ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Modelado de tráfico fronthaul para redes 5G y 6G

(On the modelling of fronthaul traffic for 5G and 6G networks)

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autora: Irma Irastorza Gutiérrez

Julio- 2025

ETS de Ingenieros Industriales y de Telecomunicación

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Irma Irastorza Gutiérrez
Director del TFG: Ramón Agüero Calvo, Luis Francisco Diez Fernández
Título: "Modelado de tráfico fronthaul para redes 5G y 6G"
Title: "On the modelling of fronthaul traffic for 5G and 6G networks"

Presentado a examen el día: 24 de Julio de 2025

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del tribunal:
Presidente (Apellidos, Nombre): Suárez Rodríguez, Almudena
Secretario (Apellidos, Nombre): García Gutiérrez, Alberto Eloy
Vocal (Apellidos, Nombre): Torres Jiménez, Rafael Pedro

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

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

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

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

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

Agradecimientos

A mis padres, y en especial a mi padre, por su apoyo incondicional y por haber sido un guía y mentor académico a lo largo de esta carrera.

A Tairé, compañera incansable durante estos cuatro años, por compartir no solo la carrera, sino también una valiosa amistad.

A Llaguno, por su paciencia y por acompañarme durante todo el proceso de realización de este TFG.

A Sofía, Noiva y Paula, amigas de toda la vida, por estar ahí para consolarme y apoyarme desde fuera.

Finalmente, quiero agradecer a mis tutores, Ramón, Luis y Fátima, por el seguimiento constante, la facilidad para atender mis consultas y la confianza que han mantenido en mí durante estos meses intensos de trabajo.

Resumen

Nos encontramos ante una transformación inevitable en la arquitectura de las redes de acceso radio de nueva generación, impulsada por el crecimiento constante del tráfico y la complejidad de los servicios que deben soportar las estaciones base. Durante años, estas unidades monolíticas asumieron todas las funciones, pero las exigencias actuales reclaman un cambio profundo que permita mayor flexibilidad y eficiencia. Surge así el paradigma de la desagregación, apoyado en técnicas de virtualización capaces de desplegar funciones de red sobre hardware genérico, incluso de forma distribuida y dinámica, reduciendo costes y optimizando recursos. Sin embargo, ¿es realmente viable centralizar todas las funciones sin comprometer el rendimiento? Para dar una respuesta resulta imprescindible analizar la red fronthaul, ese segmento crítico que conecta las funciones distribuidas y que debe garantizar prioridades estrictas y bajas latencias sin afectar al resto del tráfico. En este trabajo se desarrolla una aplicación propia sobre el simulador `NS-3`, denominada `CustomApplication`, escrita en `C++` y diseñada para reproducir y analizar patrones de tráfico en enlaces fronthaul desagregados. La herramienta permite generar tráfico UDP ajustable y registrar eventos detallados para su posterior procesamiento y comparación con modelos analíticos de teoría de colas (M/M/1, M/D/1, G/G/1 y aproximación de Kingman). Se han implementado dos modos de operación: un modelo sintético basado en distribuciones estadísticas y un modelo realista fundamentado en parámetros físicos como numerología, ancho de banda, capas MIMO y ciclos TDD. Los resultados obtenidos permiten conocer la aplicabilidad de los modelos teóricos y aportan una base sólida para la optimización del tráfico fronthaul en arquitecturas 5G y 6G, consolidando así un marco de referencia para futuras investigaciones y despliegues reales.

Palabras clave: redes 5G, redes 6G, fronthaul, desagregación, virtualización, `NS-3`, generación de tráfico, teoría de colas.

Abstract

We are facing an inevitable transformation in the architecture of next-generation radio access networks, driven by the constant growth of traffic and the increasing complexity of services that base stations must support. For years, these monolithic units handled all functions, but current demands call for a profound change that enables greater flexibility and efficiency. This gives rise to the paradigm of disaggregation, supported by virtualization techniques capable of deploying network functions on generic hardware, even in a distributed and dynamic manner, thus reducing costs and optimizing resources. However, is it truly feasible to centralize all functions without compromising performance? To answer this question, it is essential to analyze the fronthaul, that critical segment that connects the distributed functions and must ensure strict priorities and low latencies without impacting the rest of the traffic. In this work, a custom application has been developed on the `ns-3` simulator, called `CustomApplication`, written in `C++` and designed to reproduce and analyze traffic patterns in disaggregated fronthaul links. The tool allows for the generation of adjustable UDP traffic and the logging of detailed events for subsequent processing and comparison with analytical queueing theory models (M/M/1, M/D/1, G/G/1, and Kingman's approximation). Two modes of operation have been implemented: a synthetic model based on statistical distributions and a realistic model based on physical parameters such as numerology, bandwidth, MIMO layers, and TDD cycles. The results obtained allow assess the applicability of the analytical models and provide a solid foundation for optimizing fronthaul traffic in 5G and 6G architectures, thus establishing a reference framework for future research and real-world deployments.

Keywords: 5G networks, 6G networks, fronthaul, disaggregation, virtualization, ns-3, traffic generation, queuing theory.

Índice general

Índice de figuras

Índice de tablas

Lista de códigos

Índice de acrónimos

3GPP	3rd Generation Partnership Project.
5G	Quinta Generación.
5G NR	5G New Radio.
6G	Sexta Generación.
API	Application Programming Interface.
BBU	Baseband Unit.
BFP9	Block Floating Point 9 bits.
CDF	Cumulative Distribution Function.
CPRI	Common Public Radio Interface.
C-RAN	Centralized Radio Access Network.
CU	Unidad Centralizada.
DU	Unidad Distribuida.
eCPRI	enhanced Common Public Radio Interface.
eMBB	Enhanced Mobile Broadband.
I/Q	In-phase and Quadrature.
IoT	Internet de las Cosas.
IP	Internet Protocol.
KPI	Key Performance Indicator.
MAC	Media Access Control.
Massive MIMO	Massive Multiple Imput Multiple Output.
MIMO	Multiple Imput Multiple Output.
mMTC	massive Machine-Type Communications.

NRB	Number of Resource Blocks.
OFDM	Orthogonal Frequency Division Multiplexing.
PDF	Probability Density Function.
PRB	Physical Resource Block.
QoS	Calidad de servicio.
RAN	Red de acceso.
RoE	Radio over Ethernet.
RU	Unidad de Radio.
SCS	Subcarrier Spacing.
SDN	Software-Defined Networking.
SRv6	Segment Routing.
SyncE	Synchronous Ethernet.
TCP	Transmission Control Protocol.
TDD	Time Division Duplex.
TSN	Time-Sensitive Networking.
UDP	User Datagram Protocol.
URLLC	Ultra-Reliable Low-Latency Communications.

Capítulo 1

Introducción

1.1. Motivación

La evolución de las redes móviles ha estado impulsada históricamente por la necesidad de adaptarse al crecimiento sostenido del tráfico de datos y a la aparición de servicios cada vez más exigentes. Aplicaciones como el Internet de las Cosas (IoT), los vehículos autónomos, la industria 4.0, el streaming en ultra alta definición y los entornos inmersivos de realidad aumentada o virtual, demandan características clave como baja latencia, alta fiabilidad y la capacidad de gestionar millones de dispositivos conectados de forma simultánea [1]. Esta presión constante sobre las infraestructuras actuales ha puesto de manifiesto la necesidad de un nuevo paradigma en el diseño y gestión de redes móviles.

En este contexto, las redes Quinta Generación (5G) / Sexta Generación (6G) no suponen únicamente una mejora en velocidad respecto a generaciones anteriores, sino una profunda renovación en su arquitectura. Una de las innovaciones fundamentales es la desagregación funcional de Red de acceso (RAN) en tres bloques: Unidad de Radio (RU), Unidad Distribuida (DU) y Unidad Centralizada (CU) [2] [3]. Esta separación permite mejorar la escalabilidad, optimizar la asignación de recursos y adaptar la red a distintos perfiles de tráfico. Además, se incorporan tecnologías avanzadas como la multiplexación masiva, Massive Multiple Imput Multiple Output (Massive MIMO), la utilización flexible de múltiples numerologías, entendida como el conjunto de valores que caracterizan la configuración de la capa física, en particular el espaciado entre subportadoras, el tiempo de cada símbolo y la extensión del prefijo cíclico dentro de un sistema OFDM, y modos de operación dinámicos como el Time Division Duplex (TDD) adaptativo [4] [5].

Desde el punto de vista técnico, la magnitud de los retos es considerable. Por ejemplo, en escenarios experimentales con redes 5G implementadas mediante la plataforma srsRAN, se han reportado tasas de tráfico del plano de usuario superiores a los 1474 Mbps por puerto de antena, utilizando una configuración de 100 MHz de ancho de banda y un espaciado entre subportadoras de 30 kHz [6]. Estas cifras reflejan no solo la capacidad del sistema, sino también la necesidad crítica de un diseño eficiente del plano de transporte y de una planificación adecuada del tráfico, para evitar cuellos de botella y asegurar que se cumplan los requisitos de Calidad de servicio (QoS).

Para afrontar este desafío, resulta esencial contar con herramientas que permitan analizar, validar y optimizar el comportamiento de la red antes de su despliegue real. Los entornos de simulación, como

el simulador de redes NS−3 que se va a usar en este trabajo, proporcionan una plataforma flexible para modelar configuraciones realistas, explorar distintas políticas de tráfico y experimentar con condiciones límite sin los elevados costes de una infraestructura física.

No obstante, para que las simulaciones sean interpretables y comparables con el comportamiento real de un sistema, deben estar respaldadas por un marco teórico riguroso. La teoría de colas ofrece este soporte analítico, a través de modelos como M/M/1, M/G/1, que permiten caracterizar el rendimiento del sistema ante distintos patrones de llegada y servicio. Estas herramientas son fundamentales para obtener métricas como el tiempo medio de espera, la utilización del canal o la probabilidad de pérdida, y ofrecen una referencia clara frente a la cual validar los resultados obtenidos por simulación.

En este trabajo se estudia precisamente esa intersección entre teoría y simulación. A través del diseño de escenarios experimentales en NS−3, se analiza el comportamiento de un nodo dentro de una red 5G / 6G en condiciones controladas. Se registran con precisión los instantes de envío y recepción de los paquetes, lo que permite construir distribuciones empíricas (Probability Density Function (PDF), Cumulative Distribution Function (CDF)) y extraer métricas relevantes como los tiempos de transmisión del sistema. Posteriormente, estos resultados se comparan con los valores teóricos proporcionados por los modelos analíticos previamente mencionados, con el objetivo de evaluar su validez en entornos simulados, pero realistas.

Asimismo, el estudio contempla cómo distintas configuraciones del tráfico, como el tamaño de los paquetes, el intervalo entre envíos o el número de capas MIMO, afectan al comportamiento del sistema desde una perspectiva probabilística. Este enfoque permite no solo entender con mayor profundidad el funcionamiento interno de estas redes, sino también proponer criterios para el dimensionamiento adecuado de los recursos, la detección temprana de congestión y la garantía de calidad de servicio. En última instancia, se busca contribuir al diseño de redes más robustas, adaptativas y preparadas para afrontar los desafíos del ecosistema digital moderno.

1.2. Objetivos

Para alcanzar el propósito general del trabajo, se establecen los siguientes objetivos específicos:

1. Diseñar una aplicación personalizada en NS−3 desde cero con dos enfoques. El primero es un enfoque sintético, en el que la aplicación se forma con variables aleatorias tanto para el tamaño de los paquetes como para los tiempos entre llegadas. El segundo enfoque consiste en diseñar la aplicación para permitir la configuración de los parámetros típicos de un nodo 5G / 6G (numerología, tráfico, ancho de banda, etc.) y simular el envío de paquetes representativos del tráfico fronthaul, entendido como el flujo de datos que circula entre RU y DU en arquitecturas desagregadas.

2. Calcular métricas estadísticas relevantes, como funciones, PDF, CDF y tiempos de transmisión, para caracterizar el tráfico y la respuesta del nodo ante distintas condiciones.

3. Aplicar modelos teóricos de teoría de colas, como pueden ser los modelos M/M/1, M/G/1, G/G/1 y la aproximación de Kingman para analizar el sistema desde una perspectiva matemática y compararlo con los resultados empíricos.

4. Evaluar el impacto de las variaciones en los parámetros de tráfico sobre el rendimiento del nodo, proporcionando una visión cuantitativa útil para el diseño, dimensionamiento y optimización de redes móviles de nueva generación.

1.3. Estructura del documento

Para facilitar la comprensión del enfoque adoptado y del desarrollo realizado, este documento se organiza en varios capítulos que abordan de forma progresiva los distintos aspectos del trabajo. Se parte del contexto teórico y tecnológico, para luego describir con detalle el diseño del entorno de simulación, su implementación y la posterior evaluación de los resultados obtenidos. A continuación, se presenta la estructura del documento:

- El capítulo 1 introduce la motivación del estudio, exponiendo los retos actuales en el ámbito de las redes móviles y justificando la relevancia de aplicar simulación y teoría de colas en el contexto de las tecnologías 5G y 6G. Además, se recogen los objetivos generales y específicos del trabajo, así como la estructura de la memoria.
- En el capítulo 2 se desarrollan los fundamentos teóricos y tecnológicos que sustentan el proyecto. Se parte del análisis del proyecto 6GBLUR, una iniciativa orientada a transformar la arquitectura del plano de transporte en redes móviles de próxima generación. A continuación, se abordan los conceptos centrales de 5G, se desglosa su arquitectura desagregada (RU, DU, CU), y se analizan elementos clave como el fronthaul y la multiplexación masiva. Finalmente, se introduce la teoría de colas como herramienta analítica para modelar el comportamiento del sistema bajo distintas condiciones de carga, sentando las bases para su posterior comparación con los resultados obtenidos por simulación.
- El capítulo 3 presenta el diseño del entorno de simulación desarrollado en `NS-3`, incluyendo la implementación de una aplicación configurable para generar tráfico en redes 5G y 6G. Se describen dos enfoques de modelado (aleatorio y determinista), así como la topología empleada, los parámetros clave y los mecanismos de captura de métricas utilizados para el posterior análisis del rendimiento.
- En el capítulo 4 se presentan y analizan los resultados obtenidos a partir de las simulaciones realizadas. Se examinan métricas como los tiempos de llegada y transmisión de los paquetes, así como las distribuciones correspondientes, comparando los datos empíricos con los resultados teóricos proporcionados por los modelos de colas estudiados.
- El capítulo 5 recoge las conclusiones extraídas del trabajo, evaluando el cumplimiento de los objetivos propuestos y reflexionando sobre las limitaciones y posibles mejoras del estudio. Finalmente, se proponen líneas de trabajo futuro orientadas a profundizar en la modelización del tráfico y en la validación de otros modelos teóricos bajo nuevas configuraciones de red.

Capítulo 2

Fundamentos teóricos y tecnológicos

Este capítulo expone los fundamentos tecnológicos y analíticos que sirven de base para contextualizar los retos abordados en este trabajo. En primer lugar, se presentan iniciativas actuales como el proyecto 6GBLUR [1], orientado a transformar la arquitectura de red para responder a los exigentes requisitos de las futuras redes 6G, con especial atención en el segmento fronthaul y a la virtualización de funciones de red. Sobre esta base, se analizan a continuación las características de las redes 5G desagregadas, describiendo su estructura modular basada en RU-DU-CU, así como la organización de recursos en el segmento radio y su influencia directa en la planificación del plano de transporte.

Posteriormente, se introduce la teoría de colas como herramienta analítica fundamental para comprender el comportamiento de sistemas de comunicación sujetos a tráfico variable. Esta disciplina permite caracterizar aspectos como la dinámica de llegada y procesamiento de paquetes, así como establecer modelos que permiten analizar el rendimiento del sistema bajo distintas condiciones de carga. Su aplicación proporciona una base sólida para interpretar métricas clave y contrastar los resultados de simulación con comportamientos teóricos esperados.

[1] https://6g-blur.cttc.es/

2.1. Proyecto 6GBLUR

El proyecto 6GBLUR representa una iniciativa clave orientada a reconfigurar la arquitectura del plano de transporte, con un enfoque centrado en la integración de tecnologías abiertas, la virtualización de funciones de red y el cumplimiento de estrictos requisitos de QoS en el fronthaul, con el propósito final de preparar las redes móviles y anticipar los desafíos de la futura generación 6G. Este tramo crítico de la red, que conecta la RU con la DU en arquitecturas desagregadas, debe ser capaz de gestionar tráficos extremadamente exigentes en capacidad, latencia, sincronización y fiabilidad. La Figura 2.1 ilustra un ejemplo representativo de esta arquitectura desagregada, donde múltiples nodos O-RU se conectan mediante una red de transporte priorizada hacia una DU centralizada.

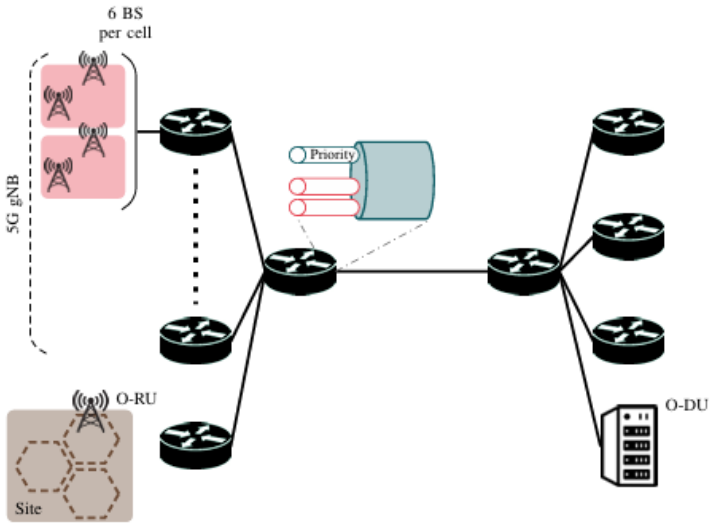


Figura 2.1: Transporte priorizado entre O-RU y O-DU en red 5G desagregada. Fuente: [6]

Para lograr este objetivo, la arquitectura contempla tres planos funcionales, que se traducen en tipos de tráfico diferentes en el segmentl fronthaul: el plano de datos (user plane), el plano de control y el plano de gestión, cada uno con mecanismos específicos que contribuyen a mejorar el transporte de información entre la RU y la DU.

Plano de datos: transmisión eficiente y determinista

El plano de datos propuesto por 6GBLUR está concebido para garantizar una transmisión determinista, eficiente y escalable entre los distintos elementos de la red. Para ello, se introducen varios mecanismos clave. En primer lugar, se emplea una estrategia de agregación de tráfico mediante túneles dedicados, que encapsulan los flujos con parámetros homogéneos de QoS, permitiendo así tratarlos de forma diferenciada según sus requisitos de latencia o tolerancia al retardo.

Además, se incorpora el scheduling (planificación) basado en tiempo (Time-Aware Scheduling), apoyado en tecnologías como Time-Sensitive Networking (TSN) y Segment Routing (SRv6), que permite reservar franjas temporales específicas para asegurar la entrega puntual de los paquetes. La priorización del tráfico también juega un papel central en esta arquitectura, mediante el uso de colas jerárquicas que asignan recursos a los paquetes en función del tipo de servicio, como Enhanced Mobile Broadband (eMBB), Ultra-

Reliable Low-Latency Communications (URLLC), massive Machine-Type Communications (mMTC), optimizando así el uso del enlace según la criticidad del flujo.

Por último, se aplica compresión funcional mediante el algoritmo Block Floating Point 9 bits (BFP9), lo que permite reducir de forma significativa el volumen de datos transportado al comprimir flujos In-phase and Quadrature (I/Q). Este enfoque permite gestionar señales fronthaul multicanal Multiple Imput Multiple Output (MIMO) con una eficiencia mucho mayor que en los esquemas tradicionales basados en muestras sin comprimir de 16 bits.

