es importante resaltar las Figuras 26a,b,c, ya que muestran que el algoritmo intra-slice mantiene su objetivo de intentar igualar los grados de satisfacción independientemente del algoritmo que se esté aplicando a nivel inter-slice.

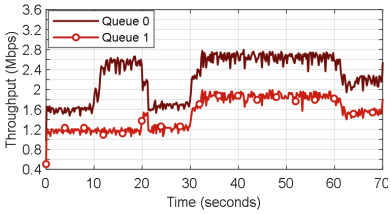
Por último, cabe destacar, que el throughput del slice que cede (slice 0) se mantiene y satisface para todos los casos. Existen ocasiones en los que este es inferior al esperado (Fig.24), segundos 20 y 30, debido a retrasos en la readaptación de quantum cuando la tasa requerida aumenta. Sin embargo, el buffer se vacía rápidamente en los segundos siguientes. En el segundo 23, el throughput supera el límite MBR del slice (4'5Mbps), pero esto se debe a que no se excede su airtime garantizado del 30 %.

De manera complementaria a las figuras comentadas se encuentra la Figura 27. En la Fig.27a y Fig.27d se muestran el throughput y la distribución de airtime, respectivamente, en el caso de redistribución para conseguir un grado de satisfacción similar. En ellas se observa una redistribución del airtime de manera natural.

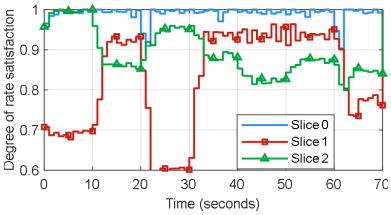
En la Fig.27b y Fig.27e se muestran el throughput y la distribución de airtime, en este caso del slice 1 cuando la redistribución se hace siguiendo la prioridad. Este slice está también formado por dos colas, por lo que se observa una redistribución del airtime de manera natural. En este caso, el objetivo es resaltar que, durante el intervalo de 32 a 60 segundos, aunque el slice no excede el MBR asociado (4'5 Mbps), gracias al airtime garantizado de las colas, el throughput alcanzado por la cola 0 puede superar su MBR (2'5 Mbps). Comparando la Fig.27c, donde el grado de satisfacción se calcula respecto a la tasa requerida ($R_{demanded}[s]$), sin estar limitado por el MBR, con la Fig.22b, se puede observar claramente cómo el algoritmo impone los umbrales de MBR del slice.



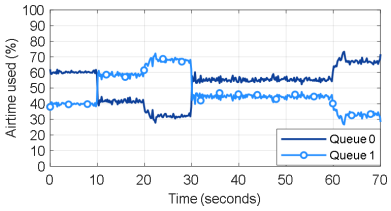
(a) Throughput de las colas del slice 2. Opción de igualar grado de satisfacción



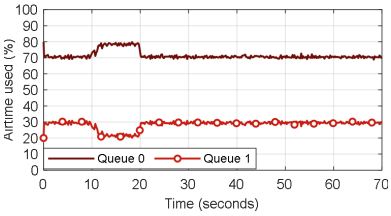
(b) Throughput de las colas del slice 1. Opción de prioridad



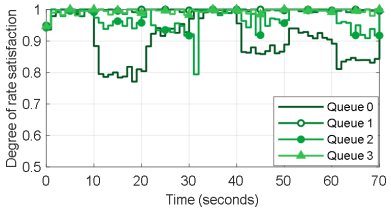
(c) Grado de satisfacción de los slice en función del $R_{demanded}$. Opción de prioridad



(d) Distribución del airtime entre las colas del slice 2. Opción de igualar grado de satisfacción



(e) Distribución del airtime entre las colas del slice 1. Opción de prioridad



(f) Grado de satisfacción de las colas del slice 2 en función del $R_{demanded}$. Opción de peso estático

Figura 27: Experimento 2. Evolución del peso de las colas del slice 2

Del mismo modo, si se compara la Fig.27f, donde el grado de satisfacción de las colas del slice 2 está calculado respecto a la tasa demandada real ($R_{demanded}[s]$), con la Fig.26c, se observa nuevamente cómo los algoritmos imponen los umbrales de MBR.

