ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

OPTIMIZACIÓN CONJUNTA DEL NIVEL DE SPLIT Y SCHEDULING EN REDES 5G

(Joint optimization of functional split level and scheduling in 5G networks)

Para acceder al Titulo de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Víctor González Carril

Julio - 2019

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Víctor González Carril
Directores del TFG: Luis Francisco Diez Fernández y Ramón Agüero Calvo
Título:Optimización conjunta el nivel de split y scheduling en redes 5G
Title: Joint optimization of functional split level and scheduling in 5G networks

Presentado a examen el día: 11 de julio de 2019

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:

Presidente (Apellidos, Nombre):	Mauro Lomer Barboza
Secretario (Apellidos, Nombre):	Ramón Agüero Calvo
Vocal (Apellidos, Nombre):	Marta García Arranz

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

Agradecimientos

Me gustaría expresar mis agradecimientos al director de este trabajo, Ramón Agüero, así como a Luis Francisco Diez por acompañarme a lo largo de todo el proyecto y ayudarme en las fases más complicadas.

A lo largo de los últimos meses, varios profesores e investigadores del Grupo de Ingeniería Telemática han influido positivamente en el trabajo y en mi experiencia a la hora de llevarlo a cabo, lo cual les agradezco.

Por supuesto, mis compañeros de clase han hecho posible pasar buenos ratos a lo largo de todo el grado y durante el desarrollo de este trabajo. Les doy las gracias ya que una mentalidad positiva es una parte clave a la hora de mantener la motivación en el día a día.

Por último, agradezco a mis padres el apoyo ya no solo durante el desarrollo del trabajo, sino a lo largo de todos estos años en los que han confiado en mis decisiones.

Resumen

A día de hoy, parece generalmente aceptado que las técnicas de virtualización serán una pieza fundamental en las futuras tecnologías 5G. Sin embargo, soluciones totalmente centralizadas como Centralized Radio Access Network (C-RAN) que llevan todo el cómputo a una máquina central podrían no ser viables, debido a las necesidades de los enlaces que conectan estos recursos con las estaciones base y al gran coste que supone el cambio de arquitectura. Para solucionar este problema nacen los *splits* funcionales, que equilibran la usabilidad y el rendimiento permitiendo dividir el procesamiento entre las estaciones base y los recursos en la nube. Este concepto evoluciona a los *splits* funcionales flexibles, que son capaces de adaptarse en tiempo real a las necesidades de la red. No es algo novedoso el interés en esta idea, pero se le ha prestado poca atención a su interacción con el *scheduling*, o el orden en el que se procesan las tramas. En este trabajo se analiza su comportamiento conjunto, así como las estrategias que permiten minimizar el retardo del tráfico. Se va a comparar un modelo de optimización conjunta con soluciones parciales, que fijan una de las dos variables para permitir cálculos más rápidos. En general, los resultados muestran que en ciertas situaciones fijar el *scheduling* resulta en un comportamiento similar al de la optimización conjunta. Además, se ha observado que es más eficiente optimizar los *splits* funcionales fijando el *scheduling* que optimizar el *scheduling* fijando los *splits* funcionales.

Abstract

Nowadays, it is well known that network function virtualization will be a key enabler to meet the stringent requirements of 5G networks. However, fully centralized approaches, such as Cloud Radio Access Network (C-RAN), might not be feasible, considering the particular needs of the fronthaul links and the large cost of implementing such architectural shift. In this sense, functional splits bring a practical solution, which trades off performance and practicability. This concept evolves into flexible functional split, which is able to adapt to the needs of the network in real time. In spite of the growing interest on flexible functional split, little attention has been paid to the interaction of split selection and scheduling. In this document, joint strategies that minimize traffic delay are analyzed. The global optimum solution will be compared with partial optimizations, that can be more suitable in practical implementations, using different scenarios. In general, fixed scheduling behaves alike the global optimum. Furthermore, it will be observed that it is more efficient to optimize the split degree for fixed scheduling setups, than deciding a scheduling policy for a particular split configuration.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En las redes de quinta generación, una de las características que se pretende implementar es el paradigma Network Function Virtualization (NFV), a partir de las técnicas Software Defined Networking (SDN) o redes definidas por software. Lo que esto implica es la posibilidad de crear las redes de forma virtual en máquinas potentes en la nube, de tal forma que se compartan los recursos entre varios sistemas y de esta forma se incremente la eficiencia de los mismos. En redes celulares previas, como 4G, se propusieron técnicas descentralizadas que funcionan con un número limitado de dispositivos y conexiones. Esto es debido a que el procesamiento de la información se lleva a cabo directamente en las estaciones base, y por tanto elementos como la escalabilidad se ven perjudicados. Sin embargo, los estrictos requisitos de las redes 5G precisan una mayor coordinación entre los elementos de la red, lo cual solo se puede obtener con una entidad central de mayor potencia, o lo que es lo mismo, con soluciones más centralizadas.

Para ello, una posibilidad es separar las funcionalidades de la red de acceso radio o Radio Access Network (RAN), de tal forma que parte de ellas están virtualizadas y centralizadas mientras que el resto se mantienen en el punto de acceso, cercanas al usuario. Esto permite que el procesado de la información se lleve a cabo de manera coordinada entre la estación base o Remote Radio Head (RRH) y la entidad central, denominada Base-Band Unit (BBU). Se pueden entonces considerar arquitecturas totalmente centralizadas, como Centralized-RAN, donde el punto de acceso o RRH solo se encarga de funciones Radiofrecuencia (RF) al nivel más básico y por tanto el procesado se realice casi enteramente en la BBU. No obstante, esta aproximación requiere capacidades muy elevadas de transmisión de los enlaces de *fronthaul* entre las RRHs y su correspondiente BBU, lo cual puede tener costes excesivos en situaciones reales. Por ello, tanto en el entorno académico como en el industrial y en organismos de estandarización [1][2] se está trabajando para definir soluciones que permitan una selección flexible del nivel de centralización, más conocido como *split* funcional. Un *split* funcional flexible implica una mayor adaptabilidad, donde en función del tráfico, de la red o de las necesidades de los usuarios en cada momento, el procesado de la información tiene lugar con un mayor o menor nivel de centralización.

Este cambio de paradigma no solo supone una reducción notoria en el coste, sino

que además añade la posibilidad de permitir a las RRHs cooperar entre ellas de forma sencilla. Debido a los requisitos de la arquitectura 5G, así como al gran incremento que se espera en la densidad de las redes de acceso, esta cooperación se convierte en una necesidad. Cuando se virtualizan las funciones de red (NFV), una decisión clave a tener en cuenta es el *split* funcional a utilizar, es decir qué funcionalidades se llevan a los recursos compartidos en la BBU y cuáles se mantienen cerca del usuario, en la RRH. Por otra parte, entra en juego el orden en que las tramas se transmiten de la BBU a las diversas RRHs que gestiona, también conocido como *scheduling*. Este concepto, de la misma forma que el *split* funcional, tiene un elevado impacto en el retardo de las tramas, que necesita ser lo más bajo posible para cumplir con los estrictos requisitos de las tecnologías de comunicación de quinta generación.

En este trabajo se analiza la interacción de estos dos elementos al mismo tiempo, en una arquitectura basada en *splits* funcionales flexibles. Se toma como base el trabajo realizado en [3], donde se da una descripción teórica del problema. En resumen, llevar a la BBU más funcionalidades implica una mayor carga computacional y una densidad variable y elevada de tráfico que se transmite continuamente a través de los enlaces de *fronthaul*, que conectan la BBU con la RRH correspondiente. En el trabajo mencionado, se habla de la complejidad del problema de optimización conjunta así como la posibilidad de simplificarlo fijando una de las dos variables, selección de *split* o *scheduling*. Sin embargo, estos problemas solo se plantean teóricamente y no se da ninguna solución ni resultado práctico. En este trabajo se aborda la implementación de las soluciones planteadas en [3] y se estudia su rendimiento en escenarios reales.

1.1 Trabajo relacionado y aportaciones

Como ya se ha mencionado, NFV es considerado un habilitador para el desarrollo de arquitecturas que permitirán cumplir con los estrictos requisitos de la tecnología 5G. La filosofía detrás de este concepto es mover algunas de las funcionalidades que tradicionalmente se situaban en la estación base a un sistema virtualizado, potencialmente en la nube, de forma que se simplifique la coordinación entre elementos de acceso. Uno de los principales problemas de la centralización total es la alta demanda en los enlaces fronthaul, que solo se puede satisfacer con fibra óptica [4][5]. Como ya se ha comentado, la propuesta actual más interesante consiste en un diseño alternativo de estos enlaces [6], donde existe una separación en *splits* funcionales de tal forma que solo una parte de las operaciones de procesado se centraliza. Un análisis más completo de los *splits* funcionales actualmente propuestos para la arquitectura 5G se presenta en [7] y sus referencias.

Avanzando un paso más, nace la idea de utilizar *splits* funcionales flexibles que se adapten a cada situación en particular, ya sea en base a los requerimientos de retardo, la densidad de tráfico o las condiciones de los enlaces. En [8] y [9] se describen las principales características de este concepto, aunque también se va a profundizar en ello en este trabajo. Por otra parte, la implementación de este concepto en una arquitectura

5G se plantea en [10], y se lleva a cabo en un entorno de laboratorio en [11].

Existen diferentes trabajos en los que se han estudiado las implicaciones de la selección flexible del *split*, así como su interacción con otras técnicas de red. Trabajos como [12] y [13] proponen soluciones basadas en optimización energética, mientras que la combinación con esquemas de gestión de transporte en redes ópticas se analiza en [14] y [15]. Además, algunos estudios como [16] y [17] han considerado la interacción entre *splits* funcionales flexibles y *scheduling*. Sin embargo, esta línea de investigación no ha recibido demasiada atención a pesar de ser un concepto clave, y es la principal contribución de este documento. A raíz de ello, las aportaciones de este trabajo son:

- Se provee una implementación de los modelos que se exponen brevemente en [3] para minimizar el retardo.
- Se evalúan los resultados en diversos escenarios, y se analizan su comportamiento y su rendimiento en situaciones prácticas realistas.
- Dada la complejidad del problema de optimización de selección de *split* y *scheduling*, su aplicación práctica podría no ser viable en algunos escenarios. Por ello se analiza el rendimiento de optimizaciones parciales en las que uno de los parámetros es conocido.

1.2 Estructura

Tras la introducción realizada en este capítulo, donde se han presentado la motivación del trabajo, los conceptos principales y trabajos previos relacionados, el resto del documento sigue la estructura que se comenta a continuación. En el capítulo 2 se describe la evolución de las redes móviles con el fin de contextualizar la situación actual, y se introducen los conceptos básicos con los que se va a trabajar más adelante. En el capítulo 3 se presentan los modelos que se van a utilizar y a analizar, con detalles y explicaciones a partir de su pseudocódigo y un ejemplo sencillo. El capítulo 4 contiene una explicación de los escenarios en los que se han ejecutado los modelos comentados anteriormente, y una exposición detallada de los resultados obtenidos con análisis y comparativas de los mismos. Por último, en el capítulo 5 se recapitulan los resultados y las conclusiones, y se enumeran las publicaciones realizadas a partir de dichos resultados. Finalmente, se comentan las diferentes líneas de trabajo que surgen de este proyecto.

Capítulo 2

Conceptos previos

Con el objetivo de contextualizar la situación actual, en este capítulo se va a analizar la evolución de las redes móviles en los últimos años, haciendo hincapié en los elementos que afectan a este trabajo. Posteriormente se va a explicar en qué consiste la quinta generación, describiendo los requisitos impuestos a la tecnología. A raíz de estos requisitos surgen conceptos clave para este trabajo, que se van a describir en las últimas secciones.

2.1 Evolución de las redes móviles

2G - Global System for Mobile communications - GSM

En la segunda generación de redes celulares (estudio en detalle [18]), más conocida como Global System for Mobile communications (GSM), se ven por primera vez grandes avances en las comunicaciones móviles a nivel global como el uso de una modulación digital. Las bandas de frecuencia asignadas a esta tecnología son de 25MHz, divididos en 128 canales que dan cobertura hasta a 8 usuarios simultáneamente, lo que supone un ancho de banda de 200KHz por canal. Esta tecnología hace uso de un esquema de duplexado Frequency Division Duplex (FDD), a la par que utiliza Time Division Multiple Access (TDMA) para gestionar el acceso múltiple. Aunque las redes de segunda generación están principalmente diseñadas para soportar servicios de voz, también han sido ampliamente utilizadas para dar servicios de datos que no requieren mucha capacidad, mediante el estándar General Packet Radio Service (GPRS). A pesar de que las capacidades para el transporte de datos que proporcionan las redes 2G están lejos de satisfacer las demandas de muchos de los servicios actuales, su arquitectura presenta un diseño clave para las redes futuras, la centralización. En GSM, existe una separación entre la Base Transceiver Station (BTS) y la Base Station Controller (BSC). La primera de ellas es la parte de la estación base a la que se conectan los usuarios directamente, mientras que la segunda posee una capacidad de cómputo más elevada y es capaz de llevar a cabo casi la totalidad de las operaciones. La filosofía a seguir es la siguiente: los usuarios están conectados a las BTSs y envían la información hacia ellas,

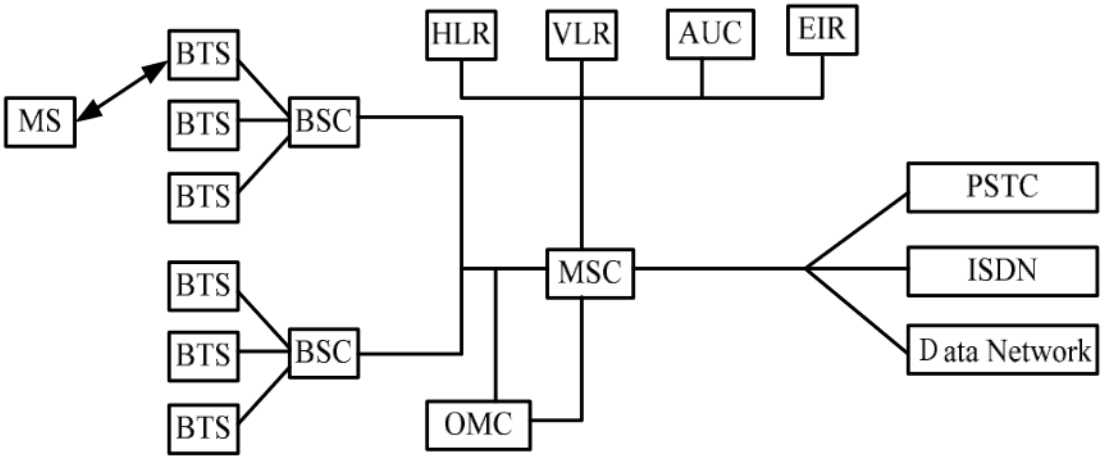

Figura 2.1: Estructura GSM

pero en lugar de procesarse en dichas estaciones, varias de ellas están conectadas a una BSC que se encarga de llevar a cabo la gestión y el control de las mismas, así como el procesado del tráfico. Con este esquema centralizado se consigue que las BTSs tengan un coste relativamente bajo ya que no necesitan una gran capacidad de cómputo, y que, gracias a la BSC, sean capaces de cooperar entre ellas para mejorar su rendimiento global. Como se aprecia en la Figura 2.1, varias BSCs están conectadas a su vez a una Mobile Switching Center (MSC) y aparecen más funcionalidades en la parte interna de la red, pero no se van a detallar ya que es el concepto de centralización el que conecta directamente con este trabajo.