Plano de control: gestión dinámica de recursos

El plano de control habilita una orquestación dinámica de los recursos de red, respondiendo en tiempo real a las variaciones en la carga o cambiando en función de los perfiles de servicio. Entre sus funciones más relevantes se encuentra la monitorización continua del enlace fronthaul, que permite adaptar de forma dinámica el nivel de compresión, la granularidad del tráfico o la configuración de los túneles en función de la demanda del sistema.

Asimismo, este plano contempla la asignación de recursos por slot, considerando variables como la calidad del canal o la configuración de la capa física, lo que permite un uso más eficiente y contextualizado del transporte. Por último, se integra con controladores Software-Defined Networking (SDN), lo que posibilita la aplicación de políticas de red definidas por software, permitiendo modificar la topología lógica o reencaminar flujos de tráfico sin interrumpir el servicio.

Plano de gestión: supervisión agregada y optimización

El plano de gestión proporciona una visión agregada y a largo plazo del comportamiento de la red, aspecto fundamental para garantizar su evolución, mantenimiento y adaptación continua. Una de sus funciones principales es la configuración de topologías lógicas específicas para cada escenario de despliegue, como por ejemplo entornos urbanos de alta densidad o redes industriales con requisitos críticos.

Asimismo, este plano permite aplicar mecanismos de slicing sobre el fronthaul, lo que posibilita la reserva de recursos físicos diferenciados para distintos servicios o para operadores virtuales independientes. Además, desempeña un papel clave en el análisis de Key Performance Indicator (KPI), incorporando técnicas de inteligencia artificial orientadas a anticipar posibles eventos como congestiones, fallos o variaciones significativas en los patrones de tráfico.

En conjunto, la arquitectura planteada por 6GBLUR sienta las bases para abordar las exigencias del transporte fronthaul en redes móviles de nueva generación. A continuación, se analizan en mayor profundidad las características de las redes 5G desagregadas, sobre las cuales se apoya esta visión evolutiva.

2.2. La red fronthaul en arquitecturas 5G desagregadas

En las redes 5G desagregadas, el fronthaul representa el segmento de red encargado de conectar la Unidad de Radio RU con la Unidad Distribuida DU, permitiendo una separación funcional entre la capa física y el resto del procesamiento de señal. Esta arquitectura, inspirada en el paradigma Centralized Radio Access Network (C-RAN), favorece una gestión más flexible y escalable de los recursos al trasladar ciertas funciones a entornos virtualizados de alto rendimiento. En este contexto, las funciones clásicas de la Baseband Unit (BBU) se dividen entre dos bloques lógicos: CU, que centraliza las tareas de nivel superior, y DU, ubicada próxima a la RU y encargada de funciones de menor complejidad [7] [2] [8].

Este reparto funcional se articula mediante el concepto de functional split, que determina en qué punto exacto se dividen las funcionalidades entre cada unidad. Su correcta elección permite encontrar un equilibrio entre la eficiencia computacional, los requisitos de sincronización y el consumo de ancho de banda en el enlace de transporte. Entre las distintas opciones propuestas, el split 7.2x destaca por su amplia adopción, ya que permite descargar parte del procesamiento a la DU sin imponer exigencias excesivas al enlace fronthaul [9].

El fronthaul, por tanto, debe ser capaz de transportar flujos de datos altamente sensibles a la latencia, como las muestras I/Q, así como la señalización y la sincronización necesarias para garantizar un funcionamiento coherente entre nodos. Para ello, se emplean enlaces ópticos de alta capacidad combinados con protocolos especializados como enhanced Common Public Radio Interface (eCPRI) o Radio over Ethernet (RoE), que ofrecen una eficiencia notablemente superior frente a estándares heredados como el protocolo Common Public Radio Interface (CPRI) clásico [10].

Uno de los retos técnicos más exigentes es garantizar una latencia unidireccional inferior a los 250 microsegundos en configuraciones con split 7.2x, además de mantener una sincronización precisa mediante mecanismos como Synchronous Ethernet (SyncE), esenciales para coordinar adecuadamente los distintos elementos de la red desagregada [11].

Para ilustrar esta partición funcional, la Figura 2.2 muestra la distribución de tareas entre la DU y la RU en el contexto del split 7.2x, señalando qué bloques del procesamiento físico se ejecutan a cada lado del enlace.

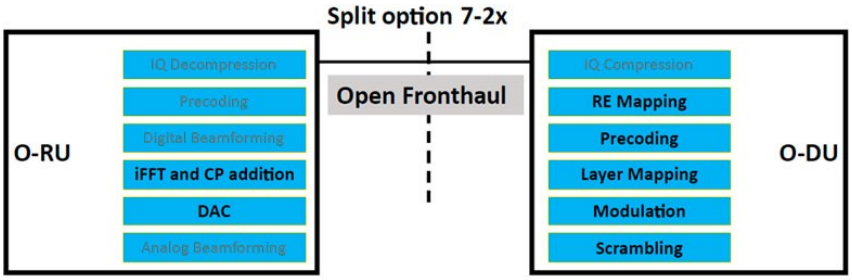


Figura 2.2: Reparto funcional entre la O-DU y la O-RU bajo la opción de split 7.2x. Fuente: [12]

Desde el punto de vista del tráfico, el fronthaul debe soportar un volumen considerable de datos, especialmente en configuraciones MIMO y escenarios con agregación de portadoras. Esta carga es, además, altamente variable, dependiendo de parámetros como la numerología, el espaciado entre subportadoras y la duración de los slots.

En 5G/NR se admiten múltiples numerologías, entendidas como distintas configuraciones de la forma de onda. Estas numerologías implican que la estructura de la trama de radio varíe ligeramente según el tipo seleccionado, aunque existen ciertos elementos fijos: la duración de una trama de radio es siempre de 10 ms y la de una subtrama es siempre de 1 ms, independientemente de la numerología empleada. Lo que sí cambia es el número de ranuras que se colocan dentro de cada subtrama, ya que este se adapta a las propiedades físicas definidas por la numerología. Por otro lado, el número de símbolos dentro de cada ranura no depende de la numerología, sino del tipo de configuración de la ranura. En concreto, para una ranura con prefijo cíclico normal, el número de símbolos por ranura es siempre 14, valor usado en este trabajo.

La Tabla 2.1 resume los parámetros correspondientes a la estructura de la trama radio en función de la numerología μ, mostrando el número de símbolos por ranura, el número de ranuras por trama y el número de ranuras por subtrama.

μ	$N^{\text{slot}}_{\text{symb}}$	$N^{\text{frame},\mu}_{\text{slot}}$	$N^{\text{subframe},\mu}_{\text{slot}}$
0	14	10	1
1	14	20	2
2	14	40	4
3	14	80	8
4	14	160	16

Tabla 2.1: Parámetros de la estructura de la trama de radio en función de la numerología. Fuente: [13]

En el caso específico de la numerología cero ($\mu = 0$), el espaciado entre subportadoras es de 15 kHz y la organización temporal de la trama adopta su forma más sencilla. Tal como se aprecia en la Figura 2.3, una trama radio tiene siempre una duración de 10 ms y se compone de 10 subtramas consecutivas. En esta numerología cada subtrama corresponde exactamente a un slot, por lo que el número total de slots por frame también es 10, cada uno ocupando 1 ms. Si se observa con mayor detalle, cada slot está constituido por 14 símbolos OFDM cuando se emplea un prefijo cíclico normal, distribuidos de manera uniforme a lo largo de ese milisegundo. Esta configuración proporciona una estructura temporal estable y ampliamente utilizada como referencia en los despliegues de 5G, ofreciendo una buena cobertura y robustez, aunque con menor flexibilidad para servicios que requieren una granularidad temporal más fina.

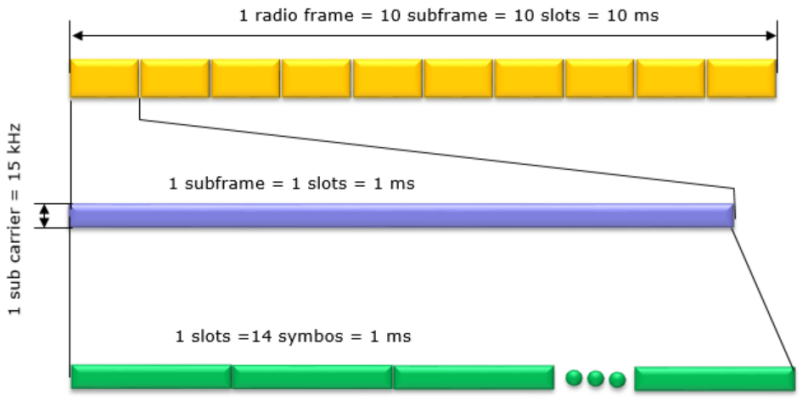


Figura 2.3: Estructura temporal de la trama de radio para la numerología cero ($\mu = 0$).

Además, esta estructura radio condiciona de forma directa la forma en que se generan y transportan los paquetes en el enlace fronthaul, ya que el tráfico a transmitir debe acomodarse a la granularidad temporal de los slots y a las capacidades físicas definidas por la numerología seleccionada. Por ello, el diseño de este tramo de red requiere mecanismos de adaptación dinámica que permitan mantener la calidad del servicio ante fluctuaciones abruptas de carga.

Para dar respuesta a estos retos, se han desarrollado soluciones como la compresión funcional de flujos I/Q, la implementación de colas jerárquicas con priorización según el tipo de tráfico (eMBB, URLLC, etc.), técnicas de scheduling sensible al tiempo y mecanismos de slicing para segmentar el enlace entre distintos servicios u operadores virtuales. Estas técnicas no solo permiten mantener los requisitos de calidad, sino también optimizar la eficiencia del uso del canal.

En este trabajo, se analiza el comportamiento del tráfico fronthaul generado por un entorno 5G o 6G que incluye múltiples estaciones base. Mediante simulaciones realizadas con el simulador `NS-3`, se estudian métricas como los tiempos de transmisión, la distribución estadística del tamaño de los paquetes o la capacidad del enlace, con el objetivo de caracterizar el impacto de dicho tráfico sobre el rendimiento de la red.

Cabe destacar que, según estimaciones del IEEE 1914 Working Group, la aplicación de compresión funcional puede reducir el uso de ancho de banda en el fronthaul hasta en un 25 %, lo que a su vez se traduce en un ahorro energético cercano al 30 % en la infraestructura de transporte [11]. Este tipo de avances refuerza la importancia de modelar con precisión este tramo de la red para optimizar tanto el rendimiento como la eficiencia.

2.3. Gestión del tráfico y análisis de red

En las redes 5G desagregadas, y en particular en el segmento fronthaul el tráfico no solo se caracteriza por un elevado caudal de datos, sino también por su marcada variabilidad temporal y su sensibilidad extrema a la latencia. La gestión eficiente de dicho tráfico resulta crucial para garantizar el cumplimiento de los objetivos de QoS, especialmente en escenarios heterogéneos donde conviven flujos con requisitos profundamente dispares, como los asociados a comunicaciones eMBB, URLLC, mMTC.

Dicho tráfico presenta fluctuaciones dinámicas en volumen, prioridad y criticidad, lo que exige mecanismos avanzados de control capaces de adaptarse en tiempo real. Para afrontar esta complejidad, la red incorpora políticas de clasificación que discriminan los paquetes según sus atributos de QoS, técnicas de priorización orientadas a satisfacer los requisitos más estrictos, estrategias de asignación dinámica de recursos en función de la carga y algoritmos de planificación y control de admisión que permiten mitigar la congestión [14].

La correcta operación de estos mecanismos se apoya en una monitorización exhaustiva del comportamiento interno del sistema, basada en indicadores clave de rendimiento. Estos no solo permiten verificar si se mantienen las garantías de servicio, sino que además sirven como referencia para ajustar dinámicamente la configuración de red. En el marco de este trabajo, donde se emplea el simulador `NS-3` para registrar métricas como los tiempos de transmisión y recepción de paquetes, los KPI se convierten en herramientas esenciales para cuantificar el rendimiento efectivo del sistema y detectar potenciales cuellos de botella.

No obstante, el comportamiento de este tipo de tráfico al agregarse flujos de varias estaciones base no

puede abordarse desde una perspectiva determinista, ya que las llegadas de paquetes, los tiempos de procesamiento y las interacciones entre nodos están sujetos a un elevado grado de aleatoriedad inherente al propio entorno de comunicaciones. En este contexto, resulta indispensable recurrir a modelos matemáticos estocásticos como los proporcionados por la teoría de colas.

2.4. Modelado estocástico del tráfico y teoría de colas

Para comprender el comportamiento del tráfico, en arquitecturas desagregadas 5G y 6G como se ha comentado anteriormente, resulta imprescindible apoyarse en un marco teórico que permita modelar y predecir el rendimiento del sistema bajo condiciones de carga variable. En este contexto, la teoría de colas se establece como una herramienta esencial, ya que ofrece modelos analíticos capaces de describir el flujo de paquetes y evaluar métricas clave como el tiempo medio de espera, la utilización de los recursos o la probabilidad de congestión.

A continuación se introducen los conceptos esenciales de esta disciplina aplicados al tráfico aleatorio, incluyendo la caracterización de procesos de llegada y servicio, la definición de métricas como la carga del sistema y el tiempo medio de servicio, la notación formal empleada para describir distintos tipos de colas y la relación de Little, así como los principales modelos analíticos utilizados para evaluar el rendimiento de nodos y enlaces en redes.

2.4.1. Fundamentos del tráfico aleatorio en redes

La Tabla 2.2 resume los elementos fundamentales que definen un sistema de colas, junto con su notación y representación habitual en la literatura. Esta clasificación resulta útil para comprender las estructuras analíticas que se emplean en el modelado de sistemas de comunicaciones.

Tabla 2.2: Parámetros fundamentales de un sistema de colas

Elemento	**Símbolo / Notación**	**Descripción**
Proceso de llegadas	λ	Tasa media de llegada de paquetes al sistema
Proceso de servicio	μ	Tasa media de servicio del servidor
Número de servidores	s	Cantidad de servidores disponibles simultáneamente
Capacidad del sistema	K	Número máximo de paquetes permitidos en el sistema
Población fuente	M	Número de fuentes generadoras de paquetes (si es finita)
Disciplina de cola	FIFO, LIFO, etc.	Política de orden en que se procesan los paquetes en espera

2.4.2. Carga del sistema y tiempo medio de servicio

En el análisis de sistemas de colas, uno de los parámetros fundamentales para caracterizar el rendimiento del sistema es la carga, denotada habitualmente por ρ. Este valor adimensional representa el grado de utilización del recurso de comunicaciones (p.e. un servidor o un interfaz), esto es, la proporción de tiempo durante la cual se encuentra ocupado atendiendo solicitudes. Formalmente, la expresión (2.1) se define como el producto entre la tasa media de llegadas λ (medida en paquetes por segundo) y el tiempo medio de servicio $\mathbb{E}[S]$.

$$\rho = \lambda \cdot \mathbb{E}[S] \tag{2.1}$$

Este parámetro actúa como variable de control crítica, ya que determina en gran medida la estabilidad del sistema: para que una cola no crezca indefinidamente, es necesario que $\rho < 1$, es decir, que el servidor sea capaz de atender las peticiones a un ritmo superior al de llegada.

El tiempo medio de servicio $\mathbb{E}[S]$ depende del tamaño de los paquetes que transitan por el sistema y de la capacidad de transmisión del canal. Si se considera un paquete de tamaño medio L (en bits) y un enlace con capacidad C (en bits por segundo), entonces el tiempo necesario para transmitir dicho paquete se estima como la relación (2.2).

$$\mathbb{E}[S] = \frac{L}{C} \tag{2.2}$$

Esta relación permite traducir características físicas del sistema, como el tamaño medio de los paquetes o la capacidad del enlace, en parámetros analíticos clave dentro del marco de la teoría de colas. La carga ρ actúa como variable crítica que determina el comportamiento del sistema: a medida que se aproxima a la unidad, el retardo crece de forma no lineal, indicando una progresiva saturación y deterioro del rendimiento [15].

2.4.3. Notación de Kendall y Relación de Little

La clasificación formal de los sistemas de colas se realiza habitualmente mediante la notación de Kendall, representada como A/B/C/D/E/F. En esta nomenclatura, el primer parámetro (A) indica la distribución del tiempo entre llegadas, donde es común utilizar ‘M’ para distribuciones exponenciales, ‘D’ para deterministas y ‘G’ para distribuciones generales. El segundo parámetro (B) describe la distribución del tiempo de servicio bajo los mismos criterios. El tercer elemento (C) especifica el número de servidores (recursos) disponibles en el sistema. El cuarto (D) representa la capacidad total del sistema, incluyendo tanto los clientes en servicio como los que están en espera. El quinto (E) hace referencia al tamaño de la población fuente, es decir, al número máximo de clientes que pueden generar solicitudes. Por último, el parámetro (F) define la disciplina de servicio en la cola, como por ejemplo FIFO (First-In, First-Out), LIFO (Last-In, First-Out) o esquemas con prioridad [16].

Una de las herramientas más fundamentales en teoría de colas es la Relación de Little, ilustrada en la expresión (2.3), que establece que el número medio de paquetes en el sistema L es igual al producto entre la tasa media de llegada λ y el tiempo medio que un paquete permanece en el sistema W [17].

$$L = \lambda \cdot W \tag{2.3}$$

Esta relación, válida bajo condiciones generales de estabilidad y estacionariedad, se aplica tanto al sistema completo como exclusivamente a la cola. Es una expresión especialmente útil para validar resultados analíticos y simulados, al permitir calcular tiempos de permanencia a partir de valores medios de ocupación.

2.4.4. Modelos clásicos aplicables en 5G y 6G

A partir de estos fundamentos, se introducen los modelos más empleados para representar el comportamiento de nodos en redes 5G. La Tabla 2.3 sintetiza los más utilizados en el análisis del rendimiento de este tipo de redes de comunicaciones.

Tabla 2.3: Modelos de colas utilizados para el análisis de tráfico en redes 5G

Modelo	**Características**	**Aplicación en 5G**	**Fórmulas clave**
M/M/1	Llegadas según Poisson, tiempos de servicio exponenciales, un único servidor, buffer de espera con capacidad infinita	Referencia básica para enlaces punto a punto con comportamiento estocástico ideal	$\mathbb{E}[T] = \frac{\mathbb{E}[S]}{1-\rho}$
M/D/1	Llegadas Poisson, tiempos de servicio deterministas	Escenarios con tamaño de paquete fijo y capacidad constante	$\mathbb{E}[T] = \frac{1}{2} \cdot \frac{\rho}{1-\rho} \cdot \mathbb{E}[S] + \mathbb{E}[S]$
M/G/1	Llegadas Poisson, tiempos de servicio con distribución arbitraria (general)	Modelado de tráfico mixto con diferentes perfiles de servicio	$\mathbb{E}[T] = \mathbb{E}[S] + \frac{\lambda \cdot \mathbb{E}[S^2]}{2(1-\rho)}$
G/G/1	Llegadas y servicio con distribuciones arbitrarias, 1 servidor, sistema general	Condiciones realistas con tráfico no estacionario y distribución variable	$\mathbb{E}[T_q] \leq \frac{\rho}{1-\rho} \cdot \frac{C_t^2 + C_s^2}{2} \cdot \mathbb{E}[S]$

En el contexto de los modelos M/G/1 y G/G/1 descritos en la Tabla 2.3, adquieren especial relevancia dos expresiones ampliamente reconocidas en teoría de colas: la fórmula de Pollaczek–Khintchine y la aproximación de Kingman.

