UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Herramienta de simulación de una red *end-to-end* óptico-móvil

Informe de Proyecto de Grado presentado por

Micaela Rodríguez

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, Claudina Rattaro
Lucas Inglés

Montevideo, 17 de julio de 2024

Agradecimientos

En primer lugar, quiero expresar mi más profundo agradecimiento a mis tutores, Alberto, Claudina y Lucas, cuya guía y soporte fueron fundamentales para la realización de este proyecto.

Gracias a Alberto por contarme sobre el proyecto y despertar mi interés a inicios del año pasado, así como por su apoyo constante y ayuda en base a su experiencia y conocimiento. Destaco especialmente sus charlas, que enfatizaban la relevancia de cada hito y me ayudaban a no perder de vista el valor de lo que estaba haciendo, lo cual fue una gran motivación para seguir avanzando y dar lo mejor de mí.

A Claudina, gracias por todo el apoyo y conocimiento brindado a lo largo de todo el proyecto, así como por sus opiniones sinceras y destacando sus correcciones continuas, que sin duda fueron muy valiosas para mí y para el desarrollo de este proyecto.

Asimismo, gracias a Lucas por todas sus correcciones exhaustivas y consejos para la documentación. Su mirada más cercana a la de un estudiante implicó un nuevo punto de vista más allegado a mí y que resulto muy enriquecedor para el proyecto.

Me gustaría hacer un reconocimiento especial a Alberto y Claudina, por sus charlas y valiosos consejos, los cuales tal vez iban más allá de sus responsabilidades como tutores académicos, pero demuestran que además de ser excelentes profesionales también son excelentes personas y fueron un apoyo fundamental para mí durante todo el proceso.

Quiero agradecer a todos los integrantes del proyecto CSIC, los seminarios brindados fueron un gran aprendizaje para mí y para poder lograr el desarrollo de este proyecto. Especialmente a Máximo por la ayuda con OMNeT++ y la implementación de la red óptica. Ha sido un honor trabajar con un equipo tan variado y multidisciplinario.

Por último, pero no menos importante, quiero agradecer a mi familia y amigos por el apoyo incondicional y siempre confiar en mí. Fue muy importante para mi sentirme acompañada durante todo el camino recorrido. Agradezco especialmente a mis padres Luis y Mónica, por siempre estar ahí y a mi hermano Jonathan por su especial interés en este proyecto, leyéndolo y dedicando su tiempo para aportarme su opinión sincera.

Muchas gracias a todos por haber sido parte de este proyecto y contribuir aportando desde donde les fue posible.

Resumen

Hoy en día, las telecomunicaciones son imprescindibles en la vida de cualquier persona. A medida que pasa el tiempo, el consumo digital aumenta a altas velocidades, incrementando drásticamente la demanda de ancho de banda y exigiendo servicios de calidad sin interrupciones. Aquí es donde entra el 5G, la quinta generación de tecnología móvil, que promete alta capacidad de transmisión de datos, bajas latencias y la capacidad de conectar a muchos usuarios simultáneamente.

La incorporación de redes ópticas a la infraestructura de telecomunicaciones marcó un antes y un después, mejorando significativamente la velocidad de transmisión y las tasas de datos. Estas redes son consideradas la base para las futuras comunicaciones en la sociedad moderna y serán clave para las comunicaciones móviles de quinta y sexta generación. De esta manera, sería posible cumplir con los requerimientos de las nuevas generaciones.

Contar con herramientas de simulación es muy relevante, ya que permiten a los investigadores y desarrolladores realizar pruebas de concepto, evaluar distintos escenarios de la realidad, realizar pruebas de compatibilidad entre redes ópticas y móviles, detectar problemas de forma temprana y obtener análisis detallados, entre otras cosas. Hasta el momento, no existe una plataforma de código abierto que combine estas dos tecnologías. Por esta razón, en este proyecto se diseña una herramienta de simulación óptico-móvil, donde se pueda reflejar la convergencia de la tecnología 5G y las redes ópticas.

Para ello, será necesario comprender ambos dominios en profundidad, conociendo las fortalezas y debilidades de las redes ópticas y móviles 5G. La unión de estas dos tecnologías completamente distintas no será sencilla, y habrán muchas pruebas fallidas hasta que se logre la convergencia.

Una vez terminada, esta herramienta permitirá realizar varias simulaciones con diversas topologías para evaluar los resultados y, en caso de ser positivos, llevarlo a la realidad algún día.

Este proyecto es parte de un proyecto más abarcativo llamado “Convergencia entre redes 5G/6G y redes ópticas: un enfoque holístico” con objetivos más ambiciosos, que además incluye el desarrollo de una arquitectura para la monitorización de una red de telecomunicaciones óptico-móvil, entre otras cosas. Este proyecto de grado, en particular, aporta una herramienta que combine el dominio óptico y móvil, permitiendo realizar simulaciones de distintos escenarios y tomar estadísticas de los mismos. Los resultados obtenidos podrían usarse para

crear un dataset y entrenar algoritmos de inteligencia artificial que predigan si se cumplirán ciertos requerimientos de 5G dada una topología y una demanda. Por lo tanto, los resultados contribuirán en gran medida a seguir invirtiendo en esta idea y acercarnos cada día a una mejor calidad de vida gracias a los avances tecnológicos.

Palabras clave: Redes ópticas, Redes 5G, Simulación

Índice general

Acrónimos

2G 2nd Generation

3G 3rd Generation

3GPP 3rd Generation Partnership Project

4G 4th Generation

5G 5th Generation

5GCN 5G Core Network

AMF Access and Mobility Management Function

ANTEL Administración Nacional de Telecomunicaciones

API Application Programming Interface

AR Augmented Reality

BER Bit Error Rate

BGP Border Gateway Protocol

BVT Bandwidth Variable Transponder

CBR Constant Bit Rate

CDMA Code Division Multiple Access

CPUs Central Processing Units

CSIC Comisión Sectorial de Investigación Científica

DCI Downlink Control Information

DCM Dispersion Compensation Module

DEMUX Demultiplexor

DiffServ Differentiated Services

DINATEL Dirección Nacional de Telecomunicaciones

DL Downlink

DWDM Dense Wavelength Division Multiplexing

D-AMPS Digital Advanced Mobile Phone System

DFT-OFDM Discrete Fourier Transform Orthogonal Frequency Division Multiplexing

DN Data Network

DP-16QAM Dual Polarization 16-QAM

DP-QPSK Dual Polarization QPSK

eMBB Enhanced Mobile BroadBand

ENB Evolved NodeB

ENDC E-UTRA/NR Dual Connectivity

EON Elastic Optical Network

EPC Enhanced Packet Core

E-UTRA/NR Evolved Universal Terrestrial Radio Access/New Radio

FDD Frequency Division Duplex

FDMA Frequency Division Multiple Access

GHz Gigahertz

GNB Next Generation NodeB

GNU GPLv2 GNU General Public License version 2

GPRS General Packet Radio Service

GSM Global System for Mobile communications

GTP-U GPRS Tunneling Protocol - User plane

HPC High Performance Computing

HSPA High Speed Packet Access

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronics Engineers

IMT International Mobile Telecommunications

IoT Internet of Things

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ITU International Telecommunication Union

ITU-R ITU Radiocommunication Sector

LAN Local Area Network

LCoS Liquid Crystal on Silicon

LDP Label Distribution Protocol

LTE Long-Term Evolution

MANET Mobile Ad hoc Network

MIMO Multiple Input Multiple Output

mMTC Massive Machine-Type Communication

MPLS Multiprotocol Label Switching

MUX Multiplexor

NIC Network Interface Card

NoC Network on Chip

NR New Radio

NSA Non Stand Alone

OADM Optical Add-Drop Multiplexer

OE Óptico-Electrónicos

OFDM Orthogonal Frequency Division Multiplexing

OFDMA Orthogonal Frequency Division Multiple Access

ONS Optical Network Simulator

OSNR Optical Signal to Noise Ratio

OSPF Open Shortest Path First

OTN Optical Transport Network

OXC Optical Cross-Connect

PC Personal Computer

PDC Personal Digital Cellular

PDCCH Physical Downlink Control Channel

PDSCH Physical Downlink Shared Channel

PF Proportional Fairness

PLI Physical Layer Impairments

PPP Point-to-Point Protocol

PUSCH Physical Uplink Shared Channel

QAM Quadrature Amplitude Modulation

QoS Quality of Service

QoT Quality of Transmission

QPSK Quadrature Phase Shift Keying

RAN Radio Access Network

RB Resource Block

RE Resource Element

RMLSA Routing, Modulation Level, and Spectrum Allocation

ROADM Reconfigurable Optical Add-Drop Multiplexer

RR Round Robin

RSVP-TE Resource Reservation Protocol - Traffic Engineering

RWA Routing and Wavelength Assignment

SA Stand Alone

SAN Storage Area Network

SDM Spatial Division Multiplexing

SMF Session Management Function

TCP Transmission Control Protocol

TDD Time Division Duplex

TDMA Time Division Multiple Access

TRP Total Radiated Power

UE User Equipment

UDP User Datagram Protocol

UDM Unified Data Management

UL Uplink

UPF User Plane Function

URLLC Ultra-Reliable and Low-Latency Communications

VoIP Voice over Internet Protocol

VR Virtual Reality

WAN Wide Area Network

WCDMA Wideband Code Division Multiple Access

WDM Wavelength Division Multiplexing

WiMAX Worldwide Interoperability for Microwave Access

WSS Wavelength Selective Switch

Wi-Fi Wireless Fidelity

Capítulo 1

Introducción

En este capítulo se define el problema abordado, la creación de una herramienta de simulación óptico-móvil. Se dejará en claro la importancia de esta, así como los objetivos generales y específicos de este proyecto. A su vez en la última sección se explica la organización del documento para una mayor comprensión del contenido.

1.1. Motivación

En la era actual, la sociedad se encuentra inmersa en un contexto de creciente dependencia de la conectividad digital. Desde la comunicación cotidiana hasta las complejas operaciones industriales, la necesidad de una conectividad rápida, confiable y de alta capacidad se ha vuelto indispensable. Cada día surgen más casos de uso donde se necesitan mejores condiciones de comunicación por ejemplo telemedicina de alta definición, conducción autónoma, realidad aumentada (AR) y realidad virtual (VR) a gran escala, internet de las Cosas (IoT) a gran escala, manufactura inteligente y automatización industrial, y se podría seguir con muchas más. En estos escenarios, la tecnología 5G emerge como un catalizador clave para satisfacer estas demandas crecientes y hacer posible todas estas cosas que hace algunos años no eran posibles. 5G llegó para transformar radicalmente la forma en que interactuamos con el mundo que nos rodea.

Como es bien sabido, la mejor tecnología de infraestructura para lograr comunicaciones a grandes velocidades y transmitiendo una gran cantidad de datos, son las redes ópticas. Esto se ha demostrado a lo largo de los años desde el inicio de su implementación y se mantienen en continua evolución, por ejemplo, con el surgimiento de las redes ópticas elásticas (EON) que prometen mejoría en la utilización de los recursos.

Si se utilizan redes ópticas elásticas para la comunicación de redes 5G, resulta difícil concebir la magnitud de las ventajas que podrían surgir. Pero como toda la historia de las redes de comunicaciones, antes de dar un gran paso, que también requiere una gran inversión, se realizan muchos prototipos y pruebas.

Este proyecto empuja la frontera del conocimiento e intenta ir un paso adelante, creando una herramienta de simulación óptico-móvil. El valor de la herramienta es inmenso dado que no existen plataformas de simulación óptico-móvil de código abierto y podría ser el primer avance hacia una nueva infraestructura de comunicaciones. Esta herramienta será de gran utilidad para estudiar el comportamiento de la red bajo diferentes sistemas de asignación de recursos, investigar diferentes efectos como la congestión en redes, etc.

Por otro lado, es importante de mencionar que los avances de este proyecto no quedarán en el olvido, pues es parte de un proyecto más amplio de CSIC I+D titulado "Convergencia entre redes 5G/6G y redes ópticas: un enfoque holístico". Este proyecto involucra a muchas más personas, por ejemplo, otros estudiantes creando su proyecto de grado, estudiantes de doctorado, profesores de la UdeLaR expertos en el dominio y otros. La Administración Nacional de Telecomunicaciones (Antel) es uno de los interesados directos del proyecto, que podría hacer una inversión a futuro para crear una nueva infraestructura si se logran buenos resultados. Además, cuenta con el apoyo de la Dirección Nacional de Telecomunicaciones (DINATEL) y otros organismos.