3G - Universal Mobile Telecommunications System - UMTS

En lo que se refiere a la tercera generación de redes celulares, centrando la atención en Universal Mobile Telecommunications System (UMTS) [19], la idea detrás de la arquitectura de la red no cambia en el aspecto que concierne a este trabajo. En este caso, el usuario, ahora denominado User Equipment (UE), se conecta a una estación denominada Node B que realiza operaciones más sencillas a nivel físico, mientras que tras ella se encuentra el Radio Network Component (RNC) que controla y gestiona los recursos a un nivel superior, encargándose de uno o varios Node B. En la Figura 2.2 se observa este esquema, en el que se repite la componente de centralización y donde, de nuevo, existen funcionalidades a más alto nivel no relacionadas con el alcance de este trabajo. Siendo la estructura similar, los avances más significativos en UMTS respecto a GSM se encuentran en su tecnología de acceso Wideband Code Division Multiple Access (WCDMA), con CDMA2000 como su competidor, que posee una mayor eficiencia espectral y permite la transmisión de datos a velocidades mayores. Además hace uso de bandas de frecuencia en torno a los 1900 MHz y 2200 MHz que permiten una velocidad de trama de 3.84 Megachips por segundo, cálculo que se obtiene a partir

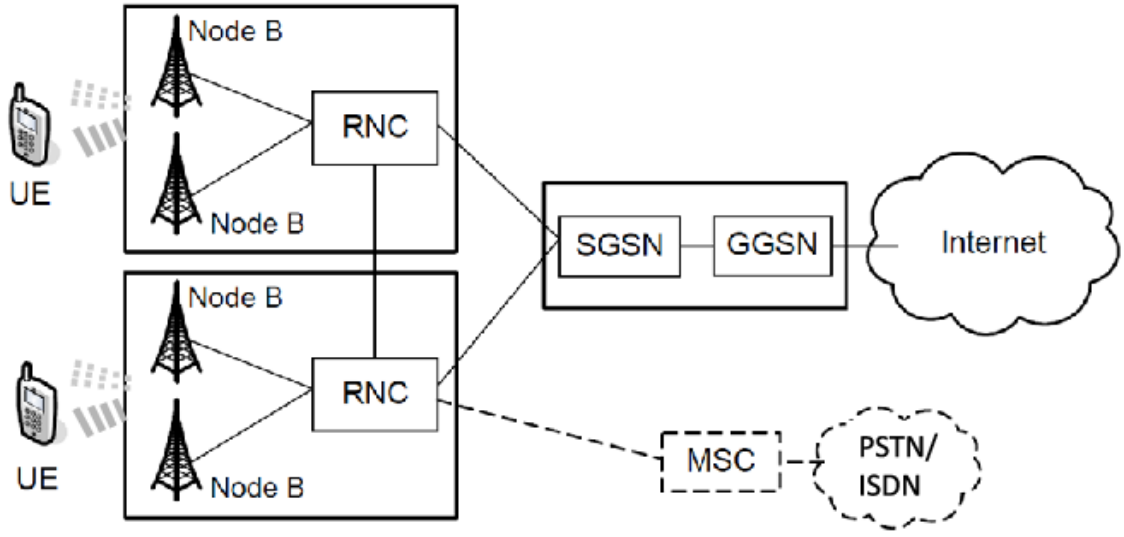

Figura 2.2: Estructura UMTS (fuente: [20])

de la estructura de las tramas. Cada trama mide 10 milisegundos y se compone de 15 slots, donde cada slot incluye 2560 chips. La tasa de chips definida en UMTS se relaciona directamente con la tasa de bit, a través del Spreading Factor (SF). El SF a utilizar depende de cada usuario y de cada caso particular, y se utiliza para ensanchar la señal en el dominio de la frecuencia y poder enviar información referida a varios usuarios al mismo tiempo y en el mismo canal. Utilizando un código, el receptor es capaz de des-ensanchar la información de un usuario en concreto y dejar el resto de señales ensanchadas, de tal forma que se recupera la secuencia de bits original y el resto de señales no interfiere debido a que, al estar ensanchadas, su potencia es mucho menor.

4G - Long Term Evolution - LTE

La cuarta generación [21] introduce un cambio de paradigma en el esquema de centralización, con la tecnología Long Term Evolution (LTE). Aprovechando la mejora de la tecnología y su reducción en coste, y considerando el gran incremento en la densidad de tráfico debido al comienzo de las comunicaciones Machine-To-Machine (M2M) y Device-To-Device (D2D), se comienza a pensar en llevar la inteligencia al extremo de la red descargando de tráfico y de procesamiento tanto a la parte interna de la red como a los enlaces. Como se representa en la Figura 2.3, aparecen unos elementos denominados enhanced/evolved Node B (eNB), que implementan los Node B tratados en UMTS y al mismo tiempo el plano de control y gestión que previamente se realizaba en el RNC. Este esquema también tiene claras desventajas, que se van potenciando a medida que se incrementa la demanda a lo largo de los últimos años, como la escasa escalabilidad o la limitada cooperación entre los eNBs. La tecnología de acceso evoluciona una

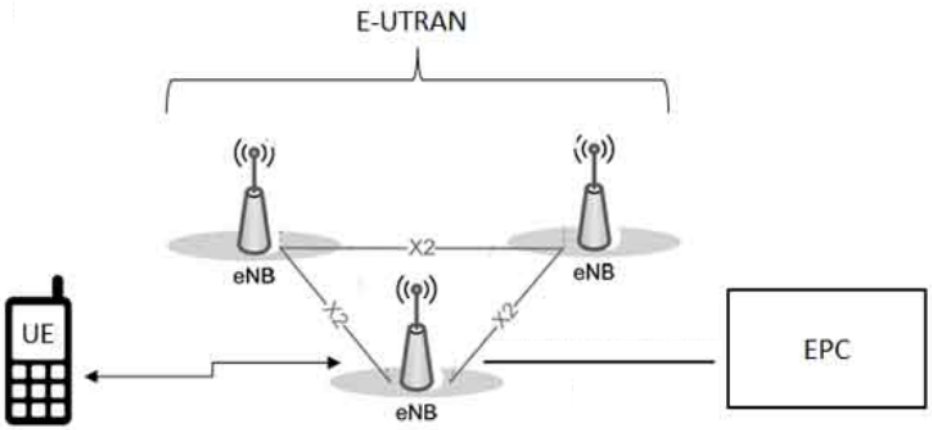

Figura 2.3: Estructura LTE (fuente: [22])

vez más, a Orthogonal Frequency Division Multiple Access (OFDMA), que reparte las portadoras de manera ortogonal en cada canal y asigna una o varias portadoras a cada usuario, garantizando así una calidad de servicio a medida para cada caso. La posición de las sub-portadoras se puede adaptar en tiempo real en función de las condiciones del canal o de la demanda, siendo una tecnología más flexible y con una mayor eficiencia espectral. Las bandas de frecuencia utilizadas se sitúan alrededor de 750MHz y de 2550MHz. LTE es la tecnología contemporánea junto con LTE Advanced, aunque no por ello se han abandonado las generaciones anteriores. No obstante, el incremento masivo de las comunicaciones M2M y el Internet de las cosas (IoT) provocan que se quiera dar el salto a la quinta generación de redes móviles con unos requerimientos más exigentes.

2.2 La quinta generación

De acuerdo a estudios como [23] o [24], las redes celulares de quinta generación son el objetivo a cumplir para 2020, con diversos retos respecto a la anterior generación. Se busca una red inalámbrica sin límites, no solo con mayor velocidad y capacidad sino que además solucione problemas de coste, eficiencia energética, escalabilidad y adaptabilidad frente al enorme incremento de las comunicaciones que ya se han comentado: IoT, M2M... Las redes 5G están definidas por consenso con una serie de parámetros y requisitos, véase [25]. Comparando con LTE-Advanced:

- Capacidad hasta 1000 veces superior.
- Velocidad de los datos entre 10 y 100 veces superior.

- Latencia de extremo a extremo inferior a 5 milisegundos.
- Número de comunicaciones simultáneas muy superior, del orden de 100 veces.
- Un coste sostenible y una elevada eficiencia energética, así como una calidad de servicio consistente.

Es importante aclarar por qué es necesario un incremento tan sustancial respecto a la anterior generación, si desde el punto de vista del usuario la tecnología actual parece funcionar perfectamente. Todos estos requisitos no son en vano, sino que responden a unos objetivos no tan tangibles, como la posibilidad de compartir cualquier tipo de información desde cualquier lugar y en cualquier momento con todo el mundo, que es uno de los grandes retos de la IoT. Otro de los motivos por los que se imponen unos requisitos tan exigentes es el enorme incremento en la cantidad de dispositivos que tienen conectividad a Internet. Los slots de tiempo disponibles actualmente, así como el ancho de banda que se utiliza en la cuarta generación, no son suficientes para albergar las conexiones simultáneas que se van a demandar en los próximos años. Uno de los requisitos estándar es conseguir una latencia (o retardo) máximo de 5 milisegundos para cualquier tipo de comunicación. Con la tecnología LTE, el procesado de las tramas se realiza completamente en los eNB, que son a su vez las estaciones a las que el usuario se conecta directamente. Sin embargo, los recursos de cada eNB son limitados y esto puede acarrear problemas en situaciones de alta densidad de tráfico, por no mencionar que cumplir unos requisitos estrictos de latencia puede ser un desafío insuperable con esta arquitectura. Para ello se propone una arquitectura que se encuentra a medio camino entre el enfoque centralizado de 2G/3G y la solución plana de 4G.

2.3 Cloud-RAN y split funcional

En la arquitectura definida para 5G, los usuarios se conectarán a una unidad denominada RRH, pero las funciones de la BBU se encuentran virtualizadas y potencialmente separadas. Las RRHs estarán conectadas a la BBU por un enlace que recibe el nombre de *fronthaul*. En la arquitectura 4G, este enlace es un cable que conecta dos partes coexistentes en la misma estación (eNB). En 5G la localización física de ambos elementos ya no es la misma, con lo cual el *fronthaul* puede ser implementado de maneras muy diversas: cables de cobre, fibra óptica de sílice, el medio inalámbrico, etc. Este hecho tiene la ventaja de la adaptabilidad, ya que cada caso puede ser totalmente diferente en función de las necesidades. Sin embargo, el modelado de las arquitecturas se complica ya que el tipo de enlace y su capacidad se convierten en un parámetro más a tener en cuenta.

Para cumplir con los requisitos de las redes celulares 5G, es necesario realizar una serie de cambios y mejoras que permitan un enorme incremento en las capacidades. En lugar de definir una tecnología de acceso fija (como lo eran WCDMA o OFDMA) se trata de integrar los esquemas de las generaciones previas y flexibilizar la decisión

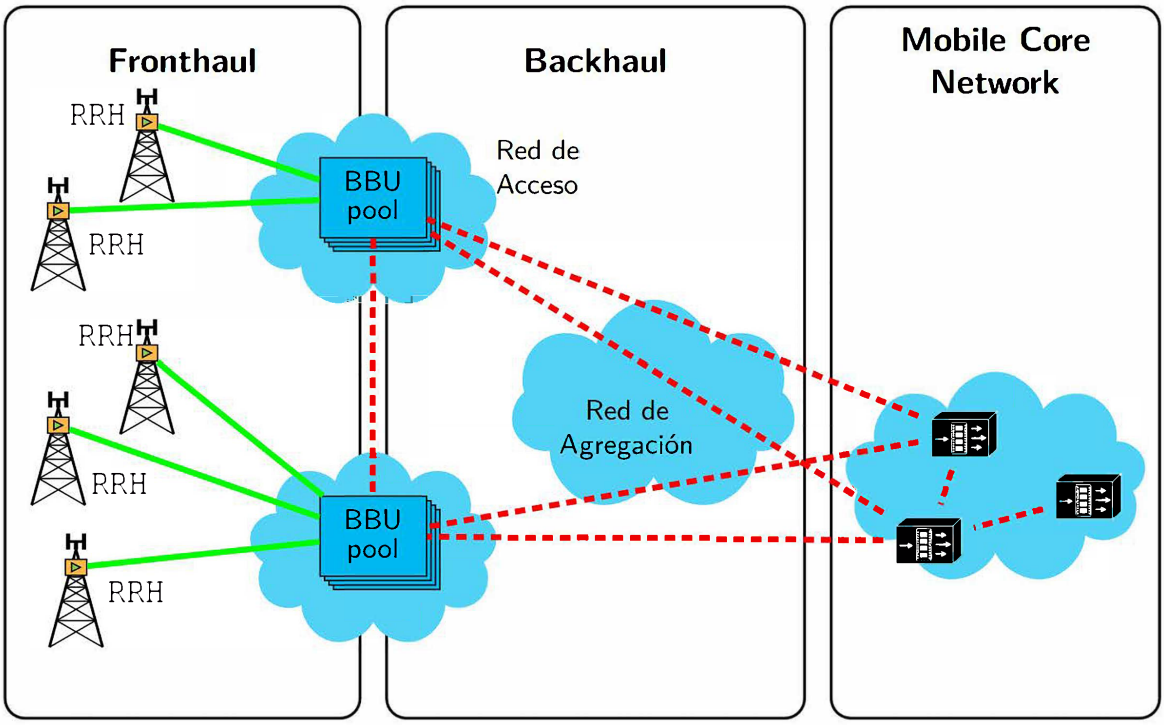

Figura 2.4: Arquitectura *Cloud-RAN* (fuente: [26])

en función de las necesidades, haciendo uso de multiplexación en tiempo, frecuencia, espacio, código, o nuevas aproximaciones como se analizan en [27]. La zona del espectro a utilizar, para cumplir con el requisito de velocidad de transmisión, ha de situarse en las bandas cuya longitud de onda sea del orden de los milímetros; en frecuencia esto es el rango de 3 a 300 GHz, incluyendo la posibilidad de desviar tráfico a bandas sin licencia. Adicionalmente, se va a hacer uso de un esquema Cloud-Radio Access Network (*Cloud-RAN* o C-RAN). Cabe destacar que el uso de las siglas C-RAN puede dar lugar a equivocación, puesto que se corresponden tanto con *Cloud-RAN* como con *Centralized-RAN*. Para evitar lugar a dudas, se emplea de aquí en adelante el nombre completo. El caso de *Centralized-RAN* se ha explicado previamente, y se corresponde con una red de acceso radio centralizada como la utilizada en 2G o 3G. El uso del esquema *Cloud-RAN* implica que la BBU a la que están conectadas una serie de RRHs no es necesariamente una máquina, sino que pueden ser varias actuando como un recurso común. Un esquema simplificado de esta arquitectura se muestra en la Figura 2.4. Esto rompe con las arquitecturas de RAN centralizada (segunda generación) o RAN distribuida (cuarta generación), que se han visto en apartados previos. *Cloud-RAN* permite un esquema más flexible en el que los recursos son accedidos por las distintas RRHs en función de las necesidades en cada momento. Aunque en este trabajo no se aborden, cabe indicar que la implementación de arquitecturas *Cloud-RAN* se basa en técnicas de NFV o virtualización de las funciones de red. Esto quiere decir que los dispositivos hardware como routers o balanceadores de carga que se encargan de

funciones relacionadas con la red, se sustituyen por dispositivos equivalentes definidos por software que se ejecutan de manera virtual en la nube. Este concepto se relaciona directamente con *Cloud-RAN*, ya que son los entornos de nube los que permiten llevar las diversas funcionalidades a uno o varios servidores externos y separados de la RRH.