La primera proporciona un resultado exacto para el tiempo medio en cola cuando el proceso de llegadas es Poisson pero el tiempo de servicio sigue una distribución genérica, relación que aparece en la tabla. Su valor reside en que permite incorporar la variabilidad del servicio mediante su segundo momento estadístico, sin necesidad de conocer la distribución completa.

Por su parte, la fórmula de Kingman constituye una aproximación general aplicable al modelo G/G/1, en el que tanto el proceso de llegadas como el de servicio siguen distribuciones arbitrarias. Como se observa en la ecuación (2.4), el retardo medio en cola $W_q^{(p)}$ depende directamente del tiempo medio de servicio $\mathbb{E}[S]$, de la carga del sistema ρ y de los coeficientes de variación de los procesos de llegada y

servicio. Esta relación pone de manifiesto que, a medida que la carga se aproxima a la unidad o cuando se aumenta la variabilidad de los procesos, el retardo crece de forma significativa, lo que permite anticipar el comportamiento del sistema en escenarios con tráfico altamente disperso.

$$W_q^{(p)} = \text{máx}\left\{0,\ \mathbb{E}[S] \cdot \frac{1}{1-\rho} \cdot \frac{C^2[T] + C^2[S]}{2} \cdot \ln\left(\frac{\rho}{1-p}\right)\right\} \tag{2.4}$$

Ambas fórmulas constituyen pilares fundamentales para la evaluación analítica del rendimiento en sistemas de comunicaciones, especialmente cuando se pretende ir más allá de los supuestos idealizados del modelo M/M/1 [18].

En definitiva, cada uno de los modelos de teoría de colas presentados proporciona una herramienta valiosa para representar distintos escenarios de tráfico, permitiendo estimar métricas clave como el tiempo medio de espera, la probabilidad de congestión o la utilización de los recursos. Su correcta aplicación es esencial para interpretar el comportamiento del fronthaul bajo diferentes configuraciones y cargas de red.

Con ello, se consolidan los fundamentos necesarios para abordar el análisis del sistema desde una doble perspectiva: tecnológica y analítica. A partir de aquí, el estudio se orienta hacia el diseño, simulación y validación de escenarios realistas, con el fin de evaluar el impacto de distintas configuraciones de tráfico sobre el rendimiento de la red.

Capítulo 3

Propuesta y desarrollo

La fase de desarrollo constituye el núcleo de este trabajo, e incluye el diseño y posterior implementación de una aplicación personalizada sobre el simulador `NS-3`, orientada a reproducir distintos patrones de tráfico en el contexto de redes 5G desagregadas. Este entorno de simulación permite modelar el comportamiento de una comunicación punto a punto entre un nodo emisor y uno receptor, representando de forma simplificada el enlace entre componentes funcionales separados de la red, como ocurre en arquitecturas desagregadas de acceso o transporte.

La lógica de generación de tráfico implementada responde a dos enfoques diferenciados: por un lado, un modelo sintético basado en distribuciones estadísticas, representativo de escenarios de tráfico heterogéneo y dinámico; por otro, un modelo realista que calcula los parámetros a partir de características físicas del canal de radio, como la configuración TDD, la numerología o el ancho de banda disponible.

Adicionalmente, esta configuración permite simular múltiples flujos simultáneos desde un mismo nodo origen, lo que posibilita estudiar escenarios de agregación de tráfico. El análisis de estos casos, incluyendo comparativas del retardo en función del número de flujos, se presenta en el capítulo siguiente.

El módulo desarrollado, denominado `CustomApplication`, ha sido programado en `C++` y está diseñado para ser completamente configurable mediante parámetros accesibles por línea de comando. Asimismo, permite trazar eventos relevantes durante la simulación, almacenando los resultados en ficheros estructurados aptos para su posterior análisis. Esta flexibilidad ha permitido evaluar de forma sistemática el impacto de distintas configuraciones típicas en despliegues 5G y 6G.

En las secciones siguientes se detalla la arquitectura de la aplicación, las decisiones de diseño adoptadas y los mecanismos empleados para la recolección y tratamiento de resultados. También se incluyen fragmentos representativos de código que ilustran los aspectos clave de la lógica implementada.

3.1. Entorno de simulación y herramientas

El desarrollo de este trabajo se ha llevado a cabo utilizando el simulador `NS-3`, una plataforma de simulación por eventos discretos diseñada específicamente para el modelado y análisis de redes de comunicaciones. Se trata de una herramienta ampliamente utilizada en el ámbito académico y de investigación, que permite emular el comportamiento de protocolos de red, enlaces físicos, dispositivos terminales y aplicaciones de usuario bajo un entorno controlado y replicable.

`NS-3` está escrito en C++ y proporciona una Application Programming Interface (API) que permite a los desarrolladores crear nodos virtuales, conectar dichos nodos a través de canales de comunicación con características configurables, instalar pilas de protocolos (por ejemplo Transmission Control Protocol (TCP) e Internet Protocol (IP)), definir aplicaciones personalizadas y trazar eventos relevantes durante la ejecución de la simulación. Su arquitectura modular lo convierte en una herramienta extremadamente flexible, capaz de adaptarse tanto a estudios de alto nivel (como la evaluación de protocolos de encaminamiento) como a modelos más detallados que representan comportamientos físicos en el nivel de enlace o capa Media Access Control (MAC) [19].

Una de las principales ventajas de `NS-3` frente a otras herramientas de simulación es su integración con librerías estándar del sistema operativo, la posibilidad de generar tráfico real mediante sockets nativos y su capacidad para exportar resultados en formatos estructurados, lo que facilita su análisis mediante herramientas externas como por ejemplo, `Python`, que se usará en este trabajo.

La Figura 3.1 muestra una visión esquemática de la arquitectura interna de un nodo en `NS-3`, incluyendo los principales componentes que lo integran y su interacción a través de interfaces y canales simulados.

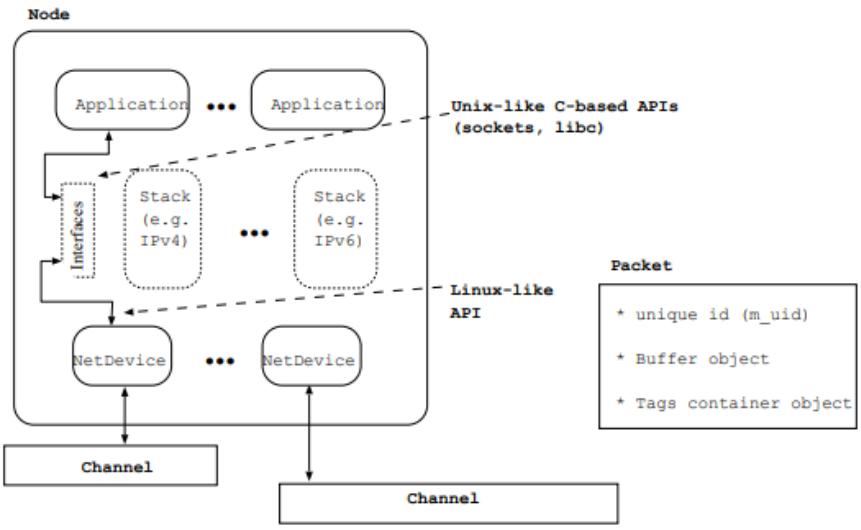


Figura 3.1: Estructura interna de un nodo en `NS-3`. Fuente: [19]

En el contexto concreto de este trabajo, se ha utilizado `NS-3` para modelar un escenario punto a punto simplificado, representativo de un enlace fronthaul entre funciones desagregadas de la red. Este enfoque permite abstraer la complejidad del despliegue real, centrándose en el estudio del comportamiento del tráfico bajo distintas configuraciones de generación de paquetes. Se ha optado por utilizar el módulo `PointToPoint` para definir el canal entre un nodo emisor y un nodo receptor, con parámetros configu-

rables como la capacidad del enlace (en bps) y la latencia (en milisegundos). Esta elección responde a la necesidad de disponer de un modelo simple pero suficientemente preciso, para analizar métricas como el retardo de transmisión, la ocupación del canal y la variabilidad de los tiempos de llegada.

El motor de eventos de `NS-3` permite programar acciones específicas en instantes de tiempo definidos, con una precisión de nanosegundos, lo cual resulta fundamental para la evaluación de tráfico de tipo fronthaul, donde pequeñas variaciones temporales pueden tener un impacto significativo sobre la calidad del servicio. Además, la herramienta incluye un sistema de trazado y recolección de estadísticas que ha sido extendido en este trabajo mediante el desarrollo de módulos personalizados, lo que ha permitido capturar tiempos de envío y recepción de paquetes, así como atributos adicionales como el tamaño de los mismos.

La generación de tráfico ha sido implementada a través de una clase propia, `CustomApplication`. Esta clase ha sido diseñada para ofrecer la máxima flexibilidad en la configuración del tráfico generado, permitiendo tanto la simulación de escenarios aleatorios (basados en distribuciones estadísticas) como deterministas (calculados a partir de parámetros físicos del sistema). La lógica de esta aplicación, así como su integración con el escenario definido en el archivo `escenario.cc`, se describen en detalle a continuación.

3.1.1. Arquitectura general de `CustomApplication`

Para dotar al entorno de simulación de una lógica de tráfico adaptable y precisa, se ha implementado una clase propia denominada `CustomApplication`, derivada de `ns3::Application`. Esta clase encapsula toda la funcionalidad necesaria para generar tráfico User Datagram Protocol (UDP) desde un nodo origen, que permite la parametrización tanto de aspectos físicos como estadísticos.

La clase se encuentra diseñada bajo una arquitectura modular que permite seleccionar entre dos modos de operación: uno basado en modelos estocásticos de tráfico, y otro fundamentado en parámetros físicos de capa baja, modelo realista. La selección de modo se realiza a través del atributo booleano `UseRandomModel`, configurable desde la línea de comandos al momento de lanzar la simulación.

`CustomApplication` contiene métodos específicos para calcular el tamaño de los paquetes, el intervalo entre envíos, el Number of Resource Blocks (NRB) y la duración de los ciclos TDD, según el caso. Además, incorpora mecanismos de trazado de eventos que permiten capturar los tiempos de transmisión de cada paquete, almacenándolos para su posterior análisis estadístico.

Para contextualizar la integración de esta aplicación dentro del entorno de simulación, la Figura 3.2 muestra un diagrama de flujo que sintetiza los principales módulos de `NS-3` empleados y la relación funcional entre ellos. Su finalidad es ofrecer una visión global del proceso, desde la configuración inicial hasta la obtención de los resultados exportados para su análisis posterior.

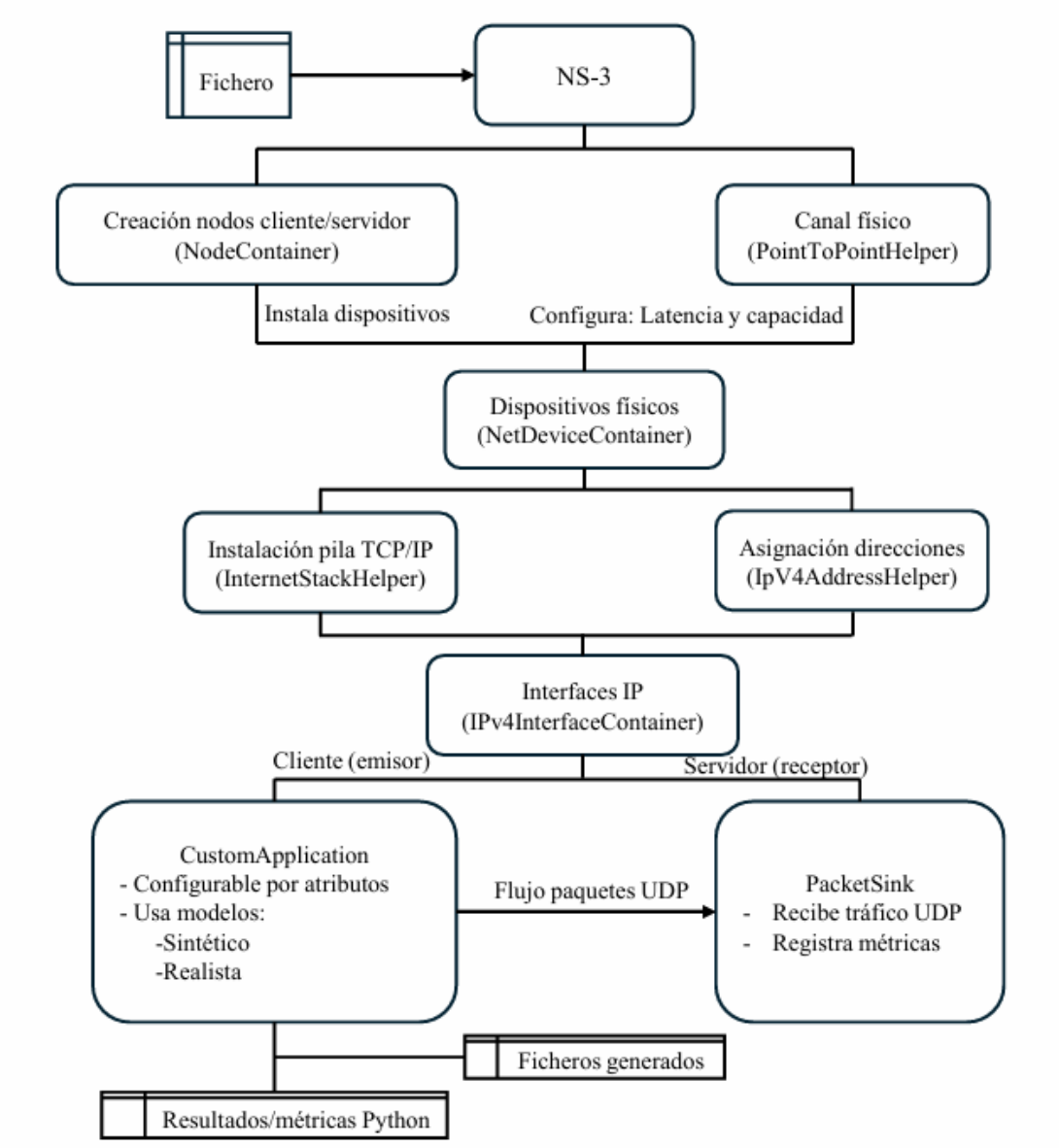


Figura 3.2: Diagrama de flujo de módulos y ficheros generados en la simulación.

En la parte superior de la Figura 3.2, se observa que la simulación parte de un fichero de configuración que alimenta al entorno de simulación `NS-3`. Este fichero establece los parámetros necesarios para la inicialización del escenario y de sus distintos componentes, incluyendo aspectos como la capacidad del enlace, el ancho de banda y el tamaño de los paquetes, entre otros, los cuales se describen con mayor detalle en el apartado siguiente. La creación del escenario dentro del simulador se realiza mediante dos procesos paralelos que constituyen la base de la red simulada: por un lado se lleva a cabo la creación de nodos cliente y servidor mediante el módulo `NodeContainer`, mientras que por otro lado se configura el canal físico a través del módulo `PointToPointHelper`. Esto último permite definir características esenciales del enlace, tales como el retardo y la capacidad del canal.

Ambos procesos confluyen en la creación de los dispositivos físicos que actúan como interfaces de red instaladas sobre cada nodo, representados por el `NetDeviceContainer`. A partir de este punto se desarrolla la configuración lógica de la red. En el lado izquierdo del diagrama se muestra la instalación

de la pila de protocolos TCP/IP a través de `InternetStackHelper`, lo que dota a los nodos de capacidades de comunicación estándar. En el lado derecho se refleja la asignación de direcciones IP utilizando `Ipv4AddressHelper`, lo que permite identificar de manera unívoca cada dispositivo en la red.

El siguiente bloque corresponde a las interfaces IP obtenidas mediante el `Ipv4InterfaceContainer`, resultado de combinar la pila TCP/IP con las direcciones asignadas. Estas interfaces distinguen claramente entre el nodo cliente, que actúa como emisor, y el nodo servidor, que actúa como receptor.

En la parte inferior se sitúan las aplicaciones y la lógica de funcionamiento. Sobre el nodo cliente se ejecuta la clase `CustomApplication`, diseñada para ser configurable mediante atributos y para trabajar con dos tipos de modelos de generación de tráfico: uno sintético, basado en distribuciones estadísticas (constante, uniforme o exponencial), y un modelo realista, que refleja parámetros físicos como el número de PRB, la numerología o el patrón TDD. Esta aplicación es la responsable de generar el flujo de paquetes. En el nodo servidor se instala una aplicación, `PacketSink`, que actúa como receptor UDP. Este módulo recibe los paquetes generados por el cliente y registra métricas relevantes como el instante exacto de llegada y el tamaño de cada paquete. El flujo de datos queda representado en el diagrama mediante una flecha horizontal que indica el envío de paquetes UDP desde el cliente hacia el servidor.

Finalmente, tanto `CustomApplication` como `PacketSink` producen ficheros con información detallada sobre los eventos de transmisión y recepción. En cada ejecución se generan automáticamente tres tipos de ficheros: por un lado, un fichero de cliente y otro de servidor por cada flujo concurrente configurado en la simulación, donde se recogen los tamaños y el retardo total de cada paquete, y adicionalmente un fichero separado que almacena los parámetros aleatorios empleados en dicha ejecución (por ejemplo, la numerología o el número de capas MIMO). Estos ficheros se integran en la parte final del flujo, representados como resultados o métricas exportadas, que pueden ser procesadas posteriormente con herramientas externas, en este caso basadas en `Python`, para favorecer el análisis, generando gráficas y permitiendo comparar resultados. De este modo, el diagrama permite comprender de forma visual y ordenada cómo se construye y configura el escenario de simulación, cómo interactúan los módulos de `NS-3` durante la ejecución y cómo se obtiene la información de salida necesaria para evaluar el comportamiento del sistema.

3.1.2. Parámetros configurables

Una de las ventajas principales de la aplicación desarrollada es su alto grado de configurabilidad. Todos los parámetros relevantes para la generación de tráfico han sido expuestos mediante atributos que pueden modificarse en tiempo de ejecución a través de la línea de comandos del simulador.

La Tabla 3.1 resume los atributos más relevantes de la clase `CustomApplication`, junto con su tipo, descripción y valor por defecto.

Esta parametrización hace posible reproducir distintos escenarios de tráfico con un simple cambio en la línea de comandos del ejecutable, facilitando el estudio sistemático de distintos casos sin necesidad de recompilar el código.

