UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Diseño y Desarrollo de una Arquitectura de Monitorización para una Red de Telecomunicaciones Óptico-Móvil

Informe de Proyecto de Grado presentado por

Leandro Alfonso, Nicolás Rivoir

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Alberto Castro
Lucas Inglés, Claudina Rattaro

Montevideo, 26 de agosto de 2024

Agradecimientos

Durante la clausura de este proyecto, sentimos que la mejor manera de describir nuestra experiencia es citando el proverbio: “Si quieres ir rápido, camina solo. Si quieres llegar lejos, ve acompañado.”. Este proyecto es el resultado del esfuerzo y apoyo de muchas personas a lo largo del camino, y es por esto que dedicamos los siguientes agradecimientos.

A nuestras familias: Gracias por su apoyo incondicional durante toda nuestra carrera. Su comprensión y constante aliento fueron el motor para superar los desafíos presentados y seguir adelante con determinación.

A nuestros tutores: Alberto, Claudina y Lucas, gracias por ser guías en nuestros últimos pasos, por la confianza y por su paciencia frente a las incertidumbres que enfrentamos a lo largo de este proyecto.

Finalmente, nos gustaría dar un agradecimiento a nuestra institución, sin la cual no tendríamos las herramientas que nos permitieron llevar adelante el presente proyecto. Gracias por ser incansable, desafiándonos semestre a semestre, pero siempre brindándonos el apoyo necesario de la mano de los docentes, que siempre estuvieron ahí para nosotros. Gracias por nunca bajar la vara en cuanto a lo que se espera de nosotros.

A todos los que han sido parte de este proyecto, gracias por su invaluable contribución y por ayudarnos a llegar hasta aquí. Este logro no habría sido posible sin ustedes.

Resumen

En la última década, la industria de las telecomunicaciones ha experimentado un crecimiento significativo debido al aumento en la demanda de ancho de banda, impulsada por aplicaciones de alto consumo como la transmisión de video en alta definición, juegos en línea y servicios de comunicación en tiempo real. Este aumento en la demanda ha llevado a la necesidad de tecnologías avanzadas como las redes 5G para satisfacer las crecientes necesidades de conectividad. Las redes 5G prometen mejoras significativas en términos de velocidad de transmisión, latencia y confiabilidad, lo que las hace ideales para soportar el creciente tráfico de datos y la diversidad de aplicaciones emergentes.

La monitorización de redes desempeña un papel esencial en la implementación de algoritmos de asignación de recursos. Proporciona una visibilidad continua del estado de red, lo que permite optimizar el uso de los recursos en tiempo real y adaptarse a las condiciones cambiantes del tráfico. Además, una monitorización efectiva es crucial para la detección temprana de problemas (eg. una falla de enlace) y la aplicación de estrategias de mitigación, asegurando así un servicio robusto y de alta calidad para los usuarios finales.

En este proyecto se diseñó e implementó una arquitectura de monitorización para una red de telecomunicaciones óptico-móvil utilizando el simulador ns-3 y el módulo de extensión 5G-LENA. La motivación detrás de este trabajo se centra en la necesidad de contar con herramientas avanzadas que permitan no solo simular redes 5G, sino también monitorizar su rendimiento, detectar fallas y localizar enlaces defectuosos de manera proactiva.

La arquitectura propuesta permite la extracción en tiempo de simulación de un conjunto de métricas que cuantifican el estado general de red, así como la identificación y notificación de enlaces con problemas de rendimiento. Esta capacidad se logra mediante el uso de módulos de ns-3, desarrollados en el contexto de este proyecto, que extienden la funcionalidad del "*FlowMonitor*" (una herramienta integrada en ns-3 para recopilar estadísticas detalladas de flujo) para incluir la recopilación detallada de estadísticas por paquete y la identificación dinámica de enlaces problemáticos.

El proyecto incluye la implementación de una herramienta que monitoriza y detecta fallas en redes óptico-móviles, ajustando dinámicamente el tiempo de extracción de métricas según el estado de la red (tasa elástica de muestreo). La herramienta es escalable y configurable, permitiendo su uso en diversos escenarios de red y perfiles de tráfico. Además, se definió un formato de *logs* en XML

para facilitar la revisión y análisis de los resultados de la simulación. Como productos de este proyecto, se dejan a disposición de la comunidad un conjunto de datasets generados durante las simulaciones, y el código fuente completo disponible en un repositorio público.

En resumen, este proyecto proporciona una solución integral para la monitorización de redes 5G, ofreciendo una herramienta que no solo facilita la simulación de estas redes avanzadas, sino que también permite una gestión eficiente y proactiva de la red mediante la detección y localización de fallas. Esta arquitectura de monitorización es un paso crucial hacia la implementación de redes inteligentes y autosustentables, capaces de gestionar sus recursos y detectar problemas de manera autónoma, reduciendo así los costos operacionales y mejorando la calidad del servicio.

Palabras clave: Redes 5G, ns-3, 5G-LENA, monitorización, simulación, redes óptico-móviles

Índice general

Acrónimos

- **3GPP**: 3rd Generation Partnership Project
- **5G**: Quinta Generación (de redes móviles)
- **6G**: Sexta Generación (de redes móviles)
- **ANTEL**: Administración Nacional de Telecomunicaciones
- **BWP**: Bandwidth Part (Parte de Ancho de Banda)
- **CA**: Carrier Aggregation (Agrupación de Portadoras)
- **CBR**: Constant Bit Rate (Tasa de Bits Constante)
- **CC**: Component Carrier (Portador de Componentes)
- **CSIC**: Comisión Sectorial de Investigación Científica
- **CTTC**: Centro Tecnológico de Telecomunicaciones de Cataluña
- **CU**: Centralized Unit (Unidad Centralizada)
- **DevOps**: Development and Operations (Desarrollo y Operaciones)
- **DINATEL**: Dirección Nacional de Telecomunicaciones
- **DL**: Downlink (Enlace Descendente)
- **DU**: Distributed Unit (Unidad Distribuida)
- **eMBB**: Enhanced Mobile Broadband (Banda Ancha Móvil Mejorada)
- **EON**: Elastic Optical Network (Red Óptica Elástica)
- **EPC**: Evolved Packet Core (Núcleo de Paquetes Evolucionado)
- **5GCN**: 5G Core Network(Núcleo de red 5G)
- **FDD**: Frequency Division Duplexing (Duplexación por División de Frecuencia)
- **FIFO**: First In, First Out (Primero en Entrar, Primero en Salir)

- **FN**: False Negative (Falso Negativo)
- **FP**: False Positive (Falso Positivo)
- **FTP**: File Transfer Protocol (Protocolo de Transferencia de Archivos)
- **gNB**: Next Generation NodeB (Nodo B de Próxima Generación)
- **GPL**: General Public License (Licencia Pública General)
- **GPLv2**: General Public License version 2
- **GUI**: Graphical User Interface (Interfaz Gráfica de Usuario)
- **HTTP**: Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto)
- **I+D**: Investigación y Desarrollo
- **IDE**: Integrated Development Environment (Entorno de Desarrollo Integrado)
- **INET**: Internet Networking Framework
- **IoT**: Internet of Things (Internet de las Cosas)
- **IP**: Internet Protocol (Protocolo de Internet)
- **ISP**: Internet Service Provider (Proveedor de Servicios de Internet)
- **LAN**: Local Area Network (Red de Área Local)
- **LENA**: LTE-EPC Network Simulator (Simulador de Red LTE-EPC)
- **LGPL**: Lesser General Public License (Licencia Pública General Menor)
- **LTE**: Long-Term Evolution
- **MAC**: Medium Access Control (Control de Acceso al Medio)
- **MIMO**: Multiple Input Multiple Output (Múltiple Entrada Múltiple Salida)
- **mMTC**: Massive Machine Type Communications (Comunicaciones Masivas Tipo Máquina)
- **NED**: Network Description Language (Lenguaje de Descripción de Red)
- **NMS**: Network Management Station (servidor que ejecuta una aplicación de gestión de red)
- **NR**: New Radio (Nueva Radio)

- **NGMN**: Next Generation Mobile Networks (Redes Móviles de Próxima Generación)
- **ns-3**: Network Simulator 3
- **OPEX**: Operational Expenditure (Gastos Operativos)
- **OSI**: Open Systems Interconnection (Interconexión de Sistemas Abiertos)
- **PCAP**: Packet Capture (Captura de Paquetes)
- **PDCP**: Packet Data Convergence Protocol (Protocolo de Convergencia de Datos de Paquetes)
- **PGW**: Packet Data Network Gateway (Puerta de Enlace de Red de Paquetes de Datos)
- **PHY**: Physical Layer (Capa Física)
- **PR**: Precision (Precisión)
- **QoE**: Quality of Experience (Calidad de Experiencia)
- **QoS**: Quality of Service (Calidad de Servicio)
- **RAT**: Radio Access Technology (Tecnología de Acceso de Radio)
- **RE**: Recall
- **RLC**: Radio *Link* Control (Control de Enlace de Radio)
- **RU**: Radio Unit (Unidad de Radio)
- **SD-EON**: Software-Defined Elastic Optical Network (Red Óptica Elástica Definida por Software)
- **SINR**: Signal-to-Interference-plus-Noise Ratio (Relación Señal-Interferencia-más-Ruido)
- **SNMP**: Simple Network Management Protocol (Protocolo Simple de Administración de Red)
- **SNR**: Signal-to-Noise Ratio (Relación Señal-Ruido)
- **TDD**: Time Division Duplexing (Duplexación por División de Tiempo)
- **TCP**: Transmission Control Protocol (Protocolo de Control de Transmisión)
- **TN**: True Negative (Verdadero Negativo)
- **TP**: True Positive (Verdadero Positivo)
- **UDP**: User Datagram Protocol (Protocolo de Datagramas de Usuario)

- **UE**: User Equipment (Equipo de Usuario)
- **UL**: Uplink (Enlace Ascendente)
- **UML**: Unified Modeling Language (Lenguaje Unificado de Modelado)
- **URLLC**: Ultra-Reliable Low-Latency Communications (Comunicaciones Ultra Confiables y de Baja Latencia)
- **VoIP**: Voice over IP (Voz sobre IP)

Capítulo 1

Introducción

En este capítulo se presenta una visión general del proyecto, abordando su contexto, motivación y objetivos. Se inicia con una exploración de los factores que impulsan el desarrollo de redes 5G y la necesidad de herramientas avanzadas de monitorización. Posteriormente, se detalla el contexto específico del proyecto dentro de una iniciativa de investigación más amplia. Se establecen los objetivos generales y específicos del trabajo, y se concluye con una descripción de la estructura del documento, proporcionando al lector una hoja de ruta clara para los capítulos subsiguientes.

1.1. Motivación

En los últimos años, la industria de las telecomunicaciones ha experimentado una transformación radical, impulsada principalmente por el crecimiento exponencial en la demanda de altos valores de ancho de banda y velocidad de transmisión de datos. Este aumento se debe a la proliferación de aplicaciones como la transmisión de video de alta definición, juegos en línea y servicios de comunicación en tiempo real. Además, el despliegue de infraestructura de fibra óptica y otras tecnologías de ancho de banda superior ha facilitado un acceso más rápido y confiable a Internet, lo que a su vez ha incentivado el uso de estas aplicaciones de alto consumo.

En este contexto, la transición hacia tecnologías de quinta generación (5G) se presenta como una solución clave para satisfacer las crecientes necesidades de conectividad. Las redes 5G prometen un incremento sustancial en el ancho de banda de red, latencias más bajas y mayor confiabilidad, lo que las hace ideales para soportar el creciente tráfico de datos y la diversidad de aplicaciones emergentes (Crawshaw, 2021).

La urgencia por desarrollar y desplegar redes 5G en todo el mundo responde a la necesidad de una infraestructura de telecomunicaciones que pueda soportar la creciente demanda de datos y la proliferación de dispositivos conectados. Con el advenimiento del Internet de las Cosas (IoT), vehículos autónomos, realidad

aumentada y otras tecnologías emergentes, las redes actuales se ven desafiadas a mantener el ritmo en términos de capacidad, velocidad de trasmisión y latencia. La implementación de 5G promete revolucionar sectores como la salud, la educación, la industria y el transporte, ofreciendo oportunidades sin precedentes para el desarrollo económico y social.

Sin embargo, la transición hacia 5G presenta diversos desafíos. La complejidad inherente de estas redes, en conjunto con la necesidad de garantizar una cobertura amplia y confiable, demanda herramientas avanzadas de simulación y monitorización que permitan evaluar su costo de implementación y realizar el despliegue de una manera estratégica y planificada. Aún hay numerosos obstáculos por superar y se requiere una considerable cantidad de investigación en este campo. Para los investigadores, disponer de simuladores es esencial para llevar a cabo pruebas de concepto y evaluaciones detalladas de distintos algoritmos, configuraciones y estrategias de gestión de tráfico.

La implementación de una red 5G no solo implica una inversión considerable en infraestructura, sino también una cuidadosa planificación y simulaciones previas. Estas etapas son cruciales para optimizar el uso de recursos y anticipar posibles problemas. En este contexto, los simuladores de redes desempeñan un papel fundamental, permitiendo a los ingenieros diseñar y evaluar diversos escenarios antes de proceder con el despliegue real. Así, se asegura una transición más eficiente y efectiva hacia las nuevas tecnologías de telecomunicaciones.

Este proyecto se enmarca dentro de un esfuerzo colaborativo más amplio, que involucra a la academia y entidades gubernamentales. A través de la innovación y la investigación, el objetivo es contribuir a la construcción de una infraestructura de telecomunicaciones robusta y preparada para el futuro, que pueda soportar las demandas de una sociedad cada vez más conectada y digitalizada. El objetivo principal del proyecto CSIC, proyecto marco de este trabajo, se centra en la implementación de una red "inteligente" que sea autónoma, es decir, que será capaz de gestionar los recursos de red por su cuenta y monitorizarse a sí misma para detectar posibles fallas, localizarlas y actuar en consecuencia de forma proactiva. Esto supone una reducción significativa en los costos operacionales de red (OPEX).

En el presente proyecto, se construye una herramienta que realiza monitorización y detección de fallas en redes óptico-móviles, esta tarea será realizada eficientemente y con una estrategia que se adapte al degrado de la red momento a momento. Se espera que este proyecto pueda contribuir al proyecto CSIC en la construcción de una red autogestionada.

De forma resumida, el sistema de monitorización construido realiza el seguimiento en el tiempo de la evolución de ciertas métricas de calidad de servicio, y si en algún momento alguna de las métricas registradas supera un umbral definido, se desencadena una rutina para localizar el punto de falla en la red. Este modo de operar, permite que no sea necesario monitorizar la red constantemente en busca de puntos de fallas, ya que resultaría en una mayor demanda de recursos computacionales y eventualmente de recursos de red, haciendo que, por ejemplo, las simulaciones demoren más tiempo en ejecutarse.

Es importante destacar que el estudio de métricas se realiza a nivel de capa

de aplicación, con la intención de reportar problemas ocurridos en la capa física, como la caída de un enlace. Sin embargo, es posible que los errores reportados se deban a capas superiores, donde puede ocurrir congestión por retransmisión de paquetes, sobrecarga de red debido al uso excesivo, problemas de enrutamiento, entre otros. Es responsabilidad del usuario identificar el fenómeno causante del mal desempeño de la red, la herramienta creada se limita a detectar el mal funcionamiento de la red e indicar el enlace con peor rendimiento.

El sistema de monitorización fue implementado en el simulador ns-3, utilizando el módulo de extensión para redes 5G llamado nr (5G-LENA) y desarrollando clases adicionales para cumplir con el objetivo planteado. La herramienta es escalable en escenarios de diversos perfiles de tráfico, diferentes tamaños de redes, diferentes umbrales en las métricas definidas y diferentes propiedades de la simulación definidas en cualquiera de las capas de red. Dentro de las herramientas creadas, se incluye la extracción de las métricas a un archivo externo y la posibilidad de visualizarlas para ver su evolución a lo largo del tiempo de simulación en un gráfico. Las herramientas antes mencionadas serán de vital importancia para la red inteligente del proyecto en el cual se integra, permitiéndole monitorizar el estado de red y localizar las fallas para actuar en consecuencia, y de ser necesario, ajustar y reasignar los recursos de red pertinentes. De esta forma, la red estaría en un ciclo constante de monitorización, asignación de recursos, reaccionando de forma inmediata a los cambios de demanda en la red y actuando en consecuencia.

1.2. Contexto

La monitorización de red es de gran utilidad para su gestión, donde se tienen tareas como el aprovisionamiento de recursos, enrutamiento, detección de anomalías, detección de fallas y detección de la degradación de la QoE (calidad de experiencia de los usuarios).

Este proyecto se encuentra dentro de un proyecto más grande titulado "Convergencia entre Redes 5G/6G y Redes Ópticas: Un Enfoque Holístico" (Proyecto CSIC I+D) que propone: estudiar arquitecturas de red definidas por software (SD-EON) para control holístico de redes 5/6G y redes ópticas elásticas, proponer nuevos modelos de integración, desarrollar algoritmos de aprendizaje automático para la asignación de *slices* en entornos de multi-dominio tecnológico, proponer arquitectura de monitorización aliada a los algoritmos propuestos, evaluar herramientas de simulación para evaluar los algoritmos y como resultado tener una plataforma de simulación disponible para evaluar los algoritmos de asignación de recursos en un enfoque multi-tecnológico.

Dentro de los interesados en el proyecto se encuentra ANTEL, para evaular la implantación de la red 5G. También se cuenta con el apoyo de la Dirección Nacional de Telecomunicaciones (DINATEL). Como resultado de este proyecto se beneficiará al sector privado, porque se dejarán las herramientas y algoritmos disponibles al público, como al sector académico, por la experiencia y capacitación por parte del equipo de investigación de la Facultad de Ingeniería de la

Universidad de la República en las redes 5G.

1.3. Objetivos del proyecto

El objetivo central de este proyecto consiste en desarrollar una arquitectura de monitorización diseñada para extraer, en tiempo de simulación, un conjunto de métricas que logren cuantificar representativamente el estado general de la red. Adicionalmente, será capaz de identificar y notificar al usuario sobre cualquier enlace defectuoso, permitiendo así al administrador de red realizar las correcciones necesarias, es decir, el proyecto presenta una herramienta de infraestructura.

Para facilitar el uso y la adaptabilidad del sistema, la arquitectura permitirá la modificación de las propiedades específicas de red a través de parámetros. Este enfoque tiene como objetivo minimizar la necesidad de programación por parte del usuario, permitiéndole concentrarse en la configuración y en su tarea específica de simulación. De esta manera, se reduce el esfuerzo en programación y se optimiza el proceso de simulación.

Con el fin de contribuir a la comunidad científica, se dejará disponible el proyecto en un repositorio GitLab (Leandro Alfonso y Nicolás Rivoir, 2024). Esto no solo permitirá que otros usuarios accedan y utilicen la herramienta, sino que también facilitará la colaboración y el desarrollo continuo de recursos para la simulación de redes 5G. Además, se tiene previsto publicar un informe detallado en una revista científica para dar a conocer el proyecto a la comunidad académica y profesional, promoviendo así un mayor interés y adopción del trabajo realizado.

1.4. Descripción del documento

Se presentará la estructura general del documento. El documento comienza con una sección introductoria que contextualiza el dominio y las tecnologías implicadas. Posteriormente, se analizan en profundidad las herramientas evaluadas y utilizadas, destacando su contribución real o potencial a la resolución del problema planteado. A continuación, se presenta una descripción detallada de la herramienta desarrollada, incluyendo sus aspectos técnicos. Para ilustrar su funcionalidad, se expone un caso práctico donde la herramienta se emplea para localizar una falla intencionalmente provocada en una simulación. Finalmente, se ofrecen las conclusiones del estudio y se proponen futuras mejoras y ampliaciones para la herramienta.

Estas son las secciones del documento:

- **Introducción**: Presenta el contexto del proyecto, incluyendo la motivación detrás del desarrollo de una arquitectura de monitorización para redes 5G. Se establece el marco teórico necesario para comprender el alcance y la importancia del trabajo realizado.

- **Revisión de antecedentes**: Ofrece un análisis de los simuladores existentes y herramientas de monitorización similares en el campo de las redes 5G. Esta sección proporciona una base comparativa para entender la contribución única de la herramienta desarrollada.
- **ns-3 y 5G-LENA**: Profundiza en el framework ns-3 y la extensión 5G-LENA, que fueron las plataformas principales utilizadas en el proyecto. Se detalla la estructura, los aspectos clave de su arquitectura y las características principales que fueron relevantes para el desarrollo de la herramienta de monitorización.
- **Nuestros aportes**: Esta sección presenta en detalle el proyecto desarrollado, incluyendo la arquitectura de la herramienta de monitorización, sus componentes principales y las decisiones técnicas tomadas durante su implementación. Se explican las innovaciones y mejoras introducidas sobre las plataformas base.
- **Experimentación**: Describe los experimentos diseñados y ejecutados para validar el funcionamiento correcto de la herramienta. Se presentan casos de uso representativos que demuestran la eficacia y versatilidad de la arquitectura de monitorización en diferentes escenarios de red 5G.
- **Ingeniería de software**: Describe la planificación y gestión del proyecto, desde un punto de vista de Ingeniería de Software. Se detalla la gestión de esfuerzo, las herramientas utilizadas y el cronograma del proyecto.
- **Conclusiones y Trabajo Futuro**: Resume los logros principales del proyecto, reflexiona sobre los desafíos enfrentados y las lecciones aprendidas. Además, propone direcciones para futuras investigaciones y mejoras potenciales de la herramienta, considerando las tendencias emergentes en redes 5G y más allá.

Capítulo 2

Dominios y Revisión de Antecedentes

En este capítulo se exploran los dominios tecnológicos esenciales que sustentan el proyecto, enfocándose principalmente en las redes 5G y las redes ópticas. Además, se detallan las estrategias de monitorización en redes de telecomunicaciones y se presenta una revisión de herramientas de simulación, culminando en la elección del entorno de trabajo para el desarrollo del proyecto.

2.1. Dominios

En esta sección se abordan las tecnologías que sustentan el proyecto en cuestión: las redes 5G y las redes ópticas. Además, se analiza la situación actual de estas tecnologías en Uruguay y su impacto en el desarrollo tecnológico y socio-económico del país.

2.1.1. Redes 5G: Un Avance Tecnológico en la Comunicación Móvil

Las redes móviles de quinta generación (5G) son la última evolución de las redes móviles inalámbricas. Una red móvil inalámbrica es un sistema de telecomunicaciones que permite a los usuarios conectarse y comunicarse a través de dispositivos móviles sin necesidad de cables. Estas redes han evolucionado a lo largo de diferentes generaciones (desde 1G a 5G) para ofrecer cada vez mejores capacidades de comunicación, mayores tasas de transmisión de datos y nuevos servicios.

El desarrollo y la estandarización de las redes 5G son coordinados por el *3rd Generation Partnership Project* (3GPP), una colaboración entre organizaciones de estándares de telecomunicaciones a nivel mundial. El 3GPP trabaja en la definición de protocolos y especificaciones técnicas necesarias para asegurar la interoperabilidad y la compatibilidad global de las redes 5G (3GPP, 2022).

Esta tecnología constituye una revolución en las telecomunicaciones móviles, ya que incrementa significativamente la velocidad de transmisión de datos, alcanzando hasta 10 Gb/s y reduciendo la latencia a menos de un milisegundo. Los avances tecnológicos que traen estas redes benefician el desarrollo de nuevas aplicaciones y servicios como la realidad aumentada, vehículos autónomos y el IoT (Shafi y cols., 2017). Además, las redes 5G introducen nuevas tecnologías como el MIMO masivo (*Multiple Input Multiple Output*) y la formación de haces (*beamforming*), que mejoran aún más la eficiencia y el rendimiento de red (Rappaport y cols., 2017). El MIMO masivo permite el uso de múltiples antenas en los dispositivos y las estaciones base para aumentar la capacidad de transmisión de datos, mientras que la formación de haces dirige de manera precisa las señales de radio hacia los dispositivos, mejorando la calidad de la conexión y reduciendo la interferencia.

Un componente esencial de las redes 5G es el *New Radio* (NR), que establece los estándares para la comunicación inalámbrica entre dispositivos y estaciones base. El NR se caracteriza por su alta flexibilidad, adaptándose a un gran número de bandas de frecuencia, desde bandas inferiores a 1 GHz hasta frecuencias milimétricas superiores a 24 GHz. Esta flexibilidad permite que las redes 5G ofrezcan una amplia cobertura y una masiva capacidad de transmisión de datos.

Las redes 5G no solo ofrecen mayores velocidades y menor latencia, sino que también introducen mejoras significativas en la capacidad de la red para manejar un mayor número de dispositivos conectados simultáneamente. Esta capacidad es crucial para soportar el crecimiento explosivo del Internet de las Cosas (IoT), donde se espera que miles de millones de dispositivos estén conectados a la red.

En resumen, las redes 5G representan un avance significativo en la tecnología de comunicación inalámbrica, proporcionando velocidades de transmisión de datos sin precedentes, baja latencia y una capacidad de conexión masiva. Estas redes son una evolución de las tecnologías móviles anteriores, como LTE, pero superan sus limitaciones al operar en bandas de frecuencia más altas y ofrecer un diseño ultra-eficiente en términos de energía y compatibilidad futura. Se espera que las redes 5G transformen diversos sectores de la industria y la sociedad, habilitando una nueva era de aplicaciones y servicios digitales avanzados (Dahlman, Parkvall, y Sköld, 2018).

2.1.2. Características del Tráfico en Redes 5G

Los tipos de tráfico en redes 5G se clasifican según sus diferentes características y niveles de importancia. Esta clasificación es esencial para comprender las diversas aplicaciones y casos de uso que las redes 5G pueden soportar, así como las exigencias técnicas específicas que cada tipo de tráfico impone sobre la infraestructura de red. En el contexto de las redes 5G, se identifican tres principales categorías de tráfico: *Banda ancha móvil mejorada* (*eMBB*), *Comunicaciones masivas de tipo máquina* (*mMTC*), y *Comunicaciones ultra confiables y de baja latencia* (*URLLC*) (Figura 2.1).

- **Banda ancha móvil mejorada (*eMBB*):** Esta categoría se centra en la provisión de altas velocidades de datos y capacidad de tráfico en áreas densamente pobladas. Es crucial para aplicaciones como la transmisión de video de alta definición y la realidad virtual.
- **Comunicaciones masivas de tipo máquina (*mMTC*):** Este tipo de tráfico está orientado a soportar una gran cantidad de dispositivos conectados simultáneamente, típicos del Internet de las Cosas (IoT). Ejemplos incluyen sensores industriales y dispositivos domésticos inteligentes.
- **Comunicaciones ultra confiables y de baja latencia (*URLLC*):** Este tráfico es fundamental para aplicaciones que requieren una latencia extremadamente baja y alta confiabilidad, tales como vehículos autónomos, telemedicina y sistemas de control industrial.

Cada una de estas categorías presenta diferentes requerimientos en términos de velocidad transmisión de datos, latencia, densidad de conexiones y eficiencia energética, lo que resalta la versatilidad y capacidad de las redes 5G para adaptarse a múltiples escenarios de uso. La correcta gestión y priorización de estos tipos de tráfico es fundamental para asegurar el rendimiento óptimo de red y la satisfacción de las diversas necesidades de los usuarios y aplicaciones (Figura 2.1). Para cumplir con los requerimientos tan exigentes, es indispensable contar con un conocimiento detallado de la performance de la red, lo que permite anticiparse a fallas o actuar en consecuencia, es por este motivo que la monitorización en las redes 5G es de vital importancia.