El proyecto CSIC tiene como objetivos desarrollar algoritmos de aprendizaje automático para la asignación de *slices* multidominio tecnológico, estudiar arquitecturas de red definidas por *software* para control holístico de redes 5/6G y redes ópticas elásticas (EON), proponer una arquitectura de monitorización para redes óptico-móviles, evaluar la herramienta de simulación óptico-móvil creada, entre otras. El objetivo final es contar con una plataforma completa de simulación disponible para evaluar los algoritmos de asignación de recursos en un enfoque multitecnológico. La misma se pondrá a disposición de la comunidad en un repositorio público para que llegue a más personas, las cuales podrían aprender de este y/o contribuir para su mejoría.

1.2. Objetivos

Este proyecto tiene como objetivo general crear una herramienta de simulación de redes óptico-móviles que permita probar distintas topologías de este estilo. Esta herramienta debe ser capaz de recopilar estadísticas de las simulaciones obtenidas para poder usarlas en el proyecto CSIC como datos de entrada para crear un algoritmo de aprendizaje automático que asigne recursos.

Para lograr la creación de la herramienta en su totalidad se deben cumplir ciertos objetivos específicos:

- Estudiar el dominio de redes ópticas y redes de telecomunicaciones 5G para entender el problema en profundidad.
- Elegir la plataforma de simulación más adecuada de acuerdo con los requerimientos del dominio.
- Realizar pruebas de convergencia de las redes ópticas y móviles hasta hallar la mejor solución para la herramienta.

- Brindar distintas topologías y dar la posibilidad de utilizar diversas configuraciones.
- Capturar estadísticas de la herramienta.
- Documentar de manera precisa y clara el trabajo realizado para que más personas puedan entenderlo y usarlo como base para futuros avances en el tema.

1.3. Organización del documento

En esta sección se describe brevemente como se estructura el documento para introducir un orden a lo que se está leyendo.

En el capítulo 2 se revisan los antecedentes de entornos de simulación existentes para luego poder utilizar el más adecuado. A su vez se da un marco teórico de las redes ópticas y las redes de telecomunicaciones 5G, con los conceptos básicos requeridos para la comprensión de los siguientes capítulos.

En el capítulo 3 se analizan las plataformas de simulación estudiadas en el capítulo anterior para poder decidir cuál es la más adecuada para las necesidades del proyecto.

En el capítulo 4 se describe el desarrollo de la herramienta en su totalidad. Comienza describiendo cómo funciona el entorno de simulación elegido, luego las pruebas realizadas y las decisiones tomadas paso a paso para terminar la herramienta con éxito.

En el capítulo 5 se explica cómo usar la herramienta creada. Esto implica cómo correr una simulación, qué parámetros se pueden configurar, cómo sacar estadísticas y tres simulaciones de ejemplo con sus respectivas conclusiones.

En el capítulo 6 se detalla la ingeniería de software de todo el proyecto. Habla tanto de la organización del proyecto como de cuánto tiempo fue dedicado a cada una de las actividades de un proyecto de software.

Finalmente, en el capítulo 7 se describen las conclusiones generales del proyecto y trabajos futuros. A su vez luego en los anexos se describen detalles más técnicos para quien esté interesado en conocer un poco más a bajo nivel.

Capítulo 2

Revisión de antecedentes

En este capítulo se presenta toda la información que se revisó para crear la herramienta. Esto incluye entre tantas cosas:

- Bibliografía con respecto a ambos dominios (redes 5G y redes ópticas).
- Trabajos previos relacionados a uno de los dos dominios.
- Herramientas de simulación existentes.
- Avances en desarrollos que sirvan para la creación de la herramienta.

Como se mencionó anteriormente, este proyecto formó parte de una iniciativa más amplia que abarcó diversos conceptos no tradicionalmente incluidos en el currículo de ingeniería en computación, específicamente en áreas como las redes ópticas y las redes móviles 5G y 6G. Por ende, fue necesario realizar un estudio exhaustivo para adquirir una comprensión sólida del contexto.

La metodología adoptada involucró la realización de seminarios semanales o quincenales con la participación de todos los miembros del proyecto CSIC. Cada seminario se enfocaba en un tema específico del cual el presentador era experto, brindando su conocimiento y experiencia sobre el tema. Dentro de los seminarios se presentaron algunos trabajos previos realizados por los expertos, por ejemplo [15].

Los seminarios abordaron cuestiones como redes ópticas, telecomunicaciones e introducción al aprendizaje automático. Además, se exploraron propuestas de plataformas de simulación, siendo OMNeT++ la elegida para crear la convergencia de redes ópticas y de telecomunicaciones dentro de este proyecto. Asimismo, se presentó GNpy como un simulador de redes ópticas integral que proporciona mucha información detallada sobre fibras, retardos, entre otros aspectos relevantes.

A continuación se dará una breve introducción a los conceptos estudiados en base a estos antecedentes y bibliografía recomendada. Más adelante se hablará específicamente de las herramientas de simulación ya existentes estudiadas.

2.1. Marco teórico

En esta sección se explicarán todos los conceptos del dominio en que se trabajó, para una correcta comprensión del proyecto en su totalidad. Estos comprenden los principales temas referidos a cómo funcionan las redes móviles, en especial 5G y las redes ópticas.

2.1.1. Conceptos básicos

Espectro radioeléctrico:

El espectro radioeléctrico refiere a una porción del espectro electromagnético que abarca el rango de frecuencias de las ondas que se utilizan para las comunicaciones inalámbricas. Estas ondas electromagnéticas viajan a través del aire y pueden transportar datos, voz, video y otros tipos de información.

El espectro radioeléctrico incluye desde las frecuencias muy bajas utilizadas para la radiodifusión AM hasta las frecuencias extremadamente altas utilizadas para las comunicaciones por satélite y las tecnologías inalámbricas de alta velocidad, como el Wi-Fi y el Bluetooth.

A medida que avanza la tecnología, se van utilizando cada vez más rangos de frecuencia y por ende el espectro radioeléctrico usado va aumentando su tamaño. En la figura 2.1 se aprecia el espectro electromagnético en su totalidad y la parte que corresponde al espectro radioeléctrico.

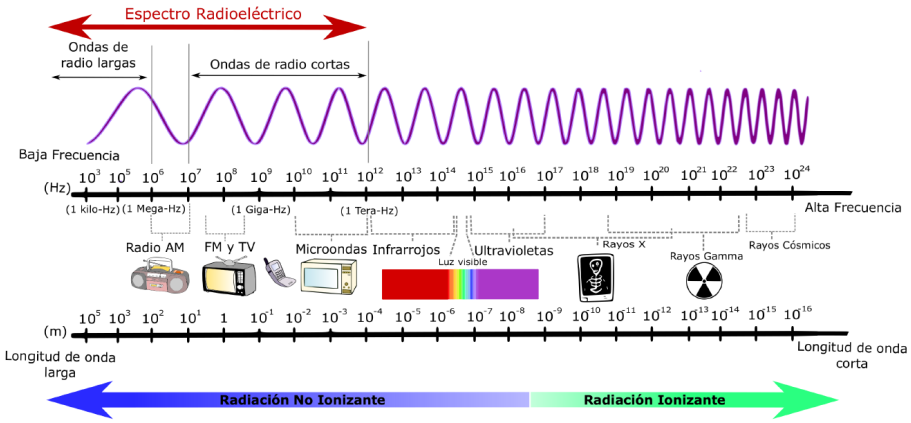

Figura 2.1: Espectro electromagnético y espectro radioeléctrico [14].

Actualmente la UIT (Unión Internacional de Comunicaciones) define el rango de frecuencias que comprende el espectro radioeléctrico es de 0 GHz a 3000 GHz. En la práctica ondas de frecuencia demasiado bajas no se usan por su limitada capacidad de transmitir información. Por otro lado, las ondas con frecuencias más altas no se han explorado mucho, pero se sabe que tienen una

mayor capacidad de transmisión de datos, pese a una menor cobertura. A modo de ejemplo y dando un pequeño adelanto, las comunicaciones 5G utilizan varias porciones del espectro, en particular debido a su alta capacidad utilizan las frecuencias conocidas como bandas milimétricas (en el entorno de los 30 GHz). Pero esta gran capacidad tiene como contrapartida la menor área de cobertura de las radiobases, implicando que estas deban estar más próximas y en más cantidad que en las redes LTE.

Como resumen, las comunicaciones inalámbricas utilizan frecuencias del espectro para transmitir datos. A continuación se explica las distintas maneras de transmisión por el medio físico una vez que se tiene un rango de frecuencia del espectro asignado y se hará énfasis en cual técnica se utiliza particularmente para 5G.

Técnicas de acceso al medio

Dado que existen muchos usuarios que desean transmitir y recibir datos, hay que organizarlos de una manera para que puedan compartir el medio de difusión.

A partir de ahora se le llamará tráfico *uplink* a cuando el usuario transmite datos, por lo que la comunicación va desde el usuario a la radiobase. Por el contrario, se le llamará tráfico *downlink* a cuando el usuario recibe datos, o sea la información se transmite desde la radiobase al usuario.

En comunicaciones el término duplexación refiere a la capacidad de transmitir y recibir datos simultáneamente en un sistema (comunicación bidireccional). A su vez existen distintas técnicas de acceso al medio compartido. Entre ellas se encuentran las que se basan en la división de frecuencias, división del tiempo y otras más complejas:

- FDD (*Frequency Division Duplex*): Se refiere a una técnica de transmisión que divide el espectro de frecuencia en dos bandas separadas: una para el *uplink* y otra para el *downlink*. Esto significa que ambas operaciones (*uplink* y *downlink*) ocurren simultáneamente.

- TDD (*Time Division Duplex*): En TDD, una única frecuencia se utiliza para ambas direcciones de comunicación (*uplink* y *downlink*), pero en diferentes momentos. El tiempo se divide en intervalos, durante los cuales se alternan los períodos de *uplink* y *downlink*. Esto permite que la misma frecuencia se utilice eficientemente en ambas direcciones sin necesidad de dividir el espectro en frecuencias separadas para *uplink* y *downlink*, como en FDD.

- FDMA (*Frequency Division Multiple Access*): Es una técnica de acceso múltiple en la que el espectro de frecuencia se divide en múltiples canales separados, y cada usuario utiliza un canal diferente para comunicarse. Esto permite que varios usuarios compartan el mismo medio de transmisión simultáneamente.

- TDMA (*Time Division Multiple Access*): En esta técnica el tiempo se divide en intervalos y cada usuario utiliza un intervalo diferente para transmitir datos. Esto permite que varios usuarios compartan el mismo medio de transmisión secuencialmente en lugar de simultáneamente.

- CDMA (*Code Division Multiple Access*): Es una técnica de acceso múltiple en la que los usuarios comparten el mismo rango de frecuencias simultáneamente, pero cada usuario utiliza un código único para modular su señal. Esto permite que múltiples usuarios compartan el mismo espectro de frecuencias sin interferirse mutuamente.

- OFDMA (*Orthogonal Frequency Division Multiple Access*): Es una técnica de acceso múltiple que combina la multiplexación por división de frecuencia ortogonal (OFDM) con el acceso múltiple. Divide el espectro de frecuencia en subportadoras ortogonales y asigna grupos de subportadoras a diferentes usuarios para transmitir datos simultáneamente. Esta técnica es comúnmente utilizada en tecnologías de comunicación inalámbrica de banda ancha, como LTE y WiMAX. Dado que es la técnica utilizada en 5G a continuación se explica con más detalle cómo funciona.

Comunicaciones inalámbricas en 5G: OFDM, RE y RB

Aquí se explica un poco más cómo funciona el acceso al medio en comunicaciones inalámbricas, particularmente en OFDM y se mencionaran algunos conceptos relacionados como RE y RB. No se entrarán en detalles muy técnicos para no desviarse de lo que es más importante. Se sugiere leer el anexo A para un mayor entendimiento de qué es un símbolo, constelación, etc.

Para la transmisión de datos por la magnitud física se realiza un mapeo de bit a amplitud (símbolo). OFDM utiliza muchas subportadoras en vez de una, por lo que envía muchos símbolos en paralelo, solucionando el problema de multicamino (referirse al anexo A). Se realiza un mapeo de qué símbolo manda cada portadora en cada instante de tiempo.

La transmisión se maneja en cuadros de tiempo (T), los cuales son de 10 ms. A su vez los cuadros se dividen en subcuadros de 1 ms, y estos se dividen en ranuras (*slots*) temporales. En 5G 14 símbolos OFDM forman un *slot* pero como la duración de símbolos depende de la separación entre subportadoras (numerología) entonces la cantidad de *slots* por subcuadro es variable. En la figura 2.2 se aprecia.