Para terminar, se puede concluir a partir de los experimentos 1 y 2, que las adaptaciones locales de quantum/pesos propuestas en los niveles inter-slice e intra-slice son capaces de modificar la reasignación natural de ADRR para cumplir con requisitos específicos de QoS. De este modo, manteniendo las garantías de aquellos slices y colas que ceden, se pueden realizar ajustes predecibles y controlados.

5 CONCLUSIÓN

En este TFM se ha creado un sistema de colas usado para realizar RAN Slicing en redes WLAN. A partir de este sistema se ha desarrollado un mecanismo de Slicing ISS-QoS, donde se realiza un reparto de los recursos a nivel intra-slice e inter-slice basado en los principios de ADRR e incorporando la adaptación en la redistribución de quantum.

En primer lugar, se ha observado que cuando los recursos del AP son suficientes, la implementación garantiza a cada slice/cola los recursos establecidos en configuración, independientemente de las variaciones en las demanda o las retransmisiones. Además, se observa que cuando uno no necesita la totalidad de los recursos garantizados, se lleva a cabo una redistribución natural y justa entre el resto de los slices/colas.

Por otro lado, cuando se produce una situación de congestión, se observa el aislamiento a nivel de slice y cola, donde cada uno tiene los recursos garantizados sin afectar a lo asegurado en los demás.

Además, se han implementado una serie de propuestas para modificar la redistribución natural de los recursos. De este modo, el quantum/peso no usado por un slice/cola es repartido entre el resto de forma que se cumplan unos requisitos específicos de QoS. Esto se lleva a cabo mediante la cesión por parte de la cola con recursos sobrantes, donde se ha verificado que se siguen manteniendo las garantías de los slices/colas que cede.

Con todo ello se puede concluir que se han logrado los objetivos establecidos. Sin embargo, se identifican varias líneas futuras para ampliar el proyecto. Por un lado, continuando con la implementación de mecanismos de redistribución, sería interesante explorar otros que incluyan criterios de retardo. Por otro lado, con un enfoque más avanzado, se podría desarrollar la componente centralizada del slicing en el controlador. Esto incluiría la selección de quantums y pesos nominales en escenarios compuestos por múltiples APs, junto a su adaptación dinámica al cambiar las distribuciones de STAs o el tráfico entre los APs.

BIBLIOGRAFÍA

[1] M. Richart, J. Baliosian, J. Serrat et al., “Slicing in WiFi Networks Through Airtime-Based Resource Allocation”, J Netw Syst Management 27, 784–814 (2019). doi.org/10.1007/s10922-018-9484-x.

[2] iperf. `https://iperf.fr/`

[3] E. Coronado, R. Riggio, J. Villa1ón and A. Garrido, “Lasagna: Programming Abstractions for End-to-End Slicing in Software-Defined WLANs”, 2018 IEEE 19th Int’l Symposium on “A World of Wireless, Mobile and Multimedia Networks” (WoWMoM), Chania, Greece, 2018, pp. 14-15, doi: 10.1109/WoWMoM.2018.8449797.

[4] P. H. Isolani et al. , “SDN-based Slice Orchestration and MAC Management for QoS delivery in IEEE 802.11 Networks”, 2019 Sixth International Conference on Software Defined Systems (SDS), Rome, Italy, 2019, pp. 260-265, doi: 10.1109/SDS.2019.8768642.

[5] J. Saldana et al., “Attention to Wi-Fi Diversity: Resource Management in WLANs with Heterogeneous APs” in IEEE Access, vol. 9, pp. 6961-6980, 2021, doi: 10.1109/ACCESS.2021.3049180.