Separación de funciones

En un esquema con mayor nivel de centralización, el procesamiento de la información se lleva a cabo en una localización central conectada a varias RRHs y con una gran cantidad de recursos que le permita soportar elevadas densidades de tráfico. Sin embargo, la total centralización de los recursos también tiene considerables desventajas, como el alto coste de implementar un cambio tan brusco o la gran demanda que supondría para los enlaces que conectan la BBU con las diversas RRHs. Lo que se busca en una arquitectura de quinta generación es encontrar un balance óptimo entre ambas soluciones, donde las RRHs sean capaces de realizar procesamiento por sí mismas pero siempre puedan contar con una capacidad superior en la BBU a la que se encuentran conectadas, para realizar operaciones más costosas. Una solución práctica y viable a este problema es la aparición del concepto de *split* funcional. Un *split* funcional se puede definir como una separación de las funcionalidades entre la BBU y las RRHs de tal forma que se repartan operaciones clave como el procesamiento de las tramas. Hacer que esta separación sea adaptable, introduciendo el concepto de *splits* funcionales flexibles, significa que la interacción entre ambas sea ajustable, o en cierto modo inteligente: dependiendo de las características de la red y de las tramas que requieran un procesamiento en un momento dado, la red debe ser capaz de encontrar la distribución óptima de funciones entre la BBU y las RRHs para que el retardo conjunto de dichas tramas sea mínimo. Los posibles *splits* funcionales ya están definidos en la actualidad, teniendo en cuenta las operaciones críticas a realizar y las separaciones viables entre las mismas. Se muestra un esquema en la Figura 2.5, donde aparecen simplificados seis bloques y los posibles *splits* funcionales se representan con las líneas discontinuas (A)-(E). El orden de dichos bloques se corresponde con un enlace *uplink*, es decir, una trama que se genera en el usuario y se dirige hacia la red celular. El ejemplo se puede replicar para un enlace *downlink* simplemente invirtiendo el orden de las operaciones. El esquema siguiente se basa en el estudio realizado en [28]:

- Bloques de la Figura 2.5 - Capa física
  - *RF and AD*: Demodulación de la señal a banda base y conversión analógico-digital.
  - *CP and FFT*: Elimina el prefijo cíclico, que se utiliza como robustez ante la interferencia entre símbolos y como elemento auxiliar para la convolución circular. Esto permite tomar la FFT o transformada de Fourier rápida.
  - *RE Demap*: Permite que solamente los Resource Elements (REs) que son utilizados por el usuario pasen al siguiente bloque.

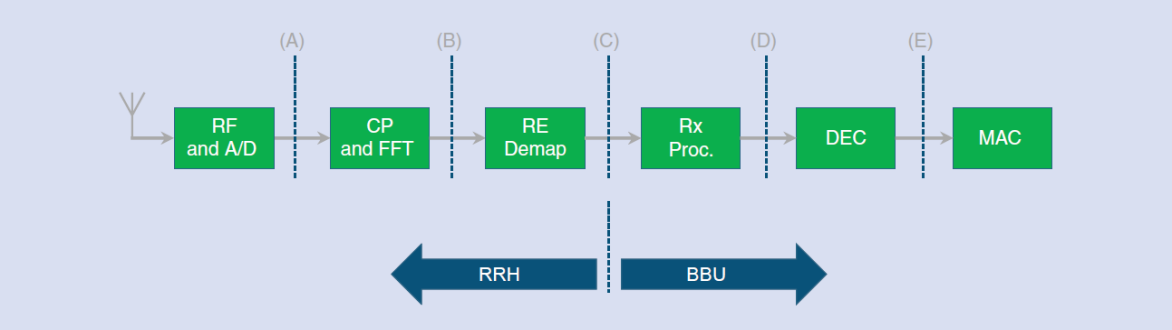

Figura 2.5: *Splits* funcionales (fuente: [29])

  - *Rx Proc*: Se realizan las funciones de ecualización del canal en el dominio de la frecuencia, transformada de Fourier discreta inversa y procesado MIMO.
  - *DEC*: Corrección de errores (FEC) y decodificación, de tal manera que se obtiene la trama a nivel de capa de enlace.

- Capa de enlace
  - *MAC*: Media Access Control, demultiplexa los bloques recibidos de la capa física en canales lógicos que envía a la funcionalidad superior, RLC. También se encarga de gestionar la calidad de servicio requerida en cada sub-portadora. Es uno de los bloques más críticos a la hora de decidir el *split* funcional.
  - *RLC*: Radio Link Control, traduce la información recibida a Protocol Data Units (PDUs) y las ordena según la secuencia de datos correspondiente. A su vez, es responsable de las retransmisiones a nivel de enlace para asegurar una mayor robustez.
  - *PDCP*: Packet Data Convergence Protocol, realiza las operaciones más cercanas al nivel de red como son la compresión de cabeceras, cifrado y verificación, gestión de *timeout* y de *handover*.

- Capa de red
  - *RRC*: Radio Resource Control, se encarga de las operaciones del plano de control.
  - *IP*: Internet Protocol, protocolo por excelencia en el plano de usuario dentro de la capa de red.

2.4 Tramas y scheduling

Antes de hablar del esquema con el que se va a trabajar, es importante aclarar el concepto de trama. Se trata de un conjunto de bytes que pueden englobar los mensajes

de distintos usuarios de la red, y que necesitan ser traducidos por el emisor a una señal que se envíe por el medio inalámbrico y procesados por el receptor para recuperar los mensajes originales. En este sentido, tanto el tiempo de cómputo de las tramas en las entidades virtuales como la capacidad de los enlaces de fronthaul pueden tener un impacto significativo en el retardo. Esto lleva a introducir una nueva filosofía a la hora de procesar las tramas, en la que se tienen en cuenta dos conceptos simultáneamente y con la que se pretende minimizar el retardo, proporcionando una solución a varios de los requisitos que se buscan en la arquitectura 5G.

Junto con los *splits* funcionales aparece otro concepto: el *scheduling*. Se refiere al orden en el cual las tramas van a ser procesadas en la BBU. Como se puede suponer, la elección de un *scheduling* u otro implica un retardo distinto en cada una de las tramas ya que las que se procesan más tarde tienen que esperar en la cola a aquellas que las preceden. Este concepto supone una nueva métrica a optimizar. Se puede intuir que, sin tener en cuenta otras métricas, la solución óptima es procesar las tramas más cortas primero (*shortest-job-first*) ya que provocan menos retardo en las posteriores. No obstante, lo que se busca en este trabajo es optimizar ambas métricas al mismo tiempo -*splits* funcionales y *scheduling*- con lo que la solución encontrada no siempre será la trivial. Más adelante se van a analizar en detalle diferentes soluciones para tratar de optimizar la manera en que una red de quinta generación gestiona las tramas, que incluyen desde una solución óptima pero de lenta ejecución a otras que sacrifican parte de la idoneidad del resultado para acelerar los cálculos fijando alguna de las métricas.

Capítulo 3

Modelado del sistema e implementación

De forma general, la evaluación de las diferentes soluciones estudiadas se plantea mediante una simulación basada en fotografías. De este modo, cada fotografía representa una realización independiente del sistema, de forma que no se considera la evaluación temporal entre fotografías. Así, en cada fotografía hay varias tramas en la cola de la BBU esperando a ser procesadas, y una vez calculada la manera de repartir las funcionalidades entre las RRHs y la BBU, y distribuidas las tramas, se puede pasar al siguiente instante de tiempo y repetir el proceso. En cada una de las fotografías se van a simular un conjunto de RRHs conectadas a una misma BBU a través de sus respectivos enlaces. Éstos se caracterizan por tener una capacidad de transmisión, mientras que las máquinas (RRHs y BBU) cuentan con una capacidad computacional que es significativamente superior en el caso de la BBU.

Adicionalmente, la dirección en la que viajan las tramas es *downlink*, lo que quiere decir que se encuentran en la parte interna de la red y se dirigen hacia el usuario, de tal forma que pasan primero por la BBU y después por su respectiva RRH para llegar al usuario a continuación.

Las soluciones que se presentan a lo largo de este capítulo buscan optimizar el comportamiento de la red, decidiendo tanto la política de *scheduling* como el nivel de *split* de cada RRH. En algunos casos se realizará una optimización parcial, dejando uno de los parámetros fijo, de tal forma que se reduce significativamente el tiempo de ejecución y los recursos consumidos en la máquina que ejecute los algoritmos. A cambio, los resultados obtenidos se alejarán del óptimo. Todo esto se va a detallar en gráficas de tal forma que se puedan apreciar numéricamente los resultados obtenidos.

Por último, cada uno de los algoritmos va a incluir en este documento un ejemplo didáctico de dimensiones reducidas, de tal forma que se pueda entender con facilidad el modelo, lo que hace el algoritmo y cómo lo resuelve. El ejemplo es sencillo y casi resoluble a mano, de manera intuitiva y sin apenas necesidad de usar el algoritmo en cuestión. También se incluye un pseudocódigo con una explicación detallada, con el fin de esclarecer la forma en la que se han implementado los algoritmos. La implementa-

ción de cada uno de los algoritmos se ha realizado en lenguaje de programación C++ como módulos del entorno de simulación Generic Wireless Network System Modeler (GWNSYM) [30].

3.1 Configuración

Antes de detallar cada una de las soluciones, en esta sección se indica la configuración del escenario estudiado. Se considera una infraestructura donde un conjunto de varias RRHs están conectadas a una BBU a través de un conjunto de enlaces. La BBU está equipada con un procesador de un solo núcleo con una cierta capacidad computacional, de tal forma que solo una trama se puede procesar simultáneamente. De la misma manera, las RRHs poseen una capacidad de cómputo y los enlaces una capacidad de transmisión. Los valores concretos de capacidades, tanto de las RRHs como las BBUs, se han fijado en base a los trabajos realizados en [31] y [32]. En las tablas 3.1 y 3.2 se resumen los parámetros con los que se va a trabajar, cuya veracidad se comenta a continuación.

En las referencias previamente mencionadas se habla de 1000 flops (ciclos de reloj del procesador) requeridos para procesar cada bit de datos en media, asumiendo una BBU con cuatro Intel(R) Xeon(R) 4870 y 128GB de RAM; sumado a unas capacidades en giga-operaciones por segundo (GOPS) de entre 200 y 400 para la BBU y aproximadamente 10 veces menos para las RRHs. En el caso de las RRHs, este valor se va a hacer más flexible (con un rango desde la capacidad de la BBU hasta 10 veces menor) con el fin de analizar de manera más fiable los resultados obtenidos. Esto se traduce en unas capacidades de procesado, así como la de transmisión, que se van a representar en Megabits por segundo (Mbps). Los posibles *splits* funcionales se representan como el ratio de funcionalidades virtualizadas, de tal forma que varían entre 0 y 1 dependiendo de qué máquina esté realizando cada una de las operaciones que requieren procesado. Para más detalle, ver [29] y [33]. Por ejemplo, con un *split* funcional de 0.5, la BBU y la RRH correspondiente realizan la mitad del procesamiento requerido cada una. El *split* funcional también afecta a la cantidad de datos que se transmiten por el enlace. El procesamiento de una trama requiere de una cantidad determinada de operaciones, que se reparten entre la RRH y la BBU. De esta forma podemos definir el nivel de split para una trama dada como la relación entre la capacidad de procesado requerida en la BBU y RRH, $fSplit = proc^{BBU}/proc^{total}$, y por tanto el *split* funcional es 0 cuando todo el procesamiento se realiza en la RRH mientras que es 1 si se hace en la BBU.

Adicionalmente, el tamaño de las tramas se basa en la duración de las mismas en la tecnología LTE, que es de 10 milisegundos. Teniendo en cuenta la capacidad estándar de una BBU calculada anteriormente (200-400 Mbps), la longitud escogida es aquella que tarda entre un microsegundo y un milisegundo en procesarse en la BBU, en concordancia con los objetivos propuestos para la quinta generación. Más información sobre longitudes medias en Internet en [34] y [35]. Esto lleva a longitudes de trama de entre 200 y 200000 bits, unidad con la que se va a trabajar para este parámetro por

Tabla 3.1: Capacidades estándar a utilizar (Mbps)

Capacidades	BBU	RRH	Enlace
Mínima	200	20	1000
Máxima	400	200	100000

Tabla 3.2: Longitudes estándar de trama (bits)

Longitudes	
Mínima	200
Máxima	200000

comodidad.

Como ya se ha mencionado, las soluciones que se presentan a lo largo de este capítulo buscan optimizar o bien el *scheduling*, o bien el nivel de *split* de cada RRH, o ambas. Optimizar estos parámetros tiene un impacto directo sobre el retardo total del sistema, es decir, la suma de los retardos que va a experimentar cada trama. La minimización del retardo es el objetivo principal de este trabajo, por ser el factor más contribuyente a la hora de cumplir los requisitos de las redes 5G. De manera más técnica, el retardo que sufre una trama i se debe a varios factores: el tiempo que tarda en procesarse en la RRH, t^i_{RRH}; el tiempo que tarda en transmitirse por el enlace, t^i_{enlace}; el tiempo que tarda en procesarse en la BBU, t^i_{BBU}; y el tiempo que tiene que esperar en la BBU a que las tramas que están antes que ella en la cola (según el *scheduling*) se procesen. Cabe destacar que los tiempos de procesado y transmisión dependen del *split* funcional seleccionado. Si denominamos d al retardo de una trama, y t al tiempo consumido por cada uno de los elementos:

$$d^i = t^i_{BBU} + t^i_{enlace} + t^i_{RRH} + \sum_{j:\pi_j<\pi_i} t^j_{BBU} \tag{3.1}$$

donde el índice i representa la trama actual, y j las demás tramas. El símbolo π representa la posición en el *scheduling*, de tal manera que si $\pi_j < \pi_i$ quiere decir que la trama j va antes en la cola de espera que la trama i. De esta forma, el último sumando de la Ecuación 3.1 representa en tiempo de espera en la cola de cada trama. De la misma manera, el retardo global del sistema (denominado D) se puede obtener como la suma de los retardos individuales de todas las tramas:

$$D = \sum_i d^i = \sum_i \left(t^i_{BBU} + t^i_{enlace} + t^i_{RRH} + \sum_{j:\pi_j<\pi_i} t^j_{BBU} \right) \tag{3.2}$$

3.2 Splits funcionales fijos

Este modelo calcula el *scheduling* óptimo para una asignación de *splits* funcionales previamente fijada. Se puede observar que la distribución óptima de tramas es aquella que trata las más cortas primero. De esta forma el retardo acumulado para las posteriores tramas es el mínimo posible. Por ende, se trata del modelo que más simplifica los cálculos. El usuario del algoritmo puede fijar diferentes *splits* funcionales para las tramas, de tal forma que halle aquel que mejores resultados brinda en términos de retardo.

3.2.1 Pseudocódigo

Se presenta en el Algoritmo 1 un simple pseudocódigo del modelo, que posteriormente se explica con más detalle.

Algorithm 1 - *splits* funcionales fijos

```
1:  procedure FixedSplits
2:      frameLength ← Configuration file
3:      capacities ← Configuration file
4:      fSplit ← Configuration file
5:      if (fSplit < 0,0 or fSplit > 1,0) then return
6:      if (nFrames != nRRH) then return
7:      sort(frameLength)
8:      for all elements in frameLength do
9:          rrhDelay = thisFrameLength * (1 − fSplit)/rrhCapacity
10:         linkDelay = thisFrameLength * (1 − fSplit)/linkCapacity
11:         bbuDelay = thisFrameLength * fSplit/bbuCapacity
12:         accDelay += bbuDelay
13:         totalDelay = accDelay + rrhDelay + linkDelay
14:     end for
15: end procedure
```

En primer lugar, se dispone de un fichero de configuración que se puede modificar fácilmente a mano o mediante programación. A partir de dicho fichero, se extrae la información de los niveles de *split* asociados a cada trama así como la longitud de las tramas a enviar a cada RRH (líneas 2-4) y después se comprueba que la configuración es consistente (líneas 5-6). A continuación, se ordenan las tramas (línea 7) de menor a mayor tamaño, que se corresponde con el *scheduling* óptimo para esta situación. Esto se puede hacer mediante funciones ya existentes en la gran mayoría de lenguajes de programación, incluyendo C++. Una vez hecho esto, se calcula el retardo de cada trama (líneas 8-14) asociado a cada uno de los elementos que componen la arquitectura y se guardan los resultados finales en las entidades.

