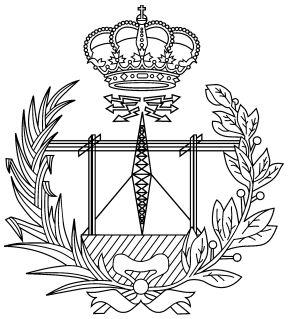
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Trabajo Fin de Máster

DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE MECANISMOS DE DIVISIÓN FUNCIONAL EN REDES DE ACCESO CELULARES SOBRE PLATAFORMAS SDR

(Design, implementation and assessment of Radio Access Networks functional split mechanisms over SDR platforms)

Para acceder al Título de

Máster Universitario en Ingeniería de Telecomunicación

Autor: Alberto Castilla Gómez

Septiembre- 2022

UNIVERSIDAD DE CANTABRIA

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

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE MÁSTER

Realizado por: Alberto Castilla Gómez
Director del TFG: Luis Francisco Diez Fernández y Ramón Agüero Calvo
Título: "Diseño, implementación y evaluación de mecanismos de división funcional en redes de acceso celulares sobre plataformas SDR"
Title: "Design, implementation and assessment of Radio Access Networks functional split mechanisms over SDR platforms"

Presentado a examen el día: 16 de septiembre de 2022

para acceder al Título de

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

Composición del tribunal:
Presidente (Apellidos, Nombre): Sánchez Espeso, Pablo Pedro
Secretario (Apellidos, Nombre): García Gutiérrez, Alberto Eloy
Vocal (Apellidos, Nombre): Sánchez González, Luis

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

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFM
(solo si es distinto del Secretario)

V^{o} B^{o} del Subdirector

Trabajo Fin de Máster N^{o}
(a asignar por Secretaría)

Agradecimientos

A Ramón Agüero y a Luis Diez por su profesionalidad, eficacia y dedicación.

A mis compañeros de clase y amigos por ayudarme y apoyarme durante esta etapa.

Por último, a los más importantes: mi familia con especial mención a mis padres y mi hermano, pilares fundamentales de mi vida.

Resumen

El desarrollo de las tecnologías 5G y 6G viene acompañado de cambios importantes en las redes de acceso, como consecuencia de la capacidad de virtualizar y ejecutar, sobre equipos de propósito general, las funcionalidades de las estaciones base que antes se debían realizar en equipos dedicados, complejos y de altas capacidades. Gracias a esta transformación surge el concepto de *functional split*. Esta virtualización, basada en *Software Defined Network/Network Function Virtualization* (SDN/NFV), ha permitido junto con el de *Cloud/Centralizad Radio Access Network* (C-RAN) una mayor cooperación de los elementos de la red, lográndose una mayor adaptación ante la aparición de nuevos servicios, con requisitos heterogéneos, y una mayor demanda. Por otro lado, se han desarrollado soluciones *Software Defined Radio* (SDR) que permiten desplegar estaciones bases de diferentes generaciones celulares en equipos de bajas prestaciones, facilitando el proceso de desarrollo de soluciones y su posterior validación por personal investigador y académico, ajeno a las operadoras. En este trabajo, se ha utilizado la solución SDR *srsRAN* (antes denominada *srsLTE*), adaptando su implementación para lograr la separación y aislamiento de diversas funcionalidades de red. De este modo, se consigue centralizar parte de su operación, que se ejecuta de forma virtualizada, en el propio equipo u otro. Además, se ha realizado un estudio del consumo computacional de la implementación realizada en diferentes equipos y la validación de las diferentes divisiones funcionales, llamadas *splits*, desarrolladas.

Palabras clave: *5G*; *functional split*; *LTE*; *RAN*; *SDN/NFV*; *SDR*; *srsRAN*

Abstract

The development of 5G and 6G technologies comes together with major changes in the access network architecture, due to the ability to virtualise and execute, in general purpose equipment, base station functionalities that previously were only deployed in specific purpose devices. This evolution gave rise to the concept of "functional split". This virtualisation capability, leveraging *Software Defined Network/Network Function Virtualisation* (SDN/NFV), together with the *Cloud/Centralised Radio Access Network* (C-RAN), has enabled a greater cooperation between network elements, easing the adaptation of new emerging services, with heterogeneous requirements, and the greater capacity demand. On the other hand, *Software Defined Radio* (SDR) solutions have been developed, and they allow the deployment of base stations (belonging to different cellular technologies) in low-capability equipment. In turn, it has simplified the process of developing solutions and their subsequent validation by the research and scientific communities, outside operators. In this work, we have modified the implementation of the SDR solution *srsRAN* (previously named *srsLTE*) to achieve the separation and isolation of different network functionalities. In this way, it is possible to centralise part of its operation, which is executed in virtualised resources, at the same or another equipment. In addition, we have also studied the computational consumption of the implemented solution in different devices, and we have assessed the feasibility of various functional divisions, called "splits".

Keywords: *5G*; *functional split*; *LTE*; *RAN*; *SDN/NFV*; *SDR*; *srsRAN*

Índice general

Índice de figuras **8**

Índice de tablas **14**

Índice de acrónimos **15**

1 Introducción **19**
1.1. Motivación . . . 19
1.2. Objetivos . . . 20
1.3. Estructura del documento . . . 20

2 Estado del Arte **22**
2.1. La evolución de las redes celulares . . . 22
2.1.1. 1ª Generación (1G) . . . 22
2.1.2. 2ª Generación (2G) . . . 22
2.1.3. 3ª Generación (3G) . . . 23
2.1.4. 4ª Generación (4G) . . . 23
2.2. 5ª Generación (5G) . . . 24
2.3. Arquitectura Cloud-RAN . . . 27
2.4. Functional Split . . . 30
2.4.1. Opciones de Functional split: descripción y justificación . . . 32
2.4.2. Requisitos del Fronthaul en cada *split*: latencia y *throughput* . . . 36

3 Instalación y herramientas utilizadas **38**
3.1. srsRAN . . . 38
3.2. zeroMQ . . . 39
3.3. Protocol Buffers . . . 40
3.4. Iperf . . . 42
3.5. Herramienta top Linux . . . 43
3.6. Traffic Control . . . 43

4 Desarrollo e implementación **45**

4.1. Visión general: diagramas de clases y estructura definida . . . 45
4.2. Diseño de los *wrappers* . . . 47
4.2.1. Atributos del *wrapper* . . . 47
4.2.2. *Sockets* de transmisión y recepción . . . 50
4.2.3. Métodos del *wrapper* . . . 52
4.2.4. Establecimiento de los *splits*: ficheros de configuración . . . 56
4.2.5. Integración de los *wrappers* en la plataforma . . . 57
4.3. Automatización de las pruebas . . . 59
4.4. Tratamiento y adaptación de los ficheros de resultados . . . 61

5 Evaluación de la implementación **64**
5.1. Equipos y escenarios . . . 64
5.2. Resultados . . . 67
5.2.1. Escenarios I y II . . . 68
5.2.2. Escenario III: comprobación del cumplimiento de latencia . . . 79

6 Conclusiones **83**
6.1. Líneas futuras . . . 84

Bibliografía **85**

A Resultados completos **88**
A.1. Escenario I y II . . . 88
A.2. Escenario III . . . 106

Índice de figuras

2.1. Estructura y funcionamiento de una red *5ª Generación* (5G) *Non-Standalone* (NSA). 25
2.2. Comparativa de los KPI del 5G SA (IMT-2020) y del 4G (IMT-Advanced) [18]. 26
2.3. Casos de uso y capacidades clave según la ITU-R M.2083 [18]. 27
2.4. *Network Slicing*, despliegue de *slices* para cada escenario de la ITU. Traducción y adaptación de [25]. 27
2.5. Arquitectura *Cloud/Centralized Radio Access Network* (C-RAN). 29
2.6. Evolución de la arquitectura C-RAN hacia el concepto de *functional split*. 30
2.7. Pila de protocolos de una *Estación Base* (EB) *Long Term Evolution* (LTE). Traducción y adaptación de [31]. 31
2.8. Splits funcionales entre *Centralized Unit* (CU)-*Distributed Unit* (DU). Traducción y adaptación de [28]. 32

3.1. Proceso de generación de las clases *.proto*. 41
3.2. Integración y empleo de las clases *.proto* en el desarrollo. 42
3.3. Establecimiento de *qdiscs*, clases y filtros en un interfaz de comunicaciones. 44

4.1. Comparativa entre la versión original y la final, comunicación PDCP $\leftrightarrow$ RLC. 47
4.2. Diagrama de clases de la implementación original: relaciones de herencia, agregación y asociación. 49
4.3. Diagrama de clases de la implementación final: relaciones de herencia, agregación, asociación y comunicación de las capas con los *wrappers*. 49
4.4. Comunicación PDCP $\leftrightarrow$ RLC: agentes que permiten el intercambio de información. 50
4.5. *Sockets* establecidos entre *rlc_pdcp_wrapper* y *pdcp_rlc_wrapper* para la comunicación PDCP $\leftrightarrow$ RLC. 50
4.6. Operativa de trabajo en función de la configuración del *wrapper*. 50
4.7. Clase *fs_socket*: atributos y métodos 51
4.8. Operativa de trabajo de los métodos implementados. Ejemplo: *discard_sdu*() 53
4.9. Configuración de la plataforma *srsRAN* para el establecimiento de la conexión y comunicación. 57
4.10. Procedimiento de despliegue de los diferentes escenarios en función de la configuración. . . 59

4.11. Procedimiento de automatización de las pruebas. Se muestra el escenario en el cual se divide la red en varios equipos, de manera que los procesos se ejecutan por separado. Mediante los colores de los procesos, se indican qué procesos se ejecutan sobre el mismo dispositivo. Si por el contrario se despliega toda la red en el mismo equipo, todos los procesos se realizan en el mismo dispositivo. 61

5.1. Topología desplegada para las pruebas del Escenario II. 65
5.2. Despliegue real en el laboratorio para las pruebas del Escenario II. 66
5.3. Topología desplegada para las pruebas del Escenario III. 66
5.4. Valores medios del uso de memoria de la versión *LEGACY* en el Escenario I en función del dispositivo y del número de PRBs. 68
5.5. Valores medios del uso de memoria en el *split PDCP_RLC* en el Escenario I en función del dispositivo y del número de PRBs. 69
5.6. Valores medios del uso de memoria en el *split RLC_MAC* en el Escenario I en función del dispositivo y del número de PRBs. 69
5.7. Valores medios del consumo de CPU de la versión *LEGACY* en el Escenario I en función del dispositivo, el número de PRBs y la tasa *iperf* de operación. 71
5.8. Valores medios del consumo de CPU del *split PDCP_RLC* en el Escenario I en función del dispositivo, el número de PRBs y la tasa *iperf* de operación. 72
5.9. Valores medios del consumo de CPU del *split RLC_MAC* en el Escenario I en función del dispositivo, el número de PRBs y la tasa *iperf* de operación. 72
5.10. Comportamiento de la CU en términos del consumo de CPU en el *split PDCP_RLC* en el Escenario I. 73
5.11. Evolución del comportamiento de la eNB a nivel de CPU en cada uno de los dispositivos de prueba para las tasas de 100 PRBs. Cada uno de los *boxplots* de cada uno de los 3 escenarios representados se corresponden (de izquierda a derecha) con las tasas de 6, 12 y 18 Mbps. . . 73
5.12. Evolución del comportamiento de la CU-DU del *split PDCP_RLC* a nivel de CPU en los Escenarios I y II para las tasas de 100 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 6 Mbps, los segundos con 12 Mbps y los terceros con 18 Mbps. 75
5.13. Evolución del comportamiento de la CU-DU del *split RLC_MAC* a nivel de CPU en los Escenarios I y II para las tasas de 100 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 6 Mbps, los segundos con 12 Mbps y los terceros con 18 Mbps. 75
5.14. Esquema de la comunicación PDCP $\longleftrightarrow$ RLC en el envío de tráfico UL en el *PDCP_RLC_SPLIT*. 76
5.15. Tráfico enviado por el *pdcp_rlc_wrapper* en el *split PDCP_RLC* en los Escenarios I y II en función del equipo, el número de PRBs y la tasa *iperf*. 77
5.16. Tráfico enviado por el wrapper *rlc_pdcp_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*. 77
5.17. Esquema de la comunicación RLC $\longleftrightarrow$ MAC en el envío de tráfico UL en el *RLC_MAC_SPLIT*. 77
5.18. Tráfico enviado por el wrapper *rlc_mac* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 78

5.19. Tráfico enviado por el wrapper *mac_rlc* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 78
5.20. Tráfico enviado por el wrapper *rrc_mac* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 79
5.21. Valores medios del uso de memoria de la CU-DU en el *split PDCP_RLC* tanto en el caso sin latencia como en el caso en el que se añade la latencia de 30 ms mientras se envía tráfico UL. 80
5.22. Valores medios del consumo de CPU de la CU-DU en el *split PDCP_RLC* tanto en el caso sin latencia como en el caso en el que se añade la latencia de 30 ms mientras se envía tráfico UL. 80
5.23. Comportamiento de la CU-DU en el *split PDCP_RLC* para cada una de las tasas de 15 PRBs en función de la existencia o no de *latencia* adicional en el *fronthaul* mientras se envía tráfico UL. Para los dos supuestos (sin y con latencia), la primera pareja (verde y azul) de boxplots se corresponden con 1 Mbps, la segunda con 2 Mbps y la tercera con 4 Mbps. 81
5.24. Tráfico enviado por *pdcp_rlc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL 82
5.25. Tráfico enviado por *rlc_pdcp_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL 82

A.1. Comportamiento de la CU en términos del consumo de CPU en el *split RLC_MAC* en el Escenario I. 88
A.2. Evolución del comportamiento de la eNB a nivel de CPU en cada uno de los dispositivos de prueba para las tasas de 15 PRBs. Cada uno de los *boxplots* de cada uno de los 3 escenarios representados se corresponden (de izquierda a derecha) con las tasas de 1, 2 y 4 Mbps. . . . 89
A.3. Evolución del comportamiento de la eNB a nivel de CPU en cada uno de los dispositivos de prueba para las tasas de 25 PRBs. Cada uno de los *boxplots* de cada uno de los 3 escenarios representados se corresponden (de izquierda a derecha) con las tasas de 4, 6 y 7.5 Mbps. . . 89
A.4. Evolución del comportamiento de la eNB a nivel de CPU en cada uno de los dispositivos de prueba para las tasas de 50 PRBs. Cada uno de los *boxplots* de cada uno de los 3 escenarios representados se corresponden (de izquierda a derecha) con las tasas de 4, 8 y 12 Mbps. . . . 90
A.5. Evolución del comportamiento de la CU-DU del *split PDCP_RLC* a nivel de CPU en los Escenarios I y II para las tasas de 15 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 1 Mbps, los segundos con 2 Mbps y los terceros con 4 Mbps. 90
A.6. Evolución del comportamiento de la CU-DU del *split PDCP_RLC* a nivel de CPU en los Escenarios I y II para las tasas de 25 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 4 Mbps, los segundos con 6 Mbps y los terceros con 7.5 Mbps. 91
A.7. Evolución del comportamiento de la CU-DU del *split PDCP_RLC* a nivel de CPU en los Escenarios I y II para las tasas de 50 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 4 Mbps, los segundos con 8 Mbps y los terceros con 12 Mbps. 91

A.8. Evolución del comportamiento de la CU-DU del *split RLC_MAC* a nivel de CPU en los Escenarios I y II para las tasas de 15 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 1 Mbps, los segundos con 2 Mbps y los terceros con 4 Mbps. 92
A.9. Evolución del comportamiento de la CU-DU del *split RLC_MAC* a nivel de CPU en los Escenarios I y II para las tasas de 25 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 4 Mbps, los segundos con 6 Mbps y los terceros con 7.5 Mbps. 92
A.10. Evolución del comportamiento de la CU-DU del *split RLC_MAC* a nivel de CPU en los Escenarios I y II para las tasas de 50 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 4 Mbps, los segundos con 8 Mbps y los terceros con 12 Mbps. 93
A.11. Evolución del comportamiento de la eNB o CU-DU en el *PC1* de los *splits* implementados para las tasas de 15 PRBs (1, 2 y 4 Mbps). 93
A.12. Evolución del comportamiento de la eNB o CU-DU en el *PC1* de los *splits* implementados para las tasas de 25 PRBs (4, 6 y 7.5 Mbps). 94
A.13. Evolución del comportamiento de la eNB o CU-DU en el *PC1* de los *splits* implementados para las tasas de 50 PRBs (4, 8 y 12 Mbps). 94
A.14. Evolución del comportamiento de la eNB o CU-DU en el *PC1* de los *splits* implementados para las tasas de 100 PRBs (6, 12 y 18 Mbps). 95
A.15. Evolución del comportamiento de la eNB o CU-DU en el *PC2* de los *splits* implementados para las tasas de 15 PRBs (1, 2 y 4 Mbps). 95
A.16. Evolución del comportamiento de la eNB o CU-DU en el *PC2* de los *splits* implementados para las tasas de 25 PRBs (4, 6 y 7.5 Mbps). 96
A.17. Evolución del comportamiento de la eNB o CU-DU en el *PC2* de los *splits* implementados para las tasas de 50 PRBs (4, 8 y 12 Mbps). 96
A.18. Evolución del comportamiento de la eNB o CU-DU en el *PC2* de los *splits* implementados para las tasas de 100 PRBs (6, 12 y 18 Mbps). 97
A.19. Evolución del comportamiento de la eNB o CU-DU en el *PC3* de los *splits* implementados para las tasas de 15 PRBs (1, 2 y 4 Mbps). 97
A.20. Evolución del comportamiento de la eNB o CU-DU en el *PC3* de los *splits* implementados para las tasas de 25 PRBs (4, 6 y 7.5 Mbps). 98
A.21. Evolución del comportamiento de la eNB o CU-DU en el *PC3* de los *splits* implementados para las tasas de 50 PRBs (4, 8 y 12 Mbps). 98
A.22. Evolución del comportamiento de la eNB o CU-DU en el *PC3* de los *splits* implementados para las tasas de 100 PRBs (6, 12 y 18 Mbps). 99
A.23. Tráfico enviado por el wrapper *gtpu_rrc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*. 99
A.24. Tráfico enviado por el wrapper *mac_rrc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*. 100
A.25. Tráfico enviado por el wrapper *rrc_mac_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*. 100

A.26. Tráfico enviado por el wrapper *rrc_rlc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*. 101
A.27. Tráfico enviado por el wrapper *rlc_rrc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*. 101
A.28. Tráfico enviado por el wrapper *rrc_pdcp_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*. 102
A.29. Tráfico enviado por el wrapper *pdcp_rrc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*. 102
A.30. Tráfico enviado por el wrapper *gtpu_rrc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 103
A.31. Tráfico enviado por el wrapper *mac_rrc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 103
A.32. Tráfico enviado por el wrapper *pdcp_rrc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 104
A.33. Tráfico enviado por el wrapper *rrc_pdcp_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 104
A.34. Tráfico enviado por el wrapper *rlc_rrc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 105
A.35. Tráfico enviado por el wrapper *rrc_rlc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*. 105
A.36. Valores medios del uso de memoria de la CU-DU en el *split RLC_MAC* tanto en el caso sin latencia como en el caso en el que se añade la latencia de 30 ms mientras se envía tráfico UL. 106
A.37. Valores medios del consumo de CPU de la CU-DU en el *split RLC_MAC* tanto en el caso sin latencia como en el caso en el que se añade la latencia de 30 ms mientras se envía tráfico UL. Para los dos supuestos (sin y con latencia), la primera pareja (verde y azul) de boxplots se corresponden con 1 Mbps, la segunda con 2 Mbps y la tercera con 4 Mbps. 106
A.38. Comportamiento de la CU-DU en el *split RLC_MAC* para cada una de las tasas de 15 PRBs en función de la existencia o no de *latencia* adicional en el *fronthaul* mientras se envía tráfico UL. 107
A.39. Tráfico enviado por *gtpu_rrc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL 107
A.40. Tráfico enviado por *pdcp_rrc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL 108
A.41. Tráfico enviado por *rrc_pdcp_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL 108
A.42. Tráfico enviado por *gtpu_rrc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL 109
A.43. Tráfico enviado por *pdcp_rlc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL 109
A.44. Tráfico enviado por *rlc_pdcp_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL 110

A.45. Tráfico enviado por *pdcp_rrc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL . . . 110
A.46. Tráfico enviado por *rrc_pdcp_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL . . . 111
A.47. Tráfico enviado por *gtpu_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL . . . 111
A.48. Tráfico enviado por *mac_rlc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL . . . 112
A.49. Tráfico enviado por *rlc_mac_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL . . . 112
A.50. Tráfico enviado por *pdcp_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL . . . 113
A.51. Tráfico enviado por *rrc_pdcp_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL . . . 113
A.52. Tráfico enviado por *rlc_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL 114
A.53. Tráfico enviado por *rrc_rlc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL 114
A.54. Tráfico enviado por *gtpu_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL . . . 115
A.55. Tráfico enviado por *mac_rlc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL . . . 115
A.56. Tráfico enviado por *rlc_mac_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL . . . 116
A.57. Tráfico enviado por *rlc_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL 116
A.58. Tráfico enviado por *rrc_rlc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL 117
A.59. Tráfico enviado por *pdcp_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL . . . 117
A.60. Tráfico enviado por *rrc_pdcp_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL . . . 118

Índice de tablas

2.1. Requisitos temporales del *split Medium Access Control* (MAC) 37

4.1. Tipos de mensajes enviados por el *socket*: campos y tamaños. 55
4.2. Capas desplegadas en cada elemento en función del tipo de *split* 58
4.3. Estado de trabajo de las separaciones funcionales del *stack* de protocolos. En color verde y el símbolo ✓ se resaltan aquellas separaciones que se han implementado y validado, en amarillo y ⁛ las que se encuentran en fase de desarrollo y validación. En rojo y ✗, aquellas que no se han acometido y, por último, en gris se representan aquellas separaciones que no existen. . . 58
4.4. Parámetros de configuración de las pruebas realizadas. 60
4.5. Línea cabecera donde se indican qué campos conforman las medidas. En color rojo, se indican qué campos se descartan y, en verde, cuáles se mantienen. 62
4.6. Campos que constituyen las métricas *iperf*. En color rojo, se indican aquellos campos que se descartan mientras que, en verde, se indican los que se conservan. 62

5.1. Características de los dispositivos utilizados en la implementación. 64
5.2. Direcciones IP y roles que asumen los equipos en el Escenario II. 65
5.3. Direcciones IP y roles que asumen los equipos en el Escenario III. 66

Índice de acrónimos

3G	*3ª Generación*.
3GPP	*Third Generation Partnership Project*.
4G	*4ª Generación*.
5G	*5ª Generación*.
5GC	*5G Core*.
AI	*Artificial Intelligence*.
AMPS	*Advanced Mobile Phone System*.
AR	*Augmented Reality*.
ARQ	*Automatic Repeat Request*.
BBU	*Base Band Units*.
BS	*Base Station*.
BW	*Bandwidth*.
CDMA	*Code Division Modulation Access*.
CoMP	*Coordinated Multi-Point transmission and reception*.
CPRI	*Common Public Radio Interface*.
CPU	*Central Processing Unit*.
C-RAN	*Cloud/Centralized Radio Access Network*.
CU	*Centralized Unit*.
DL	*Downlink*.
D-RAN	*Distributed RAN*.
DU	*Distributed Unit*.
EB	*Estación Base*.
EDGE	*Enhanced Data GSM Enviroment*.
eMBB	*Enhanced Mobile Broadband*.

eNB	*eNodeB*.
EPC	*Evolved Packet Core*.
ETSI	*European Telecommunications Standards Institute*.
E-UTRA	*Evolved UTRA*.
FDMA	*Frequency Division Modulation Access*.
FO	*Fibra Óptica*.
FR	*Frequency Range*.
GPRS	*General Packet Radio Service*.
GSM	*Global Systems for Mobile Communications*.
GTP-U	*GPRS Tunnelling Protocol User Plane*.
HARQ	*Hybrid Automatic Repeat Request*.
HSPA	*High Speed Packet Access*.
HSS	*Home Subscriber Service*.
ICI	*Inter-Cloud Interfaces*.
IoT	*Internet of Things*.
IP	*Internet Protocol*.
JSON	*JavaScript Object Notation*.
KPI	*Key Performance Indicators*.
LTE	*Long Term Evolution*.
LTE-A	*LTE Advanced*.
M2M	*Machine to Machine*.
MAC	*Medium Access Control*.
MIMO	*Multiple-Input Multiple-Output*.
ML	*Machine Learning*.
MME	*Mobility Management Entity*.
MMS	*Multimedia Messaging Service*.
mMTC	*Massive Machine Type Communicactions*.
NFV	*Network Function Virtualization*.
NR	*New Radio*.
NSA	*Non-Standalone*.
OFDMA	*Orthohonal Frequency Division Modulation Access*.

PDCP	*Packet Data Convergence Packet*.
PDU	*Protocol Data Unit*.
PHY	*Physical*.
PID	Process ID.
PRACH	*Physical Random Access Channel*.
PRB	*Physical Resource Block*.
QoS	*Quality of Service*.
RAN	*Radio Access Network*.
RF	*Radio frequency*.
RLC	*Radio Link Control*.
RRC	*Radio Resource Control*.
RRU	*Remote Radio Units*.
RTT	*Round Trip Time*.
RU	*Radio Unit*.
S/P-GW	*S/Packet Gateway*.
S1-AP	*S1-Application Protocol*.
SA	*Standalone*.
SC-FDMA	*Single Carrier Frequency Division Modulation Access*.
SDAP	*Service Data Adaption Protocol*.
SDN	*Software Defined Network*.
SMS	*Short Message Service*.
SO	*Sistema Operativo*.
STCP	*Stream Transmission Control Protocol*.
TACS	*Total Access Communication System*.
TC	*Traffic Control*.
TCP	*Transmission Control Protocol*.
TDMA	*Time Division Modulation Access*.
UDP	*User Datagram Protocol*.
UE	*User Equipment*.
UL	*Uplink*.
UMTS	*Universal Mobile Telecommunications System*.
URLLC	*Ultra-reliable and Low Latency Communications*.
USRP	*Universal Software Radio Peripheral*.
V2X	*Vehicle-To-Anything*.

VoIP	*Voice over IP*.
VR	*Virtual Reality*.

Capítulo 1

Introducción

1.1. Motivación

El desarrollo e implantación de la tecnología 5G está permitiendo satisfacer los requisitos de nuevos escenarios de comunicaciones y servicios, muy heterogéneos en términos de tasa binaria, latencia o fiabilidad, principalmente. Por ejemplo, servicios multimedia, como la Realidad Virtual (*Virtual Reality*, VR) o la Realidad Aumentada (*Augmented Reality*, AR), requieren una tasa binaria elevada. Otros servicios precisan una alta fiabilidad de las comunicaciones y una baja latencia, como los destinados a actividades relacionadas con vehículos autónomos o el control remoto de robots médicos utilizados para tele-cirugía o servicios como las comunicaciones vehiculares (*Vehicle-To-Anything*, V2X). Asimismo, otro tipo de servicios, tales como los que se enmarcan en el Internet de las Cosas (*Internet of Things*, IoT), precisan dar soporte a la comunicación entre enormes cantidades de dispositivos [1].

Sin embargo, las arquitecturas de red de las generaciones anteriores fueron diseñadas con el objetivo principal de maximizar la capacidad que percibían los usuarios y no consideraban los nuevos escenarios que se presentan en el 5G. A fin de dar respuesta a los requisitos heterogéneos anteriormente referidos, las arquitecturas 5G han adoptado nuevos paradigmas de comunicaciones, tales como SDN/NFV (*Software Defined Network/Network Function Virtualization*) o el C-RAN (*Cloud/Centralized Radio Access Network*) [2]. El primero permite implementar las funcionalidades de red como componentes *software*, lo que facilita su despliegue, actualización y gestión. El segundo hace uso de Computación en Nube (en inglés, *Cloud Computing*) aplicada a la red de acceso, permitiendo virtualizar y centralizar funcionalidades tradicionalmente asociadas a las estaciones base.

En la arquitectura C-RAN coexisten dos tipos de elementos: las *Remote Radio Units* (RRU), simplemente *Radio Unit* (RU), y las *Base Band Units* (BBU). Las RRU realizan las funcionalidades correspondientes a la capa radio, incluso a nivel físico, por lo que, se sitúan en los emplazamientos donde están las antenas, mientras que las BBU llevan a cabo el resto de las funcionalidades propias de la estación base. A su vez, una BBU se puede separar en dos sub-componentes, que se reparten las tareas a realizar: CU y DU, dando lugar al concepto de *functional split*, que hace referencia al reparto de las funciones entre CU y DU [1, 3].

Mediante esta arquitectura se consigue compartir recursos, reduciéndose tanto costes como consumo energético. Asimismo, la capacidad de adaptación de las arquitecturas C-RAN permiten soportar las

diferentes clases de tráfico que demandan los servicios emergentes del 5G [2].

De forma paralela al *functional split* aparece el concepto de *Network Slicing*. Si la división funcional se puede ver como una separación vertical del las funcionalidades, tales como protocolos de la red de acceso, el *Network Slicing* propone una segmentación horizontal de los recursos de red. De esta forma, los recursos físicos de red (comunicaciones, almacenamiento y cómputo) se dividen en redes lógicas denominadas *slices*, cada una de ellas con configuración y requisitos independientes [2]. Este nuevo enfoque de red se plantea como una solución idónea para la provisión de los nuevos servicios de 5G.

1.2. Objetivos

Aunque la división funcional está presente en los estándares, tanto del *Third Generation Partnership Project* (3GPP) como de la *European Telecommunications Standards Institute* (ETSI), las actividades de investigación sobre ellos se suelen centrar en enfoques teóricos o escenarios muy limitados. Esto se debe a la falta de entornos que implementen este nuevo paradigma de la red de acceso. Hoy en día existen implementaciones abiertas de las funcionalidades de las estaciones base *4ª Generación* (4G) y 5G, tales como OAI[1] [4] o srsRAN[2] [5], sin embargo, en el momento de comenzar este trabajo, no implementaban la división funcional, o se limitaban a las funcionalidades que están siendo adoptadas a nivel comercial, dificultando la investigación de otras opciones. En este contexto, se ha realizado la implementación de diferentes *functional splits* sobre la plataforma *srsRAN* (denominada previamente como *srsLTE*), llevando a cabo además un análisis de rendimiento. Dicho estudio aborda el uso de la *Central Processing Unit* (CPU) y memoria, así como la generación de tráfico en cada uno de los *splits* implementados. De forma más concreta, los objetivos del trabajo son:

- Diseño *software* de la arquitectura *functional split* a partir del código base de *srsRAN*. Este diseño se ha realizado intentando evitar, en la medida de lo posible, la modificación del código existente, facilitando la integración con versiones futuras.
- Implementación de la separación de unidades funcionales y del sistema de comunicaciones entre ellas de forma transparente.
- Despliegue de la solución *functional split* en diferentes equipos para validar su funcionalidad. Este punto incluye el desarrollo de *scripts* para automatizar el despliegue de las unidades CU y DU.
- Validación de rendimiento de la solución desarrollada sobre varios equipos con diferentes capacidades y comparación con la solución por defecto, en la que no hay separación funcional. Esta validación se ha realizado en términos de carga computacional, uso de memoria y tráfico generado para la comunicación de los elementos distribuidos.

1.3. Estructura del documento

Este documento se ha dividido en 6 capítulos. Tras la introducción planteada en las secciones anteriores, en las que se ha expuesto la motivación y los objetivos del proyecto, el resto del documento se estructura

[1] Página web oficial de OAI: `https://openairinterface.org`
[2] Página web oficial de *srsRAN*: `https://www.srsran.com`

del siguiente modo:

- En el Capítulo 2 se presenta una descripción de la evolución de las redes celulares, con el propósito de contextualizar la situación actual. Tras ello, se exponen las principales características de las redes 5G y, por último, se explica el concepto de *functional split*, sus requisitos, tipos y su implementación en la actualidad.
- El Capítulo 3 resume las principales herramientas utilizadas en el proyecto y se argumenta la funcionalidad y características de cada una de ellas. Asimismo, se explica por qué se ha decidido emplear estas herramientas y no otras similares.
- El Capítulo 4 se detalla la implementación realizada, tomando como punto de partida el desarrollo *srsRAN*. Se describe tanto el diseño *software* del desarrollo, como el despliegue completo de la solución implementada, y el entorno sobre el que se ha llevado a cabo.
- En el Capítulo 5 se detallan los diferentes escenarios en los que se han realizado la validación del desarrollo, y se proporciona una explicación detallada de los resultados obtenidos.
- El Capítulo 6 concluye el documento, resumiendo los principales resultados y conclusiones obtenidas. Además, se exponen las posibles líneas de trabajo futuras que surgen a partir del trabajo realizado.

Capítulo 2

Estado del Arte

A lo largo de este capítulo se presentará, de forma breve, la evolución que se ha venido produciendo en las redes celulares hasta llegar a las actuales de quinta generación. A continuación, se realizará una descripción más detallada de las arquitecturas C-RAN y las soluciones *functional-split*.

2.1. La evolución de las redes celulares

A lo largo de este apartado, se sintetizan las principales características de las generaciones de redes celulares predecesoras a las 5G, con el propósito de contextualizar el trabajo.

2.1.1. 1ª Generación (1G)

La Primera Generación de redes móviles emergió en 1979 en Japón y, a lo largo de la década de los 80, se extendió por Europa y Estados Unidos. Basada en tecnología analógica, *Frequency Division Modulation Access* (FDMA) como mecanismo de acceso al medio y conmutación de circuitos, ofrecía un servicio de telefonía básica y alcanzaba tasas de 2 kbps [6, 7].

Trabajaba en las bandas de 450, 800 y 900 MHz principalmente, y utilizaba canales de 30 kHz de ancho de banda. Entre los estándares desarrollados, destacan *Advanced Mobile Phone System* (AMPS) y *Total Access Communication System* (TACS). El primero de ellos, que operaba a 800 MHz, se utilizó en Estados Unidos y Australia, mientras que TACS (900 MHz) se adoptó en Europa [7, 8].

Su uso se limitaba a la provisión de llamadas telefónicas y, además, se caracterizaba por tener una baja seguridad (no se cifraba la información, por lo que podía ser interceptada por un tercero no autorizado), así como calidad y fiabilidad limitadas. Asimismo, los móviles 1G tenían un gran tamaño y les penalizaba una escasa duración de la batería [6, 7, 9].

2.1.2. 2ª Generación (2G)

En los años 90, se inicia el desarrollo del primer sistema digital de comunicaciones móviles, mediante la tecnología *Global Systems for Mobile Communications* (GSM) que opera a 900 y 1800 MHz y tiene una cobertura prácticamente global [8]. Este estándar, basado en *Time Division Modulation Access* (TDMA) y conmutación de circuitos, ofrece anchos de banda comprendidos entre los 30 y 200 kHz, alcanzándose tasas que se sitúan entre los 14.4 y 64 kbps. Además de las llamadas telefónicas, GSM permite enviar

mensajes de texto y multimedia: *Short Message Service* (SMS) y *Multimedia Messaging Service* (MMS) [7, 9].

Entre sus principales características destacan la mejora en seguridad de las comunicaciones, gracias a la transmisión de la voz cifrada durante las llamadas telefónicas, una mejor calidad de las llamadas, la identificación de llamadas y el *roaming* internacional [6, 7, 9].

Con el propósito de incrementar la tasa binaria se desarrolló la tecnología *General Packet Radio Service* (GPRS), conocida como 2.5G. Incorporaba conmutación de paquetes y conseguía alcanzar velocidades máximas de 144 kbps, permitiendo la provisión de servicios como el envío y/o recepción de correos electrónicos o la navegación web a través de redes celulares [7, 9]. Posteriormente, se desarrolló un nuevo estándar, denominado *Enhanced Data GSM Enviroment* (EDGE), que permitió alcanzar tasas binarias de hasta 384 kbps [10].

2.1.3. 3ª Generación (3G)

A principios de los años 2000, los organismos de estandarización aprobaron la recomendación IMT-2000, que establece los requisitos técnicos de la tecnología *3ª Generación* (3G) [11]. Las redes celulares 3G fueron estandarizadas por el 3GPP como *Universal Mobile Telecommunications System* (UMTS). Este estándar, además de definir la estructura de la red, adopta *Code Division Modulation Access* (CDMA) como mecanismo de acceso al medio, establece el uso de conmutación de paquetes, así como la integración con Internet mediante el protocolo *Internet Protocol* (IP) [8, 11]. Además, ofrece la capacidad de proveer servicios y aplicaciones de valor añadido. La infraestructura permite la convergencia de diferentes tipos de redes (fijas, móviles) y servicios (voz, datos, multimedia) [11].

Este estándar opera en las bandas de 900 MHz y 2.1 GHz con portadoras de 5 MHz, ofreciendo tasas mínimas de 2 Mbps para usuarios de poca movilidad y de 348 kbps para aquellos usuarios situados en vehículos en movimiento [11]. Entre las principales características del 3G destaca el incremento de velocidad, una mayor calidad de las comunicaciones, un aumento de la seguridad, así como una mayor cobertura y la posibilidad de incrementar la densificación de la red. Estas mejoras respecto a las generaciones previas permitieron ofrecer nuevos servicios como TV streaming, videollamadas, juegos en línea, IPTV, geolocalización (*tracking*) o aplicaciones de mensajería instantánea [6, 7, 9], y la aparición de los *smartphones*.

Posteriormente, se desarrolló *High Speed Packet Access* (HSPA), que permitió aumentar la tasa binaria hasta los 14 Mbps en bajada y 5.7 Mbps en subida, gracias a la optimización de los recursos de red utilizados [12].

2.1.4. 4ª Generación (4G)

A finales de 2004, el 3GPP inició el desarrollo del nuevo estándar *Evolved UTRA* (E-UTRA), con el objetivo de diseñar un nuevo interfaz de red. A partir de este proyecto, surgió *Long Term Evolution* (LTE), el precursor del 4G. Entre las principales características de LTE destaca un significativo aumento de la tasa binaria (100 Mbps en bajada y 50 Mbps en subida), una reducción notable de la latencia (inferior a los 10 milisegundos), una mejora de la eficiencia espectral, y la capacidad de soportar velocidades de los usuarios móviles superiores a los 350 km/h. Otro aspecto relevante es su compatibilidad con otras redes

de generaciones previas, como GSM o UMTS [13, 14].

Posteriormente se desarrolló *LTE Advanced* (LTE-A), que permitió aumentar la tasa binaria teórica máxima por estación base de LTE hasta 1 Gbps para usuarios que no estuvieran en movimiento, o que se desplazan a baja velocidad. Se proporciona una red adecuada para servicios en tiempo real o comunicaciones máquina a máquina, en inglés, *Machine to Machine* (M2M) y la provisión de nuevos servicios como la TV Móvil en alta definición, el *streaming* de video, o *Voice over IP* (VoIP), es decir, telefonía IP [7, 14]. La Cuarta Generación, LTE-A, trabaja en las bandas de 800 MHz, 1.8 y 2.6 GHz, empleando *Orthohonal Frequency Division Modulation Access* (OFDMA) como mecanismo de acceso al medio y portadoras de 180 kHz [15].

Por otro lado, 4G se caracteriza por una alta integración con redes IP, donde la comunicación está orientada a paquetes, permitiendo la convergencia de los servicios de voz y datos. Además, consigue mejorar la movilidad y la seguridad de las comunicaciones, y aumentar la calidad de los servicios, o *Quality of Service* (QoS), de voz [7].

2.2. 5ª Generación (5G)

La quinta generación de redes celulares, conocida como el 5G, ha sido estandarizada por el 3GPP. Como método de acceso al medio, al igual que 4G, utiliza OFDMA (la variante *Single Carrier Frequency Division Modulation Access* (SC-FDMA) para el enlace ascendente). Además, para favorecer su integración, se establecieron 2 tipos de despliegues de redes 5G: 5G *Non-Standalone* (5G NSA) y 5G *Standalone* (SA). Seguidamente, se explican las principales características y diferencias de ambos enfoques [16-18]:

- **5G NSA**: Aprobada en diciembre de 2017 en la Release 15 [19] del 3GPP, se apoya en el uso de la infraestructura de red 4G, concretamente en el *Evolved Packet Core* (EPC), el núcleo de la red.

  De esta manera, las redes 4G y 5G pueden coexistir, centrándose el despliegue 5G en el interfaz radio, denominado *New Radio* (NR). Este nuevo interfaz radio puede operar en dos rangos de frecuencias, conocidos como *Frequency Range* (FR) [20]:

  - ***Frequency Range*** 1 (4.1-7.125 GHz): Se utiliza para transportar la mayor parte del tráfico de las comunicaciones celulares del 5G. Ofrece una mayor cobertura y una buena penetración en edificios.
  - ***Frequency Range*** 2 (24.25-52.6 GHz): Se corresponde con la banda milimétrica (*mmWave*), y emplea antenas de bajo radio de cobertura destinadas a aplicaciones de corto alcance. Además, ofrece una mayor tasa binaria que el rango FR 1. Al caracterizarse por una cobertura limitada, demanda una mayor cantidad de antenas y, por lo tanto, un aumento de los costes.

  Respecto a las comunicaciones, la interacción entre los usuarios y las antenas 5G se realizan empleando protocolos 5G, mientras que la comunicación entre las estaciones base y el resto de elementos de la red se efectúa utilizando los protocolos de 4G. En la Figura 2.1 se presenta de forma gráfica la estructura y el comportamiento de una red 5G NSA:

  El objetivo final de este tipo de despliegue es tratar de mejorar las características de las redes LTE, permitiendo alcanzar tasas cercanas a los 2 Gbps en el canal descendente, así como una mayor estabilidad y fiabilidad en las comunicaciones.

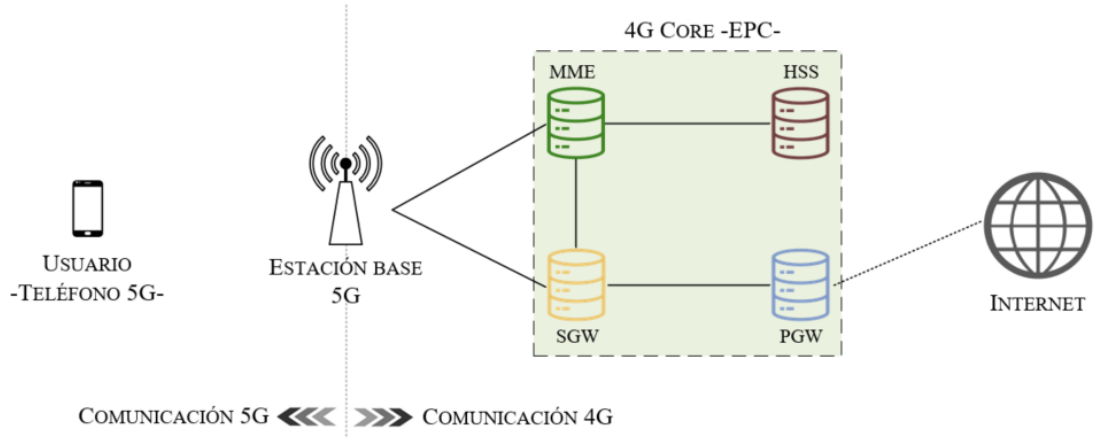


Figura 2.1: Estructura y funcionamiento de una red 5G NSA.

- **5G SA**: Este tipo de despliegue, definido en el Release 16 [21], requiere una evolución completa de la red. Esto implica tanto a la parte radio, adoptando 5GNR, como al *core* de la red, sustituyendo el uso del núcleo de 4G (*Evolved Packet Core* (EPC)) por el *5G Core* (5GC). Además de las capacidades propias del EPC en términos de movilidad, tarificación, seguridad, encaminamiento o control de políticas, el 5GC [22] ofrece nuevas características, tales como:

  - **Control de políticas 5G**: Mejora de la gestión de los recursos y servicios que puede emplear cada dispositivo conectado a la red.
  - **Análisis de datos**: Destinados a aplicaciones de *Artificial Intelligence* (AI) y *Machine Learning* (ML), recopila y analiza la información del núcleo de la red para permitir realizar las operaciones propias de la red 5G de forma más eficiente.
  - **Gestión y selección de *slices* en la red 5G**.

  Desde el punto de vista de las prestaciones, mediante el despliegue 5G SA se incrementa la tasa binaria en *Downlink* (DL) hasta los 20 Gbps y se rebaja la latencia de las comunicaciones hasta 1 ms. Asimismo, se logra mejorar la eficiencia espectral y energética de la red, se consigue un aumento de la capacidad del sistema en cuanto a los usuarios a los que puede dar servicio y una mejora de la capacidad de la red para proveer servicios a usuarios que viajan a velocidades superiores a los 500 km/h. En la Figura 2.2, se presenta una comparativa de los *Key Performance Indicators* (KPI) del 5G (IMT-2020) vs LTE-A (IMT-Advanced).

Con el desarrollo del 5G, se ha conseguido desplegar servicios y funcionalidades con requisitos de red muy variados. Concretamente, en la recomendación ITU-R M.2083 [23] se han definido los 3 escenarios principales que permiten dividir los diferentes servicios en función de los KPI del 5G (véase la Figura 2.2) que demandan. Seguidamente, se explican cada uno de los escenarios, y en la infografía de la Figura 2.3 se presentan los requisitos que precisa cada caso de uso [1, 18, 23]:

- ***Enhanced Mobile Broadband* (eMBB)**: Se corresponde con el escenario que se ha de desplegar en zonas con una elevada densidad de usuarios que precisan velocidades elevadas, una buena cobertura y movilidad. Está centrado en servicios multimedia, como la VR/AR.

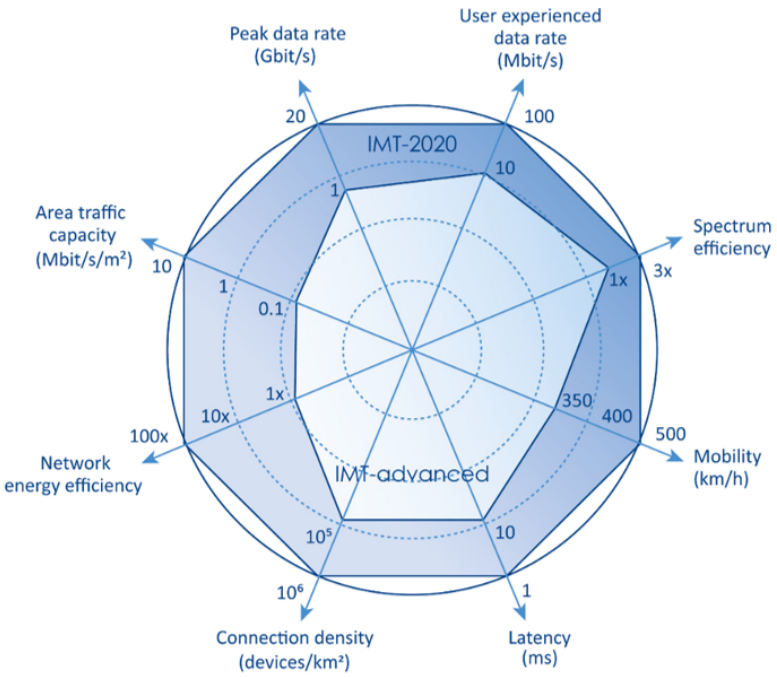


Figura 2.2: Comparativa de los KPI del 5G SA (IMT-2020) y del 4G (IMT-Advanced) [18].