2.1.3. Redes 5G en Uruguay

En Uruguay, el despliegue de redes 5G comenzó con un acontecimiento pionero en América Latina cuando la Administración Nacional de Telecomunicaciones (ANTEL) lanzó la primera red comercial 5G en abril de 2019, inicialmente en áreas seleccionadas de Maldonado y Nueva Palmira. La evolución de la tecnología 5G en Uruguay continuó con un creciente interés por parte del sector industrial y gubernamental. En julio de 2022, la Unidad Reguladora de Servicios de Comunicaciones autorizó a los principales operadores del país a realizar pruebas técnicas con esta tecnología. Aunque estas pruebas fueron temporales, establecieron una base sólida para futuras implementaciones comerciales.

A finales de 2022, se programó una licitación para asignar espectro en la banda de 3.5 GHz, crucial para el despliegue de redes 5G. En 2023, Uruguay comenzó con los despliegues comerciales de redes 5G en todos los departamentos del país por parte de los tres principales operadores de redes móviles: ANTEL, Movistar y Claro. Este hito significativo en la expansión de la tecnología 5G en Uruguay fue posible gracias a la licitación de espectro en la banda de 3.5 GHz que se realizó en mayo de 2023. ANTEL, en particular, planeó una inversión de 43 millones de dólares durante 2023 para el despliegue de 5G (International Trade Administration, 2024; GlobeNewswire, 2023). Adicionalmente, se ha discutido la posibilidad de crear una infraestructura de red 5G unificada que sería

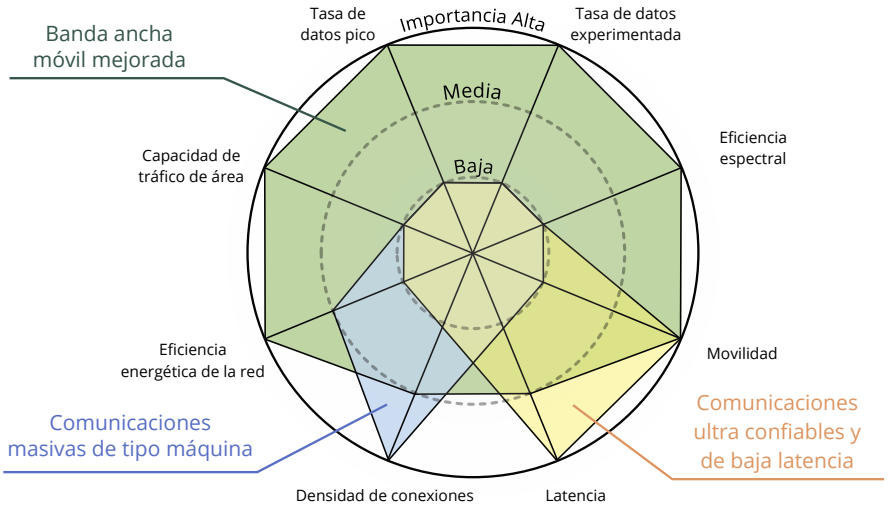

Figura 2.1: Características del tráfico en redes 5G, imagen adaptada de (Mohr, 2017).

compartida por todos los operadores, promoviendo una implementación más eficiente y sostenible. Esta estrategia refleja una tendencia global hacia la cooperación en el desarrollo de infraestructuras de telecomunicaciones, equilibrando la competencia con la sostenibilidad.

El futuro de Uruguay en el contexto de las redes 5G se proyecta prometedor, con un crecimiento anticipado en el uso de esta tecnología debido a la creciente demanda y los desafíos que impone la modernidad. A medida que la sociedad se vuelve cada vez más digitalizada y dependiente de la conectividad, la implementación de redes 5G se vuelve crucial para satisfacer las necesidades de alta velocidad, baja latencia y mayor capacidad de conexión. Los beneficios que las redes 5G traen son significativos. En este escenario, Uruguay se posiciona como un país líder en la adopción de tecnologías avanzadas, lo que contribuye a su desarrollo sostenible y a su competitividad en el ámbito global.

2.1.4. Redes Ópticas

Las redes ópticas obtienen su nombre por la fibra óptica utilizada para transmitir datos mediante señales de luz y son un componente fundamental en la infraestructura de telecomunicaciones moderna. Son capaces de transmitir grandes cantidades de datos a largas distancias con alta velocidad y mínima pérdida de señal, lo que las hace ideales para soportar las demandas crecientes de servicios digitales y de conectividad (López y Velasco, 2016)

Una de las principales características de las redes ópticas es su capacidad para manejar grandes anchos de banda, mucho mayores que las tecnologías de

transmisión tradicionales. La fibra óptica puede soportar velocidades de hasta terabits por segundo, facilitando el rápido desarrollo de aplicaciones que requieren grandes transferencias de datos, como la transmisión de video de alta definición, servicios en la nube y centros de datos.

Además de su alta capacidad, las redes ópticas ofrecen una latencia muy baja, que es crucial para aplicaciones que requieren respuestas en tiempo real, como los juegos en línea, la telemedicina y los sistemas de control industrial. La baja latencia y la alta velocidad de las redes ópticas complementan la infraestructura de las redes 5G, especialmente en áreas donde la densidad de datos y la demanda de servicios son excepcionalmente altas (Velasco y Ruiz, 2017).

2.1.5. Complementariedad entre Redes 5G y Redes Ópticas

Las redes ópticas y las redes 5G son tecnologías complementarias que, al combinarse, crean una infraestructura de telecomunicaciones robusta y eficiente. Las redes ópticas actúan como red troncal (*backbone*) para las redes 5G, proporcionando enlaces de alta capacidad y baja latencia que transportan datos desde las estaciones base 5G hasta los centros de datos y viceversa (Velasco y cols., 2017).

Esta sinergia es crucial, ya que aunque las redes 5G ofrecen velocidades extremadamente altas y pueden gestionar múltiples conexiones simultáneas, su alcance efectivo es más limitado debido al uso de frecuencias más altas en comparación con sus predecesoras (como las redes 4G), que tienen menor capacidad de penetración y alcance. Por esta razón, las redes ópticas son esenciales para interconectar múltiples nodos 5G, asegurando una cobertura continua y amplia, especialmente en áreas urbanas densamente pobladas.

El diagrama de la Figura 2.3 ilustra esta complementariedad. El Centro de Datos, se encuentra en el núcleo de la infraestructura, gestionando el procesamiento y almacenamiento de datos. Este centro está conectado mediante enlaces de fibra óptica a *Switches* y *Routers*, que distribuyen el tráfico a lo largo de la red. Las Estaciones Base 5G, están distribuidas estratégicamente y conectadas al Nodo de Agregación, que actúa como punto de concentración del tráfico antes de enviarlo al Centro de Datos. Los dispositivos de usuario final se conectan de forma inalámbrica a las Estaciones Base 5G, permitiendo una comunicación en tiempo real y de alta velocidad.

Es importante señalar que, aunque no fue el escenario abordado en nuestro proyecto, existe otra configuración común en las redes 5G que merece mención. En este escenario alternativo, ilustrado en la Figura 2.2, la estación base 5G se distribuye en tres componentes principales: RU (*Radio Unit*), DU (*Distributed Unit*) y CU (*Centralized Unit*). Las conexiones entre estos componentes suelen realizarse mediante redes ópticas, dando lugar a los conceptos de *fronthaul* (entre RU y DU), *midhaul* (entre DU y CU) y *backhaul* (entre CU y el núcleo de red). Esta arquitectura distribuida permite una mayor flexibilidad en el despliegue de red y optimiza el uso de recursos, aunque también introduce nuevos desafíos en términos de latencia y sincronización.

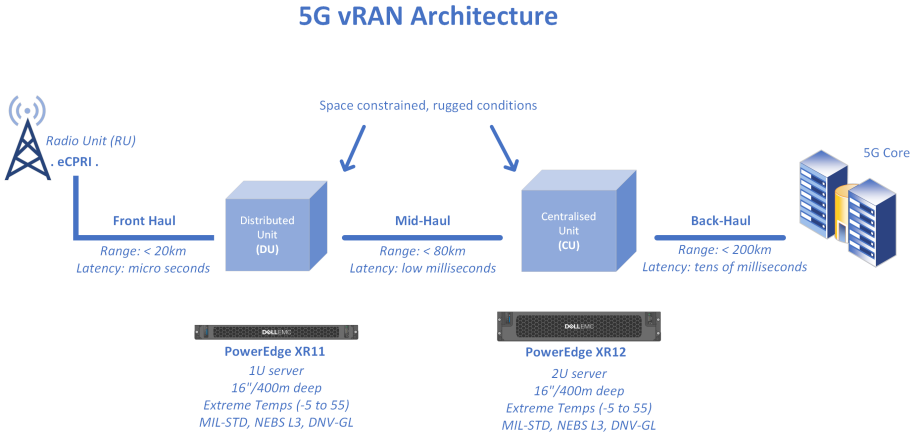

Figura 2.2: Arquitectura vRAN en redes 5G, mostrando los componentes principales: Unidad de Radio (RU), Unidad Distribuida (DU) y Unidad Centralizada (CU). Las conexiones entre estos componentes se realizan a través de enlaces ópticos, definidos como *fronthaul*, *midhaul* y *backhaul*, cada uno con diferentes rangos y latencias. Imagen extraída de (Hegarty, 2021).

Además, la implementación conjunta de estas redes facilita el desarrollo de aplicaciones avanzadas que requieren alta capacidad de datos y baja latencia. Por ejemplo, en el ámbito de los vehículos autónomos, las redes ópticas proporcionan la infraestructura necesaria para procesar grandes cantidades de datos provenientes de los sensores de los vehículos, mientras que las redes 5G permiten la comunicación en tiempo real entre los vehículos y los sistemas de gestión de tráfico (Barzegar y cols., 2023).

En resumen, las redes 5G potencian las aplicaciones móviles e IoT con sus capacidades de alta velocidad y baja latencia, mientras que las redes ópticas garantizan que estas aplicaciones funcionen de manera eficiente y sin interrupciones a través de una infraestructura de comunicaciones de alta capacidad y baja latencia. Esta combinación de tecnologías permite mejorar la cobertura y calidad del servicio, facilitando el desarrollo de aplicaciones avanzadas y servicios más eficientes en diversos sectores.

2.2. Estrategias de Monitorización en Redes

La monitorización efectiva en redes de telecomunicaciones es fundamental para asegurar la calidad del servicio, la detección de fallas y el mantenimiento eficiente de la infraestructura. Recientes avances en este campo han propuesto enfoques innovadores para optimizar la monitorización de redes. Por ejemplo, Mouchet et al. (Mouchet y cols., 2020) presentan heurísticas escalables para mejorar la latencia de red, abordando el desafío de minimizar conjuntamente

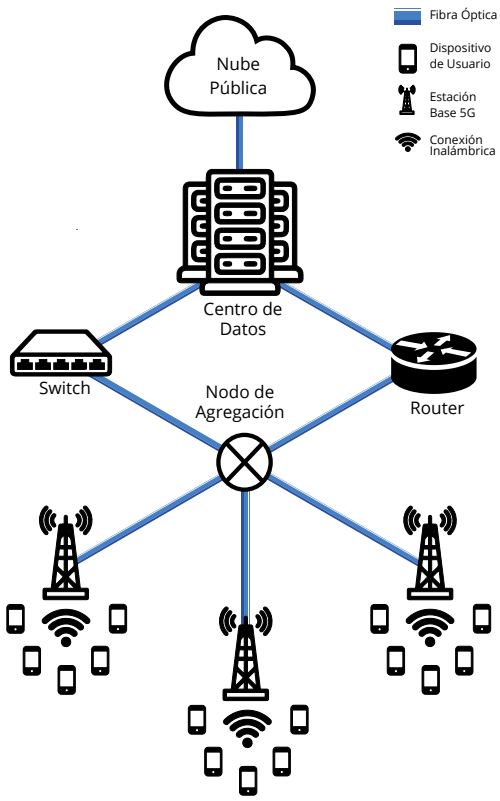

Figura 2.3: Complementariedad entre Red Óptica y Red 5G.

el costo de sondeo y el retraso en el enrutamiento. Su trabajo propone soluciones que son lo suficientemente rápidas para proporcionar resultados eficientes en tiempos prácticos, lo cual es crucial en el contexto de redes complejas y dinámicas como las redes 5G. Los métodos de monitorización más populares son la inyección de *probes* y la monitorización de tráfico existente (Tseng y cols., 2019). A continuación, se detallan ambos enfoques y se discute su integración con el protocolo Simple Network Management Protocol (SNMP), una herramienta esencial en la gestión de redes.

2.2.1. Inyección de *Probes*

La inyección de *probes* implica enviar paquetes de datos específicos, conocidos como *probes*, a través de la red con el propósito de medir y evaluar el rendimiento de la misma. Este método permite a los administradores de red obtener información valiosa sobre aspectos críticos como la latencia, la pérdida de paquetes y la disponibilidad del camino de red. Los *probes* son diseñados para minimizar su impacto en el tráfico regular, asegurando que la monitorización no

interfiera con el funcionamiento normal de la red.

Este método es particularmente útil en escenarios donde se requiere un diagnóstico detallado y en tiempo real del estado de red. Al controlar activamente el flujo de estos paquetes de prueba, los técnicos pueden identificar rápidamente problemas de rendimiento y tomar medidas correctivas de manera proactiva. Ver ejemplos de propuestas que se basan en *probes* y su análisis de trabajos relacionados en el artículo "*Network Performance Analysis Using Packets Probe for Passive Monitoring*" (Alkenani y Nassar, 2022).

2.2.2. Monitorización de Tráfico Existente

A diferencia de la inyección de *probes*, la monitorización de tráfico existente se basa en analizar los datos que ya están siendo transmitidos a través de la red. Este enfoque aprovecha las estadísticas y los metadatos generados por el tráfico regular para evaluar el rendimiento de red. Al observar el flujo natural de la comunicación, los administradores pueden detectar anomalías, medir la utilización de red y entender mejor las dinámicas del tráfico sin necesidad de introducir tráfico adicional (Velasco y cols., 2019).

Este método es menos intrusivo y puede proporcionar una visión continua y pasiva del estado de red, lo que lo hace ideal para la monitorización y planificación de la capacidad a largo plazo.

2.2.3. Simple Network Management Protocol (SNMP)

El Simple Network Management Protocol (SNMP) es un protocolo utilizado para la gestión de transferencia de información en redes y juega un papel crucial. Proporciona un marco estandarizado para la recopilación y organización de información sobre dispositivos en red. SNMP opera a través de la gestión de información estructurada en las MIBs (Management Information Bases) y permite la interacción entre dispositivos gestionados y **sistemas de gestión de red** (NMS, por sus siglas en inglés *Network Management Station*) mediante operaciones básicas como GET, SET y TRAP (Mauro y Schmidt, 2005; Feit, 1995).

SNMP se puede emplear efectivamente con ambos métodos de monitorización mencionados anteriormente. En el contexto de la inyección de *probes*, SNMP permite recolectar y analizar los resultados de las pruebas de rendimiento. Esto se logra mediante el uso de operaciones SNMP para consultar las MIBs y obtener datos específicos de rendimiento, facilitando la interpretación y la respuesta a los problemas detectados (Mauro y Schmidt, 2005).

Por otro lado, en la monitorización de tráfico existente, SNMP facilita la recopilación de datos operativos y estadísticas del tráfico de manera regular. Los agentes SNMP instalados en dispositivos de red recopilan y envían información al NMS, proporcionando una visión integrada del desempeño de red. Esto incluye información sobre el uso de ancho de banda, errores de transmisión y estado de los dispositivos, permitiendo a los administradores de red tomar decisiones

informadas para optimizar el rendimiento de red (Mauro y Schmidt, 2005; Feit, 1995).

La combinación de SNMP con estos métodos de monitorización no solo optimiza la gestión de red, sino que también proporciona una plataforma robusta para la detección proactiva de problemas y la mejora continua de la infraestructura de telecomunicaciones. Al implementar estas técnicas, las organizaciones pueden asegurar altos estándares de calidad y eficiencia en sus operaciones de red, adaptándose mejor a las demandas tecnológicas actuales y futuras (Stallings, 1993).

2.3. Herramientas de Simulación y Elección del Entorno de Trabajo

Al momento de realizar esta revisión de antecedentes, no se encontró ninguna herramienta que permita simular la topología de una red 5G, monitorizar la simulación para sacar métricas, detectar cuándo la red está fallando por medio de la monitorización de las métricas y localizar en qué enlace se produce el error, tal como la herramienta que se presenta en este informe. Sin embargo, se encontraron plataformas que permiten simular redes 5G y, a lo sumo, extraer métricas estadísticas del comportamiento de red. Este análisis también servirá de apoyo para la justificación del porqué se eligió ns-3 (ns-3 Consortium, 2024) con 5G-LENA (CTTC OpenSim Research Unit, 2024) para construir la herramienta: qué ofrece, ventajas y desventajas.

El uso de ns-3 y su módulo 5G-LENA fue una sugerencia proporcionada por los tutores; igualmente, como se explicará más adelante, durante el proyecto se intercambió con los investigadores del CSIC las ventajas y desventajas de distintas herramientas de simulación. Es por ello que a continuación se listan las herramientas de simulación que permiten simular redes 5G más populares que se consideraron de mayor relevancia y sus principales características.

2.3.1. ns-3

Ns-3 (ns-3 Consortium, 2024) es un simulador de redes de eventos discretos de código abierto, bajo la licencia GNU GPLv2. A diferencia de otros simuladores, ns-3 ofrece tecnologías multi-RAT (Multiple Radio Access Technologies) y multi-band (puede operar en múltiples bandas de frecuencia). Es una herramienta muy completa con alto grado de abstracción, que permite la implementación de una gran variedad de protocolos en todas las capas de red; está pensada para admitir extensiones e implementaciones de nuevas tecnologías que puedan surgir en el futuro. Por otra parte, presenta un gran desafío a la hora de aprender a usarla, debido a que tiene una curva de aprendizaje pronunciada. Sin embargo, cuenta con una gran comunidad, que responde preguntas en los foros y reporta problemas en el software, que le da un valor agregado a la herramienta. También posee extensa documentación y tutoriales simples de seguir, explicados al detalle y con ejemplos didácticos para aprender haciendo. Los ejemplos son

progresivamente más complejos y van presentando diversos componentes de la herramienta poco a poco. Estos ejemplos son de gran utilidad para aprender a usar la herramienta.

Otra de las carencias de ns-3 son las herramientas de visualización. Posee algunas clases que permiten escribir información en archivos externos, para luego graficarlos post simulación con *GNUPlot*. Sin embargo, son pocos los tipos de gráficos que se pueden realizar con la herramienta en sí, requiriendo que el usuario agregue código adicional si desea hacer visualizaciones más complejas. También ofrece herramientas de *logging* a nivel de simulación y de *debug* para que el programador pueda registrar mensajes, errores y advertencias.

2.3.2. 5G-LENA

5G-LENA es una biblioteca de código abierto desarrollada por el equipo OpenSim del CTTC (Centro Tecnológico de Telecomunicaciones de Cataluña) y licenciada bajo GNU GPLv2 (Patriciello, Lagen, Bojovic, y Giupponi, 2019; Koutlia, Bojovic, Ali, y Lagen, 2022). Esta extensión de ns-3 permite simular redes 5G de manera completa, cubriendo desde la capa de aplicación hasta la capa física. 5G-LENA soporta una amplia gama de funcionalidades propias de las redes 5G, incluyendo la gestión de movilidad, comunicaciones ultra confiables y de baja latencia (*URLLC* por sus siglas en inglés *Ultra-Reliable and Low-Latency Communication*), y la implementación de técnicas avanzadas de acceso radioeléctrico como MIMO y *beamforming*.

5G-LENA es altamente configurable y modular, lo que facilita su adaptación a diversos escenarios de investigación y desarrollo. Permite realizar simulaciones *end-to-end*, proporcionando una plataforma robusta para el estudio de nuevas tecnologías y protocolos dentro del ecosistema 5G. Esta biblioteca es ideal para investigadores que necesitan un entorno flexible y extensible para probar diferentes configuraciones de red y evaluar el rendimiento en distintos escenarios.

La comunidad activa que respalda 5G-LENA, junto con la amplia documentación disponible, facilita el aprendizaje y la implementación de simulaciones complejas. Además, su integración con ns-3 permite a los usuarios aprovechar una gran cantidad de ejemplos y recursos existentes, acelerando el proceso de desarrollo y experimentación.

2.3.3. OMNeT++

OMNeT++ (Varga, 2010) es un simulador de redes de eventos discretos de código abierto, conocido por su interfaz gráfica intuitiva que permite diseñar y visualizar arquitecturas de red fácilmente. Esta característica es especialmente útil para ver el tráfico de red en tiempo real y ajustar parámetros de simulación de manera interactiva.

Utilizado ampliamente en la investigación académica, el desarrollo industrial y la educación, OMNeT++ es altamente modular y permite la integración de componentes personalizados para simular una variedad de redes, desde LAN hasta redes móviles y ad hoc. La comunidad activa y la extensa documentación

disponible facilitan el aprendizaje y la implementación de simulaciones complejas.

OMNeT++ también soporta la integración con bibliotecas y *frameworks* como INET (Mészáros, Varga, y Kirsche, 2019) y Veins (Sommer, German, y Dressler, 2011), ampliando sus capacidades. Ofrece herramientas de *logging* y *debugging* para registrar mensajes, errores y advertencias durante la simulación, lo que facilita la identificación y corrección de problemas. Además, OMNeT++ permite la visualización de resultados postsimulación con herramientas externas como *GNUPlot* y la generación de informes detallados para el análisis y presentación de resultados.

2.3.4. Simu5G

Simu5G (Nardini, Sabella, Stea, Thakkar, y Virdis, 2020) es un simulador de redes 5G de código abierto, licenciado bajo LGPL, que ha ganado popularidad debido a su robustez y flexibilidad. Fue construido sobre la plataforma OMNeT++, aprovechando su capacidad de proporcionar una interfaz gráfica intuitiva para el diseño y visualización de simulaciones. Simu5G es altamente configurable y extensible, lo que lo convierte en una herramienta ideal para investigadores y desarrolladores que deseen experimentar con nuevas tecnologías y protocolos de 5G.

Simu5G soporta una amplia gama de funcionalidades para redes 5G, incluyendo la simulación de arquitecturas de red avanzadas, la gestión de movilidad, el soporte para múltiples usuarios y dispositivos, y la implementación de técnicas de comunicación avanzada como MIMO y *beamforming*. Además, permite la simulación de escenarios complejos de tráfico y la evaluación de rendimiento de diversas configuraciones de red.

Una de las características destacadas de Simu5G es su capacidad para integrarse con otras herramientas y bibliotecas, lo que facilita la incorporación de modelos personalizados y la realización de estudios más detallados. La comunidad activa y la amplia documentación disponible son otros puntos fuertes que apoyan a los usuarios en el aprendizaje y uso efectivo de la herramienta.

2.3.5. Definición de la Herramienta de Simulación

En esta sección se presenta la evaluación y selección de la herramienta de simulación adecuada para implementar la herramienta de monitrización, enfocándose en la comparación entre ns-3 con 5G-LENA y OMNet++ con Simu5G.

Durante la etapa de capacitación, se realizaron seminarios para que los integrantes del proyecto del CSIC aprendieran sobre el dominio de las telecomunicaciones y redes 5G. Dentro del seminario, se realizaron presentaciones donde cada equipo estudió un simulador diferente. Los simuladores estudiados fueron ns-3 con 5G-LENA y OMNet++ con Simu5G, debido a que son los más utilizados y los que más se ajustaban a los requisitos que presentaban los problemas a resolver.

Luego de las presentaciones, se evaluaron las ventajas y desventajas de cada simulador, como se muestra en la tabla 2.1. Si bien la decisión de usar ns-3 con 5G-LENA ya estaba dada por los tutores, motivo por el cual fue estudiada para presentar al resto del equipo del CSIC en uno de los seminarios, igualmente se planteó un momento de reflexión para evaluar si OMNET++ con Simu5G sería una mejor herramienta para el problema a resolver utilizando la información aportada por los seminarios. Finalmente, se reafirmó junto a los tutores la decisión inicial de usar ns-3 con 5G-LENA debido a los motivos que se explicarán a continuación.

Los dos simuladores, OMNeT++ y ns-3, son de código abierto y están programados en C++. OMNeT++ se destaca por su interfaz gráfica intuitiva, que facilita la visualización y manipulación de las simulaciones. Esta característica hace que OMNeT++ sea especialmente accesible para usuarios que prefieren trabajar con una interfaz visual. Además, OMNeT++ ofrece una arquitectura modular y extensible, lo que permite una fácil integración de componentes personalizados y una mayor flexibilidad en el diseño de simulaciones.

Por otro lado, ns-3 es conocido por proporcionar un control más detallado sobre las simulaciones. Este simulador permite configurar cada nivel del stack de protocolos, desde el nivel físico hasta las aplicaciones, lo que ofrece una granularidad y precisión que puede ser crucial para ciertos tipos de investigaciones y proyectos. Aunque ns-3 tiene una curva de aprendizaje más pronunciada, su capacidad para personalizar y extender las clases existentes lo hace una herramienta poderosa para usuarios avanzados.

En cuanto a la comunidad y el soporte, ambos simuladores cuentan con un respaldo significativo tanto en el ámbito académico como en la industria. Sin embargo, ns-3 es particularmente apreciado en entornos de investigación debido a su flexibilidad y la precisión de sus modelos. La extensa comunidad de ns-3 también facilita la resolución de problemas y la colaboración en el desarrollo de nuevas herramientas y funcionalidades.

En conclusión, debido a la naturaleza del proyecto, enfocado en la monitorización y la necesidad de una gran flexibilidad y capacidad de personalización, se decidió trabajar con ns-3 y 5G-LENA como las herramientas de simulación más adecuadas. La elección de ns-3 se basa en su capacidad para configurar en todos los niveles del stack de simulación y su detallado control sobre los parámetros de la simulación, lo que resulta esencial para los objetivos del proyecto. Aunque OMNeT++ ofrece una interfaz gráfica amigable y una menor curva de aprendizaje, la complejidad y precisión adicionales que proporciona ns-3 son cruciales para este proyecto específico. Debido a la complejidad de esta herramienta, esta será explicada en más detalle en el Capítulo 3.

Simulador	ns-3 con 5G-LENA	OMNeT++ con Simu5G
Interfaz Gráfica	No, aunque tiene animaciones básicas.	Sí, con una GUI basada en QT y un IDE basado en Eclipse.
Comunidad y Documentación	Muy buena, con extensa documentación y numerosos tutoriales.	Buena, aunque no tan extensa como la de ns-3.
Completitud de la Herramienta	Muy Completa, soporta una amplia gama de protocolos y escenarios.	Más o menos completa, algunas áreas pueden requerir módulos adicionales.
Eficiencia Computacional	Buena, aunque puede necesitar optimización para simulaciones a gran escala.	Muy Buena, especialmente eficiente para simulaciones complejas.
Escalabilidad	Alta, soporta simulaciones muy escalables.	Alta, adecuada tanto para simulaciones pequeñas como grandes.
Soporte de Protocolos	Amplio, incluye Wi-Fi, WiMAX, LTE y 5G.	Amplio, especialmente con el marco INET.
Facilidad de Uso	Media, con una curva de aprendizaje pronunciada.	Alta, gracias a su GUI y diseño modular.
Integración con Hardware Real	Sí, se puede integrar con hardware real.	No, no soporta integración nativa con hardware real.
Precisión de Simulación	Alta, conocida por sus capacidades de simulación precisas.	Alta, resultados de simulación muy precisos.
Flexibilidad	Alta, ofrece gran flexibilidad en configuraciones de simulación.	Alta, diseño modular permite gran flexibilidad.
Velocidad de Simulación	Media, la velocidad puede variar según la complejidad.	Alta, conocida por tiempos de simulación más rápidos.
Requisitos de Hardware	Moderados, se puede ejecutar en hardware moderadamente potente.	Altos, requiere hardware más potente para simulaciones detalladas.
Soporte para Múltiples Lenguajes	C++ y enlaces opcionales para Python.	C++ y el lenguaje de descripción de red NED.
Herramientas de Análisis Integradas	PCAP para análisis con herramientas como Wireshark.	Extensas capacidades de visualización y análisis a través de su GUI y módulos integrados.
Actualizaciones y Mantenimiento	Actualizaciones regulares, mantenido por una comunidad global.	Actualizaciones periódicas, soporte y mantenimiento a través de licencias comerciales y comunidad.
Capacidad de Extensión	Alta, fácil de extender mediante módulos adicionales en C++.	Alta, modular y extensible mediante el uso de NED y C++.
Facilidad de Instalación y Configuración	Moderada, requiere compilación desde el código fuente.	Relativamente sencilla, instalación guiada con instaladores disponibles.
Soporte para Virtualización	Soporte para integración con sistemas virtualizados.	Puede integrarse con entornos virtualizados, pero no nativamente.
Interoperabilidad	Alta, permite integración con implementaciones de código real y otras herramientas de simulación.	Alta, especialmente a través del marco INET y otros módulos de extensión.
Consumo de Recursos de Hardware	Moderado, puede requerir optimización para simulaciones complejas o de alta carga de procesamiento.	Alto, especialmente cuando se usa la interfaz gráfica. Simu5G, al ser de más alto nivel, consume menos recursos en general, pero cuando se usa la interfaz gráfica consume más recursos que ns-3.