Tabla 3.1: Atributos configurables de la clase `CustomApplication`

Nombre	Tipo	Descripción	Valor por defecto
`MeanPacketSize`	`Uinteger`	Tamaño medio del paquete	1024
`MaxPacketSize`	`Uinteger`	Tamaño máximo permitido	1536
`Capacity`	`Double`	Capacidad de la aplicación (Mbps)	100.0
`UseRandomModel`	`Boolean`	Selección modelo sintético o realista	true/false
`PacketSizeDistribution`	`String`	Distribución del tamaño de paquetes	Const, expo, uniform
`InterArrivalDistribution`	`String`	Distribución entre llegadas	Const, expo, uniform
`Numerology`	`Uinteger`	Numerología del sistema (SCS)	1, 2, 3
`Bandwidth`	`Uinteger`	Ancho de banda en GHz	1
`CompressionAlgorithm`	`String`	Algoritmo de compresión IQ	BFP9
`Layers`	`Uinteger`	Número de capas MIMO	4, 8, 16

3.1.3. Configuración de los nodos y el canal punto a punto

En el Código 3.1 se muestra la creación de dos nodos mediante la clase `NodeContainer`, que se conectan posteriormente utilizando el módulo `PointToPointHelper`. Este módulo permite definir características clave del enlace como la capacidad (DataRate) y la latencia (delay).

```
NodeContainer nodes;
nodes.Create(2); // Node 0: Client, Node 1: Server

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("1Gbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("0us"));

NetDeviceContainer devices;
devices = pointToPoint.Install(nodes);
```

Código 3.1: Creación de nodos y configuración del canal punto a punto

Posteriormente, en el Código 3.2 se instala la pila de protocolos TCP, IP en los nodos mediante el helper `InternetStackHelper` y se asignan direcciones IP a través del helper `Ipv4AddressHelper`.

```
InternetStackHelper stack;
stack.Install(nodes);
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign(devices);
```

Código 3.2: Asignación de direcciones IP y pila de red

3.1.4. Instalación de la aplicación cliente y parámetros de simulación

La aplicación `CustomApplication` se instala en el nodo cliente, nodo 0, como se observa en el Código 3.3 y tiene una duración de 8960 segundos para poder conseguir la generación de, aproximadamente, un millón de paquetes. Esta aplicación es completamente parametrizable desde línea de comandos, permitiendo ajustar el tamaño medio de los paquetes, la distribución estadística, la numerología, el ancho de banda, el algoritmo de compresión y el número de capas MIMO.

```
CustomApplicationHelper appHelper;

appHelper.SetAttribute("MeanPacketSize", UintegerValue(meanPacketSize));
appHelper.SetAttribute("Capacity", DoubleValue(100.0));
appHelper.SetAttribute("UseRandomModel", BooleanValue(useRandomModel));
appHelper.SetAttribute("PacketSizeDistribution", StringValue(packetDist));
appHelper.SetAttribute("InterArrivalDistribution", StringValue(interDist));
appHelper.SetAttribute("Numerology", UintegerValue(numerology));
appHelper.SetAttribute("Bandwidth", UintegerValue(bandwidth));
appHelper.SetAttribute("CompressionAlgorithm", StringValue(compression));
appHelper.SetAttribute("Layers", UintegerValue(layers));

ApplicationContainer clientApp = appHelper.Install(nodes.Get(0));
clientApp.Start(Seconds(1.0));
clientApp.Stop(Seconds(8960.0));
```

Código 3.3: Instalación de la aplicación cliente y configuración de parámetros

3.1.5. Recepción del tráfico y captura de métricas

En el nodo receptor se instala una aplicación de tipo `PacketSink`, que actúa como servidor UDP. Este componente se encarga de recibir los paquetes enviados por la aplicación instalada en el nodo emisor y registrar métricas relevantes, como el instante exacto de llegada y el tamaño de cada paquete. De este modo, permite disponer de datos precisos para el análisis posterior del tráfico recibido y facilita la evaluación del comportamiento del enlace bajo distintas configuraciones.

La instalación se realiza mediante `PacketSinkHelper`, donde se define la dirección y el puerto de escucha, así como el tipo de socket `ns3::UdpSocketFactory`. La aplicación se activa y detiene en momentos específicos de la simulación, lo que permite delimitar el periodo de captura y sincronizarlo con el resto de eventos del escenario.

Para completar este proceso se integra la clase auxiliar `CustomTracerHelper`, que enlaza dinámicamente con la señal de recepción del `PacketSink` mediante el sistema de configuración de `NS-3`. Cada vez que se recibe un paquete, esta generación de trazas registra en un fichero `.txt` la información correspondiente, como tiempo y tamaño, lo que permite generar posteriormente estadísticas y representaciones gráficas utilizando herramientas externas como `Python`. Gracias a este mecanismo, el nodo receptor no solo actúa como destino de tráfico, sino también como punto de recogida de datos detallados para el análisis de resultados. La integración de este trazado de eventos, tanto en el cliente como en el servidor, puede observarse en el Código 3.4.

```
Address sinkAddress(InetSocketAddress(Ipv4Address::GetAny(), port));
PacketSinkHelper packetSinkHelper("ns3::UdpSocketFactory", sinkAddress);
ApplicationContainer serverApp = packetSinkHelper.Install(nodes.Get(1));
serverApp.Start(Seconds(1.0));
serverApp.Stop(Seconds(9600.0));

std::string filename = folder + "/server_log_" + std::to_string(i) + ".txt"
    ;
auto tracer = std::make_unique<CustomTracerHelper>(filename, false);

Config::ConnectWithoutContext(
    "/NodeList/1/ApplicationList/" + std::to_string(i) + "/$ns3::PacketSink
        /Rx",
    MakeCallback(&CustomTracerHelper::RxPacketTracer, tracer.get()));
```

Código 3.4: Instalación dinámica del receptor y configuración del trazado

3.1.6. Resumen de la topología simulada

Este diseño modular y parametrizable permite centrar el análisis en el comportamiento del tráfico generado, evitando introducir complejidad innecesaria a nivel de topología y facilitando la evaluación del impacto de cada configuración sobre las métricas recogidas en los nodos extremos. La Figura 3.3 muestra una representación esquemática del escenario lógico que se pretende emular, en la que se aprecia una configuración con múltiples sectores cuya agregación se transporta a través de un enlace de capacidad C hacia una unidad DU, reflejando el tipo de arquitectura desagregada que se busca modelar en el entorno simulado. Toda esta lógica se encuentra implementada en el fichero `escenario.cc`, que integra la creación de nodos, la configuración del canal, la instalación de las aplicaciones y el establecimiento de los mecanismos de trazado necesarios para la recolección de métricas.

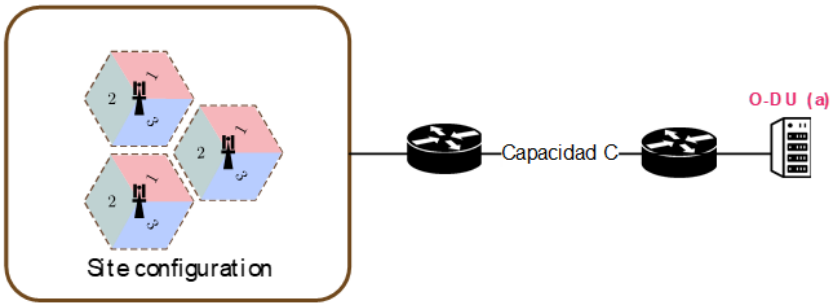


Figura 3.3: Representación esquemática la arquitectura desagregada 5G y 6G.

Este escenario constituye la base sobre la cual se desarrollan los distintos experimentos recogidos en este trabajo. Gracias a la flexibilidad del entorno `NS-3` y al diseño modular de la clase `CustomApplication`, es posible simular distintos patrones de tráfico y estudiar su impacto en métricas como el retardo, la variabilidad temporal y la ocupación del canal. A continuación, se detallan los dos modos de operación implementados en la aplicación, orientados a representar enfoques de generación de tráfico con distinta naturaleza: uno sintético y otro determinista.

3.2. Modelo sintético

El primer enfoque implementado en la aplicación `CustomApplication` consiste en la generación de tráfico basada en un modelo sintético, representativo de entornos donde el comportamiento del tráfico no es determinista y presenta una alta variabilidad temporal. Este tipo de tráfico es característico de ciertos servicios en redes 5G, como eMBB o flujos provenientes de aplicaciones interactivas, donde no existe un patrón fijo de transmisión.

En esta modalidad, tanto el tamaño de los paquetes como el tiempo entre llegadas se modelan mediante distribuciones probabilísticas, que son seleccionadas por el usuario. Estas distribuciones se pueden configurar a través de parámetros de línea de comandos, lo que permite experimentar con diferentes grados de aleatoriedad y evaluar su impacto sobre métricas como el retardo y la ocupación del canal.

3.2.1. Distribuciones disponibles y lógica de decisión

La lógica de tráfico aleatorio se activa mediante el parámetro `UseRandomModel`. Al habilitarlo, la clase `CustomApplication` emplea generadores de variables aleatorias para producir valores dinámicos en tiempo de simulación. La primera corresponde a la modalidad *Constant*, en la que todos los valores se mantienen fijos e iguales al valor medio especificado, proporcionando un patrón completamente regular. En segundo lugar, se ha empleado la distribución *Uniform*, que genera valores de manera equiprobable dentro de un rango definido de forma simétrica alrededor de la media, introduciendo cierta variabilidad controlada. Por último, se ha utilizado la distribución *Exponential*, caracterizada por una alta dispersión, lo que resulta especialmente útil para modelar tráfico con ráfagas o retardos muy variables.

3.2.2. Cálculo del tamaño de los paquetes

Previo al envío de cada paquete, su tamaño se determina de manera dinámica. En el caso del modelo sintético, dicho cálculo se basa en la distribución estadística configurada para el experimento. El Código 3.5 muestra la lógica implementada para este proceso.

```
if (m_useRandomModel) {
    double meanSize = 512;
    double maxSize = m_maxPacketSize;
    if (m_packetSizeDistribution == "Constant") {
        m_packetSize = m_meanPacketSize;
    } else if (m_packetSizeDistribution == "Uniform") {
        m_packetSize = m_randomSize->GetValue(minSize, maxSize); // A
            random value is generated between the minimum and maximum packet
             size
    } else if (m_packetSizeDistribution == "Exponential") {
        Ptr<ExponentialRandomVariable> exp = CreateObject<
            ExponentialRandomVariable>();
        exp->SetAttribute("Mean", DoubleValue(m_meanPacketSize));
        m_packetSize = exp->GetValue();
    }
}
```

Código 3.5: Cálculo dinámico del tamaño de paquete según la distribución seleccionada

Esta lógica permite generar tamaños de paquete variables, reproduciendo condiciones más cercanas a un entorno real. De este modo, la simulación refleja de forma más precisa cómo la variabilidad en el tamaño de los paquetes influye en la carga del sistema y en el tiempo medio de servicio.

3.2.3. Determinación del intervalo entre paquetes

Una vez determinado el tamaño de cada paquete, el siguiente paso consiste en calcular el tiempo esperado entre envíos sucesivos, conocido como *inter-arrival time*. Para ello, se utiliza el tamaño medio de cada paquete, expresado en bits (`meanPacketBits`), que se divide entre la capacidad inyectada al enlace (`injectedCapacity`), expresada en bits por segundo. El resultado de esta operación es un valor en segundos que indica, de forma aproximada, cada cuánto tiempo debería generarse un paquete para respetar el caudal configurado en la simulación.

A partir de este valor medio, el sistema aplica una transformación adicional según la distribución estadística seleccionada por el usuario. De este modo, es posible generar intervalos constantes, uniformemente distribuidos dentro de un rango definido, o bien intervalos con mayor variabilidad, modelados mediante una distribución exponencial.

```
1  double meanPacketBits = m_meanPacketSize * 8.0;
2  double injectedCapacity = m_capacity * 1e6; // bps
3  double timeBetweenArrivals = meanPacketBits / injectedCapacity;
4
5  if (m_interArrivalDistribution == "Constant") {
6      m_interval = Seconds(timeBetweenArrivals);
7  } else if (m_interArrivalDistribution == "Uniform") {
8      m_randomTime->SetAttribute("Min", DoubleValue(timeBetweenArrivals *
           0.5));
9      m_randomTime->SetAttribute("Max", DoubleValue(timeBetweenArrivals *
           1.5));
10     m_interval = Seconds(m_randomTime->GetValue());
11 } else if (m_interArrivalDistribution == "Exponential") {
12     Ptr<ExponentialRandomVariable> exp = CreateObject<
           ExponentialRandomVariable>();
13     exp->SetAttribute("Mean", DoubleValue(timeBetweenArrivals));
14     m_interval = Seconds(exp->GetValue());
15 }
```

Código 3.6: Cálculo del tiempo entre llegadas en modelo aleatorio

3.2.4. Generación de tráfico mediante eventos recursivos

En este modelo, la generación de tráfico se implementa a través de un mecanismo iterativo: cada vez que se transmite un paquete, se programa de forma automática el siguiente envío, utilizando el motor de eventos discretos de `NS-3`. De este modo, se mantiene un flujo constante de paquetes sin necesidad de recurrir a bucles explícitos, garantizando además que se respete el intervalo aleatorio calculado previamente.

El funcionamiento interno de esta lógica, donde la función `SendPacketRandom` calcula dinámicamente el tamaño del paquete, registra tanto el instante de envío como el tamaño en sus respectivas estructuras, crea y envía el paquete a través del socket configurado y, finalmente, vuelve a planificar su propia ejecución utilizando `Simulator::Schedule`, como puede observarse en el Código 3.7.

```
1  void CustomApplication::SendPacketRandom() {
2    CalculatePacketSize();
3    m_packetSizes.push_back(m_packetSize);
4
5    ns3::Time now = ns3::Simulator::Now();
6    m_arrivalTimestamps.push_back(now);
7
8    Ptr<Packet> packet = Create<Packet>(m_packetSize);
9    m_socket->SendTo(packet, 0, m_peerAddress);
10   m_txTrace(packet);
11
12   Simulator::Schedule(m_interval, &CustomApplication::SendPacketRandom,
         this);
13 }
```

Código 3.7: Lógica de envío de paquetes en modelo sintético

Este enfoque basado en eventos permite optimizar el uso de recursos en la simulación, evitando ciclos de espera innecesarios, logrando así una implementación más limpia y eficiente dentro del modelo sintético.

3.3. Modelo realista

Este modelo implementa una lógica de generación de tráfico determinista, orientada a representar con fidelidad entornos de red donde el tráfico sigue un patrón rígido, condicionado por la capa física. Este enfoque está particularmente alineado con los requisitos del plano de usuario en redes tanto 5G como 6G desagregadas, como ocurre en los enlaces fronthaul que conectan las unidades RU y DU. A diferencia del modelo sintético, este método prescinde de distribuciones estadísticas para calcular los intervalos de envío y tamaños de paquetes, basándose en su lugar en configuraciones específicas de numerología, ancho de banda, número de capas MIMO y algoritmos de compresión I/Q.

3.3.1. Asignación de recursos radio y cálculo del NRB

En el estándar 5G New Radio (NR), la asignación de recursos físicos se gestiona mediante unidades denominadas Physical Resource Blocks (PRBs), que representan segmentos discretos del canal en el dominio frecuencia-tiempo. Cada PRB está formado por 12 subportadoras contiguas, cuya duración temporal depende de la numerología μ.

$$\text{SCS} = 15 \cdot 2^{\mu} \text{ kHz} \tag{3.1}$$

En la relación (3.1), el espaciado entre subportadoras (Subcarrier Spacing (SCS)) determina tanto la duración de los símbolos como la granularidad temporal de la transmisión. Combinando este parámetro con el ancho de banda total asignado al canal, se obtiene el número máximo de PRBs disponibles. Dichos valores están normalizados por las especificaciones de 3rd Generation Partnership Project (3GPP) y se almacenan en una tabla de correspondencia implementada internamente como un contenedor `map`, tal y como se muestra en el Código 3.8, donde el primer valor corresponde al SCS, el segundo al ancho de banda y el tercero al mayor valor posible de PRBs disponibles.

```
1 static const std::map<std::tuple<uint32_t, uint32_t>, uint32_t> nrbTable =
    {
2   {{15, 5}, 25}, {{15, 10}, 52}, {{15, 20}, 106}, {{30, 50}, 133},
3   {{60, 100}, 135}, {{120, 100}, 66}, {{240, 400}, 0}
4 };
```

Código 3.8: Tabla de correspondencias para el número máximo de NRB

A partir de esta tabla, la función `CalculateNRB()` determina el número máximo de bloques de recursos admitidos en función de la numerología y el ancho de banda configurados. Sobre esta base se aplica la distribución seleccionada por el usuario, constante, uniforme o exponencial, para obtener el número efectivo de PRBs que se utilizarán en el intervalo actual. Por ejemplo, si se elige como distribución la uniforme, se tomará un valor aleatorio comprendido entre 0 y el valor máximo de PRBS obtenido en la tabla. El procedimiento se detalla en el Código 3.9:

```
uint32_t scs = 15 * (1 << m_numerology); // The SCS is calculated

auto it = nrbTable.find({scs, m_bw}); // The combination of SCS and BW is
    searched
if (it != nrbTable.end()) {
  uint32_t maxNrb = it->second; // The maximum value of PRBs is found

  if (m_nrbDistribution == "Constant") {
        m_nrb = maxNrb;
  } else if (m_nrbDistribution == "Uniform") {
        m_randomNRB->SetAttribute("Min", DoubleValue(0));
        m_randomNRB->SetAttribute("Max", DoubleValue(maxNrb));
        m_nrb = static_cast<uint32_t>(m_randomNRB->GetValue());
  } else if (m_nrbDistribution == "Exponential") {
        Ptr<ExponentialRandomVariable> exp = CreateObject<
            ExponentialRandomVariable>();
        exp->SetAttribute("Mean", DoubleValue(maxNrb / 2.0));
  }
}
```

Código 3.9: Determinación del número de bloques de recursos en tiempo de simulación

De este modo, la asignación de recursos físicos no solo respeta las restricciones impuestas por el estándar, sino que además incorpora flexibilidad para introducir variabilidad en los experimentos, permitiendo analizar el comportamiento del sistema bajo diferentes patrones de carga y asignación.

3.3.2. Determinación del tamaño del paquete

Una vez determinado el número de PRBs disponibles, es posible calcular con exactitud el tamaño del paquete que se generará. Cada bloque de recursos transporta información modulada en sus componentes en fase (I) y en cuadratura (Q), y el tamaño final del bloque depende del algoritmo de compresión I/Q configurado. Este algoritmo establece dos parámetros fundamentales: W, que indica el número de bits asignados a cada componente, y F, que corresponde a un campo adicional de escalado.

El tamaño bruto de un único PRB, expresado en bits, se obtiene mediante la relación (3.2)

$$\text{PRB}_{\text{size}} = 2 \cdot W \cdot 12 + F \tag{3.2}$$

Como se muestra en la ecuación (3.2), para obtener el tamaño total del paquete (`IQ_data`) en bytes, se multiplica el tamaño de un bloque por el número de PRBs seleccionados, y se divide entre 8.

$$\text{IQ}_{\text{data}} = \frac{\text{NRB} \cdot \text{PRB}_{\text{size}}}{8} \tag{3.3}$$

Esta lógica se implementa en el método `CalculatePacketSize()`, encargado de realizar el cálculo en tiempo de simulación. El fragmento de código correspondiente se muestra en el Código 3.10:

```
int PRB_size = (2 * iqWidth * 12 + scalerNum);
m_packetSize = m_nrb * PRB_size / 8;
```

Código 3.10: Cálculo del tamaño del paquete en función del número de PRB y la compresión IQ

Gracias a este procedimiento, el tamaño de cada paquete generado refleja de forma precisa la configuración física y el esquema de compresión definidos, permitiendo obtener mediciones realistas y coherentes con las especificaciones del estándar.

3.3.3. Intervalo entre paquetes y esquema TDD

La generación de tráfico sigue la estructura temporal definida por la capa física. En 5G New Radio (5G NR), un slot está compuesto por 14 símbolos Orthogonal Frequency Division Multiplexing (OFDM), y su duración depende, según una relación inversa, de la numerología, de acuerdo con (3.4).

$$\text{Duración}_{\text{slot}} = \frac{1}{1000 \cdot 2^{\mu}} \tag{3.4}$$

El intervalo entre envíos, calculado en la ecuación (3.5) también depende del número de capas MIMO activas, ya que cada capa introduce un flujo adicional de datos y reparte el tiempo de transmisión entre todas ellas.

$$\text{Intervalo}_{\text{paquetes}} = \frac{\text{Duración}_{\text{slot}}}{14 \cdot \text{Capas}} \tag{3.5}$$

Este valor se emplea posteriormente como parámetro en el planificador de eventos de `NS-3`, encargado de programar los envíos dentro de cada símbolo del slot, garantizando la sincronización temporal definida por la capa física.

3.3.4. Ejecución cíclica y patrón TDD

La transmisión de paquetes se organiza en ciclos TDD definidos mediante un patrón configurable, como "`DDDDSSSSUU`", donde cada carácter representa un tipo de slot: Downlink (D), Shared (S) o Uplink (U). Únicamente en los slots de tipo D o S se programan envíos de paquetes, mientras que los slots U se reservan para tráfico de subida o permanecen inactivos.