En las redes móviles 5G un *resource block* (RB) es la unidad básica de asignación de recursos de frecuencia y tiempo en el dominio de la radio. Es la unidad mínima de la banda de frecuencia en las redes móviles 5G que se asigna a un usuario móvil para la transmisión y recepción de datos. Cada *resource block* consta de 12 subportadoras consecutivas en el dominio de la frecuencia y 14 símbolos. Los dispositivos móviles pueden ser asignados uno o varios *resource blocks*, dependiendo de la cantidad de ancho de banda que se requiere para una transmisión específica.

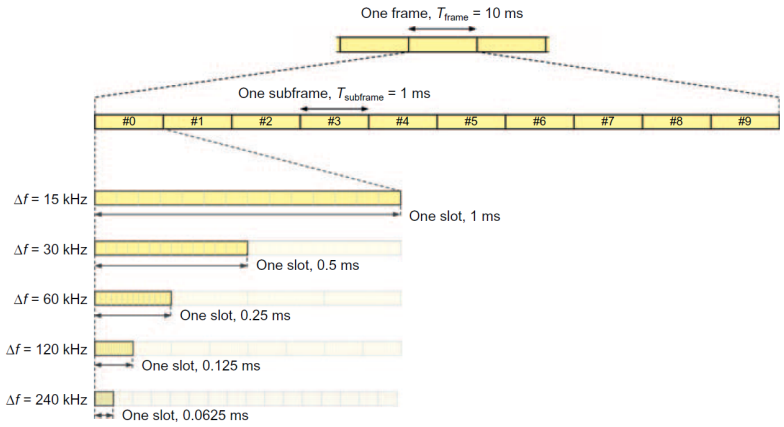

Figura 2.2: División de subcuadros en *slots* [6].

A su vez un *resource element* (RE) es también una unidad importante que se define como una subportadora con duración de un símbolo OFDM. Se puede representar con un tupla (k,l) donde k el índice de dominio de frecuencia y l es la posición del símbolo en el dominio temporal.

2.1.2. Redes móviles

Teniendo una base de cómo funcionan las comunicaciones inalámbricas, es hora de adentrarse en las redes móviles y particularmente 5G.

Resulta relevante entender el surgimiento de 5G a partir de sus antepasados. Por esta razón se mencionará brevemente la historia de las redes celulares. En las últimas cuatro décadas, se han desarrollado cuatro generaciones de comunicaciones móviles:

- Años 80': Basada en transmisión analógica.
- Años 90': Transmisión digital, con tecnologías como GSM (*Global System for Mobile communications*), D-AMPS (*Digital Advanced Mobile Phone System*), PDC (*Personal Digital Cellular*), y CDMA.
- Década del 2000: Introducción de la tecnología 3G, que proporcionaba acceso inalámbrico más rápido y de mayor calidad, destacando HSPA (*High Speed Packet Access*).
- 2010s: Desarrollo de LTE, una tecnología más eficiente con mayor capacidad de datos, soportando tanto FDD como TDD y tecnologías de antenas avanzadas.
- 5ta Generación: Evolución de LTE, destacando por ser más eficiente y ofrecer mayor capacidad de datos.

La estandarización fue clave para el éxito de las comunicaciones móviles. En el 98, las organizaciones que estaban desarrollando CDMA, se juntaron y crearon el 3GPP (*3rd Generation Partnership Projec*) para terminar el 3G, basado en WCDMA (*Wideband CDMA*).

Luego se creó un 3GPP2 para desarrollar CDMA2000 pero dominó 3GPP y se continuó con el desarrollo de 4G y 5G.

5G-NR

La quinta generación conocida como 5G surge debido al continuo crecimiento en la demanda de datos, donde el LTE ya no es suficiente para afrontar las necesidades actuales. Para las comunicaciones 5G se utiliza la tecnología NR, un nuevo acceso de radio que viene como una mejora del LTE conocida como *New Radio*.

5G es especificado por el *3rd Generation Partnership Project* (3GPP) y diseñado para proporcionar velocidades de datos más rápidas, menor latencia y mayor capacidad para soportar una amplia gama de aplicaciones y dispositivos conectados. *New Radio* utiliza una serie de tecnologías avanzadas, como antenas masivas de entrada múltiple y salida múltiple (MIMO), acceso múltiple por división de frecuencia ortogonal (OFDM), y la agregación de portadoras para lograr estas mejoras en rendimiento y eficiencia. A su vez NR permite numerologías OFDM variables. Variar la numerología implica variar la separación de subportadoras, como el número de subportadoras es constante, esto significa que varía el ancho de banda total. Para obtener más información, referirse a la maestría [15] donde también incluyen ejemplos para una mejor comprensión.

Reutiliza muchas estructuras del LTE, pero una de sus diferencias es que al ser una nueva tecnología de radio no se restringe a la necesidad de ser compatible con tecnologías del pasado (lo cual limitaría su potencial). Sin embargo NR es compatible con la evolución de las redes 4G (LTE) ya que permite el uso de *Discrete Fourier Transform spread* OFDM (DFT-OFDM) en el UL. Además, está diseñado para ser flexible y escalable, lo que permite a los operadores de redes móviles adaptarse a las necesidades cambiantes de los usuarios y aplicaciones. Para más información referirse a [6].

5G tiene tres casos de uso definidos por la ITU-R en la recomendación M.2083-0 [17]:

- *Enhanced Mobile BroadBand* (eMBB): Refiere a tasas altas de datos, y mucho volumen de tráfico. Un ejemplo de donde se necesita es en áreas con mucha densidad de usuarios donde los usuarios necesitan una conexión de alta velocidad.

- *Massive Machine-Type Communication* (mMTC): Muchos dispositivos y tipos de dispositivos distintos por ejemplo sensores, actuadores, etc. También incluye soporte para IoT (*Internet of things*). Requiere bajo coste y consumo de energía de los dispositivos.

- *Ultra-Reliable and Low-Latency Communications* (URLLC): Muy poca latencia, muy fiable y disponible para satisfacer los requerimientos de la actualidad. Algunos ejemplos incluyen automatización de fábricas y cirugías medicas a distancia.

Estos casos de uso se crearon para distinguir las distintas clases de requerimientos, pero no son mutuo exclusivos, ni exactos. O sea pueden darse combinaciones entre ellos y no incluir todas las características de uno. Un ejemplo es si se quiere baja latencia y una alta tasa de datos.

Bandas de espectro:

Las bandas para la primera y segunda generación eran frecuencias de alrededor de 800-900 MHz. 3G se basó en bandas sobre los 2GHz y de ahí nuevas bandas (bajas y altas) fueron añadidas, abarcando desde los 450 MHz hasta 6 GHz

Bandas con menor frecuencia son mejores para cubrir áreas extensas, urbanas, suburbanas y entornos rurales. La propagación de frecuencias altas hace que sea difícil de usarlas en estos escenarios. Estas altas frecuencias han sido más utilizadas para mejorar las capacidades de despliegues densos.

El escenario eMBB requería mejores tasas de datos, mejores capacidades. En algunos lados, donde los requerimientos son de tasas de datos extremas y con mucho tráfico, se considera el uso de hasta 60 GHz.

Se asignan rangos distintos de frecuencia para subida y bajada (*paired bands*) y también *unpaired bands* con un solo rango de frecuencias compartido para *uplink* y *downlink*. Las *paired bands* usan FDD, mientras que las *unpaired bands* usan TDD.

Requisitos de 5G

La Unión internacional de Telecomunicaciones (ITU) utiliza el término IMT-2020 para referirse a la visión y los requisitos de la próxima generación de redes de comunicaciones móviles, conocidas como 5G. Describe así ocho requisitos a cumplir:

- *Peak Data Rate:* Máxima tasa de datos conseguible bajo condiciones ideales. Va a depender de la cantidad de espectro disponible para el despliegue del operador, además de la eficiencia espectral. Se espera que frecuencias más altas lleven a un mayor *peak data rate*. Particularmente los requerimientos mínimos son de 20 Gbit/s para *downlink* y 10 Gbit/s para *uplink*.

- *User Experienced Data Rate:* La tasa de datos que puede conseguir cubriendo una gran parte de los usuarios. Se define un valor objetivo de 100Mbit/s para áreas urbanas/suburbanas, se espera que las 5G puedan llegar a dar 1Gbit/s *indoor*.

- Eficiencia espectral: Promedio de rendimiento de datos por hertzio de espectro y por celda o por unidad de equipo de radio (TRP). El objetivo se estableció en 3 veces más que la eficiencia espectral de 4G, pero dependerá del escenario de implementación.
- *Area traffic capacity:* Depende de la eficiencia espectral y de la banda ancha disponible, y cuán densa es la red.
- *Network energy efficiency:* El valor objetivo está puesto en que no puede ser más grande que las redes desplegadas hoy en día. Esto quiere decir que la energía usada debe disminuir la suficiente para compensar el aumento del tráfico previsto.
- *Latency:* El tiempo desde que la fuente envía hasta que el destinatario recibe. En concreto, se anticipa que la latencia sea diez veces menor que la establecida por IMT-Advanced.
- *Mobility:* Objetivo de 500km/h para trenes de alta velocidad, es más que IMT-Advanced. Es una capacidad clave en el escenario URLLC en caso de comunicación crítica con vehículos a alta velocidad, y en simultáneo la latencia.
- *Connection density:* Número total de dispositivos conectados por unidad de área. Se tiene como requerimiento mínimo 1.000.000 dispositivos por km^2.

Por otro lado, existen cinco ambientes para ser evaluado.

- Punto de acceso interior eMBB: Oficinas, shopping mall. Peatones y usuarios estacionarios con una gran densidad de usuarios.
- Urbano Denso eMBB: Mucha densidad de usuarios y el tráfico se basa en los usuarios peatonales y vehiculares.
- Rural eMBB: Área extensa y con cobertura amplia. Soporta peatonales, vehiculares y de alta velocidad.
- Urbano Macro mMTC: Dispositivos de tipo máquina, un gran número.
- Urbano Macro URLLC: Ultra fiable y baja latencia.

Se evalúan en base a simulaciones, análisis e inspecciones, dependiendo del requerimiento.

Despliegue

Existen dos modalidades de despliegue: Non Stand Alone (NSA) y Stand Alone (SA).

En NSA se pueden crear distintos escenarios, uno de ellos puede implicar instalar radiobases 5G (gnb) en una red LTE y otro escenario sería con radiobases

LTE (enb) en una red 5G ya implementada. Esta modalidad permite despliegues rápidos. SA por otro lado, define todas las radiobases de la misma tecnología y a su vez se implementan todas las funcionalidades definidas en 5G.

En la figura 2.3 se logran visualizar distintas combinaciones de SA y NSA. Esta imagen fue obtenida de [28] La primera arquitectura normalizada en la *Release 15* del 3GPP es la opción 3, la cual es una arquitectura NSA. La misma usa *Enhanced Packet Core* (EPC) como núcleo y dos radiobases (una enb y una gnb). La radiobase enb actúa como celda primaria (pCell) manejando la señalización de los ues. Mientras que el gnb actúa como celda secundaria (pCell) con el objetivo de aumentar la capacidad de transmisión de datos. Las opciones 1,2 y 5 son algunas combinaciones posibles de SA, la primera en particular corresponde a una arquitectura de 4G.

Cualquiera sea la arquitectura, se puede ver que todas comprenden al menos un núcleo (EPC o 5GCN), una radiobase (gnb o enb) y un user equipment (ue).

A modo de adelanto, la herramienta de simulación óptico-móvil se creó a partir de una red SA como la de la opción 2. Se empezó con una arquitectura de una única gnb como celda primaria y luego se extendió a dos gnbs. Comprendía a su vez un núcleo 5GCN y una cantidad variable de ues. Referirse al capítulo 4 para continuar sobre este tema.

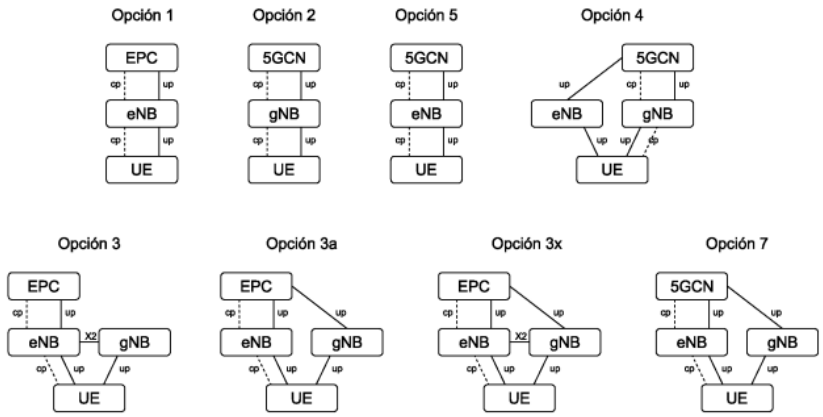

Figura 2.3: Combinaciones de distintos despliegues NSA y SA [28].