- ***Massive Machine Type Communicactions*** **(mMTC)**: Se trata de un escenario donde se comunican una gran cantidad de dispositivos que se envían pequeños volúmenes de datos. Precisa un bajo coste, una latencia baja y un consumo energético reducido, para garantizar una larga duración de las baterías de los dispositivos. El IoT, las comunicaciones V2X o las *Smart Cities* son algunas de las aplicaciones de este caso de uso.

- ***Ultra-reliable and Low Latency Communications*** **(URLLC)**: Tiene requisitos estrictos en términos de latencia y disponibilidad. Este escenario cubre las necesidades de aplicaciones de seguridad y críticas. Los vehículos autónomos, el control remoto de robots industriales o de tele-cirugía son algunas de los servicios que se enmarcan en este contexto.

Como se ha comentado previamente, estos escenarios se caracterizan por tener requisitos de red muy diferentes en términos de latencia, ancho de banda, movilidad, eficiencia, etc. Mediante el *Network Slicing* se consigue implementar estos casos de uso sobre la misma red física.

El *Network Slicing* permite desplegar múltiples redes lógicas extremo a extremo e independientes sobre la misma infraestructura física. En cada una de esas redes lógicas, denominadas *slices*, se puede adaptar la configuración, adecuándola para implementar un escenario concreto en función de los servicios que vaya a proveer. Por lo tanto, se trata de una solución eficiente y económica para poder desplegar los diferentes escenarios presentados previamente sobre la misma red física [24].

Además, cada *slice* puede abarcar varias partes de la red (terminal, acceso, transporte y troncal) y puede ser desplegado por múltiples operadores, separando a sus clientes en diferentes *slices*. Esta separación de la red física en subredes lógicas virtuales es un proceso totalmente transparente para los usuarios finales [24]. En la Figura 2.4, se presenta la definición de varios *slices* para los escenarios definidos por la ITU-R.

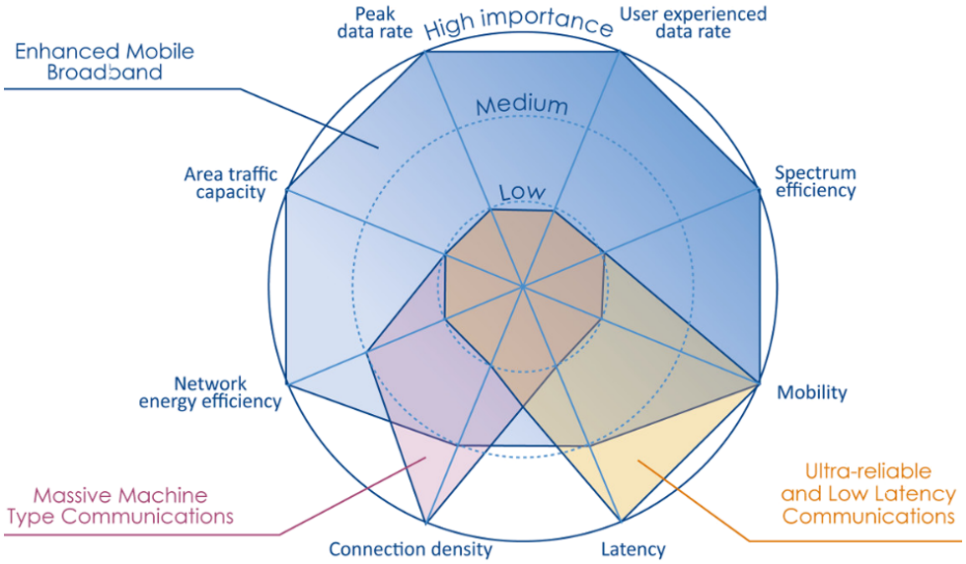


Figura 2.3: Casos de uso y capacidades clave según la ITU-R M.2083 [18].

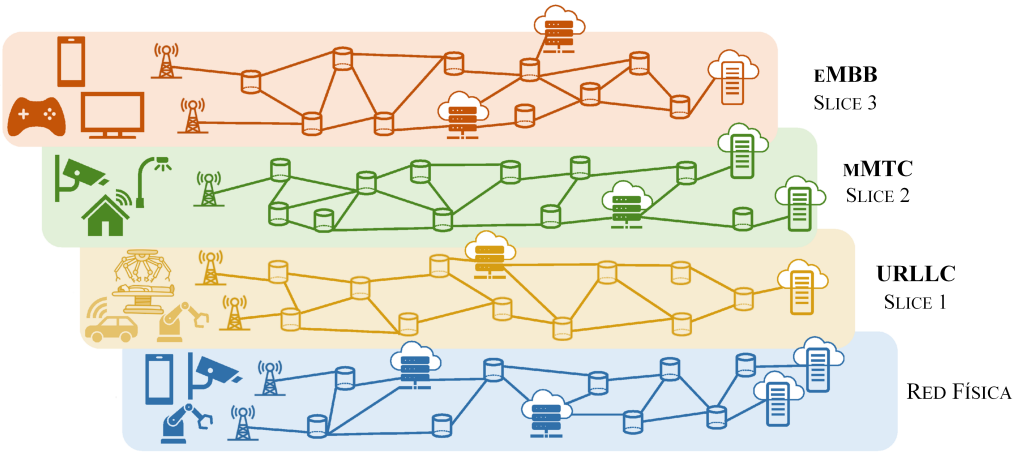


Figura 2.4: *Network Slicing*, despliegue de *slices* para cada escenario de la ITU. Traducción y adaptación de [25].

2.3. Arquitectura Cloud-RAN

Como antecedente a la arquitectura Cloud-RAN, se encuentra la solución *Distributed RAN* (D-RAN) donde todas las funcionalidades de la estación base, o *Base Station* (BS), se realizan en la propia torre. Allí, coexisten dos tipos de elementos [1]:

- RRU: Efectúan las funcionalidades de la capa radio e incluso de la capa física. Están situadas en la parte superior de la torre, junto a las antenas.
- BBU: Localizadas en las proximidades de la torre, se comunican con la RRU típicamente a través de un interfaz *Common Public Radio Interface* (CPRI). Asimismo, en esta arquitectura cada BBU controla una RRU.

D-RAN presenta diversas desventajas en cuestiones de la gestión del espacio y las capacidades. Por ejemplo, gran parte de las BS de las redes celulares se dimensionan para satisfacer periodos de máxima capacidad, garantizando una determinada QoS. Sin embargo, en ciertas ocasiones los recursos de las BS están infrautilizados, y no se aprovecha dicha capacidad. Por otro lado, las BS tampoco están diseñadas para atender las cantidades de tráfico que se generan en determinados eventos (conciertos, partidos de fútbol, etcétera) y la capacidad de la estación no es suficiente para atender todo este tráfico [26].

Cloud-RAN es una tecnología emergente que permite la integración de las características y los beneficios de la Nube en la red 5G. Esta arquitectura proporciona ventajas en términos de latencia, capacidad de la red, gestión óptima de los recursos y eficiencia energética [27]. Además, se trata de una solución flexible que permite adaptar las capacidades de la BS en función de las necesidades.

En C-RAN, se realiza una separación física de los componentes de la BS. Mientras que las RRU permanecen en la torre, las BBU se centralizan en los llamados BBU *pools*. De este modo, se concentran determinadas funciones que anteriormente se realizaban *in situ* en la propia torre, lo que habilita una coordinación más estrecha entre las BBU. Así, se pueden controlar de forma conjunta varias RRU, lográndose una mejor gestión y uso de los recursos, además de una potencial reducción de los costes derivada de la centralización [26].

En esta arquitectura, se distinguen cuatro tipos de elementos [26, 27]. Seguidamente, se presenta una descripción básica de cada uno de ellos y, en la Figura 2.5, se muestra una infografía donde se describe esta topología:

- **BBU *pools***: El *pool* opera como un equipo centralizado, compuesto por múltiples BBU (donde cada una es un nodo virtual), que efectúa el procesado y la gestión de las señales que las RRU envían al *pool* a través del *fronthaul*.

  Algunas tecnologías de virtualización como *Software Defined Network* (SDN) o *Network Function Virtualization* (NFV) permiten efectuar un aislamiento lógico de los recursos del *pool*, mientras que los recursos físicos (almacenamiento, CPU y red) se comparten de forma dinámica y escalable entre las BBU.

- **RRU**: Situadas junto a las antenas, realizan el procesamiento digital, la conversión analógico-digital (y digital-analógico) así como la amplificación y el filtrado de las señales recibidas o enviadas por el canal radio. Al igual que en la arquitectura D-RAN, las RRU realizan las funcionalidades de las capas radio e incluso física.

- **Fronthaul**: Interfaz que comunica el conjunto de las BBU con las RRU. Se caracteriza por tener un elevado ancho de banda y una alta sensibilidad a la latencia, y se encarga del transporte tanto de los datos como la información de gestión y control.

  Aunque se puede implementar utilizando tecnología óptica e inalámbrica, se suele optar por emplear *Fibra Óptica* (FO) debido a que proporciona una mayor velocidad y menor retardo.

- **Backhaul**: Interfaz de comunicaciones que conecta el *pool* de RRU con el *core* de la red 5G.

Aunque la arquitectura C-RAN aporta beneficios, su adopción precisa una red *fronthaul* de gran capacidad y muy baja latencia, lo que a su vez implica una gran inversión en su despliegue. Para poder

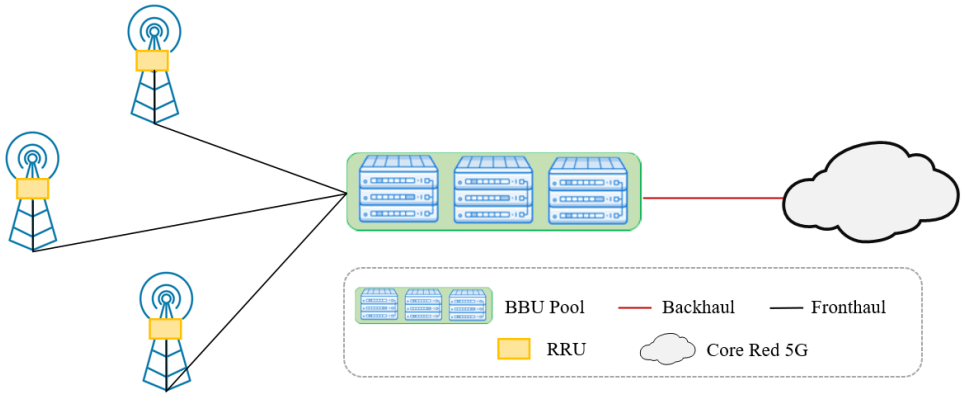


Figura 2.5: Arquitectura C-RAN.

reducir estos requisitos en el enlace de comunicaciones, surge el concepto de *functional split* (definido por el 3GPP en el *Technical Report* 38.801 [28]), según el cual se pueden determinar qué funciones se realizan en la nube de forma centralizada y cuáles no.

Ante este cambio de arquitectura, se producen dos modificaciones principales: la división del BBU en dos elementos (el CU y DU) y el cambio de nomenclatura del RRU, que pasa a ser denominado como RU. A continuación, se describen brevemente cada uno de estos componentes [29], y en la Figura 2.6 se presenta la evolución de la arquitectura C-RAN:

- ***Radio Unit* (RU)**: Situada en las torres de comunicaciones, se encarga de realizar las funcionalidades radio y algunas de la capa física. Sus principales consideraciones de diseño son un tamaño, peso y consumo energéticos reducidos.
- ***Centralized Unit* (CU)**: Unidad centralizada que ejecuta las funcionalidades de las capas superiores y puede controlar una o varias DU. Su comunicación con las DU se realiza a través del *midhaul*[1]. Las funciones concretas que realiza dependen del *split funcional* desplegado.
- ***Distributed Unit* (DU)**: Unidad situada cerca del RU responsable de las funciones de las capas inferiores y parte de la capa física. Está controlada por una CU y, al igual que en el caso del elemento centralizado, sus funciones exactas dependen del *split*.

Además de las entidades de estandarización, existen iniciativas que buscan acelerar la implementación y despliegue de estas nuevas arquitecturas de red. Entre ellas, la *O-RAN Alliance*, junto con el *Small Cell Forum*, es la más destacada con un ámbito internacional. Está compuesta por operadores móviles, proveedores e instituciones académicas e investigadoras cuyo principal objetivo es adaptar las *Radio Access Network* (RAN) para conseguir que sean más inteligentes, abiertas, virtualizadas y completamente interoperables. Sus objetivos y responsabilidades se pueden clasificar en cuatro grandes grupos [30]:

- Elaboración de especificaciones y *white papers* sobre las arquitecturas RAN, sus objetivos, propósitos y planes para cumplirlos. Estos documentos son de uso libre, por lo que cualquiera puede emplearlos para crear soluciones RAN abiertas, innovadoras y competitivas.

[1]En la literatura, en algunos casos, a este interfaz de comunicaciones se le denomina *fronthaul* o *Inter-Cloud Interfaces* (ICI). Es decir, en algunas ocasiones no se realiza distinción entre el enlace de comunicaciones del CU-DU ni DU-RU.

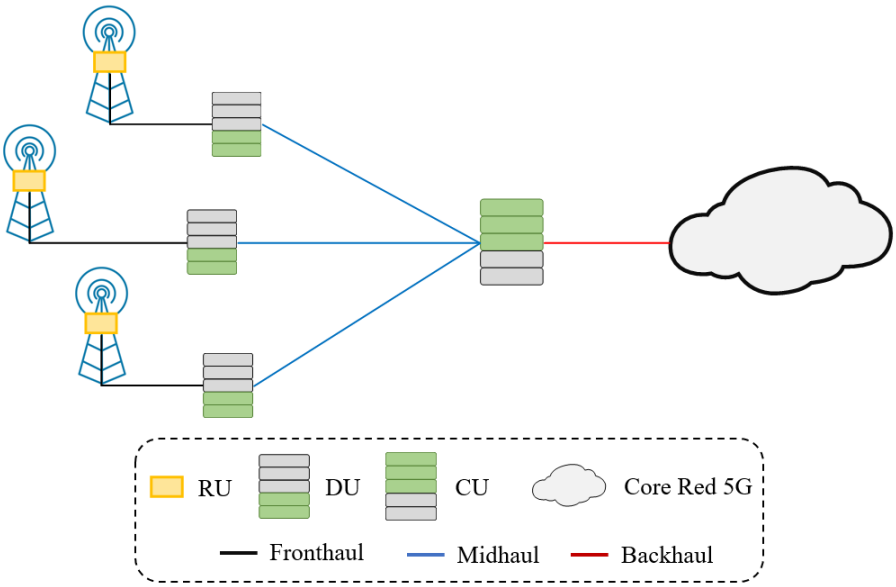


Figura 2.6: Evolución de la arquitectura C-RAN hacia el concepto de *functional split*.

- Implementaciones *software* de código abierto. Estas iniciativas se centran en adecuar la implementación al desarrollo de la arquitectura y las nuevas especificaciones. Su propósito es conseguir unificar y acelerar el desarrollo y despliegue de este tipo de arquitecturas. Esta funcionalidad es realizada por la O-RAN Software Community, que se trata de una asociación de *Oran Alliance* con *Linux Foundation*.
- Para realizar la verificación e integración de las implementaciones, proporciona a sus miembros entornos abiertos y colaborativos para realizar estas tareas.
- Exposición y presentación de las implementaciones y los proyectos RAN realizados por sus miembros.

Dentro de las implementaciones de la O-RAN se incluyen las versiones de CU y DU que está adoptando la industria, y que implementa un *functional split* concreto.

2.4. Functional Split

Como se ha comentado anteriormente, el concepto de *split funcional* permite dividir las funcionalidades propias de una BS entre la *Centralized Unit* y la *Distributed Unit*. De este modo, parte de estas tareas se ejecutan de forma centralizada, mientras que otras se despliegan en las cercanías de las torres de comunicaciones celulares.

Con el propósito de contextualizar y especificar cuáles son las funciones propias de una BS, en la Figura 2.7 se indica la pila de protocolos de una BS LTE[2], así como una breve explicación [31] sobre cada uno de ellos:

- *Physical* (PHY): Se encarga del envío de la información de la capa MAC al aire. Además, es responsable del control de potencia y del establecimiento del enlace.

[2]La pila de protocolos de las estaciones 5G añade el *Service Data Adaption Protocol* (SDAP), pero no se incluye en los posibles *functional-split* definidos por el 3GPP.

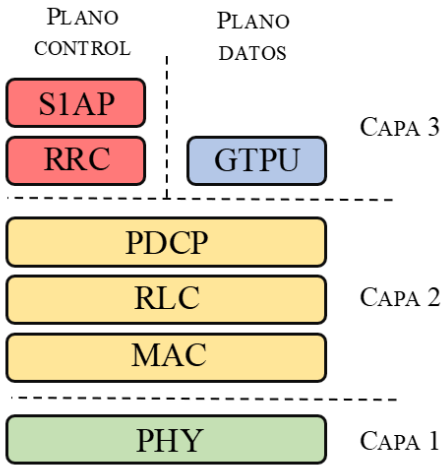


Figura 2.7: Pila de protocolos de una EB LTE. Traducción y adaptación de [31].

- MAC: Multiplexa la información en uno o varios canales lógicos. Además, se encarga de la planificación de las transmisiones *Uplink* (UL) y DL a través de esquemas de retransmisión, corrección de errores y estableciendo prioridades entre los canales lógicos.
- *Radio Link Control* (RLC): Gestiona los canales lógicos o las portadoras de cada uno de los usuarios conectados a la estación. Presenta diferentes modos de trabajo, en función de la existencia de retransmisiones, concatenaciones o segmentaciones de las *Protocol Data Unit* (PDU) enviadas.
- *Packet Data Convergence Packet* (PDCP): Encargado del cifrado de la información (tanto del plano de datos como del de control) y garantizar la integridad del tráfico de control. También realiza otras operaciones, como la compresión de cabeceras, el descarte de tramas duplicadas y la entrega ordenada de las tramas.
- *Radio Resource Control* (RRC): Gestiona la comunicación de control de la BS con los usuarios. Se encarga del establecimiento, mantenimiento y liberación de las conexiones de control que se realizan con cada usuario. Además, controla las funciones de cifrado e integridad.
- *S1-Application Protocol* (S1-AP): Se encarga de gestionar el tráfico del plano de control. En concreto, conecta la EB con el EPC y se encarga de realizar el reenvío de los mensajes de movilidad desde *core* hacia los usuarios conectados a la BS.
- *GPRS Tunnelling Protocol User Plane* (GTP-U): Provee comunicación del plano de datos de los usuarios con el EPC para garantizar la salida (o llegada) de los datos hacia (o desde) Internet u otras redes.

Sobre esta pila de protocolos, el 3GPP define un total de 8 opciones donde se determinan qué funcionalidades (capas o subcapas de la pila de protocolos) se ejecutan en la CU y cuáles en la DU. La descripción y justificación del uso de estas opciones se realiza en detalle en la Sección 2.4.1 del documento.

La existencia de diversas opciones de *split* dotan a la arquitectura de una amplia flexibilidad, que se ve reflejada en aspectos como [28]:

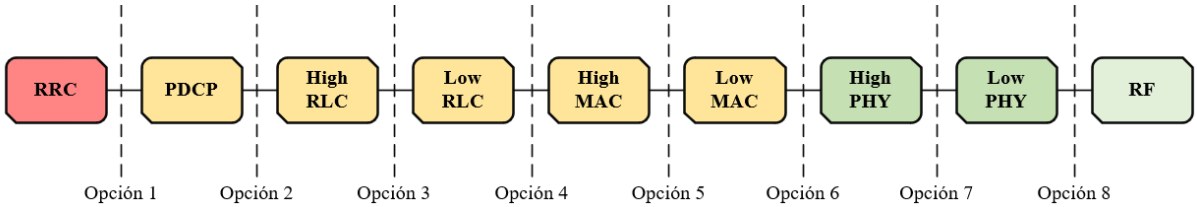


Figura 2.8: Splits funcionales entre CU-DU. Traducción y adaptación de [28].

- Las implementaciones de los elementos *hardware* son más flexibles y permiten desplegar soluciones escalables y rentables.
- La división de las funcionalidades entre la CU y DU facilita la coordinación del rendimiento y la gestión de la carga de tráfico. Además, permite optimizar el rendimiento en tiempo real y el uso de soluciones SDN/NFV en arquitecturas RAN.
- El cambio de *split* en función del caso de uso permite variar los requisitos de latencia y ancho de banda de las comunicaciones.

Otro aspecto interesante sobre el *functional split* es la posibilidad de conmutar el tipo de *split*. Este cambio implica la necesidad de reestructurar la distribución de las funcionalidades y reconfigurar tanto la CU como DU. Esta modificación se puede realizar en tiempo de ejecución, esto es, dinámicamente, sin la necesidad de reestablecer la red, pero implica una mayor complejidad en el *midhaul* [28].

2.4.1. Opciones de Functional split: descripción y justificación

La elección del nivel de *split*, es decir, cómo dividir las funcionalidades entre la CU y DU viene determinada por diversos factores, relacionados con los servicios que se desean proveer en la red así como los diferentes escenarios y limitaciones de la red física que se despliega. Otros aspectos a considerar son [28]:

- Necesidad de soportar diferentes QoS por cada uno de los servicios ofrecidos en la red.
- Diferentes densidades de usuarios y demanda en cada una de las zonas geográficas que cubre la red desplegada. Este factor tiene un impacto a nivel de coordinación de la red de acceso.
- Capacidad de la red para operar con redes de transporte con diferentes niveles de rendimiento en términos de tasa y/o latencia. En este sentido, el *Small Cell Forum* define en [32] cuáles son los requisitos en rendimiento que han de cumplir las diferentes opciones de *split*.

En la Figura 2.8 se presentan las 8 clases de *functional split* detalladas en el *Technical Report* 38.801 [28]:

Opción 1: RRC/PDCP *split*

Mientras que la capa RRC está situada en la CU, el resto de capas de la pila de protocolos se localizan en el DU (PDCP, RLC, MAC y PHY) y RU (*Radio frequency* (RF)). De este modo, todo el plano de datos está situado en la DU, y el RRC está centralizado.

Se trata de una solución adecuada para casos de uso como *Edge Computing* o aplicaciones de baja latencia que requieren que los datos de usuario se encuentren cerca de los puntos de transmisión.

Sin embargo, la separación de RRC y PDCP implica que la securización del interfaz de comunicaciones pueda afectar al rendimiento de esta clase de *split*.

Opción 2: PDCP/RLC *split*

Esta división se caracteriza por tener dos alternativas. En la primera, denominada Opción 2-1, se realiza únicamente la separación del plano de datos. Las capas PDCP y RRC se sitúan en la CU, mientras el resto están en la DU. Esta opción se caracteriza por:

- Permitir centralizar la agregación de tráfico de los puntos de transmisión NR y E-UTRA, y facilitar la gestión del tráfico en ambos nodos de transmisión.
- Es la configuración que establece la conectividad dual de LTE, por lo que se trata del *split* más sencillo de estandarizar.
- Mejor comportamiento frente a la retransmisión de PDUs RLC.
- La capacidad de trabajo de LTE-NR y el *split* es beneficiosa, al menos, en el plano de datos.

En la segunda variante, *Opción 2-2*, se separan las capas PDCP y RRC en dos entidades centralizadas. De este modo, se divide el plano de datos del plano de control en la CU. Al igual que la opción anterior, permite la agregación centralizada de tráfico y su gestión desde NR y E-UTRA. Además, esta opción permite la separación del plano de datos mientras se dispone de RRC centralizado.

No obstante, en este *split* es necesario garantizar la coordinación de las configuraciones de seguridad de las diferentes instancias de PDCP.

Opción 3: High RLC/Low RLC *split*

La Opción 3 se caracteriza por tener 2 subtipos basados en la separación de funcionalidades de tiempo real y funciones que no operan en tiempo real.

La configuración 3-1 se basa en la división de RLC en dos subcapas donde la superior, situada en la CU, se encarga de las funciones de segmentación, y la inferior, localizada en la DU, de *Automatic Repeat Request* (ARQ) y otras funciones de RLC.

Aunque se trata de una solución sensible a la latencia existente en el *fronthaul*, la justificación para su implementación se resume en las siguientes ventajas:

- Proporciona un mejor control de flujo en el *split*.
- La centralización de ARQ permite securizar las comunicaciones de datos y señalización. Además, aporta robustez frente a escenarios con condiciones de transporte adversas, gracias a la centralización de ARQ y la ordenación de los segmentos.
- Las DU pueden controlar un mayor número de usuarios, ya que no almacenan información sobre el estado RLC y tampoco necesitan el contexto de cada usuario para su operación.

- Permite la agregación de tráfico centralizado y la gestión de carga desde NR y E-UTRA.
- Reducción de los requisitos de procesado y de almacenamiento en los *buffer* de las unidades distribuidas, como consecuencia de la centralización de ARQ.

Al igual que la Opción 3-1, la configuración 3-2 divide la capa RLC en dos partes. En este caso, la inferior se encarga de la transmisión del enlace *Downlink* y de la recepción del enlace *Uplink* mientras que la superior se encarga de la transmisión UL y la recepción DL. La parte baja se sitúa en la DU y la superior en la CU, garantizándose una entrega en tiempo real al PDCP de los datos que recibe RLC.

Esta opción no es sensible a la latencia de la comunicación CU-DU ni tampoco genera ningún tipo de restricción adicional en el *fronthaul*. Al igual que las opciones previas, permite realizar un control de la carga de tráfico y una agregación centralizada desde los puntos de transmisión E-UTRA y NR. Por otro lado, el control de flujo se realiza en la CU mediante el uso de 2 *buffers*: uno en la DU para el tráfico DL, y otro en la CU, para el UL.

Sin embargo, en este *split* surgen retardos adicionales en el envío de PDUs de estado RLC como consecuencia de que, a diferencia de otras divisiones donde no se separa RLC, es necesario que se envíe la información de estado desde la CU hacia la DU antes de poder enviarla por el interfaz radio.

Opción 4: RLC/MAC *split*

En esta opción, las capas RRC, PDCP y RLC se encuentran centralizadas mientras que las capas MAC y PHY están en la DU.

En el contexto de la pila de protocolos LTE, no se espera un beneficio al utilizar esta configuración. Sin embargo, se encuentra en revisión para la pila de protocolos de NR.

Opción 5: intra MAC *split*

El *split* intra-MAC se caracteriza porque las capas PDCP, RRC, RLC y la parte alta de MAC están en la CU. La parte baja de MAC, así como la capa física, se sitúan en la unidad distribuida. Al separar la capa MAC se consigue que las funcionalidades y servicios MAC puedan estar en la CU, DU o ambas.

La parte alta de MAC se encargará de la gestión de las múltiples subcapas inferiores MAC, siendo la entidad responsable de la toma de decisiones del *scheduling*. Además, permitirá la adopción de técnicas como *Coordinated Multi-Point transmission and reception* (CoMP) para el tratamiento de interferencias intercelulares.

En cuanto a la parte baja de MAC, existen funcionalidades sensibles al retardo (como *Hybrid Automatic Repeat Request* (HARQ)) o cuyo rendimiento es proporcional a la latencia (por ejemplo, el control de acceso aleatorio). Además, sus funcionalidades radio ejecutan el procesado de la información de *scheduling*. Entre las principales características de este *split* destacan la reducción de los requisitos de latencia y tasa binaria en el *fronthaul*, la agregación centralizada y la gestión del tráfico en E-UTRA y NR y, por último, la gestión eficiente de las interferencias entre múltiples células y la utilización de técnicas de *scheduling* mejoradas.

Sin embargo, esta alternativa añade complejidad al interfaz de comunicaciones del CU-DU. Se complica la definición de operaciones de *scheduling* y la toma de decisiones está sujeto a las latencias del *fronthaul*.

Opción 6: MAC/PHY *split*

Mientras la capa MAC y las capas superiores están centralizadas, la capa física y la capa radio se localizan en el DU y RU, respectivamente. El interfaz de comunicaciones del CU-DU envía información relacionada con los datos de usuario, configuración, *scheduling* (asignación de bloques de recursos, configuración de las antenas, etcétera) y medidas. Esta opción se caracteriza por tener las siguientes ventajas:

- Permite la agregación centralizada de tráfico desde los puntos de transmisión de NR y E-UTRA. Además, puede facilitar la gestión de la carga de tráfico entre NR y E-UTRA.
- Reducción de los requisitos del *fronthaul* en términos de tasa binaria, ya que los datos útiles se corresponden con bits de bloques de transporte.
- Como la capa MAC está centralizada, tanto la transmisión conjunta como el *scheduling* centralizado son soluciones posibles.

Sin embargo, este *split* tiene requisitos temporales a nivel de subtrama muy exigentes en la comunicación entre MAC y PHY, por lo que el *Round Trip Time* (RTT) del *fronthaul* puede ocasionar problemas de sincronización y *scheduling* del HARQ.

Opción 7: intra PHY *split*

Este *split* se caracteriza por disponer la parte superior de la capa física, junto a las capas superiores de la pila de protocolos, en la unidad centralizada, mientras que la parte baja de PHY reside en la DU. Ofrece la posibilidad de desplegar hasta 3 configuraciones (7-1, 7-2 y 7-3), así como elegir opciones diferentes para el enlace ascendente y el DL. Un uso adecuado de las opciones permitirá la reducción del ancho de banda necesario entre la CU y DU.

La centralización de la capa MAC permite realizar tanto un *scheduling* centralizado como un procesado conjunto (tanto en transmisión como en recepción) de la información. Además, permite la agregación de tráfico y la gestión de la carga desde NR y E-UTRA. Por último, reduce los requisitos del *fronthaul* en ancho de banda. Sin embargo, este *split* requiere una baja latencia en las comunicaciones entre las subcapas PHY.

En la primera configuración (Opción 7-1) se propone que la DU realice las funciones de filtrado del *Physical Random Access Channel* (PRACH). Gracias a esta variación, se permite implementar receptores más avanzados, mejorándose las comunicaciones. La Opción 7-2 añade varias funcionalidades en la DU, tanto en DL (funciones de prefiltrado y la deslocalización de recursos) como en UL (funciones de precodificación y localización de recursos), que permiten también el uso de receptores más avanzados. Finalmente, la tercera opción (cuyo uso es exclusivo para *Downlink*) establece que únicamente la codificación de la capa física está centralizada, mientras que el resto de funciones de la capa física se realizan de forma distribuida. Esta subdivisión permite reducir la tasa binaria demandada en el *fronthaul*.

Opción 8: PHY/RF *split*

En este tipo de *split* se centralizan todas las capas de la pila de protocolos con excepción de la capa RF, que se ejecuta en la RU. En esta configuración se consigue una coordinación muy estrecha de la RAN, y puede dar soporte eficiente de funcionalidades como la movilidad, el balanceo de carga de tráfico,

Multiple-Input Multiple-Output (MIMO) o la transmisión/recepción multipunto coordinada. Entre sus beneficios, destacan:

- Agregación centralizada de tráfico en E-UTRA y NR y mayor facilidad de gestión del tráfico entre los puntos de transmisión de la red.
- La elevada coordinación y centralización permite tener mayor rendimiento y una gestión de los recursos radio más eficiente.
- La separación de RF y PHY permite aislar completamente los componentes de RF de las actualizaciones de la capa física, consiguiéndose mejorar la escalabilidad del *split*. Además, permite el reuso de los componentes RF para las capas físicas de varias tecnologías, como 3G o LTE, y la agrupación de los recursos PHY, lográndose un dimensionamiento más rentable.
- Se consigue una reducción de costes, tanto del sistema como del emplazamiento, como consecuencia de la compartición de recursos RF entre operadoras.

A pesar de sus ventajas, este *split* se caracteriza por tener unos requisitos muy estrictos en términos de ancho de banda y latencia en el *fronthaul*, que pueden ocasionar un mayor consumo de los recursos, un aumento de los costes y limitaciones respecto a la topología de red que se puede desplegar.

2.4.2. Requisitos del Fronthaul en cada *split*: latencia y *throughput*

El *Small Cell Forum* define los requisitos [32] que ha de cumplir el enlace de comunicaciones que conecta la CU y DU para garantizar el correcto funcionamiento de cada *split*, sin limitar la calidad de experiencia de los usuarios. Estos requisitos se dividen en dos grupos: latencia o *delay* y ancho de banda.

Respecto a la latencia, se definen 4 límites para el *delay* existente en un único sentido de las comunicaciones (es decir, la mitad del RTT): Ideal (250 μs), Near-Ideal (2 ms), Sub-Ideal (6 ms) y Non-Ideal (30 ms).

Además, en cada *split* se aporta una explicación del potencial impacto en caso de superarse alguno de estos valores. Esta argumentación se clasifica en tres grupos: 'N/A' (con esa latencia, el *split* es inviable), Insignificante, en inglés, *'Negligible'* (compatibilidad con este *delay*) o una descripción concreta del impacto y la degradación en rendimiento que supone.

En cuanto al *throughput*, se presentan las capacidades mínimas en UL y DL necesarias para el correcto funcionamiento del sistema tanto en el plano de datos como en el de control. Asimismo, se justifica tanto la tasa bruta demandada como la real, donde se incluye la sobrecarga necesaria para la operación de los protocolos.

Seguidamente, se sintetizan los requisitos en cada caso, así como el impacto y la degradación que ocasiona el incumplimiento de los límites definidos.

En la Opción 1, PDCP/RRC *split*, se indica que únicamente el *split* se ve degradado con latencias no ideales (30 ms) como consecuencia de la lentitud de los procesos de RRC. Respecto a la tasa binaria, se necesita una velocidad total de 187.5 Mbps en descarga y 62.5 Mbps en subida. En ambos casos, aproximadamente 10 Mbps se destinará al tráfico de control, mientras el resto se corresponde con datos.

En el caso del *split* PDCP/RLC, el límite de latencia de 30 ms (*No-Ideal*) supone una limitación en la calidad de experiencia del usuario en servicios como los juegos en línea. En cuanto a las necesidades de velocidad, se requieren las mismas tasas en bajada y subida que en la Opción 1.

El *split* RLC/MAC depende del esquema de control de flujo implementado. En caso de que no haya control de flujo, el interfaz ha de tener una latencia ideal (250 μs) mientras que si hay control de flujo la situación es más laxa. En este caso, el *split* es capaz de operar con retardos hasta 30 ms y, en cuanto a la tasa, continúa teniendo los mismos requisitos que las opciones anteriores.

El *split* MAC se puede implementar en escenarios que se caractericen por tener peores condiciones en términos de latencia. Para ello, es necesario que el *scheduler* MAC sea capaz de anticiparse en el envío de la información de planificación antes de que esta sea estrictamente necesaria.

En este caso, para cada uno de los niveles de latencia establecidos, se define tanto el parámetro *HARQ report turnaround*[3] como el número máximo de retransmisiones HARQ que se pueden realizar por cada notificación de *scheduling*. Como se puede observar en la Tabla 2.1, a medida que aumenta la latencia, también se incrementa el tiempo de notificación HARQ y el número de retransmisiones.

Tabla 2.1: Requisitos temporales del *split* MAC

Tipo de latencia	HARQ report turnaround	Número de retransmisiones HARQ
Ideal	8 ms	0
Near-Ideal	12 ms	1
Sub-Ideal	20 ms	2
Non-Ideal	68 ms	8

Sin embargo, este *split* mantiene los mismos requisitos de tasa que los *splits* previos: 187.5 Mbps en DL y 62.5 Mbps en subida.

Al igual que en el *split* MAC, el PHY/MAC también define un tiempo máximo y retransmisiones máximas HARQ. Además, se especifica que este escenario concreto es incapaz de operar frentes a latencias no ideales (30 ms) y para los niveles se establecen los mismos límites que en el caso del *split* anterior (véase la Tabla 2.1). Por otro lado, los requisitos de tasa permanecen invariables respecto a los *splits* presentados con anterioridad.

Por último, el *split* PHY tiene los mismos requisitos temporales que el *split* MAC, siendo incapaz de operar en escenarios con latencias no ideales y definiendo los mismos límites de HARQ para el resto de casos de *delay*.

Sin embargo, respecto a la tasa binaria el *fronthaul* demanda unas características más estrictas. Además, cuantas más funciones se ejecuten en la CU, se necesitará una mayor velocidad. Concretamente, en función de cuántas tareas realice la CU, la tasa varia desde los 173 Mbps hasta los 2457.6 Mbps en descarga y desde los 452 Mbps hasta los 2457 Mbps en subida.

[3] Se corresponde con el tiempo máximo que transcurre desde que la DU recibe un mensaje de *scheduling* hasta que genera la respuesta oportuna y comienza su transmisión por el *fronthaul*.

Capítulo 3

Instalación y herramientas utilizadas

En este capítulo se realiza una descripción de las herramientas que se han utilizado a lo largo del proyecto, presentando sus principales características y, en el caso de que existan otras alternativas, justificando los motivos de su elección.

3.1. srsRAN

srsRAN se trata de un desarrollo *software* de código abierto y libre, que permite desplegar redes 4G y 5GNR. En este trabajo se ha usado la implementación 4G, ya que cuando se inició el desarrollo la implementación 5G no estaba disponible. Concretamente, se ha empleado la versión 21.04 [33] de *srsRAN*, que permite el despliegue de 3 elementos de las redes celulares:

- *srsEPC*: Implementación ligera del *core* de la red, donde se incluyen elementos como el *Mobility Management Entity* (MME), *Home Subscriber Service* (HSS) y el *S/Packet Gateway* (S/P-GW). El MME se responsabiliza de la movilidad y el control, mientras que el HSS almacena información del usuario e interviene en la autenticación y autorización del usuario. Por último, el S/P-GW permite la comunicación de los usuarios con redes externas a través de Internet.
- *srsENB*: Implementación completa del *stack* de una BS LTE, junto con las funcionalidades de 5G NSA.
- *srsUE*: Al igual que en el caso de *srsENB*, implementa la pila de protocolos completa de un dispositivo de usuario LTE.

La implementación *srsRAN* la ha realizado la empresa *Software Radio Systems*, que hace un uso comercial de ella. Esta solución está implementada en C++ y C, y permite desplegar redes LTE y 5G haciendo uso de *hardware* comercial para implementar el interfaz radio.

En este trabajo se ha modificado la implementación de la *Base Station* (*srsENB*) para realizar la división de sus funcionalidades entre CU y DU. Esta implementación se ha conseguido mediante el uso de otras herramientas y desarrollos, tales como *zeroMQ* o *Protocol Buffers*, que se detallan en las siguientes secciones.

Una vez realizado el desarrollo, se han utilizado otras herramientas como *top*, *iperf* y *traffic Control* para la evaluación del correcto funcionamiento de la implementación, así como un estudio en profundidad de la carga computacional consumida en cada escenario desarrollado.

3.2. zeroMQ

ZeroMQ[1] se trata de una librería de uso libre que proporciona un entorno asíncrono y de alto rendimiento para el envío atómico de mensajes[2] a través de *sockets*. Además, respecto a otras soluciones, no está orientada al mensaje, y no requiere de un intermediario (denominado *broker*) para su funcionamiento.

Además, se caracteriza por su flexibilidad, ya que soporta múltiples patrones de comunicación (entre otros, *Request-Response*, *Publish-Subscribe* o *Client-Server*[3]) y es capaz de operar con múltiples protocolos como *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP) o *WebSockets*. Otros aspectos significativos de esta librería es que no introduce latencia en las comunicaciones, y su notable escalabilidad.

Concretamente, en la implementación se ha escogido la configuración *Request-Response* frente a otras soluciones. El patrón petición-respuesta está diseñado para operar en arquitecturas orientadas al servicio y dispone de dos modos de funcionamiento: *sockets* síncronos y asíncronos. Además, permite el envío de información bidireccionalmente, característica que otros esquemas, como *Publish-Subscribe*, no permiten.

En este patrón de operación, se distinguen 4 roles: *REQ* y *REP* (ambos síncronos) y *DEALER* y *ROUTER* (ambos asíncronos) que se pueden combinar de múltiples formas, generando *sockets* síncronos, asíncronos o explotar las características de ambos, entremezclando diferentes tipos. Seguidamente se explican brevemente los 4 tipos de *sockets*, sus principales características y diferencias [34].

REQ

Este rol se corresponde con un tipo de *socket* empleado por un cliente para enviar solicitudes y recibir respuestas de un determinado servicio, donde se realiza una secuencia alternada entre el envío y, posteriormente, la recepción de mensajes. Su uso se puede emplear tanto con el rol *REQ* como *ROUTER*.

Se caracteriza porque cada petición enviada se planifica empleando *Round-Robin* y cada respuesta recibida se corresponde con la petición previamente enviada. Además, en el caso de que el otro extremo no responda, su estado permanecerá bloqueado hasta que el evento pertinente se produzca. Por último, en este tipo de rol no se produce el descarte de ningún mensaje.

REP

Esta configuración se emplea por un servicio para la recepción de peticiones de clientes y el envío de respuestas. Este rol también sigue la filosofía alterna, pero la secuencia es al revés: en primer lugar, se recibe para, posteriormente, enviar un mensaje. Este rol se utiliza tanto con pares *REQ* como *DEALER*.

Las peticiones recibidas se encolan siguiendo una planificación *Fair-Robin* y cada respuesta se envía

[1] Página web oficial: `https://zeromq.org/`

[2] La atomicidad es la propiedad que garantiza que una operación (en este caso, el envío del mensaje a través del socket) se finaliza correctamente. En el hipotético caso de que se produzca algún percance que genere un envío erróneo o parcial, se descartaría la operación completa.

[3] El modelo cliente-servidor se encuentra todavía en construcción y su funcionamiento no está garantizado, de momento, en todas las versiones de *zeroMQ*.

al cliente pertinente. En el caso de que un cliente, al cual se está respondiendo, dejara de existir, se descartaría dicha respuesta.

DEALER socket

Se comunica con otros roles (*ROUTER*, *REP* o *DEALER*) enviando y recibiendo (sin ningún tipo de restricción ni secuencia predefinida) mensajes mediante algoritmos *Round-Robin*. Se caracteriza por su fiabilidad, ya que no se producen pérdidas de mensajes.

Además, en el caso de que no responda el otro extremo del *socket*, se bloqueará cualquier operación de envío del *DEALER*. Sin embargo, cuando este rol se comunica con un *REP*, en los mensajes enviados se ha de enviar una parte inicial del mensaje vacía, lo que genera una ineficiencia de la comunicación respecto a otras soluciones.

ROUTER

Se corresponde con el recambio asíncrono del rol *REP*, y se suele emplear junto con el tipo *DEALER*. Emplea un direccionamiento explícito para lograr que cada mensaje se envíe a una conexión específica. De este modo, cuando se recibe un mensaje, el *ROUTER* le añadirá un identificador que indicará qué par ha originado el mensaje, introduciéndose cierta sobrecarga. Por otro lado, cuando se envía un mensaje, se elimina este identificador y se utiliza para determinar a quién se ha de destinar.

En la implementación realizada en el proyecto, se ha decidido utilizar el esquema *Request-Response*, con roles *REQ* y *REP*, para los extremos de la comunicación entre la CU y DU. De este modo, se consigue tener un control ordenado de los mensajes que se envían los extremos, y se garantiza además que no se produzca el descarte de ninguno de los mensajes independientemente de que haya algún percance entre los extremos. Asimismo, se consigue enviar la máxima cantidad de información en cada mensaje, sin la necesidad de introducir sobrecarga o no emplear parte del mismo.

Aunque se podría haber optado por una implementación basada en *sockets* básicos (TCP o UDP) se ha optado por *zeroMQ* por dos razones. La primera es la facilidad de uso en comparación con los *sockets* tradicionales, ya que evita tener que gestionar las recepciones (p.e. ante fragmentación de paquetes) y mantiene persistencia en las conexiones (p.e. si un extremo no está disponible se mantiene a la espera). Además, esta solución también se ha adoptado por *srsRAN* para implementar un emulador de canal radio entre el *srsENB* y el *srsUE*, por lo que ya está integrado.

3.3. Protocol Buffers

Protocol Buffers[4] [35], desarrollado por *Google*, proporciona un mecanismo extensible, compacto e independiente de la plataforma y el lenguaje para la serialización de datos estructurados con compatibilidad hacia adelante y hacia atrás, es decir, con capacidad tanto para serializar como deserializar la información.

Respecto al lenguaje que utiliza para la definición es similar a *JavaScript Object Notation* (JSON) pero es más rápido, ligero y capaz de generar *bindings* (vinculaciones) en un lenguaje nativo tales como Python, C++, etc.

[4] Aunque el nombre completo es Protocol Buffers, es común en la literatura el uso de sus abreviaturas *protobuffer*, *protobuf* o simplemente *proto*.

Se trata de una solución idónea para paquetes de datos estructurados que se enviarán como tráfico de red efímero, o para el almacenamiento de información. Asimismo, mantiene la compatibilidad con versiones anteriores, permitiendo ampliar, modificar o eliminar los campos existentes sin deshabilitar el servicio o actualizarlo.

Los mensajes de *Protobuffer* se definen en archivos con extensión *.proto* que posteriormente se compilan (el compilador se llama *protoc compiler*) para generar código en varios lenguajes de programación. Este código, que representa el contenido de los ficheros *.proto* en el lenguaje elegido, se integra en las implementaciones para la serialización/deserialización de las estructuras de datos.

En este caso, se ha empleado el *Protobuffer* 3.20.1-rc1 [36] para la serialización de instancias de clases y otros elementos complejos que se requieren enviar a través de los *sockets* establecidos mediante *zeroMQ*. Como la plataforma *srsRAN* está diseñada en C/C++, las clases *proto* que se han generado están programadas en C++. Seguidamente, en las Figuras 3.1 y 3.2 se muestra, de forma sintetizada, tanto el proceso de generación de las clases *.proto* como la integración del uso de estas clases en el desarrollo que se ha llevado a cabo.

La principal razón para adoptar *Protobuffer* es que permite realizar la serialización de las estructuras de forma sistemática. El lenguaje C++ no posee de forma nativa herramientas de serialización. Tradicionalmente, la conversión de estructuras de datos en listas de bytes para su envío se realiza de forma manual, sin convenciones definidas (p.e. nombres de métodos). Aunque *Protobuffer* no evita tener que tratar las estructuras de datos de forma manual, proporciona un interfaz consistente para serializar los diferentes tipos de datos (p.e. vectores, mapas, etc.). Por otro lado, *Protobuffer* también puede optimizar el tamaño de los datos enviados.