La implementación se lleva a cabo en la función `SlotDuration()`, que recorre secuencialmente el patrón y, al detectar un slot habilitado para transmisión, emplea el planificador de eventos de `NS-3` para programar catorce envíos, uno por cada símbolo OFDM de dicho slot. La lógica exacta puede verse en el Código 3.11:

```
if (currentSlot == 'D' || currentSlot == 'S') {
  for (uint32_t i = 0; i < 14; ++i) {
    Simulator::Schedule(Seconds(i * m_interval.GetSeconds()),
                        &CustomApplication::SendOnePacket, this);
  }
}
```

Código 3.11: Programación de envíos por ciclo TDD

Cuando se completa un ciclo completo del patrón, se incrementa un contador interno y se dispara el evento de trazado `CycleCompleted`, que puede utilizarse para sincronizar el análisis o establecer hitos dentro de la simulación.

3.3.5. Síntesis y aplicación del modelo

Este modelo dota a la simulación de un comportamiento realista, que refleje las condiciones físicas de redes 5G y 6G. Al depender exclusivamente de parámetros como la numerología, la compresión y la estructura TDD, permite analizar con precisión cómo influyen estos elementos en el tamaño de los paquetes, su frecuencia de envío y el uso efectivo del canal.

Como referencia orientativa, la Tabla 3.2 muestra un ejemplo de configuración realista en un entorno de red con tres celdas trisectoriales. Se recogen parámetros típicos como la banda de operación, el espaciado entre SCS, el ancho de banda asignado y el número de capas MIMO configuradas en cada sector.

Tabla 3.2: Ejemplos de configuración física en el modelo realista

Sector	**SCS (KHz)**	**BW (MHz)**	**Algoritmo**	**Capas MIMO**
1	30	100	BFP9	16
2	15	20	BFP9	4
3	15	20	BFP9	4

Gracias a esta aproximación, el modelo realista permite simular situaciones con una sincronización fina y una tasa de generación de tráfico coherente con las especificaciones de la capa física, resultando especialmente útil para evaluar escenarios donde el control temporal y estructural del tráfico es fundamental.

Antes de abordar los aspectos relativos al ciclo de transmisión, la Tabla 3.3 resume la comparativa entre los dos enfoques de generación de tráfico implementados en la aplicación. Así, permite visualizar de forma resumida las principales diferencias entre ambas configuraciones, tanto en lo referente a su lógica interna como al tipo de escenarios que permiten representar.

Tabla 3.3: Comparativa entre enfoques de generación de tráfico

Características	**Modelo sintético**	**Modelo realista**
Naturaleza del tráfico	Estocástica, basada en distribuciones estadísticas	Determinista con variabilidad entre ejecuciones
Parámetros principales	Tamaño de paquete y tiempo entre llegadas	Numerología, ancho de banda, compresión IQ, TDD, capas MIMO
Distribuciones soportadas	Constante, Uniforme, Exponencial	Variación de parámetros físicos por iteración o flujo
Periodicidad del tráfico	Irregular	Periódica sincronizada con slots y símbolos
Realismo en tráfico fronthaul	Aproximado (sintético)	Basado en configuración 5G con NRB, IQ, slots TDD
Aplicaciones típicas	eMBB, gaming, tráfico impredecible	Tráfico CPRI/eCPRI, enlaces entre O-RU y O-DU
Variabilidad entre flujos	Inherente al modelo	Introducida modificando parámetros entre flujos
Interpretación adicional	Tráfico no estructurado por naturaleza	Puede simular flujos desde estaciones base distintas
Ventaja principal	Flexibilidad y diversidad en los escenarios	Realismo físico con capacidad de variación sistemática

3.4. Inicialización y finalización del ciclo de transmisión

En el marco de `NS-3`, toda clase derivada de `ns3::Application` debe implementar dos métodos fundamentales: `StartApplication()` y `StopApplication()`, los cuales delimitan temporalmente la fase activa de la aplicación dentro del ciclo de simulación. Estos métodos constituyen el punto de entrada y de cierre de la lógica de tráfico, asegurando una correcta interacción con el motor de eventos discretos del simulador.

El método `StartApplication()` se invoca automáticamente en el instante programado para el arranque de la aplicación. Su cometido principal es preparar todos los elementos necesarios para iniciar la generación de tráfico. Para ello, configura la dirección IP y el puerto del nodo receptor a través de un objeto `InetSocketAddress`, crea un socket UDP asociado a la pila de red local y lo vincula mediante la función `Bind()`, dejándolo listo para transmitir. A continuación, evalúa el atributo `UseRandomModel` para seleccionar la lógica de transmisión: si está activo el modelo sintético, se llama al método `SendPacketRandom()`, mientras que si se opta por el modelo realista, se ejecuta la función `SlotDuration()` encargada de organizar los envíos según el patrón TDD.

El fragmento de código mostrado en el Código 3.12 ilustra de forma detallada cómo se lleva a cabo este proceso de inicialización, incluyendo la conexión de la señal `TxPacket` a una función de *callback* que permite trazar en tiempo real cada evento de transmisión:

```
void CustomApplication::StartApplication() {
  NS_LOG_FUNCTION(this);


  Ipv4Address remoteAddress("10.1.1.2"); // Destination IP address
  uint16_t remotePort = m_remotePort;

  InetSocketAddress remote = InetSocketAddress(remoteAddress, remotePort);

  m_peerAddress = remote;

  m_socket = Socket::CreateSocket(GetNode(), TypeId::LookupByName("ns3::
      UdpSocketFactory"));
  m_socket->Bind();

  if (m_useRandomModel) {
    SendPacketRandom();  //Synthetic model
  } else {
    SlotDuration();     // Realistic model
  }

  ns3::Config::ConnectWithoutContext("/NodeList/*/ApplicationList/*/$ns3::
      CustomApplication/TxPacket",
  ns3::MakeCallback(&CustomApplication::PacketTransmitCallBack, this));
}
```

Código 3.12: Implementación de StartApplication() con comentarios

Por su parte, el Código 3.13 refleja el método `StopApplication()`, que se llama cuando finaliza el intervalo temporal de la aplicación. Su propósito es cerrar correctamente el flujo de transmisión y liberar los recursos utilizados durante la simulación. En primer lugar, cancela cualquier evento de transmisión pendiente y cierra el socket UDP asociado, asegurando que no se produzcan envíos adicionales. A continuación, si se han almacenado marcas de tiempo durante la ejecución, el método genera un fichero `.csv` con los tiempos de llegada y transmisión, así como con los tamaños de los paquetes enviados. Por último, se calcula (mostrando los resultados numéricos por consola) el número total de paquetes transmitidos, el tiempo total de actividad y el intervalo medio entre transmisiones, lo cual resulta esencial para una primera evaluación cuantitativa del experimento.

```cpp
void CustomApplication::StopApplication() {
  m_running = false;

  if (m_socket) {
    m_socket->Close();
    m_socket = nullptr;
  }

  if (!m_txTimestamps.empty()) {
    std::ofstream output("resultados.csv");
    output << "arrival_time,tx_time,packet_size\n";
    for (size_t i = 0; i < m_txTimestamps.size(); ++i) {
      output << m_arrivalTimestamps[i].GetSeconds() << ","
             << m_txTimestamps[i].GetSeconds() << ","
             << m_packetSizes[i] << "\n";
    }
    output.close();
  }
}
```

Código 3.13: Finalización y volcado de resultados en `StopApplication`

Esta estructura modular asegura una ejecución controlada del ciclo de vida de la aplicación, desde su inicialización hasta la consolidación final de los resultados. Además, permite que la lógica de generación de tráfico, ya sea aleatoria o determinista, se active de manera completamente transparente para el usuario, garantizando la coherencia de los experimentos y facilitando su posterior análisis. Gracias a esta implementación, la aplicación desarrollada resulta fácilmente extensible y adaptable a otros escenarios o configuraciones, consolidando su utilidad como herramienta flexible de simulación en el entorno `NS-3`.

A lo largo de este capítulo se ha presentado en detalle el proceso de diseño e implementación de la aplicación `CustomApplication` en el simulador `NS-3`, incluyendo la configuración del escenario de red, la arquitectura de la clase principal, los parámetros configurables y la lógica específica de generación de tráfico en sus dos modos operativos: aleatorio y determinista. Se han descrito los aspectos técnicos fundamentales que permiten reproducir condiciones realistas de tráfico en enlaces fronthaul, haciendo uso de variables controlables como la numerología, el ancho de banda, la compresión I/Q o la distribución estadística de los paquetes.

La estructura modular de la aplicación, junto con su integración con el sistema de trazado y captura de eventos, proporciona una base sólida para el estudio cuantitativo del rendimiento del sistema bajo múltiples configuraciones. Esta capacidad resulta clave para llevar a cabo un análisis detallado del comportamiento del tráfico simulado y su impacto sobre métricas críticas como el retardo, la variabilidad temporal o la eficiencia del canal.

En el siguiente capítulo se analizan los resultados obtenidos mediante diferentes ejecuciones experimentales. Se exploran comparativas entre los modelos teóricos y los datos simulados, y se evalúa el impacto de los distintos parámetros sobre el rendimiento del sistema, con el fin de extraer conclusiones útiles sobre el comportamiento del tráfico en redes 5G y 6G desagregadas.

Capítulo 4

Análisis y resultados

En este capítulo se presenta de manera sistemática el proceso seguido para obtener, procesar y evaluar los resultados derivados de las simulaciones realizadas. Con el objetivo de ofrecer una visión completa, se detalla en primer lugar la metodología empleada para la recolección y el análisis de datos, así como el entorno de trabajo utilizado para procesarlos y validarlos. A continuación, se describen los diferentes enfoques de simulación implementados, que incluyen tanto modelos sintéticos con distribuciones estadísticas controladas como configuraciones realistas fundamentadas en parámetros físicos de redes 5G.

Sobre esta base, se analizan tres modelos de tráfico con distintos grados de aleatoriedad y complejidad: el modelo M/M/1, representativo de sistemas con llegadas y servicios exponenciales; el modelo M/D/1, que incorpora tiempos de servicio deterministas para estudiar el efecto de reducir la variabilidad; y el modelo G/G/1, que generaliza los anteriores al permitir distribuciones arbitrarias de llegada y servicio. Para cada caso, se comparan los valores obtenidos en simulación con las predicciones analíticas, evaluando métricas como el retardo medio, la distribución acumulada de retardos y el impacto de la carga del sistema.

Por último, se presentan los resultados de un escenario más realista con múltiples flujos concurrentes, donde se examina el comportamiento agregado del enlace fronthaul y se introducen técnicas avanzadas de estimación de percentiles mediante la aproximación de Kingman. De este modo, los apartados que siguen permiten interpretar, contrastar y justificar los resultados obtenidos, aportando una visión completa del rendimiento del sistema bajo distintos modelos y configuraciones experimentales.

4.1. Metodología de análisis y entorno de trabajo

El análisis de los resultados obtenidos a partir de las simulaciones realizadas en `NS-3` se ha llevado a cabo mediante la herramienta Jupyter Notebook, un entorno interactivo de desarrollo basado en `Python` que permite combinar código, texto explicativo, ecuaciones y visualizaciones en un único cuaderno ejecutable. Esta aproximación proporciona una gran versatilidad para el procesamiento de datos, su representación gráfica y la validación progresiva de resultados.

Aunque el núcleo de las simulaciones se desarrolla íntegramente en `NS-3`, un simulador de eventos discretos ampliamente utilizado en el ámbito de las redes de comunicaciones, el análisis posterior de los resultados se realiza de forma externa. Concretamente, las aplicaciones cliente y servidor generan ficheros `.txt` que contienen información detallada sobre los tiempos de transmisión y recepción, así como los

tamaños de los paquetes transmitidos, los cuales se almacenan automáticamente en carpetas específicas al finalizar cada simulación.

A continuación, estos ficheros son procesados mediante scripts en Python organizados en cuadernos Jupyter (`.ipynb`). Esta estructura permite ejecutar el análisis de forma modular, facilitando la trazabilidad y la reproducibilidad de los experimentos. El flujo general de trabajo se puede resumir en las siguientes fases:

- Lectura y preprocesado de datos: Los registros se cargan como estructuras `DataFrame` usando la librería `pandas`, permitiendo una manipulación eficiente de los mismos.
- Cálculo de métricas: Se extraen magnitudes clave como el retardo por paquete, la tasa de llegada, la varianza del servicio y la carga del sistema (ρ).
- Comparación con modelos analíticos: Se construyen funciones de distribución acumulada (CDF) para comparar los resultados simulados con expresiones teóricas de modelos de colas como M/M/1, M/D/1 y G/G/1.
- Visualización: Utilizando `matplotlib`, se generan gráficos comparativos (CDFs, gráficos de barras, histogramas) que permiten contrastar de manera visual el comportamiento del sistema simulado respecto a su modelo teórico.

Este procedimiento garantiza una separación clara entre la fase de simulación y el análisis, lo que favorece una evaluación sistemática y rigurosa del desempeño del sistema bajo diferentes configuraciones de tráfico.

4.2. Enfoques de simulación implementados

El estudio contempla dos enfoques complementarios para la generación de tráfico en el simulador `NS-3`, diseñados con propósitos diferenciados, pero articulados dentro del mismo entorno experimental.

El primero, de carácter aleatorio, permite modelar escenarios con variabilidad inherente en los tamaños de los paquetes y en los tiempos entre llegadas, configurable mediante distribuciones estadísticas (constante, uniforme o exponencial). Esta aproximación, reproduce el comportamiento típico de servicios como eMBB o aplicaciones interactivas, y facilita una validación progresiva frente a modelos teóricos clásicos de la teoría de colas (M/M/1, M/D/1 y G/G/1) mediante simulaciones punto a punto con un único flujo.

Por su parte, el otro modelo introduce una lógica determinista fundamentada en parámetros físicos característicos del canal de radio en redes 5G. En este caso, el tráfico se genera en función de atributos técnicos como la numerología empleada, el ancho de banda asignado, número de capas MIMO o el algoritmo de compresión IQ, enmarcados en una estructura TDD definida por slots. Esta configuración permite calcular de forma periódica el tamaño y la frecuencia de los paquetes transmitidos, emulando con mayor fidelidad el tráfico fronthaul del plano de usuario. Para incorporar cierta variabilidad controlada, se han realizado tres iteraciones por cada número de flujos concurrentes (de 1 a 4), empleando parámetros aleatorios distintos en cada caso. Estos se registran sistemáticamente en ficheros de trazas por simulación, asegurando la reproducibilidad del experimento.

4.3. Modelo sintético

Se presenta en la Figura 4.1 un esquema simplificado de una red fronthaul, compuesta por la CU y una RU conectadas mediante un enlace punto a punto. En este escenario se considera un único flujo de datos unidireccional, que parte de la CU con una tasa de llegada λ y se dirige hacia la RU, encargada de su transmisión inalámbrica final.

El enlace entre ambas entidades se modela como una conexión directa, sin nodos intermedios, lo que permite centrarse exclusivamente en las características estadísticas del tráfico y su impacto sobre el rendimiento. La topología refleja una arquitectura típica en redes móviles de acceso distribuido, donde las funciones de control y procesamiento se agrupan en la CU y se delega la funcionalidad de transmisión en la RU. Este diseño permite evaluar de forma aislada el comportamiento del sistema bajo diferentes modelos de tráfico, como los analizados en este trabajo (M/M/1, M/D/1, G/G/1), garantizando que los resultados obtenidos se deben únicamente a la dinámica del tráfico y no a factores externos como la topología o el número de flujos concurrentes.

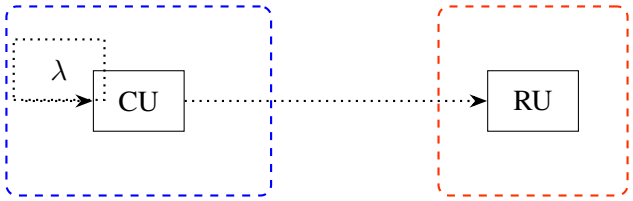


Figura 4.1: Esquema red fronthaul con un flujo entre CU y RU

En la Tabla 4.1 se resumen los principales parámetros configurados en la simulación de este modelo sintético.

Parámetro	**Valor**
Tamaño medio del paquete	1024 bytes
Capacidad del enlace	1,0.16 ,0.105 Gbps
Puerto de destino	1234
Tipo de tráfico	UDP
Número de flujos simultáneos	1

Tabla 4.1: Parámetros de simulación modelo sintético

4.3.1. Modelo M/M/1 - Exponencial/Exponencial

Esta sección presenta un análisis detallado de los resultados obtenidos a partir del modelo M/M/1, implementado mediante la aplicación personalizada desarrollada sobre el simulador `NS-3`. El objetivo principal es validar que la generación de tráfico aleatorio en la simulación es correcta, y representa fielmente el comportamiento de un sistema de colas con llegadas y servicios distribuidos exponencialmente.

Además, se analiza el retardo total experimentado por los paquetes y se comparan los resultados simulados con las expresiones teóricas conocidas del modelo M/M/1. Estas comparaciones permiten comprobar la validez del enfoque y sirven como referencia para evaluar otros modelos posteriores (como M/D/1 y G/G/1), bajo las mismas condiciones de carga del sistema.

Validación de la Generación de Tráfico Aleatorio

Antes de evaluar métricas como el retardo, es necesario verificar que el tráfico generado por la aplicación sigue las distribuciones estadísticas esperadas. Se han analizado:

- La distribución del tamaño de los paquetes

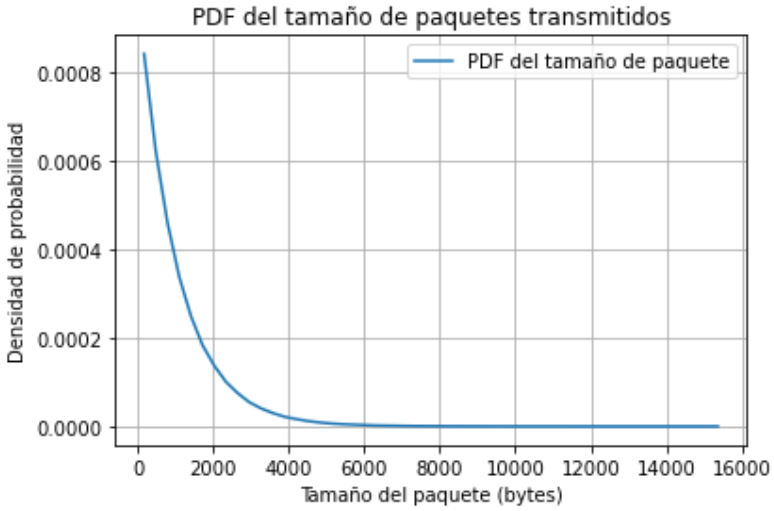


Figura 4.2: Función densidad de probabilidad del tamaño de paquetes

En la Figura 4.2 se muestra la función PDF del tamaño de los paquetes generados durante la simulación. La distribución es exponencial, coherente con el modelo M/M/1, donde los tiempos de servicio, proporcionales al tamaño del paquete, se corresponden con dicha distribución. La mayoría de los paquetes tienen tamaños pequeños, con una probabilidad significativamente mayor por debajo de los 2000 bytes. A medida que el tamaño aumenta, la densidad decae rápidamente. Esto implica que, aunque pueden generarse paquetes grandes, su aparición es muy poco frecuente. El valor medio del tamaño de los paquetes se sitúa en torno a 1023.31 bytes, representando el parámetro de escala de la exponencial. El punto de corte con el eje y para $x = 0$ corresponde al valor de la densidad en ese instante, y está dado por la inversa de la media del tamaño de los paquetes: $f(0) = \frac{1}{\mathbb{E}[X]} = \frac{1}{1023{,}31} \approx 0{,}0009772\ B^{-1}$. Este valor coincide con lo observado gráficamente, validando que el generador de tráfico implementa correctamente una distribución exponencial [20].