Tabla 2.1: Tabla comparativa de los simuladores ns-3 con 5G-LENA y OMNET++ con Simu5G.

Capítulo 3

Introducción a ns-3 y 5G-LENA

Este capítulo presenta una visión detallada de ns-3 y 5G-LENA, dos herramientas fundamentales utilizadas en la simulación de redes de comunicación avanzadas. Por un lado ns-3, un simulador de redes de eventos discretos de código abierto, ofrece una plataforma robusta y flexible para modelar una amplia gama de tecnologías de red. Por otra parte, 5G-LENA extiende las capacidades de ns-3 para abordar específicamente las complejidades de las redes 5G *New Radio* (NR). A lo largo de este capítulo se explorarán los componentes clave, la arquitectura y las funcionalidades de ambas herramientas, proporcionando una base sólida para comprender su papel crucial en la investigación y desarrollo de redes de próxima generación.

3.1. Ns-3

A continuación se profundizará en detalle sobre las diferentes características técnicas de ns-3. Toda la información presentada fue extraída del manual oficial de ns-3 (ns-3 Consortium, 2024) y manual oficial de 5G-LENA (CTTC OpenSim Research Unit, 2024).

Ns-3 es un simulador de redes de código abierto de eventos discretos. Está escrito en C++ y se ejecuta en la terminal de comandos. No tiene una interfaz gráfica para que el usuario interactúe.

Ns-3 no solo es un simulador que puede emular los protocolos del *stack* de Internet, sino que se abstrae para poder abarcar mucho más que eso, y así simular un gran abanico de *stack* de protocolos de red y tecnologías. Por ese motivo, se establece un diseño orientado a objetos con clases abstractas que luego pueden ser implementadas de distintas maneras para incluir nuevos protocolos o tecnologías de red. Esto permite una mayor flexibilidad, modularización del sistema y posibilidad de extensibilidad por parte del equipo de desarrolladores y la comunidad. A continuación se describen las principales clases que hacen

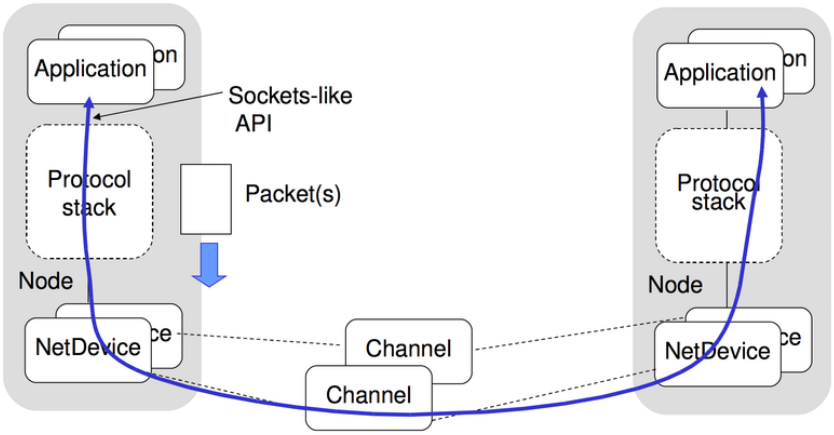

Figura 3.1: La imagen ilustra la estructura básica de una simulación de red en ns-3. Cada nodo en la red contiene una *stack* de protocolos y dispositivos de red (*NetDevice*). Los nodos se comunican entre sí a través de canales (*Channel*). Las aplicaciones en cada nodo interactúan con la pila de protocolos utilizando una *API* similar a *sockets*, y los paquetes se transfieren a través de los dispositivos de red y los canales entre nodos. Esta arquitectura modular y orientada a objetos permite la implementación de diferentes tecnologías y protocolos de red. Imagen extraída de "*Router Modelling and Packet Level Cache Implementation for Content Aware TCP (CATCP)*" (Budigere, 2016).

posible realizar una simulación de una red en ns-3 junto a como interactúan entre sí (ver Figura 3.1).

3.1.1. Nodo

En ns-3, un nodo es un dispositivo que se conecta a la red. En el *stack* de Internet se conoce como host y es la abstracción de un dispositivo de cómputo básico. Estos nodos son implementados por la clase de C++ *Node*.

3.1.2. Aplicación

Cada dispositivo de cómputo tiene un software que gestiona el uso de los recursos de cómputo como la memoria, el procesador, el acceso a disco o el uso de la tarjeta de red. Las aplicaciones se encargan de solicitar acceso a estos recursos por medio de este software para realizar sus tareas. En ns-3 no existe el concepto de Sistema Operativo, sin embargo, sí existe el concepto de Aplicación, que pretende simular las aplicaciones de usuario del mundo real. Estas aplicaciones crean actividades o rutinas que luego generan tráfico de red. Las aplicaciones son implementadas por la clase de C++ *Application* y pueden crearse nuevas

aplicaciones implementando la interfaz definida.

3.1.3. Canal

En ns-3 la información viaja por los canales, conectando los nodos de la red. Esta abstracción admite desde una conexión entre nodos por medio de un simple cable, hasta un switch Ethernet o una conexión Wifi inalámbrica con una representación tridimensional de los objetos que obstruyen la comunicación. Es la abstracción más básica de subred. Los canales son representados por la clase de C++ *Channel*.

3.1.4. Dispositivo de Red

En terminologías de Internet, esta abstracción se refiere a la tarjeta de red. Esta abstracción abarca tanto el software como el hardware que deben ser instalados en un nodo para que pueda comunicarse con otros nodos por medio de un canal. Es posible que un nodo se conecte a más de un canal haciendo uso de múltiples *Dispositivos de Red*. Los *Dispositivos de Red* se implementan con la clase de C++ *NetDevice*.

3.1.5. *Topology Helpers*

En ns-3, a medida que las topologías de red se hacen más grandes y complejas, el código se torna verboso, teniendo muchas líneas de código muy similares encargadas de la configuración e instanciación de nodos y sus protocolos, dispositivos de red, canales y aplicaciones, así como configuración de IPs y direcciones MAC. Estas tareas se tornan muy repetitivas y tediosas. Los helpers de ns-3, pretenden dar funcionalidades de configuración a un mayor nivel de abstracción y evitar estos problemas, dando una sintaxis más simple y obviando pasos de bajo nivel.

3.1.6. Planificador de Eventos (*Scheduler*)

Ns-3 es un simulador de eventos discretos, y para manejar los eventos se hace uso de un planificador. El planificador atiende los eventos que están en la cola en orden según el tiempo de simulación que tienen asignados. Cada evento atendido puede o no generar más eventos que también son agregados a la cola. La simulación termina cuando ya no hay más eventos en la cola del planificador. En el caso de los eventos recurrentes, es decir, que se reagendan a sí mismos, es necesario ejecutar la función *stop* para que termine la simulación, de lo contrario no terminará.

3.1.7. *Probe*

En ns-3, los objetos de tipo *Probe* son esenciales para la monitorización y recolección de datos durante una simulación. Estos objetos se asocian a variables

específicas y registran sus valores a lo largo de la simulación. Los *Probes* son especialmente útiles para la extracción de métricas.

Para implementar un *Probe*, se utiliza la clase de C++ *Probe*, que se especializa mediante distintas subclases dependiendo del tipo de datos que se desea monitorizar. Los *Probes* pueden ser configurados para capturar una amplia variedad de métricas, tales como el *delay* de los paquetes, la pérdida de paquetes, el uso de ancho de banda, entre otros. Estos datos pueden luego ser utilizados para analizar el rendimiento de red y para optimizar configuraciones y parámetros.

Además de la captura de métricas, los *Probes* pueden ser utilizados en conjunto con herramientas de trazado y registro para proporcionar una visión detallada del comportamiento de la red. Esta integración facilita la identificación de problemas y cuellos de botella, permitiendo ajustes precisos y mejoras en el diseño de la red.

3.1.8. *Flow Probe*

Los objetos de tipo *FlowProbe* en ns-3 están diseñados específicamente para monitorizar el flujo de paquetes en puntos determinados de la red. A diferencia de los *Probes* generales que pueden monitorizar una variedad de variables, los *FlowProbes* se enfocan en la observación detallada del tráfico de red, proporcionando información crítica sobre el flujo de datos entre nodos.

El *FlowProbe* se integra con el *FlowMonitor*, un componente global en ns-3 que recoge y organiza la información de todos los *FlowProbes* desplegados en la red. Cada *FlowProbe* registra datos como el *delay*, *throughput*, *jitter* y la pérdida de paquetes, y reporta estos datos al *FlowMonitor*. Esta información es esencial para analizar el comportamiento de los flujos de tráfico de red e identificar problemas como congestión o rutas ineficientes.

La implementación de los *FlowProbes* se realiza mediante la clase de C++ *FlowProbe*, que puede ser extendida y personalizada para capturar métricas adicionales según las necesidades específicas de la simulación. Los datos recolectados por los *FlowProbes* son agregados y pueden ser exportados a formatos de archivo estándar para su posterior análisis con herramientas externas.

El uso de *FlowProbes* es especialmente beneficioso en simulaciones complejas donde es crucial entender cómo se comportan los flujos de datos bajo diferentes configuraciones y condiciones de red.

3.1.9. *FlowMonitor*

FlowMonitor (Carneiro, Fortuna, y Ricardo, 2010), ver figuras 3.2 y 3.3, es un módulo de ns-3 que facilita al usuario la extracción de un conjunto de métricas, que son usadas frecuentemente para monitorizar el rendimiento de una red, de forma sencilla, sin necesidad de agregar código extra a la simulación. Las métricas utilizadas son: tasa de bits, duración, *delay*, tamaños de paquete y *ratio* de pérdida de paquetes, y son guardadas en la clase de C++ *Stat*. El módulo analiza el tráfico de toda la red y guarda estas métricas en un archivo, para que luego puedan ser revisadas. Se debe de tener en cuenta que debido a la

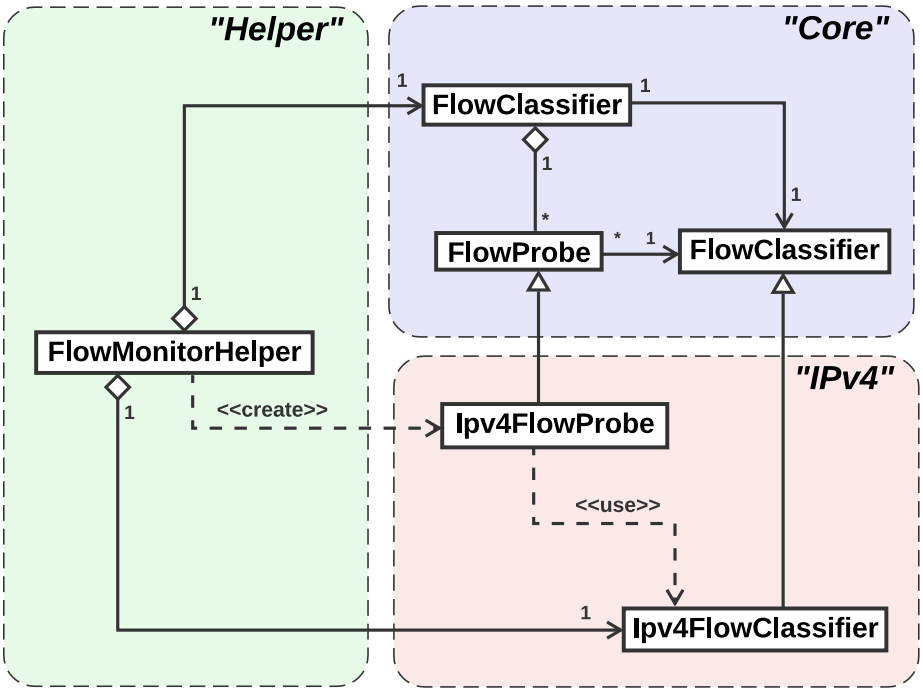

Figura 3.2: Diagrama de Componentes de "*FlowMonitor*" (Carneiro y cols., 2010).

tarea de monitorizar y extraer datos de la red, se agrega un *overhead* que luego es percibido en los tiempos totales de ejecución de simulación.

3.1.10. Tracing

El sistema de *tracing* de ns-3 le permite al usuario registrar los eventos de interés para la simulación sin la necesidad de grabar todos los eventos e inundar de datos el archivo de salida, como ocurre en el caso de los sistemas de paquete de salida predefinidos (*pre-defined bulk output*). Para construir el sistema de *tracing* se hace uso de los objetos *TraceSource*, que son invocados cuando suceden eventos de interés y tienen acceso los datos relacionados con dicho evento, permitiéndole registrar la información necesaria. Las entidades que usan los *TraceSources* son llamadas *trace sinks*.

3.2. 5G-LENA

5G-LENA es un proyecto de código abierto que extiende las capacidades del simulador de redes ns-3 para incluir la simulación de redes 5G *New Radio* (NR). Este módulo permite a los investigadores y desarrolladores estudiar, evaluar y

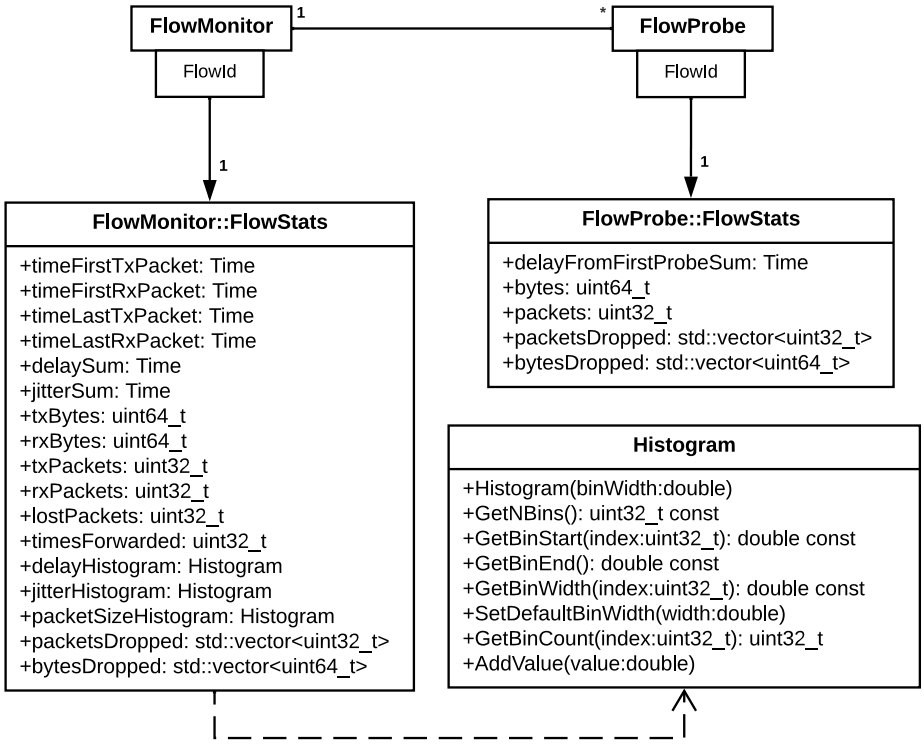

Figura 3.3: Diagrama *UML* de "*FlowMonitor*".

mejorar diversas tecnologías y protocolos asociados con las redes móviles de quinta generación.

3.2.1. Objetivos y Características de 5G-LENA

El principal objetivo de **5G-LENA** es proporcionar una plataforma flexible y extensible para la investigación y desarrollo de tecnologías 5G. Algunas de las características más destacadas de **5G-LENA** incluyen:

- **Simulación Completa de Red 5G NR**: 5G-LENA permite la simulación de todos los componentes de una red 5G, incluyendo estaciones base (*gNBs*), dispositivos de usuario (*UEs*) y el núcleo de red 5G (ver Figura 3.4).

- **Modelos Físicos y de Canal**: Incluye modelos avanzados de la capa física y de canal, que permiten simular condiciones de propagación realistas y la interacción de señales de radio en entornos complejos. Algunos ejemplos son: modelo de interferencia, modelo de espectro y el modelo de formación de haces (*beamforming*).

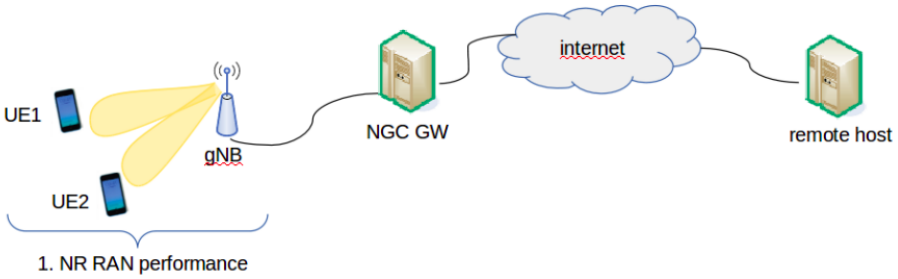

Figura 3.4: Diagrama de una arquitectura NR *end-to-end* extraído de (CTTC OpenSim Research Unit, 2024).

- **Flexibilidad y Extensibilidad**: Gracias a su diseño orientado a objetos, 5G-LENA puede ser fácilmente extendido para incluir nuevos modelos y protocolos, facilitando la investigación de nuevas tecnologías y mejoras en la red 5G.
- **Compatibilidad con ns-3**: 5G-LENA está completamente integrado con ns-3, lo que permite a los usuarios aprovechar todas las funcionalidades y herramientas disponibles en ns-3 para sus simulaciones.

3.2.2. Arquitectura de 5G-LENA

La arquitectura de 5G-LENA se basa en una serie de módulos que representan los diferentes componentes y funcionalidades de una red 5G. Entre los módulos principales se incluyen:

- **Módulo de la Capa Física (PHY)**: Este módulo simula la transmisión y recepción de señales de radio, incluyendo aspectos como la modulación, codificación y los efectos del canal de propagación.
- **Módulo de la Capa de Enlace (MAC)**: Simula las funcionalidades de la capa de enlace, como la programación de recursos, control de acceso al medio y la gestión de la calidad de servicio (QoS).
- **Módulo de Red (RLC y PDCP)**: Incluye las funcionalidades de la capa de control de enlace radio (RLC) y del protocolo de convergencia de datos de paquetes (PDCP), esenciales para la gestión y transmisión de datos en la red.
- **Nodos de Red**: Representan las estaciones base (*gNBs*) y los dispositivos de usuario (UEs), que interactúan con los diferentes módulos para simular una red 5G completa.

3.2.3. Usos y Aplicaciones de 5G-LENA

5G-LENA es utilizado en una amplia variedad de investigaciones y aplicaciones, incluyendo:

- **Evaluación de Rendimiento**: Permite evaluar el rendimiento de diferentes configuraciones y tecnologías en una red 5G, analizando métricas como la tasa de bits, latencia y pérdida de paquetes.
- **Desarrollo de Nuevas Tecnologías**: Facilita el desarrollo y prueba de nuevas tecnologías y protocolos para redes 5G, como técnicas avanzadas de acceso al medio, algoritmos de programación y mejoras en la gestión de recursos.
- **Simulación de Escenarios Complejos**: 5G-LENA permite la simulación de escenarios complejos y realistas, incluyendo la coexistencia de múltiples celdas, movilidad de usuarios y condiciones de propagación variables.

3.2.4. Integración con otros Módulos de ns-3

Además de su funcionalidad específica para 5G, 5G-LENA se integra con otros módulos de ns-3 para proporcionar una plataforma de simulación completa y versátil. Esto incluye simulación de tráfico de aplicaciones y herramientas de análisis y visualización de resultados.

3.2.5. Desarrollo y Comunidad

El desarrollo de 5G-LENA es llevado a cabo por una comunidad activa de investigadores y desarrolladores, que contribuyen con nuevas funcionalidades, mejoras y correcciones. El proyecto está alojado en una plataforma de código abierto, lo que facilita la colaboración y el acceso a los últimos avances en simulación de redes 5G.

5G-LENA aporta positivamente a las capacidades de ns-3 para proporcionar una herramienta poderosa y flexible para la investigación y desarrollo de tecnologías 5G. Su diseño modular y extensible, junto con su integración con ns-3, lo convierten en una plataforma ideal para la simulación y evaluación de redes móviles de próxima generación.

3.2.6. En suma

En resumen, ns-3 y 5G-LENA representan herramientas versátiles para la simulación y análisis de redes modernas, especialmente en el contexto de las tecnologías 5G. La flexibilidad y extensibilidad de ns-3, combinadas con las capacidades específicas de 5G que ofrece 5G-LENA, proporcionan a investigadores y desarrolladores un entorno completo para explorar, evaluar y optimizar diversos aspectos de las redes de comunicación avanzadas. La modularidad de

estas herramientas ofrece un punto de partida sólido para futuras extensiones y mejoras.

Capítulo 4

Nuestros Aportes

Los aportes del proyecto constituyen el análisis, planificación e implementación de un sistema de monitorización de redes 5G-NR que sea capaz de:

- Detectar anomalías en una red a nivel capa de transporte. Esto implica detectar si un flujo de paquetes tuvo un rendimiento destacadamente inferior al del resto de los flujos adyacentes durante un intervalo de tiempo. Esto deberá realizarse desde la propia red, es decir, sin indicaciones externas.
- Dado un flujo de paquetes IP con problemas de rendimiento, localizar dinámicamente el enlace *point-to-point* a nivel de capa física con peor rendimiento, por ejemplo, detectar el enlace que esté introduciendo la peor medida de *delay* en el flujo.
- Funcionar en una red capaz de manejar tráfico 5G NR con enlaces ópticos (representados por *links* genéricos de ns-3), con el objetivo que el ambiente de prueba sea un fiel reflejo de las nuevas tecnologías sobre las que se están desarrollando avances.
- Demostrar tener una capacidad de accionar en función del estado de congestión de red.

4.1. Plataforma de desarrollo

A continuación se describe la plataforma de desarrollo utilizada para construir la herramienta presentada en este trabajo. Para desarrollar un sistema de monitorización de redes 5G, primero se debe contar con una plataforma que sea capaz de simular las redes 5G, interactuar con la simulación y realizar mediciones en la red. La simulación de redes 5G representa un desafío significativo debido a los numerosos y complejos requisitos técnicos que estas redes presentan en cada una de sus capas [1].

[1]Siempre que se habla de capas de red se hace referencia al modelo *OSI* de cinco capas: Capa de Aplicación, Capa de Transporte, Capa de Datos, Capa de Enlace y Capa Física,

A continuación se describen los requerimientos técnicos de esta plataforma de simulación.

Capa Física

- Modelado de la transmisión de señales de radio, incluyendo modulación, codificación y técnicas de multiplexación.
- Uso de las características específicas de la tecnología 5G NR, como la numerología, las bandas de frecuencia y el ancho de banda del canal.
- Simulación de los efectos del canal inalámbrico, como la atenuación, el desvanecimiento y el ruido.
- Configuración de parámetros de propagación de la señal y de modelos de pérdida.

Capa de Enlace de Datos

- Modelado de la capa MAC (Medium Access Control), incluyendo técnicas de acceso al medio, como *TDD* (*Time Division Duplex*) y *FDD* (*Frequency Division Duplex*).
- Implementación de protocolos de enlace específicos de 5G NR como RLC (Radio *Link* Control).
- Gestión de recursos de radio, incluyendo planificación de la asignación de recursos y esquemas de programación.

Capa de Red

- Modelado de la arquitectura de red 5G, incluyendo la estructura de red central (*core network*) y red de acceso (*access network*).
- Implementación de protocolos de enrutamiento, como el protocolo IP (*Internet Protocol*) en sus versiones 4 y 6.
- Configuración de la topología de red, incluyendo la distribución y la interconexión de los nodos de red.
- Simulación de mecanismos de movilidad, como *handover* y *roaming*, en entornos heterogéneos de red.

---

siendo la Capa Física la capa uno y siguiendo secuencialmente el orden la numeración para el resto.

Capa de Transporte (Transport Layer)

- Modelado de protocolos de transporte, como TCP (*Transmission Control Protocol*) y UDP (*User Datagram Protocol*).
- Implementación de mecanismos de segmentación de datos.
- Simulación de la gestión de la congestión y el control de flujo en redes 5G de alta velocidad y baja latencia.

Capa de Aplicación (*Application Layer*)

- Modelado de la interacción entre las aplicaciones y las capas inferiores de red, incluyendo la calidad de servicio (QoS).
- Simulación de aplicaciones, servicios y los diferentes perfiles de tráfico típicos de 5G, ver Tabla 4.1.

4.2. Desarrollo del Escenario

Habiendo decidido que la plataforma de desarrollo sería ns-3 con 5G-LENA, el siguiente desafío fue la construcción de un escenario de simulación representativo de una red 5G. Para la construcción del escenario se utilizó como base un ejemplo provisto por el mismo módulo nr "cttc-nr-traffic-ngmn-mixed". A continuación se pasa a describir los aspectos principales de este escenario.

4.2.1. Topología de Red Simulada

Por topología se entiende los componentes de la simulación, como estos se encuentran dispuestos e interconectados. En este caso la simulación presenta diferentes elementos, siendo los principales: radio bases (*gNBs*), dispositivos de usuario (*UEs*), el núcleo de red (*5GCN*) y nodos remotos a la red, que simulan el acceso a Internet, como se muestra en la Figura 3.4. Las radio base se ubican en una grilla hexagonal con tres sectores por cada una. En el contexto de las redes móviles, este tipo de escenario implica organizar las estaciones base y los dispositivos de usuario en una estructura hexagonal similar a una colmena de abejas (ver Figura 4.1). Las estaciones base, o nodos de *gNB*, se distribuyen estratégicamente para proporcionar cobertura inalámbrica, mientras que los dispositivos de usuario se mueven entre las células de cobertura de estas estaciones base. Este enfoque se elige por su eficiencia en cobertura y capacidad de red. Adicionalmente, se parametrizó la cantidad de *gNBs* dispuestos en el escenario, a modo de facilitar la experimentación. Este diseño de grilla hexagonal permite observar cómo se distribuyen las áreas de cobertura y cómo se afecta el SINR (Relación Señal-Interferencia-más-Ruido) en diferentes coordenadas, como se muestra en la Figura 4.2.