A continuación se explica más sobre el núcleo de las arquitecturas 5GCN, dado que comprende algunos conceptos fundamentales para la construcción de la herramienta de simulación óptico-móvil.

5GCN

El núcleo *5G Core Network (5GCN)* es la parte central de la arquitectura de red de la tecnología 5G. Se encarga de gestionar las funciones de control y gestión de la red. Se compone de varios elementos funcionales clave que trabajan juntos para permitir una conectividad de alta velocidad, baja latencia y una mayor capacidad para una variedad de casos de uso. Algunos de los componentes principales de la *5G Core Network* incluyen:

- *Session Management Function* (SMF): Esta unidad es responsable de establecer y gestionar las sesiones de datos para los usuarios. Es crucial para la entrega eficiente de servicios en tiempo real y el control de la calidad del servicio (QoS) en la red 5G.

- *Access and Mobility Management Function* (AMF): Maneja la autenticación y la autorización de los usuarios, así como la movilidad, permitiendo que los dispositivos se conecten y desconecten de manera transparente mientras se mueven entre diferentes celdas y redes.

- *Unified Data Management* (UDM): Almacena y gestiona la información de suscripción de los usuarios, incluidos los perfiles de servicio y la información de autenticación, lo que permite la autenticación y la autorización eficientes de los usuarios en la red.

- *Network Exposure Function* (NEF): Proporciona una API (interfaz de programación de aplicaciones) para permitir la integración de servicios y aplicaciones de terceros en la red 5G. Esto facilita el desarrollo de nuevos casos de uso y servicios innovadores que aprovechan la capacidad y la flexibilidad de la red.

- *User Plane Function* (UPF): Es un componente esencial de la *5G Core Network* que juega un papel fundamental en el enrutamiento y la gestión del tráfico de datos. Es responsable de la gestión de la política de red, la entrega de datos de usuario y la aplicación de funciones de red específicas según sea necesario. Proporciona el punto de interconexión entre la infraestructura móvil y la Red de Datos (DN), es decir, el encapsulado y desencapsulado de los datos del usuario en el protocolo de tunelización GPRS para los paquetes del plano de usuario (GTP-U).

Todos los estos elementos mencionados son parte del plano de control de la red, salvo el UPF que pertenece al plano de usuarios. Se hace un gran énfasis en este último por ser de los más importantes a la hora de la creación de la herramienta de simulación.

El UPF es responsable de enrutar y gestionar el tráfico de datos en la red 5G. Es necesario tener al menos un UPF en todas las redes móviles, a veces se conectan más de uno seguidos para mejorar el rendimiento de la red. Algunas de sus funciones principales incluyen:

- Enrutamiento de paquetes: La UPF determina la ruta más eficiente para el tráfico de datos, asegurando la entrega rápida y confiable de los paquetes de datos.
- Gestión de políticas de red: Aplica políticas de red específicas para controlar el flujo de datos, la asignación de recursos y la calidad del servicio según los requisitos del servicio y las necesidades del usuario.
- Gestión de sesiones: La UPF mantiene y gestiona las sesiones de datos para garantizar una conectividad continua y estable para los usuarios, incluso mientras se mueven entre diferentes celdas y redes.
- Optimización de la entrega de contenido: Puede realizar funciones de optimización de la entrega de contenido, como el almacenamiento en caché y la compresión de datos, para mejorar la eficiencia y el rendimiento de la red.

En la figura 2.4 se puede apreciar todos los elementos de 5GCN y como se relacionan en una arquitectura 5G completa.

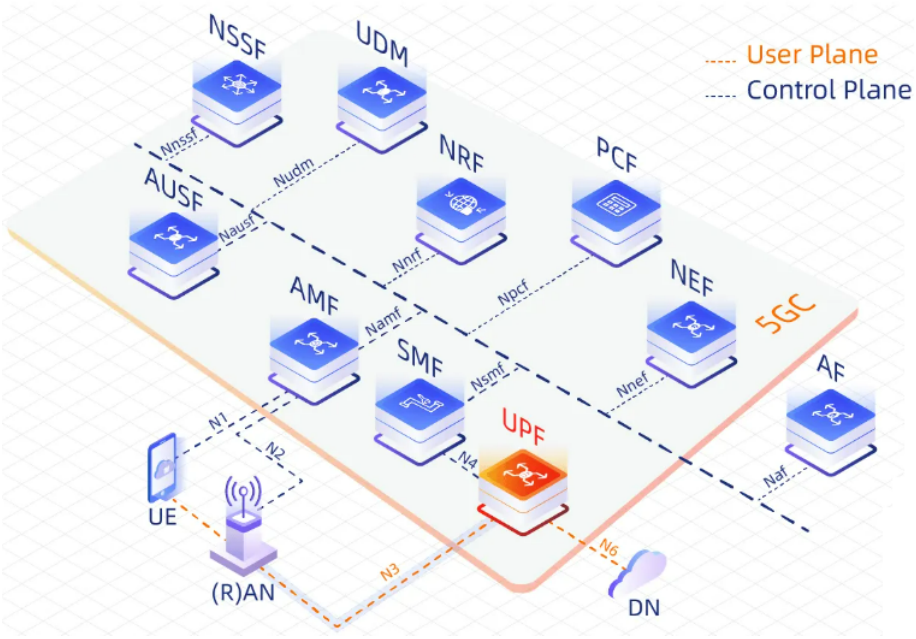

Figura 2.4: Elementos de 5GNC en arquitectura 5G [16].

Uplink y *downlink* en 5G

Para establecer una comunicación entre radiobase y *user equipment* (ue) primero existen algunos pasos a seguir, los cuales utilizan determinados mensajes

de control (DCI y UCI) dependiendo si la comunicación es *uplink* o *downlink*.

Para la comunicación *downlink* (desde radiobase a ue) se utiliza el Canal de control PDCCH para enviar información de control desde la radiobase a los dispositivos móviles. Esta información de control incluye instrucciones para la asignación de recursos de enlace descendente (por ejemplo, asignación de subportadoras, programación de transmisiones de datos) y también puede incluir indicaciones sobre la configuración y el ajuste de parámetros de la conexión inalámbrica. Esta información es trasmitida por el mensaje de información de control para bajada (DCI). Finalmente el ue recibe la información a través del canal de datos físico en enlace descendente (PDSCH).

Para la comunicación *uplink* (de ue a radiobase) el ue manda un mensaje de control de subida (UCI) a la radiobase para avisarle su deseo de subir contenido. Esta última luego de recibir dicho mensaje utiliza el DCI para notificar al ue los recursos que puede transmitir y parámetros de transmisión que debe utilizar. Finalmente el ue transmite sus datos por del canal físico en enlace ascendente (PUSCH).

2.1.3. Redes ópticas

Las redes ópticas tienen muchas ventajas en comparación con las redes de cable coaxial, entre ellas se tienen:

- Bajo costo a largo plazo: La fibra óptica tiende a requerir menos mantenimiento debido a su mayor resistencia a las condiciones ambientales. Esto reduce los costos asociados con la reparación y el reemplazo de componentes. Lo que significa que tiene una vida útil más larga que el cable coaxial.

- Velocidad de transmisión: Mientras que el cable coaxial puede ofrecer velocidades de hasta varios gigabits por segundo, la fibra óptica puede proporcionar velocidades de hasta varios terabits por segundo, lo que la hace ideal para aplicaciones que requieren una gran cantidad de ancho de banda, como la transmisión de vídeo de alta definición, videoconferencias y servicios de *streaming*.

- Distancia de transmisión: La fibra óptica puede transmitir señales a distancias mucho mayores que el cable coaxial sin experimentar pérdida de señal significativa. Esto la hace ideal para su uso en redes de larga distancia, como las redes de área amplia (WAN), donde se necesitan conexiones estables y confiables a través de distancias extensas.

- Confiabilidad: La fibra óptica es inmune a las interferencias electromagnéticas, a diferencia del cable coaxial que puede verse afectado por interferencias de radiofrecuencia y electromagnéticas. Esto hace que la fibra óptica sea más confiable en entornos donde hay muchas fuentes de interferencia, como entornos industriales o áreas urbanas densamente pobladas.

Sin embargo, por más que un despliegue de redes ópticas sea más conveniente a largo plazo, cambiar todo el despliegue del pasado de una única vez es una inversión demasiada costosa. Por este motivo se comenzó incorporando las redes ópticas dentro de redes de comunicaciones ya existentes, debiendo resolver la coexistencia de ambos tipos de redes.

Se le llaman redes transparentes, a las redes ópticas que evitan el pasaje por nodos intermediarios Óptico-electrónicos (OE). Realizan en *switching* a nivel óptico, lo que se hablará más adelante en la subsección 2.1.3. Se le llama redes opacas a las que requieren conversiones OE. Transforman la señal óptica en señal eléctrica para interpretarla y luego esta se reenvía pero con algunas leves diferencias.

Las redes transparentes evitan las transformaciones óptico-electro-óptico que crean cuellos de botella, aumentan el retardo y costo en la red por la electricidad. Sin embargo por más que tener redes 100 % transparentes sea lo mejor, el pasaje se hace gradualmente y mientras tanto se necesita esa conversión.

Arquitectura óptica *end-to-end*

El funcionamiento de las redes ópticas para transmitir información de un origen hacia un destino se da a través de la multiplexación de muchas señales en una sola. Esta arquitectura que se explicará a continuación se llama DWDM (*Dense Wavelength Division Multiplexing*) y se ilustra en la figura 2.5.

Primero se asigna a cada usuario transmisor un rango de espectro determinado. Cada uno de estos rangos de espectro se les llama canal, color o *lightpath*. Luego un multiplexor (MUX) recibe todas estas señales y las combina en una sola señal como haz de luz, la cual es enviada por la fibra óptica. La fibra cuenta con amplificadores ópticos en línea, por ejemplo el DCM (*Dispersion Compensation Module*) que sirven para compensar la dispersión cromática de la fibra óptica (concepto que se retomará más adelante). Esta señal llega a un demultiplexor (DEMUX) y este descompone dicha señal en los distintos canales originales y los asigna a cada dispositivo destino.

El concepto de *slot* aquí es distinto al de redes móviles. Un *slot* en redes ópticas es lo mismo que un canal o color. El ancho del canal queda determinado por una división de rangos de frecuencias fijo en una frecuencia central. Esto se da porque se usa una grilla fija de la que se hablará más en detalle a continuación.

Clasificaciones de las redes ópticas

Ya se mostró una arquitectura *end-to-end* donde se explica el comportamiento básico de los elementos de la red, pero ahora ¿cómo se realizan los despliegues reales de redes de gran escala en el mundo? Dependiendo de la escala que se esté manejando existen distintos tipos de red.

Las redes cores conectan grandes regiones. Suelen ser redes de alta velocidad y están diseñadas para ser altamente confiables y escalables. Utilizan dispositivos y enlaces de alta capacidad para manejar grandes volúmenes de tráfico de datos.

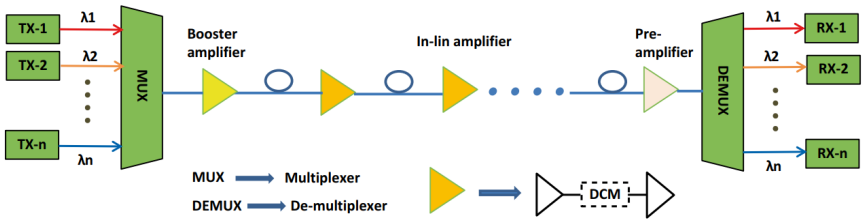

Figura 2.5: Arquitectura DWDM [22].

Las redes metro, como se puede deducir por su nombre, cubren un área geográfica más limitada que suele ser una ciudad o un área metropolitana. Suelen ofrecer velocidades de transmisión más altas que las redes de área local (LAN) convencionales, pero no alcanzan las velocidades de las redes core. También interconectan edificios, empresas, universidades, etc.

Las redes de alcance son las que conectan la red metro con el cliente final. Estos tres segmentos logran juntos la comunicación, cada uno con distintas funcionalidades y capacidad. Las redes de alcance conectan al usuario final con la red metro y esta a su vez concentran el tráfico recibido para enviarlo a las redes core.

Existe una relación jerárquica entre los *routers* de metro y los *routers* de tránsito (de red *core*) para disminuir el número de puertos utilizados. Los *routers* de metro se encargan de la agregación de datos mientras que los *routers* de tránsito realizan el enrutamiento. A su vez los *routers* de tránsito se los coloca junto a un *optical cross-connect* (OXC) que tiene la misma funcionalidad de hub pero a nivel óptico.