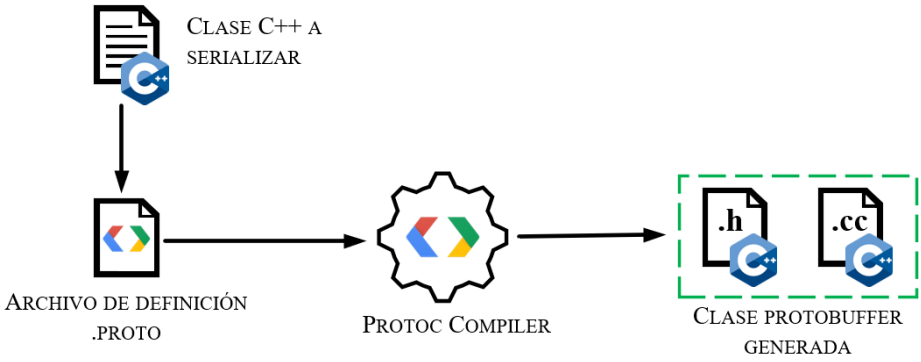


Figura 3.1: Proceso de generación de las clases *.proto*.

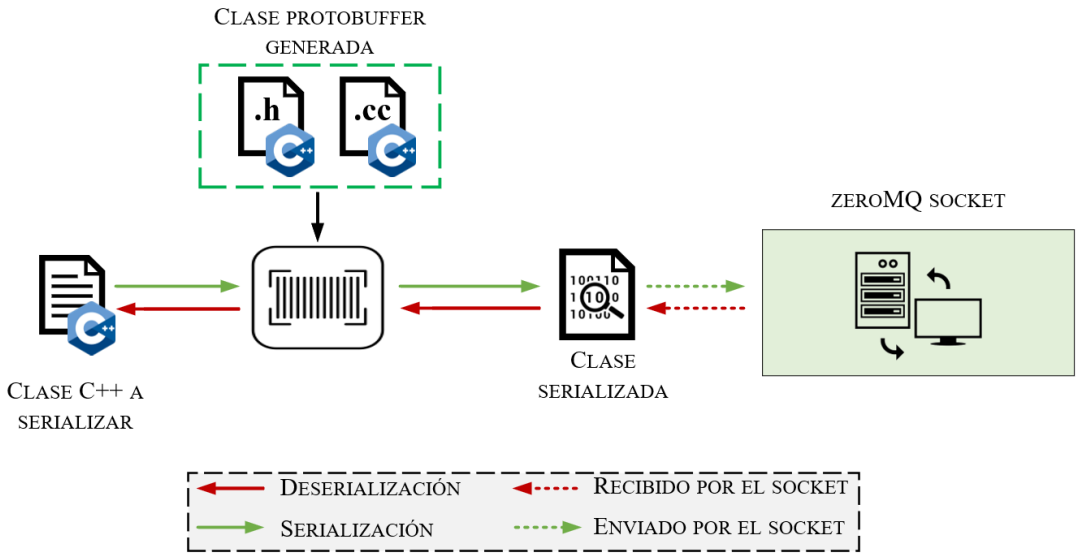


Figura 3.2: Integración y empleo de las clases *.proto* en el desarrollo.

3.4. Iperf

Iperf[5] se trata de una herramienta que permite generar flujos de datos de tráfico sintético[6] con el propósito de evaluar el comportamiento, la estabilidad y fiabilidad de una determinada red de comunicaciones IP. Trabaja con un modelo de cliente-servidor y es capaz de generar tráfico TCP, UDP o *Stream Transmission Control Protocol* (STCP). Permite medir el ancho de banda existente en la comunicación, así como la cantidad de paquetes perdidos, el *delay* y el *jitter*. Además, se trata de una aplicación sencilla y fácilmente configurable mediante el uso de *flags* u opciones que permiten determinar aspectos como [37]:

- La velocidad de la fuente, es decir, la tasa binaria con la que *iperf* genera los datos que el cliente enviará al servidor a través del canal de comunicaciones. La *flag* a utilizar es *-b <bits/segundo>*.
- La duración de cada prueba, es decir, el tiempo durante el cual la fuente genera datos a la tasa establecida. Por defecto, este tiempo está establecido a 10 segundos. Para ello, se utiliza la flag: *-t <segundos>*.
- Como alternativa a la duración de la conexión, también se puede establecer la cantidad de datos que se desea transmitir mediante la flag *-l <número_bytes>*.
- El tiempo entre muestras, es decir, la frecuencia con la que *iperf* reportará información respecto al *Bandwidth* (BW), *delays*, pérdidas, etcétera. Para ello, se emplea la bandera *-i <segundos>*.
- Selección de la versión de IP empleada (*-4* para IPv4 o *-6* para IPv6) y del protocolo de transporte: TCP, UDP (*-u*) o STCP (*–stcp*). Por defecto, se emplea TCP sobre IPv4.

Esta herramienta se ha utilizado junto a *top* en la fase de evaluación de la implementación realizada donde se ha establecido un cliente-servidor entre el *User Equipment* (UE) y el EPC, a través de la BS, y se ha enviado tráfico UDP a diferentes tasas, para el estudio de cada *split* frente a diferentes tasas.

[5]Página web oficial de *iperf*: `https://iperf.fr/`

[6]El tráfico sintético hace referencia a tráfico irreal, es decir, que los datos que envía no se corresponden con ninguna aplicación y son pseudoaleatorios.

3.5. Herramienta top Linux

Se trata de una herramienta[7] que permite realizar una monitorización dinámica en tiempo real de los procesos y/o *threads* que se están ejecutando en un determinado sistema.

Para cada uno de los procesos proporciona una gran cantidad de información, donde destacan los siguientes campos:

- *%CPU*: Porcentaje de CPU consumido en la última medida realizada por el proceso evaluado.
- *%MEM*: Porcentaje de memoria física utilizada por el proceso desde la última monitorización.
- *PID*: Se corresponde con el identificador del proceso, Process ID (PID). Permite realizar un filtrado para monitorizar únicamente aquellos que resultan interesantes en la operación.
- *S*: Aporta información sobre el estado actual de dicha tarea: inactiva, dormida indefinidamente, en ejecución, pausada, etcétera.
- *TIME*: Especifica el tiempo que dicho proceso lleva consumiendo CPU.
- *COMMAND*: Indica el nombre del comando que ha generado dicho proceso. Se trata de un campo útil para realizar una búsqueda visual de aquellos procesos de interés.

Una de las principales características de esta herramienta es su sencillez y capacidad para realizar modificaciones en la monitorización mediante el uso de las *flags* u opciones. De este modo, se puede determinar la frecuencia de medidas[8] con una precisión de décimas de segundo: *-d <segundos.décimas>*. Por otro lado, se puede realizar un filtrado de los procesos monitorizados mediante su identificador de proceso, el PID, usando la opción *-p <PID1,PID2,...>*.

En este trabajo se ha utilizado esta herramienta en el proceso de evaluación de cada uno de los *splits* implementados, con el propósito de analizar el consumo de CPU y memoria por cada uno de los agentes propios de la red implementada: EPC, *eNodeB* (eNB), CU y DU en función de cada escenario estudiado.

3.6. Traffic Control

Traffic Control (TC)[9] es una herramienta de *Linux* que permite configurar el *scheduler* del *kernel* del *Sistema Operativo* (SO) así como gestionar y manipular la transmisión y recepción de paquetes sobre los interfaces de red. Sus funcionalidades se pueden resumir en cuatro puntos:

- *Conformado de tráfico*: Permite controlar la velocidad de transmisión de paquetes y suavizar las ráfagas de tráfico, lográndose mejorar el comportamiento de la red.
- *Scheduling*: Permite mejorar el comportamiento con tráfico interactivo reservando parte de la capacidad para el tráfico a ráfagas.

[7] Manual de *Linux* de la herramienta *top*: `https://man7.org/linux/man-pages/man1/top.1.html`

[8] Por defecto, está fijada una frecuencia de medidas de 3 segundos. Sin embargo, mediante esta funcionalidad se puede realizar un análisis más detallado y exhaustivo.

[9] Manual de TC: `https://man7.org/linux/man-pages/man8/tc.8.html`

- *Establecimiento de políticas*: Encargado del control, establecimiento de prioridades o descarte del tráfico de entrada en función de ciertos parámetros.
- *Descarte de paquetes*: Encargado de eliminar el exceso de tráfico que se produce cuando se supera un determinado umbral de ancho de banda.

Para la ejecución de estas funciones en TC se definen tres tipos de elementos: *qdiscs*, *clases* y *filtros*. Aunque se puede establecer una jerarquía más compleja, por lo general los *qdiscs* están compuestos por *clases* y estas a su vez por *filtros*.

Un *qdisc*, abreviatura de *queuing disciplines*, se corresponde con un *scheduler*. Es el principal bloque del TC. Generalmente, cada interfaz de red tiene dos: uno para el tráfico de entrada (*ingress qdisc*) y otro para el tráfico de salida (*egress* o *root qdisc*). Para poder definir *filtros* en un *qdisc* es obligatorio el establecimiento de *clases*.

Respecto a las clases, la flexibilidad de este elemento permite aumentar la complejidad mediante la generación de clases hijas o el establecimiento de *qdiscs* dentro de la propia clase establecida. Además, cada clase puede disponer de uno o múltiples filtros para el tratamiento del tráfico. Un *filtro* es el elemento más complejo pero, a su vez, el más útil, ya que permite establecer prioridades, políticas o generar limitaciones al tráfico en términos de latencia y/o tasa.

En la Figura 3.3, se presenta un pequeño ejemplo gráfico de implementación de TC sobre un interfaz. Respecto al *qdisc* del tráfico de entrada, se ha establecido una clase compuesta por 9 filtros. En cuanto al tráfico de salida, se ha generado un *qdisc* sin clases y, por tanto, sin filtros.

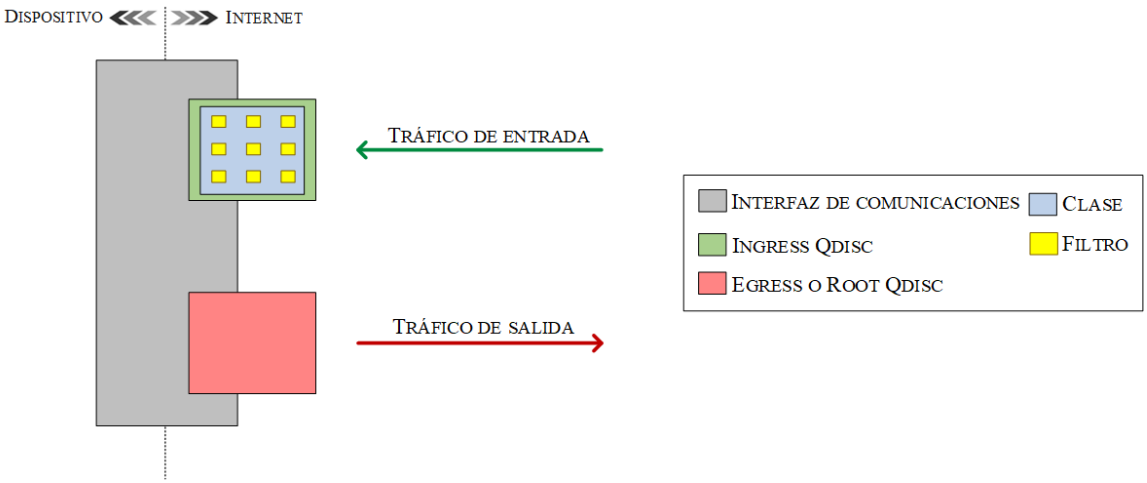


Figura 3.3: Establecimiento de *qdiscs*, clases y filtros en un interfaz de comunicaciones.

En este trabajo, se ha utilizado esta herramienta con un propósito más genérico. Concretamente, se han establecido en el *fronthaul* una serie de filtros que permiten modificar la latencia existente en este interfaz, con el objetivo de estudiar cómo es el comportamiento de la red y sus diferentes *splits* cuando se superan los requisitos temporales establecidos por el *Small Cell Forum* en [32], presentados en la Sección 2.4.2.

Capítulo 4

Desarrollo e implementación

En este capítulo se detalla el desarrollo realizado sobre la pila de protocolos de la BS LTE (Figura 2.7) de la plataforma *srsRAN*. Sobre ella se ha modificado la forma en la que sus capas se comunican entre sí, permitiendo realizar la separación de las funcionalidades de la BS y el establecimiento de *splits funcionales*. En primer lugar, se aporta una visión global sobre cómo se estructuraba la implementación de la BS en la plataforma, y cómo queda tras la implementación realizada, justificando los elementos añadidos y su motivación. Posteriormente, se presenta el concepto de *wrapper*, sus atributos y los métodos que permiten la separación de las funcionalidades de la BS. Tras ello, se expone cómo se despliegan los *splits* y cómo se integran los *wrappers* en *srsRAN*. Por último, se argumenta el procedimiento de automatización de las pruebas, así como el tratamiento de los ficheros de resultados.

4.1. Visión general: diagramas de clases y estructura definida

En la Figura 4.2 se presenta el diagrama de clases de la implementación inicial de *srsRAN* de la *Estación Base* sobre la cual se ha realizado el desarrollo del proyecto. Como se puede observar, se distinguen 2 tipos de elementos: las capas de la pila de protocolos (representadas en color verde) y los interfaces que permiten la comunicación entre las capas del *stack* (en color naranja). De forma resumida, haciendo uso de la orientación a objetos y capacidades de definir herencias en C++, los interfaces se definen como clases abstractas de las que heredan las clases que representan los protocolos e implementan los interfaces concretos. A continuación, se pondrán ejemplos concretos de implementación.

Antes de ello, es importante destacar la nomenclatura de los nombres de los interfaces, la cual se ha mantenido en los elementos que se han añadido en el proyecto para implementar los *splits*. Cada interfaz se nombra según la siguiente estructura: *a_interface_b* que se puede interpretar como *el interfaz que 'a' muestra a 'b'* o, dicho de otro modo, el conjunto de funciones que 'b' puede ver y utilizar de 'a'. Por ejemplo, *pdcp_interface_rlc* sería el interfaz que PDCP muestra a RLC, o las funciones que RLC utiliza para comunicarse con PDCP. Otro aspecto relevante son las relaciones establecidas entre las diferentes clases e interfaces. Concretamente, se han empleado tres tipos: herencias, agregaciones y asociaciones.

- Las herencias permiten que una clase herede y reutilice los métodos y atributos de otra, sin la necesidad de tener que volver a implementarlos. Además, facilita el despliegue de clases genéricas que se utilizarán para generar herencias para clases más especializadas. En el desarrollo del proyecto,

los protocolos heredan de los interfaces, permitiendo que los primeros dispongan de todos los métodos requeridos para el intercambio de información entre los niveles del *stack*.

Por ejemplo, PDCP hereda de *pdcp_interface_rlc*, *pdcp_interface_rrc* y *pdcp_interface_gtpu*. Por su lado, RLC, RRC y GTP-U tendrán punteros (agregación) de PDCP, que cada uno verá como el interfaz que le permite interactuar con PDCP.

- Las agregaciones se basan en entender un objeto como la composición de varios [38], donde la clase agregada se comporta como un atributo de la que lo contiene. En este contexto, se han empleado agregaciones para que cada una de las capas puedan acceder a los interfaces que empleará para comunicarse con el resto de capas. Así, por ejemplo, el interfaz *rlc_interface_pdcp* (usado para la comunicación PDCP$\rightarrow$ RLC) se agrega a la clase PDCP.

- Por último, el tercer tipo de relación utilizado, son las asociaciones, que describen una conexión entre varios objetos. En este caso, existe una asociación entre aquellas capas que pueden comunicarse entre sí: por ejemplo, PDCP está asociado con RLC, RRC y GTP-U, mientras que no está asociado con la capa PHY, ya que no dispone de comunicación directa con dicha capa (no existen interfaces phy-pdcp ni pdcp-phy).

En la Figura 4.3 se presenta la implementación final que se ha realizado en el desarrollo. En primer lugar, es importante destacar la definición de un nuevo elemento: el *wrapper*. Este actúa como un intermediario, es decir, a diferencia de la implementación inicial, donde la comunicación entre las capas era directa por medio de punteros a los interfaces, en este caso ya no lo es. Así, por ejemplo, para las comunicaciones PDCP $\leftrightarrow$ RLC, los *wrappers rlc_pdcp_wrapper* y *pdcp_rlc_wrapper* actúan como punto medio en la comunicación, tal como se indica a continuación.

$$\text{PDCP} \longleftrightarrow \textit{rlc_pdcp_wrapper} \longleftrightarrow \textit{pdcp_rlc_wrapper} \longleftrightarrow \text{RLC}$$

Al igual que las capas, los *wrappers* también heredan de los interfaces los métodos que necesitan para la comunicación entre las capas. Por ejemplo, *pdcp_rlc_wrapper* hereda del interfaz *pdcp_interface_rlc*. Además, la mayor parte de los *wrappers* están asociados a las dos capas que comunican: por ejemplo, *pdcp_rlc_wrapper* tiene conexión con PDCP y RLC. Únicamente, hay 4 excepciones: *gtpu_rrc_wrapper*, *rrc_gtpu_wrapper*, *phy_rrc_wrapper* y *rrc_phy_wrapper*, ya que ninguno de ellos tiene puntero de RRC. Esto se debe a que las comunicaciones en las que intervienen estos *wrappers* son unidireccionales (de RRC hacia PHY o GTP-U) donde *gtpu_rrc_wrapper* y *phy_rrc_wrapper* envían la información generada por RRC hacia GTP-U y PHY respectivamente. Mientras que *rrc_gtpu_wrapper* y *rrc_phy_wrapper* se encargan de la recepción y de-serialización de los mensajes.

Otra diferencia respecto a la implementación inicial es que se han modificado las asociaciones de las capas entre sí. De este modo, se asocian a los *wrappers* que se emplearán para comunicarse. Así, en el caso la comunicación PDCP $\rightarrow$ RLC, PDCP tendrá conexión con *rlc_pdcp_wrapper* en vez de con RLC.

A modo de resumen, en la Figura 4.1 se representan, tanto para la versión inicial como para la implementación final, las capas PDCP y RLC, sus interfaces y *wrappers*. Como se puede ver, las clases RLC y PDCP heredan de *rlc_interface_pdcp* y *pdcp_interface_rlc*, respectivamente. A su vez, PDCP posee (agregación) un puntero a la implementación de *rlc_interface_pdcp* que hace RLC. Por su parte,

RLC posee un puntero a la implementación de *pdcp_interface_rlc* que hace PDCP. De este modo, cuando PDCP quiere enviar información a RLC (enlace descendente) usará los métodos definidos en el interfaz *rlc_interface_pdcp* y viceversa.

Como se puede observar en la imagen inferior de la Figura 4.1, los *wrappers* actúan como punto intermedio en la comunicación entre los protocolos. Por ejemplo, el protocolo RLC, en lugar de tener un puntero a la implementación de *pdcp_interface_rlc* que hace PDCP, tiene un puntero a la implementación que hace el *wrapper* correspondiente (*pdcp_rlc_wrapper*). A su vez, el *wrapper* es capaz de comunicarse con PDCP. La inserción de los *wrappers* permite interceptar la comunicación entre cada par de protocolos de forma transparente a la implementación existente. Es decir, no se necesitan hacer modificaciones en el funcionamiento interno de los protocolos, y todos ven los mismos interfaces, aunque las entidades que los implementan sean diferentes.

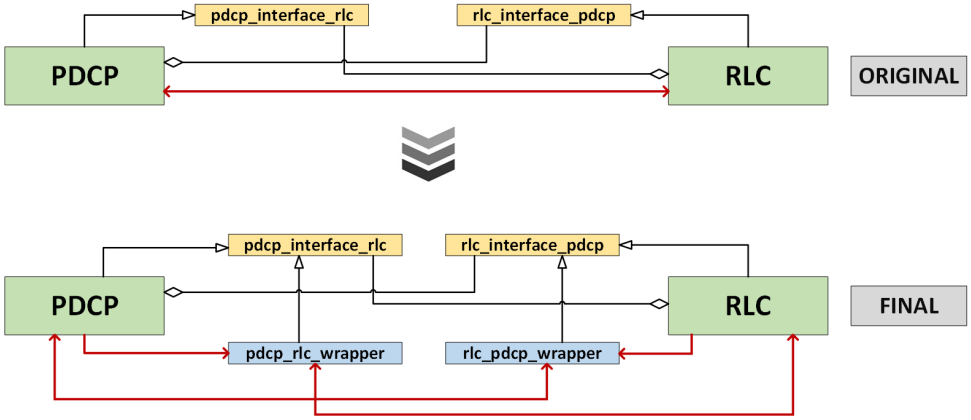


Figura 4.1: Comparativa entre la versión original y la final, comunicación PDCP ↔ RLC.

4.2. Diseño de los *wrappers*

Seguidamente se explica en mayor profundidad el diseño de los *wrapper*. Para evitar repeticiones, se describirán únicamente los elementos y métodos del *wrapper rlc_pdcp*, ya que salvo por los métodos propios de los interfaces el diseño es el mismo en todos los casos.

El *wrapper rlc_pdcp* es uno de los agentes que permite la comunicación PDCP ↔ RLC. En la Figura 4.4 se muestra una representación más detallada sobre las capas y *wrappers* que posibilitan la comunicación PDCP ↔ RLC: sus atributos y métodos.

4.2.1. Atributos del *wrapper*

Cada uno de los *wrappers* posee una serie de parámetros. Estas variables son privadas, por lo que, únicamente pueden ser accedidas desde la propia clase. A continuación, se describen los tipos de variables definidos, así como su utilidad.

- **Acceso a las capas**: Variables tipo puntero que permiten al *wrapper* tener conexión con las capas que comunica. Estas variables son las que establecen la relación de asociación del *wrapper* con las capas. En este caso, se corresponden con las variables *m_pdcp* y *m_rlc*. Por lo general, cada *wrapper*

tiene conexión con 2 capas, exceptuando los wrappers *gtpu_rrc*, *rrc_gtpu*, *phy_rrc* y *rrc_phy* donde ninguno tiene puntero de RRC.

- **Sockets**: Se corresponden con instancias (*m_socket_sender* y *m_socket_receiver*) de la clase *fs_socket*, que define los parámetros y métodos para el establecimiento de los *sockets* zeroMQ entre los pares de *wrappers*. Cada *wrapper* dispone de 2 *sockets* configurados en modo *Request-Response*: uno de ellos para la transmisión (en este caso, para PDCP → RLC) y otro para la recepción (RLC → PDCP).

  Los *sockets* se emplearán únicamente para comunicar dos capas que se encuentren separadas. Por ejemplo, en el caso de la comunicación entre PDCP-RLC, se emplearán *sockets* en el *split PDCP_RLC*, ya que en ese escenario PDCP estará en la CU y RLC en la DU. En la Figura 4.5, se presentan los 2 *sockets* zeroMQ que se establecen entre los wrappers para conseguir la comunicación remota entre las capas PDCP y RLC.

- **Buffers**: Se corresponde con el *buffer* donde se serializará la información que se ha de enviar (o se ha recibido) a través del *socket*. Se caracteriza por tener un tamaño de 20000 bytes. Para evitar problemas de lectura/escritura en el *buffer*, se ha optado por disponer de colas separadas para la transmisión y recepción de información.

- **Tipo de configuración del wrapper**: Variable de tipo FsConf_e (se trata de un enumerado que tiene dos posibles estados: LOCAL o REMOTE) que especifica el tipo de comunicación que se desplegará entre ambas capas.

  Cuando es de tipo *LOCAL*, ambas capas se encuentran en el mismo elemento (CU o DU). Por lo tanto, no es necesario el envío de información a través del socket, y la comunicación se establecerá mediante las variables punteros. En el caso del *wrapper rlc_pdcp*, este emplearía *m_rlc* para el envío de información desde PDCP hacia RLC.

  Sin embargo, si es de tipo *REMOTE*, se producirá el envío de información hacia el otro *wrapper*, y sería éste quien llame a la función local. Aunque se explicará en mayor profundidad en la sección 4.2.3, en la Figura 4.6 se presenta esta operativa de forma gráfica.

- **Parámetros para los *sockets***: Se corresponden con las duplas dirección IP y puerto, necesarias para el establecimiento del *socket*. Cada *wrapper* dispone de 2 duplas: la primera correspondiente a su dirección IP y al puerto donde escuchará el *socket* de recepción: *m_socket_receiver*. La otra dupla se corresponde con el otro *socket*, por lo que, es la dirección y puerto donde el *socket* de transmisión deberá enviar la información.

- **Mutex**: Se encarga de evitar que varios hilos empleen un método de forma simultánea, lográndose eludir situaciones como conflictos en la escritura de los *buffers*.

- **Variables auxiliares**: Aunque su uso no es necesario para el correcto funcionamiento del *wrapper*, se ha empleado para poder realizar su evaluación. La variable *logFile* indica el nombre del fichero donde se almacenará información relativa a la cantidad de datos enviados por el *socket*.

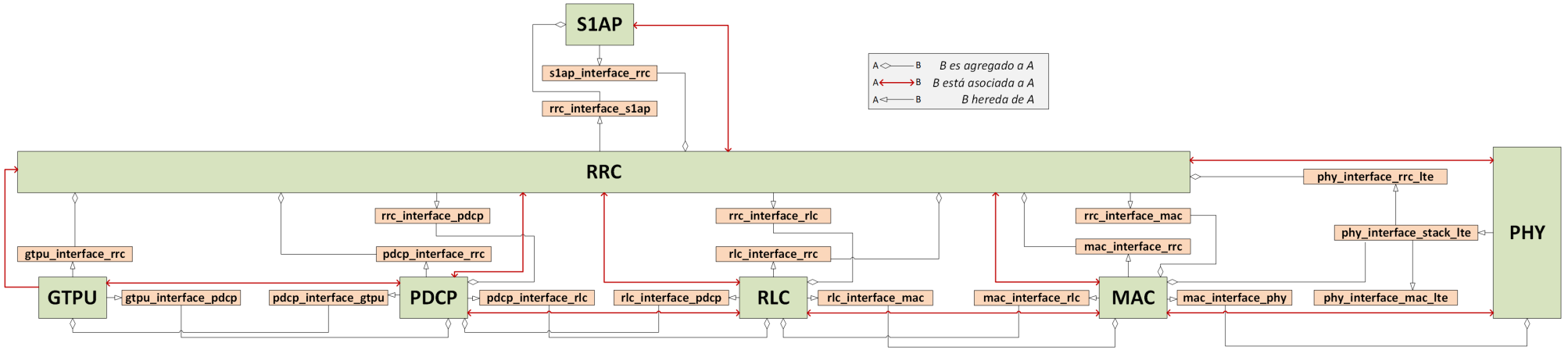


Figura 4.2: Diagrama de clases de la implementación original: relaciones de herencia, agregación y asociación.

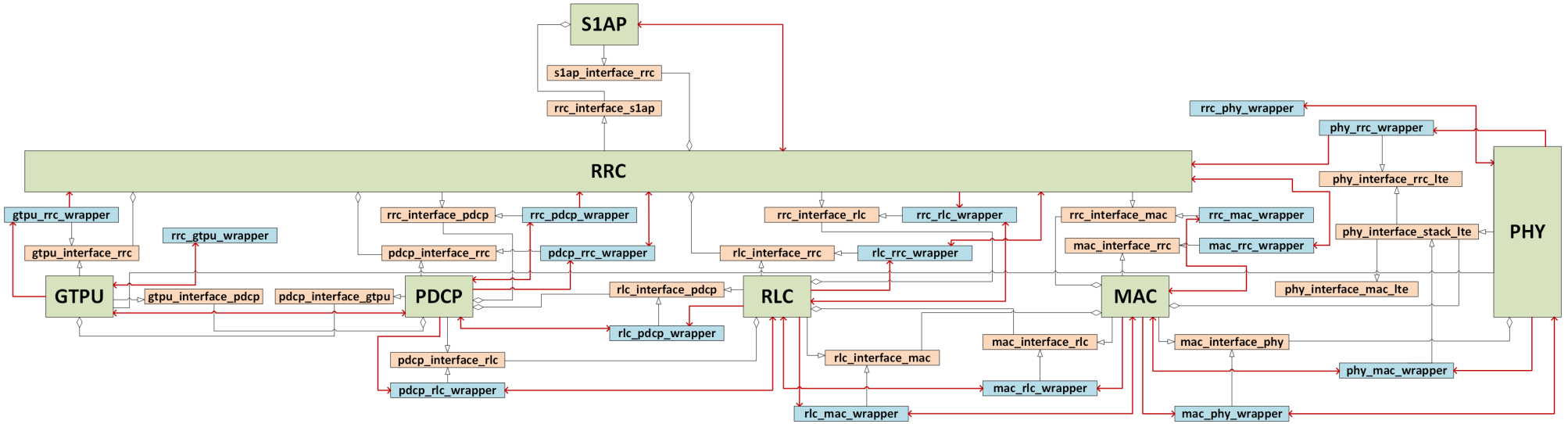


Figura 4.3: Diagrama de clases de la implementación final: relaciones de herencia, agregación, asociación y comunicación de las capas con los *wrappers*.

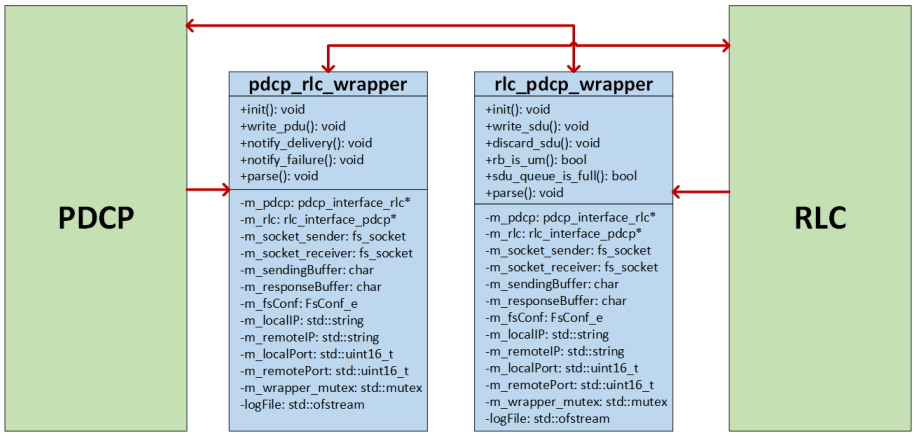


Figura 4.4: Comunicación PDCP ↔ RLC: agentes que permiten el intercambio de información.

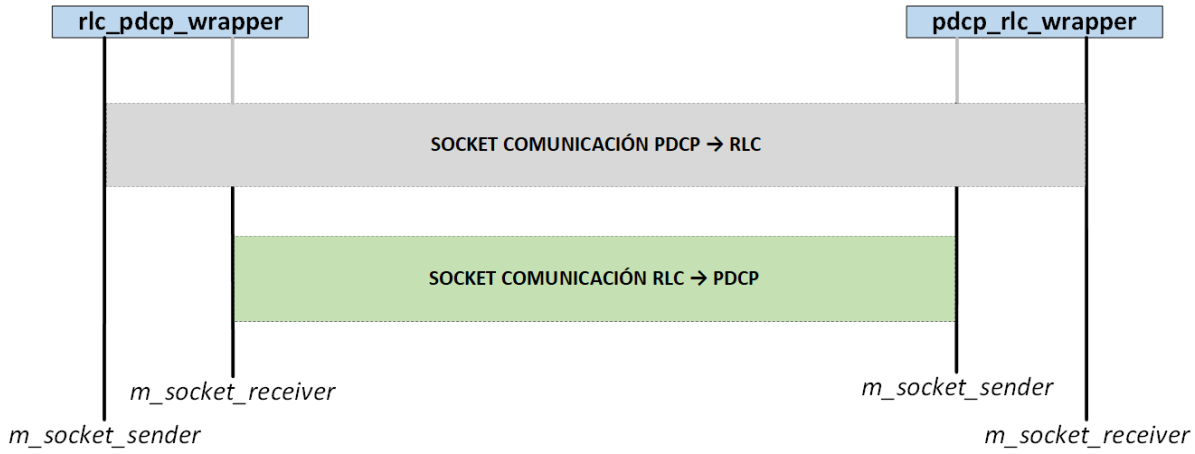


Figura 4.5: *Sockets* establecidos entre *rlc_pdcp_wrapper* y *pdcp_rlc_wrapper* para la comunicación PDCP ↔ RLC.

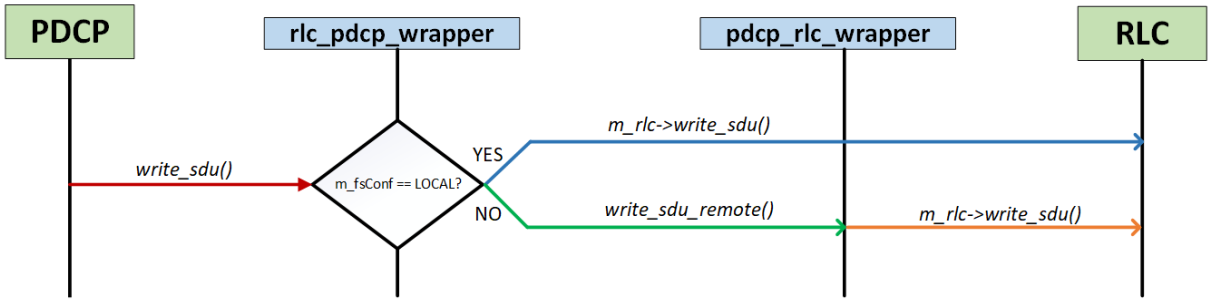


Figura 4.6: Operativa de trabajo en función de la configuración del *wrapper*.

4.2.2. *Sockets* de transmisión y recepción

Para poder definir los *sockets* y emplearlos, se ha definido una clase denominada *fs_socket* que aglutina todas las variables y métodos necesarios. En la Figura 4.7 se presentan de forma resumida las características de la clase y, en los siguientes párrafos, se explica la función de cada atributo y método.

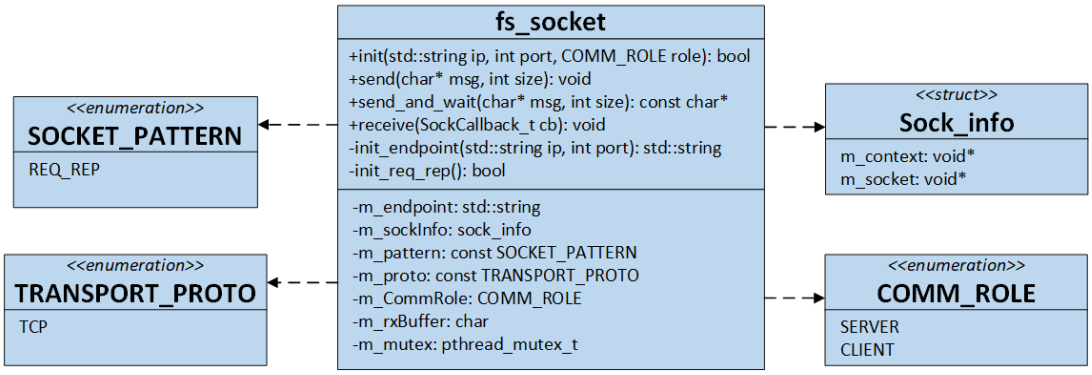


Figura 4.7: Clase *fs_socket*: atributos y métodos

Parámetros del *socket*

- **Endpoint**: Es un *string* que sintetiza el protocolo de transporte, así como la dirección IP y el puerto donde escuchará el *socket*. Se caracteriza por tener la siguiente estructura: *<protocolo>://<IP>:<puerto>*.
- **Información del *socket***: Se trata de una estructura (*sock_info*) que almacena el contexto (*m_context*) y un puntero (*m_socket*) que permitirá acceder al *socket* generado para poder usarlo.
- **Patrón**: Se corresponde con el tipo de *socket* que se establece. Como se argumentó en la Sección 3.3, zeroMQ permite desplegar varias clases de patrones; sin embargo, en esta implementación siempre se utiliza el esquema *Request-Response* (REQ-REP). Así, este atributo (de tipo enumerado) indica el tipo de *socket*, aunque siempre es *REQ-REP*.
- **Protocolo de transporte**: Denota el protocolo de transporte que se utiliza en los sockets. Aunque *zeroMQ* puede operar con UDP o *WebSockets*, todas las pruebas se han realizado sobre TCP. Se trata de una variable tipo enumerado.
- **Rol**: En los dos extremos del *socket*, se definen roles diferentes. Como se ha mencionado, a nivel zeroMQ se definen los roles *REQ* y *REP*. Sin embargo, para evitar confusiones se ha decidido utilizar una nomenclatura más familiar: denominando *cliente* (*CLIENT*) al *REQ* y *servidor* (*SERVER*) al *REP*. Por lo tanto, esta variable, correspondiente con un tipo enumerado, indica qué rol tiene el *wrapper* en el *socket* establecido.
- **Buffer de recepción**: Se corresponde con la variable donde se almacenan los bytes recibidos por el *socket* para su posterior tratamiento. Tiene un tamaño fijo de 20000 bytes.
- **Mutex**: Variable que impide que varios procesos o hilos intenten hacer uso del *socket* de un *wrapper* de forma simultánea. Se evitan así problemas como que 2 o más hilos intenten enviar a la vez.

Sobre los atributos de patrón, protocolo de transporte y rol, es importante resaltar que se han definido variables y clases enumeradas con el propósito de facilitar una posible extensión a otros patrones, protocolos y roles en el futuro.

Métodos del *socket*

- **bool init(std::string ip, int port, COMM_ROLE role)**

  Función de inicialización del *socket*. Se encarga de llamar a la función que genera el *endpoint* (*init_endpoint*), crear el contexto y llamar al método que define el *socket* REQ-REP: *init_req_rep*.

  Internamente, este método llama al *init_req_rep* que se encarga de instanciar e inicializar el *socket*, manteniéndolo a la escucha cuando se configura como *SERVER*) o estableciendo la conexión en caso de que tenga el rol *CLIENT*. En ambos casos, almacena el puntero del *socket* para su posterior utilización. Además, define una serie de *timeouts* para la notificación de errores en caso de que se produzca algún percance con el *socket*.

- **void send(char* msg, int size)**

  Método utilizado para el envío de información a través del *socket*. Este tipo de mensajes no requiere respuesta, por lo que no se realiza ninguna espera. Se emplea para responder a los mensajes recibidos enviando primitivas de respuesta o, cuando sea necesario, la información retornada por funciones nativas de la implementación de *srsRAN*.

- **const char* send_and_wait(char* msg, int size)**

  Realiza el envío de datos a través del *socket* e inmediatamente espera la recepción de una respuesta por parte del otro extremo. La respuesta se almacena en el *buffer* de recepción (m_rxBuffer) y, tras ello, se retorna al *wrapper*. Su actividad está protegida mediante el *mutex m_mutex*, evitándose que varios hilos intenten enviar por el *socket* cuando este está esperando una respuesta.

- **void receive(SockCallback_t cb)**

  Método encargado de la recepción de mensajes. Este proceso se ejecuta en un hilo independiente, y está en escucha continua. Cuando recibe información, almacena dicha información en el *buffer* y llama al Callback, es decir, a una función del *wrapper* denominada *parse()*, que se encargará del tratamiento de la información recibida.

Respecto a los tres últimos métodos, correspondientes al uso del socket establecido, es importante mencionar que se tratan de funciones públicas, es decir, pueden ser accedidas por otras clases no *fs_socket*. En este caso, serán utilizadas por los *wrappers*. Por otro lado, es necesario mencionar que dependiendo del rol establecido se utilizarán unas funciones u otras.

4.2.3. Métodos del *wrapper*

A continuación, se procede a explicar la filosofía y operativa de trabajo de los métodos de los *wrappers*. Para ello, se utilizará la función *discard_sdu* (comunicación PDCP $\rightarrow$ RLC) como apoyo de la explicación, aunque también se argumentarán otros posibles escenarios que no se dan en ese caso. En la Figura 4.8 se presenta de forma gráfica y sintetizada cómo la operación de las funciones desarrolladas.

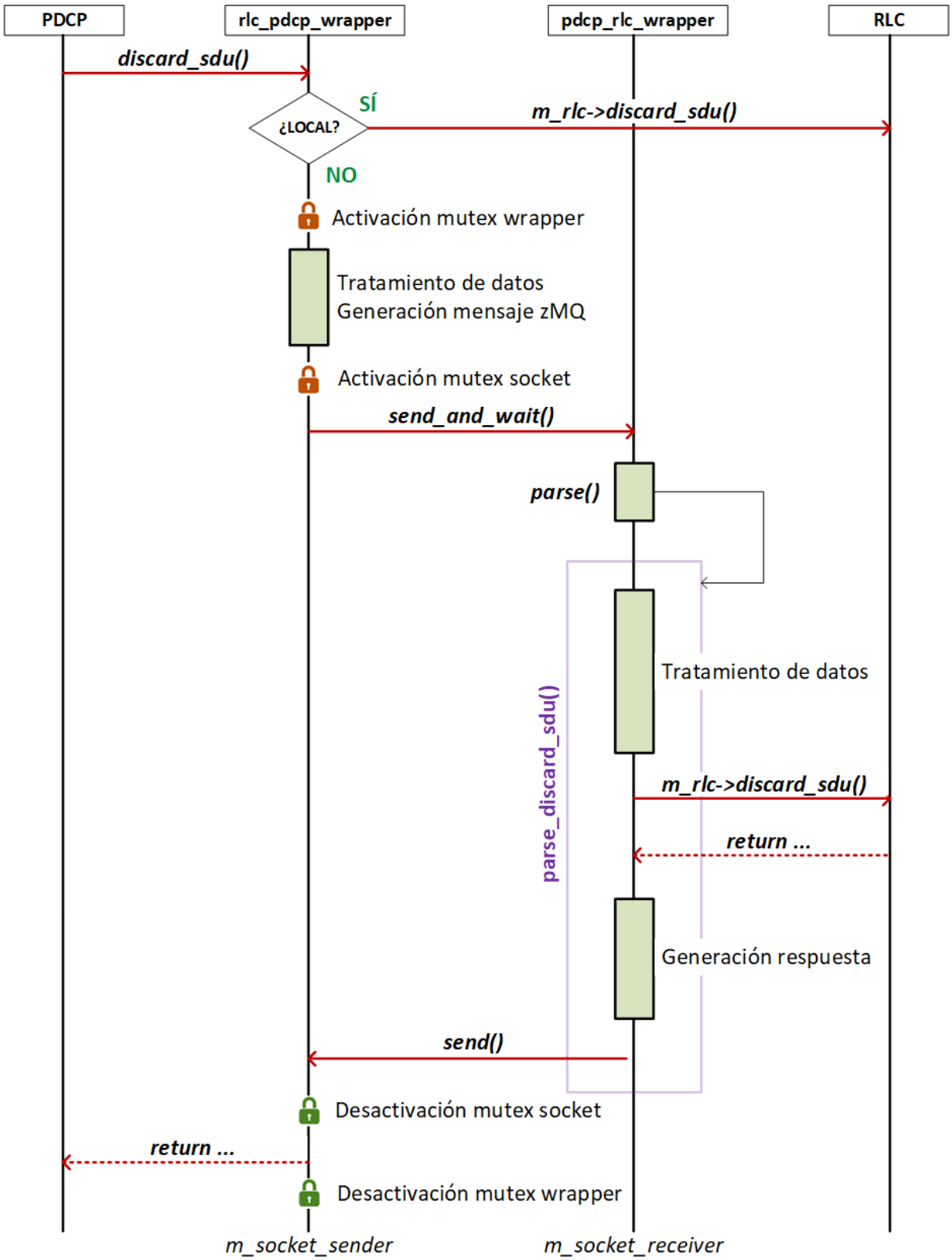


Figura 4.8: Operativa de trabajo de los métodos implementados. Ejemplo: *discard_sdu*()

En primer lugar, cuando una capa realiza una llamada a función para interactuar con otra capa, en el *wrapper* pertinente se realiza una comprobación de su configuración. En caso de que esta sea *LOCAL*, es decir, que las dos capas que se van a comunicar se encuentran en el mismo punto (en la CU o DU), la

comunicación se realiza a través de punteros. En el caso de *discard_sdu*, con una configuración *local*, el *wrapper rlc_pdcp* empleará su puntero *m_rlc* para ejecutar la función y enviar la orden deseada directamente a RLC.

Sin embargo, si las capas se encuentran en elementos separados, es necesario realizar el envío de la información a través del *socket*, para que sea el otro *wrapper* quien realice la llamada a la función en la capa remota. En este caso, *rlc_pdcp_wrapper* enviaría a *pdcp_rlc_wrapper* los datos necesarios para llamar a la función *discard_pdu*. Una vez recibidos los parámetros requeridos, utilizando su puntero *m_rlc*, el wrapper *pdcp_rlc_wrapper* realizará la llamada.

Posteriormente, se detalla la gestión de los datos pasados a las funciones para generar los mensajes que se envían por los sockets *zeroMQ*.

Tratamiento de los datos: generación del mensaje *zeroMQ* y envío a través del *socket*

Primero, se incluye una variable de 1 byte que indica la primitiva (función que se está llamando). Encabezará el mensaje y se corresponde con un valor de un tipo enumerado predefinido. Este campo será utilizado por el otro *wrapper* para identificar la función a la que se quiere llamar.

Posteriormente, se procede a tratar las variables que realmente necesitará el otro extremo. En función de los tipos de los parámetros, existen tres estrategias principalmente:

- **Tipo I: enteros, flotantes.** En este caso, se corresponden con variables de un tamaño fijo y conocido. Para garantizar la operativa de la implementación en cualquier dispositivo, se realiza la conversión de las variables a *Big Endian*.

  De este modo, la interpretación de los mensajes enviados y recibidos por el *socket* es siempre la misma. En el otro extremo, se realiza la operación inversa, transformando los datos *Big Endian* al formato que utiliza el procesador en concreto.

  Respecto a este tipo de variables, es importante destacar que la implementación se ha realizado para que ambos extremos conozcan su tamaño. Por lo tanto, no es necesario enviar en el mensaje ninguna cabecera que indique el tamaño de la variable, reduciéndose la sobrecarga.

- **Tipo II: colas, PDUs.** Se corresponden con datos de tamaño variable. Por lo tanto, a diferencia del caso anterior, no es posible que los extremos conozcan *a priori* su tamaño.

  Sin embargo, este tipo de datos viene siempre acompañado por otros parámetros del primer tipo, que ambos extremos sí conocen sus tamaños. Por lo que no es necesario, en la práctica, incluir una cabecera que especifique el número de bytes de la variable Tipo II que se mandan: a partir del número total de bytes que ocupa el mensaje y del tamaño de las variables de Tipo I, se puede conocer fácilmente cuánto ocupa dicho parámetro.

- **Tipos III: estructuras y clases.** Este tipo de datos son más complejos, por lo que se han tratado con *protoBuffer*. Para ello, es necesario generar el fichero de definición de la clase (.proto), realizar su compilación para crear las librerías necesarias para la serialización/de-serialización y, por último, escribir los datos.

Al igual que en el caso anterior, aunque estas variables tienen un tamaño que es desconocido por los *wrappers*, no es necesario incluir una cabecera que indique el tamaño de la variable, ya que las librerías *protoBuffer* se encargan de gestionarlo.

Normalmente, los parámetros que se han de enviar son una combinación de los tipos presentados. Sin embargo, en el conjunto de los parámetros nunca existe más de una variable de Tipo II o III en el mismo mensaje, siempre viene acompañada por uno o varios parámetros de Clase I y se trata de la última variable colocada en el *buffer*. Estas certezas facilitan enormemente la implementación, ya que no se requiere incluir cabeceras con los tamaños parciales de las diferentes variables, sino que es suficiente con enviar a través del *socket* una variable que indique el tamaño total del mensaje. Seguidamente, en la Tabla 4.1 se presentan los diferentes formatos de los mensajes existentes en la implementación:

Tabla 4.1: Tipos de mensajes enviados por el *socket*: campos y tamaños.