3.3 Scheduling fijo

El modelo de *scheduling* fijo comienza con la premisa de que las tramas ya tienen un orden fijo en el cual se deben procesar, de tal forma que se centra en optimizar los *splits* funcionales que se van a elegir para cada una de ellas. A diferencia del anterior la solución ya no es trivial, y se requiere de algún tipo de modelado que permita resolver el problema de manera sencilla. Basado en el trabajo realizado en [3], se ha diseñado una estructura de Directed Acyclic Graph (DAG) en el que cada nodo representa una trama con un *split* funcional determinado, y se ordenan por filas y columnas según su posición en el *scheduling* y el *split* funcional que les corresponde, respectivamente. Como se comenta a continuación, la estructura del grafo permite plantear el problema como uno de camino corto, para cuya resolución existen algoritmos eficientes.

3.3.1 Ejemplo sencillo

La Figura 3.1 muestra un ejemplo de grafo donde existen 3 RRHs y 2 posibles *splits*. Se puede apreciar que el *scheduling* es fijo, ya que las RRHs siguen un orden predeterminado y lo único que el algoritmo hace es escoger un *split* funcional para cada una de ellas. Los nodos *Src.* y *Dst.* son nodos auxiliares generados por el algoritmo con el fin de resolver el problema como el camino de coste mínimo desde *Src.* hasta *Dst.* En la figura se marca con flechas continuas una posible solución, donde para la primera RRH se escoge el *split* funcional 1, para la segunda se vuelve a escoger el 1, y para la tercera el *split* 2. La flecha discontinua refleja otra posible solución con un camino ligeramente diferente, ya que en este caso para la RRH 2 se ha escogido el *split* 2. Se observa que, al ser un DAG, el camino seleccionado por el algoritmo pasa siempre y solamente una vez por cada trama, escogiendo así el split, y un camino del nodo fuente al nodo destino representa una asignación de *splits* funcionales para el conjunto de tramas. En el caso de existir más *splits* funcionales posibles, el grafo contaría con más filas de tal forma que los caminos potenciales aumentan de manera exponencial. De la misma forma, añadir más RRHs implica agregar nuevas columnas, lo que se traduce en un factor multiplicativo al número de caminos posibles.

3.3.2 Pseudocódigo

Se presenta en el Algoritmo 2, de manera complementaria, el pseudocódigo de este modelo seguido de una explicación detallada del mismo. Los pasos que ya se han comentado anteriormente se resumirán por comodidad del lector.

Como ya se ha visto en el caso anterior, en primer lugar se leen los parámetros básicos en el fichero de configuración y se comprueba que están dentro de los límites (sección *CheckParams*). A continuación se comienza a construir el grafo. Para ello lo que se ha utilizado es una lista enlazada, donde cada elemento es una estructura que representa un nodo y contiene un enlace a todos aquellos nodos con los que está conectado, así como un coste asociado a dicho enlace. Estos costes se asignan en la

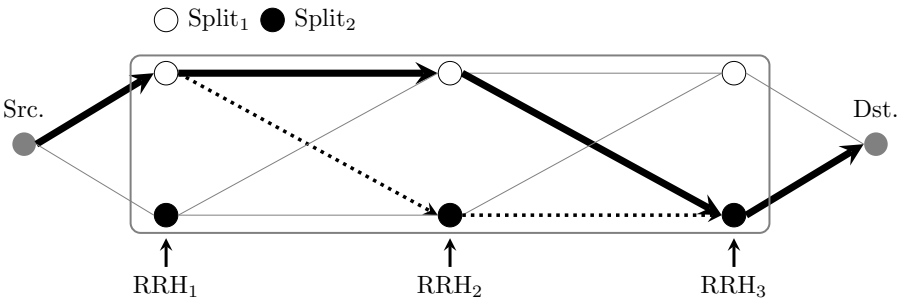

Figura 3.1: Ejemplo de grafo para el algoritmo de *scheduling* fijo, con 3 RRHs y 2 *splits* posibles. *Src* y *Dst* son nodos auxiliares.

Algorithm 2 - Scheduling fijo

```
1:  procedure FixedSplits
2:  CheckParams:
3:      frameLength, scheduling ← Configuration file
4:      capacities, possibleSplits ← Configuration file
5:      if (anySplit < 0,0 or anySplit > 1,0) then return
6:      if (nFrames != nRRH) then return
7:  AssignCost:
8:      for all elements in frameLength do
9:          for all elements in possibleSplits do
10:             rrhDelay = thisFrameLength * (1 − fSplit)/rrhCapacity
11:             linkDelay = thisFrameLength * (1 − fSplit)/linkCapacity
12:             bbuDelay = thisFrameLength * fSplit/bbuCapacity
13:             cost = bbuDelay * η + rrhDelay + linkDelay
14:             InsertNode(source, dest, cost)
15:         end for
16:     end for
17:     InsertNode(nodeF, dest, cost = 0)
18: Execute:
19:     predecessors = Dijkstra(graph)
20:     GetDelay(predecessors)
21: end procedure
```

sección *AssignCost*, donde se asignan los costes de los enlaces desde los nodos de cada RRH a la siguiente. Cabe destacar que, en la línea 13, el coste asociado a la BBU se multiplica por un factor que aparece como η. Este factor existe para que, al sumar el coste de todos los enlaces elegidos por el algoritmo de Dijkstra, se pueda obtener el retardo total sin ninguna operación adicional. La explicación de este hecho parte de la definición de retardo que se hizo previamente en la Sección 3.1. El retardo total presentado en la Ecuación 3.2 se puede reformular para tener en cuenta el retardo que cada trama provoca a las posteriores en el *scheduling*. Así, los retardos asociados a la BBU se pueden agrupar de la siguiente manera:

$$\begin{aligned}\sum_i \left(t^i_{BBU} + \sum_{j:\pi_j<\pi_i} t^j_{BBU} \right) = \sum_i \sum_{j:\pi_j \leq \pi_i} t^j_{BBU} = \\ = \sum_i t^i_{BBU} \cdot (N - \pi_i + 1)\end{aligned} \tag{3.3}$$

donde N es el número total de tramas. De esta forma, se puede definir el retardo total del sistema en función del retardo provocado por la trama actual en las posteriores (además del propio), en lugar de aquel que las anteriores provocan en la trama actual. Por claridad se va a denotar este retardo con la letra g, tal como se indica a continuación:

$$D = \sum_{i \in \mathcal{R}} g^i = \sum_i \left(t^i_{RRH} + t^i_{enlace} + t^i_{BBU} \cdot (N - \pi_i + 1) \right) \tag{3.4}$$

Con esto, se puede apreciar que el factor η que aparece en el pseudocódigo se corresponde con la expresión $(N - \pi_i + 1)$ en la ecuación 3.4. Por último, continuando con la sección *Execute*, se aplica el algoritmo de Dijkstra al grafo. Se obtiene una lista de predecesores de cada nodo, es decir, el nodo que lo precede en el camino óptimo calculado por el algoritmo. A partir de esta lista, se pueden obtener los retardos de cada trama por separado, porque ya se conocen los *splits* funcionales escogidos para cada una, y guardar los resultados finales en las entidades.

3.4 Optimización conjunta

La optimización conjunta, en cierta manera, combina los dos modelos anteriores: trata de optimizar tanto los *splits* funcionales como el *scheduling* para cada trama. En este caso ninguno de los parámetros es fijo, con lo que se puede intuir que la complejidad del problema crece sustancialmente respecto a los anteriores. Para resolverlo, se presentaron varias opciones como un grafo con la misma estructura que el anterior o el uso de técnicas heurísticas como la relajación de Lagrange [36] o el uso de etiquetas [37]. Sin embargo, todas ellas presentan diversos problemas como su excesiva complejidad (y por tanto, consumo excesivo de recursos como memoria o tiempo de ejecución) o su falta de flexibilidad para adaptarse a este modelo. Finalmente se dio con una solución

alternativa que consiste en formular el modelo como un problema de optimización lineal, de tal forma que se pudiera resolver con herramientas conocidas y fiables como GLPK [38]. Se define una variable binaria, $x_{i,j}$, que toma el valor 1 si el enlace entre los nodos i y j es seleccionado por el algoritmo, y 0 en caso contrario. Por simplicidad, se define w_{ij} para denotar el coste del enlace que conecta cualquier par de nodos (i, j). Esta variable toma valor infinito para aquellos casos donde no hay enlace. Con todo esto, y siendo $\mathcal{V}$ el conjunto de todos los nodos, la formulación es la siguiente:

Problem 1 (Representación como problema de optimización lineal)**.**

$$\textbf{\textit{min.}} \quad \sum_{i,j} x_{ij} \cdot w_{ij} \tag{3.5}$$

$$\textbf{\textit{s.t.}} \quad \sum_{i \in \mathcal{V}/k} x_{ik} - \sum_{i \in \mathcal{B}/k} x_{ki} = T_k \qquad \forall k \in \mathcal{V} \tag{3.6}$$

$$\sum_{i \in \mathcal{V}/\mathcal{V}_i} x_{ik} = 1 \qquad \forall k \in \mathcal{V}_i \tag{3.7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{V} \tag{3.8}$$

Como se observa en la Ecuación 3.5, este problema trata de minimizar el coste total de los enlaces seleccionados, o lo que es lo mismo, encontrar un camino válido de coste mínimo dentro de las restricciones.

Además, se han definido una serie de restricciones para asegurar que el problema emula el sistema. La ecuación 3.6 obliga a que el flujo entrante y saliente de los nodos pasivos sea igual a 0. Asimismo, fuerza al nodo origen a enviar una unidad de flujo y al nodo destino a ser el sumidero de dicha unidad. De esta forma, el camino elegido necesariamente recorre todo el grafo y con ello el problema se convierte en encontrar el camino de coste mínimo. La variable T_k, por tanto, vale 1 y -1 para los nodos origen y destino respectivamente, y 0 para los demás. La restricción 3.7 provoca que el flujo total que pasa por cada RRH sea igual a la unidad. En el modelado del grafo, escoger una RRH en el *scheduling* es equivalente a elegir su trama asociada. Dado que se asume que en cada instancia del problema hay una única trama por RRH, la restricción 3.7 asegura que cada RRH se escoja una única vez. Por último, la restricción 3.8 define la variable x como binaria. Se puede observar que el problema resultante es un Binary Linear Program (BLP), que se ha demostrado que es *NP-completo* y complejo de resolver [39][40]. Además, el tamaño del problema crece de manera exponencial tanto con el número de RRHs como con los posibles *splits*. Para resolverlo, se ha hecho uso de la herramienta GLPK. Como se puede intuir, el problema de optimización conjunto llega a resultados más eficientes que cualquiera de los anteriores, que de hecho son óptimos para esta configuración. A cambio, su consumo de recursos computacionales es significativamente mayor. En apartados posteriores se hará una comparativa entre todos los modelos para valorar las diferencias y los casos en los que merece la pena usar cada uno de ellos.

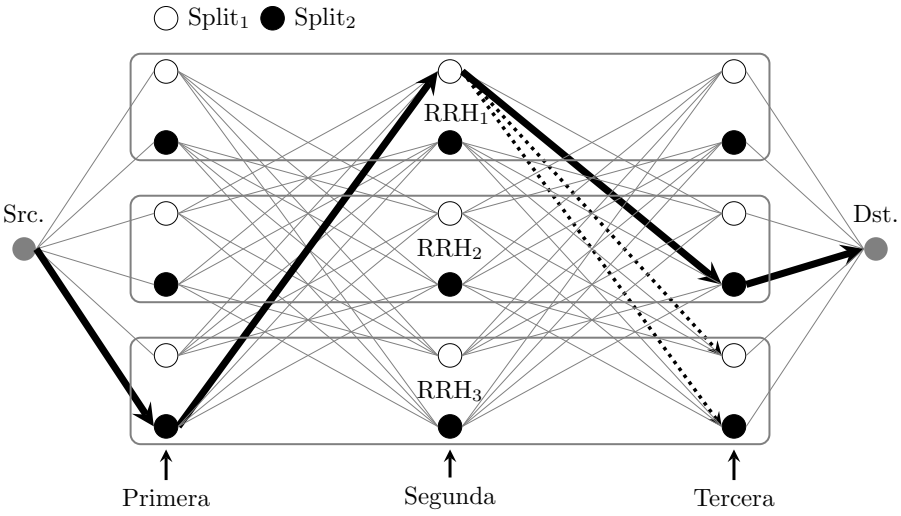

Figura 3.2: Ejemplo de grafo para el algoritmo de optimización conjunta, con 3 RRHs y 2 *splits* posibles. *Src* y *Dst* son nodos auxiliares.

3.4.1 Ejemplo sencillo

La figura 3.2 muestra un ejemplo de grafo donde existen 3 RRHs y 2 posibles *splits*. Como se puede observar, para cada RRH se utilizan dos filas que se corresponden con el número de *splits* que el algoritmo puede seleccionar. En la figura se marca con flechas de trazo continuo una posible solución. El *scheduling* de dicha solución consiste en la trama 3 con el *split* 2, seguida de la 1 con *split* 1 y por último la 2 con *split* 2. Cabe destacar que, tras seleccionar la trama 3 en el *scheduling*, se tiene que incluir una restricción que impida que la misma trama se seleccione en posteriores posiciones del *scheduling*. Este suceso se refleja con flechas discontinuas. Como se observa, la adición de nuevos *splits* funcionales posibles se traduce en un incremento del número de filas que a su vez está multiplicado por el número de RRHs, provocando de nuevo un crecimiento exponencial en el número de posibles soluciones. Más notorio es el efecto de añadir una RRH al escenario, ya que esto provoca la aparición tanto de una nueva columna representando una nueva posición en el *scheduling* como de un recuadro correspondiente a dicha RRH, suponiendo un incremento incontrolable de soluciones, de orden factorial.

3.4.2 Pseudocódigo

En el Algoritmo 3 se presenta el pseudocódigo de este modelo, que muestra la manera de convertirlo en el problema planteado en 3.5 y traducirlo según la representación utilizada por GLPK. Los pasos que ya se han comentado anteriormente se resumirán una vez más por comodidad.