Tipo de Tráfico	Descripción
UDP CBR	Tráfico de Datos en Modo Ráfaga. Implica la transmisión de datos en paquetes UDP (User Datagram Protocol) con un flujo de bits constante.
FTP	Transferencia de archivos utilizando el protocolo FTP en modo secuencial.
NGMN FTP	Tráfico de transferencia de archivos definido por NGMN. Sigue los estándares y recomendaciones de la NGMN (Next Generation Mobile Networks Alliance).
NGMN VIDEO	Tráfico de video definido por NGMN. Implica la transmisión de contenido multimedia según las especificaciones de NGMN.
HTTP	Tráfico de hipertexto. Se utiliza para acceder a sitios web y transferir contenido como texto, imágenes y multimedia a través del protocolo HTTP.
NGMN GAMING	Tráfico de juegos definido por NGMN. Implica la transmisión de datos relacionados con juegos en línea según las especificaciones de NGMN.
NGMN VOIP	Tráfico de Voz sobre IP (VoIP) definido por NGMN. Implica la transmisión de comunicaciones de voz a través de redes IP, como Internet.
NGMN MIXED	Combinación de diferentes tipos de tráfico. Por ejemplo, FTP, HTTP, transmisión de video, VoIP y juegos en línea, con distribución proporcionada (por ejemplo, 10 % FTP, 20 % HTTP, 20 % transmisión de video, 30 % VoIP, 20 % juegos).

Tabla 4.1: Descripción de los tipos de tráfico disponibles en la simulación.

4.2.2. Consideraciones sobre los enlaces ópticos

Es importante señalar que actualmente no existe un simulador de redes ópticas basado en ns-3. Crear un simulador desde cero está fuera del alcance de este proyecto de grado. Por lo tanto, se tomarán como enlaces ópticos los enlaces genéricos de ns-3, aprovechando su flexibilidad y modularidad para representar las conexiones necesarias en las simulaciones de redes 5G.

Sin embargo, es crucial tener en cuenta que el uso de enlaces genéricos puede introducir ciertas limitaciones, como la falta de precisión en la representación de características específicas de enlaces ópticos reales, lo que podría afectar la exactitud de algunos resultados de la simulación. Esta simplificación, aunque necesaria para el alcance actual del proyecto, abre la puerta a futuras mejoras y expansiones del sistema de simulación para incorporar modelos más precisos de redes ópticas.

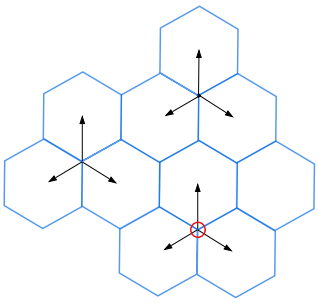

Figura 4.1: Estructura de *gNBs* utilizada en los escenarios de experimentación.

4.2.3. Principales aspectos del escenario

A continuación se hace un listado de los principales aspectos del escenario sobre el cual se simularon las redes 5G-NR que se monitorizaron.

- **Configuración del Escenario:** El código inicializa varios parámetros de simulación como el modo de operación (*TDD* o *FDD*), la dirección del tráfico (*DownLink* o *UpLink*), y el número de Equipos de Usuario (UEs) por radio base (*gNB*).

- **Topología de Red:** Se utiliza la clase "*HexagonalGridScenarioHelper*" perteneciente al módulo base de ns-3, para crear una topología de red de teselado hexagonal, donde se colocan *gNBs* (estaciones base) y UEs en un patrón específico, como se mencionó en la Sección 4.2.1.

- **Configuración del Espectro:** El espectro se divide en bandas, cada una conteniendo uno o más portadores de componentes (*CCs*), y cada CC se divide además en partes de ancho de banda (*BWPs*). La configuración del espectro se establece en función del escenario y el modo de operación seleccionados (*TDD* o *FDD*).

- **Modelos de Canal y Modelos de Error:** El código configura varios modelos de canal, como el "*ThreeGppChannelModel*" y modelos de pérdida,

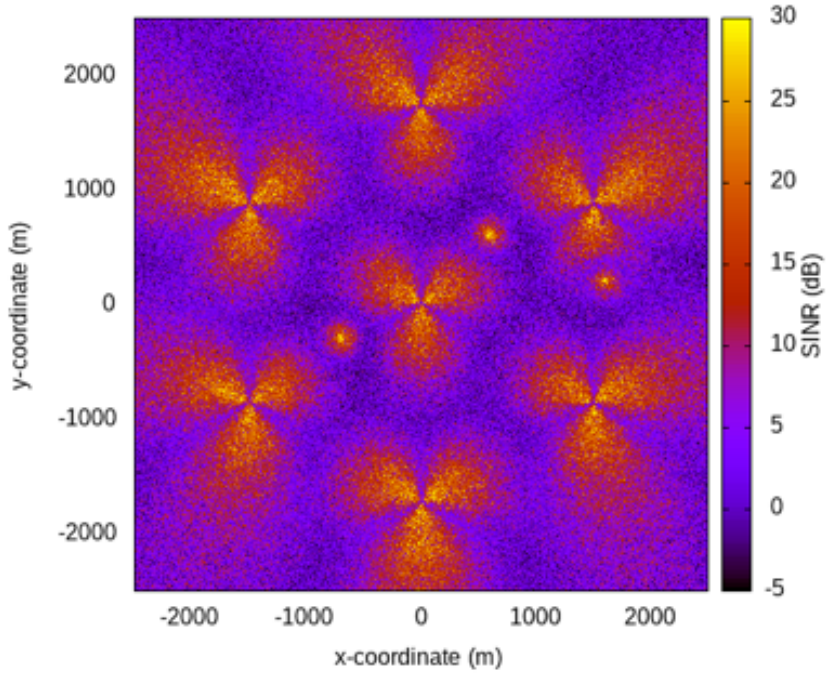

Figura 4.2: Ejemplo de áreas de cobertura de las *gNBs*, extraído del manual de 5G-LENA (CTTC OpenSim Research Unit, 2024). Se muestra el SINR (por sus siglas en inglés Signal-to-Interference-plus-Noise Ratio) en decibeles por cada coordenada (x, y).

así como modelos de error para transmisiones tanto de enlace *DownLink* como de enlace *UpLink*.

- ***BeamForming*:** Se utiliza "*IdealBeamformingHelper*" para manejar la formación de haces de la simulación.
- **Configuración del Stack NR:** La clase "*NrHelper*" se utiliza para crear y configurar los componentes del stack de protocolos NR, incluyendo la capa física, capa MAC y capa RLC. Se configuran varios atributos como la numerología, potencia de transmisión, algoritmos de planificación y configuraciones de antena para *gNBs* y *UEs*.
- **Instalación de Dispositivos de Red:** Se utiliza "*NrHelper*" para instalar los dispositivos de red NR (dispositivos *gNB* y UE) en los contenedores de nodos respectivos. Se crean contenedores de dispositivos separados (*gnbSector1NetDev*, *gnbSector2NetDev*, *gnbSector3NetDev*, *ueSector1NetDev*, *ueSector2NetDev*, *ueSector3NetDev*) para cada sector.
- **Integración 5GCN:** Se utiliza "*NrPointToPointEpcHelper*" para inte-

grar el *Core* de Paquetes Evolucionado (5GCN) con la red NR. Este proceso asegura una comunicación fluida entre los componentes de red, como se ilustra en la Figura 3.4.

- **Generación de Tráfico:** El código incluye varios modelos de tráfico (ver Tabla 4.1), pero el modelo de tráfico específico se selecciona en función del parámetro "trafficTypeConf".

- **Acceso a Internet:** Para una simulación completa se agrega un nodo "*remoteHost*", que actúa de elemento externo a la red, encontrándose disponible en Internet.

4.3. Sistema de monitorización

En esta sección se busca proporcionar una comprensión completa del sistema de monitorización desarrollado y su funcionamiento, presentando su diseño e implementación. Se define el comportamiento esperado del sistema y los conceptos clave involucrados. Luego, se detalla la implementación de la función principal de monitorización ("*reportFlowStats*") y se describe el desarrollo de una solución a medida para obtener un análisis más detallado del rendimiento de red. Finalmente, se introduce la función "*nodeToNodeTrigger*", que permite identificar y diagnosticar enlaces con mal rendimiento en red simulada.

4.3.1. Comportamiento del sistema

Para diseñar el sistema de monitorización, se definió inicialmente su comportamiento esperado. Este comportamiento define cómo se esperaba que el sistema de monitorización funcionara en un escenario de uso típico.

En la Figura 4.3, se muestra el rendimiento de red sobre el tiempo de simulación (ms). El rendimiento de red podrá configurarse, tomando el valor del *delay*, *jitter*, *throughput* o una combinación de las métricas anteriores. La gráfica muestra el rendimiento de red sobre el tiempo, mientras que los círculos representan instancias de medición en las que el sistema tiene acceso a la información de flujos de paquetes, sus tiempos de *delay*, la cantidad de paquetes enviados, entre otros. También se puede observar una línea de umbral, denominada "*threshold*", que representa el límite de lo que se considera un rendimiento de red aceptable. Si se traspasa el umbral, se deberá tomar acción para indagar en el causante del empeoramiento del rendimiento y localizar el enlace responsable (el de peor rendimiento). Es importante destacar que dependiendo de la métrica utilizada, este umbral puede usarse de tope superior o inferior, ya que es deseable tener valores altos de *throughput*, pero valores bajos de *delay* y *jitter*.

Notar que cuando el rendimiento de red está por encima del umbral, los intervalos de tiempo entre los puntos de medición son más espaciados, pero en el momento que queda por debajo, los intervalos comienzan a ser más cortos, a modo de monitorizar más seguido los cambios de red. Cuando se vuelve a

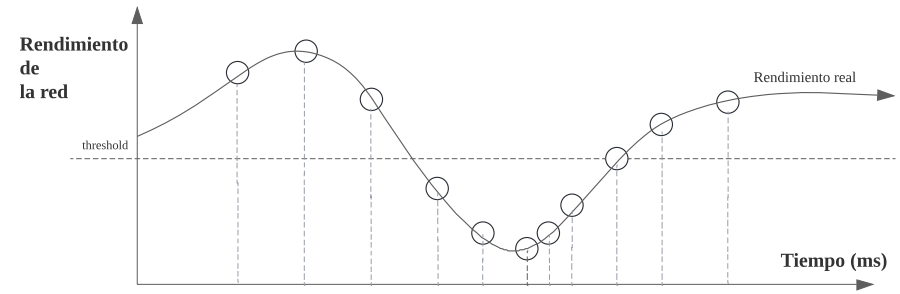

Figura 4.3: La Figura muestra el rendimiento de red a lo largo del tiempo de simulación (ms). Los círculos representan instancias de medición del rendimiento real de red. La línea horizontal ("*threshold*"), indica el umbral de rendimiento aceptable. Cuando el rendimiento de red está por encima del umbral, los intervalos de medición son más espaciados y cuando el rendimiento cae por debajo del umbral, los intervalos de medición se vuelven menos espaciado.

estar por encima del umbral, los intervalos de medición se restablecen a su valor inicial.

Los valores de los umbrales para cada variable queda a cargo del usuario responsable de la red, ya que cada red tiene sus particularidades (explicado en la Sección 2.1.2 : "Características del Tráfico en Redes 5G" y Tabla 4.1: "Descripción de los tipos de tráfico disponibles en la simulación.") y puede considerar diferentes valores aceptables para las métricas presentadas. Por ejemplo, en una red dedicada a videoconferencias, el *delay* y *jitter* tendrán umbrales más estrictos debido a la sensibilidad de estas aplicaciones a retrasos y variaciones en el tiempo de entrega de los paquetes. Por otro lado, en una red orientada al intercambio de archivos, el *throughput* podría ser la métrica más crítica, con umbrales que toleran mayores fluctuaciones en el *delay* y *jitter*, siempre y cuando el volumen de datos transmitidos se mantenga alto.

4.3.2. Implementación

Se partirá especificando el concepto de umbral que se mencionó en la Sección 4.3.1.

Definición del umbral

El umbral que se utilizara será una combinación de las siguientes métricas:

- ***Delay*(ms)**: Latencia, es el tiempo que demora un paquete entre que es enviado y recibido.
- ***Throughput*(mbps)**: Es la taza envio de información.

- ***Jitter*(ms)**: Es la variación de latencia o *delay* entre dos paquetes.

Se definirá que una métrica "supera" un umbral si:

1. El *delay* es menor que el umbral.
2. El *throughput* es mayor que el umbral.
3. El *jitter* es menor que el umbral.

Funcionamiento de "*reportFlowStat*"

Dado que el problema a resolver consiste en desarrollar un sistema de monitorización, se exploró el módulo "*FlowMonitor*", presentado en la Sección 3.1.9, ya que presenta herramientas de acceso a información de monitorización útiles para su implementación. A partir de este módulo, se creó una función de monitorización llamada "*reportFlowStats*", que utiliza los métodos y datos guardados para medir y analizar el rendimiento punto a punto (o a nivel de flujo) de una simulación de ns-3 con 5G-LENA. A continuación, se describe su funcionamiento en detalle.

Esta función invoca los métodos implementados por "*FlowMonitor*", y compara diferentes métricas de rendimiento de red con un "*threshold*". En traspasar el "*threshold*" (arriba en el caso del *delay*), genera *logs* del sistema que dan indicios de un mal rendimiento de red. También aumenta las frecuencias de medición de métricas de rendimiento para detectar y diagnosticar problemas con mayor precisión y rapidez, permitiendo una respuesta más efectiva y oportuna.

En primer lugar, la función define un tipo estructurado de datos llamado "*TrackedStats*" que contiene métricas representativas del rendimiento de red, como el *delay*, *delay* medio, *delay* del último paquete y *meanJitter* o fluctuación media (ver Figura 4.4). Además, se definen dos intervalos de tiempo de medición, un intervalo de tiempo mayor para cuando la red tiene un buen rendimiento, y un intervalo menor para el caso contrario.

Luego, la función invoca métodos pertenecientes a las clases "*FlowMonitor*", "*FlowClassifier*" y "*Simulator*". Todas estas interacciones se detallan en el diagrama de secuencia UML (Figura 4.5) explicado a continuación.

Como se muestra en el diagrama de la Figura 4.5, primero se obtienen las estadísticas de flujo "*FlowStats*" del "*FlowMonitor*". Esto último solo aplica para los nodos en los que se haya instalado la clase "*FlowMonitor*".

Estas estadísticas de flujo incluyen:

- Número de paquetes transmitidos y recibidos.
- Bytes transmitidos y recibidos.
- *Throughput* (mbps).
- *Delay* (ms).
- *Jitter* (ms).

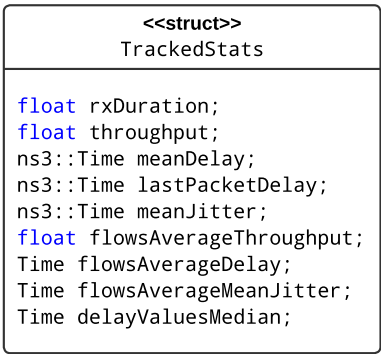

Figura 4.4: Diagrama de la clase *TrackedStats*, define el tipo de datos que persistirá las métricas en cada medición de las simulaciones.

Estas métricas se irán almacenando en la estructura "*TrackedStats*", y al finalizar la simulación serán persistidas en un archivo de salida, en una ubicación especificada por el usuario.

Adicionalmente, luego de procesar todos los flujos, la función calcula la media del *delay*, *throughput* y *jitter*, que se guarda en la variable "*FiftyTileFlowDelay*". A continuación se comparan las métricas de rendimiento calculadas con sus umbrales correspondientes (bloque *Alternative* del diagrama de secuencia 4.5). Si alguna de las métricas traspasa el umbral, se invoca la función "*nodeToNodeTrigger*" (explicada en la Sección 4.3.2), que pasará a analizar más en profundidad el *delay* entre los nodos de la red para identificar posibles cuellos de botella, en caso de que no los haya, no se ejecutan más acciones de medición hasta la siguiente llamada programada de "*reportFlowStats*".

Finalmente, se restablecen todas las estadísticas del "*FlowMonitor*" y se programa en el planificador de eventos de la clase "*Simulator*" para volver a ejecutarse después de un cierto tiempo. Este intervalo de tiempo se ajusta dinámicamente en función de si las métricas de rendimiento superaron o no los umbrales predefinidos.

En resumen, la función "*reportFlowStats*" recopila y procesa estadísticas de flujo, calcula métricas de rendimiento importantes, identifica posibles cuellos de botella en la red y se reagenda a sí misma para ejecutarse periódicamente durante la simulación.

Desarrollo de una solución a medida

Para desarrollar un sistema de monitorización capaz de localizar un enlace con mal rendimiento, se encontró que era necesario poder identificar la influencia de cada enlace nodo a nodo en las métricas punto a punto, desarrolladas en la Sección 4.3. Esto presentó un desafío, ya que tras una profunda investigación del módulo "*FlowMonitor*", se determinó que esta información no se encuentra dentro de las trazas que recolecta, o al menos no de la manera deseada. Este último

punto es mencionado porque tras investigar la arquitectura de "*FlowMonitor*", se encontró un identificador de paquetes entre las métricas de la clase "*FlowProbe*". Tras una inspección más en detalle, se determinó que no identificaban paquetes globalmente a lo largo del flujo, sino que se hacía a nivel de nodo, por lo que no era posible identificar desde donde vino cada paquete. En base a este descubrimiento, se optó por descartar la opción de utilizar este identificador, ya que no cumplía con los requerimientos del sistema de monitorización.

En vista de esta dificultad encontrada, se optó por explotar una de las mejores virtudes del módulo "*FlowMonitor*" (explicado en la Sección 3.1.9), su extensibilidad. Esto fue posible debido a que la arquitectura de "*FlowMonitor*", presenta un conjunto de clases que son fácilmente extensibles para poder modelar un comportamiento de monitorización más adecuado a las necesidades que se presenten. Un ejemplo de esto son las clases "*FlowMonitor*" y "*FlowProbe*". Estas clases se encargan de mantener las trazas del sistema punto a punto y a nivel de "*Probe*" o de nodo respectivamente. Adicionalmente, existe la clase "*Ipv4FlowProbe*" que es una clase hijo de "*FlowProbe*", que añade métodos para registrar métricas al momento de enviar y recibir paquetes IpV4 a través de cualquiera de las interfaces de un nodo.

A partir del funcionamiento de estas clases, se implementa una nueva llamada "*BigBrotherFlowProbe*", derivada de la clase "*Ipv4FlowProbe*", teniendo en cuenta estos módulos. Esta clase añade un atributo llamado "m_perPacketStats", con el objetivo de almacenar información sobre el *delay* introducido por el nodo para cada flujo y cada paquete asociado a él. Esta estructura garantiza la capacidad de reconstruir la ruta de cada paquete a través de la red, permitiendo identificar en cada momento el enlace con peor rendimiento, lo que se traduce al enlace que agrega más *delay*. Además, esta clase incluye una función "AddPacketStats" con polimorfismo, que recibe un atributo adicional llamado "*packetId*" y lo utiliza para poblar el mapa "m_perPacketStats".

Adicionalmente, se creó una clase llamada "*BigBrotherFlowMonitor*", que hereda de "*FlowMonitor*". Dado que la implementación anterior era responsable de instanciar la clase "*Ipv4FlowProbe*", y para utilizar nuestra solución se necesita instanciar "*big-brother-flow-probes*", se realizó una sobrecarga de la función ReportFirstTx: ReportFirstTx(Ptr<FlowProbe> probe, uint32_t flowId, uint32_t packetId, uint32_t packetSize).

Finalmente, también se creó una implementación de "*FlowMonitorHelper*" llamada "*bigBrotherFlowMonitorHelper*", que es una copia de la implementación provista originalmente pero en vez de instanciar los "*FlowMonitor*" instancia a "*bigBrotherFlowMonitor*". En el esquema UML (Figura 4.6), se puede ver cómo las diferentes clases se acoplan entre sí.

En resumen, la clase "*BigBrotherFlowProbe*" proporciona una forma de rastrear las estadísticas de cada paquete individual a medida que atraviesa la red, lo que permite un análisis más detallado del rendimiento y la identificación de cuellos de botella a nivel de enlace.

Funcionamiento de *nodeToNodeTrigger*

Luego de haber desarrollado una solución propia a medida capaz de generar las trazas con la información necesaria, desarrollada en Sección 4.3.2, surgió la necesidad de poder recopilar, resumir y lograr acceder esta información de una manera sencilla en un entorno de simulación. Con esto en mente, se desarrolla la función *nodeToNodeTrigger* que utiliza los métodos pertenecientes a la clase *BigBrotherFlowProbe* para acceder a información detallada sobre el rendimiento del flujo a nivel de paquete.

La función *nodeToNodeTrigger* es un componente esencial del sistema de monitorización desarrollado, que aprovecha las capacidades de la solución a medida implementada con las clases *BigBrotherFlowProbe* y *BigBrotherFlowMonitor*. Su objetivo principal es identificar el enlace con peor rendimiento en términos de *delay* entre nodos en la red simulada.

En la Figura 4.7, se puede apreciar el diagrama de secuencia de las invocaciones que componen el funcionamiento de *NodeToNodeTrigger* que se explicaran a continuación. El funcionamiento de *nodeToNodeTrigger* se puede desglosar en los siguientes pasos:

1. **Inicialización y carga de datos:**
   - La función recibe como parámetros un puntero a un "*FlowMonitor*" y la ruta a un archivo XML donde se almacenarán los resultados.
   - Carga o crea un archivo XML para almacenar las mediciones de red.
2. **Recopilación de estadísticas:**
   - Obtiene las estadísticas de flujo y los *probes* del monitor de flujo.
   - Inicializa un mapa *nodeToNodeDelay* para almacenar los *delay*s entre pares de nodos.
3. **Procesamiento de datos por flujo:**
   - Itera sobre cada flujo en las estadísticas.
   - Para cada flujo, crea un mapa *perPacketStats* que almacena información detallada sobre el recorrido de cada paquete.
4. **Análisis de *probes*:**
   - Para cada *probe* de tipo `BigBrotherFlowProbe`, extrae información detallada sobre los paquetes que pasaron por él.
   - Recopila datos sobre el ID del nodo y el *delay* acumulado para cada paquete.
5. **Cálculo de *delay*s entre nodos:**
   - Procesa la información recopilada para calcular el *delay* entre cada par de nodos consecutivos en la ruta de los paquetes.

- Actualiza el mapa *nodeToNodeDelay* con estos valores.

6. **Identificación del enlace con peor rendimiento:**
   - Encuentra el par de nodos con el mayor *delay* acumulado.
7. **Actualización del archivo XML:**
   - Registra todas las mediciones de *delay* entre nodos en el archivo XML.
   - Añade una entrada especial para el enlace con peor rendimiento.
8. **Retorno de resultados:**
   - Devuelve un par de enteros que representan los IDs de los nodos que forman el enlace con peor rendimiento.

Esta función permite aprovechar la información recopilada por *BigBrother-FlowProbe*, procesando los datos a nivel de paquete para obtener una visión clara del rendimiento de cada enlace en la red. Al identificar el enlace con mayor *delay*, proporciona una herramienta valiosa para el análisis y la optimización de la red simulada.

4.4. Creación de un formato estándar para *logs*

En los diagramas de secuencia presentados anteriormente se muestra un elemento llamando "*XMLDocument*", Figura 4.5 y 4.7. El mismo hace alusión a un archivo XML diseñado para llevar los *logs* de sistema de una manera que se puedan procesar programáticamente. El formato del mismo se puede apreciar en la Figura 4.1 y se pasará a describir a continuación.

```
<network-measurements>
    <worst-links>
        <worst-link>
            <delay-value>10000043</delay-value>
            <timestamp>10100000000</timestamp>
            <node-pair>
                <node-id>2</node-id>
                <node-id>3</node-id>
            </node-pair>
        </worst-link>
        <worst-link>
            <delay-value>4000043</delay-value>
            <timestamp>20100000000</timestamp>
            <node-pair>
                <node-id>1</node-id>
                <node-id>2</node-id>
            </node-pair>
```

```
        </worst-link>
    </worst-links>
    <delays>
        <delay>
            <node-pair>
                <node-id>0</node-id>
                <node-id>2</node-id>
            </node-pair>
            <measurements>
                <measurement>
                    <delay-value>4000043</delay-value>
                    <timestamp>10100000000</timestamp>
                </measurement>
            </measurements>
        </delay>
        <delay>
            <node-pair>
                <node-id>2</node-id>
                <node-id>3</node-id>
            </node-pair>
            <measurements>
                <measurement>
                    <delay-value>10000043</delay-value
                        >
                    <timestamp>10100000000</timestamp>
                </measurement>
                <measurement>
                    <delay-value>2000043</delay-value>
                    <timestamp>20100000000</timestamp>
                </measurement>
            </measurements>
        </delay>
        <delay>
            <node-pair>
                <node-id>1</node-id>
                <node-id>2</node-id>
            </node-pair>
            <measurements>
                <measurement>
                    <delay-value>4000043</delay-value>
                    <timestamp>20100000000</timestamp>
                </measurement>
            </measurements>
        </delay>
    </delays>
</network-measurements>
```

Listing 4.1: XML File

Durante el desarrollo del sistema de monitorización, se volvió necesario el desarrollo de una herramienta que permitiera ágilmente revisar si el mismo estaba funcionando de forma esperada. Inicialmente, se intentó depurar utilizando las *traces* del propio ns-3. Sin embargo, esto probó ser muy engorroso y limitante. Por lo tanto, se optó por utilizar la biblioteca *tiny-xml-2* (Thomason, 2024), como herramienta para construir un documento *XML* con los resultados del sistema luego de acabada la simulación. El formato construido se aprecia en el Listado 4.1.

Este formato XML representa mediciones de *delay* y enlaces críticos en una red. A continuación, se describe la estructura del archivo:

`<network-measurements>` Es el elemento raíz que contiene toda la información de las mediciones de red.

`<worst-links>` Este elemento contiene una lista de los enlaces más lentos o críticos de la red.

- `<worst-link>` Representa un enlace crítico específico.
  - `<delay-value>` El valor del *delay* medido en este enlace crítico (en nanosegundos).
  - `<timestamp>` La marca de tiempo en la que se registró esta medición del enlace crítico (en nanosegundos).
  - `<node-pair>` El par de nodos que define el enlace crítico.
    - `<node-id>` El identificador de cada nodo en el par.

`<delays>` Este elemento contiene una lista de *delays* medidos entre pares de nodos específicos.

- `<delay>` Representa las mediciones de *delay* entre un par de nodos.
  - `<node-pair>` El par de nodos para el que se registran las mediciones de *delay*.
    - `<node-id>` El identificador de cada nodo en el par.
  - `<measurements>` Contiene una lista de mediciones de *delay* individuales para este par de nodos.
    - `<measurement>` Una medición de *delay* específica.
      - `<delay-value>` El valor del *delay* medido (en nanosegundos).
      - `<timestamp>` La marca de tiempo en la que se registró esta medición (en nanosegundos).

En resumen, el archivo XML contiene información sobre los enlaces más lentos de la red (`<worst-links>`), así como mediciones detalladas de los *delays* entre pares de nodos (`<delays>`). Esto puede ser útil para identificar cuellos de botella y problemas de rendimiento en la red.

4.5. Heurística de monitorización

La heurística de monitorización es un componente crucial del sistema desarrollado. Esta heurística determina cuándo el sistema debe activar su mecanismo de detección y localización de problemas en la red.

Específicamente, la heurística de monitorización refiere a los criterios utilizados para decidir cuándo la función "reportFlowStats", explicada en la Sección 4.3.2, debe invocar a la función "nodeToNodeTrigger", explicada en la Sección 4.3.2, para buscar el enlace con peor rendimiento.

La importancia de esta heurística radica en su impacto directo sobre la eficiencia y efectividad del sistema de monitorización. Una heurística mal diseñada podría resultar en un exceso de falsos positivos, activando innecesariamente los mecanismos de búsqueda y haciendo que la herramienta de monitorización sea ineficiente en el uso de recursos computacionales. Por otro lado, una heurística demasiado conservadora podría fallar en detectar problemas reales, comprometiendo la credibilidad del sistema.