[6] J. Saldana et al., “Unsticking the Wi-Fi Client: Smarter Decisions Using a Software Defined Wireless Solution” in IEEE Access, vol. 6, pp. 30917-30931, 2018, doi: 10.1109/ACCESS.2018.2844088.

[7] Radiotap. `https://www.radiotap.org/`

ÍNDICE DE FIGURAS

1. Arquitectura implementada a alto nivel. 9
2. Implementación local del AP (plano de datos) 10
3. Estructura trama IP y uso del campo DSCP. 13
4. Esquema de transmisión de un paquete. 19
5. Distribución del airtime entre los slices y las colas. 24
6. Paquetes por segundo transmitidos correctamente en cada las colas de cada slice. 25
7. Throughtput por colas de cada slice. 25
8. Tramas retransmitidas de cada cola de los slices 1, 2 y 3. 26
9. Variaciones de MCS en la STA 3. 26
10. Resultado del experimento de la Tabla 2 si no se aplican las correcciones por retransmisiones. 28
11. Resultado del experimento de la Tabla 2 con órdenes de quantum mayores y menores. 29
12. Experimento 1. Tráfico hacia el slice 2 42
13. Experimento 1. Distribución del airtime entre los slices y entre las colas de los slices 0 y 1. 42
14. Experimento 1. Evolución del peso de las colas del slice 2 43
15. Experimento 1. Grado de satisfacción de las colas del slice 2 43
16. Experimento 1. Distribución del airtime entre las colas del slice 2 43
17. Experimento 1. Throughput de las colas del slice 2 43
18. Experimento 1 con redistribución para alcanzar DS similares. MSDUs/s colas del slice 2. 45
19. Experimento 1 con redistribución para alcanzar DS similares. AMSDUs/s colas del slice 2. 45
20. Experimento 2. Tráfico hacia las colas de los slices 0 (azul) y 1 (rojo) 46
21. Experimento 2. Evolución del quantum de los slices 46
22. Experimento 2. Grado de satisfacción de los slices 46
23. Experimento 2. Distribución del airtime entre los slices 47
24. Experimento 2. Throughput de los slices 47
25. Experimento 2. Evolución del peso de las colas del slice 2 siguiendo una redistribución para igualar sus DS 47
26. Experimento 2. Grado de satisfacción de las colas del slice 2 siguiendo una redistribución para igualar sus DS 47
27. Experimento 2. Evolución del peso de las colas del slice 2 50
28. Experimento 1. Resultados de la colas del slice 2 cuando se aplica algoritmo de redistribución proporcional a DS a nivel intra-slice 61

ÍNDICE DE TABLAS

1. Variables ADRR 15
2. Escenario de pruebas 23
3. Variables de la redistribución Intra-Slice 31
4. Variables de la redistribución Inter-Slice 38
5. Escenario de pruebas para la redistribución dinámica de quantum 41

LISTA DE ACRÓNIMOS

ACK: Acknowledgment / Reconocimiento

ADRR: Airtime Deficit Round Robin.

ADWRR: Airtime Deficit Weighted Round Robin.

AP: Access Point / Punto de acceso

CSMA/CA: Carrier Sense Multiple Access with Collision Avoidance / Acceso múltiple por detección de portadora con evitación de colisiones

IP: Internet Protocol / Protocolo de Internet

ISS: Infrastructure-Sharing Slicing.

MBR: Maximum Bit Rate / Tasa máxima de bits

MCS: Modulation and Coding Scheme / Esquema de modulación y codificación

QoS: Quality of Service / Calidad de servicio

RAN: Radio Access Network / Red de acceso radioeléctrico

RR: Round Robin

SDN: Software-Defined Networking / Redes definidas por software

SLA: Service Level Agreement / Acuerdo de nivel de servicio

STA: Estación

WLAN: Wireless Local Area Network / Red de área local inalámbrica

ANEXOS

I RR

En el esquema Round Robin (RR), las colas se recorren siguiendo un esquema cíclico. Cuando es el turno de una cola, esta envía un paquete si tiene alguno encolado. En caso de estar vacía, se pasa inmediatamente a la siguiente.