- La distribución de los tiempos entre llegadas

La Figura 4.3 representa la función PDF del tiempo entre llegadas, inter-arrival time, entre paquetes sucesivos. Se aprecia una distribución exponencial, con una alta concentración de eventos en tiempos cercanos a cero, como corresponde al tiempo entre llegadas consecutivas cuando éstas lo hacen según un proceso de Poisson. Este tipo de proceso es un modelo estocástico que describe la ocurrencia de eventos aleatorios en el tiempo, en el que los eventos son independientes y se producen a una tasa constante λ. Los tiempos entre eventos consecutivos están distribuidos exponencialmente, lo que implica que el sistema no tiene memoria: la probabilidad de que ocurra un evento no depende de cuándo ocurrió el anterior. El valor máximo de la curva (en $x = 0$) está dado por $f(0) = \lambda$, siendo $\lambda = 1/\mathbb{E}[T]$, donde $\mathbb{E}[T]$ es el valor medio del tiempo entre llegadas. En este caso, la media del tiempo entre llegadas es $\mathbb{E}[T] = 8{,}192 \times 10^{-5}$ s, por lo que se obtiene $\lambda = \frac{1}{8{,}192 \times 10^{-5}} \approx 12207{,}03\ s^{-1}$.

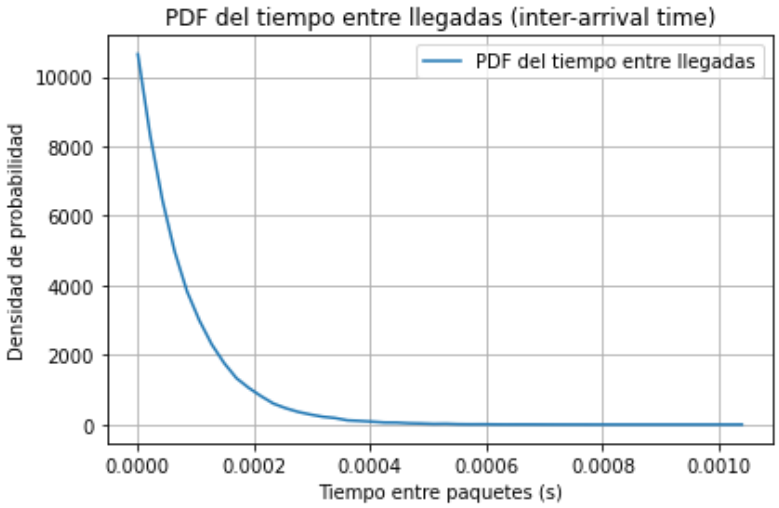


Figura 4.3: Función de densidad de probabilidad del tiempo entre llegadas sucesivas de paquetes.

Este valor concuerda con la altura inicial de la PDF observada en la gráfica, y valida que el proceso de llegada aleatoria se configura correctamente en la simulación.

Este modelo representa un sistema con alta aleatoriedad, tanto en la llegada como en el servicio. El retardo medio en la cola, denotado como $\mathbb{E}[W_q]$, está dado por la expresión (4.1):

$$\mathbb{E}[W_q] = \frac{\rho}{1-\rho} \cdot \mathbb{E}[T_s] \tag{4.1}$$

donde:

- ρ es la utilización del sistema, definida como $\rho = \lambda \cdot \mathbb{E}[T_s]$, siendo λ la tasa de llegada,
- $\mathbb{E}[T_s]$ es el tiempo medio de servicio.

Esta fórmula muestra cómo el retardo medio en la cola crece de forma no lineal a medida que aumenta la carga del sistema. En particular, cuando $\rho \to 1$, el retardo tiende a infinito, lo que indica una situación de congestión. En la ecuación se muestra el retardo medio total que experimenta un paquete en el sistema, denotado por $\mathbb{E}[W]$, y que se compone de dos contribuciones: el tiempo en la cola $\mathbb{E}[W_q]$ y el tiempo medio de servicio $\mathbb{E}[T_s]$.

Análisis del retardo total

Una vez verificado que el tráfico simulado sigue las distribuciones estadísticas correspondientes al modelo M/M/1, se procede al análisis del retardo total experimentado por los paquetes. Esta métrica se define como la suma del tiempo de espera en la cola y el tiempo de servicio.

Para validar el modelo, se calcula la media del retardo total a partir de los datos simulados y se compara con el valor teórico esperado, el cual se obtiene mediante la fórmula (4.2):

$$\mathbb{E}[W] = \mathbb{E}[W_q] + \mathbb{E}[T_s] = \frac{\rho}{1-\rho} \cdot \mathbb{E}[T_s] + \mathbb{E}[T_s] \tag{4.2}$$

La Figura 4.4 muestra dos representaciones complementarias que permiten analizar con mayor profundidad el comportamiento del retardo total experimentado por los paquetes en un sistema M/M/1. Por un lado, se muestra un gráfico de barras que compara el valor medio del retardo obtenido a partir de la simulación con el valor teórico esperado según el modelo analítico. Por otro, se incluye un histograma que refleja la distribución empírica de los retardos observados durante la ejecución.

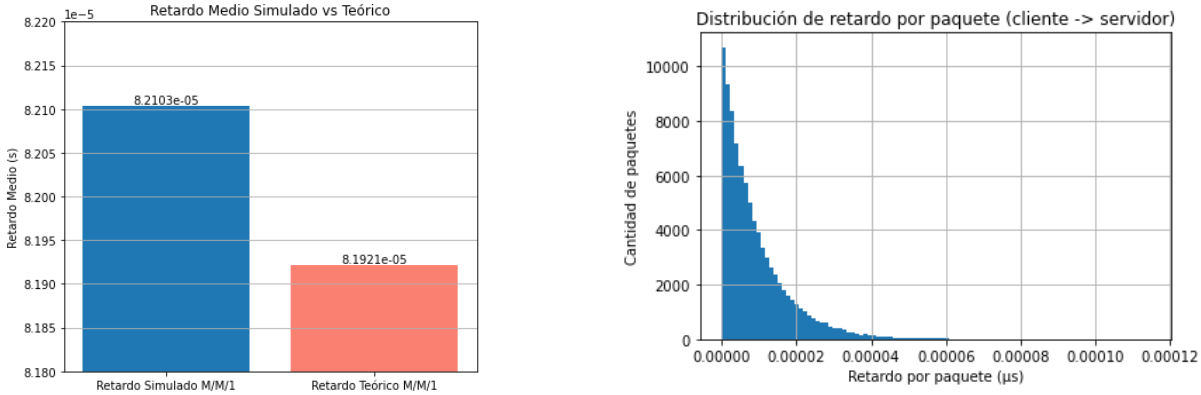


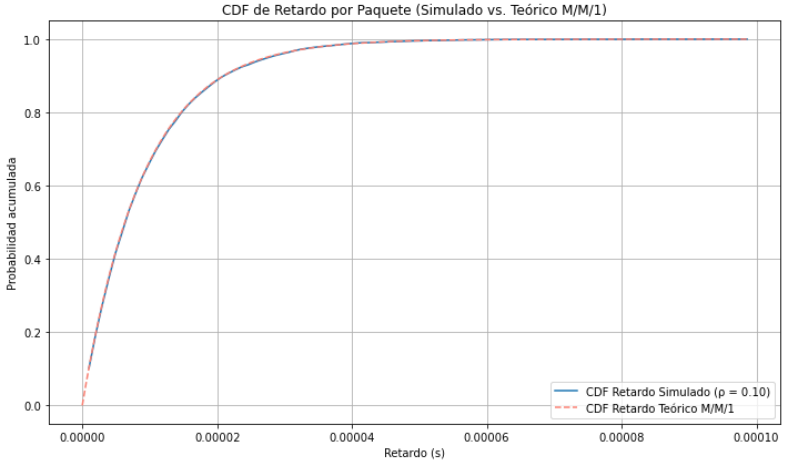
Figura 4.4: Validación empírica del retardo total simulado

En el gráfico de barras se observa que el valor medio del retardo simulado es de $8{,}2103 \times 10^{-5}$ s, mientras que el valor teórico correspondiente es de $8{,}1921 \times 10^{-5}$ s. Esta ligera diferencia, inferior al 0,25 %, confirma que la simulación reproduce fielmente el comportamiento del modelo M/M/1. La pequeña discrepancia puede atribuirse a factores inherentes a la simulación, como la variabilidad introducida por el tráfico aleatorio, o la naturaleza finita de las muestras analizadas. Aun así, la cercanía entre ambos valores valida tanto la implementación del generador de tráfico como la correcta parametrización del escenario simulado.

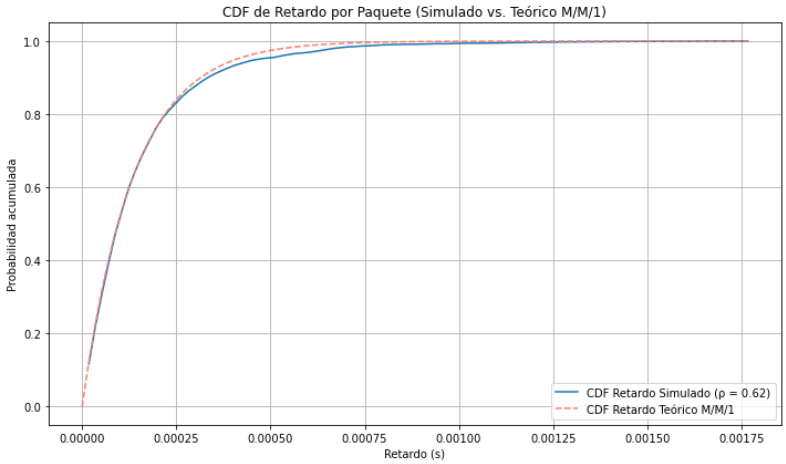
La segunda representación, un histograma, permite observar la forma de la distribución de retardos medida empíricamente. Se aprecia una clara concentración de valores en torno a retardos bajos, seguida de una cola que decrece exponencialmente. Esta forma es característica de un proceso M/M/1, donde los retardos individuales siguen una distribución exponencial, debido a la combinación de llegadas y servicios con distribuciones exponenciales negativas. El histograma, por tanto, no solo proporciona información visual sobre la dispersión de los datos, sino que también refuerza la validez del modelo probabilístico adoptado.

En conjunto, ambas visualizaciones permiten comprobar que el retardo total registrado en la simulación no solo se ajusta a los valores medios teóricos esperados, sino que también respeta la forma de la distribución de los tiempos que caracterizan un sistema M/M/1. Esto respalda la validez del enfoque seguido en el diseño y evaluación del experimento.

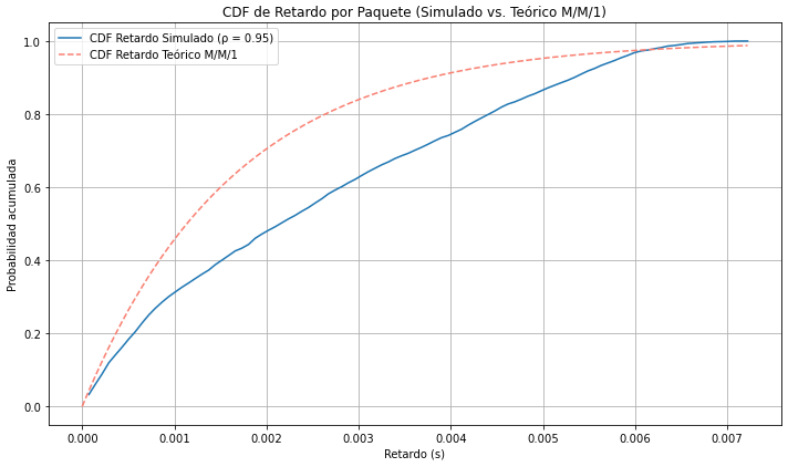
Además del análisis del retardo total medio y su distribución empírica, resulta especialmente ilustrativo estudiar cómo varía el comportamiento del sistema M/M/1 en función del grado de utilización ρ. Para ello, se han realizado varias simulaciones, manteniendo constantes las características del enlace y del tráfico, pero modificando la capacidad del canal para obtener distintos niveles de carga: $\rho = 0{,}1$, $\rho = 0{,}62$ y $\rho = 0{,}95$.



(a) $\rho = 0{,}1$



(b) $\rho = 0{,}62$



(c) $\rho = 0{,}95$

Figura 4.5: Comparación entre CDFs simuladas y teóricas del retardo total para distintos valores de ρ

La Figura 4.5 muestra las funciones CDF del retardo total para cada uno de estos escenarios, comparando los resultados obtenidos mediante simulación con las curvas teóricas correspondientes al modelo M/M/1. Cada CDF refleja la probabilidad de que un paquete experimente un retardo inferior a un determinado valor, permitiendo así observar tanto el comportamiento central como la cola de la distribución.

En la primera gráfica, correspondiente a una utilización baja ($\rho = 0{,}10$), se observa una excelente concordancia entre la CDF simulada y la curva teórica del modelo M/M/1. Ambas se solapan de forma prácticamente perfecta, lo que indica que el sistema opera en condiciones muy alejadas de la saturación. En este régimen, los retardos son generalmente bajos y estables. En concreto, se aprecia que el 95 % de los paquetes experimenta un retardo inferior a aproximadamente $1{,}5 \times 10^{-5}$ s, lo que evidencia una cola de espera casi inexistente y una alta eficiencia en el procesamiento.

En la segunda gráfica, con una utilización media ($\rho = 0{,}62$), la CDF simulada aún mantiene una forma cercana a la teórica, aunque se detecta una ligera desviación a partir del percentil 90. El retardo correspondiente al percentil 95 se sitúa en torno a $2{,}6 \times 10^{-5}$ s, lo que muestra un incremento notable respecto al caso anterior. Esto refleja el inicio de una mayor congestión en el sistema, que se manifiesta en una mayor dispersión de los retardos. A pesar de ello, la aproximación teórica sigue siendo válida y útil para la estimación general del comportamiento del sistema.

En la tercera gráfica, correspondiente a una utilización muy alta ($\rho = 0{,}95$), se aprecia una desviación considerable entre ambas curvas. La CDF simulada crece más lentamente, lo que indica una mayor variabilidad en los retardos. El percentil 95 alcanza aproximadamente $6{,}0 \times 10^{-5}$ s, un valor significativamente superior al de los casos anteriores. Este comportamiento confirma que, en condiciones cercanas a la saturación, la acumulación de paquetes y la aleatoriedad introducida por el tráfico generan colas más largas y retardos más elevados, que no son completamente capturados por la formulación analítica del modelo M/M/1.

Este análisis permite concluir que el modelo M/M/1 proporciona estimaciones teóricas precisas en condiciones de baja y media utilización del sistema, tanto en valor medio como en la forma de la distribución de retardos. No obstante, cuando la carga se aproxima a su límite teórico ($\rho \to 1$), la variabilidad crece de manera notable y el retardo se vuelve más impredecible, lo que reduce la capacidad del modelo para representar fielmente el comportamiento real. En estos casos, resulta recomendable utilizar modelos más generales o aplicar simulaciones con mayor profundidad para capturar adecuadamente la dinámica del sistema.

4.3.2. Modelo M/D/1 – Exponencial/Constante

A continuación, se procede al análisis del modelo M/D/1, el cual representa una evolución respecto al M/M/1 que se ha presentado anteriormente. Este sistema se caracteriza por mantener una distribución exponencial negativa para los tiempos entre llegadas, lo que implica un proceso de Poisson. Por otro lado, introduce un tiempo de servicio determinista y constante para todos los paquetes, lo que implica que el tamaño del paquete también sea constante. Esta modificación permite evaluar el impacto de la variabilidad en el servicio sobre el comportamiento del sistema, aislando el efecto de la aleatoriedad únicamente en el proceso de llegada.

Este modelo resulta particularmente útil en contextos donde los recursos de procesamiento o transmisión

operan a tasas constantes, como en enlaces dedicados o servidores que atienden solicitudes homogéneas. Su estudio permite identificar las mejoras en rendimiento que se obtienen al reducir la incertidumbre en los tiempos de servicio, manteniendo constante la carga del sistema.

En este escenario, el retardo medio en la cola, denotado como $\mathbb{E}[W_q]$, se puede obtener a partir de una expresión teórica (4.3), que observamos a continuación, bien conocida en la teoría de colas, y que mejora la fórmula correspondiente al modelo M/M/1 al introducir un factor correctivo, asociado a la menor variabilidad del servicio:

$$\mathbb{E}[W_q] = \frac{1}{2} \cdot \frac{\rho}{1-\rho} \cdot \mathbb{E}[T_s] \tag{4.3}$$

La presencia del coeficiente $\frac{1}{2}$ refleja una mejora significativa en el comportamiento del sistema respecto al modelo M/M/1, donde la variabilidad del servicio causa longitudes de cola más largas. Esta reducción se debe a que, al ser determinista el tiempo de servicio, se eliminan picos de congestión causados por servicios aleatoriamente largos.

En las secciones siguientes se compararán los resultados obtenidos en la simulación con los valores esperados según la formulación analítica del modelo, prestando especial atención tanto al valor medio del retardo como a la forma de su distribución acumulada.

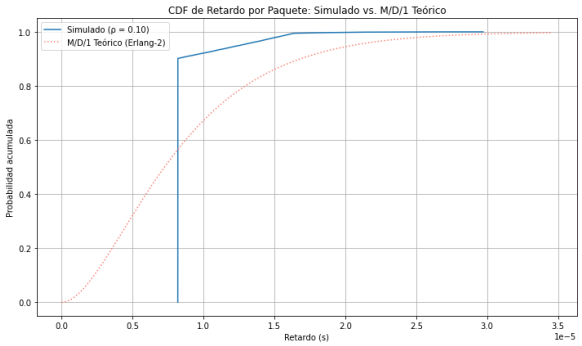
Con el objetivo de analizar el comportamiento del sistema M/D/1 bajo distintas condiciones de carga, se han realizado experimentos de simulación utilizando los mismos valores de utilización (ρ) empleados previamente en el estudio del modelo M/M/1.

Para cada uno de los escenarios evaluados ($\rho = 0{,}10$, $\rho = 0{,}625$ y $\rho = 0{,}95$), se han generado dos representaciones gráficas:

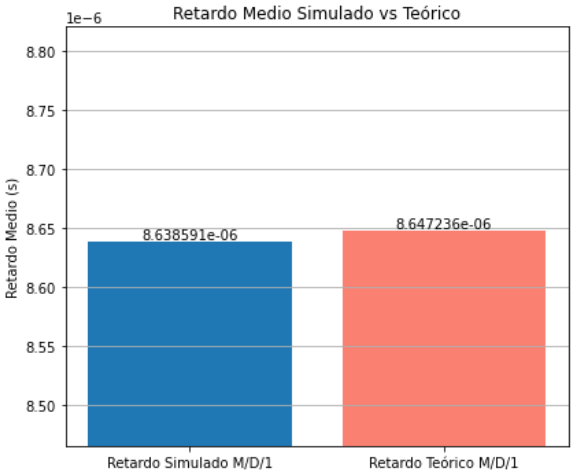
- Una función de distribución acumulada (CDF) del retardo por paquete, en la que se comparan los valores obtenidos a partir de la simulación con una aproximación teórica basada en la distribución Erlang de orden 2. Esta distribución es adecuada para modelar tiempos de servicio con baja varianza y ofrece una aproximación razonable en contextos donde el modelo M/D/1 no se ajusta con precisión.

- Un gráfico de barras que permite visualizar la comparación entre el retardo medio simulado y el retardo medio teórico, calculado a partir de la expresión analítica.