Para abordar este desafío, se desarrolló una solución que busca equilibrar la sensibilidad y la eficiencia del sistema. La aproximación se basa en el establecimiento dinámico de umbrales, utilizando como referencia el estado inicial de la red. Se determinó tomar como umbral la primera medición que se realice sobre la red. Esto involucra que el usuario deberá conscientemente invocar la primera llamada a "*reportFlowStats*", durante un periodo en el cual la red esté funcionando de manera óptima. Con esto se entiende que no se esté buscando activamente empeorar el rendimiento de la misma, sino que la primera llamada se tome para medir el comportamiento de la red bajo condiciones normales, y que este valor sirva de referencia para cuando la red experimente dificultades.

Esta estrategia tiene la ventaja que libera al usuario de tener que realizar exhaustivas pruebas, para determinar el *delay*, *throughput* y *jitter* promedio de sus simulaciones y reducir esfuerzos.

Con esta estrategia de elección de umbral, se puede decir que se utiliza una estrategia de monitorización binaria, es decir, se considera que la red está en un estado aceptable o no lo está, y el criterio que se utiliza para definir esto es estático, debido a que el umbral definido nunca cambia.

4.5.1. En suma

Habiendo explicado la visión del sistema de monitorización, los principales componentes, su funcionamiento e implementación, la siguiente etapa del proyecto consiste en validar que se comporte como es esperado.

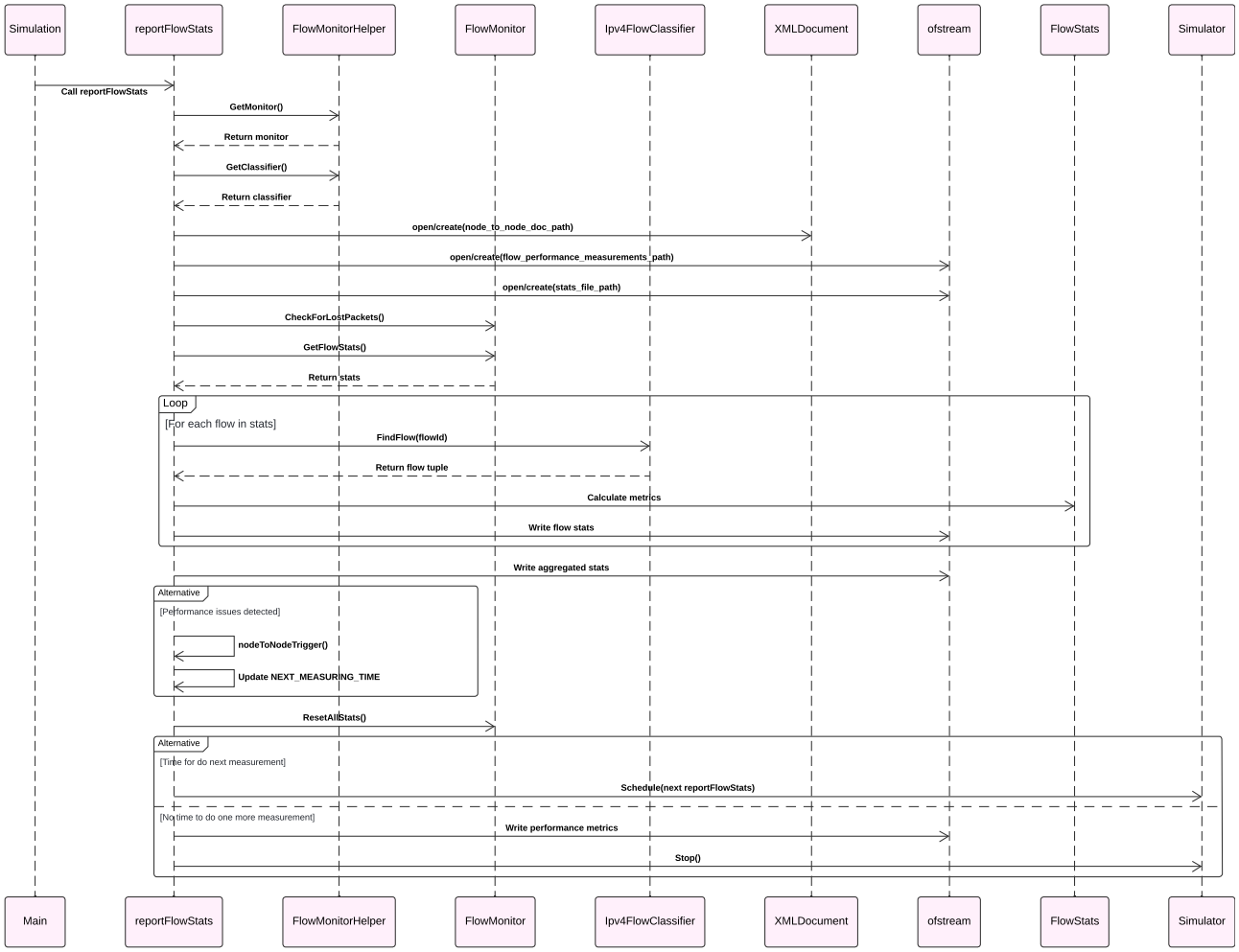

Figura 4.5: Diagrama de secuencia, invocaciones de *reportFlowStats*.

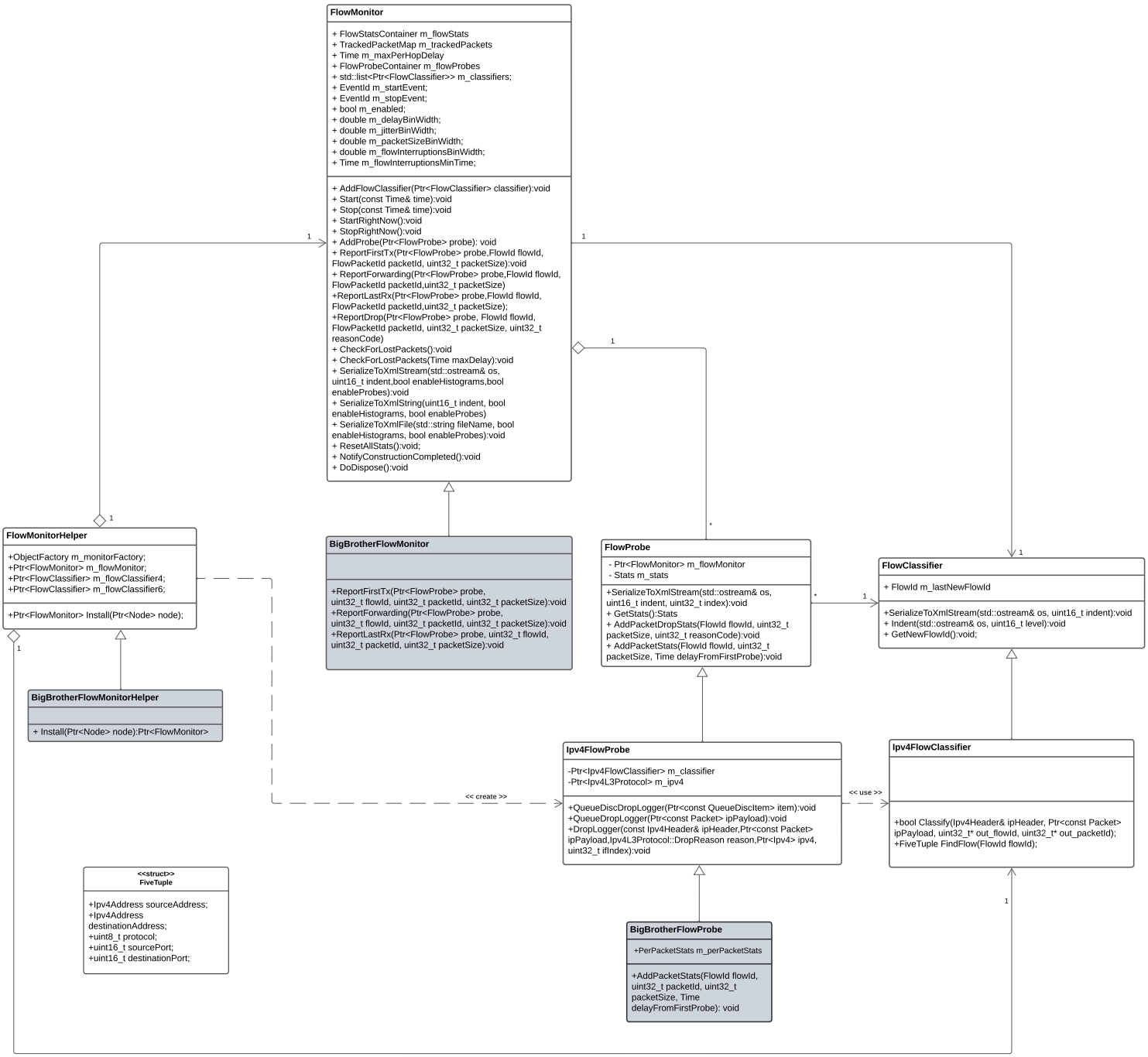

Figura 4.6: Arquitectura del módulo "*FlowMonitor*" con las clases agregadas y extendidas en color gris.

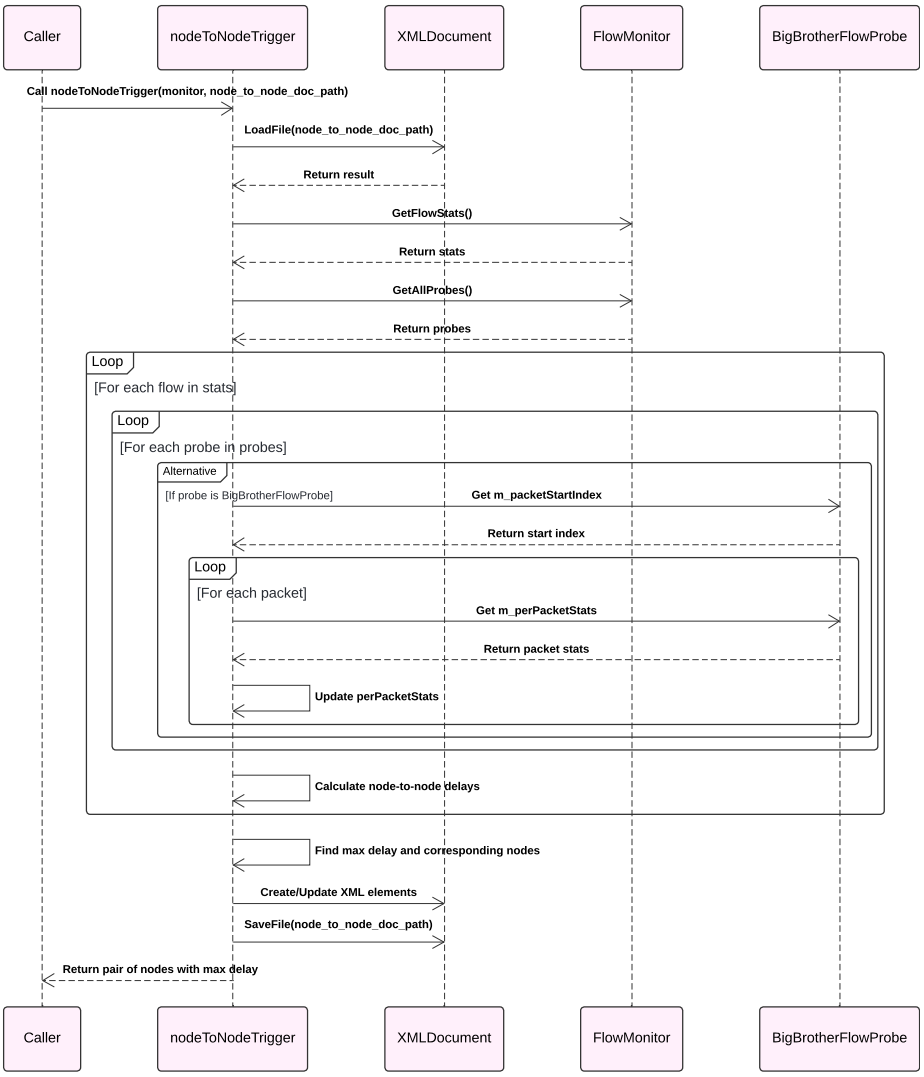

Figura 4.7: Diagrama de secuencia, invocaciones de *nodeToNodeTrigger*.

Capítulo 5

Experimentación

Para validar que el sistema de monitorización funciona correctamente, se adoptó una estrategia de validación progresiva respecto a las pruebas realizadas, comenzando con simulaciones muy sencillas que verifiquen el correcto funcionamiento de determinados aspectos del sistema, y luego complejizarla para lograr escenarios que se asemejen a la realidad de las redes 5G-NR.

A continuación se recuerdan los objetivos del sistema.

- Identificar un deterioro de rendimiento a nivel flujo a nivel *end-to-end*.
- Activar el sistema que incrementará la tasa de mediciones mientras este rendimiento se mantenga.
- Activar el sistema de identificación que buscará el *link* responsable de dicho rendimiento.
- Identificar el *link* con peor rendimiento, y calcular el *delay* que está introduciendo en la métrica *end-to-end*.

5.1. Prueba de Instalación y Escucha activa

La primera prueba, tiene como objetivo validar que las clases implementadas dentro del módulo "*FlowMonitor*", extraigan las métricas nodo a nodo correctamente dado un flujo específico. Esta capacidad es fundamental para el funcionamiento del sistema, ya que por defecto "*FlowMonitor*" no es capaz de identificar el rendimiento de enlaces nodo a nodo, únicamente *end-to-end*, o a nivel de nodo, pero sin la capacidad de identificar desde donde provienen los paquetes.

Para realizar esta validación, se optó por utilizar un escenario de prueba basado en un ejemplo disponible por defecto en ns-3 llamado "*simple-global-routing*".

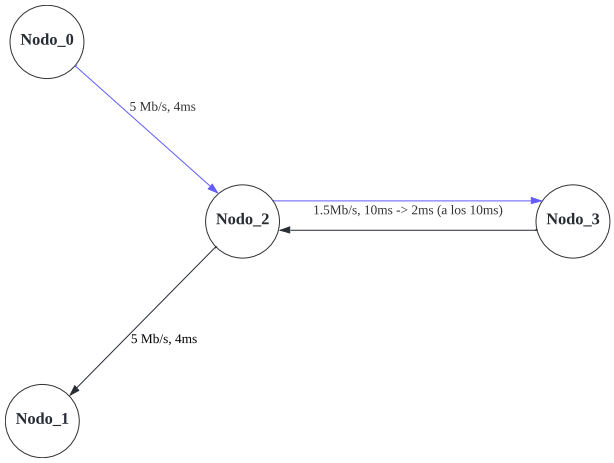

Figura 5.1: Arquitectura de simple-global-routing.

5.1.1. Escenario Demo

Este escenario está basado en un ejemplo de la sección de enrutamiento de ns-3 que fue de gran utilidad por su simpleza, permitiendo iterar sobre los experimentos más rápidamente. A continuación se pasará a detallar el escenario modificado que se utilizó para extraer mediciones. Este escenario de simulación representa una topología de red simple con cuatro nodos (`n0`, `n1`, `n2`, `n3`) conectados por enlaces punto a punto. A continuación se listan los aspectos importantes de esta simulación.

- Topología de Red (ver Figura 5.1)
  - `n0` y `n1` están conectados a `n2` a través de enlaces de 5 Mb/s y 4 ms de *delay*.
  - `n3` está conectado a `n2` a través de un enlace de 1.5 Mb/s y 10 ms de *delay* inicialmente. Luego se cambia el *delay* de este enlace a 2 ms y finalmente se cambia a 11 segundos.
- Flujos de datos
  - Hay dos flujos UDP constantes (*CBR*) de `n0` a `n3` y de `n3` a `n1`.
- Configuración
  - El tamaño de paquete UDP es de 210 bytes, con una tasa de transferencia entre paquetes de 448 Kbps.
  - Se utilizan colas *`DropTail`* (FIFO).
  - Se habilita el seguimiento de colas y recepciones de paquetes en un archivo de traza.

5.1.2. Resultados

El primer escenario de prueba se conforma de simples nodos con enlaces punto a punto, y dos flujos de paquetes que pasan por ellos, ver Sección 5.1.1. Los flujos son disjuntos en los nodos `n0` y `n1`, por lo que en el caso de utilizar la solución de monitorización, el sistema debería registrar correctamente trazas para los nodos `n0`, `n2` y `n3`, pero no para `n1` en el Flujo 1. También debería registrar correctamente trazas para los nodos `n2`, `n3` y `n1`, pero no para `n0` en el Flujo 2. La capacidad del sistema para generar trazas de baja granularidad únicamente en los nodos pertenecientes al flujo que se está midiendo, es de suma importancia para que pueda operar en simulaciones más complejas con cientos de nodos y sin generar un *overhead* limitante o trazas ineficientes.

Cuando se instala el sistema de monitorización en el escenario, usando la función "*nodeToNodeTrigger*" (explicada en Sección 4.3.2), se logra obtener la traza del sistema (ver Figura 5.2). A continuación se explicará cómo esto es evidencia de que el sistema tiene la capacidad de realizar mediciones activas únicamente en los nodos que corresponden.

Las secciones de traza de esta imagen provienen del archivo *XML* presentado en la Figura 4.1, donde cada una se encuentra asociada por medio de una flecha al enlace punto a punto que miden. Los siguientes puntos deben ser apreciados para entender por qué este experimento es exitoso.

- Los enlaces `(no,n2) y (n1,n2)` son medidos una única vez. Esto es correcto porque cada uno es utilizado por un único flujo, flujos 1 y 2 respectivamente.

- El enlace `(n2,n3)` es medido en dos ocasiones. Esto es correcto porque es accedido tanto por el Flujo 1 como por el Flujo 2, y es el único enlace que se repite en ambos flujos. Que sea el único enlace con dos elementos <`measurements`>, es indicador de éxito.

- Los *delays* medidos se corresponden con los configurados en el escenario explicado en la Sección 5.1.1, donde `(n0,n2) y (n1,n2)` tienen un *delay* de 4 ms o 4×10^6 ns y `(n2,n3)` tiene un *delay* 10 ms.

5.2. Prueba de cambios dinámicos de *Delay*

Luego de validar que el sistema es capaz de monitorizar solamente los nodos correspondientes al flujo investigado, el siguiente paso es establecer una prueba para verificar que detecte cambios en el *delay* de un enlace particular.

5.2.1. Implementación de "*ChangeLinkDelay*"

Para realizar este experimento, fue necesario contar con la capacidad de cambiar el *delay* de un enlace particular de una manera dinámica. Es decir, que

ocurra en un momento que se definirá paramétricamente, ya que con esta capacidad se podrá evaluar si el sistema de monitorización detecta adecuadamente los cambios en el *delay* de un enlace a lo largo de una simulación.

Para implementar este cambio se requirió crear una función auxiliar llamada "*ChangeLinkDelay*" (ver Listado 5.1), capaz de modificar el atributo responsable del *delay* en un enlace punto a punto.

```
1 void ChangeLinkDelay(Ptr<NetDevice> device, Time newDelay) {
2     Ptr<PointToPointChannel> p2pChannel =
          device->GetChannel()->GetObject<PointToPointChannel>();
3     if (p2pChannel) {
4         p2pChannel->SetAttribute("Delay", TimeValue(newDelay));
5     } else {
6         NS_LOG_ERROR("Failed to get PointToPointChannel object
              from the device.");
7     }
8 }
```

Listing 5.1: C++ Code

Para analizar en detalle el comportamiento del sistema y verificar el funcionamiento de la función *ChangeLinkDelay*, se realizaron simulaciones en el escenario de red 5G desarrollado en la Sección 4.2, en el que se realizaron invocaciones a *ChangeLinkDelay*. Las mismas se hicieron de tal manera para que esta función altere el *delay* de un enlace cuello de botella, durante el segundo tercio del tiempo de simulación, incrementando el *delay* de este hasta alcanzar un valor máximo, en este caso 5 ms, para luego decrementar nuevamente el *delay* al valor original. Posteriormente, se analizó los *logs* generados por esta simulación y se validó que el comportamiento haya sido el deseado.

A partir de los *logs* de esta simulación se generaron tres gráficas que muestran diferentes aspectos del rendimiento de red a lo largo del tiempo de simulación. La Figura 5.3 muestra el *delay* de cada flujo a lo largo del tiempo de simulación. Se puede observar claramente cómo la función *ChangeLinkDelay* afecta el comportamiento de la red. Inicialmente, se ve un aumento gradual del *delay*, seguido por una disminución progresiva, lo cual verifica que la función está operando como se esperaba, modificando dinámicamente el *delay* del enlace. La Figura 5.4 presenta el *throughput* de cada flujo durante la simulación. Esta gráfica complementa la información del *delay*, mostrando cómo los cambios en el *delay* afectan la capacidad de transmisión de datos en la red. Por último, la Figura 5.5 ilustra el *jitter* de cada flujo a lo largo del tiempo. El *jitter*, que representa la variación en el *delay* de los paquetes, proporciona información valiosa sobre la estabilidad de la red durante los cambios introducidos por la función *ChangeLinkDelay*. En conjunto, estas tres figuras ofrecen una visión completa del comportamiento de la red bajo los cambios dinámicos de *delay* implementados. Demuestran la efectividad de la función *ChangeLinkDelay* y la capacidad del sistema para detectar y registrar estos cambios en diversos aspectos del rendimiento de red.

5.2.2. Experimento de cambio dinámico

Habiendo validado el funcionamiento de *ChangeLinkDelay*, se continuó con el objetivo de validar que el sistema de monitorización es capaz de detectar cambios dinámicos de *delay*. Para lograr esto, se decidió utilizar el escenario de los cuatro nodos ya establecido en la Sección 5.1.1, ya que ofrece un entorno de pruebas capaz de iterar rápidamente. Adicionalmente, se hizo uso de la capacidad de ns-3 para programar eventos dentro de la simulación, a través del objeto "*Simulator*" y el método "*Schedule*". En este caso en particular, se programó que el enlace entre los nodos `(n2,n3)` del escenario de la Sección 5.1.1, cambie de tener un *delay* de 10 ms a 2 ms luego de haber transcurrido 11 ms de la simulación. El momento en el que ocurre este cambio es de relevancia, debido a que se configuró el escenario para que los flujos de paquetes ocurrieran en momentos disjuntos de la simulación. Durante los primeros 10 ms de la simulación, la aplicación responsable de crear el primer flujo estará operativa y luego se detendrá. Luego, desde los 11 ms hasta los 21 ms de la simulación, otra aplicación, responsable de crear el segundo flujo de paquetes, comenzará a operar.

En caso de que el sistema funcione correctamente, deberá generar dos trazas para el enlace `(n2,n3)`. La primera detecta el *delay* inicial de 10 ms, y la segunda, el *delay* de 2 ms. Cabe remarcar que esto es de esta manera, porque la monitorización del escenario de la Sección 5.1.1 ocurre en dos momentos bien marcados, a los 10 ms y a los 20 ms.

5.2.3. Resultados

Se registró que el sistema reporta el cambio en el *delay* del enlace. Esto puede apreciarse en la Figura 5.6, donde el enlace `(n2,n3)` tiene inicialmente un *delay* 10 ms (1×10^6 ns) y luego en su segunda medición cambia a 2×10^6 ns. La Figura 5.6, es un extracto del archivo XML generado por la simulación. El archivo completo es el presentado a modo de ejemplo en la Sección 4.1.

5.3. Experimentación en escenarios realistas

Habiendo validado las funcionalidades del sistema en escenarios controlados, explicados en las secciones 5.1 y 5.2, el siguiente paso en la experimentación es incorporar el sistema de monitorización en los escenarios realistas desarrollados en la Sección 4.2. Para elaborar el escenario de prueba es importante que los nodos simulados fallen y sean representativos del caso de uso que se cubre. Estos son aquellos que un ISP [1] pueda medir y controlar la asignación de recursos sobre ellos. Esto representa un desafío, ya que las simulaciones desarrolladas hasta el momento son pequeñas, y los únicos *links* accesibles son de aire, presentes entre las *gNBs* y los *UEs*, y la modificación de los mismos no era representativo de un caso de uso realista.

[1] ISP es una empresa proveedora de Internet

5.3.1. Adaptación del Escenario

Para abordar la limitación de enlaces medibles en las simulaciones previas, que solo incluían conexiones inalámbricas entre *gNBs* y *UEs*, se introdujeron nodos adicionales entre el final del *5GCN*, particularmente el nodo *PGW* (responsable de disponibilizar la red a Internet) y el Internet en sí mismo, representado con el nodo "*remoteHost*". Con esto lo que se logró fue simular un nuevo nodo *PGW*, pero con el beneficio de lograr ganar control sobre los nodos adyacentes al final de la red, donde en un escenario real, un *ISP* puede controlar mejor los recursos.

Con el control de esta sección de la red, se puede simular cualquier arquitectura cuyo análisis permita determinar el funcionamiento de la red. En particular, se agregó una secuencia de nodos intermedios entre el nodo *PGW* original y el *remoteHost*. Los nodos agregados se puede visualizar en la Figura 5.7

En este escenario, los nodos *pgw*, *intermediate_node_1*, *intermediate_node_2* y *remoteHost* son elementos de la clase *Node* de ns-3. Dentro de esta clase, cada nodo tiene un identificador en la simulación (es posible obtenerlo con la función *GetId()*) el cual es usado también en los *logs* para registrar información asociada. En este escenario particular, se descubrió que los *IDs* de los nodos *pgw*, *intermediate_node_1*, *intermediate_node_2* y *remoteHost* son 33, 37, 38 y 36 respectivamente.

Luego se unieron estos nodos con *links* de tipo *point-to-point* y se les asignaron valores razonables a sus dispositivos de red correspondientes. En este caso, como se quería que la capacidad de procesamiento fuera despreciable en las mediciones, se les asignó un *DataRate* de 100 Gb/s. Además, se definieron valores de *delay* despreciables para dos de los tres enlaces entre estos nodos intermedios, con la excepción de uno, que fue seleccionado enlace cuello de botella, al cual se le asignó un *delay* inicial de 8 ms.

A continuación se repitió la misma estrategia de dinámicamente cambiar el *delay* de un enlace utilizando la función "*ChangeLinkDelay*", introducida en la Sección 5.2. En este caso, se aplicó para aumentar el *delay* de un enlace diferente al cuello de botella, causando que este deje de ser el *link* con el peor rendimiento, asignándole un *delay* de 9 ms, y por la configuración establecida, el siguiente *link* con peor rendimiento es aquel entre el nodo *PGW* y el primer nodo intermedio. Se encontró, en las trazas del sistema, el cambio del enlace con peor rendimiento, validando que el sistema funciona como se esperaba.

```
<network-measurements>
    <worst-links>
        <worst-link>
            <delay-value>8000118</delay-value>
            <timestamp>1400000000</timestamp>
            <node-pair>
                <node-id>37</node-id>
                <node-id>38</node-id>
            </node-pair>
        </worst-link>
```

```
        <worst-link>
            <delay-value>9000118</delay-value>
            <timestamp>2400000000</timestamp>
            <node-pair>
                <node-id>33</node-id>
                <node-id>37</node-id>
            </node-pair>
        </worst-link>
    </worst-links>
    <delays>
    <delay>
        <delay>
            <node-pair>
                <node-id>33</node-id>
                <node-id>37</node-id>
            </node-pair>
            <measurements>
                <measurement>
                    <delay-value>118</delay-value>
                    <timestamp>1400000000</timestamp>
                </measurement>
                <measurement>
                    <delay-value>9000118</delay-value>
                    <timestamp>2400000000</timestamp>
                </measurement>
            </measurements>
        </delay>
        <delay>
            <node-pair>
                <node-id>37</node-id>
                <node-id>38</node-id>
            </node-pair>
            <measurements>
                <measurement>
                    <delay-value>8000118</delay-value>
                    <timestamp>1400000000</timestamp>
                </measurement>
                <measurement>
                    <delay-value>8000118</delay-value>
                    <timestamp>2400000000</timestamp>
                </measurement>
            </measurements>
        </delay>
    </delays>
```

Listing 5.2: Recorte de uno de los *xml logs* del sistema generado por haber ejecutado la nueva adaptación del escenario presentado en Sección 4.2. En las etiquetas de *delay* se puede apreciar como el enlace 37-38 (*intermediate_node_1*, *intermediate_node_2*) se mantiene siempre con un *delay* de 8 ms, mientras que el enlace 33-37 (*pgw*, *intermediate_nodes_1*), parte de un *delay* despreciable de 118 ns, para un segundo después dispararse a 9 ms en la segunda medición.