A continuación, en los algoritmos 7 y 8, se muestra la implementación del esquema RR como algoritmos de scheduling intra e inter slice, respectivamente.

Algoritmo 7 Inter-slice $s$cheduling Round Robin (RR)

```
1: s = 0
2: loop every
3:     Mientras la ocupación física del buffer del driver sea el umbral, espera
       #Envía paquetes del slice s si este no está vacío
4:     if Slice[s].empty == false then    ▷ Las colas de s tienen paquetes
5:         ALGORITMO INTRA-SLICE RR (Alg. 8)
6:     end if
       #Continúa con el siguiente slice
7:     if s < (N_slices - 1) then s++
8:     else s = 0
9: end loop
```

Como se muestra en el Algoritmo 7, el esquema inter-slice consiste en la visita cíclica y rotativa de los diferentes slices. Si llegado el turno del slice s, este no está vacío, es decir, alguna de sus colas tiene paquetes acumulados, estos se envían siguiendo también un esquema RR (líneas 4-6 del Alg.7). De este modo, se comienza en la cola del slice s que quedó pendiente en la última ronda ($Slice[s].Queue_index$), si estuviera vacía se pasa el turno hasta encontrar la primera con paquetes encolados (líneas 1-6 del Alg.8).

Al igual que en el esquema ADRR, existe la opción de elegir de manera selectiva e individual en cada cola, si los paquetes pueden ser o no agregados para formar una trama A-MSDU. En caso de permitirse agregación, se calcula el número máximo de paquetes que pueden agregarse en función de los que hay almacenados en la cola y del máximo establecido en configuración (línea 11 del Alg.8). Una vez se tiene calculado este número, se desencolan los paquetes, sea uno o varios (línea 12 del Alg.8), se construye la trama y se envía (líneas 13-14 del Alg.8).

Enviada la trama se pasa el turno a la siguiente cola no vacía (líneas 16-19 del Alg.8) hasta que se completa la ronda y se pasa al siguiente slice (líneas 20-22 del Alg.8 y 7-8 del Alg.7).

Como en el esquema ADRR, el proceso de scheduling es un proceso controlado. Antes de realizar cualquier acción con los paquetes, se garantiza que la ocupación física del buffer del driver sea inferior a un umbral predefinido, en nuestro caso de 10 paquetes. En caso de que no se cumpla, se espera antes de continuar (línea 3 del Alg.7 y línea 9 del Alg.8). Esto permite prevenir su desbordamiento y asegurar el cumplimiento de los requisitos de aislamiento y QoS. En este proyecto, no se puede acceder directamente a la ocupación del driver, por lo que se

emplea la interfaz auxiliar como mecanismo alternativo de monitorización. Al comparar el número de tramas, y sus números de secuencia, entre los paquetes enviados por la interfaz principal y los recibidos en la interfaz auxiliar, podemos obtener el número de paquetes que se encuentran almacenados en el driver.

Algoritmo 8 Intra-Slice para servir las colas del slice. Opción RR

```
    #Busca la primera cola de la ronda con paquetes que enviar
    ▷ Slice[s].Queue_index: Siguiente cola de la ronda
 1: q_ini = Slice[s].Queue_index
 2: i = q_ini

 3: for j = 0 to N_queues[s] do
 4:     if Queue[s, i].empty == false then break
 5:     else i++
 6: end for

    #Proceso de desencolado
 7: loop every
 8:     if Queue[s, i].empty == false then
 9:         Mientras la ocupación física del buffer del driver sea el umbral, espera

            #Opción de no agregación de paquetes
10:         npAggr = 1

            #Opción de agregación de paquetes
            ▷ Calcula el número de paquetes que pueden ser agregados (npAggr) basado en el máximo número de la agregación (valor de configuración) y el número de paquetes de la cola (A-MSDU)
11:         npAggr = Queue[s, i].MaxAggregationAllowed

12:         Queue[s, i].dequeue(npAggr)
13:         Construye la trama agregada con los paquetes desencolados
14:         Send(frame)
15:     end if

        #Continúa con la siguiente cola del slice s
16:     if i <(N_queues[s] - 1) then i++
17:     else i = 0
18:     end if
19:     Slice[s].Queue_index = i    ▷ Guarda el valor de la siguiente cola en la ronda

        #Si ha terminado la ronda, continúa con el siguiente slice
20:     if i == q_ini then
21:         break    ▷ Siguiente slice
22:     end if
23: end loop
```