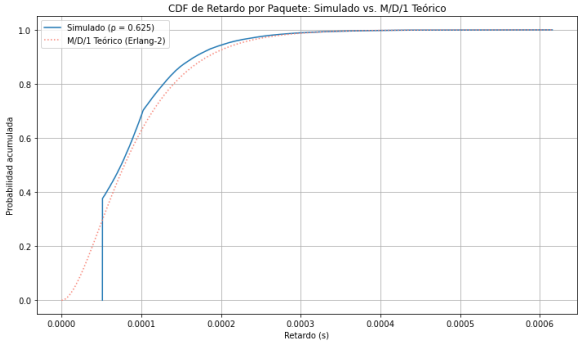
A continuación se presenta la Figura 4.6, organizadas en pares para cada valor de ρ, con la CDF a la izquierda y el gráfico de barras correspondiente a la derecha.



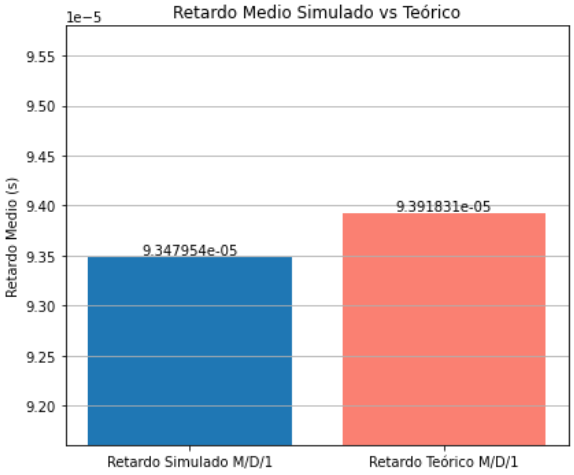
(a) CDF Retardo M/D/1 para $\rho = 0{,}10$



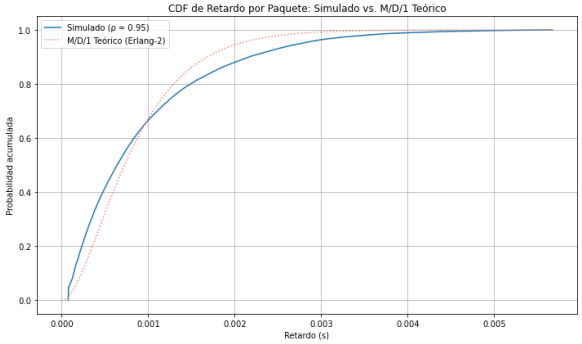
(b) Retardo medio: Simulado vs. Teórico para $\rho = 0{,}10$



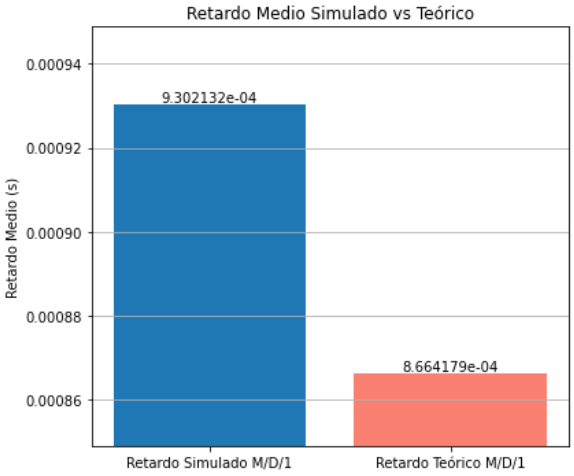
(c) CDF Retardo M/D/1 para $\rho = 0{,}62$



(d) Retardo medio: Simulado vs. Teórico para $\rho = 0{,}62$



(e) CDF Retardo M/D/1 para $\rho = 0{,}95$



(f) Retardo medio: Simulado vs. Teórico para $\rho = 0{,}95$

Figura 4.6: Resultados simulados y teóricos del modelo M/D/1 bajo distintas condiciones de carga.

En las figuras 4.6a, 4.6c y 4.6e pueden observarse varios aspectos relevantes que permiten analizar el comportamiento del modelo $M/D/1$ bajo distintas condiciones de tráfico.

En primer lugar, en las tres curvas simuladas se identifica un ascenso abrupto a partir de cierto instante en el eje temporal. Este punto coincide con el tiempo de servicio mínimo T_s asociado a cada configuración. Dado que el servidor se comporta de manera determinista, este tiempo se corresponde con la relación (4.4) entre el tamaño medio de los paquetes y la capacidad del enlace.

$$T_s = \frac{\text{Tamaño del paquete (bytes)} \times 8}{\text{Capacidad del enlace (bps)}} \tag{4.4}$$

Para cada caso evaluado se obtiene:

- Tamaño del paquete = 1024 bytes
- $\rho = 0{,}10$: Capacidad del enlace = 1 Gbps $\Rightarrow T_s = 8{,}19 \cdot 10^{-6}$ s
- $\rho = 0{,}625$: Capacidad del enlace = 0,16 Gbps $\Rightarrow T_s = 51{,}2 \cdot 10^{-6}$ s
- $\rho = 0{,}95$: Capacidad del enlace = 0,105 Gbps $\Rightarrow T_s = 78{,}02 \cdot 10^{-6}$ s

El inicio del crecimiento de la CDF coincide en todos los casos con el valor de T_s, lo que valida que el retardo mínimo observado en la simulación corresponde efectivamente al tiempo de servicio.

A medida que el valor de ρ aumenta, se observa un impacto claro sobre la forma de la distribución de retardos. Para cargas bajas, como $\rho = 0{,}10$, la mayoría de los paquetes experimentan un retardo muy cercano a T_s, con una dispersión mínima. En cambio, al incrementarse la carga (por ejemplo, para $\rho = 0{,}625$ y $\rho = 0{,}95$), la curva CDF se suaviza y desplaza hacia la derecha, indicando un aumento en la acumulación de paquetes en cola y una mayor variabilidad en los retardos.

En cuanto a la comparación teórica, se ha empleado la distribución Erlang de orden 2 como referencia. Esta es un caso particular de la distribución Gamma, una familia continua de distribuciones que modela el tiempo hasta la ocurrencia de k eventos en un proceso de Poisson. En concreto, la distribución Erlang-2 corresponde a una Gamma con parámetro de forma $k = 2$, y representa la suma de dos variables exponenciales independientes con la misma tasa. A diferencia del modelo $M/M/1$, que asume tiempos de servicio puramente exponenciales, la Erlang-2 permite capturar comportamientos con menor varianza, ajustándose mejor a entornos con tiempos de servicio constantes, como los descritos por el modelo $M/D/1$. Por ello, resulta adecuada como aproximación teórica del retardo acumulado, como se aprecia en las gráficas [21].

Por otro lado, las gráficas de barras mostradas en las Figuras 4.6b, 4.6d y 4.6f reflejan la evolución de la diferencia entre el retardo medio simulado y el teórico a medida que crece la carga del sistema. En condiciones de baja ocupación, el modelo $M/D/1$ ofrece una predicción precisa. Sin embargo, conforme el valor de ρ se aproxima a 1, el retardo simulado se incrementa notablemente respecto al valor analítico. Esto se debe a que, en situaciones de alta carga, incluso pequeñas fluctuaciones en los tiempos de llegada o servicio pueden generar colas significativas, lo cual no es capturado por el modelo idealizado. Como resultado, el retardo observado en la simulación supera al previsto teóricamente.

4.3.3. Modelo G/G/1 – Uniforme/Uniforme

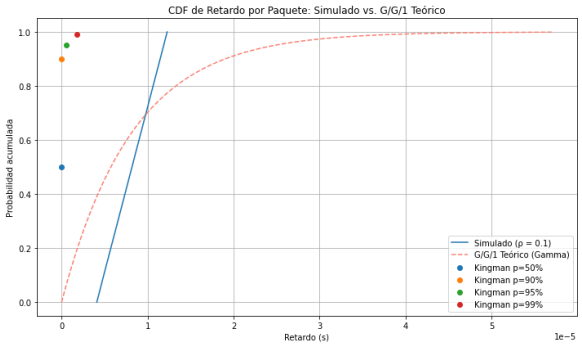
Con el objetivo de extender el análisis a un escenario más general, se ha considerado el comportamiento del sistema bajo el modelo $G/G/1$, donde tanto los tiempos entre llegadas como los tiempos de servicio presentan una distribución uniforme. Esta configuración permite evaluar situaciones más realistas con respecto a la variabilidad temporal, en contraste con los supuestos más idealizados de los modelos $M/M/1$ y $M/D/1$.

Se han utilizado los mismos niveles de carga ($\rho \approx 0{,}10$, $\rho \approx 0{,}62$ y $\rho \approx 0{,}95$), lo que facilita la comparación directa entre los distintos modelos evaluados. Para cada caso, se representa la CDF del retardo por paquete obtenida mediante simulación, junto con una aproximación teórica basada en una distribución Gamma, seleccionada por su capacidad para aproximar el comportamiento del modelo $G/G/1$ bajo ciertas condiciones de varianza. Además, se presenta un gráfico de barras comparativo entre el retardo medio simulado y el estimado teóricamente.

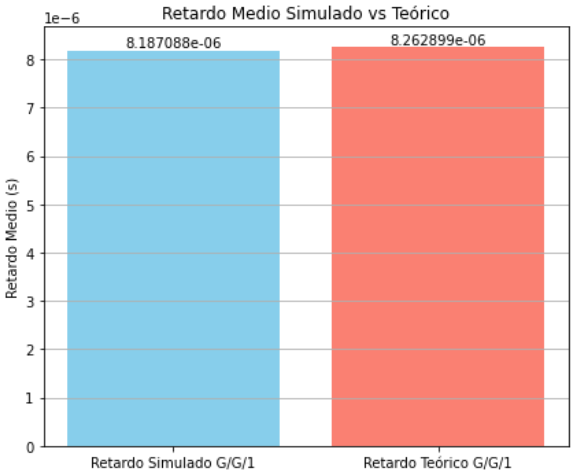
La estimación teórica del retardo medio en cola bajo el modelo $G/G/1$ se basa en la expresión (4.5):

$$\mathbb{E}[W_q] \leq \mathbb{E}[S] \cdot \frac{\rho}{1-\rho} \cdot \frac{C^2[T] + C^2[S]}{2} \tag{4.5}$$

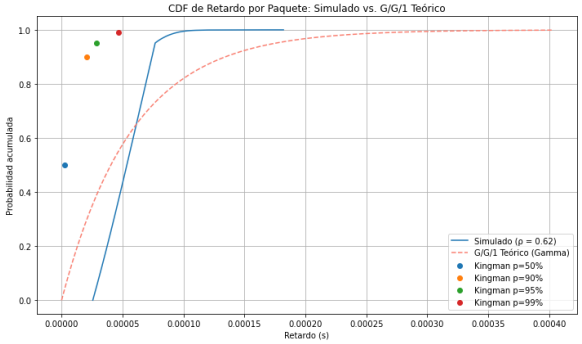
donde $C^2[T]$ y $C^2[S]$ son los coeficientes de variación cuadráticos del tiempo entre llegadas y del tiempo de servicio, respectivamente. A continuación, en la Figura 4.7, se muestran los resultados obtenidos para las tres situaciones de carga evaluadas.



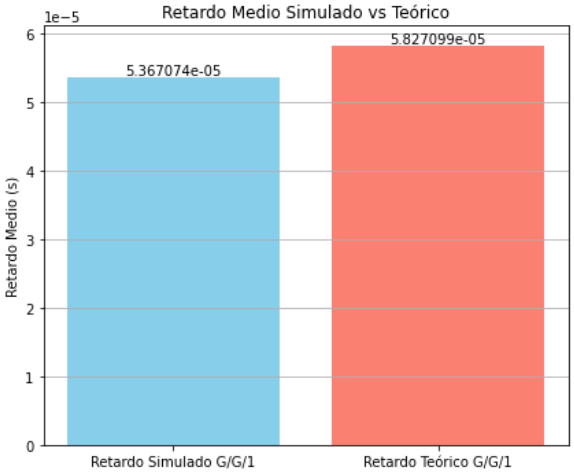
(a) CDF Retardo M/D/1 para $\rho = 0{,}10$



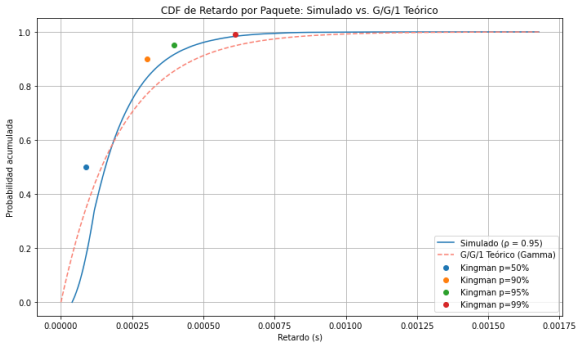
(b) Retardo medio: Simulado vs. Teórico para $\rho = 0{,}10$



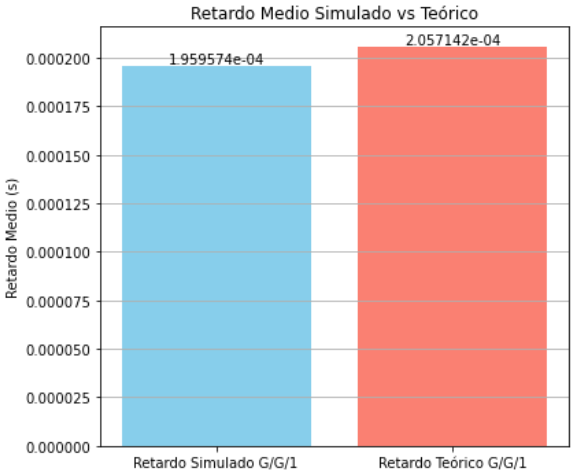
(c) CDF Retardo M/D/1 para $\rho = 0{,}62$



(d) Retardo medio: Simulado vs. Teórico para $\rho = 0{,}62$



(e) CDF Retardo M/D/1 para $\rho = 0{,}95$



(f) Retardo medio: Simulado vs. Teórico para $\rho = 0{,}95$

Figura 4.7: Comparación de resultados simulados y teóricos del modelo M/D/1 bajo distintas condiciones de carga.

En concreto, cabe destacar que los tamaños de los paquetes están modelados por una distribución uniforme en el intervalo $[512, 1536]$ bytes, lo que da lugar a un valor medio de 1024 bytes, calculado como $\mu = \frac{b+a}{2} = \frac{1536+512}{2}$. Por otro lado, los tiempos entre llegadas también siguen una distribución uniforme, generada a partir de un valor base al que se aplica una variación del $50\,\%$, es decir, el extremo inferior se obtiene multiplicando por $0{,}5$ y el superior por $1{,}5$.

A medida que se incrementa la carga del sistema, representada por el parámetro ρ, se observa un cambio significativo en la forma de las en las gráficas 4.7a. 4.7c y 4.7e. Para valores bajos de ρ, la mayoría de los paquetes experimentan retardos cercanos al mínimo, lo que se refleja en una subida rápida y pronunciada de la CDF. En cambio, conforme ρ aumenta, las curvas se suavizan y se desplazan hacia la derecha, indicando una mayor dispersión de los retardos y un comportamiento más alejado del caso determinista. Este fenómeno se explica por la acumulación de paquetes en la cola cuando el sistema se aproxima a condiciones de saturación.

Asimismo, se ha incluido en las gráficas la predicción obtenida mediante la fórmula de Kingman para distintos percentiles del retardo, empleando la expresión (4.6).

$$W_q^{(p)} = \text{máx}\left\{0,\ E[S] \cdot \frac{1}{1-\rho} \cdot \frac{C^2[T] + C^2[S]}{2} \cdot \ln\left(\frac{\rho}{1-p}\right)\right\} \tag{4.6}$$

Esta ecuación estima el retardo en cola que no se supera con una probabilidad p, lo que resulta útil para evaluar el rendimiento en términos de garantías de calidad de servicio. Por ejemplo, en la Figura 4.7e se observa que el $99\,\%$ de los paquetes tienen un retardo inferior a 625 μs

Por último, se incluye también la comparación con la distribución Gamma, utilizada como aproximación teórica de la distribución empírica del retardo. Esta elección se justifica porque la distribución Gamma es una familia flexible de distribuciones continuas que permite modelar tiempos positivos con distintas formas. En este contexto, proporciona un buen ajuste al comportamiento simulado, especialmente cuando las distribuciones de entrada no siguen un patrón exponencial ni determinista, como ocurre en el modelo $G/G/1$.

4.4. Modelo realista

En esta opción, el escenario de red modelado adquiere un carácter más realista y estructurado. A diferencia del enfoque sintético, aquí se simula un entorno determinista, donde se considera la generación de tráfico proveniente de hasta cuatro unidades de radio (O-RU), cada una emitiendo su propio flujo hacia una unidad de agregación. Cada flujo generado se caracteriza por una tasa de llegada λ_i, la cual depende de parámetros físicos como la numerología, el ancho de banda o el número de capas MIMO.

Todos los flujos se agregan en un nodo intermedio (simulado como un sumador lógico), y posteriormente se transmiten a través de un único enlace común hacia una unidad central (O-DU), que actúa como destino final. Esta configuración permite estudiar el impacto del tráfico agregado sobre el rendimiento del sistema, así como evaluar la escalabilidad del canal fronthaul ante distintos niveles de carga concurrente.

La Figura 4.8 representa esta topología, en la que los flujos individuales se combinan antes de llegar a la O-DU.

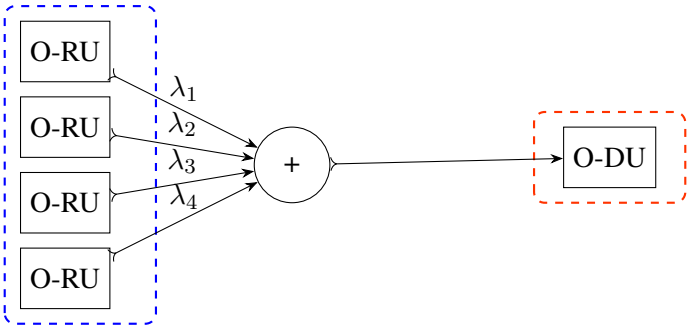


Figura 4.8: Esquema red fronthaul con hasta 4 flujos agregados desde diferentes O-RU hacia una O-DU

4.4.1. Gestión y organización automatizada de resultados

Para garantizar la trazabilidad y facilitar el análisis comparativo, se ha desarrollado un mecanismo automático de organización de los resultados obtenidos tras cada simulación. Al finalizar cada ejecución, se genera una carpeta principal con el nombre `experimentos__Nflujos`, donde `N` representa el número de flujos concurrentes configurados en esa simulación (desde 1 hasta 4). Dentro de esta carpeta, se crean subdirectorios diferenciados por iteración (`iteracion_1`, `iteracion_2`, etc.), permitiendo gestionar de forma sistemática las distintas ejecuciones realizadas bajo la misma configuración de flujos.

Cada iteración representa una ejecución con parámetros ligeramente distintos: si bien se mantiene el número de flujos constante, las características de cada flujo (numerología, ancho de banda, capas, etc.) se determinan de forma aleatoria a partir de un conjunto predefinido de valores posibles. Esta variabilidad controlada permite simular distintos escenarios realistas, manteniendo condiciones comparables. Los parámetros seleccionados en cada ejecución quedan registrados automáticamente en un archivo de texto llamado `parametros_iterX.txt`, que actúa como registro de referencia para garantizar la trazabilidad y la reproducibilidad de los experimentos.