Es sabido que la asignación de caminos en la red es un problema con solución conocida, que se puede resolver con distintos algoritmos como Dijkstra [12] o Bellman-Ford [10]. Sin embargo, los *routers* de la red óptica tienen más complejidades pues no basta con analizar la topología para la asignación de caminos. También se debe asignar un *lightpath* y eso implica que exista el ancho de banda suficiente en el espectro en el camino a asignar.

Grilla fija y flexible

La parte específica del espectro electromagnético que se usa para transmitir las redes ópticas se llama *C-band*. Esta se encuentra en el rango de longitudes de onda entre aproximadamente 1525 nm y 1550 nm. La ITU (International Telecommunication Union) definió estándares para dividir la *C-band* en distintos *slots* fijos (lo que se llama la grilla fija), usualmente espaciados por 50 o 100 GHz.

En la figura 2.6 se puede ver un ejemplos de la grilla fija dividida en *slots* de 50GHz. Mientras más ancho es el *slot* más información se puede transmitir, por ejemplo con un *slot* de 33GHz se pueden transmitir hasta 100Gb/s. Cada *slot* es

asignado a un canal, lo que significa que no está disponible para otros canales. A su vez cada canal puede estar rodeado por *guard-bands*, que son bandas de frecuencia en los extremos del canal. Los demultiplexadores pueden causar daño con los filtros que utilizan para interpretar la señal, de esta manera el objetivo de estas bandas es proteger este daño causado por los filtros y el daño causado por la propia interferencia entre canales (*cross-talk*).

Como se puede visualizar en la imagen existen dispositivos que no necesitan más banda ancha y otros en lo que la banda ancha asignada no les es suficiente. La grilla fija no es el mejor método de asignación de recursos. Existen dispositivos que trasmiten poco y aun así tienen 50 GHz asignados, desperdiciando la mayoría del canal que le fue asignado. Por otro lado, existen dispositivos con demandas que se sobresalen de lo estipulado. En estos casos se asignan más canales (a esto se le llama crear un *superchannel*) pero igualmente se desperdician recursos y tendrán una fuerte penalización de los filtros.

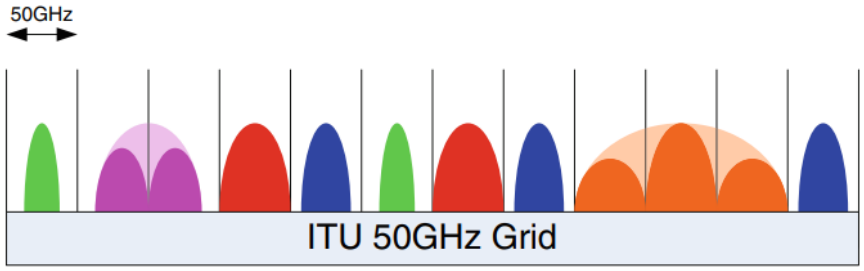

Figura 2.6: Grilla fija [22].

La solución son lo que se llaman *Elastic Optical Netwotk* (EON), donde surge una nueva idea de una grilla flexible. La grilla flexible, como su nombre lo indica, consiste en dar una mayor flexibilidad en la división de la *C-band*. Esto implica que el espectro se puede dividir de manera variable de acuerdo con la demanda, en vez de una repartición pareja como en la grilla fija. Para ello se utilizan canales de anchos de bandas chicas (por ejemplo 12,5 GHz) y concatenarlos a demanda. De esta manera se soluciona el problema si un dispositivo requiere mucho ancho de banda y a su vez se optimiza la red para que los dispositivos que no utilizan tanto ancho de banda no dispongan de más de lo necesario. La mayor diferencia en casos de alta demanda es que si un dispositivo necesita mucho ancho de banda al crear un *superchannel* con la grilla fija existen las *guard-bands* intermedias entre cada slot. En la grilla flexible al no tener las *guard-bands* en el medio, el uso del espectro es mucho más eficiente. En la figura 2.7 se observa el canal divido como grilla flexible.

Sin embargo para implementar las EON se necesita más que eso. Hasta ahora con la grilla flexible los *transponders* tenían un *bit rate* fijo, por ejemplo 10Gb/s o 40Gb/s. Así que era necesario unos *transponders* que pudieran ajustar su ancho de banda rápidamente. Estos *transponders* se llamaron BVT (*Bandwidth*

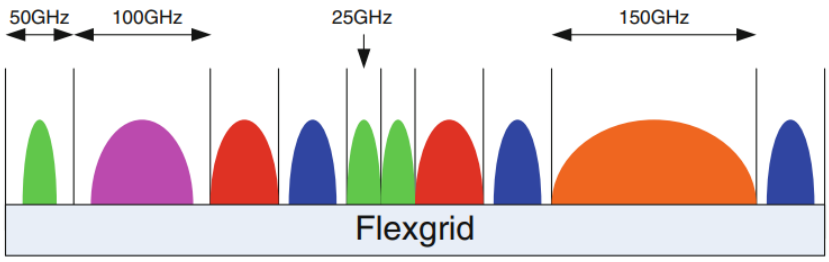

Figura 2.7: Grilla flexible [22].

Variable Transponder) y logran su objetivo cambiando el formato de modulación controlados por un *software*. La finalidad de la modulación es establecer el método para llevar información a través de una onda de luz. Este proceso implica el envío de símbolos mediante la variación de la amplitud, la frecuencia o la fase de la onda. Los BVT presentan además la ventaja de poder compensar las deficiencias del espectro eligiendo el formato de modulación adecuado para cada caso, de esta manera con un solo *transponder* se pueden mitigar muchos escenarios distintos. Algunos ejemplos de formatos de modulación son:

- QPSK - *Quadrature Phase Shift Keying*.
- QAM - *Quadrature Amplitude Modulation*.
- DP-16QAM - *Dual Polarization 16-QAM*.
- DP-QPSK - *Dual Polarization QPSK*.

Switching óptico

Para aumentar la capacidad de las redes ópticas *end-to-end* DWDM hacia redes más complejas con *switching*, se usaron 2 tipos de *switches* ópticos.

- OADM (*Optical Add-Drop Multiplexers*).
- ROADM (*Reconfigurable Optical Add-Drop Multiplexers*).

Los ROADMs son más recientes y se pueden configurar remotamente. Están compuestos por un WSS (*Wavelength Selective Switch*) y un componente *Add/-Drop*. El WSS es un componente fundamental, suele tener una fibra de entrada que tiene muchas señales de frecuencia de la *C-band*. Su objetivo es redirigir cada señal hacia una o varias fibras de salida. Sin embargo los WSS requieren una capacidad extra para trabajar en una grilla flexible ya que los canales pueden variar su tamaño. Para ello se incorpora un dispositivo de cristal líquido sobre silicio (LCoS) en los WSS para permitir el *switching* adecuado. Este dispositivo usa el principio holográfico para permitir un filtrado de banda ancha flexible junto con conmutación y enrutamiento de longitud de onda.

El componente *Add/Drop* posibilita la gestión dinámica del tráfico óptico, permitiendo la incorporación de nuevas conexiones a la red óptica y la eliminación de aquellas que deben ser dirigidas fuera de dicho nodo hacia la capa de enlace. A su vez su configuración se realiza a través de software especializado, lo que permite ajustar los anchos de banda de cada canal, establecer los rangos de frecuencia correspondientes a cada uno y definir los puertos de salida asociados.

Deficiencias de la fibra óptica

Si bien la transmisión por fibra óptica nos brinda una gran velocidad en comparación con tecnologías predecesoras, tiene también sus deficiencias. Se utiliza la sigla en inglés PLI (*Physical Layer Impairments*) para referirse a los efectos de señales luminosas que pueden degradar la señal que se está transmitiendo por la fibra. Por más información al respecto referirse a [29] .

Los efectos de PLI pueden ser lineales o no lineales. Se tienen algunos efectos lineales como la atenuación y la dispersión cromática, mientras que otros son no lineales como el efecto Kerr. Además existen medidores de los efectos, los cuales determinan la calidad de transmisión (QoT) de la fibra óptica:

- BER (*Bit Error Rate*) : Es definido como la probabilidad de que se detecten bits defectuosos.
- Q-factor: Es la relación señal a ruido en el circuito de decisión. Se calcula generalmente utilizando la relación entre la energía de la señal y la desviación estándar del ruido. Indica que tan bien se puede distinguir la señal transmitida de las perturbaciones y el ruido.
- OSNR (*Optical Signal to Noise Ratio*): Representa la relación entre la potencia de la señal óptica y la potencia del ruido óptico en un sistema de comunicación. Cuanto mayor sea el OSNR, mejor será la calidad de la señal. Es útil para determinar cuánta señal está presente en relación con el ruido en un sistema de comunicaciones ópticas. Esta fuertemente relacionada al BER y Q- Factor, a su vez es la más utilizada dado que su cálculo no depende de un ancho de banda específico a diferencia de las demás.

Dentro de las medidas algunas también son efectos del PLI. Entre ellas se tienen:

- Atenuación: Refiere específicamente a la disminución de la intensidad de la señal mientras viaja a lo largo de la fibra óptica. Se expresa en decibelios por kilómetro (dB/km) y es una medida de cuánta potencia de la señal se pierde por unidad de distancia.
- *Loss* (pérdida): Refiere a la cantidad de potencia de la señal que se pierde o se disipa durante la transmisión debido a diversos factores, como la atenuación mencionada anteriormente, pero también incluye pérdidas adicionales en los conectores, empalmes, acoplamientos y otros componentes. Utiliza como unidad decibelios (dB).

Dentro de los efectos no lineales se encuentra el efecto de Kerr y los efectos de dispersión de tipo Raman y Brillouin.

El efecto de Kerr refiere a la variación del índice de refracción de la fibra con la potencia óptica incidente. En los sistemas multicanales la potencia óptica concentrada en el núcleo de la fibra puede afectar el índice de refracción de la fibra causando el cambio de fase de los elementos electromagnéticos. En la herramienta de simulación se calcula una simplificación del efecto de Kerr.

Por otro lado, los efectos de dispersión son causados por la interacción de la luz con la estructura de la fibra misma. Los efectos no lineales a su vez se pueden categorizar dependiendo si ocurren por la interacción dentro de un mismo canal de transmisión (intra-canal) o interacción entre diferentes canales (inter-canal).

Se realiza un mayor énfasis en la dispersión cromática en una fibra óptica , la cual refiere al fenómeno en el que la velocidad de grupo de un pulso propagándose depende de la longitud de onda. Esto significa que los diferentes componentes espectrales de un pulso viajan a diferentes velocidades dentro de la fibra.

Para entender y cuantificar este fenómeno, se utilizan términos matemáticos como la segunda derivada de la constante de propagación (β_2) o el coeficiente de dispersión de velocidad de grupo (GVD). Este último se suele denotar como D y es el que se usa más adelante para calcular una dispersión cromática simplificada.

La dispersión cromática se produce por dos factores. El primero es llamado dispersión de material y es provocado porque el material usado en las mismas (sílice) varia con la frecuencia de la luz transmitida, logrando que colores distintos se transmitan a distintas velocidades. El segundo se llama dispersión de guía de onda y se produce por cómo la fibra misma guía la luz.

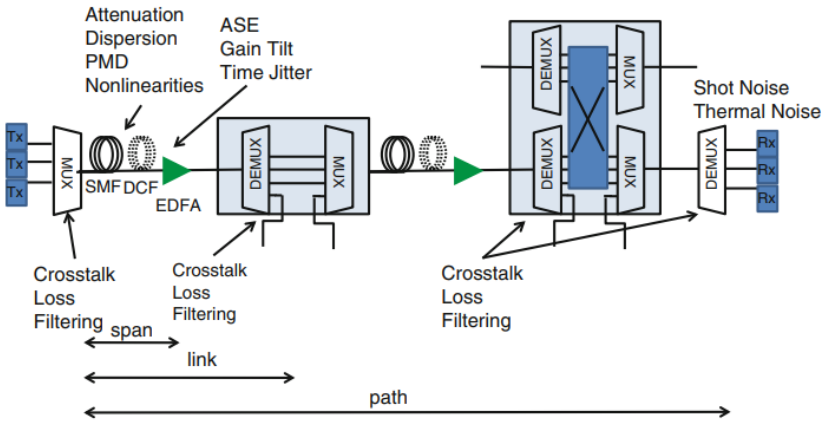

Figura 2.8: Arquitectura DWDM con manejo de dispersión cromática.