En primer lugar se leen los parámetros básicos en el fichero de configuración, se calcula el número total de nodos y se comprueba que todo está dentro de los límites

Algorithm 3 - Optimización Conjunta

```
1: procedure JointOpt
2: CheckParams:
3:     frameLength, capacities, possibleSplits ← Configuration file
4:     nNodes = nFrames * nFrames * nSplits + 2
5:     if (anySplit < 0,0 or anySplit > 1,0) then return
6:     if (nFrames != nRRH) then return
7:     if (nNodes > maxViableNodes) then return
8: AssignType&Cost:
9:     for all nNodes do
10:        type[node] = T_k
11:    end for
12:    for all possibleSplits do
13:        for all elements in frameLength do
14:            for all elements in possibleSplits do
15:                if (source → frame != dest → frame) then
16:                    rrhDelay = thisFrameLength * (1 − fSplit)/rrhCapacity
17:                    linkDelay = thisFrameLength * (1 − fSplit)/lnkCapacity
18:                    bbuDelay = thisFrameLength * fSplit/bbuCapacity
19:                    cost[source][dest] = bbuDelay * η + rrhDelay + linkDelay
20:                end if
21:            end for
22:        end for
23:    end for
24:    cost[nodeF][dest] = 0
25: Problem:
26:    glpkParams = ConvertProblem(type, cost)
27:    results = GLPK(glpkParams)
28: end procedure
```

(sección *CheckParams*). El número máximo de nodos viables se calcula en función del tiempo de ejecución, que crece de manera exponencial con las dos variables que se utilizan (número de tramas y posibles *splits*). Este número se sitúa alrededor de los 4000, equivalente aproximadamente a un caso con 20 tramas y 11 posibles *splits*, donde el tiempo de ejecución puede alcanzar los cinco minutos, complicando de forma notable su evaluación. A continuación se comienza a construir el grafo equivalente (sección *AssignType&Cost*), donde al comienzo se asignan los tipos de cada nodo como se ha visto anteriormente en la variable T_k. Posteriormente se asignan los costes de cada enlace, teniendo en cuenta que en el grafo hay conexiones que no existen. Esto se representa en la sentencia condicional de la línea 15. Su significado es que los nodos que representan una trama en la posición i del *scheduling* no pueden ir a la misma trama en la posición $i + 1$. Tras añadir los nodos f y q que completan el grafo, lo siguiente es traducirlo para que la herramienta GLPK sea capaz de trabajar con él. Si bien esto no se representa de forma detallada en el pseudocódigo, se resume en la línea 26 para no sobrecargarlo. GLPK necesita una estructura muy determinada de matrices para trabajar, y la traducción se basa en reordenar los parámetros que se han calculado en la fase previa. Esto se realiza recorriendo varias estructuras matriciales, cuyo tamaño crece de forma masiva con el número tanto de RRHs como de *splits* funcionales posibles. Denominando $\mathcal{F}$ al conjunto de *splits* funcionales (de tal forma que $|\mathcal{F}|$ es su cardinalidad, y por tanto, el número de posibles *splits* funcionales) y $\mathcal{R}$ al conjunto de RRHs, el espacio de posibles soluciones es $|\mathcal{F}|^{|\mathcal{R}|} \times |\mathcal{R}|!$, y el número de variables a introducir en el problema GLPK, $2|\mathcal{R}||\mathcal{F}| + (|\mathcal{R}| - 1) \times |\mathcal{F}||\mathcal{R}| \times (|\mathcal{F}||\mathcal{R}| - |\mathcal{F}|)$[1]. Una vez realizada la traducción, se ejecuta GLPK que devuelve los resultados en su propio formato. A partir de ellos ya se pueden guardar los parámetros de salida en las entidades como en cualquiera de los otros algoritmos.

3.5 Variaciones en los modelos

Además de los modelos que se han analizado previamente, en líneas futuras se pretende ahondar en ellos y realizar ciertas modificaciones y variaciones para obtener resultados más flexibles. Si bien no se han completado estas variaciones en el transcurso de este trabajo, sí que se ha realizado una introducción a las mismas adelantando trabajo futuro. Éstas consisten en, principalmente, dos modificaciones fundamentales: añadir la posibilidad de gestionar múltiples tramas de una misma RRH en el mismo instante de tiempo e incluir paralelización en la BBU, lo que implica la posibilidad de procesar varias tramas de manera concurrente. A continuación se resumen las ideas recogidas y las posibilidades para implementarlas, así como los problemas que suponen con la arquitectura de programación actual.

[1]El primer término, $2|\mathcal{R}||\mathcal{F}|$, se corresponde con el número de enlaces entrantes y salientes a los nodos virtuales f y q. En el segundo término se multiplica el número de columnas ($|\mathcal{R}| - 1$) por el número de filas $|\mathcal{R}||\mathcal{F}|$ y enlaces salientes de cada nodo ($|\mathcal{F}||\mathcal{R}| - |\mathcal{F}|$).

Optimización conjunta

Procesado concurrente en la BBU

A la hora de implementar esta variación, se ha de tener en cuenta un concepto importante que permitía el modelado en el caso del algoritmo base. Como se describe en la sección 3.3.2, la clave para poder modelar el problema como un grafo de manera estática se sustenta en el hecho de que el retardo de una trama, si su *split* funcional y su *scheduling* son conocidos, también es conocido y fijo. Si existiera procesado simultáneo de tramas en la BBU, el tiempo de espera de una trama en la cola (o, de la misma forma, el retardo que una trama provoca en las posteriores) no solo depende de su *split* y su *scheduling*; es diferente dependiendo del *scheduling* seleccionado para el resto de las tramas, ya que varía dependiendo del *core* o núcleo en el que se encuentre cada una. El procesado concurrente de tramas implica que la BBU tiene a su disposición varios procesadores, o bien un procesador con más de un núcleo. Debido a ello, el *scheduling* ya no es una variable de una sola dimensión: escoger en qué orden se procesan las tramas no es suficiente, es necesario especificar en qué núcleo se va a procesar cada una. Lo que esto significa es que no se puede utilizar el algoritmo tal y como está diseñado actualmente. La posibilidad más cercana es añadir dimensiones al grafo que contemplen todas las posibilidades del nuevo *scheduling*; no obstante, la complejidad del algoritmo se incrementaría de manera masiva.

La única alternativa viable que se ha vislumbrado es hacer uso de un modelo totalmente diferente, basado en la teoría de colas. Su implementación y análisis son más complejos, pero es sencillo percatarse de que el sistema que se busca se corresponde con un sistema $M/M/S$ o $M/G/S$ donde S se corresponde con el número de núcleos disponibles.

Múltiples tramas por RRH

Ahora se contempla la posibilidad de que en la BBU exista más de una trama destinada a las misma RRH en el mismo instante de tiempo. En un principio la modificación parece sencilla, sin embargo el problema se complica de manera notable. En la sección 3.3.2 se definió el retardo de una trama, d, como:

$$d^i = t^i_{BBU} + t^i_{enlace} + t^i_{RRH} + \sum_{j:\pi_j<\pi_i} t^j_{BBU} \tag{3.9}$$

De nuevo, era posible reformular este retardo en función del provocado por la trama actual en las posteriores, en lugar de aquel que las anteriores provocan en la trama actual. Sin embargo, en esta modificación entra en juego un nuevo factor que se representa a continuación

$$d^i = t^i_{BBU} + t^i_{enlace} + t^i_{RRH} + \sum_{j:\pi_j<\pi_i} t^j_{BBU} + \sum_{j:\pi^R_j<\pi^R_i} t^j_{RRH} \tag{3.10}$$

y representa el retardo que las tramas anteriores de la misma RRH provocan en la actual. Sin embargo, la ecuación 3.10 es errónea por un motivo muy simple: es perfectamente posible que el procesado de una trama en la RRH i no afecte a otras tramas en la misma RRH, ya que en el momento en que la primera trama está siendo procesada, la segunda puede estar aún esperando en la BBU, procesándose en la BBU o viajando por el enlace i. De nuevo, surge el problema de que con el modelo actual no es posible definir el coste de los enlaces del grafo como un retardo estático, ya que depende de nuevas variables que se relacionan entre sí de forma compleja. De esta forma, la definición del retardo de una trama resultaría mucho más complejo:

$$d^i = f(t^i_{BBU}, t^i_{enlace}, t^i_{RRH}, \sum_{j:\pi_j<\pi_i} t^j_{BBU}, \sum_{j:\pi^R_j<\pi^R_i} t^j_{RRH}) \tag{3.11}$$

siendo función de cinco factores donde el último es, a su vez, función del *split* funcional, el *scheduling* y en general el retardo de cada una de las tramas anteriores. Modelar este problema añadiendo más caminos posibles al grafo incrementa su tamaño de forma masiva. Por tanto, la arquitectura actual se presenta como una alternativa muy pobre a la solución de este problema, que requiere de aproximaciones distintas.

Scheduling fijo

Procesado concurrente en la BBU

Gracias a que este modelo asume un *scheduling* fijo, simplifica significativamente el problema respecto al caso de la optimización conjunta. Ahora la posición de las tramas en la cola, así como su asignación a los diferentes núcleos de la BBU, son constantes y no requieren de optimización. Por tanto, resulta perfectamente viable hacer uso del mismo algoritmo que se ha utilizado antes de considerar la variación y modificarlo ligeramente. El grafo resultante tiene exactamente los mismos elementos, con la diferencia de que el cálculo de los costes de cada enlace ya no es el que se tenía previamente

$$g^i = t^i_{RRH} + t^i_{enlace} + t^i_{BBU} \cdot (N - \pi_i + 1) \tag{3.12}$$

sino que necesita una modificación para ajustarse a este caso. Más concretamente, el término $(N - \pi_i + 1)$ representaba las tramas que van después de la i en el *scheduling* y a las cuales afecta, porque les provoca un tiempo de espera en la BBU. Sin embargo, ahora la trama i solo afecta a las tramas posteriores que además se vayan a procesar en el mismo núcleo. Por tanto, se tiene que buscar una interpretación del *scheduling* bidimensional donde π^C_i ahora represente la trama en la posición i del *scheduling* correspondiente al núcleo C, e independiente de los otros núcleos de la BBU. De esta manera, el coste g de un enlace que parte de la trama i que se encuentra en el núcleo C se puede formular de la siguiente manera:

$$g^{i,C} = t^i_{RRH} + t^i_{enlace} + t^i_{BBU} \cdot (N^C - \pi^C_i + 1) \tag{3.13}$$

donde N^C es el número de tramas asignadas al procesador C y π_i^C es la posición de la trama i en el *scheduling* del núcleo C. Como se puede observar, el problema se puede resolver tratando a cada núcleo como una cola independiente y resolviendo el grafo de manera conjunta.

Múltiples tramas por RRH

Al igual que en el caso de la optimización conjunta, esta variación provoca que el retardo de una trama dependa de su tiempo de espera en la RRH. A su vez, esta variable depende de las tramas anteriores. Su interacción se resume de la siguiente manera:

$$t^i_{RRH} = MAX\left(d^{i-1} - t^i_{BBU+enlace+RRH} \quad , \quad 0\right) \tag{3.14}$$

Es decir, el tiempo de espera en la RRH compartida por varias tramas es como mínimo 0 (no puede ser negativo), pero puede ser mayor que 0 si el retardo total de la trama anterior perteneciente a la misma RRH supera al tiempo que tarda la trama actual en llegar a la cola. Esto se puede implementar haciendo que el coste de los enlaces dependa de los anteriores, pero provoca la aparición de nuevos caminos posibles en el grafo cuyo número crece de manera exponencial con el número de tramas. Por tanto, solo resulta viable con una cantidad limitada de tramas o haciendo uso de técnicas *branch-and-cut* a la hora de construir o resolver el grafo [41], que dan lugar a una implementación compleja. Para resolver este problema, es posible que aproximaciones distintas resulten más eficientes.

Splits funcionales fijos

Procesado concurrente en la BBU

La resolución del problema de *splits* funcionales fijos difiere enormemente de las anteriores, puesto que no hace uso de un grafo. La ventaja de esto es que la complejidad del algoritmo base es muy baja, pero a cambio presenta el inconveniente de que resulta imposible realizar modificaciones sobre él. Para implementar las variaciones, la única opción viable es tratar de modelar desde cero los problemas, pudiendo así tomar aproximaciones más flexibles. En el caso del procesado concurrente de tramas en la BBU, una alternativa a tener en cuenta es la resolución por fuerza bruta de todos los *scheduling* posibles. El algoritmo planteado se basa en probar todas las combinaciones de *scheduling*, que incluyen todas las asignaciones posibles de tramas en cada núcleo. Más estrictamente, el problema crece de forma polinomial con el número de procesadores y exponencial con el número de tramas. Por supuesto, otras alternativas son viables pero la que se ha planteado mantiene un rendimiento aceptable siempre que la cantidad de tramas no sea excesiva, y brinda soluciones óptimas para la asignación *splits* funcionales fijada.

Los resultados preliminares muestran, como se puede intuir, que según aumenta el número de núcleos el retardo disminuye de una forma similar a una exponencial decreciente. Además, el *scheduling* escogido por el algoritmo tiende a repartir equitativamente la longitud total de las tramas entre los procesadores, de tal forma que el procesado que se lleva a cabo en cada uno de ellos sea lo más cercano posible. Dentro de cada núcleo, el *scheduling* seleccionado se inclina a posicionar las tramas más cortas primero. Conociendo este comportamiento, se puede pensar en una optimización que reparta lo más equitativamente posible las tramas entre los núcleos, centrando el cómputo en este cálculo y reduciendo significativamente el consumo de recursos sin alterar en exceso la idoneidad de los resultados.

Múltiples tramas por RRH

Como ya se ha comentado, implementar variaciones en esta solución supone replantear el problema desde cero. Como en este caso la única variable a optimizar es el *scheduling*, una vez más resulta viable probar todas las combinaciones posibles sin que el cómputo sea excesivamente costoso. Para el cálculo del retardo total de un camino concreto se puede hacer uso de fórmulas más sencillas que las vistas en casos previos, ya que los retardos de tramas anteriores son conocidos para cada camino (los *splits* funcionales son fijos por definición, y cada camino se corresponde con un *scheduling* fijo). Al igual que en casos anteriores, puede haber más alternativas viables para resolver este problema.

Los resultados iniciales muestran que no existe una solución trivial, puesto que planificar las tramas más cortas primero ya no resulta eficiente si las tramas de longitud similar van a la misma RRH. Esto se debe a que, para disminuir el retardo en la cola de espera de una RRH, resulta más eficiente enviar las tramas a dicha RRH separadas en el tiempo. Al intercalar en la cola de la BBU las tramas que van a diferentes RRHs, el tiempo de espera en la BBU contribuye a que dé tiempo a terminar el procesado de una trama en una RRH determinada, de tal forma que cuando llegue la siguiente no tenga que esperar.

Capítulo 4

Evaluación y análisis

En este capítulo se va a analizar el rendimiento de los algoritmos presentados previamente. Como ya se ha comentado, la complejidad del problema completo puede, en ocasiones, limitar su uso práctico, hasta el punto de volverlo inviable. Los problemas parciales son más ligeros y siempre se van a poder utilizar en la práctica, pero conllevan un deterioro en el retardo mínimo que alcanzan sus resultados óptimos. Para medir las pérdidas desde el punto de vista de la optimización, así como los costes computacionales adicionales que supone utilizar un algoritmo más potente, se han realizado una serie de ejecuciones consecutivas de las soluciones vistas previamente variando los parámetros. Tanto las capacidades como las longitudes de trama que se van a utilizar, son las que se presentaron en la sección 3.1. Se adjunta en la Tabla 4.1 un resumen que relaciona estos parámetros con la capacidad de la BBU, que en este caso se fija a 200 Mbps, para mayor comodidad a la hora de su representación gráfica y de la comprensión de los resultados. Se asume que el enlace es de alta capacidad [42], por tanto haciendo despreciable el retardo de transmisión en comparación con el resto. La capacidad relativa de las RRHs se representa como el ratio $Cap_i^R = Cap_i/Cap^B$, que indica que la capacidad relativa de la RRH i es su capacidad absoluta dividida entre la capacidad absoluta de la BBU. Se aprecia que la variable resultante es adimensional. De manera similar, la longitud relativa de las tramas es el tiempo que tardarían en procesarse si existiera un 100 % de centralización (todo se procesa en la BBU). En la fórmula $Long_i^R = Long_i/Cap^B$ se aprecia de manera más clara que la dimensión de esta variable es el tiempo, y por ello se representa en segundos. Convertir estas variables a relativas supone varias ventajas, como claridad a la hora de representar los resultados o flexibilidad a la hora de interpretarlos. Esto se debe a que los resultados son igualmente válidos para una BBU con una capacidad de 200 Mbps y para otra con una capacidad cualquiera, siempre y cuando el resto de variables se representen en función de ésta. A continuación se describirán los escenarios en los que se han evaluado cada una de las soluciones propuestas. Seguidamente se presentarán los resultados obtenidos y se concluirá el capítulo resumiendo las principales conclusiones extraídas de esta evaluación.