Por otro lado, como se comentaba en el apartado 4.1.1, el esquema intra-slice RR puede combinarse con el inter-slice ADRR. Es por eso por lo que, en caso de que se escoja esta combinación, en el Algoritmo 8 se incluiría el cálculo del airtime de los paquetes enviados ($airtime[s,i] = Queue[s,i].airtime(npAggr)$), así como la respectiva actualización del contador de déficit del slice s ($Deficit[s] = Deficit[s] - airtime[s,i]$), siguiendo así el Algoritmo 9.

Algoritmo 9 Intra-Slice para servir las colas del slice. Opción RR

#Busca la primera cola de la ronda con paquetes que enviar
▷ $Slice[s].Queue_index$: Siguiente cola de la ronda
$q_{ini} = Slice[s].Queue_index$
$i = q_{ini}$

for j = 0 to $N_{queues}[s]$ **do**
  if $Queue[s,i]$.empty == false **then** break
  else i++
end for

#Proceso de desencolado
while $Deficit[s] > 0$ **do**
  loop every
    if $Queue[s,i]$.empty == false **then**
      Mientras la ocupación física del buffer del driver sea el umbral, espera

      #Opción de no agregación de paquetes
      $airtime[s,i]$.airtime(head packet)
      $npAggr$ = 1

      #Opción de agregación de paquetes
      ▷ Calcula el número de paquetes que pueden ser agregados (*npAggr)* basado en el máximo número de la agregación (valor de configuración) y el número de paquetes de la cola (A-MSDU)
      $npAggr$ = $Queue[s,i]$.MaxAggregationAllowed
      $airtime[s,i]$.airtime($npAggr$)

      $Queue[s,i]$.dequeue($npAggr$)
      Construye la trama agregada con los paquetes desencolados
      Send(frame)
      $Deficit[s]$ -= $airtime[s,i]$
      if $Slice[s]$.empty == false **then** $Deficit[s] = 0$ ▷ Siguiente slice
    end if

    #Continúa con la siguiente cola del slice *s*
    if i <($N_{queues}[s]$ - 1) **then** i++
    else i = 0
    end if
    $Slice[s].Queue_index = i$ ▷ Guarda el valor de la siguiente cola en la ronda

    #Si ha terminado la ronda, continúa con el siguiente slice
    if $i == q_{ini}$ **then**
      break ▷ Siguiente slice
    end if
  end loop
end while

II Reparto proporcional a los grados de satisfacción

Esta propuesta redistribuye los recursos sobrantes de manera proporcional al grado de satisfacción de las colas insatisfechas.