Este fenómeno es lineal, lo que significa que puede ser compensado si se aplica una dispersión inversa adecuada. Se incorporan en las redes dispositivos para el manejo de la dispersión. En la figura 2.8 se ve un ejemplo de una red DWDM donde se controla la dispersión colocando un compensador de dispersión

por cada span de fibra (segmento de fibra contiguo).

Una vez ya se ha explicado cómo funcionan las redes ópticas y móviles haciendo énfasis en comunicaciones 5G, se investigó qué plataformas de simulación existen y cuáles son las mejores para crear simulaciones de una red óptico-móvil. En la siguiente sección se hablará de las plataformas existentes revisadas, junto con sus ventajas y desventajas para el objetivo concreto de este proyecto.

2.2. Plataformas de simulación

Antes de empezar con la implementación era necesario elegir un entorno de simulación adecuado y a su vez corroborar que efectivamente no existe ninguna herramienta de simulación que comprende los dos dominios unidos, o sea redes móviles 5G que donde la comunicación entre radiobases se de a través de una red óptica.

A priori la plataforma dada para usar era OMNeT++ con simulador óptico y Simu5G para la parte móvil. Esta herramienta parecía apropiada pero de igual manera se revisaron brevemente esta y otras herramientas para confirmarlo. En cuanto a las plataformas que simulaban redes 5G fue de gran apoyo el artículo [31], que presenta la herramienta creada Py5cheSim (se comenta un poco más adelante) y a su vez menciona otras muchas herramientas existentes que ya se habían investigado. Se mencionan a continuación todas las herramientas para simulación 5G que se vieron:

- 5G-Lena[27]: es una biblioteca *open source* del simulador ns-3. Se rige bajo la licencia de GNU GPLv2. Este simulador fue estudiado en profundidad dado que otro equipo relacionado al proyecto CSIC lo presentó en uno de los seminarios. ns-3 es un simulador de red que se basa en eventos discretos y es altamente completo. Particularmente 5G-LENA fue creado basándose en lte-LENA [8] (otra biblioteca de este simulador) y presenta una gran cantidad de parámetros a configurar, lo cual permite crear diversas simulaciones. Implementa todas las capas definidas por el estándar 3GPP. Las desventajas que posee en realidad se dan por el simulador ns-3. El mismo presenta un grado de complejidad muy alto, por lo que tiene una curva de aprendizaje muy alta. Además, carece de una herramienta de visualización de la simulación, lo cual dificulta la comprensión del simulador. Por otro lado, al ser tan completo demora un tiempo demasiado alto en procesar las simulaciones, lo cual es importante porque los escenarios que se van a simular van a constar de muchos usuarios (así como indica el caso de uso de 5G). Esto último dificulta la corrida de distintas simulaciones si no se tiene una PC lo suficientemente potente y nos limita a la hora de escalar la topología.

- Vienna [24] [30]: Es un simulador a nivel de sistema programado en matlab, está disponible su descarga con una licencia académica. También se basa fuertemente en su predecesor Vienna LTE simulator. Permite realizar simulaciones tanto a nivel de sistema como a nivel de enlace. No es tan

completo como el anterior, algunas de sus carencias son la imposibilidad de mandar tráfico *uplink* sin usar el buffer lleno, no tener modulación 256-QAM, etc.

- 5G-K-Sim [20] [7]: Es una herramienta desarrollada en c++ que permite una simulación de todos los niveles (Sistema, enlace y red). La mayor desventaja que presenta es que fue desarrollado en los principios de la estandarización del 5G y por ende no tiene tanta fidelidad con los estándares actuales.

- Simu5G [26] [25]: Es una biblioteca *open source* dentro del entorno de simulación OMNeT++. Tanto OMNeT++ como Simu5G también fueron estudiados en profundidad y presentados en los seminarios. OMNeT++ está basado en c++ y es un simulador de eventos discretos. Una ventaja de Simu5G con respecto a otros simuladores es tener soporte para diferentes numerologías, FDD, TDD, *carrier agregation*, etc. Aunque tiene como desventaja que no permite todas las opciones de asignación de recursos como *network slicing* o *mini-slot*.

- Py5cheSim [31]: Se desarrolló un simulador 5G de eventos discretos en python. Es bastante completo y se mantiene simple de usar. Se enfoca más que nada en la implementación de RAN-Slicing (*Radio Access Network-Slicing*). También tiene soporte de múltiples numerologías, FDD y TDD, funcionalidades básicas de *Carrier Aggregation* y MIMO.

- SyntheticNET [34]: Este simulador está programado en python. Se basa mayoritariamente en simular el proceso de *handover* de manera lo más real posible. Sin embargo no hay mucha información disponible del mismo.

- OpenAirInterface [19] [18]: Es un desarrollo *open source* que presenta como desventaja el no ser escalable a simulaciones con redes muy extensas.

Por otro lado, en cuanto a las redes ópticas, las herramientas de simulación que se encontraron son las siguientes:

- GNPy [21]: Esta herramienta de simulación y planificación óptica fue estudiada en profundidad dado que se realizó una presentación de la misma en los seminarios. Es una herramienta sumamente completa, donde permite crear simulaciones *end-to-end* muy realistas ya que se puede configurar tanto propiedades físicas de la red como que dispositivo en concreto se usa para cada elemento y bastantes parámetros (por ejemplo uno de ellos es la asignación del espectro). Para cada simulación calcula los k caminos óptimos en términos de distancia o de OSNR a elección. Finalmente brinda como resultado todo el recorrido del camino óptico desde nodo origen hasta nodo destino con valores de potencia entrante, potencia saliente, pérdidas, etc. por cada elemento de la red. Además brinda la información completa de todo el canal, la frecuencia, la potencia, el SNR total, etc.

- Flex Net Sim [13]: Es una biblioteca de simulación óptica implementada en c++ donde se pueden diseñar algoritmos de asignación de la grilla flexible. Tiene soporte para configurar distintas topologías, rutas, bitrates y SDM (*Spatial Division Multiplexing*).
- Optical Network Simulator (ONS) [11]: Es un simulador óptico WDM de eventos discretos programado en java. Permite configurar bastantes parámetros y brinda como resultado una salida muy completa. Esta especializado en tráfico de redes EON y permite integrar nuevos algoritmos como RWA (*Routing and Wavelength Assignment*) y RMLSA (*Routing, Modulation Level, and Spectrum Allocation*) entre otros.
- Flexigrid Restoration Elastic: Es un simulador creado sobre la plataforma OMNeT++ por uno de los participantes del proyecto CSIC hace algunos años pero que no se ha publicado. Este simulador implementaba redes ópticas flexibles teniendo especial énfasis en el plano de control.

Capítulo 3

Elección de plataforma de simulación

3.1. Clasificación de las plataformas

Una vez analizadas muchas plataformas se debe decidir cuál es la mejor para la creación de la herramienta multidominio. Para ello se creó una tabla comparativa de todas las plataformas, a modo de tener un resumen de los puntos más importantes de cada una. Se puede visualizar la tabla comparativa en 3.1, a continuación se detalla a que refieren sus columnas en el orden correspondiente:

- Nombre de la plataforma de simulación.
- Qué se especializa en simular (redes 5G o redes ópticas).
- Indica si es *Open Source*.
- Lenguaje de programación base (el más usado, ya que pueden tener más de uno).
- Nivel de completitud, o sea qué tan fiel a la realidad son las simulaciones. Se clasifican en medio, bueno, muy bueno y excelente.
- Qué tan integrable es con otra plataforma de simulación para crear la parte restante. Por ejemplo, sea un simulador de redes 5G, qué tan rápido se puede completar la parte de redes ópticas para completar la herramienta multidominio (viceversa para los simuladores de redes ópticas). Se clasifican en baja, media y alta.
- Ventajas y desventajas de la plataforma de simulación en comparación a las otras plataformas.

Cabe aclarar que las distintas clasificaciones fueron totalmente subjetivas y que se podrían discutir distintos puntos de vista. La clasificación se vio fuertemente afectada por los intereses particulares de este proyecto de grado y podría ser diferente si se busca una plataforma de simulación con otros fines de uso.

Particularmente para clasificar la integridad de las plataformas de simulación se prestó mucho énfasis en los lenguajes base donde estaban programados y la facilidad para crear la otra parte del dominio. Es lógico que es mucho más sencillo integrar dos plataformas programadas en el mismo lenguaje de programación que si tienen lenguajes distintos. A modo de ejemplificar, en este sentido la plataforma Vienna presentan una desventaja por ser la única programada en Matlab.

Plataformas de simulación	Especializado en	Open Source	Lenguaje base	Nivel de completitud	Integración	Ventajas	Desventajas
5G-Lena (ns3)	Redes 5G	Si	C++	Muy bueno	Alta	Muy completo	•Nivel de complejidad alto •Mucho tiempo para procesar simulaciones
Vienna	Redes 5G	Si (licencia academica)	Matlab	Medio	Baja	Simulaciones a nivel de sistema y de enlace	Muchas carencias
5G-K-Sim	Redes 5G	Si	C++	Medio	Media	Simulaciones a todos los niveles (sistema, enlace y red)	No actualizado a estándares actuales de 5G
Simu5G (Omnet++)	Redes 5G	Si	C++	Bueno	Alta	Soporte para numerologias, FDD, TDD	No permite asignación de recursos
Py5cheSim	Redes 5G	Si	Python	Excelente	Media	El más completo de todos	Integrabilidad
SyntheticNET	Redes 5G	Si (licencia academica)	Python	Desconocido	Media	Realismo en proceso de handover	Falta de información
OpenAirInterface	Redes 5G	Si	C	Bueno	Baja	Soporte muchas especificaciones NR	No es escalable
GNPy	Redes ópticas	Si	Python	Muy bueno	Media	Muy completo	Demasiado enfásis en la capa física
Flex Net Sim	Redes ópticas	Si	C++	Bueno	Media	Soporte para muchas topologias, rutas, algoritmos de asignacion de la grilla flexible	Falta de estudio
Optical Network Simulator	Redes ópticas	Si	Java	Bueno	Baja	•Especializado en redes EON •Fácil integrabilidad con nuevos algoritmos	Falta de estudio
Flexigrid Restoration Eslastic (Omnet++)	Redes ópticas	-	C++	Bueno	Alta	Muy completo el plano del control de la red	No tuvo un testing tan exhaustivo como los demás

Figura 3.1: Tabla de comparación de las distintas plataformas de simulación.

3.2. Conclusiones de las herramientas de simulación

Luego de estudiar estas herramientas en profundidad se llega a dos conclusiones. La primera de ellas es que si bien algunas son mejores que otras en términos de completitud (tanto en redes ópticas como con las redes 5G) ninguna de ellas tiene ambas tecnologías. Por ejemplo se podría usar GNPy para todo lo que tenga que ver con redes ópticas ya que es una herramienta muy potente y completa, pero viendo el objetivo principal que es crear la herramienta de simulación óptico-móvil debería implementarse toda la parte de redes 5G desde cero. Lo ideal sería minimizar los esfuerzos de integración.

La segunda conclusión es que el mejor entorno de simulación para crear la herramienta no va a ser el más completo de uno de los dominios, sino que va a ser el que permita la creación del dominio que falta (dado que todos los simuladores se especializan únicamente en uno de los dos dominios) y de suficiente libertad para integrar los dos dominios.

De esta manera se descartan un gran número de simuladores que comprenden únicamente un dominio sin posibilidad de la desarrollo del otro, o donde la implementación del otro dominio se corresponde a una gran cantidad de tiempo invertido por ser un desarrollo desde cero. Esto se debe a que la implementación completa de uno de los dos dominios se considera fuera del alcance de este proyecto. La integración de ambos dominios puede implicar cambios pero no debería tratarse de un desarrollo desde cero. Por ejemplo, Py5cheSim parece ser uno de los mejores simuladores de redes 5G por su sencillez y completitud, pero no permite la implementación de una red óptica, o la integración con un simulador de red óptica sería demasiado compleja por tener que programar sobre ambos simuladores.

Finalmente la herramienta elegida fue OMNeT++ con Simu5G, coincidiendo con lo planteado en el capítulo anterior sobre que es una herramienta adecuada. Luego del estudio de esta plataforma y de otras se confirmó que presenta muchos factores a favor, los cuales se describen a continuación.

Primeramente es una de los pocos entornos de simulación que permite crear los elementos de la red faltantes (en este caso la red óptica) e integrarlos fácilmente con los elementos de Simu5G, gracias a que está creado por una jerarquía de módulos que se detallará más adelante en la sección 4.1.