Tabla 4.1: Parámetros cuando la capacidad de la BBU es de 200 Mbps

Parámetros	Capacidades RRH		Longitudes de trama	
Absoluta	20 Mbps	200 Mbps	200 bits	200000 bits
Relativa	0.1	1	1μs	1ms

Tabla 4.2: Casos contemplados

Experimento	Capacidad RRH	Longitud tramas
Tramas heterogéneas	0.1 - 1 (homog.)	Distr. uniforme (1μs - 1ms)
RRH heterogéneas	0.1 - 1 (paso 0.1)	1μs , 10μs , 100μs , 1ms
Ambas heterogéneas	0.1 - 1 (paso 0.1)	Distr. uniforme (1μs - 1ms)

4.1 Escenarios

Se van a diferenciar tres tipos de escenarios, los cuales comparten la siguiente configuración:

- Capacidad de la BBU: 200 Mbps.
- Capacidad del enlace: 100000 Mbps (retardo despreciable).
- Posibles *splits*: de 0 % a 100 % de procesamiento en la BBU, con paso 10 %.

Además, se cuenta con una serie de parámetros variables, que se indican en la Tabla 4.2, y que se detallan a continuación.

Cuando se trata una variable como homogénea, su valor es fijo e igual para todas sus instancias. En una ejecución de uno de los algoritmos cuando las tramas son heterogéneas pero las RRHs son homogéneas, estas últimas tendrán todas un único valor, por ejemplo 0.7. En ejecuciones posteriores del mismo caso las RRHs pueden tener un valor diferente, pero como son homogéneas tiene que ser el mismo para todas ellas, por ejemplo 0.4. La diferencia con el caso de las RRHs heterogéneas es que, para una misma ejecución, cualquiera de ellas puede tener un valor diferente de capacidad a cualquiera de las demás. Ocurre lo mismo con el caso de las tramas, cuando son heterogéneas seguirán una distribución aleatoria uniforme en cada una de las ejecuciones. Sin embargo, si son homogéneas todas tendrán el mismo valor; los 4 valores posibles que pueden tomar para estos casos se ven en la Tabla 4.2. De este modo, el caso en que únicamente la capacidad de las RRHs es homogénea emula un despliegue de red heterogéneo sobre el que se despliegan servicios heterogéneos. Por el contrario, el caso opuesto asume una red de acceso heterogénea sobre la que se cursa tráfico homogéneo.

Con todo esto, se han contemplado dos casos en los que una de las variables es homogénea y la otra es heterogénea. A su vez, el último caso es el más general donde ambas variables se asumen heterogéneas, pudiendo tomar cualquier valor dentro de los rangos definidos. Se van a realizar 1000 ejecuciones de cada algoritmo para cada

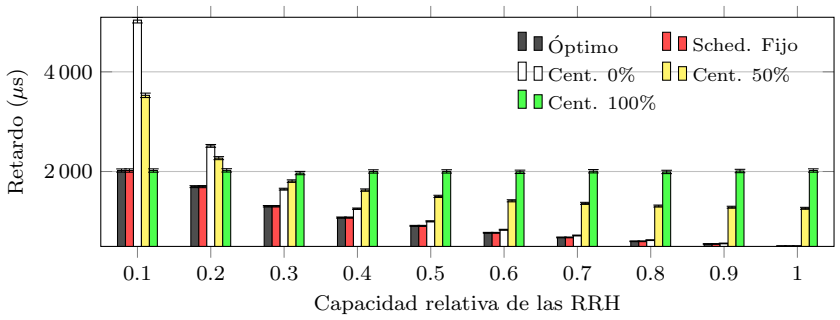

Figura 4.1: Retardo conjunto medio con tráfico heterogéneo y capacidad de cómputo de las RRHs homogénea

uno de los casos, donde a su vez las variables homogéneas van a tomar varios valores para poder apreciar su impacto en los resultados. La semilla aleatoria que se utiliza es la misma en todos los algoritmos, evitando así las desviaciones entre unos y otros y favoreciendo una comparativa más fiable entre ellos.

4.2 RRH homogéneas y tráfico heterogéneo

En el primer escenario, se considera una red de acceso a la BBU homogénea donde las RRHs tienen una capacidad fija relativa que varía desde 0.1 hasta 1. Para las distintas ejecuciones, se generan nuevas tramas aleatorias distribuidas uniformemente entre 1 microsegundo y 1 milisegundo de procesamiento en la BBU. Este caso es aplicable a aquellas redes reales donde las estaciones base que están conectadas a la misma BBU son del mismo modelo o muy similares, y por tanto cuentan con la misma capacidad de cómputo. Un esquema de este tipo permite cuantificar el efecto de aumentar o disminuir las capacidades de las RRHs, mientras que el tráfico se mantiene aleatorio y, en media, no altera los resultados. La Figura 4.1 muestra el retardo conjunto de todas las tramas, así como el intervalo de confianza del 95 %, para las distintas capacidades relativas de las RRHs. Independientemente de dichas capacidades, se observa que el esquema de *scheduling* fijo presenta los mismos resultados que la optimización conjunta, cuya solución es óptima. Esto confirma los resultados previstos a base de intuición, puesto que cuando las RRHs tienen la misma capacidad, cualquier orden de procesado de las tramas que no sea de la más corta a la más larga resulta en un incremento del retardo en las tramas posteriores que no se compensa mediante ningún otro mecanismo. Por otro lado, el modelo de *splits* funcionales fijos ha sido ejecutado con tres niveles de *split* diferentes para dotar de mayor flexibilidad a los resultados: 0 %, 50 % y 100 % de centralización. La idoneidad de este algoritmo varía según las capacidades relativas de las RRHs; cuando éstas son menos potentes, el mejor resultado es aquel con mayor nivel de centralización y viceversa. Esto también resulta lógico, puesto que cuanta más capacidad tienen las RRHs más se aprovecha la misma.

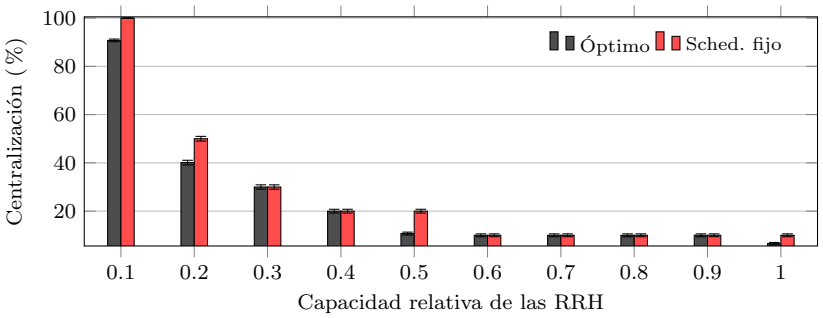

Figura 4.2: Grado de centralización promedio con tráfico heterogéneo y capacidad de cómputo de las RRHs homogénea

Sin embargo, a la hora de comparar los resultados del modelo de *splits* funcionales fijos con cualquiera de los otros dos, se aprecia un claro deterioro en el retardo. Si bien se podría alcanzar un buen resultado adaptando el nivel de *split* de acuerdo a la capacidad relativa de las RRHs, no supone ninguna ventaja frente a utilizar los otros dos desde el punto de vista de la optimización.

Respecto a la selección de *splits* funcionales, la Figura 4.2 muestra el grado de centralización obtenido por los algoritmos. En este caso no se representa el modelo de *splits* funcionales fijos puesto que no trabaja con esta variable. Como se puede observar, ambos modelos se comportan de manera muy similar hasta el punto de que las escasas diferencias no presentan discrepancia en términos de retardo. Ambas soluciones son capaces de adaptar el *split* escogido en función de las características de cada escenario, lo cual una vez más las sitúa un paso por delante del modelo de *splits* funcionales fijos. Cabe destacar el interesante comportamiento de ambos algoritmos a la hora de escoger los *splits* funcionales óptimos, que no se puede apreciar en la gráfica. Solo se asignan dos tipos de *splits*: a las tramas que se encuentran en las primeras posiciones del *scheduling* se les asigna una centralización del 0 %, y a las tramas en las últimas posiciones se les asigna el 100 % de centralización. Esto se debe a que, en las primeras posiciones, no es eficiente procesar las tramas en la BBU ya que provocan retardo adicional a todas las siguientes. Sin embargo, se alcanza un punto de inflexión a partir del cual merece la pena pasar a un procesado total en la BBU ya que tiene mayor capacidad, y la ganancia en velocidad de cómputo compensa el retardo provocado en tramas posteriores. Según nos dictan los resultados, en ningún caso un *split* funcional intermedio ofrece mayores beneficios que seguir esta filosofía. Por supuesto, en la Figura 4.2 se aprecian *splits* funcionales intermedios, pero se trata de valores promediados que se derivan del número de veces que se asignan el 0 % y el 100 %. Por último, cabe mencionar que las diferencias entre ambos algoritmos a la hora de escoger *splits* funcionales se debe a decisiones triviales. Por ejemplo, en el caso en el que las RRHs tienen capacidad relativa 1 (es decir, BBU y RRHs tienen la misma capacidad) la máquina que realice procesado de la última trama es irrelevante porque siempre da lugar al mismo retardo.

Como cabía esperar, el grado de centralización disminuye cuanto más potentes

son las RRHs. El *scheduling* óptimo siempre resulta ser aquel que asigna las tramas más cortas primero, y el retardo mínimo es alcanzado por dos de los tres modelos. El modelo de *splits* funcionales fijos muestra un rendimiento algo inferior a los otros dos, además de forzar al usuario a fijar los *splits* en función de sus necesidades. Por otro lado, tanto el modelo de *scheduling* fijo como el óptimo se ajustan automáticamente a cada caso, alcanzando mejores resultados que además son muy similares entre ellos. Posteriormente se va a analizar el rendimiento de los modelos en términos de cómputo, pero como se puede intuir, el modelo de optimización conjunta consume más recursos que cualquiera de los parciales. Teniendo esto en cuenta, parece la opción lógica hacer uso del algoritmo de *scheduling* fijo para los casos en los que las RRHs son similares: es más ligero que la optimización conjunta, y muestra resultados superiores al modelo de *splits* funcionales fijos.

4.3 RRH heterogéneas y tráfico homogéneo

Este escenario consta de una red de acceso a la BBU heterogénea donde las RRHs tienen una capacidad relativa variable, que se genera de manera aleatoria para cada una de ellas entre 0.1 y 1. En este caso el tráfico es homogéneo, es decir, todas las tramas tienen la misma longitud relativa dentro de la misma ejecución. Esta longitud puede ser de 1, 10, 100 o 1000 microsegundos. Este modelo se adapta bien a redes que gestionan o que tratan con un tipo de tráfico en concreto, y por tanto las tramas son iguales o muy similares en tamaño. Como las RRHs tienen capacidades diferentes, se puede asumir que la red en sí es heterogénea aunque trate con tráfico similar, dando así más flexibilidad al modelo. En este caso se puede analizar con precisión el efecto que tiene sobre la optimización el hecho de tratar con tramas más largas o más cortas, mientras que la capacidad de las máquinas se aleatoriza de tal forma que se mitiga su influencia. La Figura 4.3 muestra el retardo conjunto de todas las tramas, así como el intervalo de confianza del 95 %, para las distintas longitudes relativas de las tramas. Como cabía esperar, el retardo medio de las tramas aumenta con la longitud de las mismas, independientemente del algoritmo aplicado. Es más interesante la comparativa entre los mismos: si bien es cierto que la implementación óptima obtiene mejores resultados que cualquiera de los esquemas parciales, el modelo de *scheduling* fijo supera al de *splits* funcionales fijos con cualquiera de sus configuraciones, siendo la que mejor se comporta aquella que realiza todas las operaciones en las RRHs.

A diferencia del caso anterior, la discrepancia entre el esquema óptimo y los parciales es significativa, lo que lleva al usuario a tomar una decisión al aplicar los algoritmos. Se puede optar por un mejor rendimiento en lo que se refiere a resultados; o bien por una simplificación que reduzca el consumo de recursos.

En cuanto a la selección de *splits* funcionales, la Figura 4.4 muestra el grado de centralización escogido por aquellos algoritmos que lo optimizan. En este caso se observan diferencias más significativas entre ellos. Cabe recordar que la semilla utilizada para el generador de números aleatorios es la misma para ambos, lo que significa que

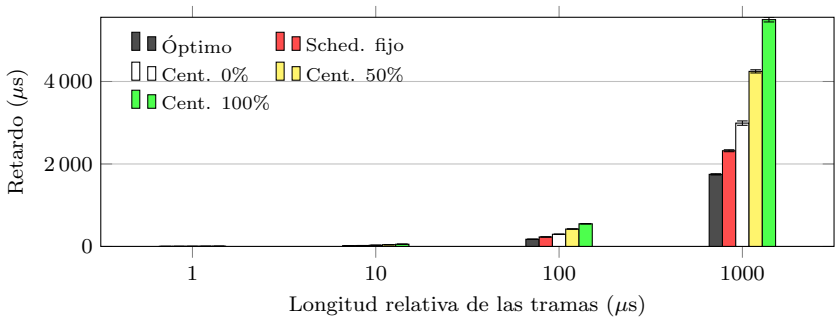

Figura 4.3: Retardo conjunto medio con tráfico homogéneo y capacidad de cómputo de las RRHs heterogénea

las desigualdades no se deben a este factor. Los resultados obtenidos son de alto interés; si bien es cierto que a medida que crece la longitud de las tramas el grado de centralización aumenta, este efecto es muy sutil y apenas alcanza un incremento de un 1 % cuando el tamaño de las tramas se ha multiplicado por 1000. Adicionalmente, hay una cuestión que se puede venir a la mente al ver estos resultados. Se había hablado de que el mejor *scheduling* posible es procesar las tramas más cortas primero, y si lo especificamos así en el algoritmo de *scheduling* fijo, los resultados deberían ser iguales a los obtenidos con el esquema de optimización conjunta. Sin embargo, aquí se observa que los retardos obtenidos son sub-óptimos y los *splits* funcionales seleccionados no coinciden exactamente, aunque son muy similares en media. Para ilustrar el por qué de este comportamiento, se debe primero recordar que, cuando se supuso que dicho *scheduling* era el óptimo, se estaba contando con que la capacidad de las RRHs era la misma para todas. De esa manera, se tiene que el retardo de una trama es independiente de la RRH a la que esté asignada. Este es el concepto clave, pues una vez que se altera la capacidad de cada RRH permitiendo que sean diferentes es cuando los resultados se desvían. Se deduce que el *scheduling* óptimo en esta situación ya no es el trivial, sino que se debe hacer uso del modelo de optimización conjunta (o del modelo de *splits* funcionales fijos en su defecto) para encontrarlo.

En este esquema, los resultados obtenidos muestran que el algoritmo a utilizar ya no es una decisión sencilla. Por un lado, la optimización conjunta brinda los mejores resultados con unas diferencias significativas respecto a los otros, a cambio de un cómputo más pesado. El modelo de *splits* funcionales fijos muestra un rendimiento inferior a los otros dos en cualquiera de sus variantes, lo que lo hace poco deseable por no tener ninguna ventaja clara frente a ellos. Por otro lado, el modelo de *scheduling* fijo alcanza unos resultados aceptables aunque se hallan por debajo de los óptimos, siendo, en cambio, menos exigente en términos de cómputo. Se trata de una solución de compromiso; en función del tipo de sistema que posea el usuario y de los resultados que busca, tanto el algoritmo de *scheduling* fijo como la optimización conjunta pueden ser los mejores candidatos en este tipo de redes heterogéneas donde se trabaja con un tráfico estable y una longitud de tramas homogénea.