Mensaje I	Primitiva (1B)	Variables Tipo I	
Mensaje II	Primitiva (1B)	Variables Tipo I	Variables Tipo II
Mensaje III	Primitiva (1B)	Variables Tipo I	Variables Tipo III

Una vez formado el mensaje, este se envía al otro extremo usando la función *send_and_wait* (esta se mantiene a la espera hasta recibir el valor de retorno de la llamada a la función) del *socket* establecido.

Recepción del mensaje en el otro extremo

En el otro *wrapper*, el socket de recepción (que está en escucha continua) recibe un mensaje. Cuando llega, el *socket* emplea su *callback*, entregando el mensaje a la función *parse*. Este método, a partir del campo inicial, es capaz de determinar de qué tipo de mensaje se trata y se le entrega a la función especializada en el tratamiento de ese paquete recibido: en el caso del ejemplo, *parse_discard_sdu*. Los datos recibidos se deben extraer del mensaje, y guardarse en las variables locales para ser usadas en la llamada a la función local.

Respecto a las colas y PDUs, se deben almacenar en su totalidad e indicar el puntero donde inicia y su tamaño. Por último, respecto a los parámetros Tipo III se deben deserializar y almacenar en las variables correspondientes.

Tras ello, el *wrapper* emplea su puntero y realiza la orden pertinente. En este caso, se utiliza *m_rlc* para llamar a *discard_sdu*. Aunque no es la situación del ejemplo, en algunas ocasiones la función llamada retorna algún valor que debe ser enviado al otro extremo.

Posteriormente, se genera un mensaje de respuesta que se enviará al otro extremo. Este estará compuesto obligatoriamente por una variable que identifica la función que está respondiendo (1 byte) y, en caso de que existan, por las variables que se han de retornar. Una vez generado, el mensaje se manda al otro *wrapper*, empleando la función *send*. Tras el envío, se desactiva el *mutex* y el *socket* vuelve a su estado de escucha.

Recepción de respuestas y finalización

El *wrapper* que inició la comunicación recibe un mensaje *zeroMQ*, poniendo fin a su periodo de escucha. En caso de que espere algún parámetro, realizará su extracción del mensaje y se lo retornará a la capa correspondiente. En caso negativo, se descarta. Por último, se desactivan los *mutex* del *wrapper* y del *socket*.

4.2.4. Establecimiento de los *splits*: ficheros de configuración

La plataforma *srsRAN* permite configurar el comportamiento de la red desplegada en profundidad mediante el uso de un total de 8 ficheros de configuración. Sin embargo, se ha trabajado únicamente con 3 de ellos: el *epc.conf*, el *enb.conf* y el *ue.conf*.

El archivo *enb.conf* es el elemento principal, ya que es el que se empleará para desplegar los diferentes *wrappers*. Para facilitar el despliegue de diferentes escenarios, se han generado nuevos ficheros de configuración específicos de cada elemento. De este modo, se dispone del *CU.conf* y del *DU.conf*, que indican la configuración de la CU y DU respectivamente. Además, se ha mantenido el fichero *enb.conf*, que se utilizará para desplegar la versión inicial (*LEGACY*), es decir, la BS completa donde se realizan todas las funcionalidades en un único elemento.

Para poder realizar la configuración y despliegue de diferentes escenarios, se ha incluido un campo en los ficheros de configuración, denominado *fs*, donde se concentran los parámetros necesarios para ello. A continuación, se presenta una breve explicación sobre los campos que componen *fs*:

- *type*: Indica el tipo de *split* que se desea implementar. En total, se disponen de 2 *splits* en funcionamiento (*PDCP_RLC* y *RLC_MAC*), así como otros que todavía están en desarollo (*RRC_PDCP_SPLIT* y *MAC_PHY_SPLIT*). Además, se define un tipo llamado *NONE*, usado para desplegar toda la pila de protocolos en un único elemento (es decir, la configuración *LEGACY*).
- *role*: Especifica el rol que tiene la instancia del programa en la red desplegada. Se definen dos posibles roles: CU y DU. En el caso de que se despliegue la BS completa, este parámetro no se revisa ni se utiliza posteriormente.
- *local_ip & remote_ip*: Se corresponden con las direcciones IP de los diferentes elementos (la CU y la DU).
- *Campo fs.ports*: Recoge todos los puertos donde escucharán cada uno de los *wrappers* en la configuración remota.

Por otro lado, es importante establecer cómo será la comunicación física entre la BS (en caso del despliegue de los *splits* es la DU quien se comunica con el usuario) y el UE. Una de las alternativas es el uso de un dispositivo físico que actúa como *Radio Unit*: un ejemplo de equipo que puede operar como tal es la *Universal Software Radio Peripheral* (USRP) B210.

Aunque esta situación es la más realista, implica la necesidad de separar la red en varios equipos (uno de ellos formado por el EPC y la Estación Base y el otro dedicado al usuario) y, además, disponer de varias USRP. Por ese motivo, en el proyecto se ha utilizado la capacidad de emulación de la comunicación

entre el UE y el eNB que proporciona srsRAN. En concreto, la comunicación de la capa física (por debajo de los splits) se realiza mediante una conexión *zeroMQ* para comunicar al usuario con la red. Aunque esta situación es menos realista, permite un despliegue más ágil de la red entera en un solo equipo, simplificando el desarrollo de nuevas funcionalidades. Por otro lado, esta emulación no afecta a la funcionalidad de los splits, ya que estos se encuentran en la BS por encima de la capa física.

Mediante la adecuación de los parámetros anteriormente descritos, se consigue desplegar la red LTE con una *Base Station* completa o dividida en dos elementos y se logra que el usuario sea capaz de conectarse a la red. Sin embargo, es necesario realizar una serie de modificaciones adicionales para lograr la comunicación y el envío de tráfico bidireccional. Seguidamente, se presenta en la Figura 4.9 un esquema gráfico donde se sitúan los elementos que intervienen en las comunicaciones, así como los parámetros que se han de configurar y que se explican a continuación.

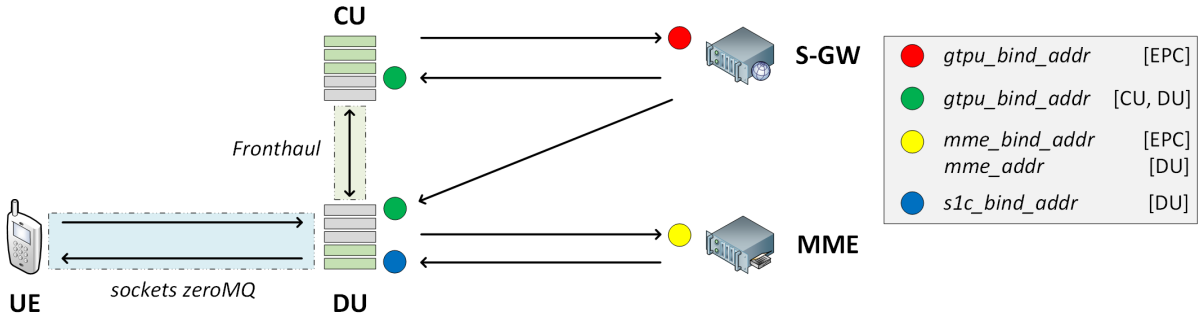


Figura 4.9: Configuración de la plataforma *srsRAN* para el establecimiento de la conexión y comunicación.

Respecto al *core* de la red (EPC), se deben configurar dos parámetros: *mme_bind_addr* y *gtpu_bind_addr*. El primero se corresponde con la dirección donde escuchará MME (encargado de la gestión de la movilidad, la autenticación y autorización del usuario en la red y otras responsabilidades de control) mientras que el segundo se trata de la dirección donde escucha S-GW (cuya función final es permitir la comunicación del usuario con otros) las peticiones GTP-U en el EPC y que permite establecer la comunicación eNB $\rightarrow$ EPC. En la Figura 4.9, estas direcciones se representan mediante los círculos amarillo y rojo respectivamente.

En cuanto a la eNB o sus respectivas divisiones (CU y DU), se deben configurar tres variables. La primera de ellas (*mme_addr*, representada en la Figura 4.9 con el color amarillo) indica en qué dirección IP está escuchando MME, y se corresponde con la dirección IP pública del equipo donde está desplegado MME. La segunda, *gtpu_bind_addr* (en la Figura 4.9, en color verde), determina dónde escuchará la BS las peticiones GTP-U. Se utiliza para la comunicación EPC $\rightarrow$ eNB. Por último, se ha de configurar el parámetro *s1c_bind_addr* (representado mediante un círculo azul) que se corresponde con la dirección donde la BS escuchará los mensajes del plano de control procedentes de MME. Mientras que en la eNB es necesario configurar las 3, en los *splits* es diferente: en la CU únicamente se indica el campo *gtpu_bind_addr* y en la DU se deben configurar todos.

4.2.5. Integración de los *wrappers* en la plataforma

El siguiente paso consiste en la integración de la estructura de *wrappers* generada dentro de la implementación nativa de *srsRAN*. De este modo, se logra desplegar hasta 2 *splits* funcionales: *PDCP_RLC*

y *RLC_MAC*.

Aunque el diseño adoptado se ha demostrado práctico y funcional, el estado actual de la implementación no incluye la separación total de todas las capas. Concretamente, no se ha logrado separar totalmente RRC (entidad de gestión) del resto de las capas de la pila, por lo que el desarrollo actual se centra en el plano de datos. Por ello, la entidad RRC se ejecuta en la DU, cuando debería estar en la CU. Por otro lado, la separación de la capa MAC de la física no está completa, por lo que tampoco se ha logrado desplegar un tercer split: *MAC_PHY_SPLIT*. En la Tabla 4.2, se presentan los *splits* desplegados y la distribución de las capas entre la CU y la DU.

Tabla 4.2: Capas desplegadas en cada elemento en función del tipo de *split*

	Tipo de *split*	
Agente	PDCP_RLC	RLC_MAC
CU	GTP-U, PDCP	GTP-U, PDCP, RLC
DU	S1-AP, RRC, RLC, MAC, PHY	MAC, S1-AP, RRC, PHY

Por otro lado, en la Tabla 4.3 se resume el estado del desarrollo de las separaciones. Entre todas las posibles separaciones, se han acometido todas con excepción de las separaciones intracapas y, actualmente continúan en fase de desarrollo las separaciones de PHY con MAC y RRC.

Tabla 4.3: Estado de trabajo de las separaciones funcionales del *stack* de protocolos. En color verde y el símbolo ✓ se resaltan aquellas separaciones que se han implementado y validado, en amarillo y ◌ las que se encuentran en fase de desarrollo y validación. En rojo y ✗, aquellas que no se han acometido y, por último, en gris se representan aquellas separaciones que no existen.

	phy	**mac**	**rrc**	**rlc**	**pdcp**	**gtpu**
phy	✗	◌	◌			
mac	◌	✗	✓	✓		
rrc	◌	✓		✓	✓	✓
rlc		✓	✓	✗	✓	
pdcp			✓	✓		
gtpu			✓			

La integración consiste simplemente en la inicialización de los *wrappers* y las capas en función de la configuración detallada en la Sección 4.2.4. Para ello, se inician tanto los *wrappers*, pasándoles las referencias a las capas como los niveles del *stack*, y modificando sus conexiones para que tengan comunicación con el resto de capas a través de ellos.

srsRAN define una clase (fichero *enb.cc*) que implementa la BS en su conjunto, instanciando al resto de clases y pasando punteros de unas a otras. Para incluir los *wrappers* en la Estación Base se ha modificado esta clase para que realice las instancias de acuerdo a la configuración.

En primer lugar, se inicializan los *wrappers* que intervienen en las comunicaciones con PHY, tanto con MAC como con RRC. Además, *enb.cc* instancia el *stack* (*enb_stack_lte.cc*), que es la clase que contiene a todas las capas (excepto RF y PHY) de la pila de protocolos.

Finalmente, es importante destacar que, en ambos ficheros, se realiza el despliegue de las capas y *wrappers* en función de los parámetros de configuración, fundamentalmente, el tipo de *split* y el rol. De

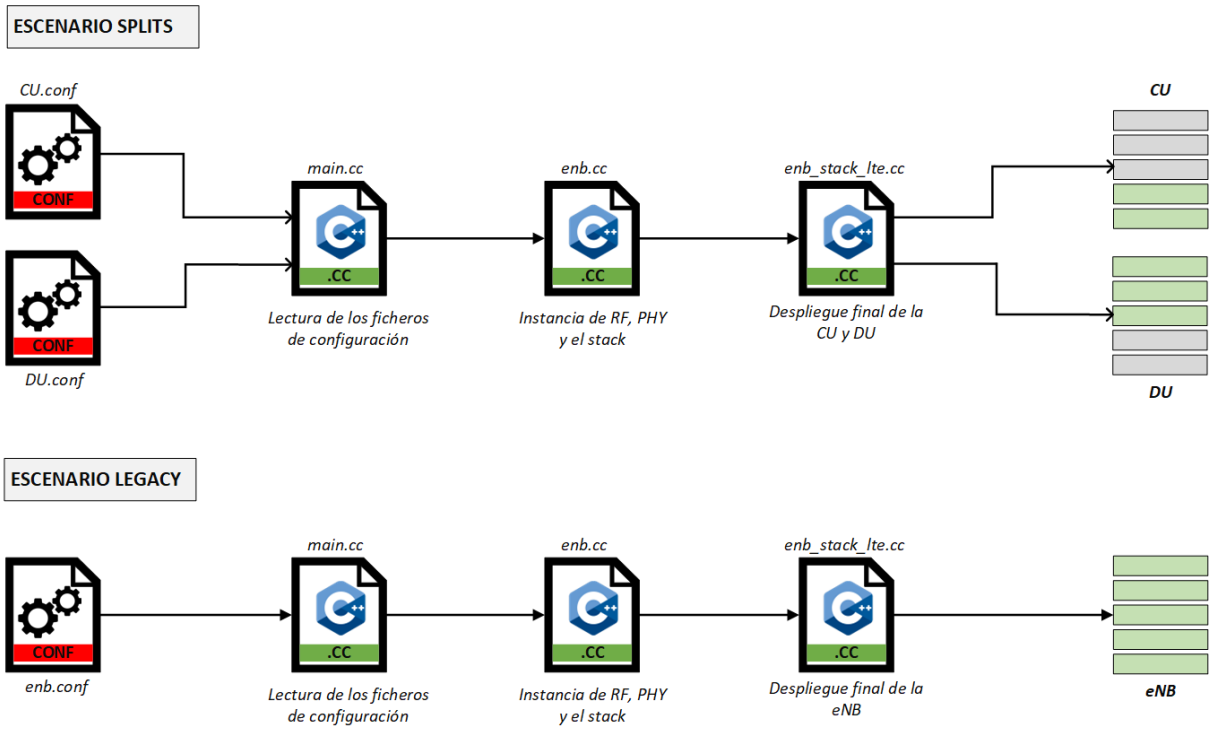


Figura 4.10: Procedimiento de despliegue de los diferentes escenarios en función de la configuración.

este modo, la capa física solo se despliega en la DU y nunca en la CU, o un *wrapper* se despliega en configuración remota solamente si las capas que comunica están separadas.

A modo de resumen de las Secciones 4.2.4 y 4.2.5, se presenta la Figura 4.10, donde se condensa todo el proceso de despliegue de los *splits*.

4.3. Automatización de las pruebas

Para poder realizar la validación de los diferentes *splits* desarrollados, se han elaborado una serie de *scripts* en *Python* que permiten controlar y desplegar los diferentes elementos que se utilizarán en la evaluación de forma ordenada. Aunque se explicará en mayor detalle en el Capítulo 5, se ha efectuado el estudio del comportamiento de la red en dos escenarios: el primero de ellos se corresponde con el despliegue total de la red en un único dispositivo y, en el segundo, se contempla la división de la red y sus elementos en diferentes equipos. En la Tabla 4.4, se presentan de forma detallada las características de las pruebas realizadas, así como los parámetros de las herramientas utilizadas.

El propósito de los *scripts* es automatizar el proceso de medida, asegurándose que se realiza de forma correcta. El procedimiento es sencillo: en primer lugar, se despliega el *core* de la red, es decir, el EPC. A continuación, se cambia el número de PRB, que indica la cantidad de recursos de radio-frecuencia utilizados. Para ello, se ha de modificar el campo *n_prb* de los ficheros de configuración (enb.conf o DU.conf, en función del tipo de *split*). Posteriormente, se inicia la *Base Station*, que puede ser el eNB o bien la CU y DU. En el caso del despliegue de algún *split*, es indiferente el orden en el que se despliegan la CU y la DU.

Tras ello, se inicia el usuario, que procede a realizar su conexión a la red. Llegado a este punto, en el

Tabla 4.4: Parámetros de configuración de las pruebas realizadas.

Características genéricas					
Número de PRBs	$[15, 25, 50, 100]$				
Tasas binarias	r = $\left\{\begin{matrix} 1 & 2 & 4 & [15\ PRBs] \\ 4 & 6 & 7{,}5 & [25\ PRBs] \\ 4 & 8 & 12 & [50\ PRBs] \\ 6 & 12 & 18 & [100\ PRBs] \end{matrix}\right\}$				Mbps
Número de iteraciones	10				
Herramienta *top*					
Frecuencia de medidas	0.2 s				
Herramienta *iperf*					
Protocolo de transporte	UDP				
Duración de la medida	60 s				
Frecuencia de medidas	0.2 s				

lado del EPC se crea un interfaz (*srs_spgw_sgi*) que actuará de puerta de salida (*router*) hacia Internet y se instancia un servidor DHCP para asignar direcciones IP en la sub-red 172.16.0.0/24. De este modo, el usuario obtiene una dirección IP perteneciente a la red 172.16.0.0/24, permitiendo la comunicación bidireccional con la red u otros usuarios.

El siguiente paso consiste en la monitorización, mediante la herramienta *top*, del consumo de CPU y memoria RAM de los elementos de la red mientras se realiza el envío de tráfico entre el usuario y la red. Para ello, se aplica un filtro a nivel de proceso, consiguiéndose observar solamente aquellos correspondientes a los elementos de interés. Además, se establece un servidor *iperf* en el interfaz *srs_spgw_sgi* y un cliente *iperf* en el lado del usuario, para que el UE envíe tráfico UDP al EPC (es decir, se trata de una comunicación *Uplink*). Además de los datos generados por las herramientas *top* e *iperf*, también se emplean las variables locales de los *wrappers* con configuración remota (variable *LogFile*, explicada en la Sección 4.2.1) para medir la cantidad de bytes que se generan y envían en cada uno de los *wrappers*.

Para cada valor de *Physical Resource Block* (PRB), se realiza el estudio de la red a tres tasas diferentes y se ejecuta un total de 10 repeticiones por cada tasa (véase la Tabla 4.4). Al final de cada iteración, se finaliza la monitorización de cada elemento y se almacenan las trazas obtenidas del *top* e *iperf* en ficheros separados, para su posterior tratamiento. En todos los casos, las trazas generadas contienen marcas que permiten mantener la referencia temporal entre los diferentes ficheros generados. Una vez finalizadas todas las ejecuciones de las tres tasas, se ha de inhabilitar la BS para modificar el número de PRBs. Por lo tanto, es necesario eliminar al usuario y posteriormente la eNB (o la CU y DU) mientras que el *core* de la red puede permanecer operativo. Una vez acabadas las pruebas para el último valor de PRB, se deshabilita totalmente la red, procediendo al proceso de lectura y procesado de los ficheros de traza.

Respecto a las medidas de tráfico en los *wrappers* remotos, es importante destacar que únicamente se almacena información para una sola repetición (de las 10 que se realizan) por cada valor de tasa (r) de cada PRB.

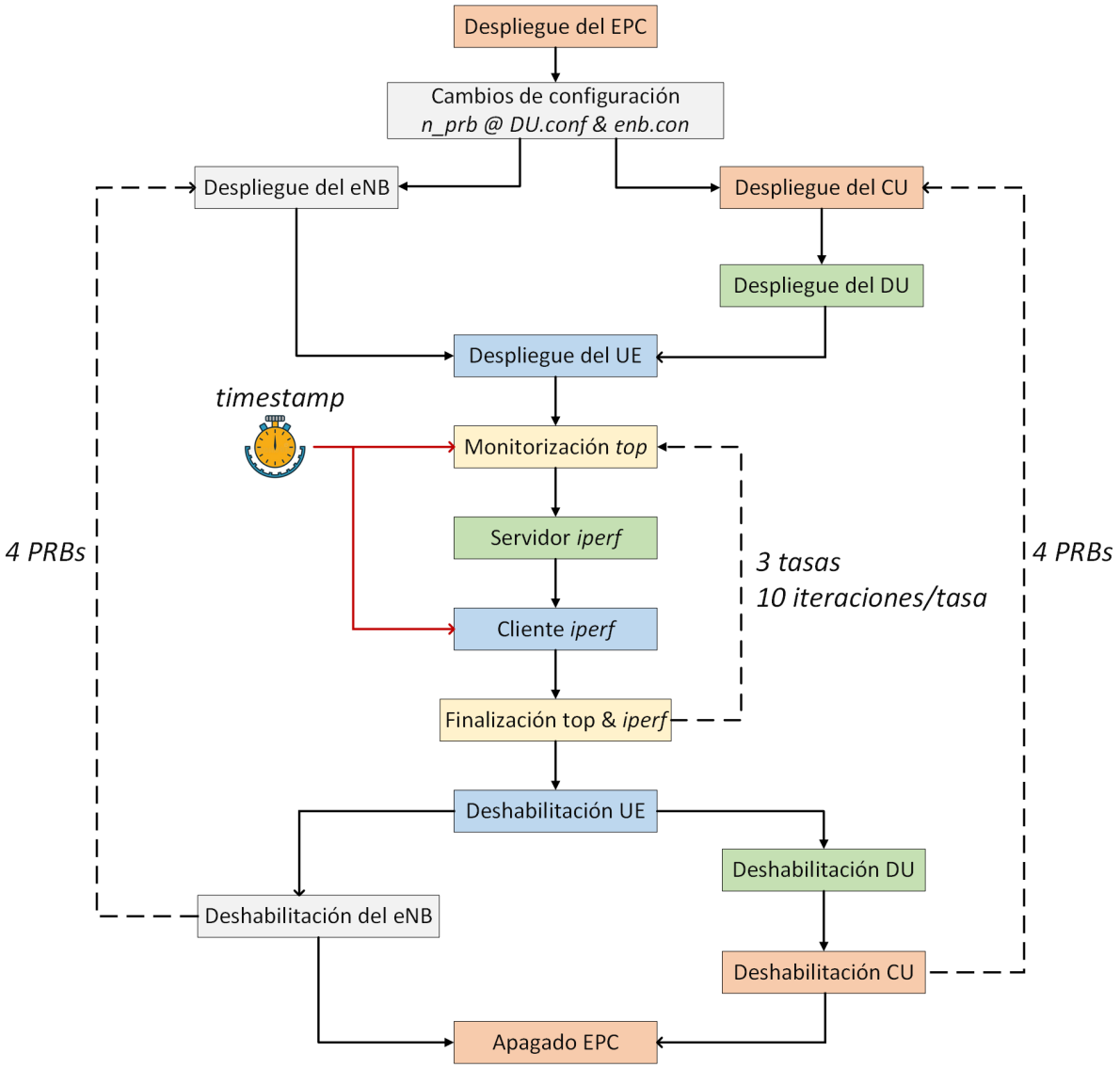


Figura 4.11: Procedimiento de automatización de las pruebas. Se muestra el escenario en el cual se divide la red en varios equipos, de manera que los procesos se ejecutan por separado. Mediante los colores de los procesos, se indican qué procesos se ejecutan sobre el mismo dispositivo. Si por el contrario se despliega toda la red en el mismo equipo, todos los procesos se realizan en el mismo dispositivo.

Para este proceso completo se han generado varios *scripts*. Uno de ellos realiza la operativa de las pruebas al completo (usado en el escenario del despliegue de la red en un único equipo) y otros *scripts* para separar la red en varios equipos: uno de ellos para el EPC y CU, otro para la DU y un tercero para el usuario. A modo de resumen, se presenta la Figura 4.11 donde se esquematiza el proceso de automatización de las pruebas.

4.4. Tratamiento y adaptación de los ficheros de resultados

Tras la realización de las pruebas, el siguiente paso consiste en la adaptación y tratamiento de los ficheros de resultados para su posterior análisis y obtención de conclusiones. El propósito de esta fase consiste en extraer las muestras tomadas por el programa *top* mientras se enviaba tráfico sintético a través

de la red mediante *iperf*. Para ello, se ha utilizado la librería *Pandas*[1] de *Python* que facilita el manejo y análisis de estructuras de datos. El procedimiento seguido para ello se puede dividir en las siguientes partes:

Adecuación de los ficheros *top*

En primer lugar, se deben descartar aquellas líneas que se encuentren incompletas como consecuencia de algún error de escritura que se haya podido producir durante las pruebas. Posteriormente, se elimina la línea inicial (véase la Tabla 4.5) donde se indican los campos que forman las muestras.

Tabla 4.5: Línea cabecera donde se indican qué campos conforman las medidas. En color rojo, se indican qué campos se descartan y, en verde, cuáles se mantienen.

TS	PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMM.

Tras ello, se debe modificar el separador decimal utilizado en las medidas del porcentaje de CPU (*%CPU*) y memoria (*%RAM*). La aplicación utiliza el separador decimal europeo (la coma, ‘,’) mientras que, para la obtención de resultados y representación en gráficas, se debe trabajar con el separador anglosajón: el punto (‘.’). Por último, se eliminan aquellos campos que no son de interés (véase la Tabla 4.5). En resumen, únicamente se mantienen aquellos parámetros referentes al porcentaje de uso de CPU y memoria utilizados en tiempo real.

Adaptación de los resultados *iperf*

Los ficheros generados por *iperf* se caracterizan por tener una estructura fija. En primer lugar, el archivo viene encabezado por 7 líneas que aportan información general sobre la herramienta (versión, SO en el que se ha usado) y la cabecera donde se indican los campos que conforman las muestras.

Posteriormente, se encuentran las medidas realizadas donde, como se comentó en la Sección 4.3, se ha incluido un campo adicional que recoge el momento en el que se realiza la medida. En la Tabla 4.6 se presentan los campos que forman las medidas *iperf*.

Por último, el fichero está formado por 8 líneas que finalizan la medida. Aunque esta contiene una línea donde se resume las principales estadísticas de la prueba, se elimina este fragmento, ya que no se utiliza.

El siguiente paso consiste en eliminar aquellos campos que no son de interés. Finalmente, únicamente se mantienen los campos del *timestamp*, tamaño de datos enviados (valor y unidades), la tasa binaria (valor y unidades) y el número total de datagramas. En la Tabla 4.6, los campos descartados se representan en color rojo y, en verde, los que se conservan. Por último, se almacenan el primer y último *timestamp* de las medidas, ya que se utilizarán en la tercera fase para sincronizar los resultados *top* de las medidas *iperf*.

Tabla 4.6: Campos que constituyen las métricas *iperf*. En color rojo, se indican aquellos campos que se descartan mientras que, en verde, se indican los que se conservan.

Timestamp	ID	Interval		Transfer		BW		Total Datagrams
		Value	Units	Value	Units	Value	Units	

[1] Página oficial de la libería *Pandas*: `https://pandas.pydata.org`

Sincronización de los ficheros

En último término, se utilizan los *timestamps* obtenidos en la fase anterior para encontrar las medidas *top* que se corresponden con el intervalo temporal de la prueba *iperf*. Para ello, se comparan los *timestamps* de las muestras *top* con las marcas temporales obtenidas con *iperf* hasta encontrar el rango que se encuentre dentro de ese rango temporal. Una vez obtenido dicho intervalo, se almacenan las medidas que se encuentren en él, descartándose aquellas que están fuera.

Capítulo 5

Evaluación de la implementación

En este capitulo se presentará la validación de los diferentes *splits* implementados, así como su rendimiento en equipos con diferentes características. Para ello, en primer lugar se describirán los dispositivos utilizados, y posteriormente se describirán los resultados obtenidos.

5.1. Equipos y escenarios

Para la evaluación de la implementación realizada bajo varios escenarios, se han utilizado tres equipos con distintas capacidades computacionales en términos de número de CPUs, frecuencia máxima de operación de estas y memoria RAM. En la Tabla 5.1, se resumen las principales características de los dispositivos utilizados así como la nomenclatura que se ha utilizado para referirse a cada equipo en la representación de los resultados.

Además, se ha utilizado un *switch* GS108 de *NetGear*, con 8 puertos de 1 Gbps cada uno, para la interconexión de los dispositivos en aquellos escenarios donde se divida la red implementada en múltiples dispositivos.

En el primer escenario (Escenario I), se realiza el despliegue de la red en un único dispositivo midiéndose el consumo computacional (CPU y memoria) que generan las entidades que conforman la BS mientras se envía tráfico. En cada uno de los tres equipos, se ha estudiado el comportamiento de la red original (*LEGACY*) donde toda la BS se despliega en un único elemento y, posteriormente, se testean los 2 *splits* desarrollados: *PDCP_RLC_SPLIT* y *RLC_MAC_SPLIT*.

Asimismo, se ha monitorizado el tráfico que se genera en cada uno de los *wrappers* implementados, es decir, se mide el tráfico intercambiado entre dos capas que se encuentren separadas (una de ellas centralizada y la otra en la DU). Para ello, se ha realizado un registro del momento en el que se envía

Tabla 5.1: Características de los dispositivos utilizados en la implementación.

Nomenclatura	Nº CPUs	Frecuencia máxima de trabajo CPU (GHz)	Memoria RAM (GB)
PC1	8	2.7	16
PC2	4	2.6	8
PC3	4	2.2	16

Tabla 5.2: Direcciones IP y roles que asumen los equipos en el Escenario II.

Rol	Equipo	Dirección IP
EPC + CU	PC1	10.0.0.4
DU	PC3	10.0.0.3
UE	PC2	10.0.0.2

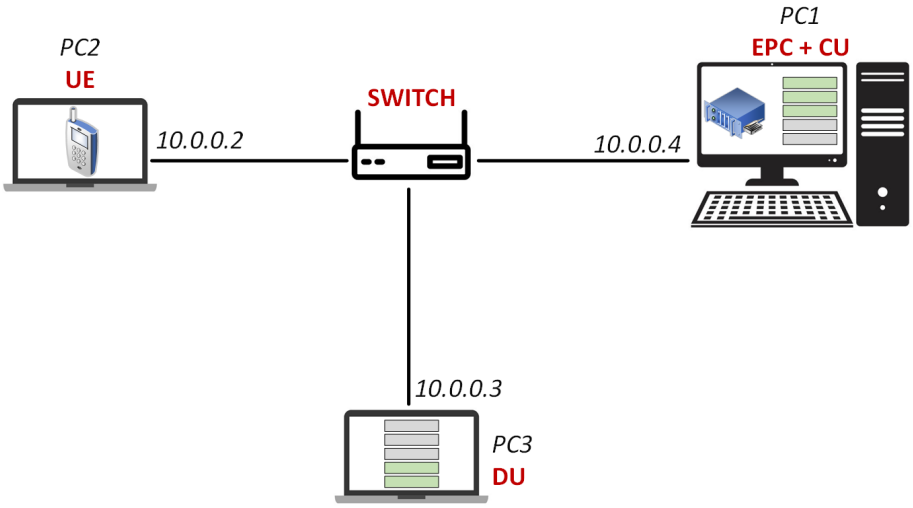


Figura 5.1: Topología desplegada para las pruebas del Escenario II.

información a través del *wrapper* y la cantidad de bytes enviados.

Por otro lado, es importante destacar que, aunque no se logró separar la capa RRC de la DU y centralizarla en los *splits* implementados, se ha decidido establecer las comunicaciones de RRC con el resto de las capas emplazadas en la DU (exceptuando con la capa física) a través de los *sockets*. En conclusión, todas las comunicaciones con RRC son remotas excepto PHY$\Longleftrightarrow$RRC. De este modo, se consigue monitorizar el tráfico de control que se genera en la BS.

En el segundo escenario (Escenario II), se realiza una división de la red en diferentes equipos y se despliegan los *splits* completamente implementados. La utilización de múltiples dispositivos permite dividir las funcionalidades de la red y lograr un estudio de la implementación en un entorno más realista. Concretamente, el *core* de la red y la CU se despliegan en el mismo dispositivo (*PC1*), la DU se despliega en el *PC3* y, por último, el usuario en el *PC2*.

El *switch* garantiza la comunicación de los tres equipos tras realizarse la asignación de direcciones IP (pertenecientes a la sub-red 10.0.0.0/24) para cada uno de ellos y el establecimiento de las rutas pertinentes. En este escenario, también se ha medido el consumo computacional así como el tráfico generado por cada uno de los *wrappers* con configuración remota.

La Tabla 5.2 resume el rol y dirección IP de cada dispositivo en este entorno. Además, en las Figuras 5.1 y 5.2 se representa la topología desplegada así como la distribución física e interconexión de los equipos para el Escenario II.

Por último, se ha desplegado un tercer escenario (Escenario III) para evaluar si los *splits* implementados cumplen con los requisitos exigidos por el *Small Cell Forum* [32]. Para ello, se ha utilizado una topología similar al Escenario II pero empleando únicamente dos equipos. En este caso, se ha usado el *PC2* para

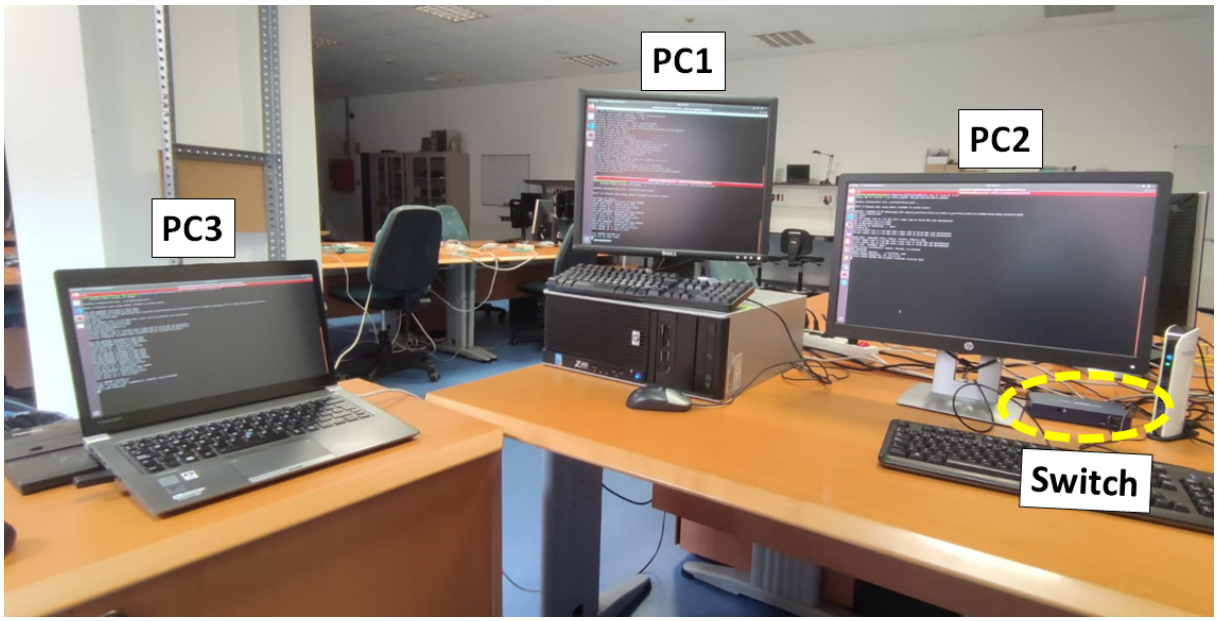


Figura 5.2: Despliegue real en el laboratorio para las pruebas del Escenario II.

Tabla 5.3: Direcciones IP y roles que asumen los equipos en el Escenario III.

Rol	Equipo	Dirección IP
EPC + CU	PC1	10.0.0.4
DU + UE	PC2	10.0.0.2

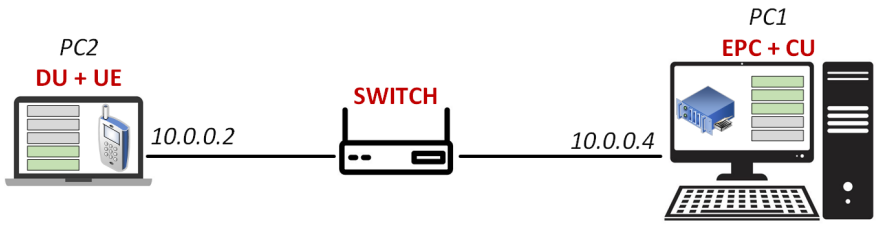


Figura 5.3: Topología desplegada para las pruebas del Escenario III.

desplegar el EPC y la CU mientras que se ha empleado el *PC1* para la implementación de la DU y del usuario. En la Tabla 5.3 y la Figura 5.3 se resumen tanto las direcciones IP como la topología desplegada en este escenario.

En este tercer escenario, para los 2 *splits* totalmente implementados se ha evaluado el comportamiento de la red a las tres tasas (1, 2 y 4 Mbps) de la configuración de 15 PRBs (véase la sección de *Características genéricas* de la Tabla 4.4) bajo dos supuestos: en el primero de ellos, no se introduce ningún retardo en las comunicaciones mientras que, en el segundo, se introduce un retardo en el *fronthaul*, es decir, en la comunicación CU-DU. En este escenario, se ha monitorizado el consumo computacional de la CU y DU en cada caso y se han realizado medidas del tráfico generado en los *wrappers* con el propósito de estudiar cómo influyen los *delays* incluidos en el tráfico generado por los *wrappers*. Tanto en el *PDCP_RLC_SPLIT* como en el *RLC_MAC_SPLIT*, se ha trabajado con el requisito de latencia *Non-ideal* definido en [32] por el *Small Cell Forum*. Además, el valor de esta latencia coincide en ambos escenarios: 30 `ms`.

5.2. Resultados

Seguidamente, a lo largo de esta sección se presentan los principales resultados que se han obtenido de la ejecución de las pruebas en los escenarios anteriormente presentados. Con el propósito de facilitar la interpretación de los resultados, a continuación se realiza una breve explicación sobre las métricas y representaciones que se van a utilizar:

1. *Valores medios*: Representación de los valores medios de CPU y RAM utilizados por el eNB o la CU y DU en cada uno de los *splits* desplegados (*LEGACY*, *PDCP_RLC* y *RLC_MAC*). Se tratan de resultados del Escenario I, donde se despliega toda la red en un único equipo.

   Respecto al uso de CPU, se realiza la representación para cada valor de PRB y tasa utilizadas en las pruebas. Sin embargo, en el caso de la memoria, únicamente se presenta su evolución respecto al número de PRBs que ofrece la BS. Esto se justifica en que el consumo de memoria es independiente, manteniendo el número de PRBs, de la tasa con la que se envían datos.

2. *Boxplots*: Respecto a este tipo de resultados, se distinguen dos tipos.

   - Para un mismo *split* y un único valor de número de PRB, se compara el comportamiento de la BS (en relación con el consumo de CPU) cuando se despliega la red en cada equipos por separado (Escenario I) o cuando se despliega entre los tres dispositivos (Escenario II)[1].
   - Para cada uno de los dispositivos con los que se han realizado las pruebas (Escenario I), se realiza una comparativa del consumo de CPU de la BS cuando se despliega con los diferentes *splits* para un valor fijo de PRB. Se corresponden con resultados del Escenario I.

3. *Tasas medias*: Para cada uno de los *wrappers* que se encuentran en configuración remota, se evalúa la tasa media de información enviada en cada uno de los *splits* en función del equipo, el número de PRBs y la tasa de tráfico de usuario generada con *iperf*. Son resultados pertenecientes a los Escenarios I y II.

4. *Escenario III*: Se representan los valores medios de uso de CPU y memoria en cada uno de los *splits* cuando no se añade ninguna latencia y cuando se introduce la latencia *Non-Ideal*. En este caso, se trabaja únicamente a las tasas de 15 PRB. Por otro lado, para cada *split* se representa mediante *boxplots* el consumo de CPU para cada tasa tanto en el caso sin latencia como con ella. Además, se realiza una comparativa del tráfico generado por cada *wrapper* para ambas configuraciones de latencia.

En las siguientes secciones, se muestran los resultados obtenidos para cada escenario. Para evitar una explicación tediosa, únicamente se presentarán aquellos más relevantes mientras que el resto de resultados se muestran en el Anexo A.

Por último, es importante destacar que en los resultados de los valores medios y *boxplots* del consumo de memoria y CPU se han adaptado los resultados para permitir compararlos los rendimientos en diferentes

[1] Como se argumentó en la presentación del Escenario II, en este caso solamente se han evaluado los *splits PDCP_RLC* Y *RLC_MAC*.

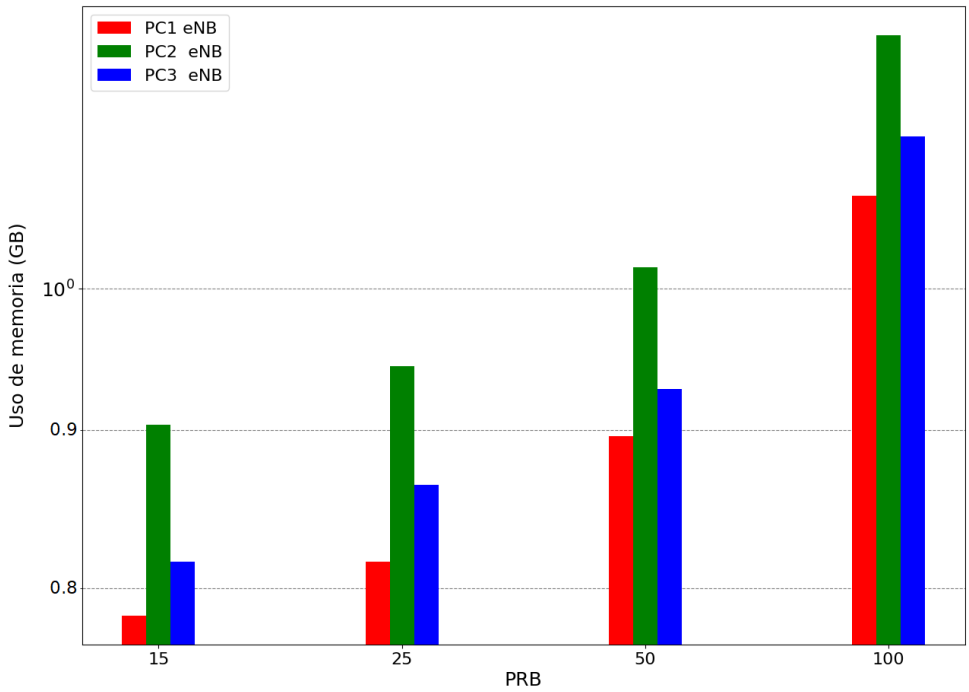


Figura 5.4: Valores medios del uso de memoria de la versión *LEGACY* en el Escenario I en función del dispositivo y del número de PRBs.

equipos. Para ello, se han escalado los % de uso de CPU obtenidos mediante el comando *top* en función de su máxima frecuencia de operación y, en el caso de la memoria, según la capacidad total de memoria disponible. De este modo, se representan los valores absolutos (GB utilizadas y frecuencia de operación del elemento) permitiéndose realizar una comparativa entre los diferentes resultados logrados.

5.2.1. Escenarios I y II

En primer lugar, en las Figuras 5.4, 5.5 y 5.6 se presenta la memoria utilizada por la BS (o sus elementos: CU y DU) en el Escenario I en función del número de PRBs, indicado en el eje de abscisas.

El motivo por el cual no se muestra la variación respecto a la tasa de datos de usuario se debe a que el consumo de memoria es totalmente independiente de ella. Cuando se despliega el elemento (eNB, CU o DU) se reserva estáticamente toda la memoria que utilizará el elemento. Para ello, todos los *buffers* internos que se usan son estáticos y poseen tamaños elevados. Esto provoca que su comportamiento sea ajeno al estado de la red (envío/recepción de tráfico, estado sin envío de tráfico, despliegue de la red o su deshabilitación) y, además, es constante a lo largo del tiempo. Por este último motivo no se representa posteriormente ningún *boxplot* sobre el uso de memoria.

Por otro lado, se puede observar que, para un mismo equipo, a medida que se incrementa el número de PRBs es mayor el uso de memoria que realiza la eNB. Este resultado es coherente, ya que cuanto mayor es el número de recursos físicos del interfaz radio, los *buffer* internos tienen que aumentar de forma proporcional.

Por otro lado, se esperaría que el uso de memoria de los tres equipos sea idéntico para el mismo elemento (eNB, CU o DU). Sin embargo, se observan diferencias entre ellos, tal como muestra la Figura 5.4. Una posible razón es el impacto de otros procesos en los equipos o la falta de exactitud del proceso

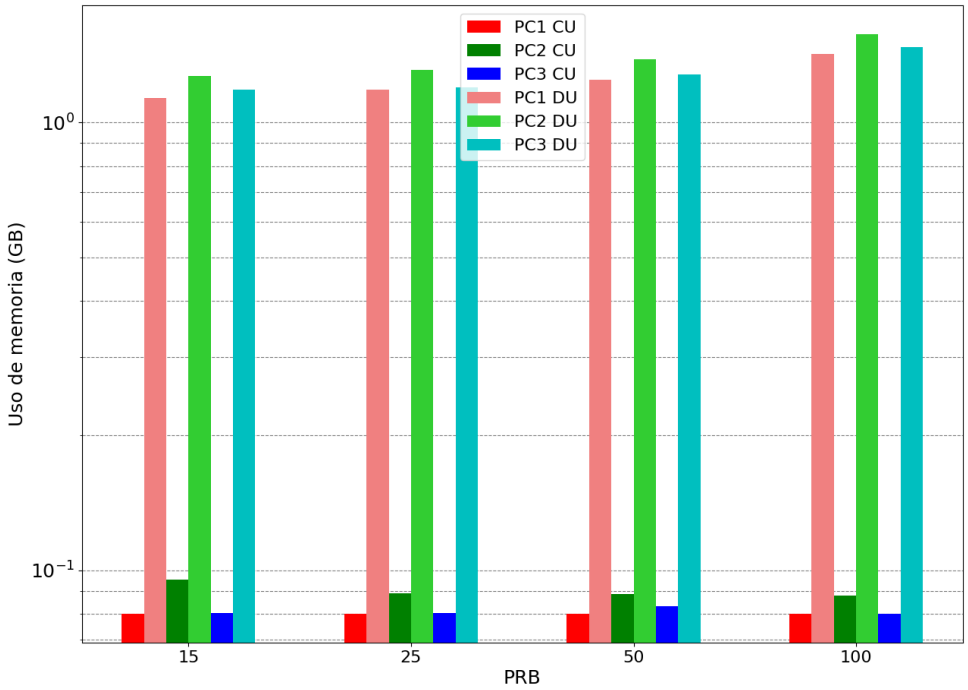


Figura 5.5: Valores medios del uso de memoria en el *split PDCP_RLC* en el Escenario I en función del dispositivo y del número de PRBs.