Su pseudocódigo, abajo expuesto, formaría parte del Algoritmo 5, empleando así las variables de la Tabla 3.

```
   #Opción Redistribuye el peso proporcionalmente a su insatisfacción
   ▷ Calcula la suma de todas las tasas insatisfechas.
73: if S_unsatisfied[s] = ∅ then
74:     if R_demanded[s,i] > MBR[s,i] para alguna i then
75:         Redefine S_unsatisfied[s] con el valor real R_demanded (repite líneas 1-7)
76:         if S_unsatisfied[s] sigue vacío then break
77:     end if
78: end if
79: total_DS = ∑_{∀i∈S_unsatisfied[s]} ds[s,i]
80: for cada cola i ∈ S_underutilized[s] ordenada de forma descendiente según su W^exp_ceded[s,i] do
        ▷ Se realiza el reparto proporcional a todas las colas insatisfechas del slice s
81:     for cada cola j ∈ S_unsatisfied[s] do
82:         w = ⌊W^exp_ceded[s,i] * ds[s,i]/total_DS[s]⌉
83:         W^exp_ceded[s,i] -= w
84:         W_ceded[s,i] += w and W_ceded[s,i,j] = w
85:         W_added[s,j] += w and W_added[s,j,i] = w
86:         W[s,i] -= w
87:         W[s,j] += w
88:         ds[s,j] += IDS^exp[s,j] * w
89:         if ds[s,j] ≥ 1 then Elimina la cola j del grupo S_unsatisfied[s]
90:     end for
91: end for
```

Se realiza la suma de todos los grados de satisfacción ($total_DS[s,i]$) de las colas insatisfechas del slice ($S_{unsatisfied}[s]$), para poder realizar correctamente las proporciones (línea 79). En caso de que todas las colas estén satisfechas en función de su $MBR[s,i]$, se calcula el grado de satisfacción con su $R_{demanded}[s,i]$ real (líneas 73-78).

Obtenido este dato, las colas infrautilizadas del slice ($S_{underutilized}[s]$), ordenadas en función de la cantidad de recursos que tienen para ceder ($W^{exp}_{ceded}[s,i]$), cederán proporcionalmente a todas las colas insatisfechas (líneas 80-90).

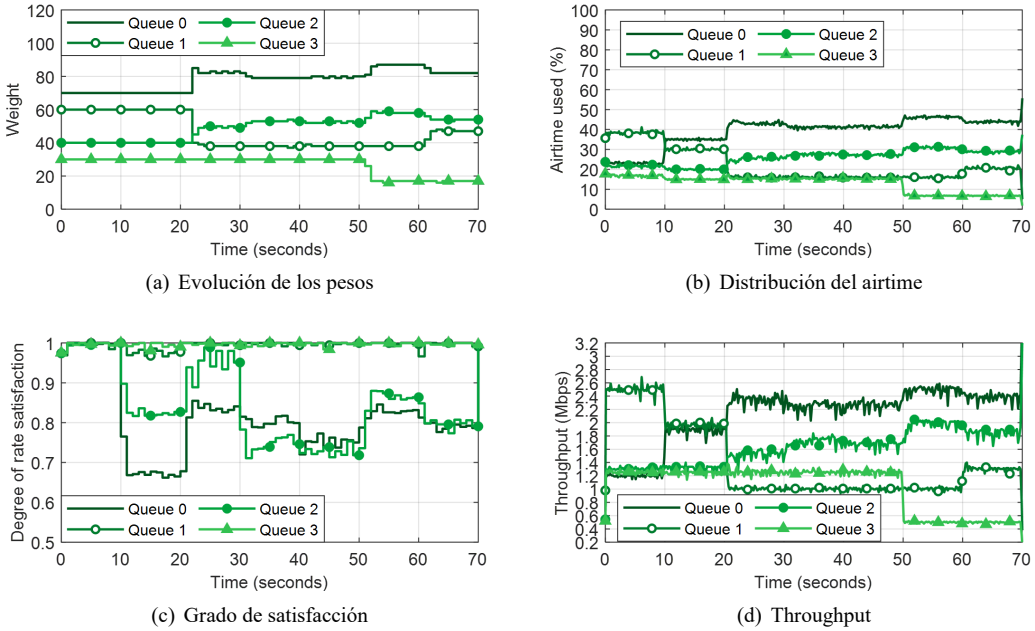
Esta proporción se calcula como la relación entre la tasa insatisfecha de la cola a recibir ($ds[s,i]$) y el total de la suma (línea 82). De esta forma, se garantiza que todas las colas insatisfechas reciban recursos, pero, a su vez, que aquellas que más lo necesiten reciban más.