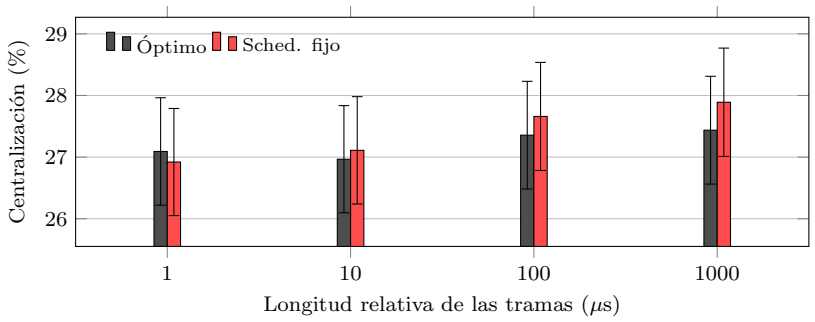

Figura 4.4: Grado de centralización promedio con tráfico homogéneo y capacidad de cómputo de las RRHs heterogénea

4.4 RRH heterogéneas y tráfico heterogéneo

En este último escenario, las RRHs tienen una capacidad relativa variable entre 0.1 y 1 como se ha visto anteriormente, generada de manera aleatoria para cada una de ellas. A su vez, el tráfico también se genera de manera aleatoria. Las tramas están distribuidas uniformemente entre 1 microsegundo y 1 milisegundo de procesamiento en la BBU. Este modelo es el más genérico, ya que muestra resultados medios donde tanto el tráfico como la capacidad de las RRHs son variables y pueden tomar cualquier valor dentro de su rango. De esta manera, es un escenario aplicable a redes de todo tipo donde se integra tráfico variable de diferentes naturalezas y cuyas RRHs pueden ser de diversos tipos y por tanto tener capacidades muy dispares. Con este tipo de modelo, teniendo en cuenta que se realizan 1000 ejecuciones de cada caso, se pueden obtener resultados muy fiables acerca del comportamiento medio de cada uno de los algoritmos, cuando no están condicionados por ningún parámetro fijo. En la Figura 4.5 se tienen los retardos conjuntos medios de cada uno de los algoritmos. Los resultados tienen cierta semejanza con los obtenidos en el caso anterior, donde el modelo óptimo destaca frente a los demás y el de *scheduling* fijo tiene un rendimiento que, aún sin alcanzar al ya mencionado, supera con creces los retardos obtenidos por el modelo de *splits* funcionales fijos en cualquiera de sus variantes. De esta forma se confirma lo que se ha ido observando en los casos anteriores, y es que el algoritmo de *splits* funcionales fijos siempre presenta peor rendimiento en comparación con el resto de opciones, ofreciendo un retardo un 50 % superior en el caso más favorable.

Una vez más, el modelo de *scheduling* fijo presenta una solución menos costosa y de calidad razonable, ya que la penalización mostrada en términos de retardo es aproximadamente un 10 %. Por otra parte, el modelo de *splits* funcionales fijos muestra un rendimiento peor que las otras alternativas.

Con respecto a la elección de *splits* funcionales, la Figura 4.6 muestra una pequeña comparativa del grado de centralización escogido por los algoritmos de optimización conjunta y de *scheduling* fijo. Se observa que, en media, el primero tiende a un resultado algo más elevado. Destaca el hecho de que, una vez liberamos los parámetros, las

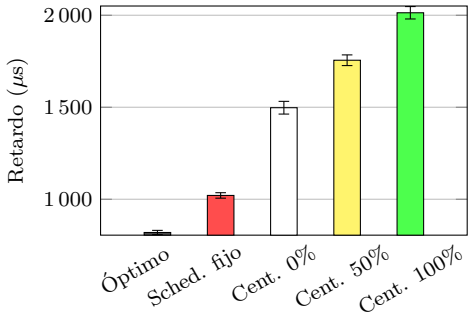

Figura 4.5: Retardo conjunto medio con tráfico heterogéneo y capacidad de cómputo de las RRHs heterogénea

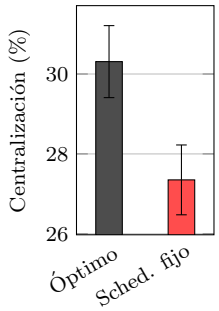

Figura 4.6: Grado de centralización promedio con tráfico heterogéneo y capacidad de cómputo de las RRHs heterogénea

diferencias entre ambos se han acrecentado hasta alcanzar un 3 % mientras que en casos previos se situaba alrededor del 0.2 %. Esto indica una clara diferencia en la elección de caminos óptimos, razón de más para hacer uso de la optimización conjunta en caso de estar buscando las mejores soluciones.

Hasta ahora se han analizado los resultados obtenidos por cada uno de los modelos, pero resulta complicado obtener una conclusión definitiva sin concretar el rendimiento en términos de cómputo de los mismos. A continuación se va a realizar un análisis del uso de recursos por cada uno de los algoritmos, tanto en tiempo como en consumo de memoria. Con ello, se espera obtener información más detallada que permita sacar conclusiones firmes acerca de la viabilidad de cada uno de ellos, así como posibles casos de uso en los que alguno resulte más rentable a pesar de sus peores prestaciones.

4.5 Uso de recursos

Las medidas de tiempo se han realizado a través del comando *time* proporcionado por Linux, en una máquina virtual con 2GB de memoria RAM y un procesador Intel(R)

Tabla 4.3: Tiempos medios de ejecución (s)

Ejecuciones consecutivas	1	5	10	50	100
Optimización conjunta (11)	3.83	18.78	37.72	156.20	308.24
Optimización conjunta (2)	0.086	0.408	0.807	3.988	8.011
Scheduling fijo	0.009	0.019	0.032	0.130	0.257
Splits funcionales fijos	0.005	0.013	0.020	0.080	0.155

Xeon(R) Silver 4110 CPU a 2.10GHz. Asimismo, el espacio de memoria utilizado se ha podido analizar gracias a la herramienta *Valgrind* [43]. Se han ejecutado los algoritmos con 10 tramas y 11 *splits* funcionales posibles, excepto el caso (2) de la optimización conjunta, donde se han elegido 2 *splits* funcionales posibles (centralización del 0 %, y 100 %) para dar más flexibilidad a los resultados. El caso con 2 *splits* posibles tiene un coste computacional significativamente menor que el caso con 11, pero, en teoría, presenta unos resultados peores. De esta manera, el algoritmo puede ser utilizado con diferentes parámetros en función de las necesidades de cada usuario y en este trabajo se analizan dos de ellas para cubrir más casos de uso. Adicionalmente, se muestran valores para un diferente número de ejecuciones consecutivas de los algoritmos. De esta manera, se evalúa con más claridad su implementación, en términos de linearidad de los tiempos de ejecución y consumo de memoria, a la hora de ser ejecutados de manera seguida.

Los tiempos de ejecución se muestran en la Tabla 4.3, donde se puede apreciar claramente cuáles son los costes temporales adicionales de utilizar los algoritmos más complejos. Todos los resultados representados son una media de 10 pruebas para cada caso (con una desviación prácticamente despreciable). Se observa que optimizar solo uno de los parámetros por separado supone un tiempo de ejecución muy pequeño, viable incluso en tiempo real. Entre ellos, el algoritmo que trabaja con un *scheduling* fijo presenta unos tiempos más elevados, aproximadamente un 150 % respecto al algoritmo de *splits* funcionales fijos. En cuanto a la optimización conjunta, se aprecia un considerable aumento incluso en el caso que cuenta con 2 *splits* funcionales a optimizar, unas 50 veces mayor que el caso de los *splits* funcionales fijos. Cuando se aumenta el número de *splits* funcionales posibles a 11, para igualarlo con el algoritmo de *scheduling* fijo, los tiempos se disparan hasta estabilizarse alrededor de los 3 segundos por ejecución.

De la misma manera, se representa el espacio utilizado en memoria en la Tabla 4.4, donde esta vez los resultados son deterministas. Las unidades representan los megabytes (MB) de memoria dinámica que el algoritmo ha utilizado, reservada y posteriormente liberada. Se aprecia un comportamiento similar a los tiempos de ejecución; en este caso la memoria utilizada por los algoritmos que fijan una de las variables es pequeño, de apenas 20MB en casos donde se han realizado muchas ejecuciones sucesivas. Esto da una pista de que se podrían llegar a implementar estos modelos en dispositivos de capacidades muy limitadas. Entre ellos existe una mínima diferencia, siendo el modelo de *scheduling* fijo el que consume aproximadamente un 120 % respecto al de *splits*

Tabla 4.4: Memoria dinámica utilizada (MB)

Ejecuciones consecutivas	1	5	10	100
Optimización conjunta [11]	286.124	1430.299	2860.519	28605.788
Optimización conjunta [2]	12.186	60.610	121.140	1210.685
Scheduling fijo	0.319	1.274	2.468	23.963
Splits funcionales fijos	0.270	1.029	1.979	19.066

funcionales fijos. La verdadera diferencia es la que existe frente al algoritmo de optimización conjunta: en el caso de tener 2 posibles *splits* funcionales, se incrementa el uso de memoria en un factor 50 nada despreciable, que aún se multiplica por más de 20 al incluir los 11 posibles *splits* funcionales. Esto finalmente resulta en un uso que alcanza los gigabytes de memoria en apenas 5 ejecuciones consecutivas.

En ambas métricas, la diferencia entre los algoritmos parciales y el de optimización conjunta son de una proporción muy elevada. Estos datos implican, como ya se había intuido, que las mejores soluciones alcanzadas por la optimización conjunta no se obtienen sin coste. En el caso de una sola ejecución, que pretende ser el más cercano a la realidad y el que mayor probabilidad tiene de ser utilizado, se requiere una máquina con una memoria RAM razonable. Esto puede provocar que no sea viable aplicar este algoritmo en pequeños controladores o microprocesadores integrados si se diera el caso. A su vez, los tiempos de ejecución son de un orden a tener en cuenta, especialmente si se consideran los 11 *splits* funcionales posibles.

4.6 Comparativa

Una vez finalizada la evaluación de las diferentes métricas, se puede comenzar a comparar de manera fiable los algoritmos en función de sus resultados y su rendimiento. Como se ha visto, la optimización conjunta alcanza los resultados óptimos que en todos los casos superan significativamente a los obtenidos por ambos modelos de optimización parcial, pero esto es a cambio de un rendimiento mucho más costoso y lento que puede suponer su inviabilidad en determinados casos. El algoritmo de *scheduling* fijo tiene unos requisitos de cómputo similares al de *splits* funcionales fijos. Si bien es cierto que el primero es ligeramente más costoso, las diferencias resultan prácticamente despreciables. Sin embargo, su disparidad en los resultados resulta en el *scheduling* fijo como claro candidato, puesto que un incremento del 50 % en términos del retardo de las tramas supone un gran sacrificio a cambio de unas ventajas ínfimas.

A la hora de medir el uso de recursos, se han tenido en cuenta dos vertientes del algoritmo de optimización conjunta. Hasta ahora se ha estado trabajando con 11 posibles *splits* funcionales, pero la opción de considerar solo 2 de ellos (centralización del 0 %, y 100 %) parece viable desde el punto de vista de los recursos computacionales. La pregunta que se viene a la mente es cuánto afecta este decremento a los resultados y si merece la pena o no tener en cuenta esta variación a la hora de realizar una comparativa.

Para ello se han realizado 1000 ejecuciones del algoritmo de optimización conjunta en sus dos versiones, con ambas variables heterogéneas. En un principio se pensaba realizar una gráfica comparativa, sin embargo se ha descubierto que los resultados son exactamente iguales. Esto implica que los *splits* seleccionados por el algoritmo son o bien 0 % o bien 100 %, independientemente de cuáles sean los otros *splits* funcionales posibles. Esto acelera enormemente el proceso sin necesidad de deteriorar los resultados en ningún sentido, lo que resulta en una enorme ventaja para el modelo de optimización conjunta.

Con todo esto, el foco se sitúa sobre los algoritmos de *scheduling* fijo y de optimización conjunta. El problema se resume en una disyuntiva muy común en el ámbito de las telecomunicaciones, que es la elección entre un proceso ligero con resultados razonables u otro más costoso que ofrece mejores soluciones. La opción a escoger acaba dependiendo del tipo de sistema en el cual se vaya a aplicar, así como los recursos disponibles para el usuario y el objetivo general del mismo. Dividiendo las posibles situaciones según estos factores, se tiene el siguiente listado:

- ***Scheduling* fijo**
  - *Redes homogéneas*: Como se analizó en la Sección 4.2, en el caso de redes homogéneas donde todas las RRHs tienen capacidades muy similares los resultados obtenidos por el algoritmo de *scheduling* fijo están a la par con los de la optimización conjunta. Esto significa que, de manera directa, el primero no posee ningún inconveniente y sin embargo ofrece un rendimiento mucho mejor, convirtiéndolo en la opción idónea.
  - *Redes de bajo coste*: Cuando se dispone de un sistema con un bajo presupuesto económico, es de suponer que se trata de una red pequeña que no necesita de una optimización inmejorable. El algoritmo de *scheduling* fijo ofrece unos resultados lo suficientemente buenos y permite que los cálculos se ejecuten en máquinas de baja capacidad, lo que lo hace ideal para esta situación.
  - *Redes de baja demanda*: En redes donde el tráfico es escaso, es evidente que no se requiere una optimización precisa ya que mayores retardos no van a provocar ningún tipo de congestión. Debido a los requisitos más laxos de este tipo de sistemas, la prioridad se centra en utilizar un modelo funcional que permita abaratar los costes lo máximo posible. Por consiguiente, implementar el algoritmo de *scheduling* fijo en máquinas poco potentes se convierte en la opción más beneficiosa.
  - *Redes donde la optimización no es estricta (tolerantes al retardo)*: En general, cualquier tipo de red que no demande soluciones óptimas se puede beneficiar de las ventajas de un algoritmo ligero sin verse afectada por sus inconvenientes. El modelo de *scheduling* fijo es, en este tipo de situaciones, la solución más competente.

- **Optimización conjunta**
  - *Redes críticas con muchos recursos*: Si se cuenta con un sistema con un alto presupuesto que tiene unos requisitos más estrictos, lo más razonable es invertir en máquinas que permitan ejecutar algoritmos más potentes en tiempo real. En estos casos en los que los recursos disponibles no son un factor limitante, el algoritmo de optimización conjunta resulta el más apto por los resultados óptimos que es capaz de obtener.
  - *Redes de alta jerarquía*: En redes más grandes que integran otros tipos de redes de menor jerarquía, cada pequeño incremento en la idoneidad de los resultados puede suponer una diferencia significativa en la prestación de los servicios. Además, estas redes cuentan con un mayor número de recursos, lo cual les permite hacer uso de algoritmos más potentes para conseguir este objetivo. Por tanto, el algoritmo de optimización conjunta resulta idóneo para estos sistemas.
  - *Entornos de investigación*: Una de las características de los sistemas implementados en entornos de investigación es que disponen de los recursos, especialmente tiempo, como para utilizar los modelos más potentes para alcanzar los mejores resultados posibles. Por definición, el algoritmo de optimización conjunta se adapta perfectamente a esta situación proveyendo las soluciones más precisas para permitir a la investigación hallar patrones y obtener conclusiones.

Capítulo 5

Conclusión

En este trabajo se han implementado y analizado diversos algoritmos para la minimización del retardo en escenarios con *splits* funcionales flexibles, donde se ha tenido en cuenta el *scheduling* como factor adicional ya que ambos tienen un gran impacto en el rendimiento de los sistemas. Se ha considerado la optimización conjunta, que brinda los mejores resultados posibles, pero a cambio supone la resolución de un BLP. Esto provoca que pueda ser inviable utilizar este algoritmo en ciertos casos prácticos, ya que es conocido que se trata de un problema *NP-completo*. Debido a esto, se han tenido en cuenta opciones alternativas que fijan uno de los dos parámetros a optimizar, de tal forma que su rendimiento mejora considerablemente a cambio de la obtención de unos resultados que se encuentran por debajo del óptimo. Una vez considerados estos modelos, considerando el trabajo realizado en [3] como base, se han programado y ejecutado sobre diversos escenarios para analizar su comportamiento en situaciones realistas.