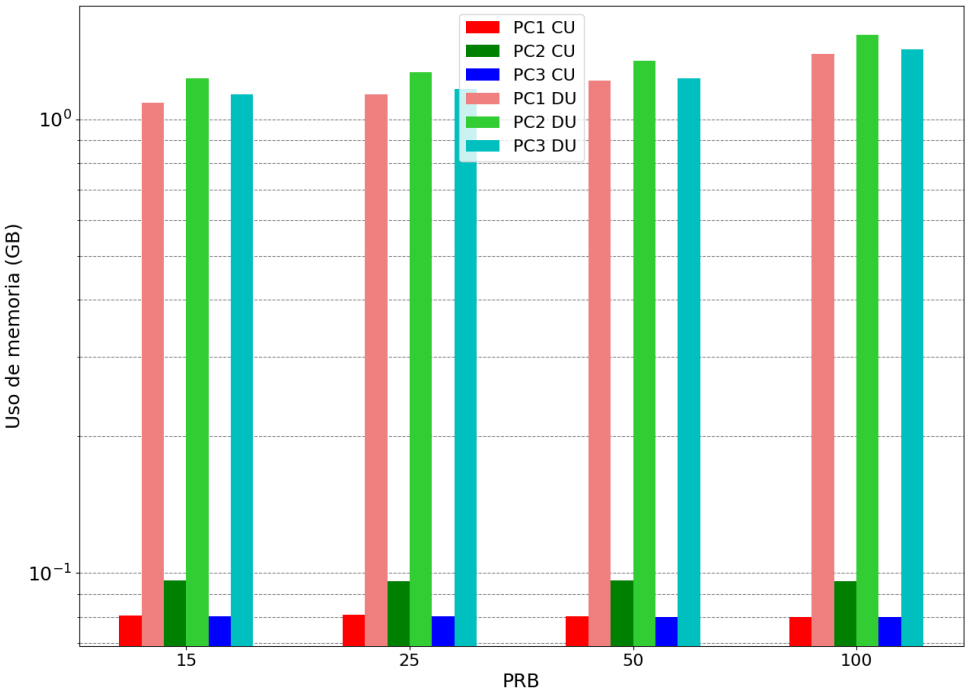


Figura 5.6: Valores medios del uso de memoria en el *split RLC_MAC* en el Escenario I en función del dispositivo y del número de PRBs.

de medida del comando *top*.

Por otro lado, también se observa una clara diferencia entre el consumo de memoria entre la CU y DU. Concretamente, en las Figuras 5.5 y 5.6 se puede observar que el uso de memoria de la CU es claramente inferior, en ambos *splits*, al uso que realiza la DU. Este resultado es coherente y se justifica en qué funcionalidades se realizan en cada elemento:

- En el caso del *split PDCP_RLC*, la mayor parte de las funcionalidades se ejecutan en la DU (PHY, MAC, RLC, RRC y S1-AP), siendo PHY y MAC las dos capas computacionalmente más demandantes. En cuanto a la CU, esta únicamente alberga las capas PDCP y GTP-U están centralizadas.
- En el *split RLC_MAC*, en la CU se realiza, además de las funcionalidades que ya se ejecutaban en el *split PDCP_RLC*, la capa RLC lo que no tiene impacto relevante en cuanto a memoria. El resto de actividades se continúan ejecutando en la DU.

Asimismo, se observa que no hay un gran cambio significativo a nivel del uso de memoria entre ambos *splits*. Esto se debe a que, únicamente se ha centralizado RLC que tiene un bajo consumo computacional y en la DU siguen desarrollándose las funciones PHY y MAC que sí son las costosas computacionalmente. Sin embargo, sí se observa una pequeña variación que es acorde con los resultados esperados: el consumo de la CU se incrementa mientras que el de la DU se reduce cuando se cambia la configuración de la red desde el *PDCP_RLC_SPLIT* hacia *RLC_MAC_SPLIT*. Por ejemplo, en el caso del *PC1* cuando se trabaja a 15 PRBs se aumenta la memoria usada por la CU de 0.08 GB a 0.082 GB y la DU se reduce de 1.136 GB a 1.088 GB.

Seguidamente, en las Figuras 5.7, 5.8 y 5.9, se representa el uso de CPU (en escala logarítmica) empleado por la BS o sus elementos en cada equipo en función del número de PRBs y la tasa con la que el UE envía tráfico en el Escenario I. En la Figura 5.7, se observa que, en el caso del *PC1*, a medida que se incrementa la tasa de envío y/o el número de PRBs, se produce un incremento en el uso de CPU. Este resultado es coherente debido a que, como se ha explicado anteriormente, el aumento de recursos físicos en el interfaz radio implica una mayor demanda computacional. Sin embargo, respecto a los *PC2* y *PC3* se puede comprobar que apenas existe variación en función del número de PRB y tasas. Este hecho se justifica en que, como ambos equipos tienen peores características en términos de CPU que el *PC1*, llegan a saturar y no son capaces de ofrecer la misma calidad en las prestaciones que *PC1*.

En el caso de los *splits*, representados en las Figuras 5.8 y 5.9, se observa que el uso de CPU de la CU es muy inferior al uso que realiza la DU. Este resultado es coherente y su argumentación es idéntica al uso de memoria: en la DU se realizan la mayor parte de las funcionalidades y, además, las más costosas computacionalmente (PHY y MAC). Asimismo, se observa que, para los tres equipos usados, se produce un crecimiento de de uso de CPU (para ambos elementos) a medida que se aumentan los PRBs y la tasa de envío.

Otro aspecto significativo es la comparación de ambos *splits* desde el punto de vista de la CPU: al igual que en el caso de la memoria, se espera que se incremente el uso de CPU en la CU cuando se pasa del *split PDCP_RLC* al *RLC_MAC* y una reducción en la DU, como consecuencia de la centralización de RLC. En el caso de 15, 25 y 50 PRB, se observa este comportamiento: el uso de CPU se rebaja ligeramente en la DU y crece en la CU siendo un resultado coherente a la argumentación anterior. Sin embargo, en

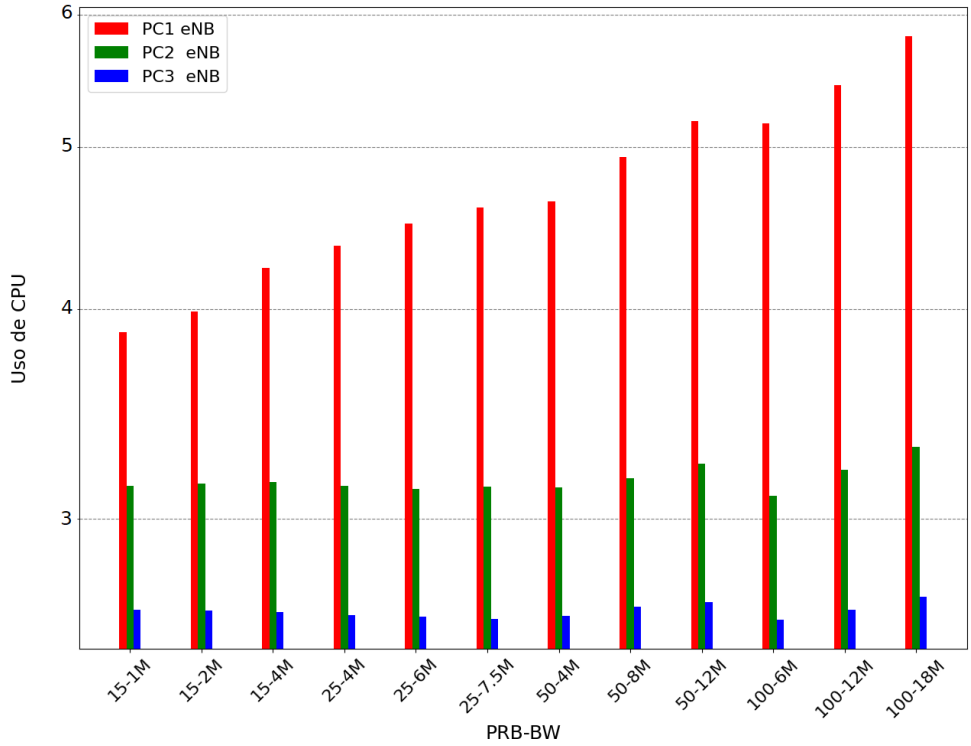


Figura 5.7: Valores medios del consumo de CPU de la versión *LEGACY* en el Escenario I en función del dispositivo, el número de PRBs y la tasa *iperf* de operación.

el caso de 100 PRB se observan unos resultados opuestos a los esperados, ya que la CU presenta mayor consume en el *PDCP_RLC_SPLIT* que en *RLC_MAC_SPLIT*. Por otro lado, el uso de CPU en la DU es muy similar pero para ambas configuraciones de *split*. La razón de este comportamiento contraintuitivo en la CU podría deberse a que el consumo de CPU de la lógica para la separación de RLC/PDCP sea mayor que el propio consumo de RLC. De esta manera, al centralizar RLC el consumo total en la CU podría verse reducido. En cualquier caso, este comportamiento necesitaría un análisis más profundo.

Por último, en la Figura 5.10 se muestra de forma ampliada el consumo de CPU de la CU en el *split PDCP_RLC*. El propósito de esta Figura es mostrar con mayor detalle el consumo que realiza la unidad centralizada. Como se puede observar, la CU tiene un bajo requisito de CPU, lo que permitiría que pueda ser desplegada en un equipo de bajas prestaciones computacionales o en la nube empleando pocos recursos computacionales.

Boxplots - distribuciones y valores atípicos

Seguidamente, se presentan los principales resultados de las distribuciones así como los valores atípicos que sigue el uso de CPU en los elementos de la EB en los Escenarios I y II para las diferentes tasas de 100 PRBs.

En primer lugar, en la Figura 5.11 se observa cómo es el comportamiento de la BS en la versión *LEGACY* en los diferentes equipos del Escenario I. En el eje de abscisas se identifica el dispositivo sobre el que se han realizado las pruebas y, en el eje de ordenadas, se indica el uso de CPU. Asimismo, es importante destacar que, para cada equipo, hay tres *boxplots*. Cada uno de ellos correspondientes a las tres tasas de datos empleadas: 6, 12 y 18 Mbps (de izquierda a derecha).

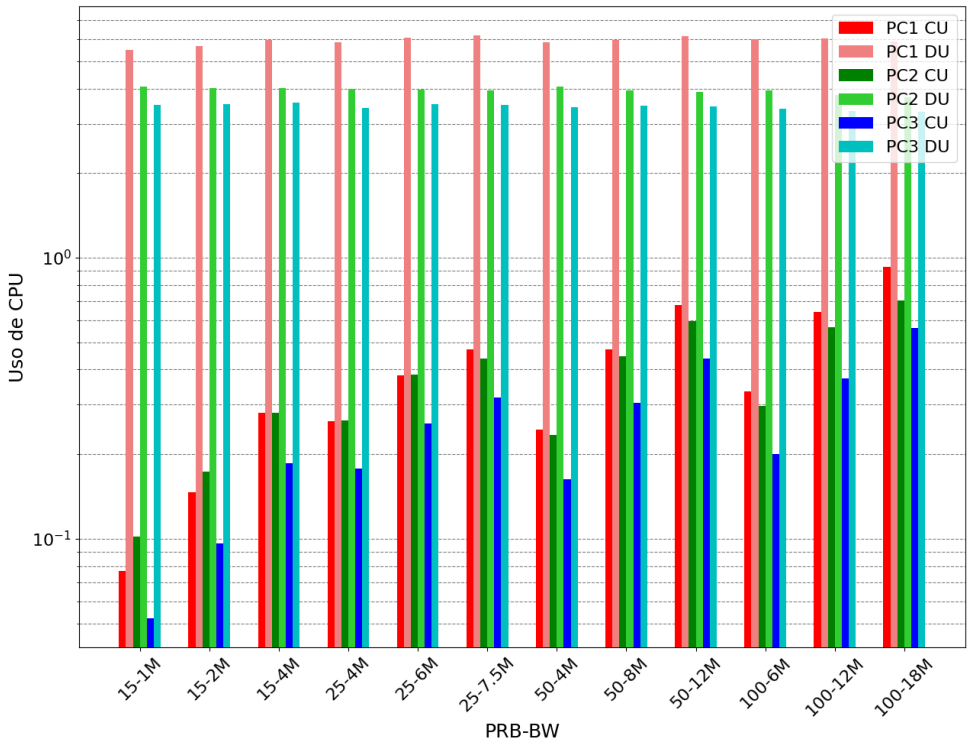


Figura 5.8: Valores medios del consumo de CPU del *split PDCP_RLC* en el Escenario I en función del dispositivo, el número de PRBs y la tasa *iperf* de operación.

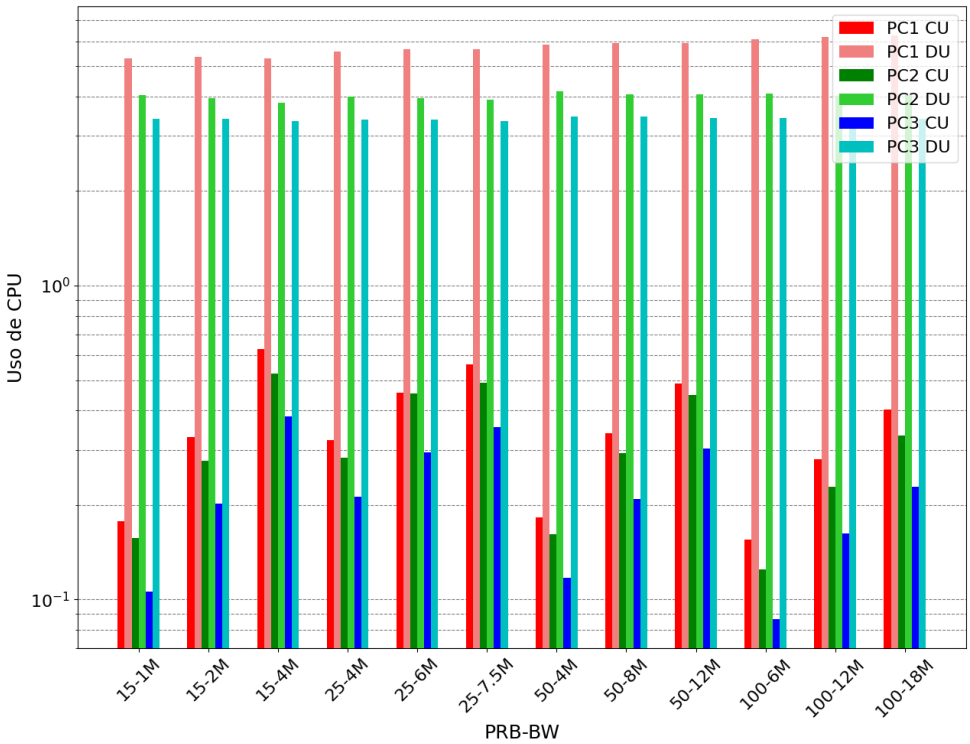


Figura 5.9: Valores medios del consumo de CPU del *split RLC_MAC* en el Escenario I en función del dispositivo, el número de PRBs y la tasa *iperf* de operación.

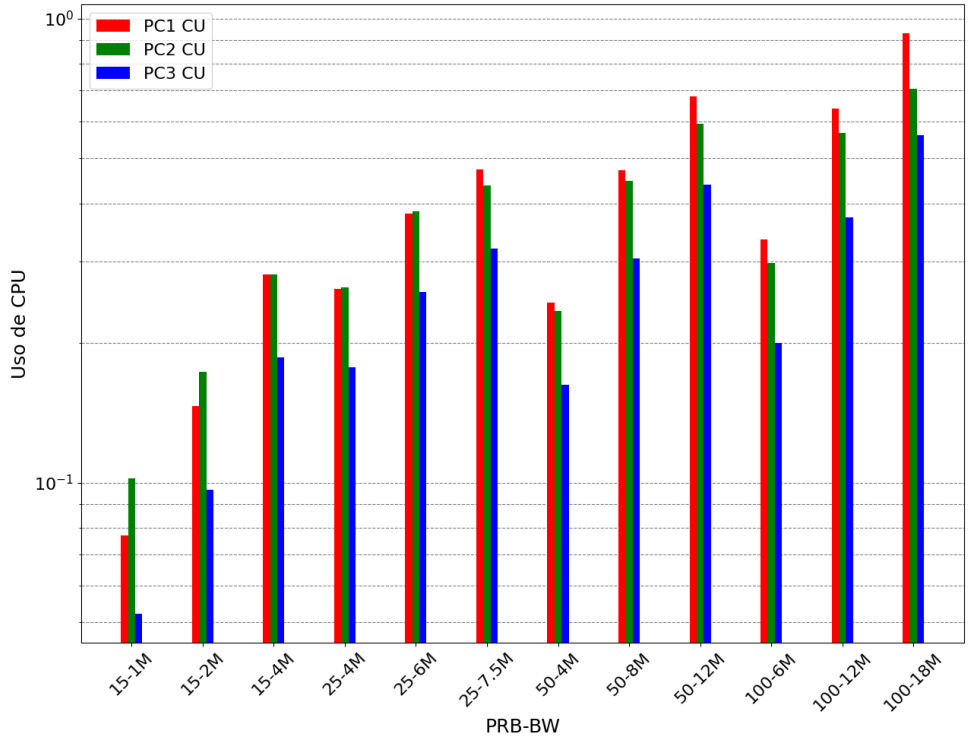


Figura 5.10: Comportamiento de la CU en términos del consumo de CPU en el *split PDCP_RLC* en el Escenario I.

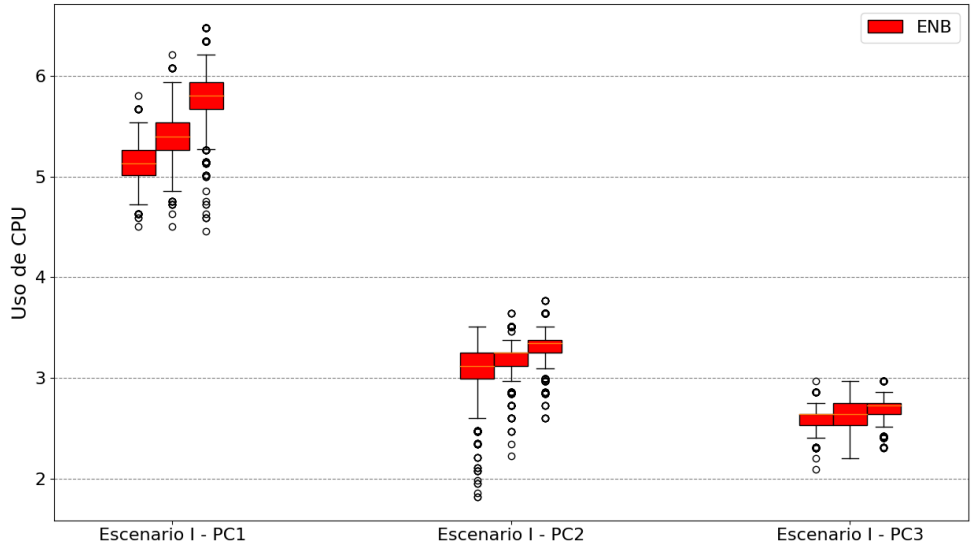


Figura 5.11: Evolución del comportamiento de la eNB a nivel de CPU en cada uno de los dispositivos de prueba para las tasas de 100 PRBs. Cada uno de los *boxplots* de cada uno de los 3 escenarios representados se corresponden (de izquierda a derecha) con las tasas de 6, 12 y 18 Mbps.

Respecto a los resultados observados, se puede concluir que, a medida que se aumenta la tasa, se incrementa el uso de CPU que emplea la implementación. Aunque este suceso se produce en los tres equipos, es más significativo en el *PC1* que en el resto y, además, se vuelve a observar que este dispositivo hace un mayor uso de la CPU. Como se ha mencionado anteriormente, esto se debería a que este equipo tiene mejores prestaciones computacionales y, a diferencia de los *PC2* y *PC3*, no satura.

En las Figuras 5.12 y 5.13, se muestra una comparativa del comportamiento de la implementación

cuando se despliegan los diferentes *splits* en los Escenarios I y II usando las diferentes tasas con la configuración de 100 PRBs. En el eje horizontal, se identifica el escenario en el que se han realizado las pruebas y, en el eje de ordenadas, el uso de CPU. Cada uno de los casos estudiados e identificados en el eje de abscisas está formado por 6 boxplots, donde cada pareja (CU y DU) se corresponde a una tasa concreta. Por ejemplo, en el Escenario I - *PC1*, los primeros boxplots verde y azul (comenzando por la izquierda) se corresponden con la tasa de 6 Mbps, los segundos con 12 Mbps y los terceros con 18 Mbps.

En ambos *splits*, se observa la misma tendencia: *PC1*, debido a sus características computacionales, es aquel que hace un mayor uso de la CPU. Mientras tanto, los equipos *PC2* y *PC3* están significativamente por debajo. El *PC2*, al tener una mayor frecuencia máxima de operación que *PC3*, puede trabajar más rápido. Por último, se presenta cómo es el consumo cuando la red se despliega en el Escenario II, Cabe recordar que en este escenario el EPC y la CU son desplegados por *PC1*, la DU en *PC3* y, por último, el usuario en *PC2*.

En el Escenario II, tanto para el *PDCP_RLC_SPLIT* como para *RLC_MAC_SPLIT*, se observa un crecimiento computacional de la CU al aumentar la tasa de datos de usuario mientras que, en el caso de la DU del *PDCP_RLC_SPLIT*, se mantiene prácticamente invariante a la variación de la tasa. En el caso de la CU, el comportamiento es coherente y, respecto a la DU, únicamente tiene un comportamiento intuitivo en el *split MAC_RLC_SPLIT*.

Al conmutar del Escenario I al II, el *PC1* deja de desplegar toda la red y únicamente se responsabiliza de la EPC y CU. Respecto al *PC3*, deja de desplegar la red completamente para, únicamente, encargarse de la DU. Por este motivo, ambos equipos pueden dedicar más recursos a estos elementos, lo que se traduce en los resultados observados.

Por otro lado, comparando ambos *splits*, se observa que cuando se cambia del *PDCP_RLC_SPLIT* al *RLC_MAC_SPLIT* el uso de CPU se reduce tanto en la CU como en la DU. Este resultado tiene coherencia en el caso de la DU, ya que necesita menos recursos para conseguir las mismas prestaciones (en ese cambio de *split* se centraliza *RLC*). Sin embargo, respecto a la CU, este comportamiento debería ser diferente, ya que al tener una mayor responsabilidad, precisa más recursos computacionales. Además, se descarta la opción de que sature en esta situación, ya que se observa un crecimiento del uso de CPU a medida que se aumenta la tasa. Como se indicó en resultados anteriores, se requeriría un análisis en mayor profundidad para explicar este comportamiento en el uso de CPU.

Tasas de tráfico de los *wrappers*

Sobre los Escenarios I y II también se ha realizado la monitorización del tráfico que envían los *wrappers* con configuración remota a través de sus *sockets*. Como se comentó anteriormente, en la DU se han establecido las comunicaciones de las capas (excepto PHY) con RRC mediante *sockets* aunque estén situados en el mismo dispositivo.

Por otro lado, es importante recalcar que el tráfico enviado en las pruebas es *Uplink*, es decir, se lo envía el usuario móvil a la BS. Esto implica que, el tráfico recibido por la estación base (correspondiente a los datos), atravesará BS desde la capa PHY hasta PDCP. Respecto al tráfico de control, este se intercambia principalmente entre las diferentes capas y RRC, o entre capas adyacentes.

En primer lugar, en las Figuras 5.15 y 5.16 se presentan los resultados de los Escenarios I y II sobre el

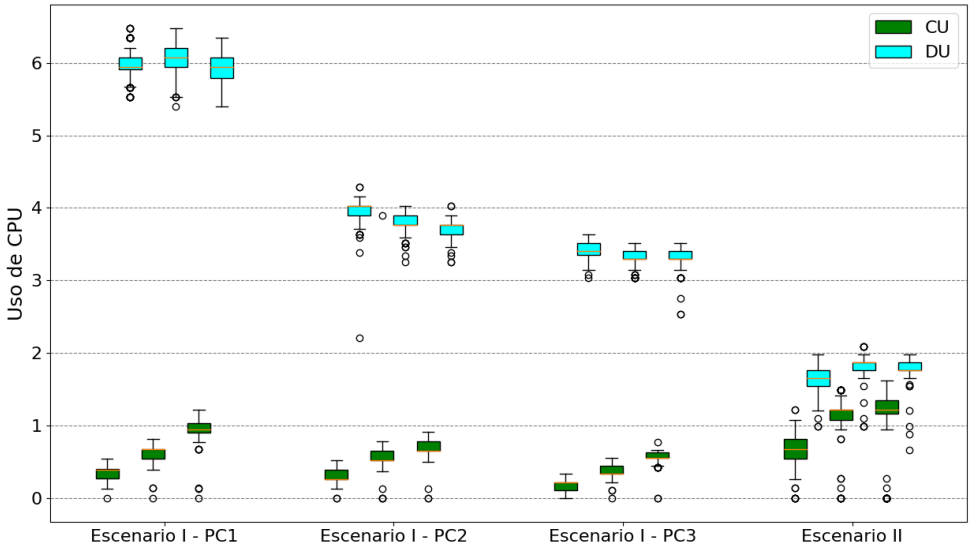


Figura 5.12: Evolución del comportamiento de la CU-DU del *split PDCP_RLC* a nivel de CPU en los Escenarios I y II para las tasas de 100 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 6 Mbps, los segundos con 12 Mbps y los terceros con 18 Mbps.

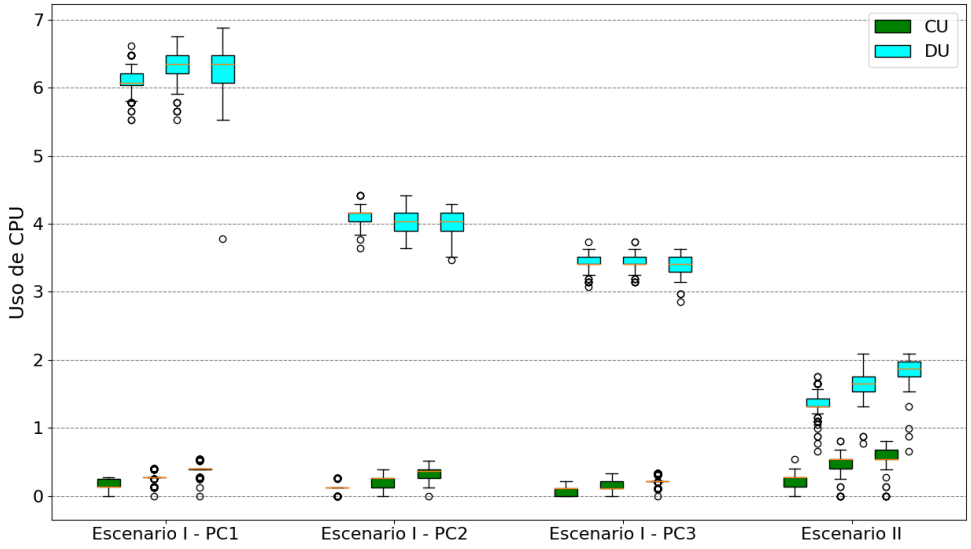


Figura 5.13: Evolución del comportamiento de la CU-DU del *split RLC_MAC* a nivel de CPU en los Escenarios I y II para las tasas de 100 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 6 Mbps, los segundos con 12 Mbps y los terceros con 18 Mbps.

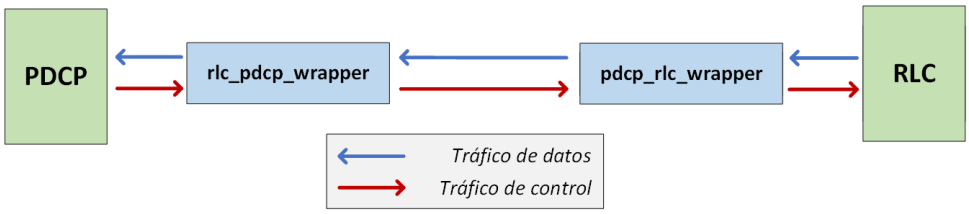


Figura 5.14: Esquema de la comunicación PDCP $\longleftrightarrow$ RLC en el envío de tráfico UL en el *PDCP_RLC_SPLIT*.

despliegue de la red con el *split PDCP_RLC*. En el eje de abscisas se representa la dupla PRB-tasa y, en el eje de ordenadas, la tasa binaria con la que el *wrapper* envía la información al otro *wrapper*. A modo de aclaración, en la Figura 5.14, se presentan de forma esquemática las capas y *wrappers* que atraviesan los datos en este caso.

El *pdcp_rlc_wrapper* es aquel que envía (dentro de la BS) los datos de usuario, por lo que, como se puede ver en la Figura 5.15 su tasa binaria es del orden de los Mbps (magnitud a la que el usuario envía los datos) mientras que *rlc_pdcp_wrapper* genera el tráfico de control en la comunicación. Por ese motivo, su velocidad de envío de información es de kbps.

Como se puede observar, respecto a los resultados del Escenario I, la tasa de envío de datos es muy similar entre los tres equipos, tal y como era de esperar. Únicamente, es destacable que, en algunos casos (se observa con claridad cuando la red tiene 100 PRBs y se envían datos a 18 Mbps), el *PC2* es incapaz de enviar a las mismas tasas que el resto de equipos. Este comportamiento se debería a que sus capacidades computacionales son más limitadas que las del *PC1* y *PC3*, fundamentalmente en relación con la memoria RAM (véase la Tabla 5.1).

Cuando el *split* se despliega entre varios equipos (Escenario II), se observa una reducción en el volumen de tráfico respecto a los resultados del Escenario I. Este hecho se hace más notable a medida que se aumenta el número de PRBs y/o la tasa de envío. Una posible razón de este fenómeno es la limitación computacional del equipo *PC3* (es el que despliega la DU) que es incapaz de operar lo suficientemente rápido para poder procesar todo el tráfico y enviarlo a través del *wrapper*. Este bajo rendimiento también se puede deber a que el *switch* haga de cuello de botella en la comunicación entre el UE y la DU. Aunque se usa el emulador de enlace basado en zeroMQ par implementar el interfaz radio, la cantidad de tráfico que se envía es muy alta y únicamente depende del número de PRB[2].

Seguidamente se presentan los resultados de la medida de tráfico en los *wrappers* remotos en los Escenarios I y II en el split *RLC_MAC*. Al igual que en el caso anterior, en la Figura 5.17 se representa de forma gráfica el flujo de *wrappers* y capas que atraviesan los datos.

Respecto a los resultados obtenidos, representados en las Figuras 5.18 y 5.19, se observa un comportamiento muy similar al visto previamente. En el Escenario I, en los tres equipos se consigue enviar prácticamente a la misma tasa aunque, en el caso del *PC2*, es ligeramente inferior debido a sus características. En el Escenario II, sucede la misma situación que en el *split PDCP_RLC*: al separarse los equipos se acentúa en mayor medida las limitaciones (frecuencia de operación) del equipo *PC3*, donde se implementa la DU.

[2]Se envían todos los símbolos en fase y cuadratura, independientemente de que transporten datos o no.

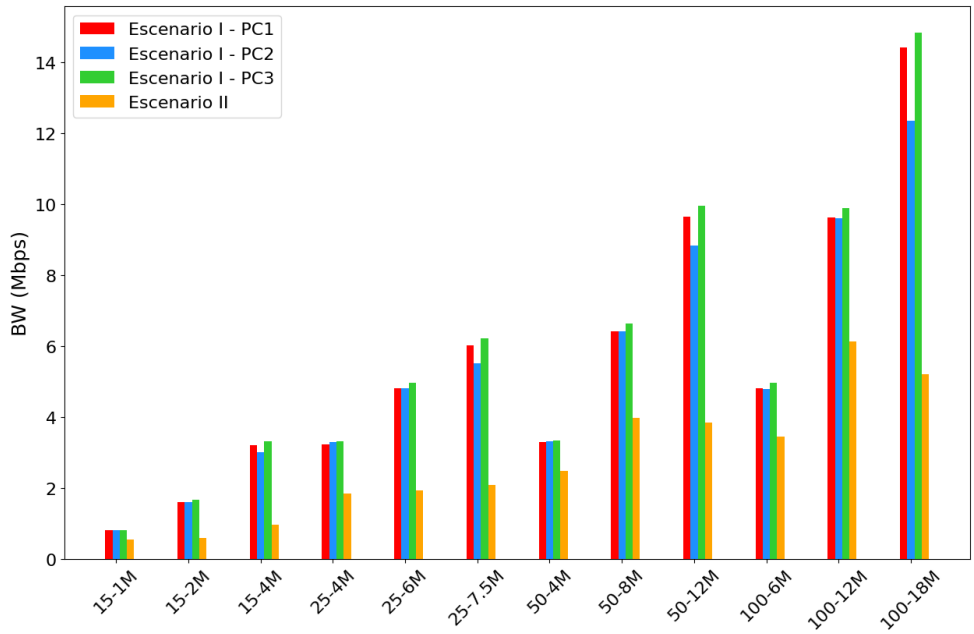


Figura 5.15: Tráfico enviado por el *pdcp_rlc_wrapper* en el *split PDCP_RLC* en los Escenarios I y II en función del equipo, el número de PRBs y la tasa *iperf*.

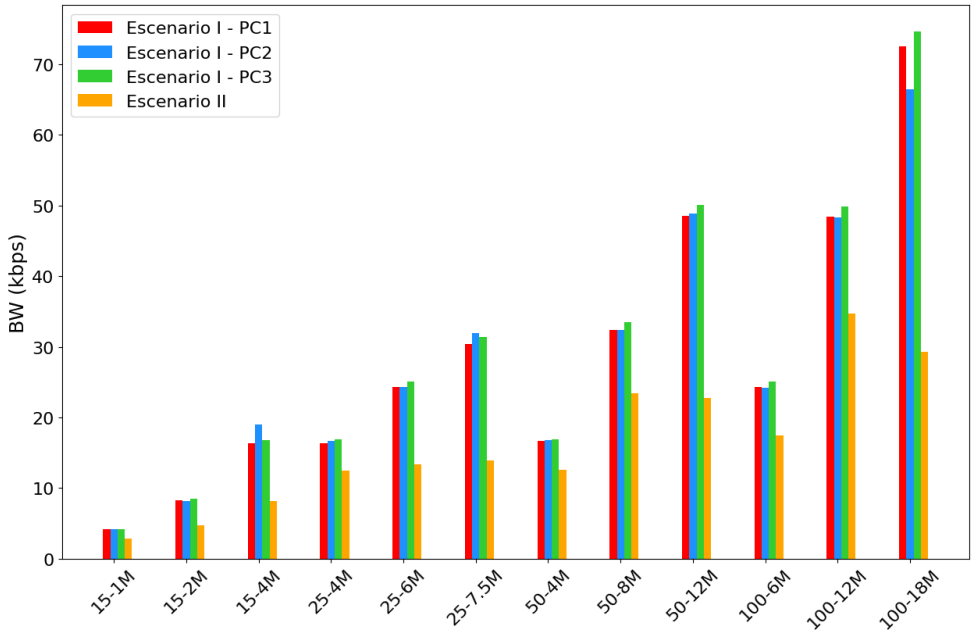


Figura 5.16: Tráfico enviado por el wrapper *rlc_pdcp_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*.

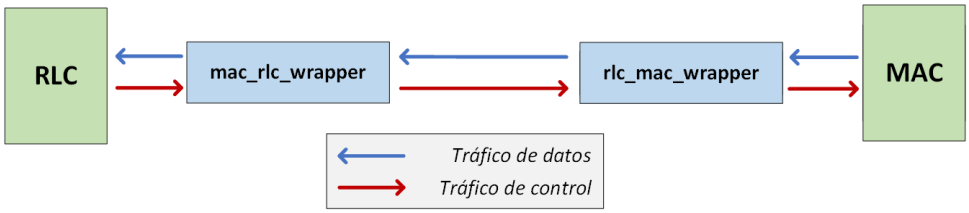


Figura 5.17: Esquema de la comunicación RLC $\longleftrightarrow$ MAC en el envío de tráfico UL en el *RLC_MAC_SPLIT*.

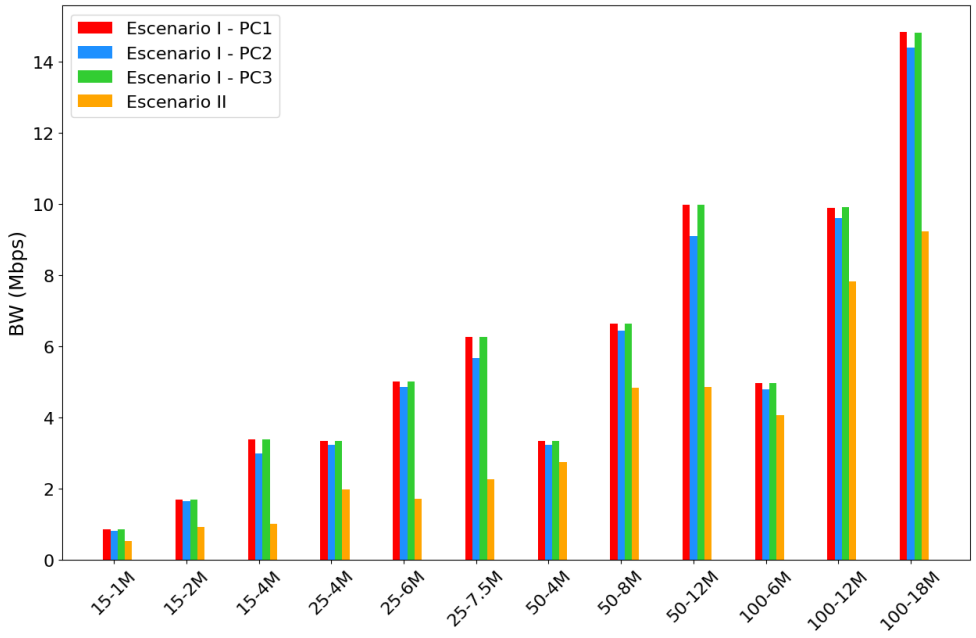


Figura 5.18: Tráfico enviado por el wrapper *rlc_mac* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

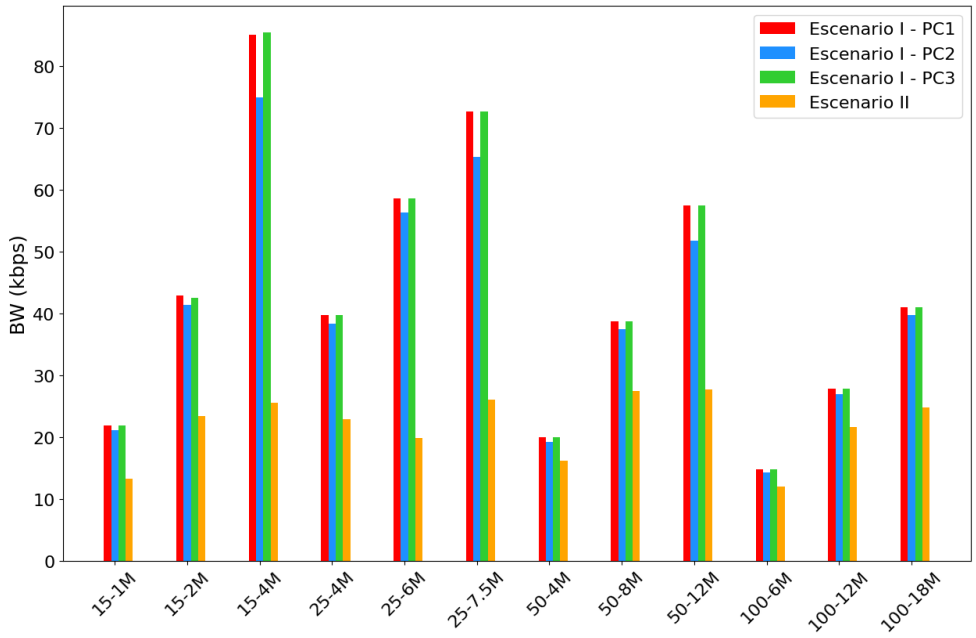


Figura 5.19: Tráfico enviado por el wrapper *mac_rlc* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

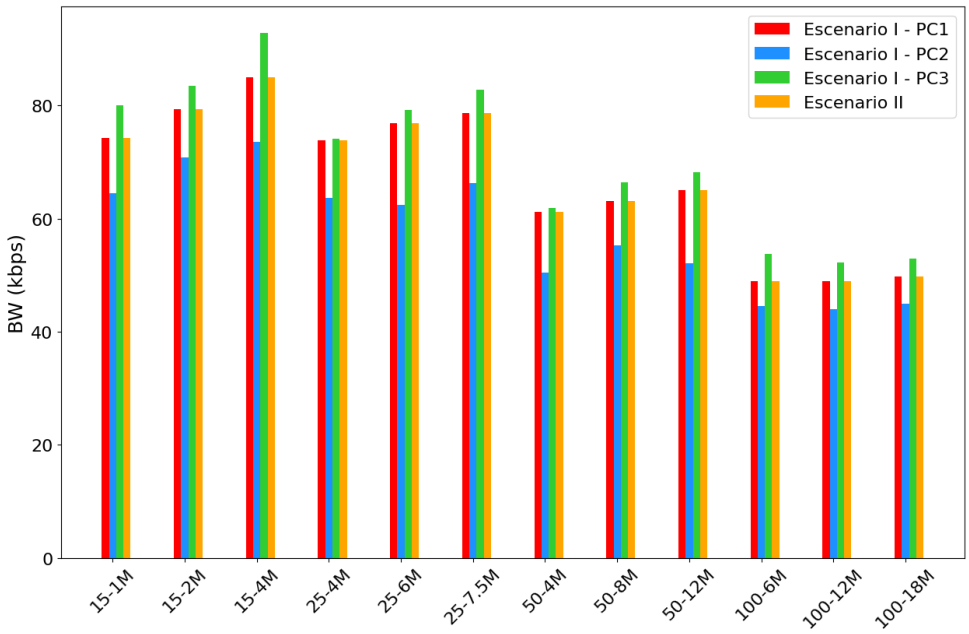


Figura 5.20: Tráfico enviado por el wrapper *rrc_mac* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

Por último, en la Figura 5.20, se presenta el tráfico enviado por el *wrapper rrc_mac* en el *split RLC_MAC*. Debido a la implementación realizada, donde no se ha logrado separar totalmente RRC del todas las capas, ambas capas se encuentran en la DU cuando realmente deberían estar separadas.

Como se puede observar, respecto a los resultados del Escenario I, en el *PC2* se presentan los peores resultados siendo incapaz de igualar las tasas que alcanzan el resto de dispositivos. En cuanto al Escenario II, se comprueba que no se produce una caída de la tasa respecto a los resultados del Escenario I. Esto se debería a que esta es una comunicación se da de forma local y la carga de tráfico es baja.

5.2.2. Escenario III: comprobación del cumplimiento de latencia

Respecto a los resultados del Escenario III, se presentan únicamente para el caso del *PDCP_RLC_SPLIT* debido a que son muy semejantes a los obtenidos en el *RLC_MAC_SPLIT* y la inclusión de ambos no aportaría información adicional. En cuanto a los resultados expuestos, los valores medios y *boxplots* se han obtenido enviado tráfico *UL* mientras que el análisis del tráfico en los *wrappers* se ha realizado tanto enviando tráfico UL como DL.

En la Figura 5.21, se presentan los valores medios del uso de memoria que precisan las unidades centralizada y distribuida en el *split PDCP_RLC* con y sin retardo en el enlace de *fronthaul*, para las tasas de 15 PRBs. En este sentido, el comportamiento de la red permanece inalterado ante la presencia o no de latencias. Esto se debe a que, como se explicó anteriormente, la implementación reserva estáticamente la memoria que utilizará en su funcionalidad y es independiente del estado de la red.

Sin embargo, como puede comprobarse en la Figura 5.22, el comportamiento en relación al consumo de CPU es muy diferente. En el caso sin *delay* (representados en tonos rojos) se observa un comportamiento coherente, donde el uso crece a medida que se aumenta la tasa de tráfico. Por otro lado, cuando se introduce la latencia el rendimiento de la red se reduce de forma acusada. Esto se debe a que estos retrasos

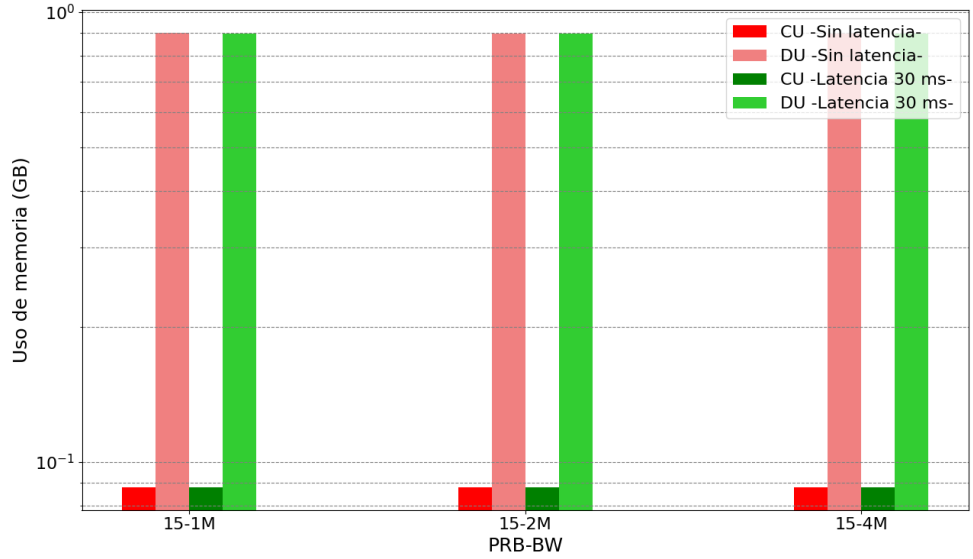


Figura 5.21: Valores medios del uso de memoria de la CU-DU en el *split PDCP_RLC* tanto en el caso sin latencia como en el caso en el que se añade la latencia de 30 ms mientras se envía tráfico UL.