Cabe destacar que el algoritmo descrito es el inter-slice, pero el intra-slice se puede deducir directamente al cambiar las variables por las de la Tabla 4. De modo que la actualización se realiza directamente sobre los valores de quantum ($Q[s]$) en vez de los pesos ($W[s,i]$).

Ahora se repite el experimento 1 de la Tabla 5 para verificar el funcionamiento de la redistribución según este algoritmo a nivel intra-slice, por lo que a nivel inter-slice se mantiene un esquema ADRR con pesos fijos.

Recordemos que en esta prueba las demandas de tráfico en los slices 0 y 1 son constantes y lo suficientemente altas como para evitar la redistribución de airtime entre los slices, así como en sus colas. De este modo, se observa que la distribución del airtime se mantiene constante en la ocupación garantizada por los valores nominales, tanto para los slices, como para las colas (los resultados son idénticos a los de la Fig.13).

A nivel intra-slice nos fijamos de nuevo en el slice 2, donde los resultados que vamos a analizar muestran la evolución del peso de sus colas (Fig.28a), la distribución del airtime (Fig.28b), los grados de satisfacción alcanzados (Fig.28c) y el throughput (Fig.28d). Estos se comparan con los obtenidos en el apartado 4.2.4.2 para el resto de los algoritmos.



(a) Evolución de los pesos

(b) Distribución del airtime

(c) Grado de satisfacción

(d) Throughput

Figura 28: Experimento 1. Resultados de la colas del slice 2 cuando se aplica algoritmo de redistribución proporcional a DS a nivel intra-slice

El comportamiento es idéntico en los periodos en los que no se lleva a cabo el proceso de la redistribución de pesos (ver estos periodos en las Fig.28a,b,c,d), comentados ya en el apartado citado, por lo que aquí se van a destacar aquellos periodos que difieren de lo anterior.

A partir del segundo 20 la demanda de tasa de la cola 0 desciende, por lo que pasado unos segundos en los que emplea recursos para vaciar los paquetes que tenía almacenados, pasa a estar infrautilizada, comenzando así la redistribución. En el periodo de 20 a 50 segundos, esta es la única cola infrautilizada, estando la cola 3 satisfecha pero sin exceso y las colas 0 y 2 insatisfechas. Es por ello que se observa que durante este periodo que la cola 1 cede pesos a

estas dos (Fig.28a), ya que el algoritmo cede a todas las colas insatisfechas proporcionalmente a su necesidad. Esto se ve reflejado también al observar el grado de satisfacción alcanzado en las colas insatisfechas (DS, Fig.28c), ya que cuando uno aumenta o disminuye también lo hace el otro.

En el segundo 50 la demanda de la cola 3 disminuye, por lo que, tras unos segundos de vaciado de la cola, esta pasa a ser otra cola infrautilizada. De este modo, se observa como los pesos de la cola 3 se reparten proporcionalmente a su DS entre las colas 0 y 2.

También es interesante observar como el reparto proporcional a los DS permite que el throughput de las colas a las que se cede tenga un comportamiento similar cuando se mantienen las demandas constantes (colas 0 y cola 2, Fig.28d).

Es importante resaltar que esta redistribución no afecta a las garantías de las colas 0 y 3 que en todo momento transmiten lo demandado (DS, Fig.28d).

A partir de estos resultados se puede concluir que la adaptación local de los pesos siguiendo esta propuesta es capaz de modificar la reasignación natural de ADRR para cumplir con el requisito específico de QoS. Esto manteniendo siempre las garantías de las colas que ceden.

No obstante, también se observa que este criterio no es tan relevante en una aplicación real, como lo son el seguimiento de una prioridad o el tratar de conseguir que los grados de satisfacción sean similares, por lo que finalmente se descartó como una de las opciones de redistribución propuestas.