El *xml log* obtenido en el Listado 5.2, es producto de dos mediciones realizadas en dos momentos de la simulación controlados: antes de realizar un cambio de *delay* en un enlace, a los 1.4 segundos comenzada la simulación y luego de este, a los 2.4 segundos comenzada la simulación. Las mediciones se realizaron con la función *nodeToNodeTrigger*, es decir, los *logs* reflejan mediciones nodo a nodo (explicado en la Sección 4.3.2). En las mediciones se evidencia el cambio del enlace con peor rendimiento, partiendo del enlace (37,38) o de los nodos (*intermediate_node_1*, *intermediate_node_2*), hacia los nodos (33,37) o los nodos (*pgw*, *intermediate_nodes_1*).

Con este experimento se da fin a la experimentación de la funcionalidad de detección del enlace con peor rendimiento en una arquitectura, validando la funcionalidad de localización de fallas del sistema de monitorización.

5.4. Prueba de correcta detección y pruebas de estrés

Medir la correctitud del sistema de monitorización probó ser un desafío, debido a que la correctitud del sistema depende directamente de cuan bien refleja la simulación un escenario realista de redes 5G. Para sobrepasar este desafío, se decidió establecer que el sistema de monitorización estaría funcionando correctamente si no se activa durante ningún flujo de paquetes estándar del escenario (presentado en la Sección 4.2) en sí, pero si se activa cuando proactivamente se busca empeorar el rendimiento de la misma por medio del uso de la función "*ChangeLinkDelay*".

Con esta idea en mente, se implementó la siguiente manera de medir la correctitud del sistema de monitorización. Cada vez que se invoca a la función "*ChangeLinkDelay*" (explicada en la Sección 5.1), el intervalo de tiempo que altere el *delay* inicial del *link*, será persistido en una variable. La misma será consultada cada vez que se invoque la función "*reportFlowStats*", y se considerará que el sistema fue correctamente invocado, si al momento de su invocación, la red se encuentra siendo activamente empeorada por la función "*ChangeLinkDelay*", y en caso contrario, el sistema habrá fallado si es invocado en un momento que la red se encuentra operando con normalidad.

Utilizar esta métrica de correctitud, permitió investigar la susceptibilidad del sistema de monitorización a reportar fluctuaciones en las métricas esperables de una red en funcionamiento normal como también, a omitir picos en las métricas

UEs	gNBs	TP	FP	FN	TN	PR	RE	CA	F1
30	3	21	1	0	17	0.954545	1	0.678571	0.971429
60	3	19	1	1	17	0.95	0.95	0.666667	0.944444
90	3	19	1	1	17	0.95	0.95	0.666667	0.944444
120	3	19	15	1	10	0.558824	0.95	0.537037	0.555556
210	21	13	5	4	15	0.722222	0.764706	0.583333	0.769231

que pueden ser indicio de un enlace fallando en una red, por tratarlos como una fluctuación más.

5.4.1. Pruebas de estrés

En la Tabla 5.4.1, se presenta la experimentación realizada siguiendo el criterio establecido en la Sección 5.4, sobre un caso de estudio. El mismo, fue el modelado de una red 5G que presenta únicamente tráfico NGMN_VIDEO DL y con paquetes UDP. Para la misma, se fue incrementalmente aumentando la cantidad de *ues* y de *gNBs*, para estudiar el comportamiento del sistema de monitorización sobre casos cada vez más complejos. Los indicadores tomados en cuenta fueron los siguientes:

- TP: True Positive = Activación del sistema correctamente
- FP: False Positive = Activación del sistema incorrectamente
- FN: False Negative = No Activación del sistema incorrectamente
- TN: True Negative = No activación del sistema correctamente
- PR: Precision = $\frac{TP}{TP+FP}$
- RE: Recall = $\frac{TP}{TP+FN}$
- CA: Accuracy = $\frac{TP+TN}{TP+TN+FN+FP}$
- F1: $\frac{2*TP}{2*TP+FN+FP}$

Durante esta etapa de la experimentación, el tiempo requerido de las simulaciones fue creciendo exponencialmente, donde las primeras simulaciones con pocos nodos tomaron alrededor de 15 minutos, mientras que las últimas comenzaron a tomar cerca de 25 horas.

Las conclusiones de este experimento fueron que el sistema se desempeñó satisfactoriamente y las métricas extraídas lo reflejan. Se entiende que el éxito del experimento se debe en medida a que el diseño del experimento favorece el funcionamiento del sistema de monitorización. Con esto se refiere a que, debido a que los enlaces donde se instala el *bottleNeck* se ven progresivamente afectados cada vez introduciendo más *delay*, resulta lógico que el sistema comience a tomar mediciones más frecuentemente debido a que ocurrieron en un momento que la

red se veía proactivamente empeorada. Cada una de estas será clasificada como TP, obteniendo así una alta tasa de valores TP y baja de FP, lo cual mejora sustancialmente el rendimiento del sistema.

5.5. Validación de heurística de medición

Como se explicó en la Sección 4.5, la heurística que se decidió utilizar para este sistema de monitorización consiste en tomar la primera medición como el umbral a ser utilizado para el resto de las mediciones. De esta manera, dejando la responsabilidad de asegurar que la primera medición ocurra durante un periodo de rendimiento óptimo de la red al usuario del sistema, así definiendo el umbral que determinará el comportamiento aceptable del sistema de monitorización.

Esta heurística fue validada por medio de experimentación. La misma consistió en evaluar si la primera medición puede tomar valores representativos de lo que se puede esperar del rendimiento"promedio" de una red.

5.5.1. Generación de un *dataset*

Como al realizar este experimento, no se contaba con ningún tipo de *dataset* que se adecue al contexto de trabajo del proyecto (simulaciones en ns-3 con 5G-LENA de un escenario particular, explicado en la Sección 4.2), fue necesario elaborar uno propio.

Para esto, lo que se realizó fue configurar el escenario para que no se introduzca ningún cuello de botella, ni que el sistema de monitorización realice ningún tipo de acción. Es decir, se configuró el escenario para que se desempeñe de forma óptima, y se enmascaró la capacidad de adaptación del sistema de monitorización, realizando en su lugar monitorizaciones pasivas cada un segundo durante un periodo de un minuto. De esta manera, lo que se obtuvo fue un escenario de simulación que se tratará como la realidad contra la cual comparar.

Para generar propiamente el *dataset* se creó un nuevo estándar de archivos con las mediciones de las simulaciones. El formato es CSV, y consiste en una fila por cada medición realizada por el sistema, que en este caso equivale a cada invocación que se realice sobre "reportFlowStats" (explicado en la Sección 4.3.2). Cada fila contiene información de las métricas *delay*, *throughput* y *jitter* de cada flujo detectado en la simulación, así también como un promedio de estas métricas a través de todos los flujos. En la Figura 5.8 se muestra este formato.

A continuación se ejecutaron un conjunto simulaciones con diferentes parámetros y las mediciones realizadas sobre las mismas, se persistieron utilizando este nuevo formato de archivo. Esto se realizó con el objetivo de generar un conjunto de datos completo que sirva para tener una referencia de los valores que las métricas *delay*, *throughput* y *jitter* pueden tomar bajo diferentes perfiles de tráfico . En particular, se realizó una configuración paramétrica entre los siguientes valores:

- Tipos de tráfico: *NGMN_MIXED*, *NGMN_VOIP*,*NGMN_VIDEO*, *NGMN_GAMING*, *UDP_CBR* y *FTP_3GPP_M1* (explicados en la tabla 4.1).

- Protocolos de transporte: *UDP* y *TCP*.
- Cantidad de *gNBs*: 3.
- Cantidad de *ues* por *gNBs*: 10 y 20.

Las mediciones obtenidas a partir de estas simulaciones son parte de los resultados del proyecto, ya son un producto en sí mismo. Estas podrán ser utilizadas por otros equipos de investigación para hacer análisis de datos sin la necesidad de tener que superar la curva de aprendizaje que representa ns-3.

Para procesar estos datos se construyó un *pipeline* que permitió, a partir de cada simulación, calcular el promedio, varianza y desviación estándar de cada métrica para cada medición tomada a lo largo de la simulación. Adicionalmente, se encuentran los valores mínimos y máximos de cada métrica para cada simulación, así como las primeras mediciones de estas métricas. Estas estadísticas se pueden apreciar en las siguientes tablas 5.1, 5.2 y 5.3.

Continuando el procesamiento de los datos obtenidos, se midió el error relativo de la primera medición de cada métrica, contra el promedio de las mismas, luego de terminada la simulación, obteniendo la tabla de valores (tabla 5.4). La Tabla 5.4, muestra las diferencias entre las mediciones iniciales y los promedios de varias métricas de red (*delay*, *throughput* y *jitter*) en diversos escenarios de tráfico en una red 5G, tras un minuto de simulación.

Lo esperado era que las mediciones iniciales aproximen a los promedios, sin embargo, se encontró que según el tipo de tráfico, se podían observar variaciones significativas.

5.5.2. Análisis por Tipo de Tráfico

FTP (File Transfer Protocol)

El tráfico FTP muestra diferencias sustanciales:

```
throughput_rel_error = 1,14 · 10^-2
```

Este valor elevado es atribuible a:

- La sobrecarga inicial para establecer la conexión TCP.
- Una velocidad de transferencia inicial más alta debido al algoritmo de control de congestión de TCP (*slow start*).
- La tendencia a la estabilización conforme avanza la transferencia.

Gaming

Para el tráfico de juegos, se observan valores relativamente bajos:

```
delay_rel_error = 0,1 ,
throughput_rel_error = 1,25 . 10^2
```

Traffic Type	Direction	UseUDP	UEs/Gnb	FirstAvgDelay	AvgDelayMin	AvgDelayMax	AvgDelayMean	AvgDelayVar	AvgDelayStddev
NGMN FTP	DL	false	20	1	1	3,65	2,75	0,94	0,97
NGMN GAMING	DL	true	20	1,7	1,7	1,92	1,87	$1{,}82 \cdot 10^{-3}$	$4{,}26 \cdot 10^{-2}$
FTP 3GPP-M1	UL	false	10	2,39	0,92	3,72	2,29	0,32	0,57
FTP 3GPP-M1	UL	true	20	1,92	0,56	2,56	1,28	0,14	0,37
NGMN GAMING	DL	false	10	0,81	0,81	1,84	1,02	0,16	0,4
NGMN VIDEO	DL	false	10	0,89	0,89	1,84	1,66	0,14	0,38
NGMN FTP	DL	true	10	12,55	0,1	12,55	0,85	8,53	2,92
NGMN VIDEO	DL	false	20	1	1	3,65	2,75	0,94	0,97
UDP CBR	DL	false	20	10,37	10,06	11,23	10,82	$7{,}35 \cdot 10^{-2}$	0,27
FTP 3GPP-M1	DL	true	20	1,92	0,56	2,56	1,28	0,14	0,37
NGMN MIXED	DL	false	10	0,75	0,75	1,44	1,08	$4{,}88 \cdot 10^{-2}$	0,22
NGMN VOIP	DL	true	20	1,81	1,53	1,81	1,64	$3{,}5 \cdot 10^{-3}$	$5{,}92 \cdot 10^{-2}$
NGMN FTP	DL	true	20	19,26	$7{,}71 \cdot 10^{-2}$	19,26	1,15	20,41	4,52
NGMN FTP	DL	false	10	0,89	0,89	1,84	1,66	0,14	0,38
FTP 3GPP-M1	DL	false	20	1,92	0,56	2,56	1,28	0,14	0,37
NGMN VOIP	DL	false	10	0,89	0,89	1,84	1,66	0,14	0,38
UDP CBR	DL	true	10	15,1	14,89	16,34	16,11	0,12	0,34
NGMN MIXED	DL	true	10	1,5	1,08	1,5	1,18	$2{,}68 \cdot 10^{-3}$	$5{,}18 \cdot 10^{-2}$
UDP CBR	DL	true	20	10,37	10,06	11,23	10,82	$7{,}35 \cdot 10^{-2}$	0,27
NGMN VIDEO	DL	true	20	1,45	1,39	1,45	1,41	$6{,}82 \cdot 10^{-5}$	$8{,}26 \cdot 10^{-3}$
UDP CBR	DL	false	10	15,1	14,89	16,34	16,11	0,12	0,34
NGMN VOIP	DL	false	20	1	1	3,65	2,75	0,94	0,97
FTP 3GPP-M1	DL	true	10	2,39	0,92	3,72	2,29	0,32	0,57
NGMN VIDEO	DL	true	10	1,37	1,33	1,39	1,35	$1{,}07 \cdot 10^{-4}$	$1{,}04 \cdot 10^{-2}$
NGMN GAMING	DL	false	20	0,83	0,83	2,65	1,19	0,51	0,71
FTP 3GPP-M1	DL	false	10	2,39	0,92	3,72	2,29	0,32	0,57
FTP 3GPP-M1	UL	false	20	1,92	0,56	2,56	1,28	0,14	0,37
NGMN MIXED	DL	true	20	1,88	1,11	1,88	1,17	$9{,}36 \cdot 10^{-3}$	$9{,}68 \cdot 10^{-2}$
NGMN MIXED	DL	false	20	0,79	0,79	2,12	1,69	0,21	0,46
NGMN GAMING	DL	true	10	1,73	1,73	1,89	1,85	$6{,}57 \cdot 10^{-4}$	$2{,}56 \cdot 10^{-2}$
NGMN VOIP	DL	true	10	1,73	1,56	1,79	1,73	$2{,}15 \cdot 10^{-3}$	$4{,}64 \cdot 10^{-2}$
FTP 3GPP-M1	UL	true	10	2,39	0,92	3,72	2,29	0,32	0,57

Tabla 5.1: Promedio de Delay, medido en milisegundos (ms), en un minuto de simulación

Esto es consistente con las características del tráfico de juegos:

- Flujo de datos más constante y requisitos de baja latencia.
- Pequeñas variaciones, posiblemente debidas a eventos específicos del juego.

Video Streaming

El *streaming* de video presenta valores bajos de error relativo cuando se utiliza con conexiones TCP:

```
delay_rel_error = 2,7 · 10^2,
throughput_rel_error = 5,04 · 10^2
```

Esto es coherente con las características del *streaming* de video:

- Tráfico constante una vez establecido el buffer inicial.

Traffic Type	Direction	UseUDP	UEs/Gnb	FirstAvgThroughput	AvgThroughputMin	AvgThroughputMax	AvgThroughputMean	AvgThroughputVar	AvgThroughputStddev
NGMN FTP	DL	false	20	$2{,}2 \cdot 10^{-4}$	$2{,}05 \cdot 10^{-4}$	$2{,}2 \cdot 10^{-4}$	$2{,}18 \cdot 10^{-4}$	$3{,}75 \cdot 10^{-11}$	$6{,}12 \cdot 10^{-6}$
NGMN GAMING	DL	true	20	$2{,}74 \cdot 10^{-2}$	$1{,}64 \cdot 10^{-2}$	$2{,}76 \cdot 10^{-2}$	$2{,}7 \cdot 10^{-2}$	$2 \cdot 10^{-6}$	$1{,}41 \cdot 10^{-3}$
FTP 3GPP-M1	UL	false	10	0,1	$1{,}51 \cdot 10^{-2}$	0,1	$3{,}87 \cdot 10^{-2}$	$1{,}87 \cdot 10^{-4}$	$1{,}37 \cdot 10^{-2}$
FTP 3GPP-M1	UL	true	20	$9{,}97 \cdot 10^{-2}$	$8{,}95 \cdot 10^{-3}$	$9{,}97 \cdot 10^{-2}$	$2{,}16 \cdot 10^{-2}$	$1{,}37 \cdot 10^{-4}$	$1{,}17 \cdot 10^{-2}$
NGMN GAMING	DL	false	10	$2{,}17 \cdot 10^{-4}$	$2{,}01 \cdot 10^{-4}$	$2{,}17 \cdot 10^{-4}$	$2{,}14 \cdot 10^{-4}$	$4{,}27 \cdot 10^{-11}$	$6{,}53 \cdot 10^{-6}$
NGMN VIDEO	DL	false	10	$2{,}17 \cdot 10^{-4}$	$2{,}01 \cdot 10^{-4}$	$2{,}17 \cdot 10^{-4}$	$2{,}14 \cdot 10^{-4}$	$4{,}27 \cdot 10^{-11}$	$6{,}53 \cdot 10^{-6}$
NGMN FTP	DL	true	10	0,16	$2{,}58 \cdot 10^{-3}$	0,16	$1{,}21 \cdot 10^{-2}$	$1{,}46 \cdot 10^{-3}$	$3{,}82 \cdot 10^{-2}$
NGMN VIDEO	DL	false	20	$2{,}2 \cdot 10^{-4}$	$2{,}05 \cdot 10^{-4}$	$2{,}2 \cdot 10^{-4}$	$2{,}18 \cdot 10^{-4}$	$3{,}75 \cdot 10^{-11}$	$6{,}12 \cdot 10^{-6}$
UDP CBR	DL	false	20	3,24	1,91	3,39	3,15	$3{,}08 \cdot 10^{-2}$	0,18
FTP 3GPP-M1	DL	true	20	$9{,}97 \cdot 10^{-2}$	$8{,}95 \cdot 10^{-3}$	$9{,}97 \cdot 10^{-2}$	$2{,}16 \cdot 10^{-2}$	$1{,}37 \cdot 10^{-4}$	$1{,}17 \cdot 10^{-2}$
NGMN MIXED	DL	false	10	$1{,}92 \cdot 10^{-4}$	$1{,}78 \cdot 10^{-4}$	$1{,}92 \cdot 10^{-4}$	$1{,}9 \cdot 10^{-4}$	$3{,}27 \cdot 10^{-11}$	$5{,}72 \cdot 10^{-6}$
NGMN VOIP	DL	true	20	$5{,}71 \cdot 10^{-3}$	$5{,}71 \cdot 10^{-3}$	$1{,}3 \cdot 10^{-2}$	$1{,}04 \cdot 10^{-2}$	$1{,}59 \cdot 10^{-6}$	$1{,}26 \cdot 10^{-3}$
NGMN FTP	DL	true	20	0,11	$1{,}33 \cdot 10^{-3}$	0,11	$7{,}69 \cdot 10^{-3}$	$6{,}93 \cdot 10^{-4}$	$2{,}63 \cdot 10^{-2}$
NGMN FTP	DL	false	10	$2{,}17 \cdot 10^{-4}$	$2{,}01 \cdot 10^{-4}$	$2{,}17 \cdot 10^{-4}$	$2{,}14 \cdot 10^{-4}$	$4{,}27 \cdot 10^{-11}$	$6{,}53 \cdot 10^{-6}$
FTP 3GPP-M1	DL	false	20	$9{,}97 \cdot 10^{-2}$	$8{,}95 \cdot 10^{-3}$	$9{,}97 \cdot 10^{-2}$	$2{,}16 \cdot 10^{-2}$	$1{,}37 \cdot 10^{-4}$	$1{,}17 \cdot 10^{-2}$
NGMN VOIP	DL	false	10	$2{,}17 \cdot 10^{-4}$	$2{,}01 \cdot 10^{-4}$	$2{,}17 \cdot 10^{-4}$	$2{,}14 \cdot 10^{-4}$	$4{,}27 \cdot 10^{-11}$	$6{,}53 \cdot 10^{-6}$
UDP CBR	DL	true	10	6,04	3,58	6,15	5,85	$9{,}34 \cdot 10^{-2}$	0,31
NGMN MIXED	DL	true	10	$5{,}91 \cdot 10^{-2}$	$1{,}17 \cdot 10^{-2}$	$5{,}91 \cdot 10^{-2}$	$1{,}98 \cdot 10^{-2}$	$2{,}85 \cdot 10^{-5}$	$5{,}34 \cdot 10^{-3}$
UDP CBR	DL	true	20	3,24	1,91	3,39	3,15	$3{,}08 \cdot 10^{-2}$	0,18
NGMN VIDEO	DL	true	20	$5{,}9 \cdot 10^{-2}$	$3{,}41 \cdot 10^{-2}$	$5{,}9 \cdot 10^{-2}$	$5{,}61 \cdot 10^{-2}$	$8{,}79 \cdot 10^{-6}$	$2{,}96 \cdot 10^{-3}$
UDP CBR	DL	false	10	6,04	3,58	6,15	5,85	$9{,}34 \cdot 10^{-2}$	0,31
NGMN VOIP	DL	false	20	$2{,}2 \cdot 10^{-4}$	$2{,}05 \cdot 10^{-4}$	$2{,}2 \cdot 10^{-4}$	$2{,}18 \cdot 10^{-4}$	$3{,}75 \cdot 10^{-11}$	$6{,}12 \cdot 10^{-6}$
FTP 3GPP-M1	DL	true	10	0,1	$1{,}51 \cdot 10^{-2}$	0,1	$3{,}87 \cdot 10^{-2}$	$1{,}87 \cdot 10^{-4}$	$1{,}37 \cdot 10^{-2}$
NGMN VIDEO	DL	true	10	$5{,}71 \cdot 10^{-2}$	$3{,}24 \cdot 10^{-2}$	$5{,}71 \cdot 10^{-2}$	$5{,}43 \cdot 10^{-2}$	$9{,}04 \cdot 10^{-6}$	$3{,}01 \cdot 10^{-3}$
NGMN GAMING	DL	false	20	$2{,}2 \cdot 10^{-4}$	$2{,}05 \cdot 10^{-4}$	$2{,}2 \cdot 10^{-4}$	$2{,}18 \cdot 10^{-4}$	$3{,}75 \cdot 10^{-11}$	$6{,}12 \cdot 10^{-6}$
FTP 3GPP-M1	DL	false	10	0,1	$1{,}51 \cdot 10^{-2}$	0,1	$3{,}87 \cdot 10^{-2}$	$1{,}87 \cdot 10^{-4}$	$1{,}37 \cdot 10^{-2}$
FTP 3GPP-M1	UL	false	20	$9{,}97 \cdot 10^{-2}$	$8{,}95 \cdot 10^{-3}$	$9{,}97 \cdot 10^{-2}$	$2{,}16 \cdot 10^{-2}$	$1{,}37 \cdot 10^{-4}$	$1{,}17 \cdot 10^{-2}$
NGMN MIXED	DL	true	20	$6{,}13 \cdot 10^{-2}$	$1{,}16 \cdot 10^{-2}$	$6{,}13 \cdot 10^{-2}$	$2{,}04 \cdot 10^{-2}$	$3{,}03 \cdot 10^{-5}$	$5{,}51 \cdot 10^{-3}$
NGMN MIXED	DL	false	20	$1{,}95 \cdot 10^{-4}$	$1{,}82 \cdot 10^{-4}$	$1{,}95 \cdot 10^{-4}$	$1{,}93 \cdot 10^{-4}$	$2{,}82 \cdot 10^{-11}$	$5{,}31 \cdot 10^{-6}$
NGMN GAMING	DL	true	10	$2{,}62 \cdot 10^{-2}$	$1{,}58 \cdot 10^{-2}$	$2{,}73 \cdot 10^{-2}$	$2{,}62 \cdot 10^{-2}$	$2{,}01 \cdot 10^{-6}$	$1{,}42 \cdot 10^{-3}$
NGMN VOIP	DL	true	10	$3{,}9 \cdot 10^{-3}$	$3{,}9 \cdot 10^{-3}$	$1{,}38 \cdot 10^{-2}$	$1{,}03 \cdot 10^{-2}$	$3{,}23 \cdot 10^{-6}$	$1{,}8 \cdot 10^{-3}$
FTP 3GPP-M1	UL	true	10	0,1	$1{,}51 \cdot 10^{-2}$	0,1	$3{,}87 \cdot 10^{-2}$	$1{,}87 \cdot 10^{-4}$	$1{,}37 \cdot 10^{-2}$

Tabla 5.2: Promedio de Throughput, medido en megabytes por segundo (mb/s), en un minuto de simulación

- Variaciones menores debido a posibles cambios en la calidad del video o *rebuffering*.

VoIP

Para el tráfico VoIP, se observa:

```
delay_rel_error = 9,62 · 10^2,
throughput_rel_error = 0,83
```

El bajo error relativo en el *delay* es esperado para VoIP debido a sus requisitos de baja latencia. El alto error negativo en el *throughput* podría indicar:

- Una fase inicial de negociación o establecimiento de llamada con mayor tráfico.

- Posibles períodos de silencio o pausas en la conversación que reducen el *throughput* promedio con el tiempo.