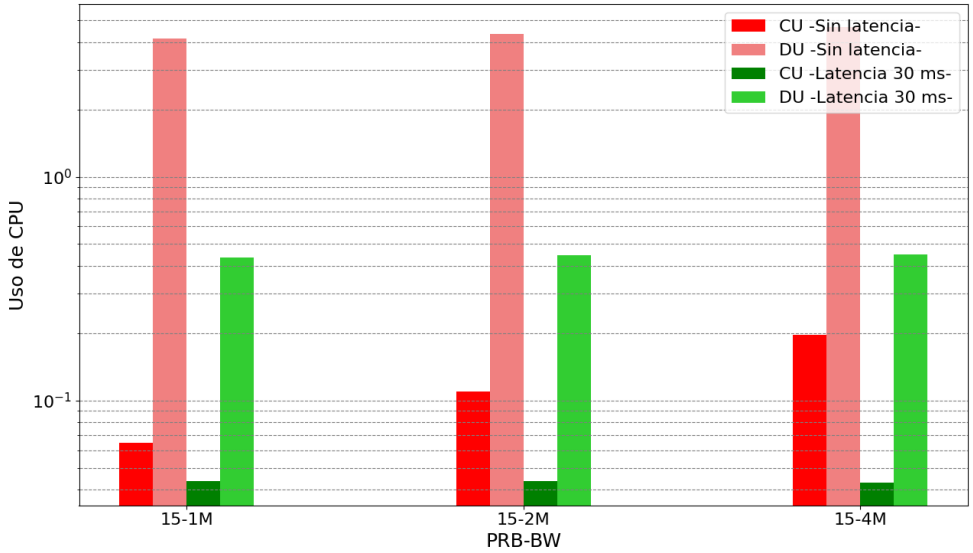


Figura 5.22: Valores medios del consumo de CPU de la CU-DU en el *split PDCP_RLC* tanto en el caso sin latencia como en el caso en el que se añade la latencia de 30 ms mientras se envía tráfico UL.

provocan que se pierda el sincronismo entre la CU y DU que se traduce en una pérdida importante de las prestaciones. Además, se observa que se produce una saturación en la red, es decir, que el uso de CPU y, por tanto, las prestaciones que ofrece no crecen a medida que se aumenta la tasa.

En la Figura 5.23 se presenta la variación de la CU y DU a las tasas de 1, 2 y 4 Mbps de 15 PRBs. Al igual que sucede con los valores medios, cuando se introduce la latencia se produce una caída del uso de CPU consecuencia de la pérdida de sincronismo y que provoca unas menores prestaciones de la red. En este sentido, esta caída computacional es más evidente en la DU, ya que es aquella que usa una mayor cantidad de recursos computacionales. Además, se observa también que, cuando se introducen los 30 `ms` de latencia, el uso de CPU no varía, independientemente de que se incremente la tasa con la que se envía el tráfico.

Por último, se presentan los resultados de la monitorización del tráfico en los *wrappers*. En este caso,

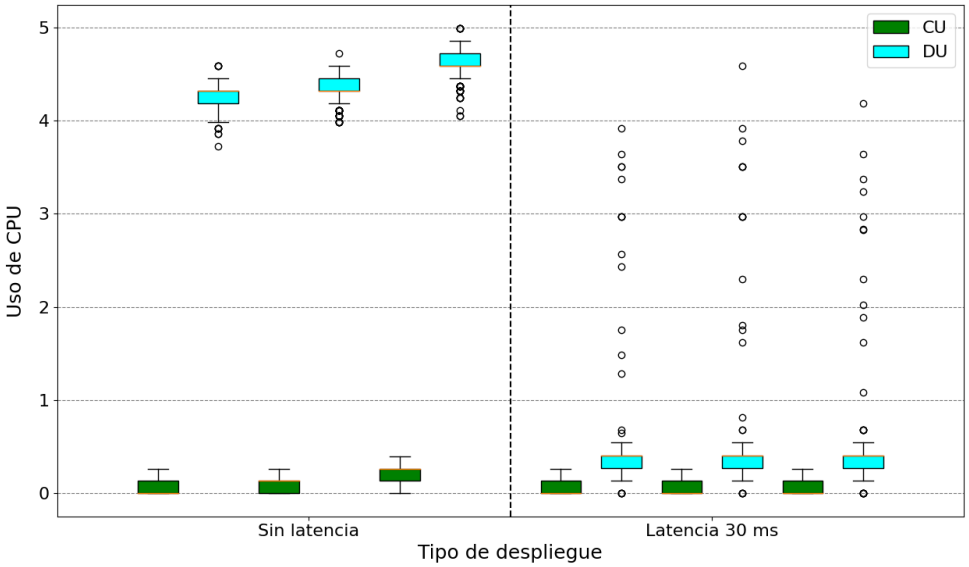


Figura 5.23: Comportamiento de la CU-DU en el *split PDCP_RLC* para cada una de las tasas de 15 PRBs en función de la existencia o no de *latencia* adicional en el *fronthaul* mientras se envía tráfico UL. Para los dos supuestos (sin y con latencia), la primera pareja (verde y azul) de boxplots se corresponden con 1 Mbps, la segunda con 2 Mbps y la tercera con 4 Mbps.

se ha enviado tanto tráfico UL como DL. Sin embargo, para evitar una explicación repetitiva solamente se muestran aquellos correspondientes al envío de tráfico UL en el *PDCP_RLC_SPLIT*, mientras que los correspondientes al DL se encuentran en el Anexo A. En las Figura 5.24, se presenta el tráfico que envía el *pdcp_rlc_wrapper* para cada uno de las tasas de 15 PRBs. Como ya se explicó anteriormente, este *wrapper* es el responsable de enviar los datos *iperf* al otro extremo. Por ese motivo, opera a mayor velocidad que el *rlc_pdcp_wrapper*, representado en la Figura 5.25.

En ambos *wrappers*, se observa un comportamiento esperado. Cuando se introduce la latencia, se produce una caída en el tráfico que se envía por cada *wrapper* como consecuencia de la pérdida de sincronismo que provoca la reducción de carga computacional observada anteriormente. Como no se consigue mantener la tasa a la que se envía el tráfico en una situación ideal, las prestaciones que ofrece la red son inferiores y, por lo tanto, la tasa efectiva a la que puede operar el usuario es menor.

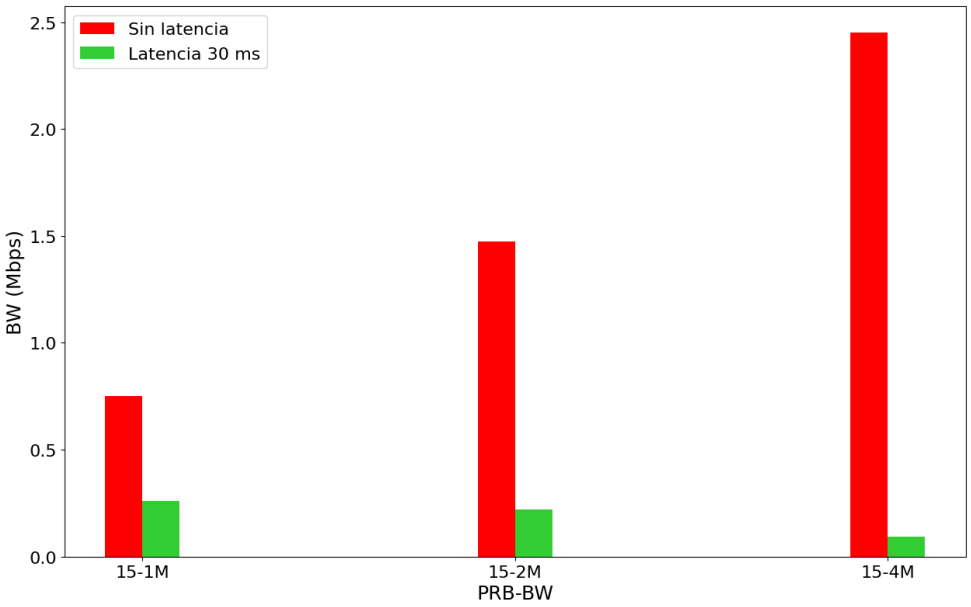


Figura 5.24: Tráfico enviado por *pdcp_rlc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL

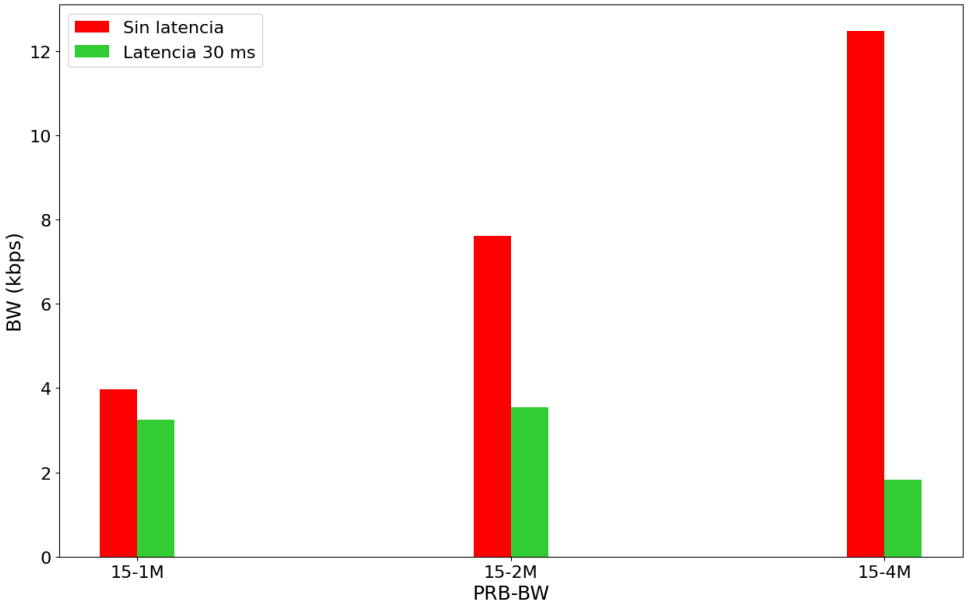


Figura 5.25: Tráfico enviado por *rlc_pdcp_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL

Capítulo 6

Conclusiones

En primer lugar, se puede decir que se han completado de manera exitosa los objetivos planteados inicialmente, al comenzar el trabajo. Se ha diseño y desarrollado un conjunto de elementos que permiten desplegar soluciones de *functional split*, habilitando la posibilidad de repartir las funciones de red entre varios nodos. A continuación se resumen las conclusiones más relevantes obtenidas a partir de la caracterización que se ha llevado a cabo.

Respecto al uso de memoria, se observa un comportamiento constante e independiente del estado de la red. Así, se percibe un crecimiento a medida que se incrementa el número de PRB, pero permanece invariante antes los aumentos de la tasa. Este comportamiento se debe a que la memoria se reserva de forma estática en la inicialización de los elementos. Cuando se divide la BS en la CU y DU, el uso de memoria de la CU es inferior al de la DU en los *splits* analizados. Este resultado es coherente, ya que la DU es la responsable de las capas más costosas computacionalmente: PHY y MAC.

En cuanto a la CPU, en el *PC1* se incrementa su uso a medida que se aumenta la tasa y/o el número de recursos físicos. En el caso de los *PC2* y *PC3* no se produce ese aumento, independientemente de los incrementos de PRBs y tasas, debido a que estos equipos saturan. En el Escenario II, la DU tiene un mayor uso computacional que la CU en ambos *splits* y su justificación es idéntica al uso de memoria: la DU despliega los niveles del *stack* que en mayor medida emplean los recursos computacionales (PHY y MAC).

Por otro lado, cuando se conmuta de *PDCP_RLC_SPLIT* a *RLC_MAC_SPLIT*, se observa un comportamiento intuitivo cuando el número de PRBs es bajo: la CU hace un mayor uso de la CPU, mientras que en la DU se reduce el consumo. Sin embargo, en la configuración con 100 PRBs se observa que el uso de la CU utiliza más CPU en el *split PDCP_RLC* que en *RLC_MAC*. Este comportamiento podría resultar sorprendente, pero se puede deber a que la separación sea más costosa que el despliegue de la capa RLC en sí. De todos modos, esta evidencia requiere un análisis más exhaustivo.

Las medidas de tráfico realizadas permiten concluir que el rendimiento de los *PC1* y *PC3* es muy similar, mientras que el *PC2*, debido a sus características, es más limitado. Cuando se despliega la red en el Escenario II, se observa una reducción del tráfico generado, más apreciable a medida que se incrementa la tasa y el número de recursos físicos. Este comportamiento se debe a la incapacidad del *PC3* para procesar y enviar el tráfico lo suficientemente rápido. Además, otro factor que puede causar esta limitación es el cuello de botella que el *switch* pueda realizar a la comunicación DU-UE.

Por último, respecto al Escenario III se observa un comportamiento invariante a la inclusión de retardos en términos del uso de memoria. Este resultado es coherente, ya que el uso de memoria es independiente del estado de la red y, como se ha comentado anteriormente, se reserva estáticamente en el despliegue inicial. En cuanto al uso de CPU, cuando se introduce latencia, se produce una caída acusada del uso, dando lugar a una saturación de la red. Este hecho se justifica en la pérdida de sincronismo que generan los *retardos* y provoca una reducción de las prestaciones de la red. Esta pérdida de sincronismo también tiene su impacto en el tráfico enviado por los *wrappers*, provocando una reducción considerable de la tasa efectiva a la que pueden transmitir los usuarios.

6.1. Líneas futuras

Como se ha comentado anteriormente, en este proyecto se ha realizado la validación de los *splits* implementados en varios escenarios. Las tareas que se han acometido en el marco de este trabajo dan lugar a varias líneas de investigación futuras, donde destacan:

- Finalización del desarrollo y validación de los *RRC_PDCP_SPLIT* y *MAC_PHY_SPLIT* acabando las separaciones de la capa PHY con MAC y RRC.
- Evaluación de los *splits* utilizando equipamiento con mejores prestaciones, con el propósito de identificar los cuellos de botella (cómputo y/o comunicaciones) en cada configuración.
- Extensión de la implementación realizada sobre la implementación 5G de *srsRAN* que se ha publicado durante el desarrollo del proyecto.
- Relacionado con el punto anterior, se puede plantear el despliegue de redes 4G/5G donde las capas superiores sean compartidas por ambas. De este modo, por ejemplo, RRC puede ser un elemento común, permitiendo la conectividad dual de 5G y LTE.
- Implementación y validación del concepto de *split dinámico*, que permita cambiar el *split* en tiempo de ejecución, es decir, sin la necesidad de deshabilitar la red. Este objetivo no solo implica la definición de un interfaz que permita interactuar con la CU y DU para desplegar las capas del *stack* en un elemento u otro, sino que también se ha de enviar el estado de un extremo a otro, lo que aumenta la complejidad.
- Estudio de la viabilidad de acometer las separaciones intracapa de PHY, MAC y RLC, ya que en estos casos no se dispone de la estructura de interfaces expuesto, y es necesaria la realización de modificaciones internas en los protocolos.

Bibliografía

[1] A. Lometti y V. Sestito. "Fronthaul in 5G Transport Networks: IEEE1914.1 Architecture and Requirements". En: *2020 22nd International Conference on Transparent Optical Networks (ICTON)*. 2020, págs. 1-4. DOI: 10.1109/ICTON51198.2020.9203468.

[2] C. Song, M. Zhang, Y. Zhan, D. Wang, L. Guan, W. Liu, L. Zhang y S. Xu. "Hierarchical edge cloud enabling network slicing for 5G optical fronthaul". En: *Journal of Optical Communications and Networking* 11.4 (2019), B60-B70. DOI: 10.1364/JOCN.11.000B60.

[3] *Open RAN 101–RU, DU, CU: Why, what, how, when? (Reader Forum)*. en-US. 2020. URL: https://www.rcrwireless.com/20200708/open_ran/open-ran-101-ru-du-cu-reader-forum (visitado 26-06-2022).

[4] T. Li, L. Zhao, F. Song y C. Pan. "OAI-based End-to-End Network Slicing". En: *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*. 2018, págs. 1-4. DOI: 10.1109/ICDSP.2018.8631616.

[5] A. Esmaeily, K. Kralevska y D. Gligoroski. "A Cloud-based SDN/NFV Testbed for End-to-End Network Slicing in 4G/5G". En: *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. 2020, págs. 29-35. DOI: 10.1109/NetSoft48620.2020.9165419.

[6] Rajiv. *Evolution of wireless technologies 1G to 5G in mobile communication*. en-US. 2018. URL: https://www.rfpage.com/evolution-of-wireless-technologies-1g-to-5g-in-mobile-communication/ (visitado 05-07-2022).

[7] R. Sood y A. Garg. "Digital Society from 1G to 5G: A Comparative Study". En: *International Journal of Application or Innovation in Engineering & Management* Volume 3 (2014), págs. 186-193.

[8] R. Agüero. "Dimensionamiento de sistemas celulares". En: *Universidad de Cantabria* Redes de Comunicaciones (2022). URL: https://www.tlmat.unican.es/siteadmin/submaterials/3557.pdf.

[9] Keenan, Martin. *The evolution of cellular networks*. en-US. 2020. URL: https://www.avnet.com/wps/portal/abacus/resources/article/the-evolution-of-cellular-networks/ (visitado 05-07-2022).

[10] *An Introduction to 1G, 2G, 3G, 4G & 5G Wireless*. en. URL: https://www.lifewire.com/1g-vs-2g-vs-2-5g-vs-3g-vs-4g-578681 (visitado 06-07-2022).

[11] *About mobile technology and IMT-2000*. 2011. URL: https://www.itu.int/osg/spu/imt-2000/technology.html (visitado 06-07-2022).

[12] *HSPA and HSPA+ for 3G Networks*. en. URL: https://www.lifewire.com/high-speed-packet-access-817467 (visitado 06-07-2022).

[13] S. Dahmen-Lhuissier. *ETSI - 4G - Long Term Evolution | LTE Standards*. en-gb. URL: https://www.etsi.org/technologies/mobile/4G (visitado 12-07-2022).

[14] *What is LTE?* en. URL: https://www.techtarget.com/searchmobilecomputing/definition/Long-Term-Evolution-LTE (visitado 12-07-2022).

[15] *LTE And LTE Advanced Frequency Bands And Spectrum Allocations*. en-US. 2012. URL: https://www.3glteinfo.com/lte-and-lte-advanced-frequency-bands-and-spectrum-aloactions/ (visitado 12-07-2022).

[16] *¿Qué significa que existan dos tipos de 5G? Diferencias y compatibilidades entre 5G NSA y 5G SA*. es. 2019. URL: https://www.xataka.com/moviles/que-significa-que-existan-dos-tipos-5g-diferencias-compatibilidades-5g-nsa-5g-sa (visitado 14-07-2022).

[17] C. Collado. *Todas las diferencias y compatibilidades entre el 5G NSA y el 5G SA*. es. URL: https://andro4all.com/tecnologia/todas-las-diferencias-y-compatibilidades-entre-el-5g-nsa-y-el-5g-sa (visitado 14-07-2022).

[18] S. Dahmen-Lhuissier. *ETSI - 5g Standards - 5g Mobile Technologies*. en-gb. URL: https://www.etsi.org/technologies/mobile/5g (visitado 14-07-2022).

[19] *Release 15 3GPP*. URL: https://www.3gpp.org/release-15 (visitado 14-07-2022).

[20] R. Dilli. "Analysis of 5G Wireless Systems in FR1 and FR2 Frequency Bands". En: *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. 2020, págs. 767-772. DOI: 10.1109/ICIMIA48430.2020.9074973.

[21] *Release 16 3GPP*. URL: https://www.3gpp.org/release-16 (visitado 14-07-2022).

[22] Nokia. *Nokia 5G Core (5GC)*. publisher: Nokia. URL: https://www.nokia.com/networks/5g-core/.

[23] *M.2083: IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond*. URL: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf (visitado 17-07-2022).

[24] *An Introduction to Network Slicing*. Inf. téc. UK: GSMA, 0201. URL: https://www.gsma.com/futurenetworks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf.

[25] Á. Gabilondo, Z. Fernández, R. Viola, Á. Martín, M. Zorrilla, P. Angueira y J. Montalbán. "Traffic Classification for Network Slicing in Mobile Networks". en. En: *Electronics* 11.7 (2022), pág. 1097. DOI: 10.3390/electronics11071097. URL: https://www.mdpi.com/2079-9292/11/7/1097 (visitado 17-07-2022).

[26] *What is C-RAN? The Evolution From D-RAN to C-RAN*. URL: https://www.ufispace.com/company/blog/what-is-cran-the-evolution-from-dran-to-cran (visitado 18-07-2022).

[27] Z. Guizani y N. Hamdi. "CRAN, H-CRAN, and F-RAN for 5G systems: Key capabilities and recent advances: CRAN, H-CRAN, and F-RAN for 5G systems: Key capabilities and recent advances". en. En: *International Journal of Network Management* 27.5 (2017), e1973. DOI: 10.1002/nem.1973. URL: https://onlinelibrary.wiley.com/doi/10.1002/nem.1973 (visitado 18-07-2022).

[28] 3GPP. *TR 38.801 V14.0.0*. Inf. téc. Release 14. 2017.

[29] rcrwireless. *Open RAN 101–RU, DU, CU: Why, what, how, when? (Reader Forum)*. en-US. 2020. URL: https://www.rcrwireless.com/20200708/open_ran/open-ran-101-ru-du-cu-reader-forum (visitado 23-07-2022).

[30] *O-RAN ALLIANCE e.V.* URL: https://www.o-ran.org/ (visitado 23-07-2022).

[31] *eNodeB architecture*. URL: https://docs.srsran.com/en/latest/usermanuals/source/srsenb/source/1_enb_intro.html (visitado 28-07-2022).

[32] *Small cell virtualization functional splits and use cases*. Inf. téc. Document 159.07.02. Small Cell Forum, 2016.

[33] *Release 21.04 · srsRAN*. en. URL: https://github.com/srsran/srsRAN/releases/tag/release_21_04.

[34] *Socket API Documentation*. URL: https://zeromq.org/socket-api/ (visitado 08-08-2022).

[35] *Overview | Protocol Buffers*. en. URL: https://developers.google.com/protocol-buffers/docs/overview (visitado 09-08-2022).

[36] *Release Protocol Buffers v3.20.1-rc1 · protocolbuffers/protobuf*. en. URL: https://github.com/protocolbuffers/protobuf/releases/tag/v3.20.1-rc1 (visitado 09-08-2022).

[37] *iPerf - iPerf3 and iPerf2 User Documentation*. URL: https://iperf.fr/iperf-doc.php (visitado 08-08-2022).

[38] J. M. Aransay Azofra. *Tema 2: Relaciones entre clases. Herencia entre clases.* URL: https://www.unirioja.es/cu/jearansa/0910/archivos/EIPR_Tema02.pdf.

Apéndice A

Resultados completos

A.1. Escenario I y II

Valores medios de CPU y memoria

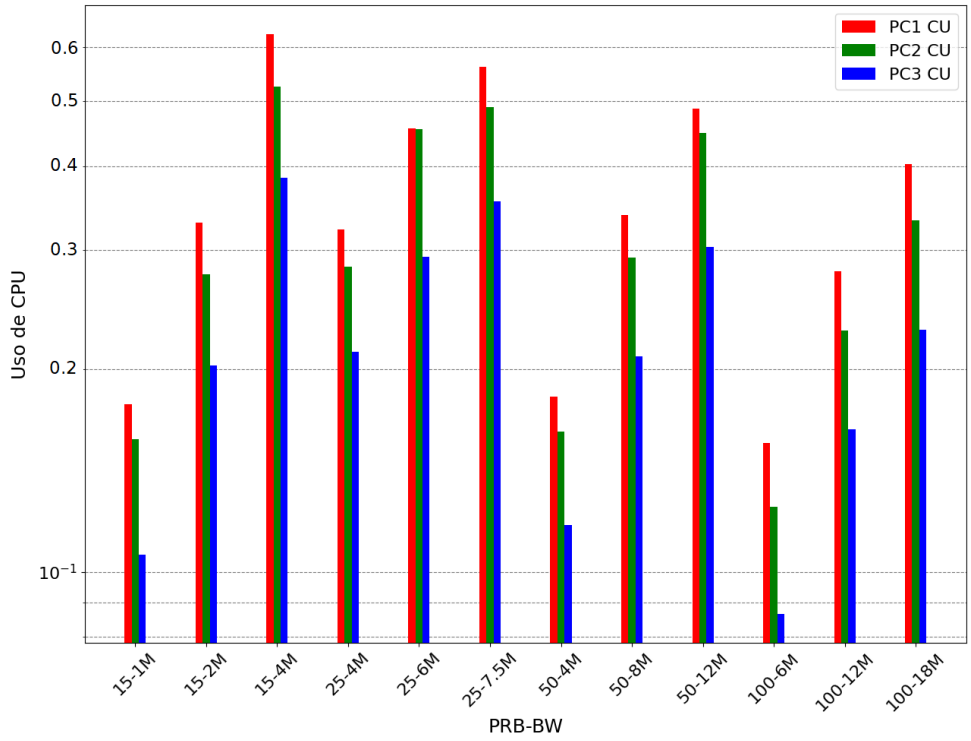


Figura A.1: Comportamiento de la CU en términos del consumo de CPU en el *split RLC_MAC* en el Escenario I.

Boxplots - distribuciones y valores atípicos

En función del *split*

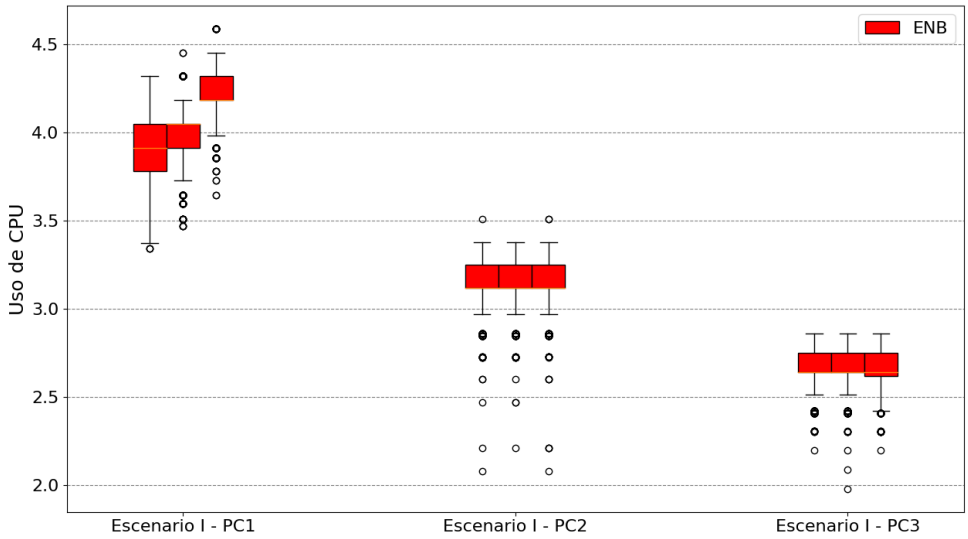


Figura A.2: Evolución del comportamiento de la eNB a nivel de CPU en cada uno de los dispositivos de prueba para las tasas de 15 PRBs. Cada uno de los *boxplots* de cada uno de los 3 escenarios representados se corresponden (de izquierda a derecha) con las tasas de 1, 2 y 4 Mbps.

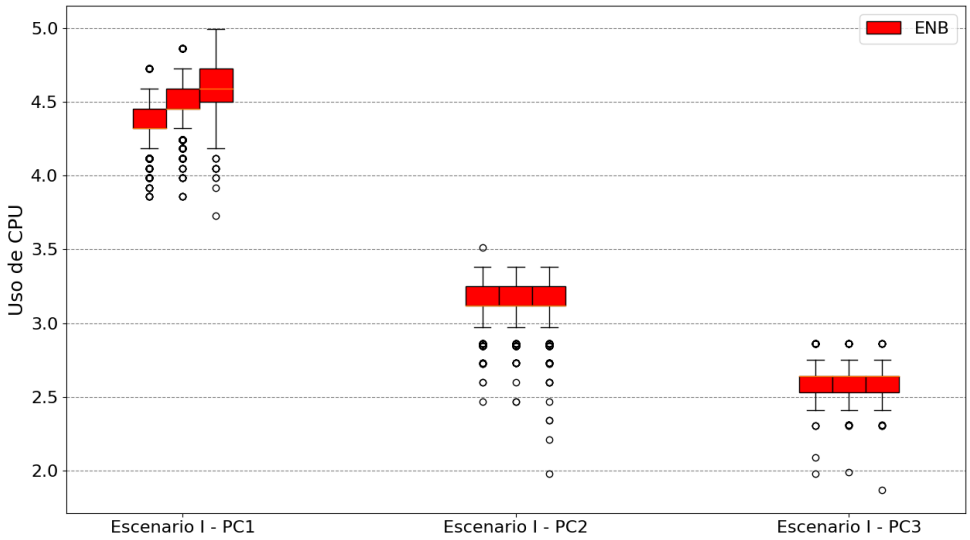


Figura A.3: Evolución del comportamiento de la eNB a nivel de CPU en cada uno de los dispositivos de prueba para las tasas de 25 PRBs. Cada uno de los *boxplots* de cada uno de los 3 escenarios representados se corresponden (de izquierda a derecha) con las tasas de 4, 6 y 7.5 Mbps.

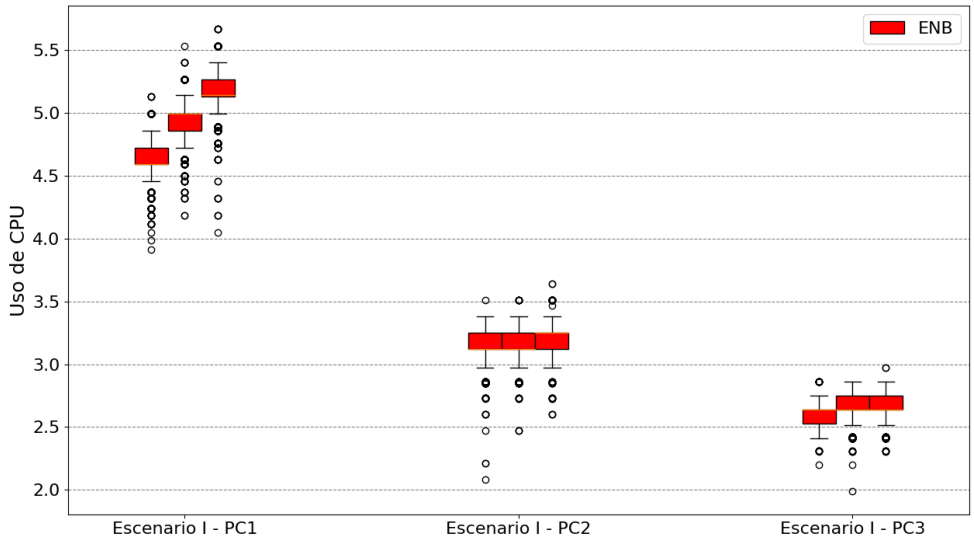


Figura A.4: Evolución del comportamiento de la eNB a nivel de CPU en cada uno de los dispositivos de prueba para las tasas de 50 PRBs. Cada uno de los *boxplots* de cada uno de los 3 escenarios representados se corresponden (de izquierda a derecha) con las tasas de 4, 8 y 12 Mbps.

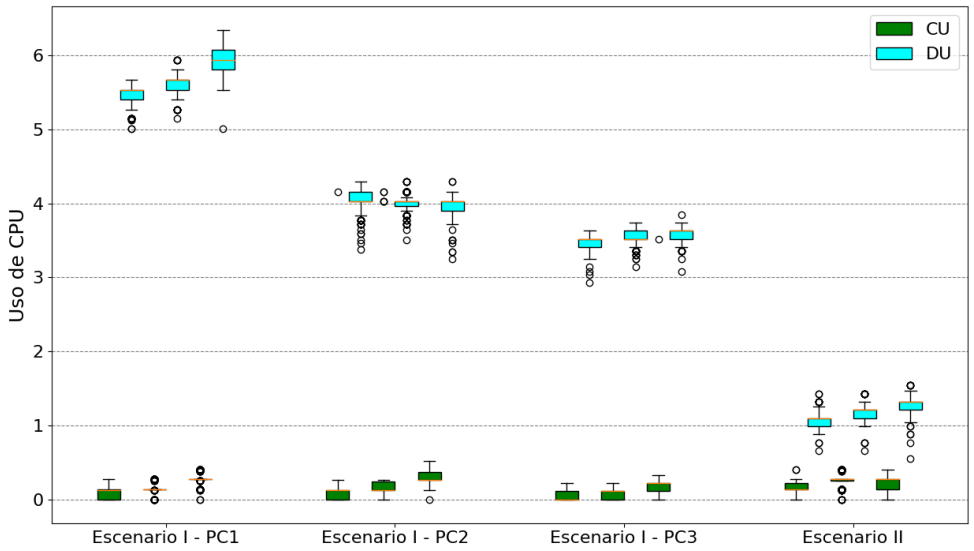


Figura A.5: Evolución del comportamiento de la CU-DU del *split PDCP_RLC* a nivel de CPU en los Escenarios I y II para las tasas de 15 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 1 Mbps, los segundos con 2 Mbps y los terceros con 4 Mbps.

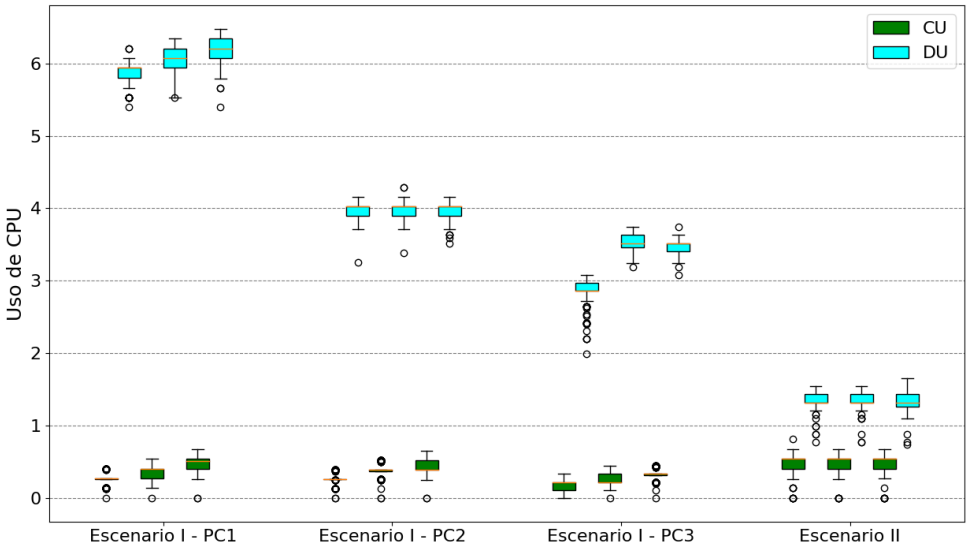


Figura A.6: Evolución del comportamiento de la CU-DU del *split PDCP_RLC* a nivel de CPU en los Escenarios I y II para las tasas de 25 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 4 Mbps, los segundos con 6 Mbps y los terceros con 7.5 Mbps.

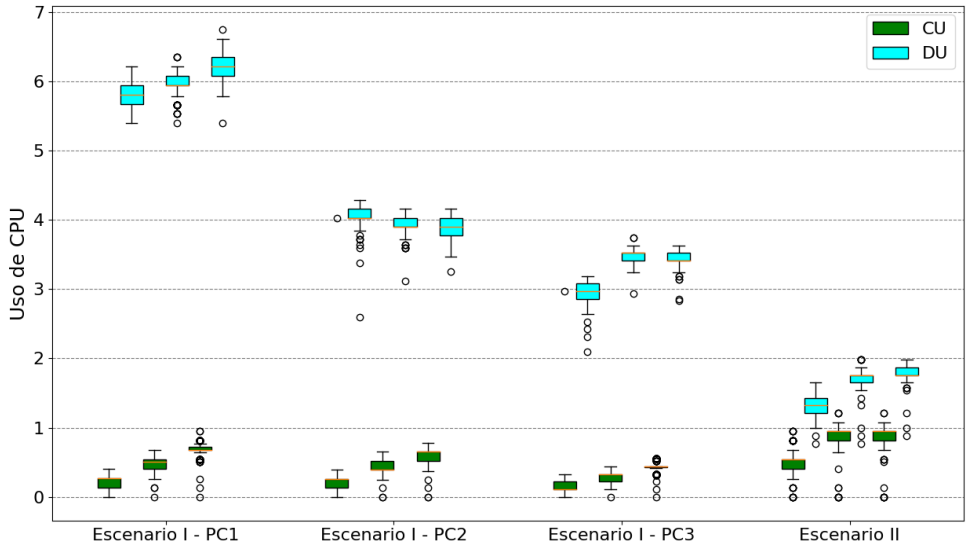


Figura A.7: Evolución del comportamiento de la CU-DU del *split PDCP_RLC* a nivel de CPU en los Escenarios I y II para las tasas de 50 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 4 Mbps, los segundos con 8 Mbps y los terceros con 12 Mbps.

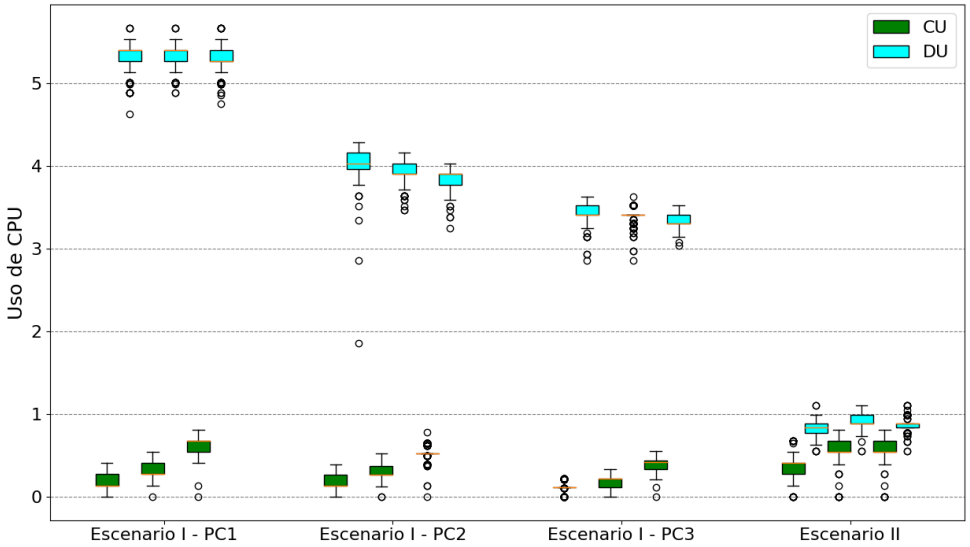


Figura A.8: Evolución del comportamiento de la CU-DU del *split RLC_MAC* a nivel de CPU en los Escenarios I y II para las tasas de 15 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 1 Mbps, los segundos con 2 Mbps y los terceros con 4 Mbps.

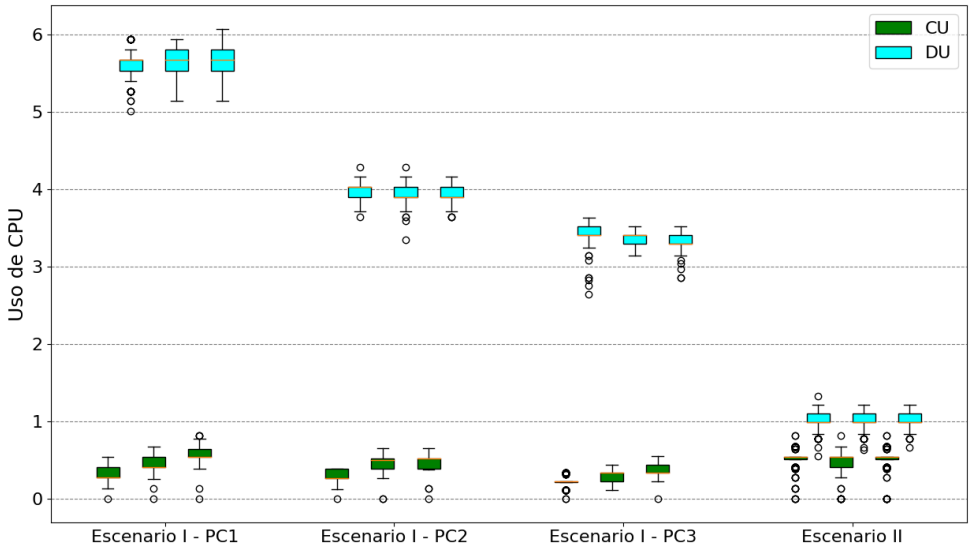


Figura A.9: Evolución del comportamiento de la CU-DU del *split RLC_MAC* a nivel de CPU en los Escenarios I y II para las tasas de 25 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 4 Mbps, los segundos con 6 Mbps y los terceros con 7.5 Mbps.

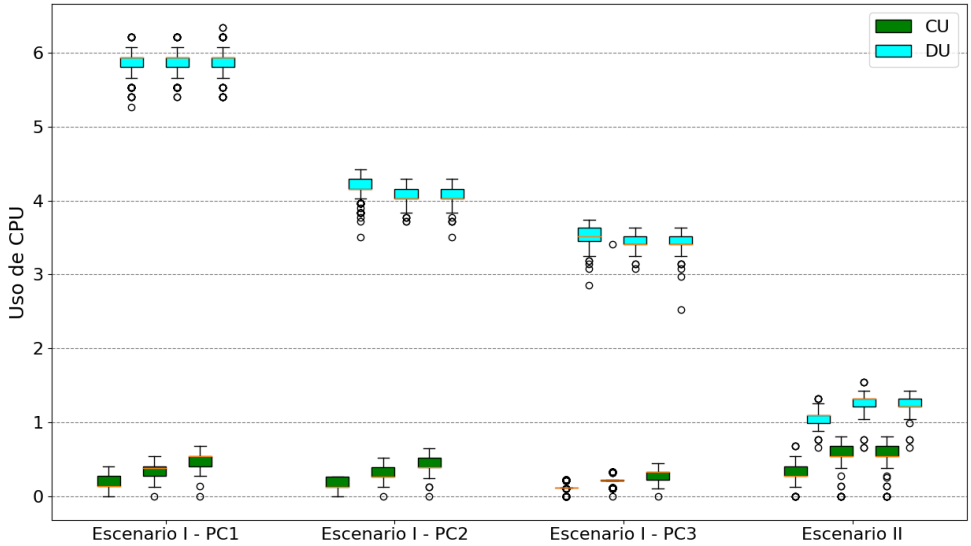


Figura A.10: Evolución del comportamiento de la CU-DU del *split RLC_MAC* a nivel de CPU en los Escenarios I y II para las tasas de 50 PRBs. Los primeros boxplots verde y azul de cada uno de los 4 escenarios (comenzando por la izquierda) se corresponden con la tasa de 4 Mbps, los segundos con 8 Mbps y los terceros con 12 Mbps.

En función del *PC*

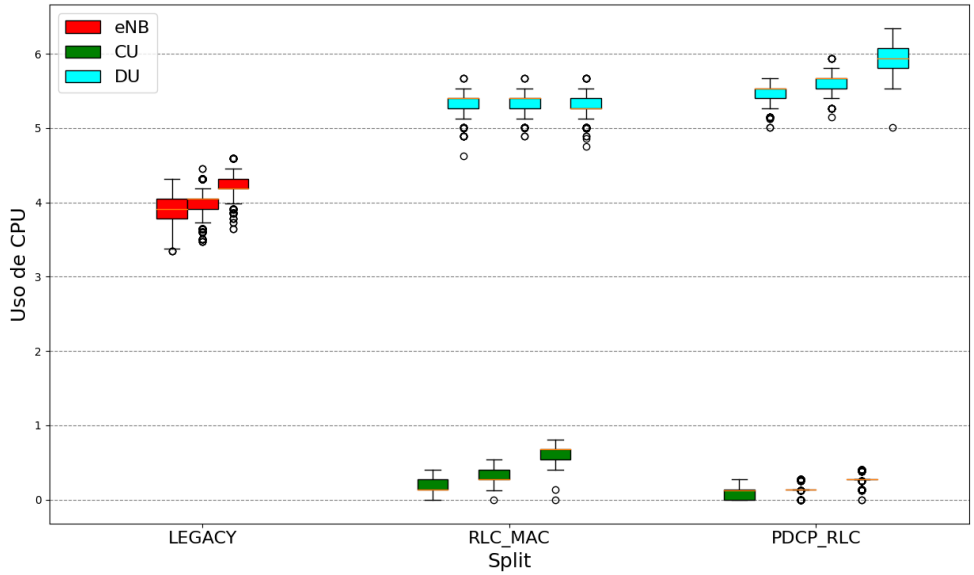


Figura A.11: Evolución del comportamiento de la eNB o CU-DU en el *PC1* de los *splits* implementados para las tasas de 15 PRBs (1, 2 y 4 Mbps).

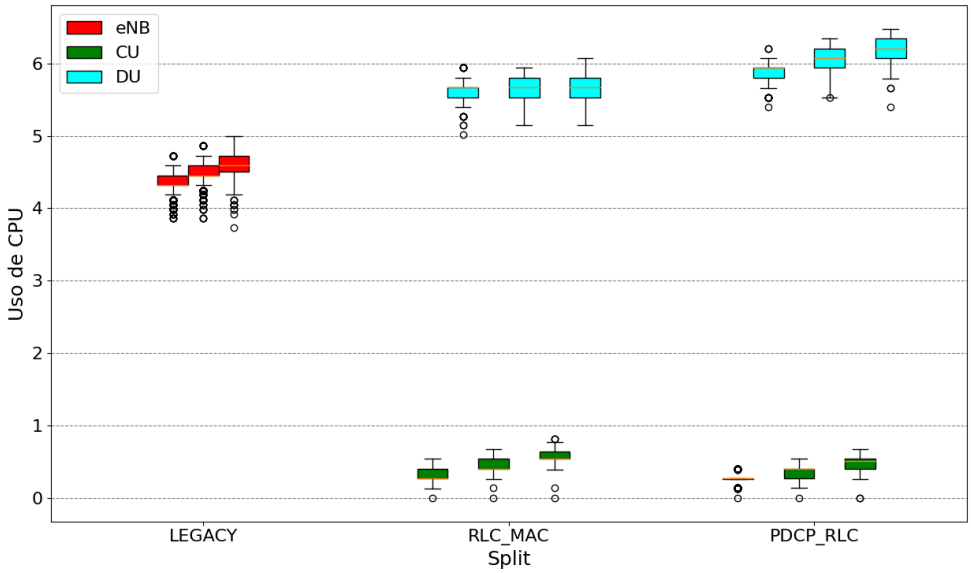


Figura A.12: Evolución del comportamiento de la eNB o CU-DU en el *PC1* de los *splits* implementados para las tasas de 25 PRBs (4, 6 y 7.5 Mbps).