Además, con el fin de unificar la información necesaria para el análisis del sistema como un todo, los archivos de transmisión generados por cada cliente se combinan en un único fichero temporal (`todos_clientes_ordenado.txt`), ordenado cronológicamente por marca de tiempo. Esta fusión es fundamental para poder calcular métricas globales del sistema como los coeficientes de variación C_T^2 y C_S^2, necesarios para estimar el retardo medio mediante la fórmula generalizada de Kingman para sistemas G/G/1.

Para mantener la consistencia estadística entre simulaciones, se ha empleado un control explícito de la aleatoriedad mediante el uso de una semilla fija en el motor de simulación de `NS-3`. Esta semilla garantiza que, para un mismo número de iteración, los resultados sean reproducibles incluso ante múltiples ejecuciones del mismo código. Asimismo, los valores aleatorios utilizados (selección de numerología, BW, layers) se obtienen utilizando generadores uniformes configurados con un rango específico para cada parámetro, lo que permite controlar la variabilidad sin comprometer la aleatoriedad.

```
1 SeedManager::SetSeed(12345);      // Fixed seed to ensure reproducibility
2 SeedManager::SetRun(runNumber); // Variation between executions
```

Código 4.1: Inicialización de semilla y número de ejecución en `NS-3`

4.4.2. Análisis de rendimiento con patrones TDD y flujos

Con el objetivo de evaluar el comportamiento del sistema bajo distintas configuraciones de flujos, se realizaron dos simulaciones utilizando un patrón TDD. En ambas simulaciones, se utilizó un patrón TDD "DDDDDDDDDD", donde todos los slots se asignan a downlink (D), permitiendo que cada flujo utilice todo el ancho de banda disponible sin interrupciones para uplink (U).

Este patrón simplificado proporciona un escenario de transmisión continua en downlink, lo que permite observar el rendimiento del sistema bajo la carga constante de tráfico proveniente de diferentes unidades de radio (O-RU).

A continuación, se analizan las diferencias observadas entre las simulaciones con 3 y 4 flujos.

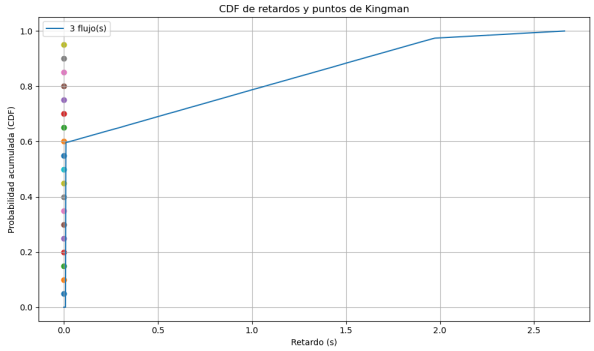


Figura 4.9: CDF para 3 flujos

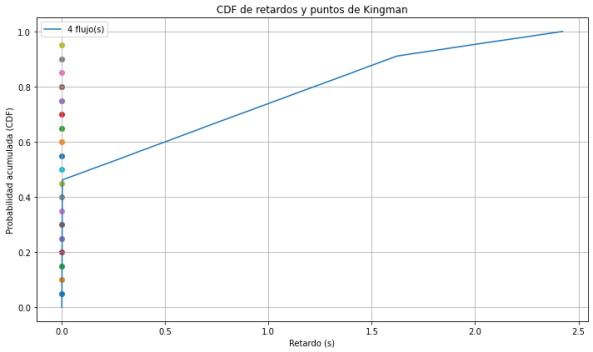


Figura 4.10: CDF para 4 flujos

En ambas gráficas, la función CDF sigue una forma válida: inicia en 0 y crece de manera monótona hasta alcanzar el valor de 1, conforme a lo que se espera de cualquier distribución acumulada. Este comportamiento asegura que, desde el punto de vista matemático, los retardos están siendo procesados correctamente. A continuación, se presentan las dos interpretaciones de la distribución de los retardos para los dos casos simulados.

En la Figura 4.9, la pendiente inicial de la CDF asciende rápidamente cerca del origen, lo que indica que la mayoría de los paquetes se entregan con retardos muy pequeños, cercanos a 0 segundos. Posteriormente, la curva experimenta un crecimiento más gradual hasta alcanzar el 100 % de los paquetes, momento en el cual se observan algunos retardos mayores, representados como valores atípicos en la gráfica.

Por otro lado, en la Figura 4.10, el comportamiento inicial es similar, con una concentración de la mayoría de los retardos cercanos a 0 segundos, lo que se traduce en una nube de puntos alrededor del eje vertical. Sin embargo, a medida que se avanza en la curva, se observa un salto más pronunciado hacia valores más altos, alrededor de los 2.5 segundos. Esto indica que empiezan a aparecer más paquetes con retardos inusualmente grandes. Este comportamiento sugiere que el sistema está experimentando una carga más alta, como lo refleja el aumento en el valor de la métrica ρ (de 0.4047 a 0.4221), lo que genera la aparición de colas y eventos de congestión.

La comparación entre ambas simulaciones es coherente. Cuando el número de flujos aumenta de 3 a 4, la carga del sistema también incrementa, aunque con una variación mínima. Esto se refleja en la CDF, donde, a pesar de que con cuatro flujos se mantiene una alta concentración de retardos pequeños, también se observa un mayor número de situaciones con retardos grandes, lo que provoca una extensión de la cola

de la distribución.

Por otro lado, los resultados muestran que la varianza del tiempo de servicio aumenta de manera muy ligera al añadir más flujos. Para el caso de 3 flujos, la varianza es de $9{,}671 \times 10^{-11}\ s^2$, mientras que para 4 flujos es de $1{,}114 \times 10^{-10}\ s^2$. Este comportamiento puede explicarse por la baja variabilidad del tráfico generado por los flujos, lo que produce un coeficiente de variación muy bajo. Al aplicar la fórmula de Kingman para estimar el retardo medio en sistemas G/G/1, los coeficientes de variación tienden a aproximarse a cero, lo que invalida el uso de este modelo para estimar el rendimiento en tales condiciones.

El modelo de Kingman se basa en la premisa de que los tiempos de servicio y las llegadas de los paquetes siguen distribuciones con una varianza significativa. Cuando la varianza del tiempo de servicio es demasiado baja, como ocurre en las simulaciones realizadas, los cálculos del modelo de Kingman se ven afectados, ya que sus coeficientes de variación se acercan a cero. Esto hace que la estimación del retardo medio sea inapropiada para este tipo de escenarios con flujos de baja variabilidad. Así, se concluye que el modelo de Kingman no es adecuado para estimar el rendimiento bajo condiciones de tráfico con baja variabilidad.

Tras observar los resultados con el patrón "DDDDDDDDDD", se procede a modificar el patrón de transmisión para el caso de 3 flujos a un patrón alternante "DUDUDUDUDU", en el cual los slots de transmisión se alternan entre downlink (D) y uplink (U). En este patrón, se transmiten datos solo en los slots de downlink, mientras que los slots de uplink no se utilizan para transmitir datos, sino que sirven como periodos de inactividad para permitir la alternancia en el acceso al canal. Este patrón alternante introduce una mayor variabilidad en el acceso al canal y genera un escenario más dinámico en el que los flujos deben esperar durante los periodos de uplink.

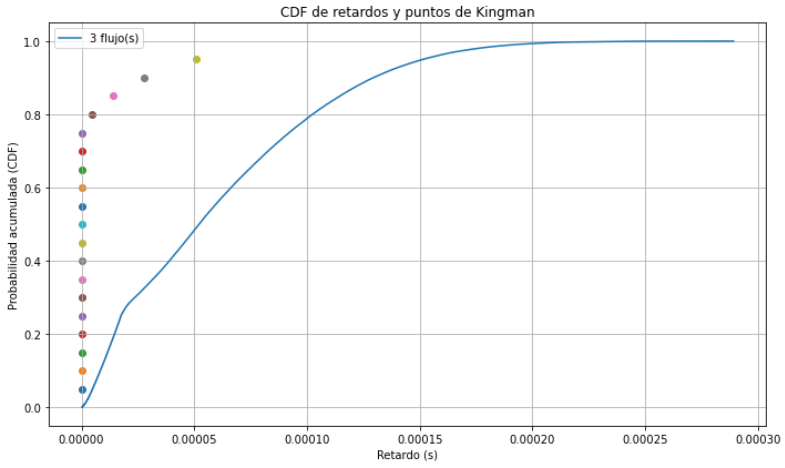


Figura 4.11: Distribución de retardos para el patrón DUDUDUDUDU con 3 flujos.

La Figura 4.11 anterior, corresponde al patrón "DUDUDUDUDU"para 3 flujos, donde se puede observar cómo cambia la distribución de los retardos en comparación con el patrón "DDDDDDDDDD".

Así, la forma de la CDF observada tiene sentido y es coherente con las expectativas. Al utilizar un patrón alternado de downlink (D) y uplink (U), la distribución de los retardos se concentra más en los valores más bajos, lo cual se debe a la intercalación de los slots de uplink entre los slots de downlink.

El efecto práctico de este patrón es que, al haber menos paquetes acumulados por unidad de tiempo, los retardos son más bajos y su dispersión es menor. Esto se refleja en la CDF, que crece rápidamente y se estabiliza en valores bajos de retardo, tal como se observa en la gráfica proporcionada.

En comparación con el patrón anterior, donde los paquetes se transmiten de manera continua en todos los slots sin interrupción, se genera una mayor presión sobre el sistema, lo que incrementa la probabilidad de retardos mayores. En el patrón DUDUDUDUDU, al transmitir en menos slots efectivos, el sistema experimenta períodos de inactividad entre las transmisiones, lo que permite reducir la variabilidad de los retardos y concentrarlos en valores más bajos. No obstante, se observa el mismo comportamiento que en el caso anterior, ya que el modelo de Kingman no puede ajustarse adecuadamente a los resultados simulados debido a que las varianzas del tiempo de servicio continúan siendo excesivamente pequeñas.

En conclusión, las simulaciones muestran que el patrón "DUDUDUDUDU"mejora el manejo de los flujos al reducir los retardos, comparado con el patrón "DDDDDDDDDD". El primer patrón genera menos congestión al intercalar periodos de inactividad. Sin embargo, ambos patrones demuestran que el modelo de Kingman no es útil para modelar los escenarios analizaddos debido a la baja variabilidad de tráfico.

Capítulo 5

Conclusiones

El trabajo desarrollado ha permitido llevar a cabo un análisis exhaustivo del rendimiento de un enlace fronthaul mediante la simulación y evaluación comparativa de diferentes modelos de tráfico, aportando resultados sólidos y contrastados frente a las expresiones analíticas de la teoría de colas. A lo largo de este estudio se han explorado y validado los modelos M/M/1, M/D/1 y G/G/1, abarcando desde escenarios sintéticos con alta aleatoriedad hasta configuraciones más realistas con distribuciones arbitrarias de llegada y servicio.

En primer lugar, se ha verificado la correcta generación de tráfico en el simulador, asegurando que las distribuciones empíricas obtenidas a partir de los registros cliente-servidor se ajustan a los modelos estadísticos previstos. Este paso ha sido esencial para garantizar la validez de los resultados posteriores. A partir de esta base, se ha evaluado el retardo total por paquete y su dependencia con la carga del sistema, comparando los valores medios y las distribuciones acumuladas simuladas frente a las teóricas. Los resultados muestran que, en condiciones de carga baja y media, los modelos M/M/1 y M/D/1 ofrecen predicciones precisas y alineadas con las simulaciones, validando así los fundamentos teóricos. En particular, la reducción de la variabilidad en el servicio del modelo M/D/1 se traduce en mejoras tangibles de rendimiento, confirmadas tanto en los valores medios como en la distribución del retardo. Por otro lado, al aproximarse a situaciones de saturación ($\rho \to 1$), se ha observado un incremento notable en la dispersión de los retardos y una desviación progresiva respecto a las curvas analíticas, fenómeno inherente a la naturaleza estocástica de los procesos de llegada y servicio bajo cargas elevadas.

La extensión al modelo G/G/1 ha permitido estudiar casos con distribuciones uniformes tanto en llegadas como en servicio, aproximando con mayor fidelidad el tráfico fronthaul característico de escenarios 5G. En este contexto, la aplicación de la fórmula de Kingman para la estimación de percentiles ha demostrado ser una herramienta útil para anticipar el comportamiento de los retardos y proporcionar garantías de calidad de servicio. Asimismo, la comparación entre los valores simulados y las aproximaciones teóricas basadas en distribuciones Gamma ha evidenciado la flexibilidad del marco analítico para adaptarse a configuraciones con mayor complejidad estadística.

Por otro lado, en el caso del modelo realista, donde se considera el tráfico agregado de hasta cuatro O-RU hacia una O-DU, se han explorado distintas configuraciones TDD para evaluar el impacto sobre la distribución de retardos. Con un patrón continuo de renuras (*slots*) downlink (`DDDDDDDDDD`), las CDF obtenidas muestran un incremento progresivo del retardo (así como la aparición de valores más elevados),

al pasar de tres a cuatro flujos, reflejando un aumento en la carga (ρ). Sin embargo, al alternar los slots mediante un patrón `DUDUDUDUDU`, los resultados muestran una clara mejora: la CDF se concentra en valores bajos de retardo y la dispersión se reduce, evidenciando un sistema menos congestionado gracias a los periodos de inactividad intercalados entre transmisiones. A pesar de ello, se observa que el modelo de Kingman no logra ajustar correctamente los resultados simulados, debido a la escasa variabilidad en los tiempos de servicio, lo que limita la utilidad de este modelo bajo escenarios con baja varianza.

Además de los resultados obtenidos, este trabajo aporta una metodología clara y reproducible, sustentada en un entorno de análisis modular mediante scripts en Python y cuadernos Jupyter. Esta herramienta no solo facilita la validación progresiva de cada experimento, sino que sienta las bases para futuras investigaciones y permite extender fácilmente el estudio a nuevas configuraciones de red, diferentes numerologías o algoritmos de gestión de tráfico más avanzados.

Como líneas de trabajo futuras, se propone ampliar el entorno experimental para incluir un mayor número de flujos concurrentes, así como incorporar modelos de tráfico específicos de escenarios 6G que reflejen patrones más complejos y heterogéneos de generación de paquetes. Asimismo, resulta de especial interés la evaluación de técnicas adaptativas de asignación de recursos, tales como algoritmos heurísticos o enfoques basados en aprendizaje automático, que permitan optimizar la latencia y mejorar la calidad de servicio en tiempo real bajo condiciones de alta carga.

Asimismo, una dirección prometedora consiste en investigar qué modelo teórico podría ajustarse mejor al comportamiento observado en las simulaciones con tráfico 5G y los parámetros físicos considerados (numerología, ancho de banda, capas MIMO, entre otros). Identificar o desarrollar un modelo teórico que represente con mayor fidelidad estas condiciones permitiría disponer de expresiones analíticas más precisas y, por tanto, de mejores herramientas para el diseño y dimensionamiento de redes fronthaul modernas.

En conjunto, estas extensiones no solo enriquecerían el análisis ya realizado, sino que también abrirían la puerta a contribuciones relevantes en el ámbito de las arquitecturas de red de nueva generación, ofreciendo un marco más robusto y adaptado a la complejidad de los sistemas de comunicación actuales y futuros.

Bibliografía

[1] V. Sahu, N. Sahu y R. Sahu. "Challenges and Opportunities of 5G Network: A Review of Research and Development". En: *American Journal of Electrical and Computer Engineering* (2024). DOI: 10.11648/j.ajece.20240801.12. URL: https://doi.org/10.11648/j.ajece.20240801.12.

[2] A. Checko, H. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger y L. Dittmann. "Cloud RAN for Mobile Networks—A Technology Overview". En: *IEEE Communications Surveys & Tutorials* 17.1 (2015), págs. 405-426. DOI: 10.1109/COMST.2014.2355255.

[3] G. O. Pérez, J. A. Hernández y D. Larrabeiti. "Fronthaul Network Modeling and Dimensioning Meeting Ultra-Low Latency Requirements for 5G". En: *Journal of Optical Communications and Networking* 10.6 (2018), págs. 573-581. DOI: 10.1364/JOCN.10.000573. URL: https://doi.org/10.1364/JOCN.10.000573.

[4] ADSLZone. *5G: qué es, cómo funciona y qué ventajas ofrece*. Online: https://www.adslzone.net/reportajes/telefonia/5g/. 2025.

[5] Universidad VIU. *Evolución de la red de comunicación móvil del 1G al 5G*. Web article. Online: https://www.universidadviu.com/es/actualidad/nuestros-expertos/evolucion-de-la-red-de-comunicacion-movil-del-1g-al-5g. 2024. (Visitado 15-07-2025).

[6] F. Khan, L. Diez, L. M. Contreras, Ó. Gil, E. Serna, D. Gregoratti y R. Agüero. "QoS-Aware scheduling policies for Open Fronthaul transport networks". En: *2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. 2024, págs. 335-340. DOI: 10.1109/MeditCom61057.2024.10621072.

[7] O-RAN Software Community. *O-DU PHY Architecture Overview*. Online: https://docs.o-ran-sc.org/projects/o-ran-sc-o-du-phy/en/latest/Architecture-Overview_fh.html. 2023.

[8] MICM. *Las redes C-RAN (Cloud Radio Access Network)*. Online: https://micm.es/noticias/las-redes-cran-cloud-radio-access-network/. 2019.

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

[10] ShareTechnote. *O-RAN – eCPRI Protocol Overview*. Online: https://www.sharetechnote.com/html/OpenRAN/OR_eCPRI.html. 2023.

[11] I. 1. W. Group. "Next Generation Fronthaul Interface (NGFI): Standardization and Energy Efficiency Strategies". En: *IEEE Communications Standards Magazine* 5.2 (2021), págs. 48-54.

[12] ResearchGate. *Open-RAN architecture split option 7.2x for DL Processing blocks*. Web page (ResearchGate). Online: https://www.researchgate.net/figure/Open-RAN-architecture-split-option-7-2x-for-DL-Processing-blocks-with-a-bold-text-are_fig3_369468827. 2025. (Visitado 17-07-2025).

[13] ShareTechnote. *5G Frame Structure — Radio Frame and Slot Configuration for* $\mu = 0$. https://www.sharetechnote.com/html/5G/5G_FrameStructure.html#RadioFrame_Structure_SlotConfig0_u_0. Accedido: 17 de julio de 2025.

[14] W. Castle. *What is the significance of QoS in private 5G networks?* https://wraycastle.com/es/blogs/glossary/what-is-the-significance-of-qos-in-private-5g-networks. Accedido: 17 de julio de 2025.

[15] J. P. Garcia. *Apuntes de Teoría de Colas*. https://personales.upv.es/jpgarcia/LinkedDocuments/Teoriadecolasdoc.pdf. Accedido: 17 de julio de 2025.

[16] E. L. Rubio. *Tema 5: Teoría de colas*. Documento PDF. Online: http://lcc.uma.es/~ezeqlr/ios/Tema5.pdf. 2025. (Visitado 17-07-2025).

[17] M. Lantigua. *Ecuaciones de Flujo Little*. Documento PDF. Online: https://es.scribd.com/document/654349768/Ecuaciones-de-Flujo-Little. 2023. (Visitado 17-07-2025).

[18] Estadistica.net. *Teoría de Colas*. 2025. URL: https://estadistica.net/IO/7-2-TEORIA-COLAS.pdf.

[19] Universidad de Cádiz. *Teoría de Colas – OCW UCA*. 2025. URL: https://ocw.uca.es/course/view.php?id=112.

[20] cppreference.com. *std::exponential_distribution<>– C++ Reference*. Web page (C++ reference). Online: https://en.cppreference.com/w/cpp/numeric/random/exponential_distribution.html. 2025. (Visitado 15-07-2025).

[21] cppreference.com. *std::gamma_distribution<>– C++ Reference*. Web page (C++ reference). Online: https://en.cppreference.com/w/cpp/numeric/random/gamma_distribution.html. 2025. (Visitado 15-07-2025).