Traffic Type	Direction	UseUDP	UEs/Gnb	FirstAvgJitter	AvgJitterMin	AvgJitterMax	AvgJitterMean	AvgJitterVar	AvgJitterStddev
NGMN FTP	DL	false	20	0	0	0	0	0	0
NGMN GAMING	DL	true	20	$4{,}92 \cdot 10^{-2}$	$1{,}48 \cdot 10^{-2}$	$5{,}08 \cdot 10^{-2}$	$3{,}02 \cdot 10^{-2}$	$5{,}72 \cdot 10^{-5}$	$7{,}56 \cdot 10^{-3}$
FTP 3GPP-M1	UL	false	10	$7{,}29 \cdot 10^{-2}$	$7{,}15 \cdot 10^{-3}$	$7{,}67 \cdot 10^{-2}$	$2{,}96 \cdot 10^{-2}$	$1{,}84 \cdot 10^{-4}$	$1{,}36 \cdot 10^{-2}$
FTP 3GPP-M1	UL	true	20	$9{,}27 \cdot 10^{-2}$	$4{,}76 \cdot 10^{-3}$	$9{,}27 \cdot 10^{-2}$	$1{,}53 \cdot 10^{-2}$	$1{,}31 \cdot 10^{-4}$	$1{,}14 \cdot 10^{-2}$
NGMN GAMING	DL	false	10	0	0	0	0	0	0
NGMN VIDEO	DL	false	10	0	0	0	0	0	0
NGMN FTP	DL	true	10	1,37	$1{,}12 \cdot 10^{-3}$	1,37	$7{,}82 \cdot 10^{-2}$	0,1	0,32
NGMN VIDEO	DL	false	20	0	0	0	0	0	0
UDP CBR	DL	false	20	0,44	0,4	0,53	0,48	$1{,}06 \cdot 10^{-3}$	$3{,}25 \cdot 10^{-2}$
FTP 3GPP-M1	DL	true	20	$9{,}27 \cdot 10^{-2}$	$4{,}76 \cdot 10^{-3}$	$9{,}27 \cdot 10^{-2}$	$1{,}53 \cdot 10^{-2}$	$1{,}31 \cdot 10^{-4}$	$1{,}14 \cdot 10^{-2}$
NGMN MIXED	DL	false	10	0	0	0	0	0	0
NGMN VOIP	DL	true	20	0,13	0,12	0,22	0,18	$4{,}2 \cdot 10^{-4}$	$2{,}05 \cdot 10^{-2}$
NGMN FTP	DL	true	20	1,48	$5{,}76 \cdot 10^{-4}$	1,48	$8{,}33 \cdot 10^{-2}$	0,12	0,35
NGMN FTP	DL	false	10	0	0	0	0	0	0
FTP 3GPP-M1	DL	false	20	$9{,}27 \cdot 10^{-2}$	$4{,}76 \cdot 10^{-3}$	$9{,}27 \cdot 10^{-2}$	$1{,}53 \cdot 10^{-2}$	$1{,}31 \cdot 10^{-4}$	$1{,}14 \cdot 10^{-2}$
NGMN VOIP	DL	false	10	0	0	0	0	0	0
UDP CBR	DL	true	10	0,69	0,64	0,8	0,78	$1{,}39 \cdot 10^{-3}$	$3{,}73 \cdot 10^{-2}$
NGMN MIXED	DL	true	10	0,11	$6{,}07 \cdot 10^{-2}$	0,12	$8{,}8 \cdot 10^{-2}$	$2{,}01 \cdot 10^{-4}$	$1{,}42 \cdot 10^{-2}$
UDP CBR	DL	true	20	0,44	0,4	0,53	0,48	$1{,}06 \cdot 10^{-3}$	$3{,}25 \cdot 10^{-2}$
NGMN VIDEO	DL	true	20	0,18	0,14	0,18	0,15	$6{,}47 \cdot 10^{-5}$	$8{,}04 \cdot 10^{-3}$
UDP CBR	DL	false	10	0,69	0,64	0,8	0,78	$1{,}39 \cdot 10^{-3}$	$3{,}73 \cdot 10^{-2}$
NGMN VOIP	DL	false	20	0	0	0	0	0	0
FTP 3GPP-M1	DL	true	10	$7{,}29 \cdot 10^{-2}$	$7{,}15 \cdot 10^{-3}$	$7{,}67 \cdot 10^{-2}$	$2{,}96 \cdot 10^{-2}$	$1{,}84 \cdot 10^{-4}$	$1{,}36 \cdot 10^{-2}$
NGMN VIDEO	DL	true	10	0,15	0,12	0,16	0,14	$8{,}09 \cdot 10^{-5}$	$9 \cdot 10^{-3}$
NGMN GAMING	DL	false	20	0	0	0	0	0	0
FTP 3GPP-M1	DL	false	10	$7{,}29 \cdot 10^{-2}$	$7{,}15 \cdot 10^{-3}$	$7{,}67 \cdot 10^{-2}$	$2{,}96 \cdot 10^{-2}$	$1{,}84 \cdot 10^{-4}$	$1{,}36 \cdot 10^{-2}$
FTP 3GPP-M1	UL	false	20	$9{,}27 \cdot 10^{-2}$	$4{,}76 \cdot 10^{-3}$	$9{,}27 \cdot 10^{-2}$	$1{,}53 \cdot 10^{-2}$	$1{,}31 \cdot 10^{-4}$	$1{,}14 \cdot 10^{-2}$
NGMN MIXED	DL	true	20	0,12	$6{,}15 \cdot 10^{-2}$	0,12	$9{,}22 \cdot 10^{-2}$	$1{,}35 \cdot 10^{-4}$	$1{,}16 \cdot 10^{-2}$
NGMN MIXED	DL	false	20	0	0	0	0	0	0
NGMN GAMING	DL	true	10	$2{,}26 \cdot 10^{-2}$	$6{,}09 \cdot 10^{-3}$	$5{,}68 \cdot 10^{-2}$	$2{,}51 \cdot 10^{-2}$	$1{,}27 \cdot 10^{-4}$	$1{,}13 \cdot 10^{-2}$
NGMN VOIP	DL	true	10	$9{,}41 \cdot 10^{-2}$	$9{,}41 \cdot 10^{-2}$	0,25	0,17	$9{,}75 \cdot 10^{-4}$	$3{,}12 \cdot 10^{-2}$
FTP 3GPP-M1	UL	true	10	$7{,}29 \cdot 10^{-2}$	$7{,}15 \cdot 10^{-3}$	$7{,}67 \cdot 10^{-2}$	$2{,}96 \cdot 10^{-2}$	$1{,}84 \cdot 10^{-4}$	$1{,}36 \cdot 10^{-2}$

Tabla 5.3: Promedio de Jitter, medido en milisegundos (ms), en un minuto de simulación

UDP CBR (Constant Bit Rate)

El tráfico UDP *CBR* muestra valores bajos de error relativo:

```
delay_rel_error = 4,4 · 10^2 ,
throughput_rel_error = 2,52 · 10^2
```

Estos valores son esperados para *CBR*, dado que este tipo de tráfico está diseñado para mantener una tasa de bits constante.

5.5.3. Resumen

En base a los resultados obtenidos, se puede concluir que:

- Los protocolos de tiempo real como Gaming, VoIP y Video Streaming tienden a mostrar menor variación entre la medición inicial y el promedio del *delay*.

- Protocolos basados en transferencia de archivos, como FTP, exhiben mayor variación debido a las fases iniciales de la transferencia. Esto tiene sentido, ya que el primer segundo probablemente captura solo la fase inicial de establecimiento de conexión y el comienzo de la transferencia.
- La dirección del tráfico (UL/DL) y la cantidad de *UEs* por *gNB* también afectan las mediciones, aunque en menor medida que el tipo de tráfico.

Esto demuestra que utilizar un umbral de métricas *delay*, *throughput* y *jitter* para cualquier tipo de tráfico es una estrategia fallida. El perfil de tráfico influye significativamente en la veracidad de estas métricas para reportar el nivel de congestión de red. Particularmente en los perfiles de tráfico que no son flujo constante de paquetes (como por ejemplo FTP), se deberá explorar otras métricas.

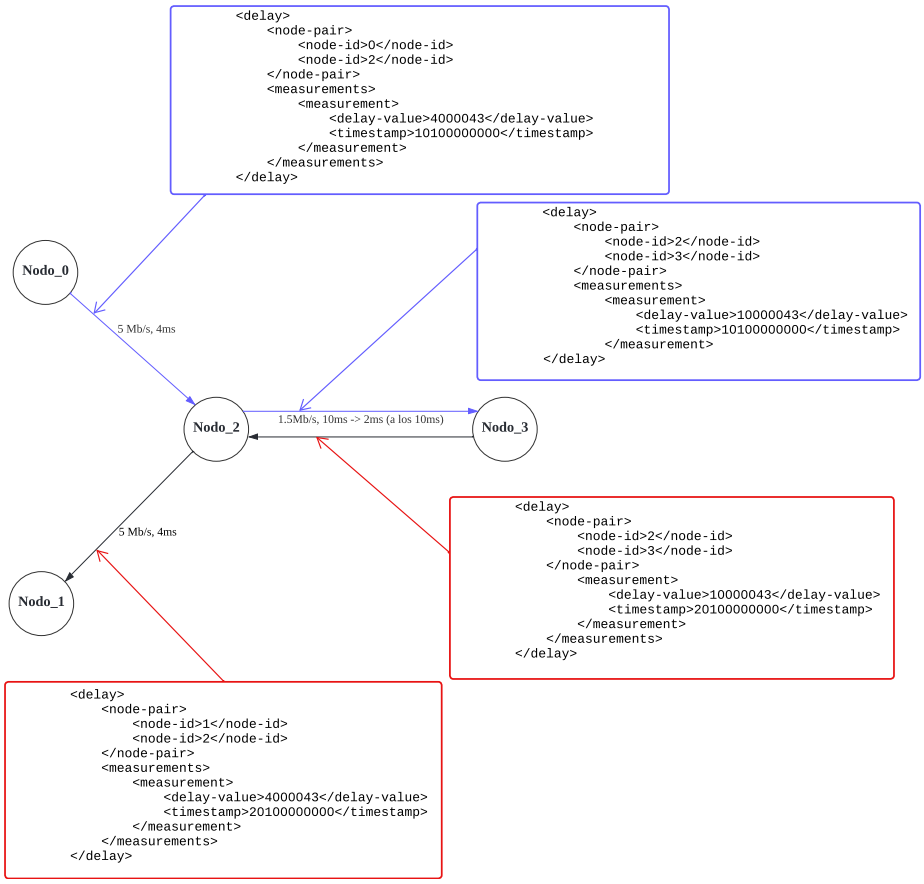

Figura 5.2: Diagrama de la topología del escenario 5.1.1, con una asociación entre las trazas generadas y paquetes enviados a través de enlaces. En azul las correspondientes al flujo 1 entre el nodo 0 y el nodo 3, y en rojo las correspondientes al flujo 2 entre el nodo 3 y el nodo 1.

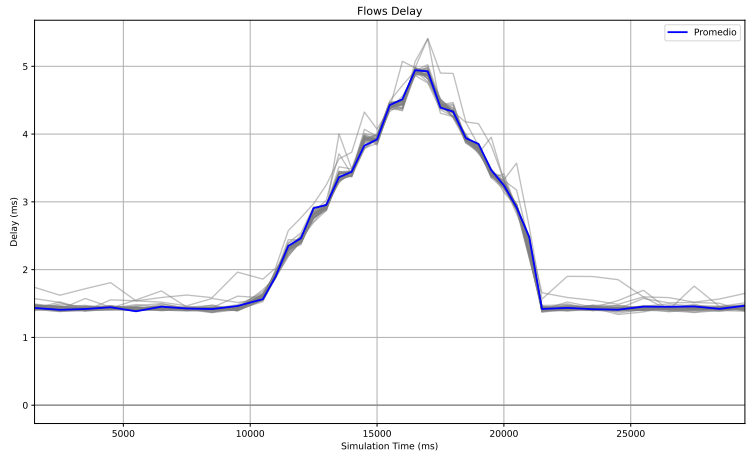

Figura 5.3: En esta gráfica se muestra el *delay* de cada flujo, medido en milisegundos (ms), en función del tiempo de simulación, medido también en milisegundos (ms). El *delay* de los flujos individuales se muestra en gris y el *delay* promedio de todos los flujos se indica con una línea azul.

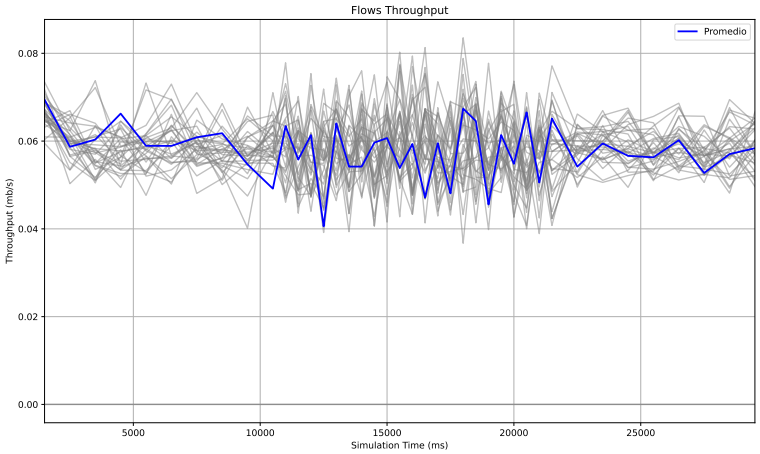

Figura 5.4: En esta gráfica se muestra el *throughput* de cada flujo, medido en megabytes por segundo (mb/s), en función del tiempo de simulación, medido en milisegundos (ms). El *throughput* de los flujos individuales se muestra en gris y el *throughput* promedio de todos los flujos se indica con una línea azul.

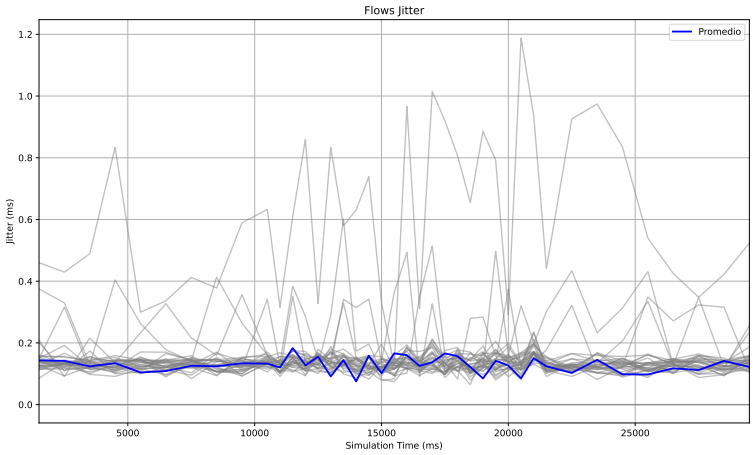

Figura 5.5: En esta gráfica se muestra el *jitter* de cada flujo, medido en milisegundos (ms), en función del tiempo de simulación, medido también en milisegundos (ms). El *jitter* de los flujos individuales se muestra en gris y el *jitter* promedio de todos los flujos se indica con una línea azul.

```
<measurements>
    <measurement>
        <delay-value>10000043</delay-value>
        <timestamp>10100000000</timestamp>
    </measurement>
    <measurement>
        <delay-value>2000043</delay-value>
        <timestamp>20100000000</timestamp>
    </measurement>
</measurements>
```

Figura 5.6: Extracto del archivo XML de la Figura 4.1, generado por la simulación. Mediciones del enlace `(n2,n3) en el escenario` 5.1.1. En azul se indica el *delay* inicial y en rojo el *delay* actualizado.

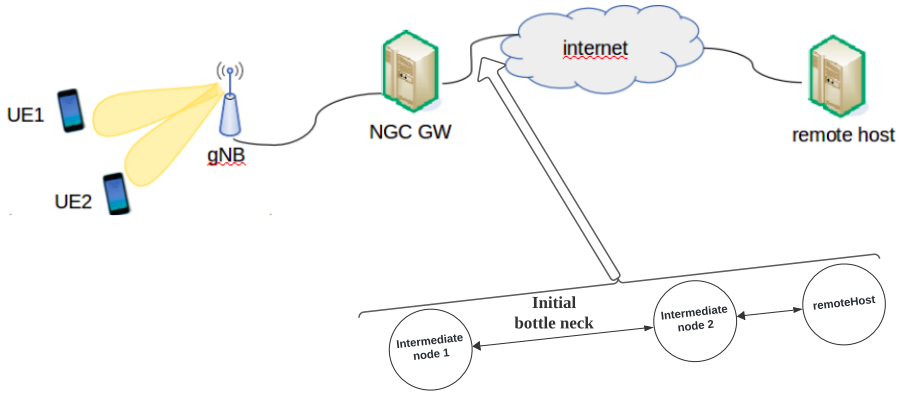

Figura 5.7: Diagrama de arquitectura NR *end-to-end* con la inserción de nodos intermedios *intermediate_node_1*, *intermediate_node_2* y *remoteHost*, ilustrando cómo se conectan en una simulación representativa de una red 5G, proporcionando un mayor control sobre los enlaces y su rendimiento. Diagrama extraído y modificado de (CTTC OpenSim Research Unit, 2024).

measurment_0_time	Avg_delay	Avg_throughput	Avg_jitter	Flow_0_delay	Flow_0_throughput	Flow_0_jitter	. . .	Flow_i_delay	Flow_i_throughput	Flow_i_jitter
measurment_1_time	Avg_delay	Avg_throughput	Avg_jitter	Flow_0_delay	Flow_0_throughput	Flow_0_jitter	. . .	Flow_i_delay	Flow_i_throughput	Flow_i_jitter
⋮										
measurment_n_time	Avg_delay	Avg_throughput	Avg_jitter	Flow_0_delay	Flow_0_throughput	Flow_0_jitter	. . .	Flow_i_delay	Flow_i_throughput	Flow_i_jitter

Figura 5.8: Diagrama representativo del formato de los archivos CSV con los que se persisten las mediciones de la experimentación de la Sección 5.5.1.

Traffic Type	Direction	UseUDP	UEs/Gnb	DelayRelError	ThroughputRelError	JitterRelError
NGMN FTP	DL	false	20	$-1{,}74$	$1{,}14 \cdot 10^{-2}$	NaN
NGMN GAMING	DL	true	20	$-0{,}1$	$1{,}25 \cdot 10^{-2}$	0,38
FTP 3GPP-M1	UL	false	10	$4{,}04 \cdot 10^{-2}$	0,61	0,59
FTP 3GPP-M1	UL	true	20	0,34	0,78	0,83
NGMN GAMING	DL	false	10	$-0{,}26$	$1{,}23 \cdot 10^{-2}$	NaN
NGMN VIDEO	DL	false	10	$-0{,}87$	$1{,}23 \cdot 10^{-2}$	NaN
NGMN FTP	DL	true	10	0,93	0,93	0,94
NGMN VIDEO	DL	false	20	$-1{,}74$	$1{,}14 \cdot 10^{-2}$	NaN
UDP CBR	DL	false	20	$-4{,}4 \cdot 10^{-2}$	$2{,}52 \cdot 10^{-2}$	$-0{,}1$
FTP 3GPP-M1	DL	true	20	0,34	0,78	0,83
NGMN MIXED	DL	false	10	$-0{,}45$	$1{,}22 \cdot 10^{-2}$	NaN
NGMN VOIP	DL	true	20	$9{,}62 \cdot 10^{-2}$	$-0{,}83$	$-0{,}33$
NGMN FTP	DL	true	20	0,94	0,93	0,94
NGMN FTP	DL	false	10	$-0{,}87$	$1{,}23 \cdot 10^{-2}$	NaN
FTP 3GPP-M1	DL	false	20	0,34	0,78	0,83
NGMN VOIP	DL	false	10	$-0{,}87$	$1{,}23 \cdot 10^{-2}$	NaN
UDP CBR	DL	true	10	$-6{,}69 \cdot 10^{-2}$	$3{,}13 \cdot 10^{-2}$	$-0{,}13$
NGMN MIXED	DL	true	10	0,22	0,66	0,19
UDP CBR	DL	true	20	$-4{,}4 \cdot 10^{-2}$	$2{,}52 \cdot 10^{-2}$	$-0{,}1$
NGMN VIDEO	DL	true	20	$2{,}7 \cdot 10^{-2}$	$5{,}04 \cdot 10^{-2}$	0,16
UDP CBR	DL	false	10	$-6{,}69 \cdot 10^{-2}$	$3{,}13 \cdot 10^{-2}$	$-0{,}13$
NGMN VOIP	DL	false	20	$-1{,}74$	$1{,}14 \cdot 10^{-2}$	NaN
FTP 3GPP-M1	DL	true	10	$4{,}04 \cdot 10^{-2}$	0,61	0,59
NGMN VIDEO	DL	true	10	$1{,}14 \cdot 10^{-2}$	$4{,}93 \cdot 10^{-2}$	$7{,}2 \cdot 10^{-2}$
NGMN GAMING	DL	false	20	$-0{,}44$	$1{,}14 \cdot 10^{-2}$	NaN
FTP 3GPP-M1	DL	false	10	$4{,}04 \cdot 10^{-2}$	0,61	0,59
FTP 3GPP-M1	UL	false	20	0,34	0,78	0,83
NGMN MIXED	DL	true	20	0,38	0,67	0,25
NGMN MIXED	DL	false	20	$-1{,}12$	$1{,}11 \cdot 10^{-2}$	NaN
NGMN GAMING	DL	true	10	$-6{,}58 \cdot 10^{-2}$	$-7{,}57 \cdot 10^{-4}$	$-0{,}11$
NGMN VOIP	DL	true	10	$2{,}68 \cdot 10^{-4}$	$-1{,}64$	$-0{,}83$
FTP 3GPP-M1	UL	true	10	$4{,}04 \cdot 10^{-2}$	0,61	0,59

Tabla 5.4: Tabla comparativa de la primera medición de cada métrica contra el promedio de la misma. Cuando el *jitter* toma el valor de cero, la columna "JitterRelError" toma el valor NaN (not a number), debido a que se calcula dividiendo por el mismo.

Capítulo 6

Ingeniería de Software

En el desarrollo de un proyecto de software, la planificación y gestión efectiva son cruciales para el éxito. Este capítulo detalla cómo se gestionó el esfuerzo del equipo de desarrollo a lo largo del proyecto. Se abordarán los métodos de estimación de esfuerzo, las herramientas de gestión de proyectos empleadas, el cronograma del proyecto y gestión del tiempo. Además, se describe el proceso de desarrollo adoptado, incluyendo las fases del desarrollo, el control de versiones y la gestión de cambios. A continuación, se detallarán los aspectos clave, proporcionando una visión completa de cómo se llevó a cabo este proyecto desde su concepción hasta su finalización.

6.1. Contexto

Este proyecto de grado se integra dentro del proyecto CSIC "Convergencia entre Redes 5G/6G y Redes Ópticas: Un Enfoque Holístico", que aborda la necesidad de integrar eficientemente las redes ópticas y las redes móviles de próxima generación (5G y 6G) para mejorar el rendimiento, la eficiencia energética y simplificar el diseño y operación de la red. A medida que las comunicaciones juegan un papel cada vez más crucial en la sociedad moderna, la convergencia de estas tecnologías se vuelve esencial. El proyecto propone desarrollar nuevas arquitecturas de redes ópticas-móviles y algoritmos de aprendizaje automático para la asignación dinámica de recursos, con el objetivo de reducir la complejidad de la red, disminuir costos operativos y mejorar la calidad del servicio.

El proyecto CSIC se compone de doce personas en total, divididas en subgrupos que se dedican a desarrollar e investigar distintos aspectos que contribuyen al objetivo del proyecto. Dentro de los proyectos que integran el CSIC se encuentra :

- "Herramienta de simulación de una red *end-to-end* óptico-móvil": elaboración de una herramienta de simulación que abarque ambos dominios.
- SimuSplit: Optimización de Recursos en Redes 5G/6G a través de plata-

formas de simulación simu5G.

- Tesis de posgrado en curso asociadas que utilizarán la herramienta presentada en este documento.

El proyecto que se presenta en este informe consiste en desarrollar una arquitectura inteligente de monitorización de redes 5G. Este proyecto, compuesto por dos personas, será potencialmente utilizado por Lucas Inglés para realizar su doctorado. Por este motivo, él es el principal interesado en el proyecto, quien tuvo el rol de cliente (mejor conocido en metodologías ágiles como *Product Owner*).

6.2. Introducción

El proyecto se estructuró en varias fases principales: aprendizaje del dominio, aprendizaje de ns-3, análisis, diseño, implementación, experimentación y documentación. Estas fases no son disjuntas en el tiempo ya que se solapan. Es importante mencionar que si bien el proyecto se integra dentro del proyecto CSIC, el desarrollo de la herramienta se realizó de forma completamente independiente y en paralelo. Es decir, la única integración realizada fue con la herramienta ns-3 y no con ninguna otra herramienta del proyecto CSIC.

Este proyecto se compone de dos personas y tres tutores. Se siguió una metodología de integración continua con reuniones periódicas para evaluar el avance, planificación y análisis de requerimientos. También se usó un canal de comunicación por texto donde se planteaban preguntas específicas para mantener una retroalimentación activa y progreso constante. Se utilizó Git para la gestión de versiones de código.

6.2.1. Metodología de Desarrollo

Debido a que el proyecto solo se integra por dos personas que desarrollan y tres tutores que hacen el rol de cliente, siendo uno solo el principal interesado (*Product Owner*), no se requirió de una metodología estricta. La poca cantidad de personas facilita la comunicación y flexibilidad en el trabajo y planificación de tareas. Esto no quiere decir que no se siguió ninguna metodología, todo lo contrario, pero se optó por una estrategia pragmática donde se priorizó el desarrollo de la herramienta sobre la gestión burocrática que sólo hubiese atrasado las fechas de entrega. Es por este motivo que procesos como la estimación del esfuerzo de cada tarea, análisis de riesgo o análisis de requisitos, se realizaron de manera informal sin dejar ningún documento o estadística de la evolución del proyecto. Así mismo, los roles desempeñados por cada integrante fueron variando y evolucionando a lo largo del tiempo en función de las tareas prioritarias en cada etapa del proyecto.

Se trabajó con una metodología de integración continua inspirada en scrum y siguiendo prácticas de *DevOps*. Debido a que el proyecto duró quince meses, la disponibilidad de los integrantes fue variando, por lo que la frecuencia de

reuniones se fue ajustando a la necesidad requerida, tanto entre integrantes como con los tutores. Fue complejo asignar tareas disjuntas debido a la intrinsicidad del proyecto, por esta razón se optó por una metodología de trabajo variable según la etapa del proyecto. En ocasiones trabajando en partes distintas para luego integrar el código y en otras haciendo programación de a pares (*pair programming*) para aprovechar al máximo las fortalezas de cada integrante.

6.3. Comunicación

A continuación se describen los medios de comunicación utilizados para este proyecto y las aplicaciones utilizadas.

Se utilizó la aplicación y sitio web *Mattermost* para la comunicación por texto con los profesores. En esta aplicación se resolvieron dudas particulares, ya sea del dominio, de ns-3 o de planificación, para comentar el avance y agendar reuniones. Se mantuvo una comunicación activa que facilitó el avance del proyecto. También se utilizó el mail pero con menor frecuencia y solo para notificar cosas concretas. Entre los integrantes hubo una comunicación aún más activa por WhatsApp y Discord, de esta manera se organizaba la asignación de tareas para no trabajar sobre la misma sección de código al mismo tiempo.

Para llevar registro de la asignación de tareas y planificación del *sprint* se utilizó un tablero en Trello. No se utilizó ningún método de estimación formal, en el momento de asignar tareas se evaluaba cuanto se creía que iba a durar cada una, asignando las que se evaluaron como más prioritarias para el desarrollo del proyecto.

También se contó con un documento Notion donde se llevó registro de las etapas del proyecto. Este medio de comunicación fue de utilidad para gestionar los *liks* del *Git*, *link* de las grabaciones de los seminarios realizados, fechas de reuniones agendadas y *link* de *OwnCloud* con documentos sobre el proyecto *CSIC*.

Las reuniones con los tutores se realizaron por Zoom, por lo general al finalizar cada etapa a modo de evaluación y dos o tres entre medio para evaluar el avance y resolver dudas. La frecuencia de las reuniones fue variando según la necesidad que fue requiriendo el proyecto a lo largo del tiempo.

6.4. Estimación y Planificación

No se siguió ningún método de estimación formal. El proceso de estimación y planificación se realizó en conjunto con los tutores. Se fueron planteando hitos para crear una red cada vez más compleja y general que abarque más casos de uso iteración a iteración. Cada vez que se lograba uno de estos hitos se agendaba una reunión con los tutores para mostrar el avance, validarlo y planificar las próximas funcionalidades para la siguiente iteración. Si no se llegaba a la fecha estimada, se avisaba y reagendaba haciendo uso del *Mattermost*.

Así mismo, ambos integrantes del equipo, mantuvieron una comunicación

activa para realizar una asignación de tareas más pequeñas, estimar los tiempos de cada una para así llegar a las fechas de entrega en tiempo y forma. En caso de encontrarnos con un obstáculo, se realizaba una reunión entre los integrantes y en caso de no poder resolverla o necesitar consultar dudas se agendaba una reunión con los tutores luego de consultar con los canales de texto.

6.4.1. Asignación de Tareas

Las tareas se enfocaron principalmente en funcionalidades, variando en ocasiones a un enfoque por tareas, sobre todo con aquellas relacionadas con la documentación. Cada tarea se asocia a una tarjeta de Trello con un título breve y descriptivo y una descripción que explica más en profundidad en qué consiste. Como el equipo se compone solo de dos personas, no se requirió dedicarle mucho esfuerzo a la prolijidad del Trello, debido a que ambos sabían las tareas que había que realizar, sirviendo más como gestor y recordatorio de las tareas pendientes.

También se intentó seguir una línea lógica para la asignación de tareas, para que un mismo integrante abarque todas aquellas de gran similitud, no duplicar esfuerzos y reducir los tiempos de implementación. En ocasiones se tuvieron que tomar decisiones de diseño o resolver problemas complejos y cruciales para el proyecto, en estos casos se realizaron reuniones entre los integrantes, haciendo *pair programming* cuando fue necesario y consultando a los tutores por *Mattermost* en caso de necesitar una segunda opinión de un experto en aspectos puntuales.