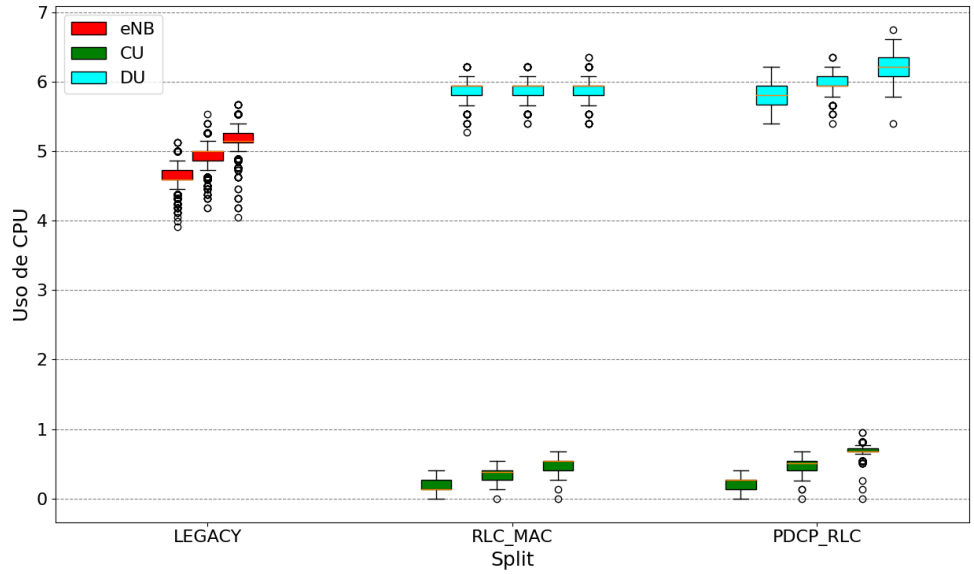


Figura A.13: Evolución del comportamiento de la eNB o CU-DU en el *PC1* de los *splits* implementados para las tasas de 50 PRBs (4, 8 y 12 Mbps).

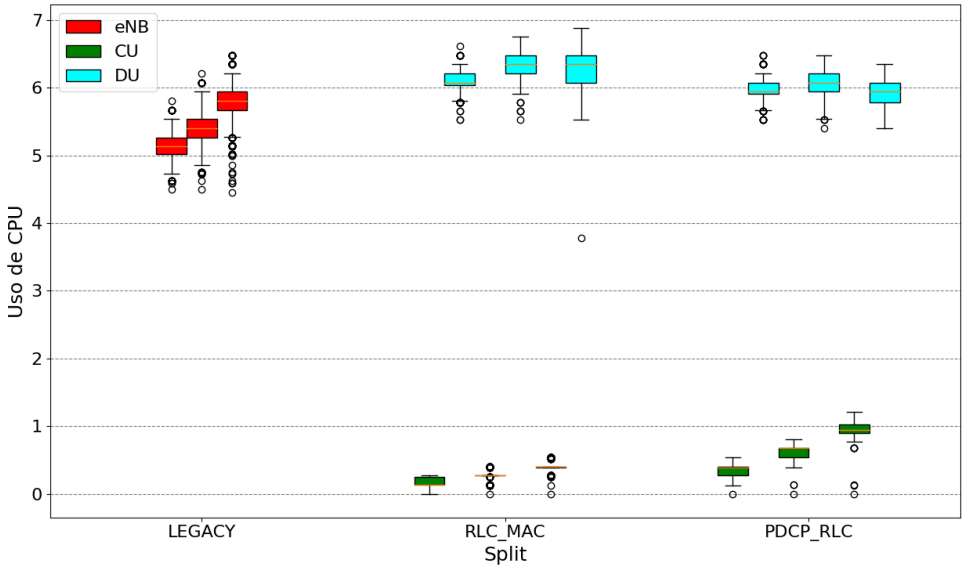


Figura A.14: Evolución del comportamiento de la eNB o CU-DU en el *PC1* de los *splits* implementados para las tasas de 100 PRBs (6, 12 y 18 Mbps).

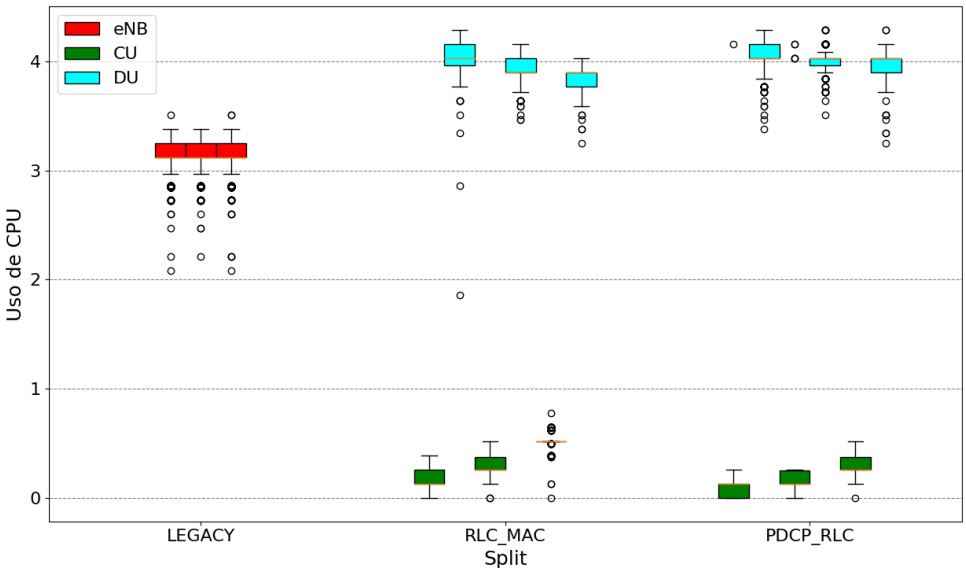


Figura A.15: Evolución del comportamiento de la eNB o CU-DU en el *PC2* de los *splits* implementados para las tasas de 15 PRBs (1, 2 y 4 Mbps).

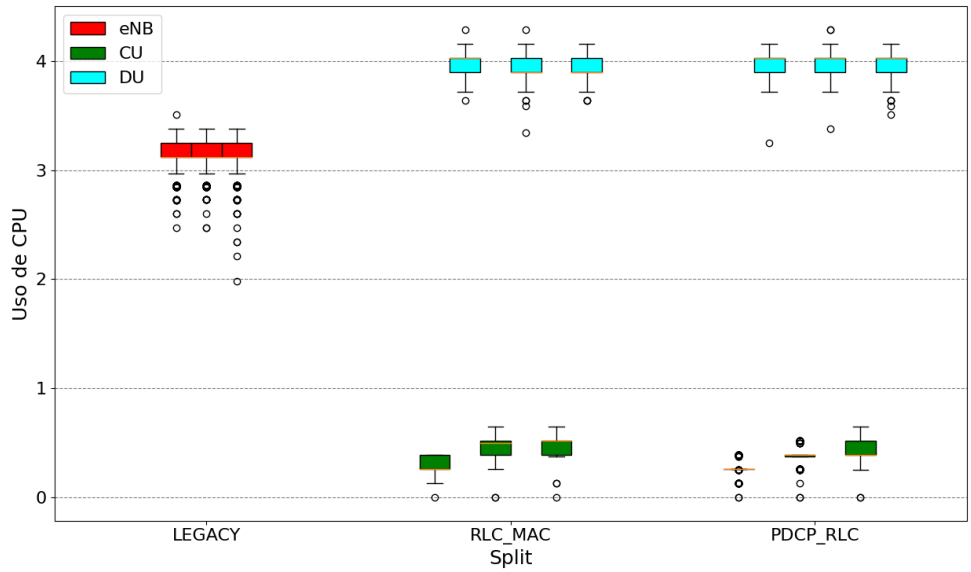


Figura A.16: Evolución del comportamiento de la eNB o CU-DU en el *PC2* de los *splits* implementados para las tasas de 25 PRBs (4, 6 y 7.5 Mbps).

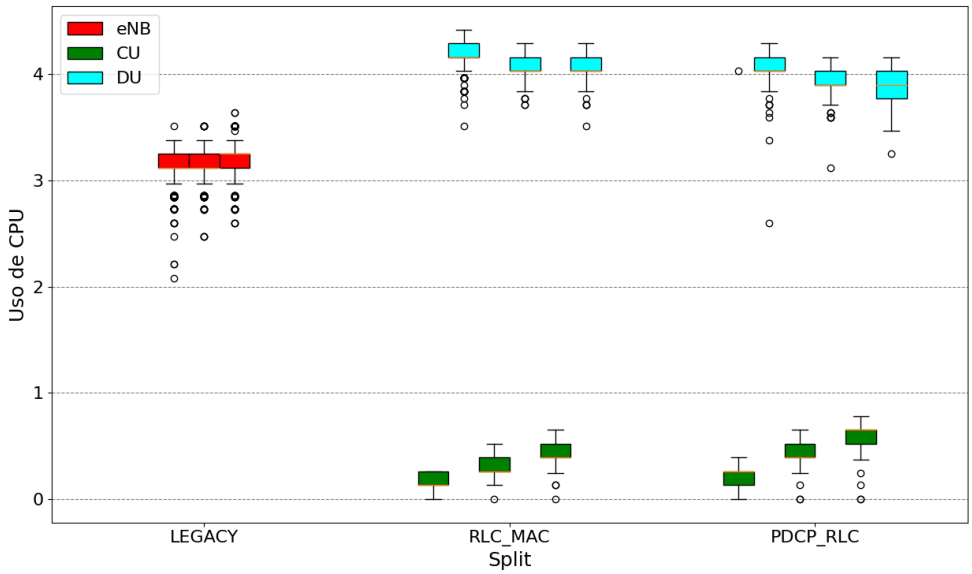


Figura A.17: Evolución del comportamiento de la eNB o CU-DU en el *PC2* de los *splits* implementados para las tasas de 50 PRBs (4, 8 y 12 Mbps).

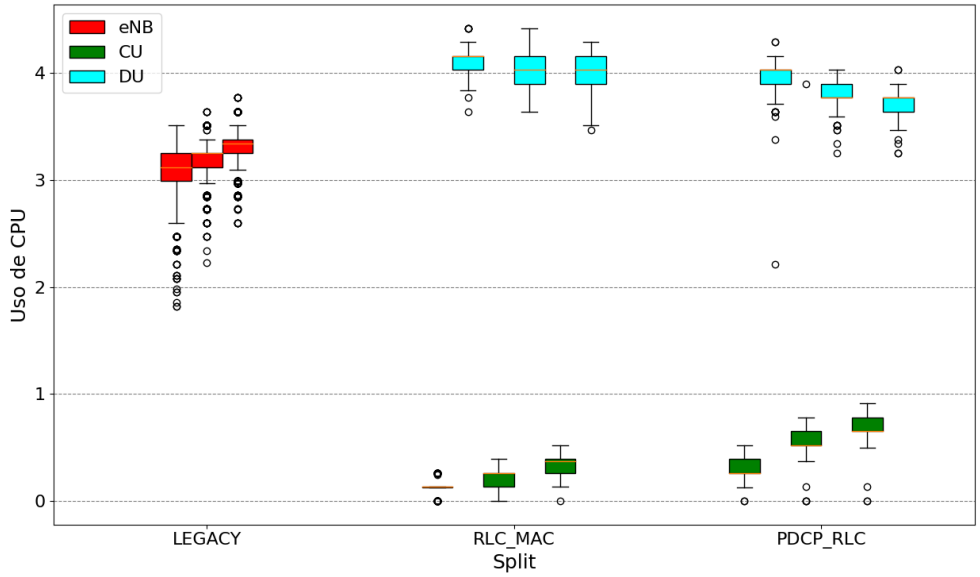


Figura A.18: Evolución del comportamiento de la eNB o CU-DU en el *PC2* de los *splits* implementados para las tasas de 100 PRBs (6, 12 y 18 Mbps).

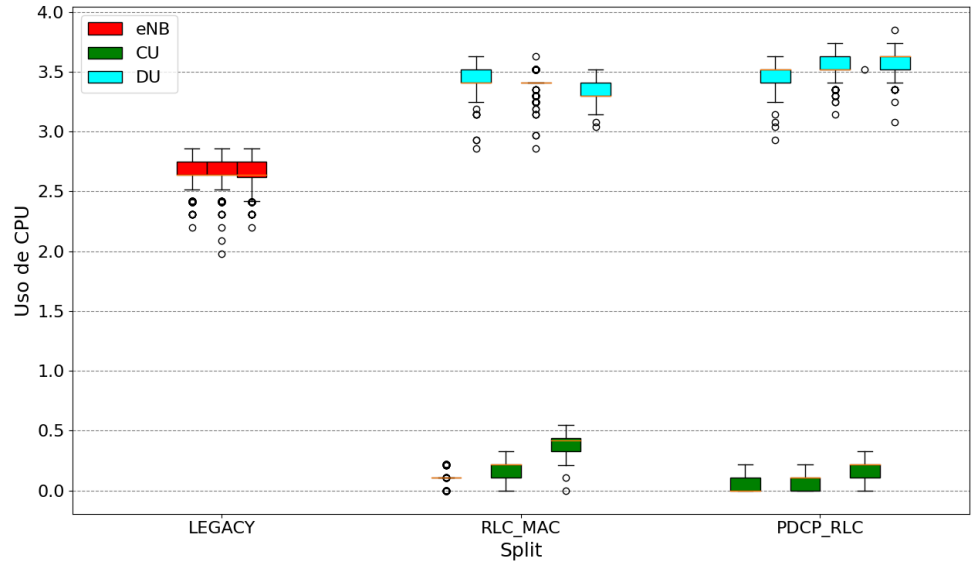


Figura A.19: Evolución del comportamiento de la eNB o CU-DU en el *PC3* de los *splits* implementados para las tasas de 15 PRBs (1, 2 y 4 Mbps).

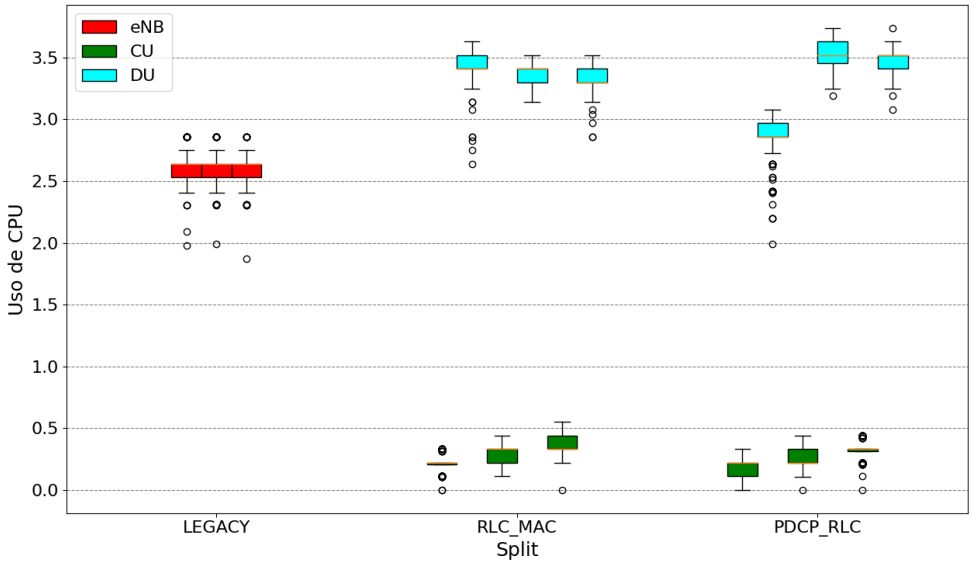


Figura A.20: Evolución del comportamiento de la eNB o CU-DU en el *PC3* de los *splits* implementados para las tasas de 25 PRBs (4, 6 y 7.5 Mbps).

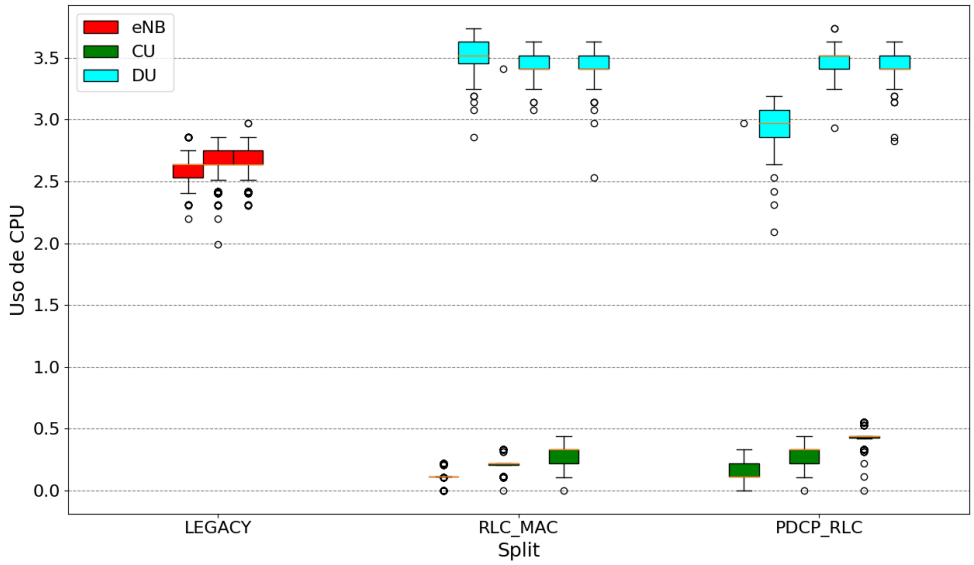


Figura A.21: Evolución del comportamiento de la eNB o CU-DU en el *PC3* de los *splits* implementados para las tasas de 50 PRBs (4, 8 y 12 Mbps).

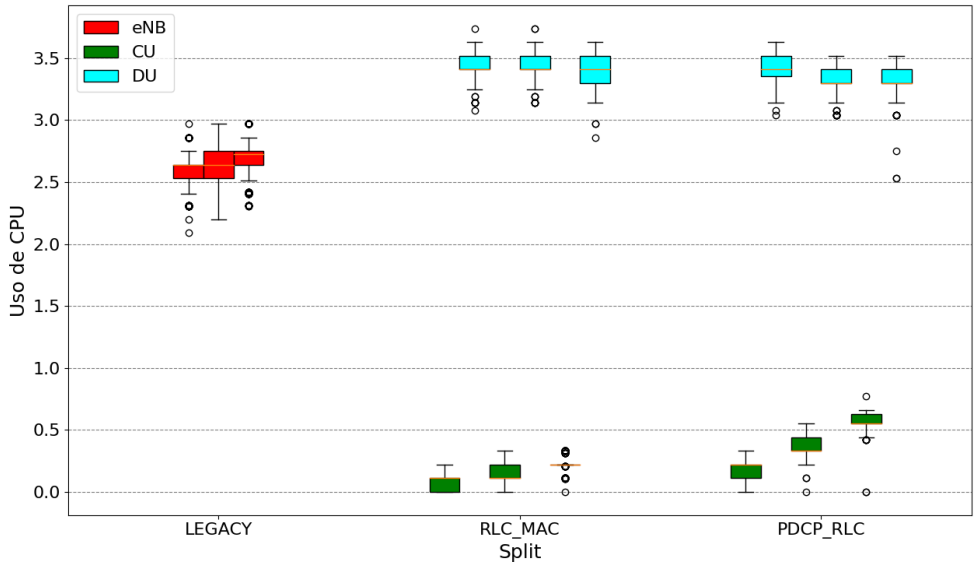


Figura A.22: Evolución del comportamiento de la eNB o CU-DU en el *PC3* de los *splits* implementados para las tasas de 100 PRBs (6, 12 y 18 Mbps).

Tasas de tráfico de los *wrappers*

Split PDCP_RLC

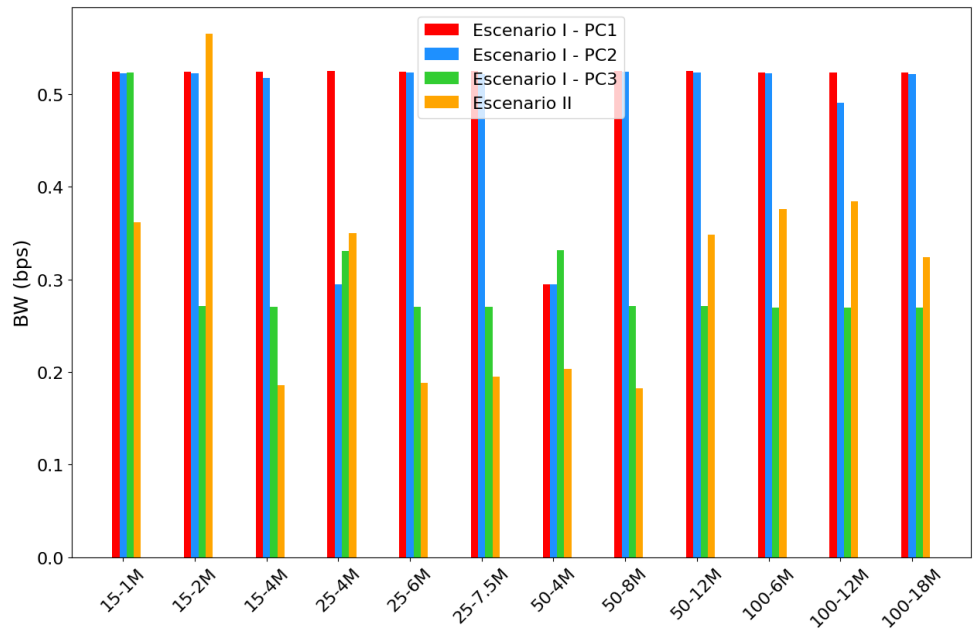


Figura A.23: Tráfico enviado por el wrapper *gtpu_rrc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*.

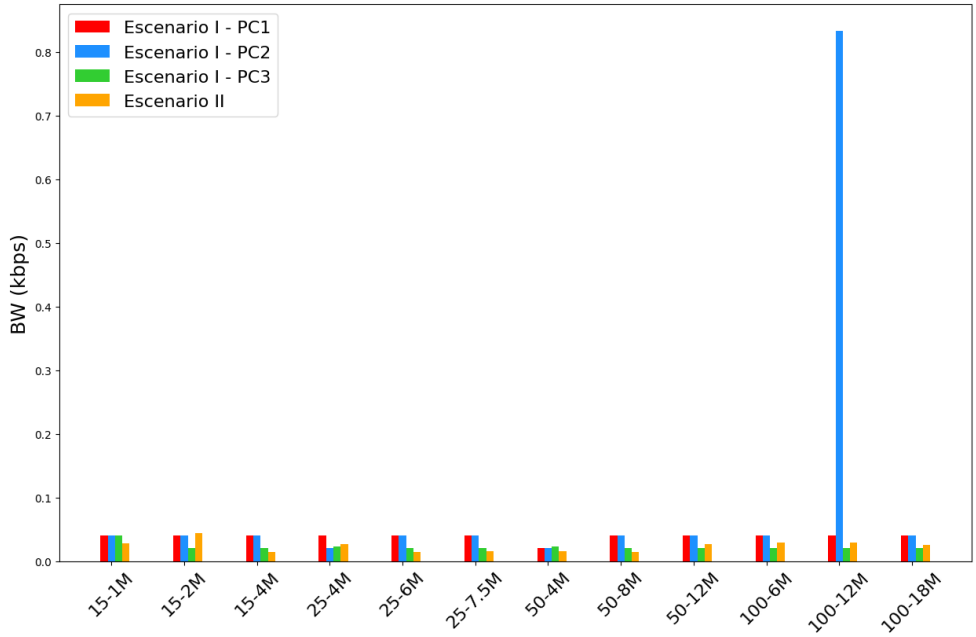


Figura A.24: Tráfico enviado por el wrapper *mac_rrc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*.

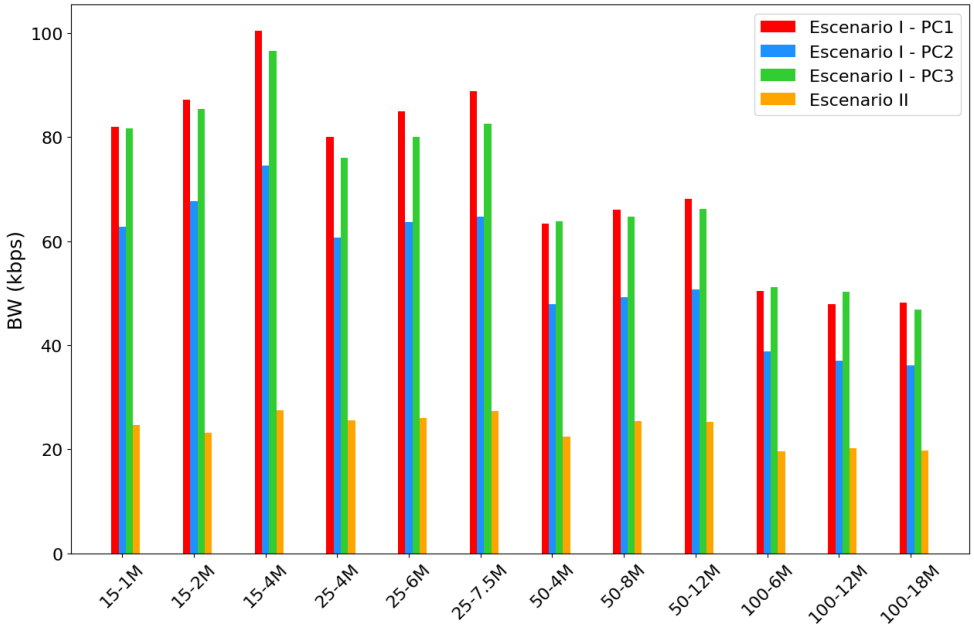


Figura A.25: Tráfico enviado por el wrapper *rrc_mac_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*.

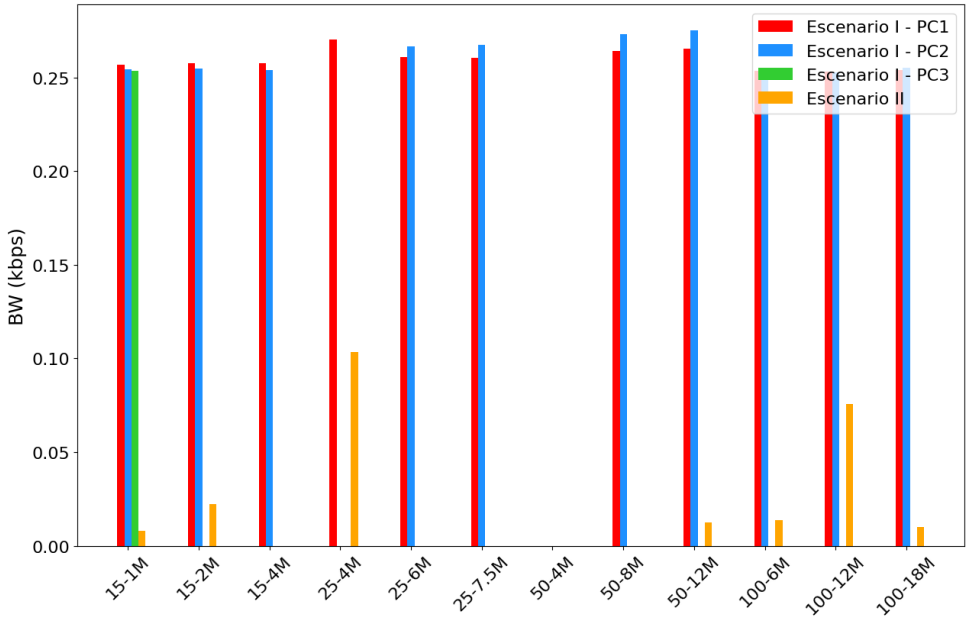


Figura A.26: Tráfico enviado por el wrapper *rrc_rlc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*.

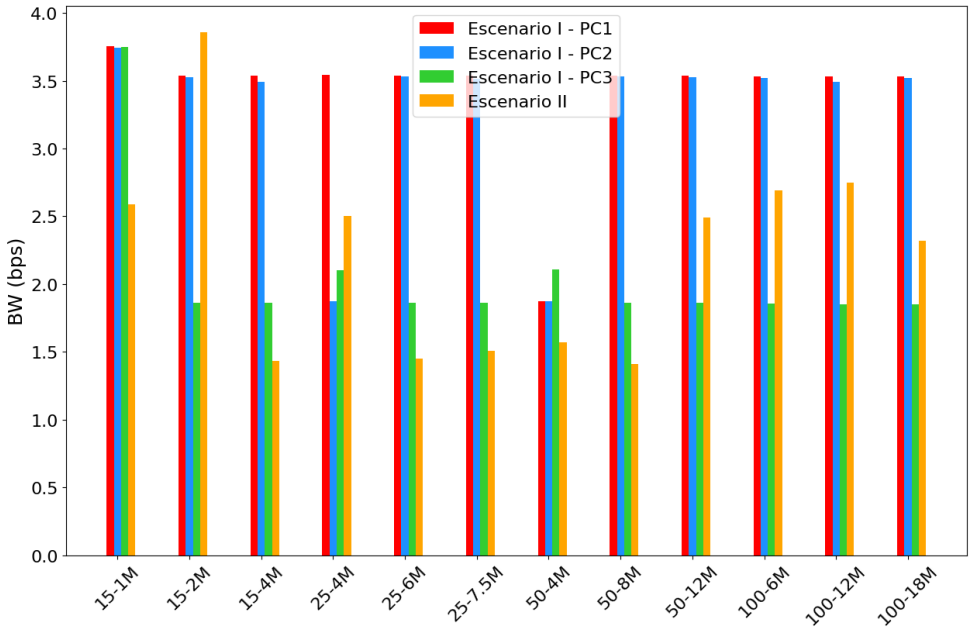


Figura A.27: Tráfico enviado por el wrapper *rlc_rrc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*.

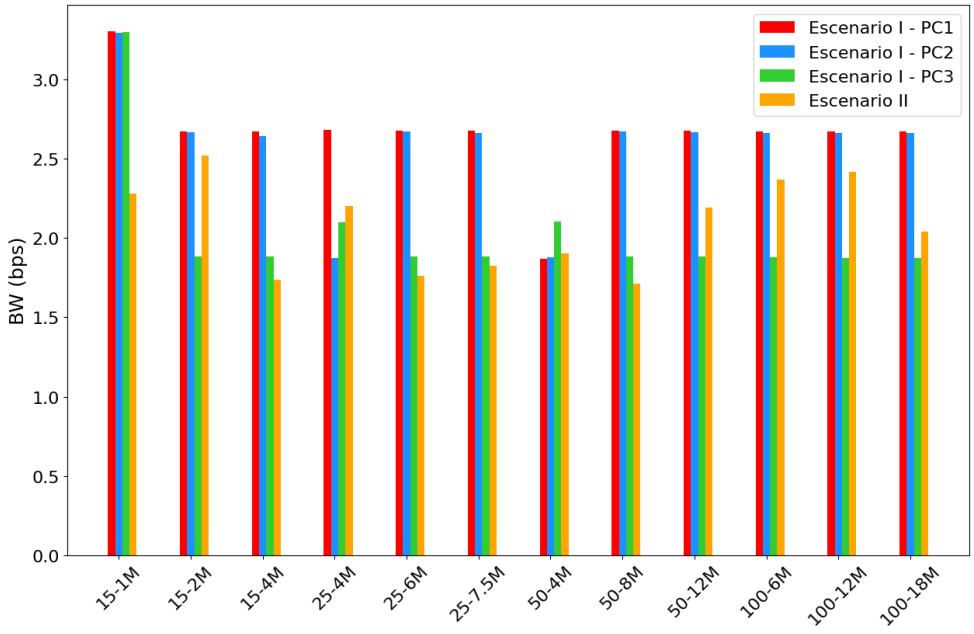


Figura A.28: Tráfico enviado por el wrapper *rrc_pdcp_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*.

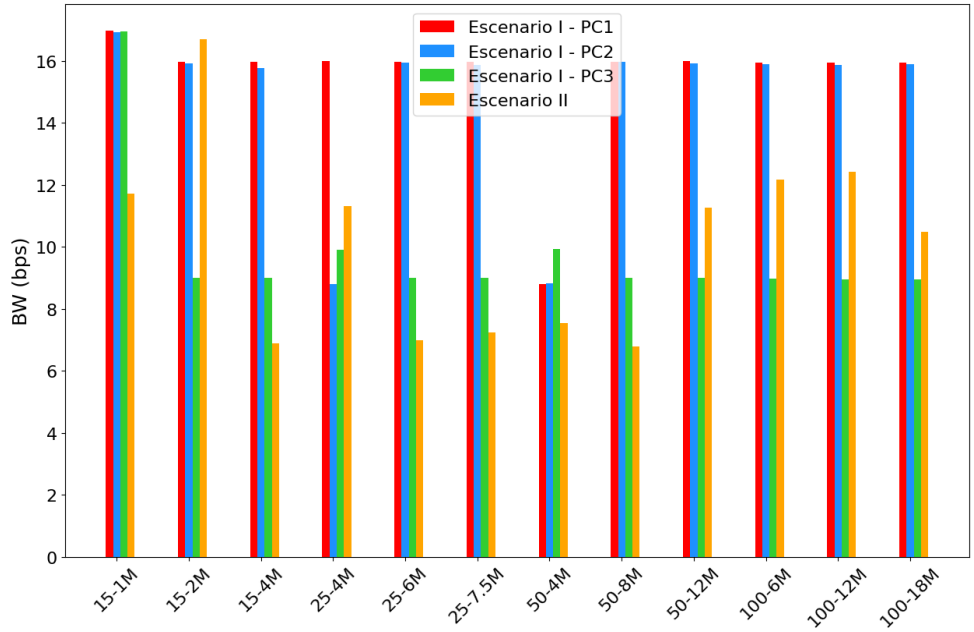


Figura A.29: Tráfico enviado por el wrapper *pdcp_rrc_wrapper* en los Escenarios I y II en el *split PDCP_RLC* en función del equipo, el número de PRBs y la tasa *iperf*.

Split RLC_MAC

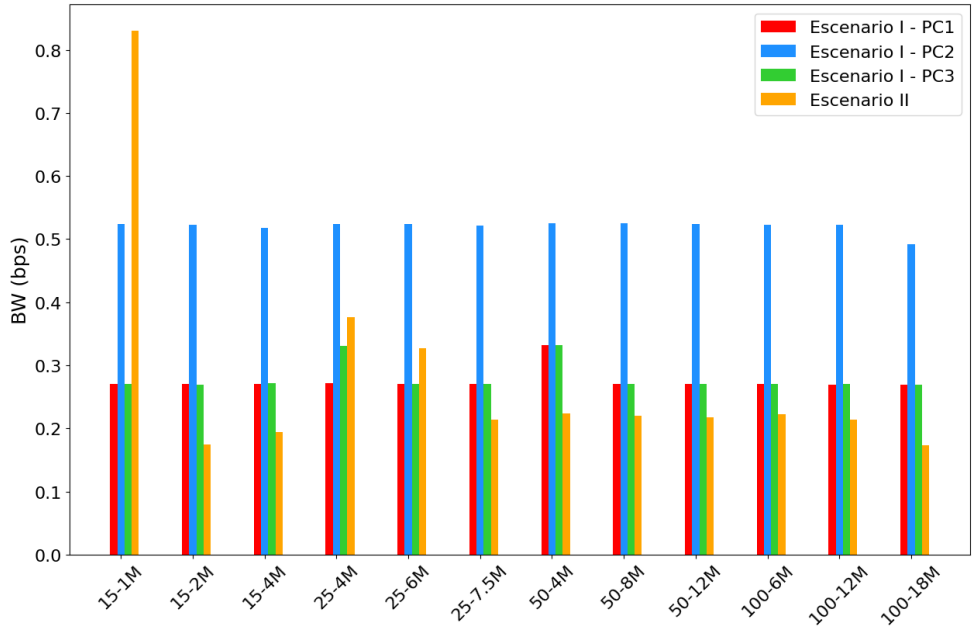


Figura A.30: Tráfico enviado por el wrapper *gtpu_rrc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

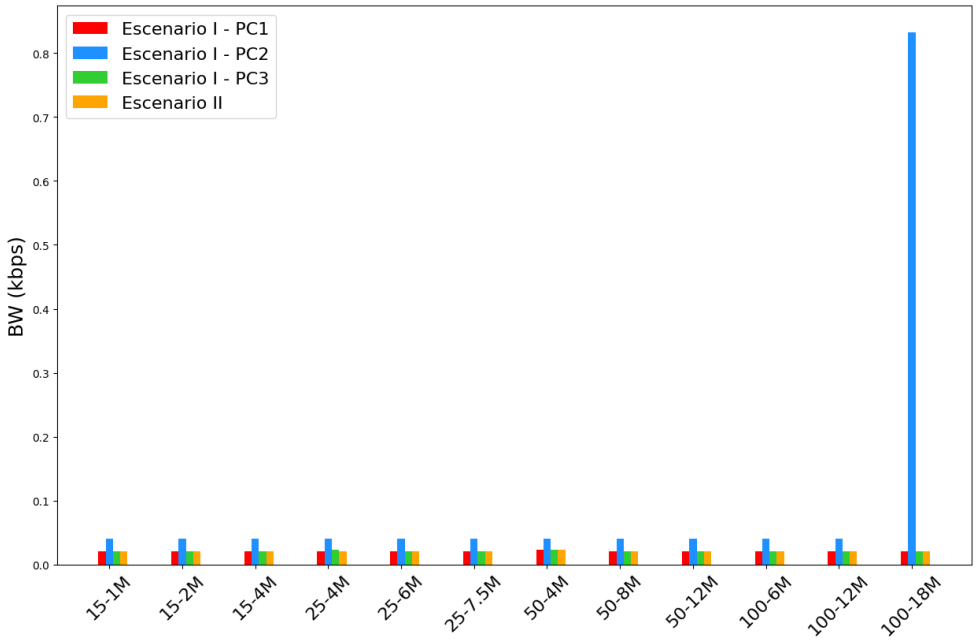


Figura A.31: Tráfico enviado por el wrapper *mac_rrc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

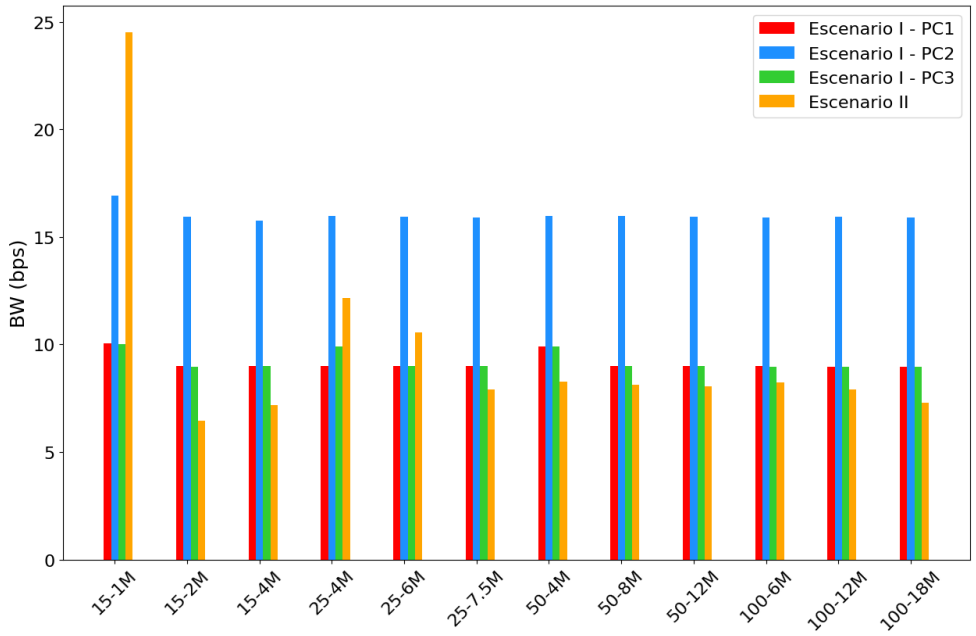


Figura A.32: Tráfico enviado por el wrapper *pdcp_rrc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

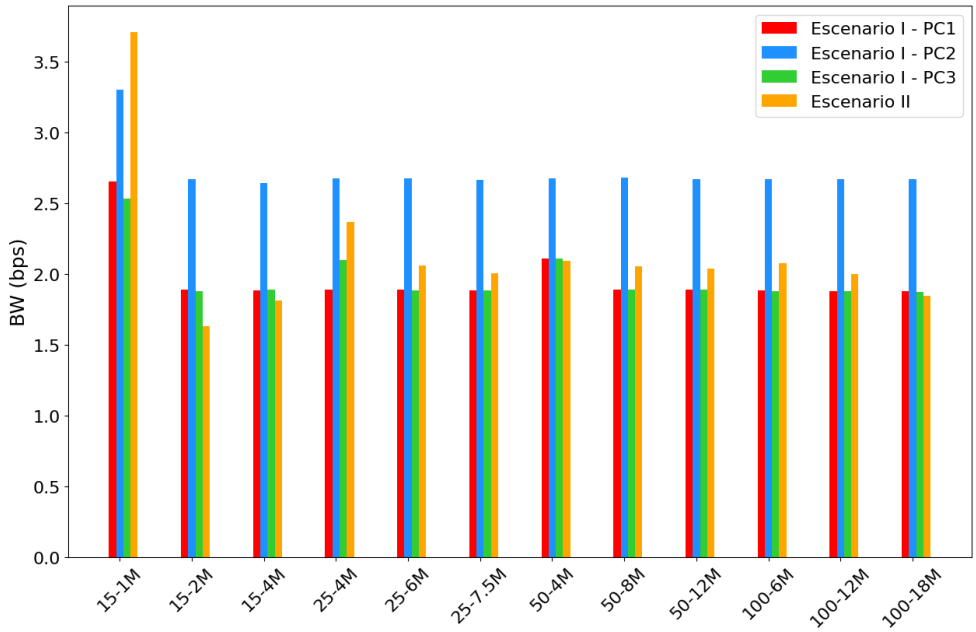


Figura A.33: Tráfico enviado por el wrapper *rrc_pdcp_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

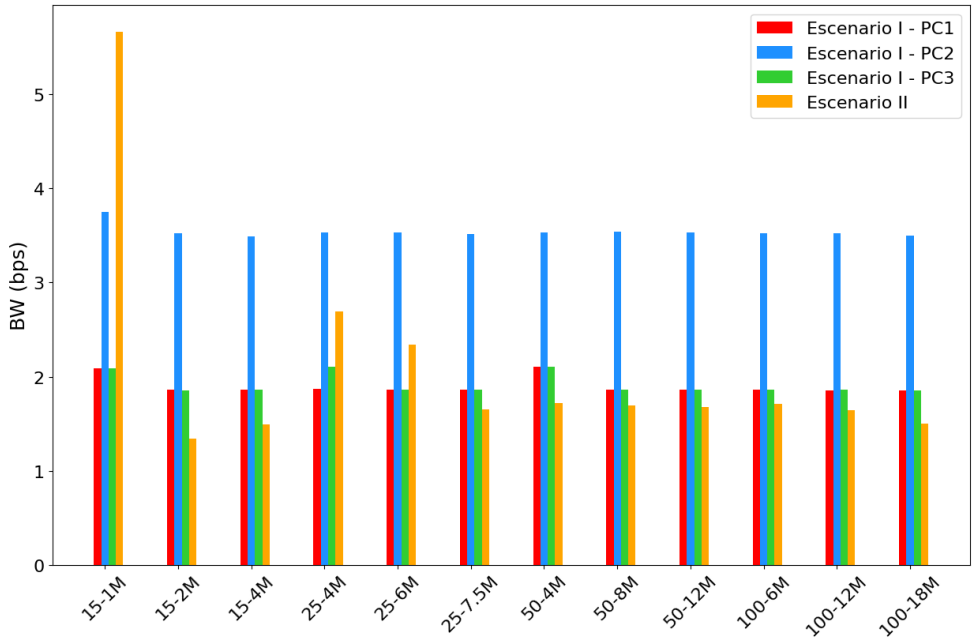


Figura A.34: Tráfico enviado por el wrapper *rlc_rrc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

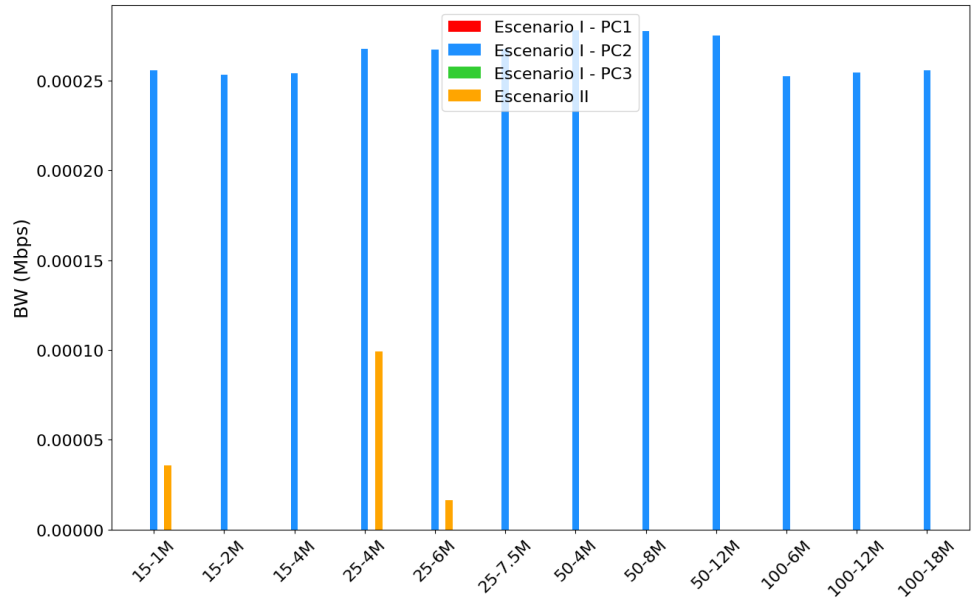


Figura A.35: Tráfico enviado por el wrapper *rrc_rlc_wrapper* en los Escenarios I y II en el *split RLC_MAC* en función del equipo, el número de PRBs y la tasa *iperf*.

A.2. Escenario III

Valores medios de CPU y memoria

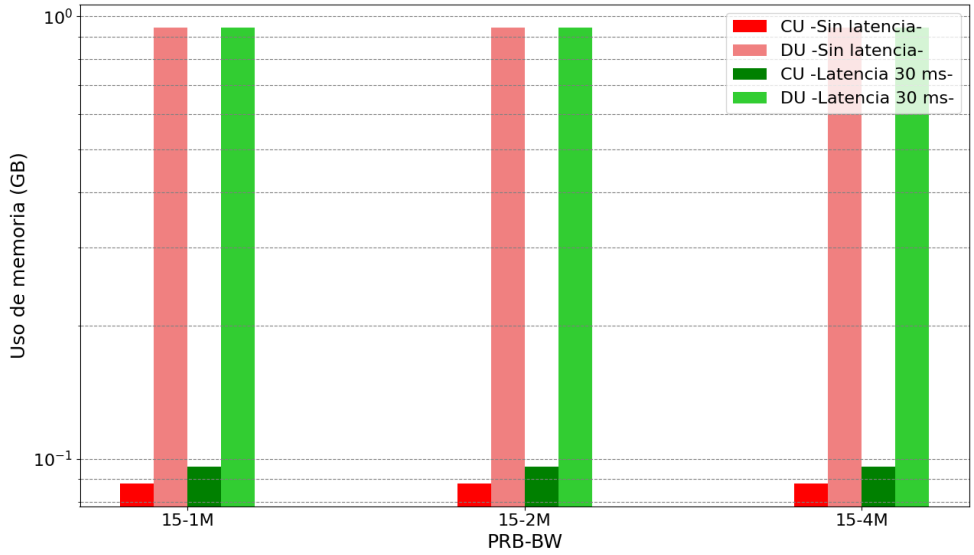


Figura A.36: Valores medios del uso de memoria de la CU-DU en el *split RLC_MAC* tanto en el caso sin latencia como en el caso en el que se añade la latencia de 30 ms mientras se envía tráfico UL.