Por otro lado, si bien Simu5G no es la herramienta más completa en cuanto a simulaciones 5G, si es suficiente para cumplir los objetivos de este proyecto como completar distintos escenarios. También presenta otras ventajas como su facilidad de comprensión en comparación a otras herramientas (por ejemplo 5G-Lena).

Además OMNeT++ presenta una herramienta de simulación gráfica para visualizar las distintas simulaciones y algunos métodos para calcular estadísticas. Este punto es realmente importante dado que para integrar dos tecnologías totalmente diferentes, un entorno gráfico ayuda a la mayor comprensión de los errores de una manera más rápida e intuitiva.

La última razón y no menos relevante es que se cuenta con simulador óptico en OMNeT++ (Flexigrid Restoration Elastic), implementado por un integrante del proyecto CSIC hace algunos años. Si bien no se utilizó este simulador directamente, si sirvió de ejemplo para una nueva implementación de otro simulador óptico y no tener que hacerlo desde cero.

A modo de resumen del capítulo se estudiaron las herramientas de simulación *open source* existentes. Se definió la implementación del entorno de simulación *end-to-end* basado en OMNeT++; en particular tomando como base Simu5G y un desarrollo de un módulo óptico que se comenzó en paralelo dentro del proyecto CSIC. En el siguiente capítulo se describe en detalle el entorno de simulación y los desarrollos realizados.

Capítulo 4

Parte Central

Para el desarrollo de la herramienta de simulación *end-to-end* de una red óptico-móvil se eligió como base la plataforma OMNeT++. La misma se encuentra disponible en [1]. En particular para la parte móvil el trabajo se basó en Simu5G (Herramienta de simulación de redes 5G) y para el segmento óptico se utilizó un desarrollo realizado en marco del proyecto de investigación CSIC.

En este capítulo primeramente se presentarán los aspectos generales de la plataforma OMNeT++. Seguidamente se presentan las bibliotecas que se usaron de OMNeT++ (INET y Simu5G) y los experimentos realizados para la comprensión de la herramienta. Se continúa explicando el simulador de redes ópticas. Se finaliza mostrando el proceso de creación de la herramienta de simulación óptico-móvil completa, enfatizando en interrogantes a resolver y las decisiones tomadas en el proceso.

4.1. Plataforma OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) es un entorno de simulación de redes y sistemas de comunicación de código abierto. Fue la plataforma elegida para el desarrollo de la herramienta de simulación óptico-móvil. Está desarrollado en C++ y se basa en el paradigma de programación orientada a objetos. Esto facilita la creación de modelos modulares y reutilizables. Permite representar diversos aspectos de las redes, como nodos, enlaces, protocolos de comunicación, colas y aplicaciones. Lo cual facilita la representación de escenarios realistas para el estudio de rendimiento y comportamiento de sistemas de comunicación. Se sugiere visitar el sitio web [1] para más información.

Esta plataforma organiza la simulación en términos de módulos y modela el comportamiento de los mismos mediante la especificación de eventos y procesos discretos. Los módulos representan entidades en el sistema que se está modelando, y se pueden organizar jerárquicamente para reflejar la estructura del sistema. Un módulo simple se programa en C++ y luego se ensamblan en componentes y modelos más grandes utilizando un lenguaje de alto nivel (NED). Todos los

módulos y agrupaciones forman la red que también es un módulo compuesto.

Los módulos se comunican entre ellos a través de mensajes, estos se envían a través de las interfaces de entrada y salida (*gates*) de cada módulo. Los distintos módulos pueden conectarse con otros módulos conectando sus *gates* directamente o mediante un canal. Las conexiones entre módulos pueden admitir distintos parámetros, por ejemplo retardo de propagación.

Además OMNeT++ ofrece un IDE basado en Eclipse, un entorno de ejecución gráfico y una serie de otras herramientas por ejemplo para el análisis de dichas simulaciones. Hay extensiones para simulación en tiempo real, emulación de red, integración de bases de datos, integración de SystemC y varias otras funciones.

Se han desarrollado incontables modelos de simulación y marcos de modelos usando OMNeT++. Muchos de estos modelos son incluso de código abierto y fueron desarrollados por investigadores de diversas áreas. Entre estos modelos se encuentran colas, modelado de recursos, protocolos de Internet, redes inalámbricas, LAN conmutadas, redes *peer-to-peer*, transmisión de medios, telefonía móvil. Redes ad-hoc, redes de malla, redes de sensores inalámbricos, redes vehiculares, NoC, redes ópticas, sistemas HPC, computación en la nube, SAN , etc.

Ofrece una biblioteca estándar llamada INET Framework [5]. La misma contiene protocolos comunes, agentes y otros modelos especialmente útiles para investigadores y estudiantes que trabajan con redes de comunicación. INET Framework se mantiene actualizado para uso de la comunidad, utilizando parches y nuevos modelos aportados por sus miembros. Varios otros marcos de simulación toman INET como base y lo extienden en direcciones específicas, como redes vehiculares (Veins, CoRE) [32], redes superpuestas/ *peer-to-peer* (OverSim) [9] , LTE (SimuLTE) [33] o redes 5g (Simu5G) [25]. Se hablará más en detalle de INET y Simu5G en la subsección 4.1.2.

4.1.1. Simulaciones y sus elementos

Existen cuatro tipos de archivos principales que son muy importantes de comprender para crear y poder ejecutar simulaciones con éxito en OMNeT++:

- Clases c++ (.h y .cc): En las mismas se definen los atributos y el comportamiento de un módulo simple (que no está compuesto por otros submódulos). A su vez en OMNeT++ existen distintas clases base que contienen funciones genéricas para todos los módulos, por ejemplo para enviar y recibir mensajes. Más adelante se brinda más información.

- Mensajes (.msg): Los mensajes se definen como clases con los atributos que se desea y OMNeT++ al compilar genera automáticamente un .h y .cc por cada mensaje para que puedan ser usados por los módulos. Los mensajes creados en general son clases derivadas de *cMessage* o *cPacket* (clases básicas de OMNeT++ que tienen el comportamiento de un mensaje que se transmite por la red).

- *Network Description* (.ned): Definen módulos simples o compuestos de otros submódulos (incluyendo la misma red). En este archivo se definen los parámetros del módulo (donde también se pueden poner valores por defecto), las *gates* de entrada y/o salida y conexiones entre los distintos submódulos que lo compone (si tiene).

- Configuración (.ini): Es un archivo de configuración donde se escriben datos de interés para la simulación de una topología. Entre estos datos se encuentran los parámetros de los módulos que la componen, tiempo de simulación, configuraciones visuales de la simulación, etc. Este archivo es absolutamente necesario para realizar simulaciones.

Clases básicas de OMNeT++

Como se menciona anteriormente OMNeT++ ofrece algunas clases básicas que tienen en su implementación los comportamientos mínimos de un objeto en la red para poder simular una topología. Todas las clases derivan de una clase madre llamada *cObject* y dependiendo del tipo de objeto (si es un mensaje, un canal, un nodo) tienen distintas ramificaciones. Algunas de ellas se pueden apreciar en la figura 4.1.

Particularmente se hablará de las dos clases más utilizadas que fueron *cSimpleModule* y *cMessage* (usadas para modelar módulos simples y mensajes respectivamente). Sin embargo cabe mencionar que existen muchas más que son realmente útiles. Entre algunas de ellas se encuentran *cGate* para modular interfaces de entrada o de salida, *cChannel* para modelar canales, etc.

La clase *cSimpleModule* es indespensable para crear nuevos módulos simples en la red. Esta clase cuenta con algunos procedimientos principales heredados que se mencionarán a continuación. Los primeros tres procedimientos de la lista se pueden redefinir, lo cual es sumamente útil para determinar el comportamiento específico de ese módulo.

- *initialize()* : Esta función se activa en la creación del módulo y justo antes de que comience la simulación. Si la inicialización es compleja y requiere varios pasos, estos se pueden definir individualmente. Cada paso se denomina *"stage"* y se pasa como parámetro a la función *initialize()*. OMNeT++ ejecutará primero *initialize()*, luego *initialize(1)*, *initialize(2)*, y así sucesivamente, hasta completar todos los pasos definidos. La cantidad total de pasos se determina mediante la función *numInitStages()*.

- *handleMessage()*: Recibe como parámetro un mensaje de tipo *cMessage* y en este procedimiento se define como se va a comportar el módulo al recibir mensajes.

- *finish()*: Esta función es llamada cuando la simulación termina exitosamente. Usualmente se utiliza para guardar las estadísticas recolectadas durante la simulación.

- *send()*: Este método recibe como parámetro el mensaje a ser enviado y el *gate* mediante el cual se va enviar el mensaje. Existen otras variaciones como por ejemplo *sendDelayed()* que incorpora un retardo en el envío o *sendDirect()* que manda directamente a otro módulo y se debe especificar el nombre del módulo destino y su *gate* de entrada.

- *scheduleAt()*: Este método recibe como parámetros un tiempo de simulación y un mensaje. Lo que hace es enviar ese mensaje a sí mismo cuando pase ese determinado tiempo de simulación a modo de "recordatorio". Es útil para disparar eventos que ocurren cada cierto tiempo.

La clase *cMessage* es de mucha utilidad para crear mensajes personalizados. Como se puede ver en la figura 4.1, esta clase deriva de la clase *cEvent* (por lo que es también un evento) y a su vez es clase madre de la clase *cPacket*.

Además de los atributos que se describan en el archivo correspondiente, se puede definir o acceder a atributos de interés ya incluidos dentro de la clase, como por ejemplo el tipo del mensaje (*kind*), prioridad y *timestamp*. También se puede duplicar el mensaje con la función *dup()* y se puede corroborar si es un mensaje enviado de un módulo para sí mismo con la función *isSelfMessage()*.

A su vez durante la simulación puede resultar de utilidad acceder a la información de control que guarda el mensaje. Se menciona a continuación algunos datos de control de ejemplo:

- Tiempo de creación del mensaje.
- Tiempo en que se envió el mensaje.
- Tiempo en que llegó el mensaje.
- Módulo que acaba de enviar el mensaje.
- *Gate* de salida del módulo que envió el mensaje.
- Módulo que acaba de recibir el mensaje.
- *Gate* de entrada del módulo que recibió el mensaje.

Para una mayor comprensión de todo lo descrito, a continuación se muestra una simulación del proyecto de ejemplo *tictoc*, brindado por OMNeT++.

Simulación de ejemplo

Se tiene una clase bastante simple llamada *Txc1*, la misma extienda la clase *cSimpleModule* y no tiene ningún atributo que no sea heredado. En su implementación redefine los procedimientos de *initialize()* y *handleMessage()*. Cuando este módulo se inicializa chequea su nombre, el cual si corresponde a "tic" genera un nuevo mensaje y lo manda por su *gate "out"*. Por otro lado su comportamiento al recibir un mensaje es enviarlo nuevamente por por su *gate* de salida *"out"*, sin importar las características del mensaje.

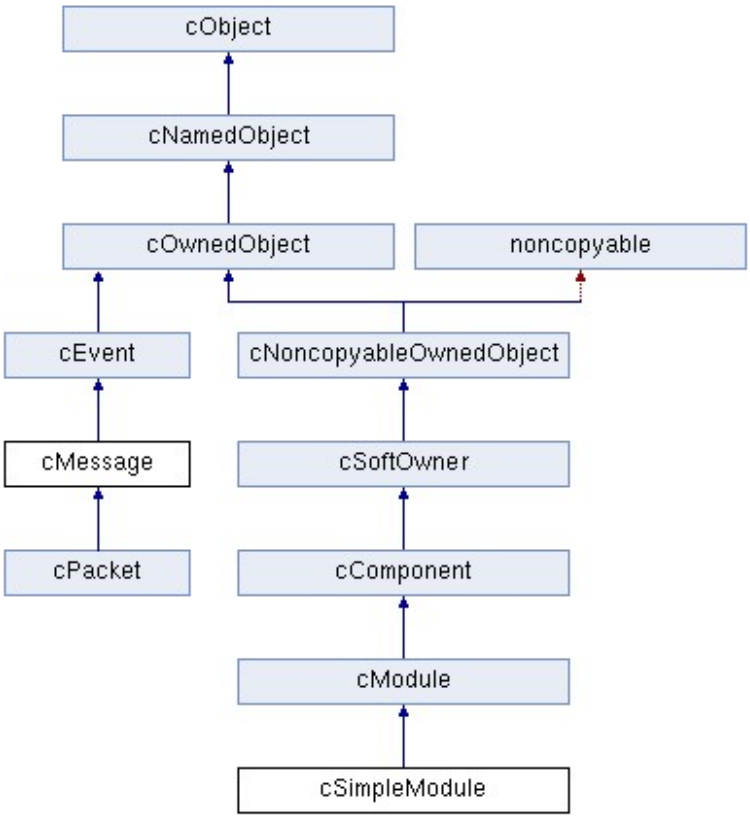

Figura 4.1: Jerarquía de clases *cMessage* y *cSimpleModule* partiendo de *cObject*, imagen creada a partir de las fuentes [3] y [4].