6.4.2. Recursos Humanos

Se puede decir que no se optó por utilizar roles formales en el equipo. Ambos integrantes desempeñaron diversas tareas según el proyecto lo fue requiriendo, que abarcaron: investigación, implementación de código y documentación. Quien sí tuvo un rol bien definido fue el tutor Lucas Inglés, quien actuó de *Product Owner* junto a Alberto Castro y Claudina Rattaro, quienes actuaron de cliente sugiriendo mejoras del producto a lo largo de las etapas de desarrollo. La experiencia de los tres en herramientas de simulación fue de gran utilidad para garantizar la calidad de la herramienta lograda.

6.4.3. Control de Versiones

Para el control de versiones se utilizó un repositorio Git haciendo uso de las buenas prácticas de *GitFlow* para organizar la integración de código. De esta forma se contaba con completa trazabilidad, pudiendo revisar y restaurar el código a versiones anteriores en caso de requerirlo. Cada brach fue asociada al nombre de la funcionalidad desarrollada e integrada una vez finalizada su implementación y testeada (de ser posible, en algunos casos no se pudo probar hasta una etapa posterior debido a dependencia a otros módulos en desarrollo).

Siempre se buscó trabajar en áreas separadas de código para minimizar los conflictos.

6.5. Gestión de Riesgos

No se realizó una metodología formal de gestión de riesgos, sin embargo, durante todo el proyecto siempre existió el riesgo negativo de no encontrar una manera de implementar particularidades del sistema con las herramientas provistas por ns-3 por defecto y tener que implementarlas por nuestra cuenta, lo que supondría un retraso en las fechas de entrega, siendo una variable importante a considerar al estimar los tiempos. De hecho, fue lo que ocurrió, ya que ns-3 no considera el flujo nodo a nodo sino punto a punto. Para localizar las fallas fue necesario agregar información, extendiendo las clases por defecto de ns-3 como se explicó previamente. Afortunadamente, esto no tuvo grán impacto en los tiempos de entrega finales, debido a la gran flexibilidad y facilidad que presenta ns-3 para agregar y extender las funcionalidades de la herramienta.

Por otra parte, existió el riesgo positivo de contar con el tutor Alberto Castro, quien cuenta con un muy buen manejo de la herramienta ns-3 a quien se supo aprovechar cuando existieron obstáculos. También, era de conocimiento que la herramienta ns-3 cuenta con una gran comunidad activa que responde en los foros y aunque no fue necesario utilizar este recurso era un beneficio potencial. También, el tutor Alberto Castro cuenta con contactos dentro del equipo de desarrollo de ns-3 a quienes se podrían haber contactado ante eventuales dificultades.

6.5.1. Cronograma del Proyecto

En esta sección se presentará el cronograma del proyecto, indicando las actividades principales e hitos significativos que definieron las distintas etapas a lo largo de sus quince meses de duración; a continuación, se ofrece un resumen:

- **Marzo a Mayo 2023 - Capacitación y aprendizaje del dominio (3 meses):** Se llevó a cabo un intenso período de capacitación en los dominios de redes 5G y redes ópticas. Esta fase incluyó la participación en seis seminarios especializados, realizados entre el 26 de mayo y el 2 de julio. Cada seminario fue presentado por expertos o grupos de trabajo, con el propósito de formar a los integrantes del proyecto CSIC en temas específicos:
  - Seminario 1: Redes Ópticas.
  - Seminario 2: Redes Móviles (evolución de 1G a 5G, proyecciones a 6G).
  - Seminario 3: Introducción al Aprendizaje Automático.
  - Seminario 4: *Deep Reinforcement Learning* y *Graph Neural Networks* para la asignación eficiente de recursos en 5G.

- Seminario 5: Omnet++ y GnPy.
- Seminario 6: ns-3.

- **Junio a Agosto 2023 - Aprendizaje de la herramienta ns-3 (3 meses):** Se dedicó un período extenso al aprendizaje y familiarización con la herramienta de simulación ns-3, fundamental para el desarrollo del proyecto. Durante esta investigación se realizaron diversas exploraciones con fines de acercarse a una primera implementación de la herramienta de monitorización y así aprendiendo sobre las distintas herramientas, características y facilidades que ofrece.

- **Setiembre a Octubre de 2023 - Topología en funcionamiento (2 meses):** Se logró implementar la topología básica de la red utilizando como base el ejemplo `cttc-nr-traffic-ngmn-mixed.cc` de ns-3. Se establecieron parámetros clave para la experimentación, incluyendo el protocolo de transporte, la configuración dúplex, el perfil de tráfico, y la dirección de tráfico (*uplink*/*downlink*).

- **Octubre a Noviembre de 2023 - Generación, análisis de *PCAPs* y investigación de *TraceSources*(1.5 meses):** Se investigó la generación de trazas postsimulación para analizar el rendimiento de la red. Además, se inició el desarrollo de un módulo para generar trazas durante la ejecución de la simulación, con el objetivo de crear un sistema que pudiera ser activado por métricas específicas.

  Adicionalmente, se investigó el uso del componente de ns-3 llamado *Trace Sources* para la extracción de métricas *node-to-node*. Este enfoque probó tener la capacidad para realizar la tarea requerida. Pero finalmente es abandonada por la complejidad que implica implementar módulos que utilicen esta herramienta. Es aquí cuando se produce un giro de la estrategia de desarrollo y se busca soluciones que permitan el acceso a métricas *node-to-node* de una manera más sencilla.

- **Noviembre de 2023 - Generación dinámica de reportes (0.5 meses):** Se identificó e implementó el uso de "*FlowMonitor*" en combinación con las características de programación de ns-3 (`Simulator::Schedule`) como la solución ideal para la monitorización dinámica de la red. Se desarrolló un script capaz de extraer estadísticas detalladas de la simulación, incluyendo métricas como *throughput*, *delay* y *jitter* para cada flujo de la red.

- **Noviembre a Diciembre de 2023 - Monitorización de bajo nivel e introducción de la red central (2 meses):** Se investigó la capacidad de realizar métricas nodo a nodo y se comenzó a trabajar en la introducción de enlaces entre *gNBs*, preparando el terreno para la futura incorporación de la red central.

- **Diciembre a Febrero 2024 - Desarrollo de métricas *node-to-node* y sistema de *triggers* (2 meses):** Se inició el desarrollo paralelo de dos

componentes clave: las clases necesarias para monitorizar el tráfico entre los nodos (ver Sección 4.3.2) y paralelamente las métricas *end-to-end* y el sistema de *triggers* basado en "*FlowMonitor*".

- **Enero a Junio 2024 - Documentación del Proyecto (6 meses):** Realizar documentación.
- **Marzo a Junio 2024 - Experimentación (4 meses):** Probar el sistema (experimentación) y crear escenarios para garantizar su correcto funcionamiento.

Inicialmente, el proyecto tenía la intención de integrar al sistema de monitorización un modelo de aprendizaje automático que fuera capaz de realizar mediciones de forma más inteligente. Esto tuvo que quitarse de los objetivos debido a restricciones de tiempo y a que llevó más de lo esperado las etapas previas. Es por esta razón que el sistema de monitorización se presenta de forma completa como herramienta, pero se dejó un poco de lado su inteligencia, teniendo una implementación que no aprovecha al máximo los potenciales de la herramienta implementada. Esto abre puertas a futuros proyectos que se puedan realizar con esta herramienta.

A lo largo de este cronograma, el equipo experimentó retrasos y ajustes en los plazos debido a la complejidad técnica del proyecto y la curva de aprendizaje asociada con ns-3 y 5G-LENA. Algunos hitos tomaron más tiempo del previsto inicialmente, lo que llevó a una reevaluación constante de las prioridades y enfoques. A pesar de estos desafíos, el equipo logró adaptarse y avanzar en el desarrollo del sistema de monitorización, aunque con algunas funcionalidades aún en proceso de refinamiento al final del período del proyecto.

Capítulo 7

Conclusiones

En este capítulo, se evalúan los resultados alcanzados, las dificultades encontradas y se discuten las posibles extensiones del trabajo realizado. A lo largo del proyecto, se logró desarrollar e implementar una arquitectura de monitorización para redes 5G utilizando ns-3 y el módulo 5G-LENA, capaz de detectar anomalías, identificar enlaces problemáticos y ajustar dinámicamente la frecuencia de extracción de métricas.

7.1. Evaluación de Resultados

Los resultados obtenidos a lo largo de las pruebas realizadas indican que el sistema de monitorización desarrollada cumple con los objetivos planteados inicialmente. A continuación, se resumen los principales logros del proyecto:

- **Identificación de enlaces de bajo rendimiento**: La herramienta es capaz de almacenar *logs* de sistema, para luego identificar dinámicamente el enlace *node-to-node* con el peor rendimiento del sistema, permitiendo una rápida localización de problemas en la red. Esto lo logra realizar de una manera abstracta que le permite instalarse en cualquier red sin conocimiento de antemano sobre la arquitectura de la misma.
- **Detección de Anomalías**: Se logra desarrollar un sistema capaz de detectar disminuciones en el rendimiento de los flujos de datos a nivel *end-to-end*.
- **Flexibilidad y Escalabilidad**: El diseñó del sistema es flexible y escalable, permitiendo su uso en diferentes arquitecturas de red y perfiles de tráfico. Esto se demostró tanto en escenarios simples como en configuraciones más complejas y realistas.
- **Estándar de *Logs* en XML**: La creación de un formato estándar de *logs* en XML facilita la revisión y análisis de los resultados de la simulación, mejorando la capacidad de depuración y análisis del sistema.

- **Creación de un *dataset***: Mediante la experimentación, se construyó un dataset con información de las métricas *delay*, *throughput* y *jitter* registradas en un conjunto de simulaciones realizadas a partir de ns-3. Este *dataset* será dejado disponible para que se pueda utilizar por otros grupos de investigación.

Además, como parte de nuestro compromiso con la comunidad científica y de desarrollo, todo el código fuente del proyecto, incluyendo la implementación de la arquitectura de monitorización, los scripts de simulación y las herramientas de análisis, se ha puesto a disposición del público en un repositorio de GitLab (Leandro Alfonso y Nicolás Rivoir, 2024). Esta decisión no solo permite la reproducibilidad de nuestros resultados, sino que también facilita la colaboración y el desarrollo continuo de herramientas para la simulación y monitorización de redes 5G.

7.2. Dificultades Encontradas

Durante el desarrollo del proyecto, se enfrentaron diversas dificultades:

- **Curva de Aprendizaje**: La complejidad de ns-3 y 5G-LENA presentó una curva de aprendizaje considerable. La extensa documentación y los ejemplos disponibles fueron esenciales para superar esta barrera.
- **Integración de Módulos**: La integración de los módulos de ns-3 y 5G-LENA, junto con la integración de otras bibliotecas como *tiny-xml-2*, requirió un esfuerzo importante en el ámbito de la configuración del sistema, en un intento de que todas funcionarán de manera cohesiva.
- **Tiempo de ejecución de simulaciones**: Las simulaciones que se deben correr para iterar/validar el sistema de monitoreo fueron creciendo exponencialmente. Alcanzando incluso simulaciones que demoraron varios días en ejecutarse. Si bien se consideraron alternativas como utilizar el *clusteruy* para realizar las mismas, esta solución no funcionó debido a incompatibilidades entre las versiones de las bibliotecas necesarias para esta tarea.
- **Desarrolló de una estrategia de validación**: Un gran desafió que se enfrentó en este proyecto fue el de encontrar una manera de validar el sistema de monitorización desarrollado. Esto se debió a que no se encontraron antecedentes de otro sistema que funcionara de una manera similar, por lo que se tuvo que encontrar una manera innovadora para poder generar una referencia contra la cual comparar el sistema de monitorización.

7.3. Reflexión Final

Este proyecto presentó múltiples oportunidades de aprendizaje y desafíos que llevó al equipo a poner a prueba las capacidades de gestionar e investigar

un área de desarrollo, desconocida previamente para este.

El equipo de investigación encuentra los resultados de este proyecto satisfactorios. Se alcanzó el objetivo de desarrollar un sistema de monitorización para redes 5G-NR, dentro de un escenario realista, asimismo, se logró generar métricas *node-to-node*, las cuales no estaban disponibles previamente y poner a prueba una estrategia para determinar cuándo realizar una monitorización detallada y cuando no.

Por otra parte, se encuentra que la estrategia de monitorización implementada, si bien funcional, es limitada y no es la mejor opción para una red-5G real, donde se pueden esperar fluctuaciones de flujo mucho más pronunciadas que aquellas encontradas durante la experimentación. A pesar de esto, el código generado durante este permite la adopción de nuevas estrategias de monitorización, que pueden ser implementadas sin mayores complicaciones.

Capítulo 8

Trabajos futuros

Dentro de esta sección se plantea la posible evolución del presente proyecto. Si bien el equipo encuentra que el sistema de monitorización desarrollado fue satisfactorio dentro de los parámetros del proyecto, esto no quita que se encuentren aspectos en los cuales mejorar.

Primeramente, se desea mencionar la estrategia de monitorización como un aspecto a mejorar. En este proyecto se trabajó únicamente con una estrategia de monitorización binaria, es decir, se considera que la red está en un estado aceptable o no lo está, y el criterio que se utiliza para definir esto es estático, debido a que el *threshold* definido nunca cambia. Esta estrategia es significativamente restrictiva. Las redes 5G, como se presentó anteriormente, son altamente fluctuantes, por lo que el utilizar una estrategia que conserva el mismo umbral a lo largo de la misma no es la opción óptima. El equipo de trabajo planteó el uso de un umbral evolutivo, que vayan aumentando o decreciendo con las medidas de la red en el momento más reciente. Incluso un factor que determine la tasa de aprendizaje entre una medida y la anterior, puede brindar mucha flexibilidad ante estas fluctuaciones. Por restricciones respecto al alcance y tiempo determinado para este proyecto, no se pudo avanzar con esta idea, por lo que lo dejamos en manos del equipo que continúe desarrollando el presente sistema de monitorización.

Adicionalmente, hay un aspecto de implementación que de ser iterado puede llegar a arrojar resultados positivos. Durante la implementación de la función "*NodeToNodeTrigger*", surgió la duda de cómo considerar el rendimiento de un enlace particular entre dos nodos. En la implementación se decidió como heurística, tomar el *delay* que haya reportado el último paquete como el *delay* del enlace al momento de la medición, pero la misma no es nada trivial, que sucede si este paquete es tan solo uno entre cientos de paquetes que pasan por este enlace; ¿realmente representa el estado del enlace? Se propone que se investiguen alternativas para esta heurística, como por ejemplo tomar el *delay* promedio de los últimos 100 paquetes.

Un aspecto crucial para el desarrollo futuro de este proyecto es la incorporación de características específicas de las redes ópticas en la simulación.

Actualmente, el sistema utiliza enlaces genéricos de ns-3 para representar las conexiones ópticas, lo cual, si bien es funcional, no captura completamente las particularidades de las redes ópticas reales. Para abordar esta limitación, se proponen dos líneas de trabajo futuro:

1. Desarrollar e implementar módulos específicos en ns-3 que simulen con mayor precisión las características de las redes ópticas, incluyendo aspectos como la atenuación de la señal, efectos de dispersión, y multiplexación por división de longitud de onda (WDM).

2. Explorar la posibilidad de integrar ns-3 con un simulador especializado en redes ópticas. Esta integración permitiría combinar las fortalezas de ns-3 en la simulación de redes 5G con las capacidades específicas de un simulador óptico, logrando así una representación más precisa y completa de una red óptico-móvil convergente.

Estas mejoras no solo aumentarían la fidelidad de las simulaciones, sino que también permitirían estudiar escenarios más realistas de interacción entre las redes 5G y las redes ópticas, contribuyendo significativamente al objetivo general del proyecto CSIC de lograr una convergencia efectiva entre ambas tecnologías.

Finalmente, recomendamos enfáticamente que se realice la inversión inicial de generar un ambiente de trabajo que permita la realización de simulaciones a través de un servicio de cómputo externo al que se pueda conectar mediante el uso de un equipo de computación doméstico. A medida que las simulaciones se hacen más y más complejas, aumentan rápidamente el tiempo que demoran en ejecutarse, limitando significativamente la capacidad de iteración sobre la solución y experimentación.

Referencias

3GPP. (2022). *5g system overview.* https://www.3gpp.org/technologies/5g-system-overview. (Accessed: 2024-07-12)

Alkenani, J., y Nassar, K. A. (2022, noviembre). Network performance analysis using packets probe for passive monitoring. *Informatica*, *46*(7), 153–160. Descargado de https://doi.org/10.31449/inf.v46i7.4307 doi: 10.31449/inf.v46i7.4307

Barzegar, S., Wang, S., Ruiz, M., Velasco, L., Richart, M., y Castro, A. (2023). Coordination of radio access and optical transport. En *2023 international conference on optical network design and modeling (ondm)* (p. 1-3).

Budigere, K. (2016). *Router modelling and packet level cache implementation for content aware tcp (catcp)* (Tesis Doctoral no publicada).

Carneiro, G., Fortuna, P., y Ricardo, M. (2010, May). Flowmonitor - a network monitoring framework for the network simulator 3 (ns-3). En *2nd international icst international workshop on network simulation tools.* doi: 10.4108/ICST.VALUETOOLS2009.7493

Crawshaw, J. (2021, enero). *Overcoming the 5g challenges of monitoring, assurance, and automation.* https://www.netscout.com. New York, NY. (Heavy Reading white paper produced for NETSCOUT)

CTTC OpenSim Research Unit. (2024). ns-3 LENA 5G: Network simulator 3 lena 5g documentation [Manual de software informático]. Open Source. Descargado de https://cttc-lena.gitlab.io/nr/nrmodule.pdf (Accessed: 2024-07-03)

Dahlman, E., Parkvall, S., y Sköld, J. (2018). *5g nr: The next generation wireless access technology.* London, United Kingdom: Elsevier.

Feit, S. M. (1995). *Snmp: A guide to network management.* McGraw-Hill.

GlobeNewswire. (2023). *Uruguay sees the region's first 5G deployment.* https://www.globenewswire.com/news-release/2020/04/24/2021801/0/en/Uruguay-sees-the-region-s-first-5G-deployment.html. (Accessed: 2024-07-04)

Hegarty, S. (2021). *A quick ru to du to cu to core of 5g vran.* https://stevehegarty.wordpress.com/2021/11/11/a-quick-ru-to-du-to-cu-to-core-of-5g-vran/. (Accessed: 2024-07-11)

International Trade Administration. (2024). *Uruguay - Information and Communication Technology.* https://www.trade.gov/country-commercial

-guides/uruguay-information-and-communication-technology. (Accessed: 2024-07-04)

Koutlia, K., Bojovic, B., Ali, Z., y Lagen, S. (2022, septiembre). Calibration of the 5g-lena system level simulator in 3gpp reference scenarios. *Simulation Modelling Practice and Theory (SIMPAT)*, *119*, 102580.

Leandro Alfonso y Nicolás Rivoir. (2024). *Monitoring topologies repository.* https://gitlab.fing.edu.uy/mobile-optical-networks/monitoring_topologies. (Accessed: 2024-07-03)

López, V., y Velasco, L. (2016). *Elastic optical networks: Architectures, technologies, and control* (1.ª ed.). Cham: Springer Cham. doi: 10.1007/978-3-319-30174-7

Mauro, D., y Schmidt, K. (2005). *Essential snmp, 2nd edition.* O'Reilly Media, Inc.

Mohr, W. (2017). *The 5g infrastructure association.* ITU-R. [Online]. Available: https://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2020/Documents/S03-15GPPP. (Accessed: Mar. 29, 2019)

Mouchet, M., Randall, M., Ségneré, M., Amigo, I., Belzarena, P., Brun, O., ... Vaton, S. (2020). Scalable monitoring heuristics for improving network latency. En *Ieee/ifip network operations and management symposium (ieee/ifip noms 2020).* Budapest, Hungary. Descargado de https://hal.laas.fr/hal-02413636 (HAL Id: hal-02413636. Submitted on 16 Dec 2019)

Mészáros, L., Varga, A., y Kirsche, M. (2019, 05). Inet framework. En (p. 55-106). doi: 10.1007/978-3-030-12842-5_2

Nardini, G., Sabella, D., Stea, G., Thakkar, P., y Virdis, A. (2020). Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks. *IEEE Access*, *8*, 181176-181191. doi: 10.1109/ACCESS.2020.3028550

ns-3 Consortium. (2024). ns-3: Network simulator 3 documentation [Manual de software informático]. Open Source. Descargado de https://www.nsnam.org/documentation/ (Accessed: 2024-07-03)

ns-3 Project. (2024). *Quick start — prerequisites.* https://www.nsnam.org/docs/installation/html/quick-start.html#prerequisites. (Accessed: 2024-07-12)

Patriciello, N., Lagen, S., Bojovic, B., y Giupponi, L. (2019, noviembre). An e2e simulator for 5g nr networks. *Simulation Modelling Practice and Theory (SIMPAT)*, *96*, 101933.

Rappaport, T. S., Xing, Y., MacCartney, G. R., Molisch, A. F., Mellios, E., y Zhang, J. (2017, dec). Overview of millimeter wave communications for fifth-generation (5g) wireless networks-with a focus on propagation models. *IEEE Transactions on Antennas and Propagation*, *65*(12), 6213–6230. (Publisher Copyright: © 1963-2012 IEEE.) doi: 10.1109/TAP.2017.2734243

Shafi, M., Molisch, A. F., Smith, P. J., Haustein, T., Zhu, P., De Silva, P., ... Wunder, G. (2017). An in-depth look at 5g standards, trials, deployment challenges, and practical implementations. *IEEE Journal on*

Selected Areas in Communications, *35*(6), 1201–1221. doi: 10.1109/JSAC.2017.2692307

Sommer, C., German, R., y Dressler, F. (2011, January). Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, *10*(1), 3–15. doi: 10.1109/TMC.2010.133

Stallings, W. (1993). *Snmp, snmpv2, and cmip: The practical guide to network management standards*. Addison-Wesley.

Thomason, L. (2024). *TinyXML2: Simple, small, efficient, c++ xml parser*. https://github.com/leethomason/tinyxml2. (Accessed: 2024-07-12)

Tseng, Y., Aravinthan, G., Berde, B., Imadali, S., Houatra, D., y Qiu, H. (2019). Re-think monitoring services for 5g network: Challenges and perspectives. En *2019 6th ieee international conference on cyber security and cloud computing (cscloud)/2019 5th ieee international conference on edge computing and scalable cloud (edgecom)* (pp. 34–39). Paris, France: IEEE. Descargado de https://ieeexplore.ieee.org/document/8784456 (Accessed: 2024-06-19) doi: 10.1109/CSCloud/EdgeCom.2019.00016

Varga, A. (2010). Omnet++. En K. Wehrle, M. Güneş, y J. Gross (Eds.), *Modeling and tools for network simulation* (pp. 35–59). Berlin, Heidelberg: Springer Berlin Heidelberg. Descargado de https://doi.org/10.1007/978-3-642-12331-3_3 doi: 10.1007/978-3-642-12331-3_3

Velasco, L., Castro, A., Asensio, A., Ruiz, M., Liu, G., Qin, C., ... Yoo, S. J. B. (2017). Meeting the requirements to deploy cloud ran over optical networks. *Journal of Optical Communications and Networking*, *9*(3), B22–B32. doi: 10.1364/JOCN.9.000B22

Velasco, L., Piat, A. C., Gonzlez, O., Lord, A., Napoli, A., Layec, P., ... Casellas, R. (2019, noviembre). Monitoring and data analytics for optical networking: Benefits, architectures, and use cases. *IEEE Network*, 100–108. Descargado de https://ieeexplore.ieee.org/document/8770528 (Accepted from Open Call) doi: 10.1109/MNET.2019.1800341

Velasco, L., y Ruiz, M. (2017). *Provisioning, recovery, and in-operation planning in elastic optical networks*. Hoboken, NJ: Wiley-Blackwell. doi: 10.1002/9781119338628

Capítulo 9

Anexo 1: Instalación de la herramienta y ejemplo de uso

En este capítulo se explica como instalar las diferentes herramientas utilizadas a lo largo del proyecto y se presenta un caso de uso. Es importante destacar que a la hora de ejecutar comandos, se asume que se trabaja con un sistema con Ubuntu como sistema operativo.

9.1. Instalación de ns-3

Para instalar la herramienta ns-3 es importante verificar que todos los prerrequisitos de sistema sean cumplidos, ver (ns-3 Project, 2024).

Adicionalmente, para poder instalar 5G-LENA, se deberán instalar los siguientes paquetes:

```
sudo apt-get install libc6-dev sqlite sqlite3
   libsqlite3-dev
```

Luego se procederá a la instalación de ns-3. Para esto se utilizará git y el repositorio oficial del proyecto. Además, se utilizará la *branch* que mantiene la versión 3.41 del proyecto, ya que es la última versión compatible con 5G-LENA.

```
git clone https://gitlab.com/nsnam/ns-3-dev.git
cd ns-3-dev
git checkout -b ns-3.41 ns-3.41
```

Para verificar que se instaló y funciona correctamente, se deberán ejecutar los test que vienen con ns-3.

```
./ns3 configure --enable-examples --enable-tests
```

```
./test.py
```

9.2. Instalación de 5G-LENA

Para instalar este módulo simplemente hay que ejecutar los siguientes comandos desde el directorio *ns-3-dev* creado anteriormente:

```
cd contrib
git clone https://gitlab.com/cttc-lena/nr.git
cd nr
git checkout -b 5g-lena-v3.0.y origin/5g-lena-v3.0.y
```

Para verificar el funcionamiento de ns-3 + 5G-LENA, se deberá correr el siguiente comando

```
./ns3 configure --enable-examples --enable-tests
```

y verificar que las opciones ***SQLite support*** y ***Eigen3 support*** están habilitadas.

9.3. Compilar el proyecto

Para compilar el proyecto se debe usar el siguiente comando

```
./ns3 build
```

En caso de éxito, se deberá observar el módulo nr entre los primeros módulos compilados.

9.4. Instalación del código del proyecto

```
cd contrib/nr
git clone
    git@github.com:lea-alfonso/ns3_topologies.git
cp -r -u contrib/nr/examples/*
    contrib/nr/ns3_topologies
mv contrib/nr/ns3_topologies contrib/nr/examples
cp -r --update=all contrib/nr/examples/flow-monitor
    src/
./ns3 build
```

9.4.1. Ejemplo de uso

Para correr el escenario de pruebas desarrollado en la Sección 4.2 se deberá correr el escenario desde el directorio ns-3-dev:

```
./ns3 run "contrib/nr/examples/topology_1_3.cc
   --outputDir=./contrib/nr/examples/experiments
   --simTag=test --bottleNeckDelay=5
   --simTimeMs=30000 --direction=DL
   --trafficTypeConf=3 --uesPerGnb=10 --numRings=0"
```

Notar los siguientes paramentros de la simulación

- ***outputDir***: Esto define donde los *logs* de sistema serán guardados.
- ***simTag***: Esto define un prefijo que se le agregará a los archivos resultantes de la simulación.
- ***bottleNeckDelay*** Esto define el *delay* que será introducido en el enlace *bottleNeck*. La unidad es en ms.
- ***simTimeMs*** : Valor numérico que define la cantidad de milisegundos durara la simulación.
- ***direction*** : Dirección del trafico de la simulación, puede tomar valores UP(*UpLink*) o DL(*DownLink*).
- ***trafficTypeConf***: Valor numérico entre el 0 y el 7 que define el perfil de trafico de la simulación. 0 - UDP CBR, 1 - FTP Model 1, 2 - NGMN FTP, 3 - NGMN VIDEO, 4 - HTTP, 5 - NGMN GAMING, 6 - NGMN VOIP, 7 - NGMN MIXED (10 % FTP, 20 % HTTP, 20 % VIDEO STREAMING, 30 % VoIP, 20 % GAMING).
- ***uesPerGnb*** : Valor que indica la cantidad de *UEs* asignados a cada *gNB*.
- ***numRings***: Valor numérico que representa la cantidad de anillas de gNBs que tendrá la topología hexagonal de la simulación.