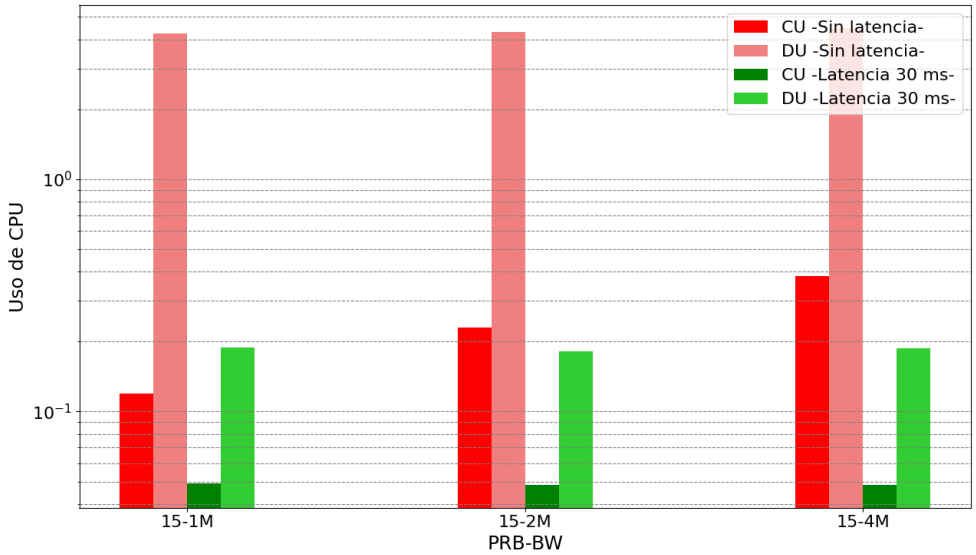


Figura A.37: Valores medios del consumo de CPU de la CU-DU en el *split RLC_MAC* tanto en el caso sin latencia como en el caso en el que se añade la latencia de 30 ms mientras se envía tráfico UL. Para los dos supuestos (sin y con latencia), la primera pareja (verde y azul) de boxplots se corresponden con 1 Mbps, la segunda con 2 Mbps y la tercera con 4 Mbps.

Boxplots - distribuciones y valores atípicos

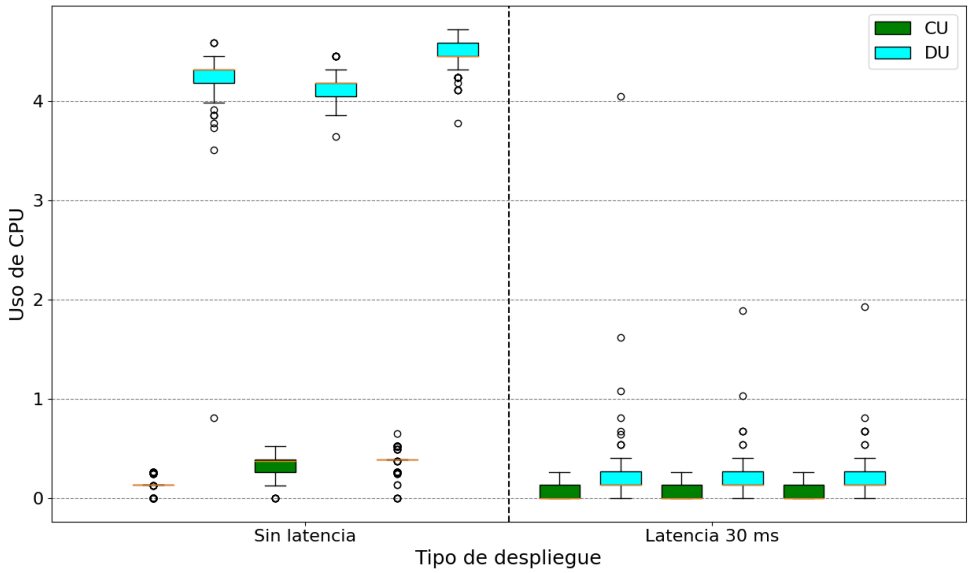


Figura A.38: Comportamiento de la CU-DU en el *split RLC_MAC* para cada una de las tasas de 15 PRBs en función de la existencia o no de *latencia* adicional en el *fronthaul* mientras se envía tráfico UL.

Tasas de tráfico de los *wrappers*

Split PDCP_RLC UL

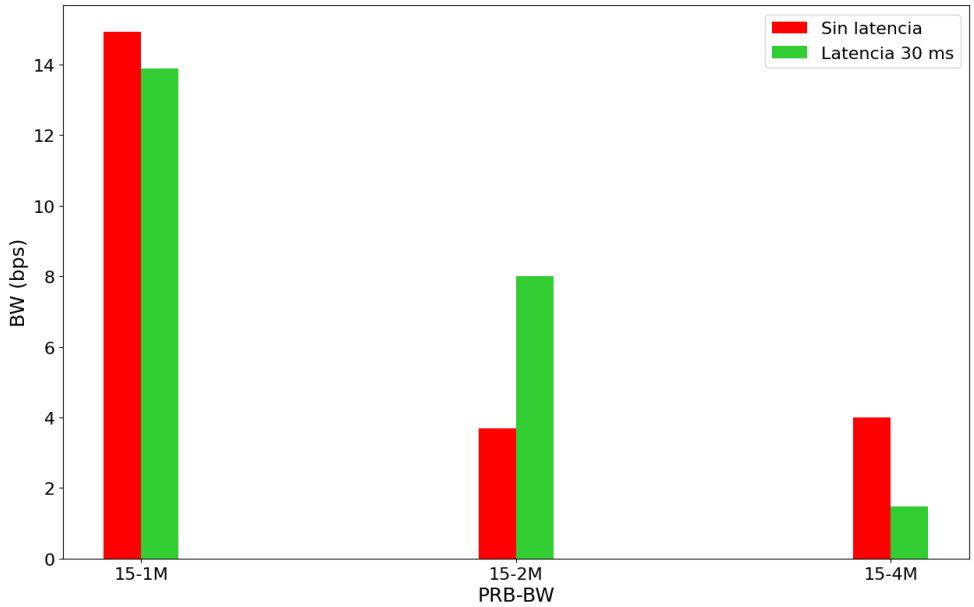


Figura A.39: Tráfico enviado por *gtpu_rrc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL

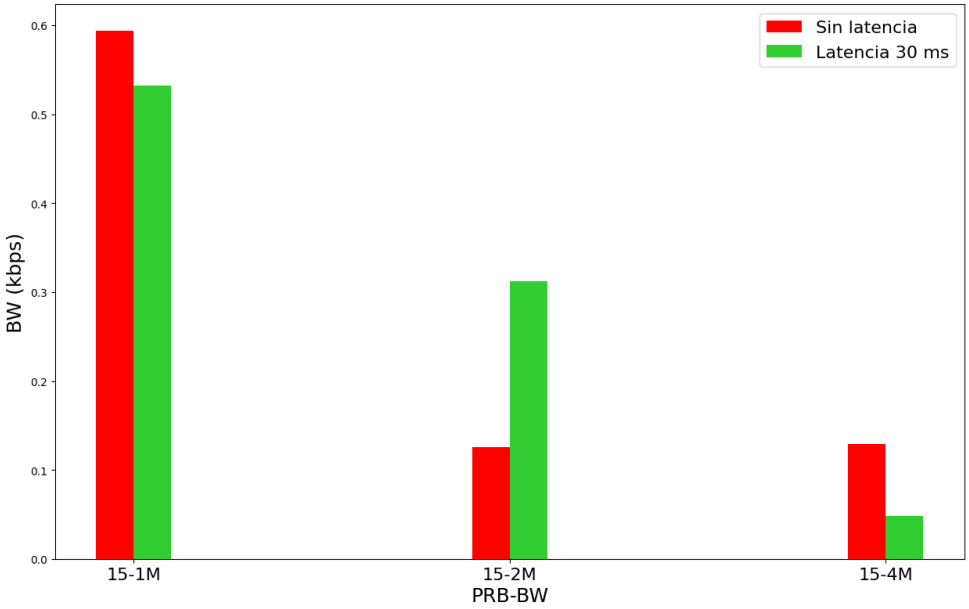


Figura A.40: Tráfico enviado por *pdcp_rrc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL

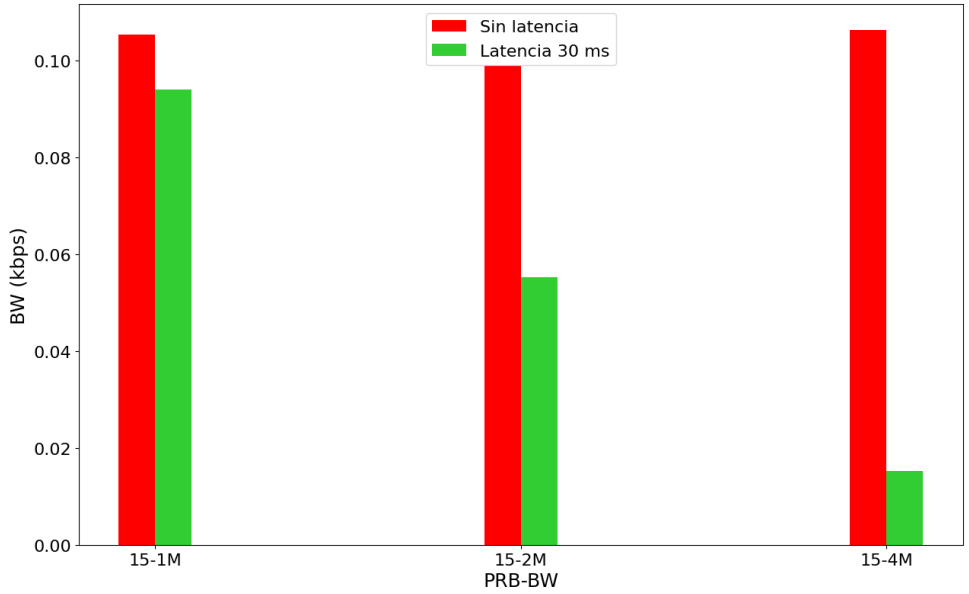


Figura A.41: Tráfico enviado por *rrc_pdcp_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico UL

Split PDCP_RLC DL

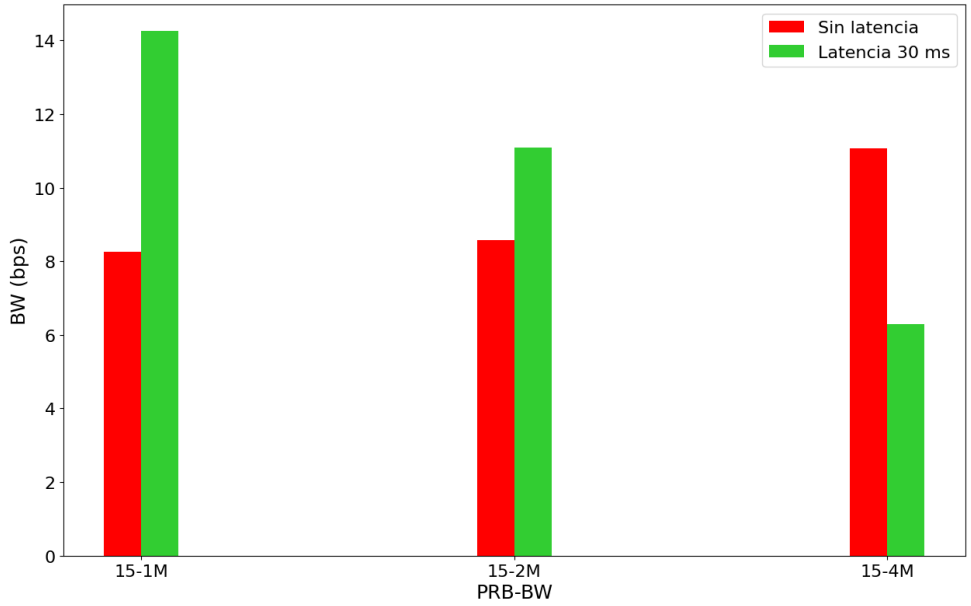


Figura A.42: Tráfico enviado por *gtpu_rrc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL

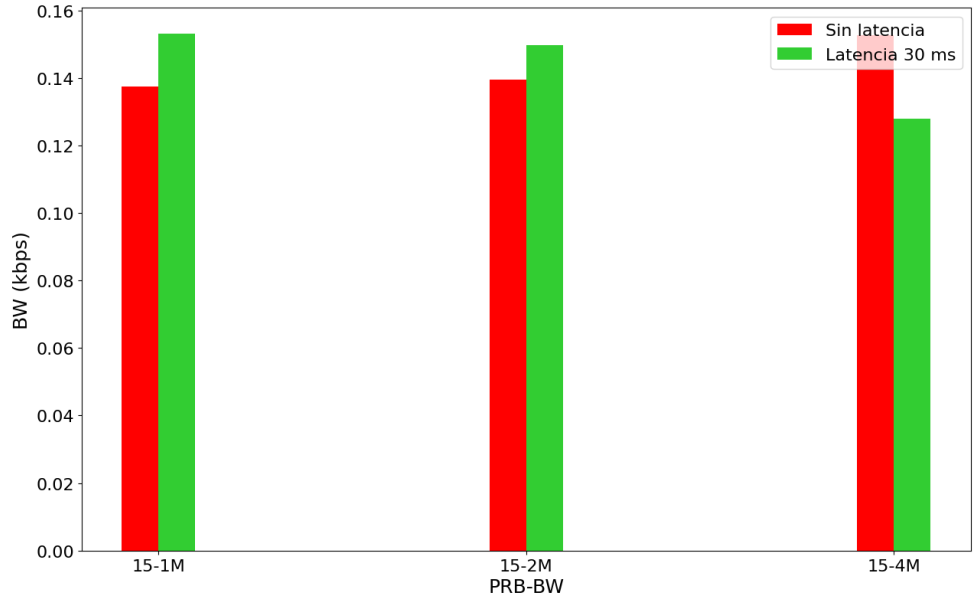


Figura A.43: Tráfico enviado por *pdcp_rlc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL

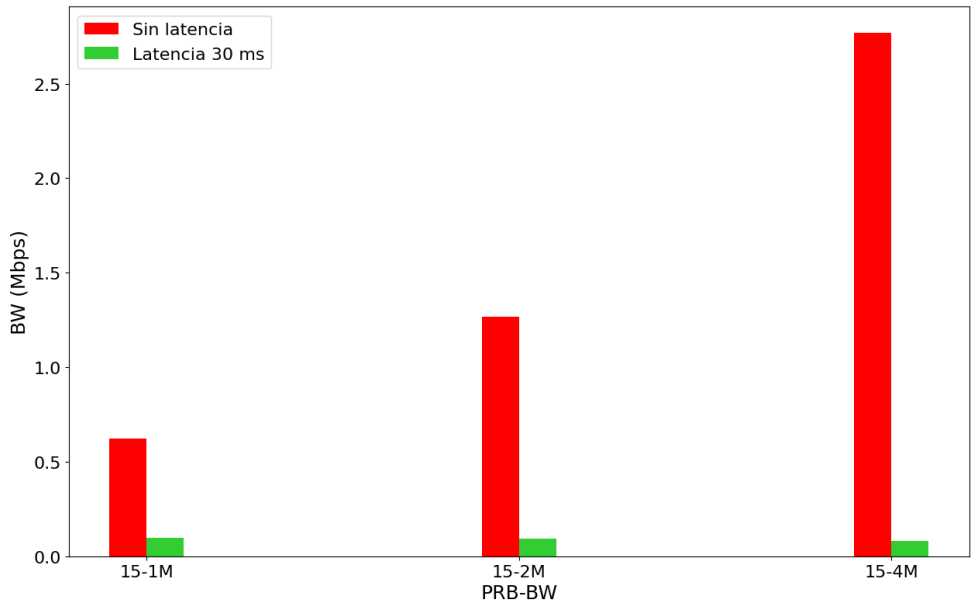


Figura A.44: Tráfico enviado por *rlc_pdcp_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL

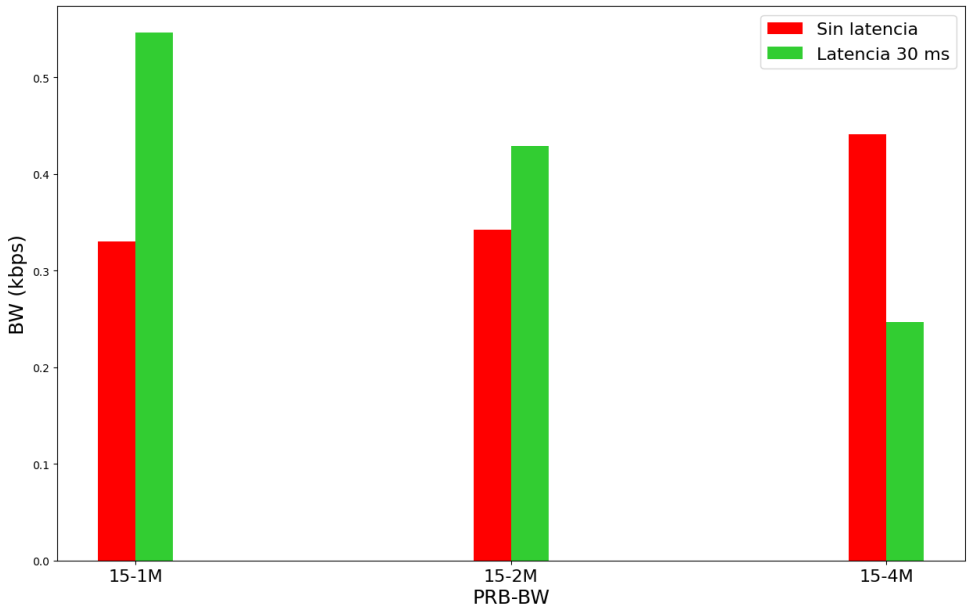


Figura A.45: Tráfico enviado por *pdcp_rrc_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL

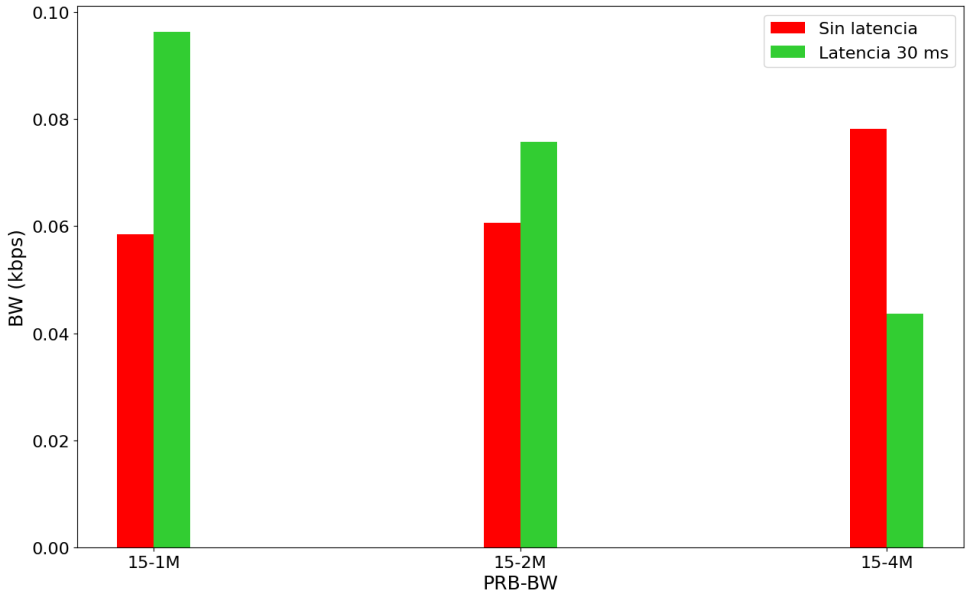


Figura A.46: Tráfico enviado por *rrc_pdcp_wrapper* en el *split PDCP_RLC_SPLIT* mientras se envía tráfico DL

Split RLC_MAC UL

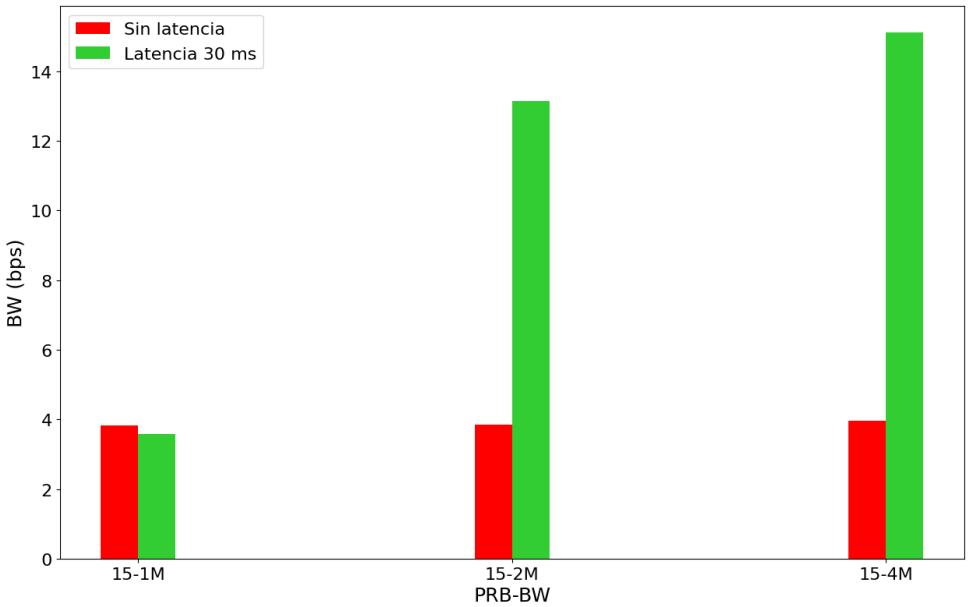


Figura A.47: Tráfico enviado por *gtpu_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL

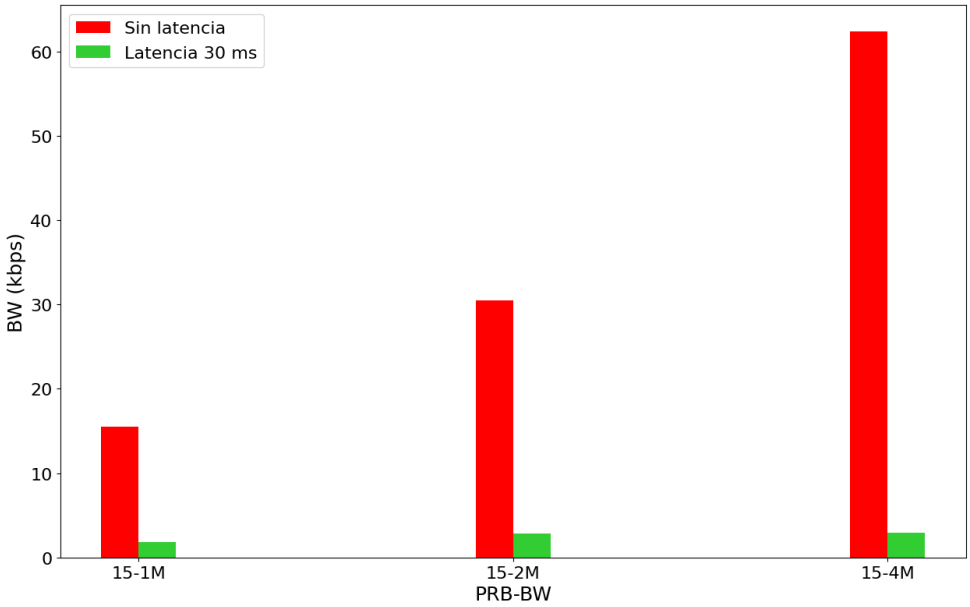


Figura A.48: Tráfico enviado por *mac_rlc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL

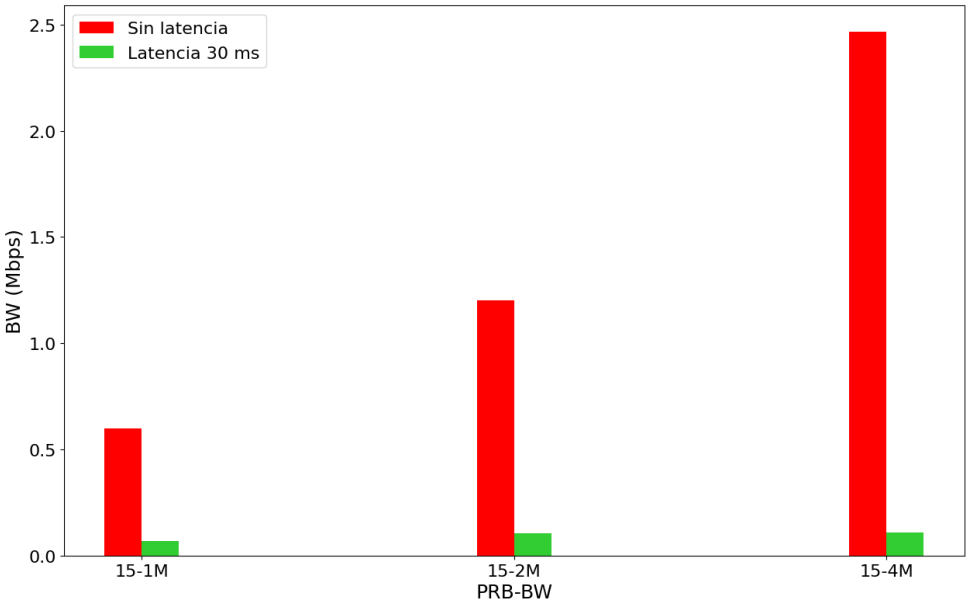


Figura A.49: Tráfico enviado por *rlc_mac_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL

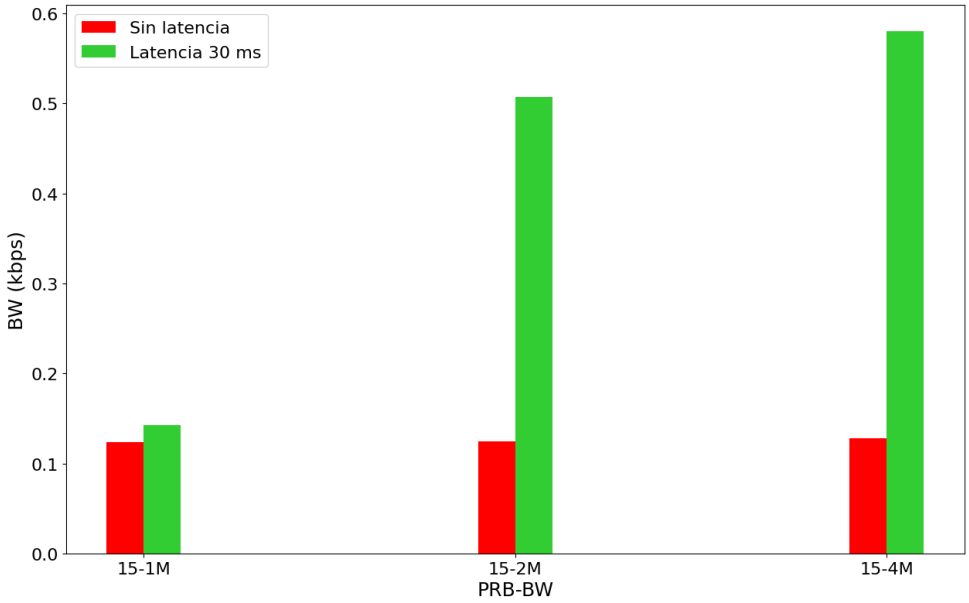


Figura A.50: Tráfico enviado por *pdcp_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL

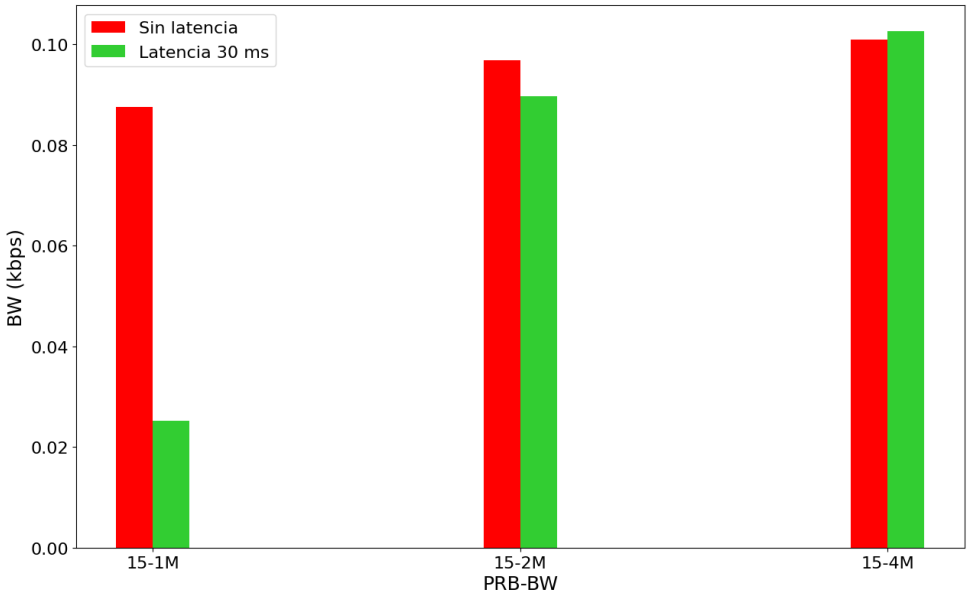


Figura A.51: Tráfico enviado por *rrc_pdcp_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL

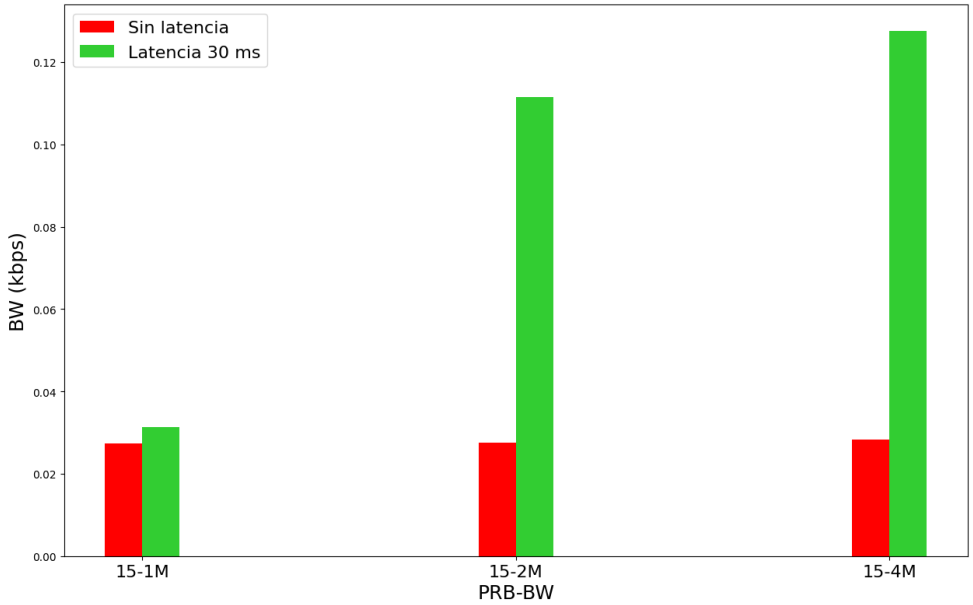


Figura A.52: Tráfico enviado por *rlc_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL

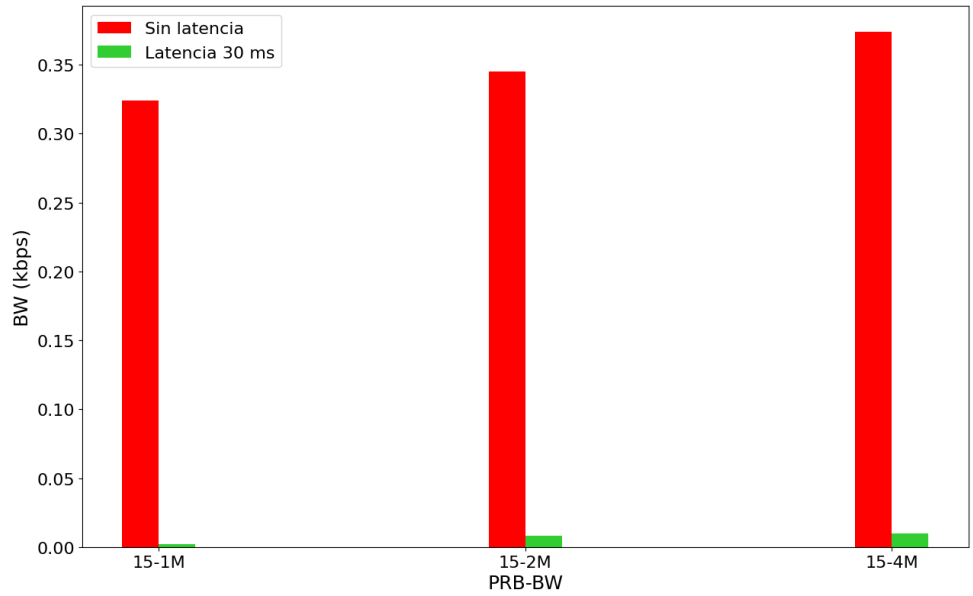


Figura A.53: Tráfico enviado por *rrc_rlc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico UL

Split RLC_MAC DL

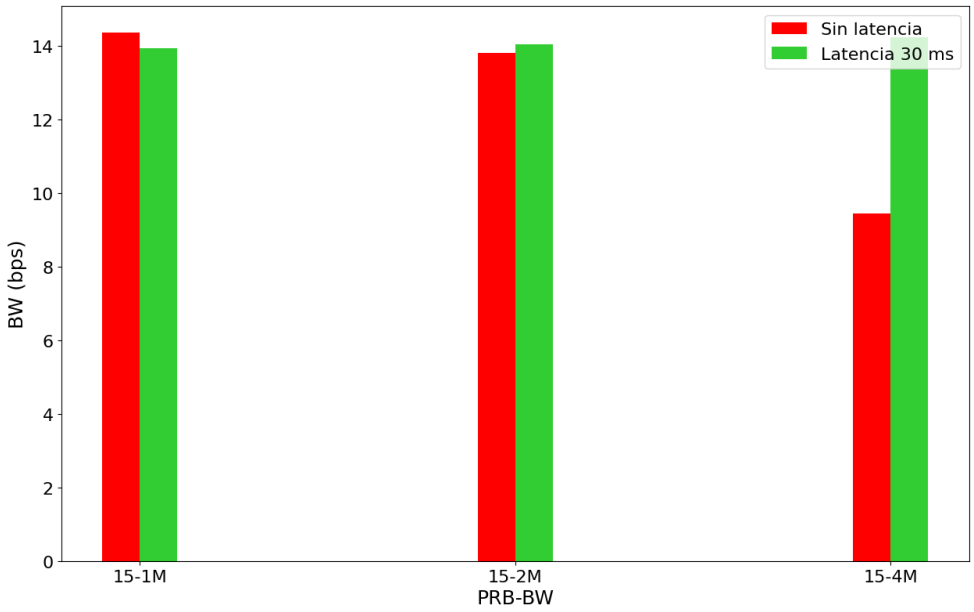


Figura A.54: Tráfico enviado por *gtpu_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL

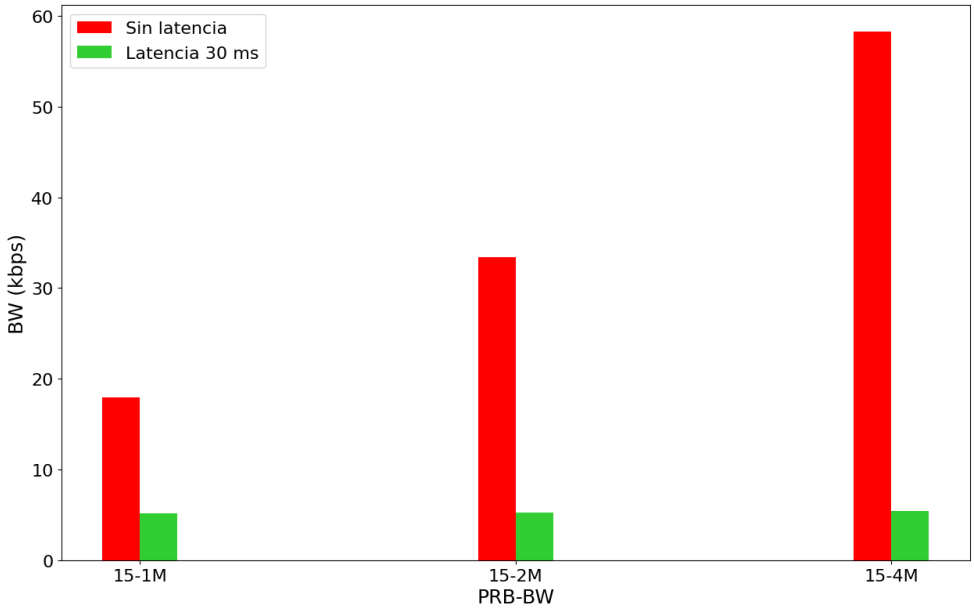


Figura A.55: Tráfico enviado por *mac_rlc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL

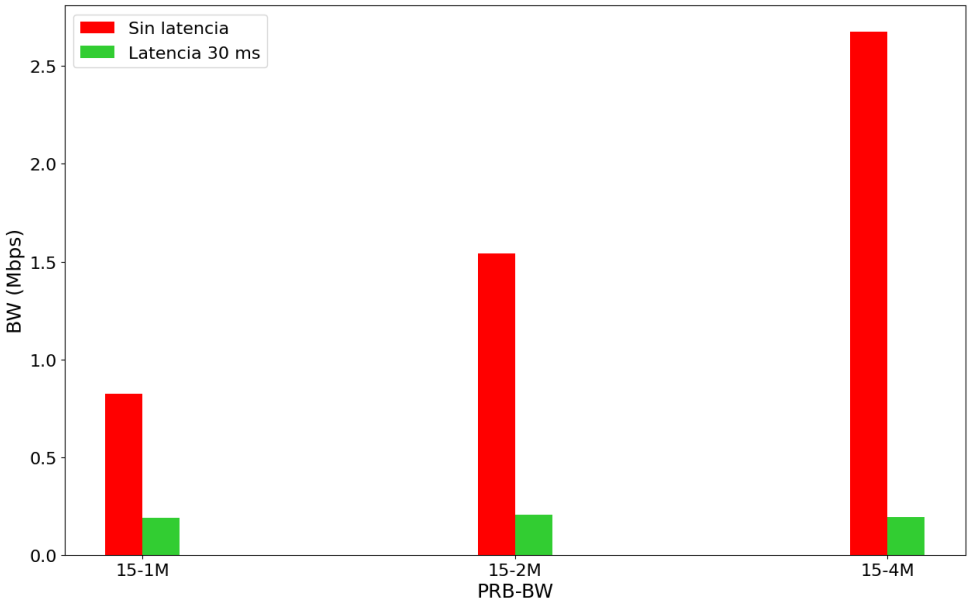


Figura A.56: Tráfico enviado por *rlc_mac_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL

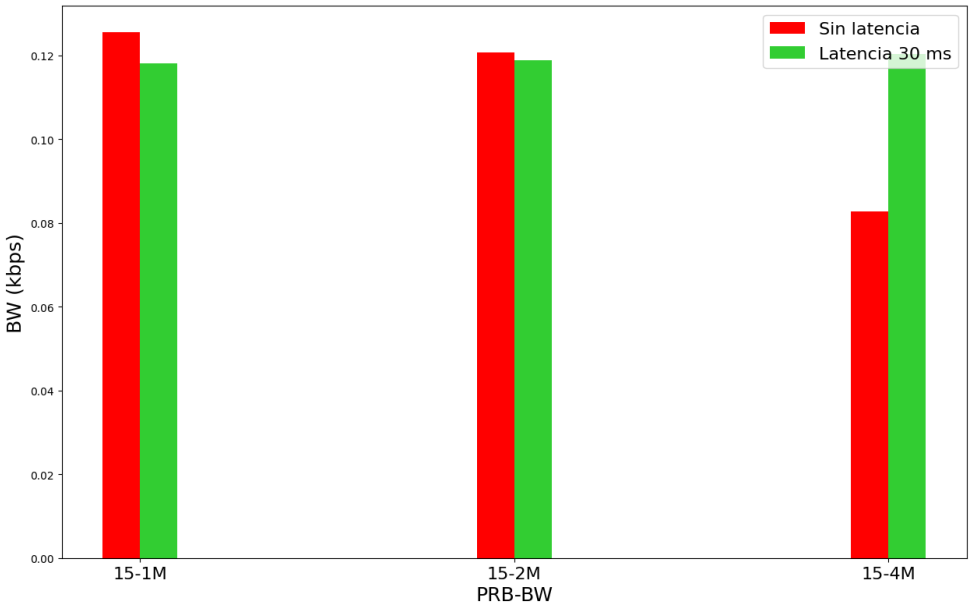


Figura A.57: Tráfico enviado por *rlc_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL

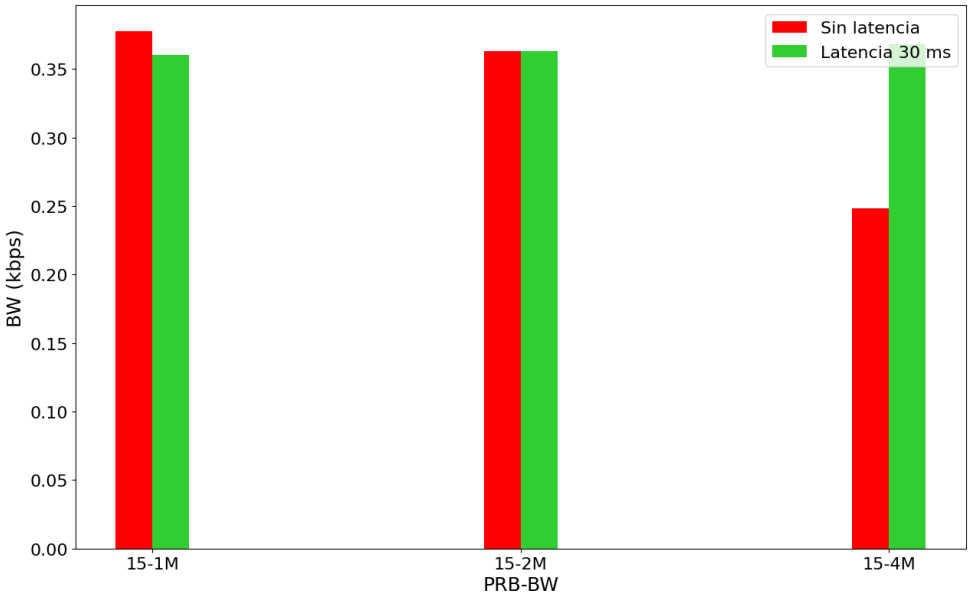


Figura A.58: Tráfico enviado por *rrc_rlc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL

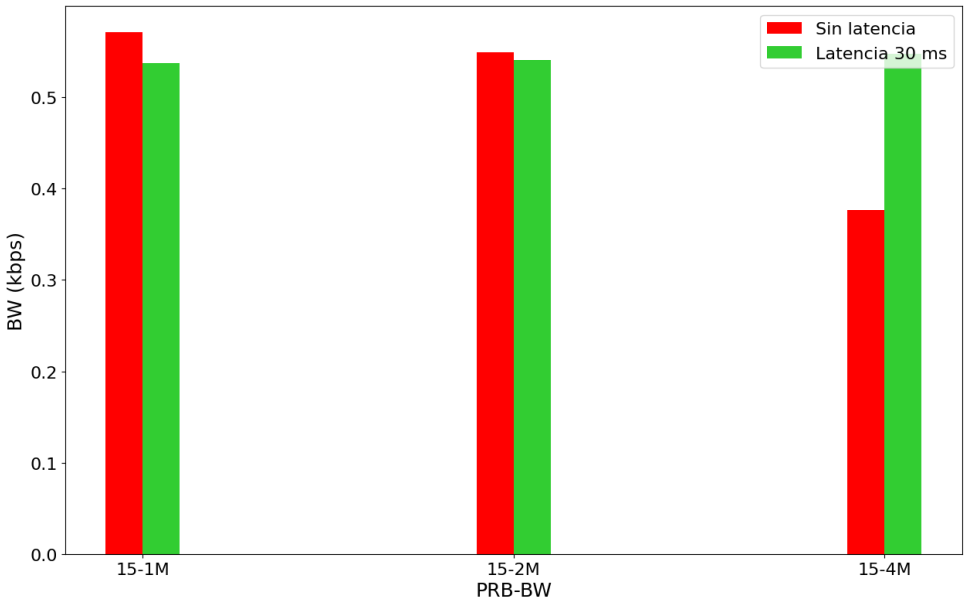


Figura A.59: Tráfico enviado por *pdcp_rrc_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL

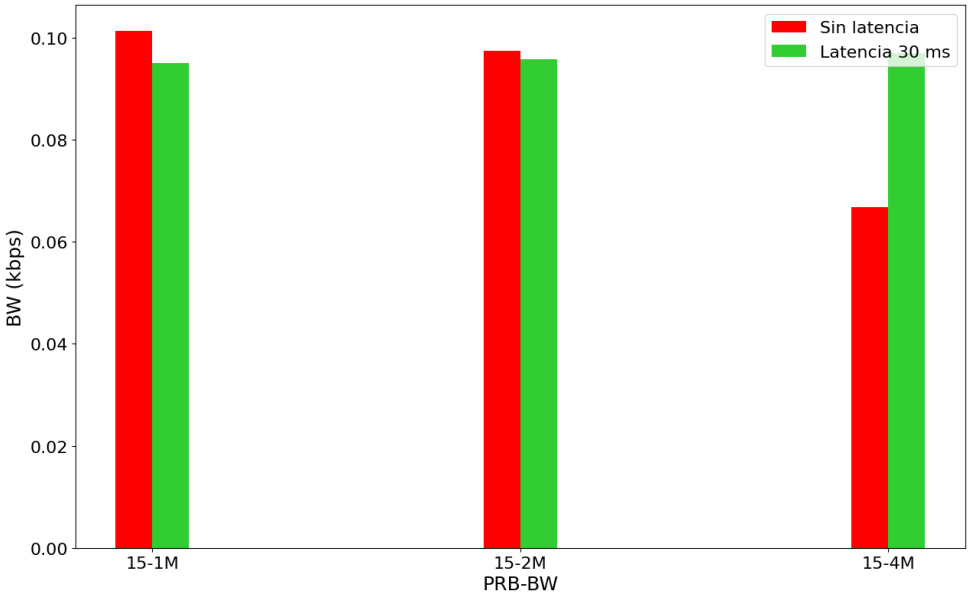


Figura A.60: Tráfico enviado por *rrc_pdcp_wrapper* en el *split RLC_MAC_SPLIT* mientras se envía tráfico DL