Los resultados muestran que en aquellos escenarios donde la red de acceso es homogénea en términos de cómputo, el retardo obtenido con el algoritmo de *scheduling* fijo es estadísticamente similar al ofrecido por la optimización conjunta. Esto no es así para redes heterogéneas en general independientemente del tipo de tráfico, donde se observa una mejora significativa en las soluciones obtenidas mediante la optimización completa respecto a las parciales. Además, la selección de *splits* funcionales elegida por el algoritmo de *scheduling* fijo es, en la gran mayoría de los casos, cercana a la óptima y distinta a mantener fijos los *splits* funcionales, lo que lleva a una mejora en los resultados. En general, el algoritmo de *splits* funcionales fijos apenas presenta ventajas: el retardo mínimo que alcanza es notoriamente superior, y resulta muy poco adaptable ante diferentes tipos de redes y de tráfico. En este sentido, en el caso de elegir hacer uso de uno de los modelos de optimización parcial, la opción idónea es el algoritmo de *scheduling* fijo independientemente de la situación particular.

Con todo esto en mente, la decisión final que queda tomar es, en el probable caso de contar con una red heterogénea, si se valora más un rendimiento más eficiente con un menor consumo de recursos o unos resultados significativamente mejores. Teniendo en cuenta las grandes diferencias en términos de tiempo de ejecución y memoria consumida por los algoritmos, se ha ofrecido una comparativa en detalle con los casos de uso en los

que el empleo de cada uno de los algoritmos resulta la opción más recomendable. En cualquier caso, la elección final dependerá de las características concretas del sistema, así como de las preferencias y objetivos prioritarios de cada situación particular.

5.1 Contribución

Este trabajo forma parte de la línea de investigación correspondiente a las redes de quinta generación del Departamento de Ingeniería de Comunicaciones de la Universidad de Cantabria, más concretamente del Grupo de Ingeniería Telemática. Actualmente, el número de trabajos y documentos que centran su atención en la interacción entre *scheduling* y *splits* funcionales flexibles es muy limitado, y por ello se ha decidido profundizar en este aspecto aportando análisis y resultados que no han sido publicados hasta ahora. Se han implementado y evaluado tres algoritmos diferentes en redes de diversas naturalezas, dando al trabajo tanto un enfoque generalista como un planteamiento más detallado en ciertas situaciones particulares reales. Los resultados han esclarecido varios puntos que requerían de un análisis detallado, algunos de los más destacables se listan a continuación:

- El *scheduling* conocido como *shortest-job-first* no siempre es el óptimo, a no ser que la red sea totalmente homogénea en cuanto a la capacidad de cómputo de las RRHs.
- En redes heterogéneas, estadísticamente hablando, el problema de optimización conjunta brinda mejores resultados. Sin embargo, los costes computacionales adicionales son muy significativos.
- Para los escenarios analizados, los *splits* funcionales flexibles presentan un comportamiento todo-o-nada, donde la centralización óptima siempre es o bien 0 % o bien 100 %. Sin embargo, no presentan ningún patrón aparente y su optimización requiere de algorítmica.
- Por este motivo, en caso de decantarse por un modelo de optimización parcial, estadísticamente fijar el *scheduling* presenta mejores resultados que fijar los *splits* funcionales.

Las contribuciones descritas han dado lugar a publicaciones en conferencias tanto nacionales como internacionales. En concreto ha sido aceptado un artículo en la conferencia IEEE 90th Vehicular Technologyy Conference (VTC-fall[1]), bajo el título *Minimizing delay in NFV 5G Networks by means of flexible split selection and scheduling*. Asimismo, este trabajo también ha sido aceptado para su presentación en las XIV Jornadas de Ingeniería Telemática (JITEL[2]).

[1] http://www.ieeevtc.org/vtc2019fall/
[2] http://jitel2019.i3a.es/

5.2 Trabajo futuro

Tomando como punto de partida los resultados en este trabajo, en el futuro se continuará analizando alternativas adicionales al esquema de optimización conjunta, especialmente en escenarios heterogéneos donde los elementos de acceso tienen diferentes capacidades de cómputo. En particular, las técnicas de *clusterización*, donde se agrupan el tráfico o las RRHs con propiedades similares, podrían simplificar el problema y reducir los costes computacionales. Al mismo tiempo, se van a incluir esquemas de cooperación entre las RRHs y variaciones en los modelos ya existentes, añadiendo la posibilidad de gestionar múltiples tramas de una misma RRH en el mismo instante de tiempo o incluyendo paralelización en la BBU, es decir, procesado simultáneo de varias tramas. Un avance de estas variaciones se ha incluído en la Sección 3.5.

Adicionalmente, se van a considerar escenarios más complejos donde se tiene en cuenta la evolución temporal de la llegada de nuevas tramas a la BBU. En este caso, se van a emplear soluciones *online* que hagan uso de los resultados proporcionados por la teoría de colas [44][45]. De esta manera, una BBU sin paralelización actuaría como un sistema *M/M/1* o *M/G/1*, proporcionando información estadística sobre los retardos medios, la probabilidad de esperar en la cola o los tiempos de espera medios en la misma.

Bibliografía

[1] IEEE 1914 Working Group. *Next generation Fronthaul Interface*. URL: http://sites.ieee.org/sagroups-1914/ (visitado 03-04-2019).

[2] *Study on new radio access technology: Radio access architecture and interfaces*. TR 38.801. Ver. 14.0.0. 3rd Generation Partnership Project (3GPP), 2017.

[3] I. Koutsopoulos. "Optimal functional split selection and scheduling policies in 5G Radio Access Networks". En: *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. Mayo de 2017, págs. 993-998. DOI: 10.1109/ICCW.2017.7962788.

[4] G. O. Pérez, J. A. Hernández y D. Larrabeiti. "Fronthaul network modeling and dimensioning meeting ultra-low latency requirements for 5G". En: *IEEE/OSA Journal of Optical Communications and Networking* 10.6 (jun. de 2018), págs. 573-581. ISSN: 1943-0620. DOI: 10.1364/JOCN.10.000573.

[5] A. Garcia-Saavedra y col. "WizHaul: On the Centralization Degree of Cloud RAN Next Generation Fronthaul". En: *IEEE Transactions on Mobile Computing* 17.10 (oct. de 2018), págs. 2452-2466. ISSN: 1536-1233. DOI: 10.1109/TMC.2018.2793859.

[6] C. I y col. "Rethink fronthaul for soft RAN". En: *IEEE Communications Magazine* 53.9 (sep. de 2015), págs. 82-88. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7263350.

[7] L. M. P. Larsen, A. Checko y H. L. Christiansen. "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks". En: *IEEE Communications Surveys Tutorials* 21.1 (mar. de 2019), págs. 146-172. ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2868805.

[8] D. Harutyunyan y R. Riggio. "Flex5G: Flexible Functional Split in 5G Networks". En: *IEEE Transactions on Network and Service Management* 15.3 (sep. de 2018), págs. 961-975. ISSN: 1932-4537. DOI: 10.1109/TNSM.2018.2853707.

[9] D. Harutyunyan y R. Riggio. "Flexible functional split in 5G networks". En: *2017 13th International Conference on Network and Service Management (CNSM)*. Nov. de 2017, págs. 1-9. DOI: 10.23919/CNSM.2017.8255992.

[10] P. Arnold y col. "5G radio access network architecture based on flexible functional control / user plane splits". En: *2017 European Conference on Networks and Communications (EuCNC)*. Jun. de 2017, págs. 1-5. DOI: 10.1109/EuCNC.2017.7980777.

[11] Y. Alfadhli y col. "Real-Time Demonstration of Adaptive Functional Split in 5G Flexible Mobile Fronthaul Networks". En: *2018 Optical Fiber Communications Conference and Exposition (OFC)*. Mar. de 2018, págs. 1-3.

[12] D. A. Temesgene, M. Miozzo y P. Dini. "Dynamic Functional Split Selection in Energy Harvesting Virtual Small Cells Using Temporal Difference Learning". En: *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. Sep. de 2018, págs. 1813-1819. DOI: 10.1109/PIMRC.2018.8580970.

[13] L. Wang y S. Zhou. "Flexible Functional Split in C-RAN with Renewable Energy Powered Remote Radio Units". En: *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. Mayo de 2018, págs. 1-6. DOI: 10.1109/ICCW.2018.8403571.

[14] A. Marotta y col. "Efficient Management of Flexible Functional Split through Software Defined 5G Converged Access". En: *2018 IEEE International Conference on Communications (ICC)*. Mayo de 2018, págs. 1-6. DOI: 10.1109/ICC.2018.8422555.

[15] Y. Li y col. "Flexible RAN: A Radio Access Network Concept with Flexible Functional Splits and a Programmable Optical Transport". En: *2017 European Conference on Optical Communication (ECOC)*. Sep. de 2017, págs. 1-3. DOI: 10.1109/ECOC.2017.8345857.

[16] C. Chang, N. Nikaein y T. Spyropoulos. "Impact of Packetization and Scheduling on C-RAN Fronthaul Performance". En: *2016 IEEE Global Communications Conference (GLOBECOM)*. Dic. de 2016, págs. 1-7. DOI: 10.1109/GLOCOM.2016.7841885.

[17] C. Chang y col. "FlexCRAN: A flexible functional split framework over ethernet fronthaul in Cloud-RAN". En: *2017 IEEE International Conference on Communications (ICC)*. Mayo de 2017, págs. 1-7. DOI: 10.1109/ICC.2017.7996632.

[18] G. Gu y G. Peng. "The survey of GSM wireless communication system". En: *2010 International Conference on Computer and Information Application*. Dic. de 2010, págs. 121-124. DOI: 10.1109/ICCIA.2010.6141552.

[19] Marc St-Hilaire. "Topological planning and design of UMTS mobile networks: a survey". En: *Wireless Communications and Mobile Computing* 9.7 (2009), págs. 948-958. DOI: 10.1002/wcm.644.

[20] Nan Jiang y col. "Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis". En: *MobiSys'12 - Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (jun. de 2012). DOI: 10.1145/2307636.2307660.

[21] U. Varshney. "4G Wireless Networks". En: *IT Professional* 14.5 (sep. de 2012), págs. 34-39. ISSN: 1520-9202. DOI: 10.1109/MITP.2012.71.

[22] A. Choudhury, A. Sharma y U. Bhattacharya. "A survey on location management in LTE network". En: *2017 International Conference on Trends in Electronics and Informatics (ICEI)*. Mayo de 2017, págs. 278-283. DOI: 10.1109/ICOEI.2017.8300932.

[23] V. S. Pandi y J. L. Priya. "A survey on 5G mobile technology". En: *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. Sep. de 2017, págs. 1656-1659. DOI: 10.1109/ICPCSI.2017.8391995.

[24] Rupendra Nath Mitra y Dharma P. Agrawal. "5G mobile technology: A survey". En: *ICT Express* 1.3 (2015). Special Issue on Next Generation (5G/6G) Mobile Communications, págs. 132 -137. ISSN: 2405-9595. DOI: 10.1016/j.icte.2016.01.003.

[25] A. Gupta y R. K. Jha. "A Survey of 5G Network: Architecture and Emerging Technologies". En: *IEEE Access* 3 (2015), págs. 1206-1232. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2015.2461602.

[26] Ramón Agüero. *Diseño y Operación de Redes Telemáticas*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/2737.pdf (visitado 22-05-2019).

[27] Y. Tao y col. "A survey: Several technologies of non-orthogonal transmission for 5G". En: *China Communications* 12.10 (oct. de 2015), págs. 1-15. ISSN: 1673-5447. DOI: 10.1109/CC.2015.7315054.

[28] L. M. P. Larsen, A. Checko y H. L. Christiansen. "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks". En: *IEEE Communications Surveys Tutorials* 21.1 (mar. de 2019), págs. 146-172. ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2868805.

[29] D. Wubben y col. "Benefits and Impact of Cloud Computing on 5G Signal Processing: Flexible centralization through cloud-RAN". En: *IEEE Signal Processing Magazine* 31.6 (nov. de 2014), págs. 35-44. ISSN: 1053-5888. DOI: 10.1109/MSP.2014.2334952.

[30] P. Rodríguez y col. "Simulación genérica a nivel de sistema para soluciones avanzadas de gestión de recursos". En: *XIII Jornadas de Ingeniería Telemática: JITEL*. Sep. de 2017. DOI: 10.4995/JITEL2017.2017.6608.

[31] K. Wang y col. "Computation Diversity in Emerging Networking Paradigms". En: *IEEE Wireless Communications* 24.1 (feb. de 2017), págs. 88-94. ISSN: 1536-1284. DOI: 10.1109/MWC.2017.1600161WC.

[32] P. Rost y col. "Benefits and challenges of virtualization in 5G radio access networks". En: *IEEE Communications Magazine* 53.12 (dic. de 2015), págs. 75-82. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7355588.

[33] N. Makris y col. "Experimental evaluation of functional splits for 5G cloud-RANs". En: *2017 IEEE International Conference on Communications (ICC)*. Mayo de 2017, págs. 1-6. DOI: 10.1109/ICC.2017.7996493.

[34] Xiao-Long Wu y col. "Packet size distribution of typical Internet applications". En: *2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP)*. Dic. de 2012, págs. 276-281. DOI: 10.1109/ICWAMTIP.2012.6413493.

[35] Z. Sun y col. "Internet QoS and traffic modelling". En: *IEE Proceedings - Software* 151.5 (oct. de 2004), págs. 248-255. ISSN: 1462-5970. DOI: 10.1049/ip-sen:20041087.

[36] A. Juttner y col. "Lagrange relaxation based method for the QoS routing problem". En: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*. Vol. 2. Abr. de 2001, 859-868 vol.2. DOI: 10.1109/INFCOM.2001.916277.

[37] Natashia Boland, John Dethridge e Irina Dumitrescu. "Accelerated label setting algorithms for the elementary resource constrained shortest path problem". En: *Operations Research Letters* 34.1 (2006), págs. 58 -68. ISSN: 0167-6377. DOI: https://doi.org/10.1016/j.orl.2004.11.011. URL: http://www.sciencedirect.com/science/article/pii/S0167637705000040.

[38] GNU Project. *GNU Linear Programming Kit*. URL: https://www.gnu.org/software/glpk/ (visitado 10-02-2019).

[39] D. Justice y A. Hero. "A binary linear programming formulation of the graph edit distance". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.8 (ago. de 2006), págs. 1200-1214. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2006.152.

[40] P. Kuendee y U. Janjarassuk. "A comparative study of mixed-integer linear programming and genetic algorithms for solving binary problems". En: *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*. Abr. de 2018, págs. 284-288. DOI: 10.1109/IEA.2018.8387111.

[41] Mohit Tawarmalani y Nikolaos V. Sahinidis. "A polyhedral branch-and-cut approach to global optimization". En: *Mathematical Programming* 103.2 (jun. de 2005), págs. 225-249. ISSN: 1436-4646. DOI: 10.1007/s10107-005-0581-8. URL: https://doi.org/10.1007/s10107-005-0581-8.

[42] Q. C. Li y col. "5G Network Capacity: Key Elements and Technologies". En: *IEEE Vehicular Technology Magazine* 9.1 (mar. de 2014), págs. 71-78. ISSN: 1556-6072. DOI: 10.1109/MVT.2013.2295070.

[43] Valgrind Developers. *Valgrind*. URL: http://valgrind.org/ (visitado 07-03-2019).

[44] Ramón Agüero. *Redes de Comunicaciones*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/2890.pdf (visitado 22-05-2019).

[45] Ramón Agüero. *Dimensionamiento y Planificación de Redes*. URL: https://www.tlmat.unican.es/siteadmin/submaterials/2707.pdf (visitado 22-05-2019).