En el archivo .NED se tienen definidas las *gates* del módulo *Txc1* así como la topología que se va a simular. El módulo *Txc1* tiene 2 *gates*, una de salida (*"out"*) y otra de entrada (*"in"*). La topología a simular se llama "tictoc1" y consta de dos submódulos, ambos son de la clase *Txc1* y la diferencia entre ambos es que uno se llama "tic" y el otro "toc". A su vez estos submódulos están conectados mediante un canal que tiene retardo de propagación de 100ms la *gate* de salida de "tic" con la *gate* de entrada de "toc" y análogamente en el otro sentido. En la figura 4.2 se pueden ver las conexiones correspondientes.

En el archivo de configuración .ini se tienen varias configuraciones distintas identificadas por su nombre, en este caso se va a correr la configuración llamada Tictoc1. Esta configuración tiene únicamente la red que va a simular, tictoc1. Sin embargo otras configuraciones más complejas tiene más datos, por ejemplo

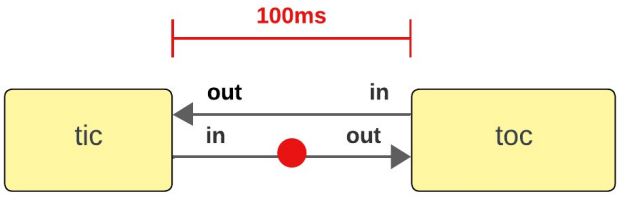

Figura 4.2: Conexiones de los módulos tic y toc.

el tiempo de simulación o parámetros de los módulos que tiene la red a simular.

Corriendo la configuración deseada del archivo .ini se inicia el entorno de ejecución gráfico que se ve como en la figura 4.3, presionando en la figura de *play* inicia la simulación. En la misma se puede observar la red (compuesta por sus dos módulos), los mensajes viajando a través de las conexiones y en la parte de abajo se ven los eventos que ocurrieron en su respectivo tiempo desde que inició la simulación. En el capítulo 5 se verá más a detalle cómo correr una simulación.

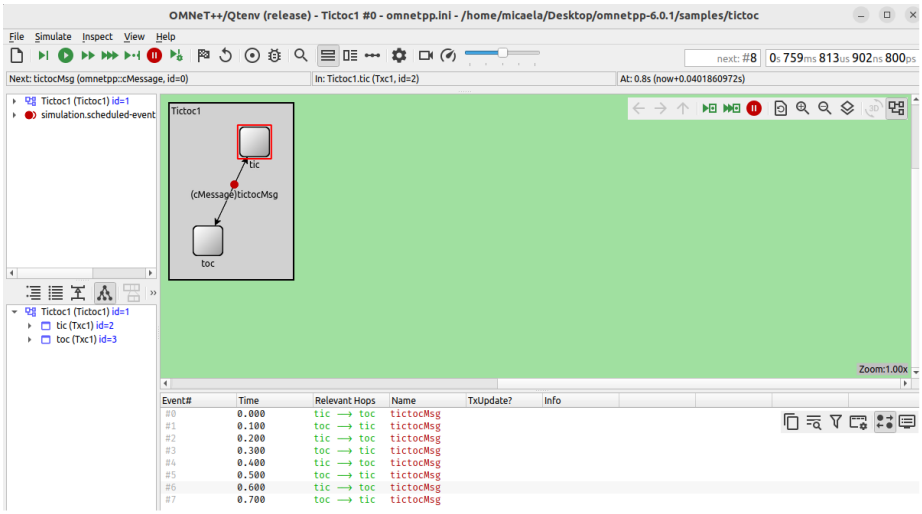

Figura 4.3: Simulación de red tic toc.

En la próxima subsección se profundizará más en INET y en Simu5G, que fueron herramientas fuertemente utilizadas para el proyecto.

4.1.2. INET y Simu5G

Como se mencionó anteriormente INET es una biblioteca de modelos de código abierto para el entorno de simulación OMNeT++. Fue creado principalmente con el objetivo de usarse para experimentación y conserva la filosofía de OMNeT++ sobre la jerarquía de módulos para crear módulos más complejos. Permite una simple integración con nuevos componentes creado por el usuario ya que existe una gran documentación sobre los componentes ya existentes, lo cual hace fácil su comprensión y modificación en caso de ser necesario.

Entre algunos componentes implementados que contiene se tienen:

- Protocolos de Internet (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.).
- Protocolos de capa de enlace cableados e inalámbricos (Ethernet, PPP, IEEE 802.11, etc.
- Protocolos MANET.
- DiffServ (Diferenciación de Servicios).
- MPLS con señalización LDP y RSVP-TE.

Además de varios modelos de aplicaciones y muchos otros protocolos y componentes.

Simu5G es la evolución de SimuLTE, una poderosa herramienta de simulación de redes LTE y LTE Advanced (creada a su vez usando la base de INET), pero incorpora nuevo acceso a radio 5G. Permite simular escenarios de red donde coexisten 4G y 5G, tanto en implementaciones *StandAlone* (SA) como *E-UTRA/NR Dual Connectivity* (ENDC).

Con estas dos herramientas fue simulada la parte móvil. A continuación se brinda una breve explicación de los módulos o técnicas más relevantes de INET y de Simu5G que fueron utilizados para esta parte.

- *NRUe* (ue): dispositivos usuarios de la red 5G que desean comunicarse con otros usuarios.
- *gNodeB* (radiobase gnb): Estación base en una red 5G. Es responsable de la comunicación inalámbrica con los dispositivos de usuario proporcionando conexiones inalámbricas de alta velocidad.
- *Upf* (User plane function): Es un componente fundamental de las redes 5G. Proporciona el punto de interconexión entre la infraestructura móvil y la Red de Datos.
- *Binder*: Este módulo es usado para guardar información de la red que necesita ser accedida, por ejemplo funcionalidades de control. Es necesario uno en cada topología y únicamente uno.

- *CarrierAgregation*: Es una técnica que permite a un dispositivo móvil utilizar múltiples bandas de frecuencia simultáneamente para transmitir datos. Esto se hace para aumentar la capacidad total de la red, mejorar la velocidad de datos y optimizar la eficiencia del espectro.

- *LTEChannelControl*: Maneja el control de canales y se ocupa del acceso al medio. Es necesario uno en toda topología que tenga dispositivos móviles.

- *Ipv4NetworkConfigurator*: Este módulo asigna direcciones IPv4 y configura un ruteo estático en toda la red. Es necesario uno en cada topología.

Estos son los elementos básicos para crear una red móvil en OMNeT++. En la siguiente subsección se hablará brevemente de las distintas redes móviles que fueron creadas, las cuales utilizan estos módulos.

4.1.3. Red móvil

En una primera instancia se desarrollaron tres topologías distintas para lograr una mayor comprensión de la tecnología y evaluar todas las variantes que nos permitía configurar la plataforma. Las topologías creadas fueron basadas en las topologías *MultiCell_Standalone* y *SingleCell_Standalone* de Simu5G.

Por otro lado, las tres topologías tenían algunos módulos en común, necesarios para que funcione la red, y a su vez alguna diferencia en la cantidad de ues y gnbs. A continuación se da una lista de los elementos comunes pertenecientes a las topologías con sus respectivos nombres característicos:

- 1 binder (módulo *Binder*): Como se menciona anteriormente, es el encargado de llevar el control de todos los módulos de la red y necesario para que todo funcione.

- 1 carrierAgregation (módulo *CarrierAgregation*): Con el cual se puede configurar cosas referidas a la numerología y espectro.

- 1 channelControl (módulo *LTEChannelControl*) : Con el que definimos el método de propagación por el medio.

- 1 configurator (módulo *Ipv4NetworkConfigurator*): El encargado de asignar las direcciones IPv4 a los demás módulos y configurar sus tablas de ruteo.

- 1 routingRecorder (módulo *RoutingTableRecorder*): Se encarga de guardar los cambios en las tablas de ruteo de todos los hosts y *routers*.

- 1 upf (módulo *Upf*): Conectado al iUpf mediante su *gate pppg* y conectado al *router* mediante su *gate filterGate*, es el encargado de enrutar los mensajes que manda la gnb y mandarlos (si corresponde) fuera del mundo de la radio hacia el *router*.

- 1 iUpf (módulo *Upf*): conectado a la(s) gnb(s) y a al upf mediante sus *gates pppg*, es quien tiene contacto directo con la gnb y lo conecta con el otro upf. La diferencia entre un iUpf y un upf estándar es que el iUpf integra la función de plano de usuario con otras funciones de red para una mayor eficiencia y optimización mientras que el upf es independiente.
- 1 *server* (módulo *StandarHost*): Terminal conecta al *router*, es quien recibe o manda mensajes hacia los ues.
- 1 *router* (módulo *Router*): Conectado al upf y al *server*, es el encargado de enrutar los paquetes que le lleguen hacia la dirección correcta.
- 1 ue[] (módulo vector *NRUe*): vector de ues, donde la cantidad específica depende de la topología y las pruebas realizadas.
- 1 o más gnb (módulo *gNodeB*): la cantidad de gnbs depende de la topología y las pruebas realizadas.

En la figura 4.4 se muestran las *gates* de cada módulo que está conectado a otros para entender que no todas sus *gates* sirven para lo mismo y es importante conectarlos de manera correcta.

El *router* y *server* tienen las mismas *gates* pero la única usada en este caso es *pppg* (la cual es un vector por lo que puede tener muchas conexiones).

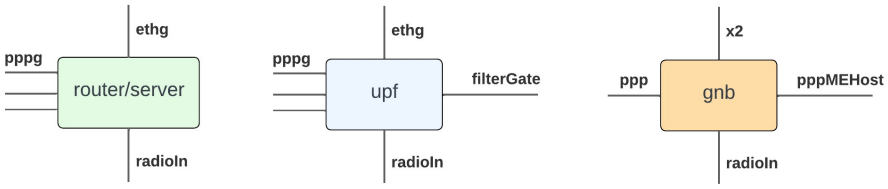

Figura 4.4: *Gates* de los módulos conectados que pertenecen a una red móvil.

Figura 4.5: Conexión entre módulos de la red móvil.

Los módulos upf y iUpf se conectan internamente y con la gnb a través de su *gate pppg*. Sin embargo con el *router* se conectan a través de su *gate filterGate*, la cual recibe todo el tráfico con destino fuera del mundo móvil. Por último las *gates* más usadas de la gnb son *x2* y *ppp*. Siendo *ppp* la *gate* que usa para conectarse a los upf y *x2* la gate que usa para conectarse con otras gnbs en

caso de requerirlo. En la figura 4.5 se aprecia cuáles son las conexiones correctas entre los distintos módulos.

A continuación se presentan cada una de las topologías y en que se diferencian entre ellas:

Topología 1:

Se caracteriza por los siguientes elementos:

- 1 gnb.
- N ues.

Como se puede apreciar en la figura 4.6 la cantidad de ues en este caso era 4, pero al ser un módulo vector este número es parametrizable y se hicieron distintas simulaciones aumentando y disminuyendo la cantidad.

En este caso se probó un único perfil de tráfico, lo que significa que todos los ues mandaban o recibían mensajes de un tipo en específico (por ejemplo VoIP, CBR, etc). Más adelante se nombran los tipos de tráfico probados e información de cada uno de ellos.

Topología 2:

Se caracteriza por los siguientes elementos:

- 1 gnb.
- N ues.
- M grupos de usuario (con distintos perfiles de tráfico).

Esta topología se ve idéntica a la topología 1 4.6, dado que la única diferencia es que se implementan distintos perfiles de tráfico. Sin embargo esto no se visualiza en la imagen, sino en las simulaciones que se realizan. Al tener distintos grupos de tráfico esto significa que los ues no tienen por qué comportarse de la misma manera. Algunos ues podrían mandar tráfico VoIP, mientras que otros reciben tráfico CBR o otras combinaciones posibles. El principal desafío de esta topología fue crear grupos de ues que se comporten de distinta manera en el mismo período de tiempo y observar cómo se reflejaba en las simulaciones. Al igual que en la topología anterior el número de ues es parametrizable.

Topología 3:

Se caracteriza por los siguientes elementos:

- K gnb.
- N ues.
- M grupos de usuario (con distintos perfiles de tráfico).

Figura 4.6: Topologías 1 y 2 - Comunicación de n ues a *server* mediante una gnb.

Esta topología es la más compleja ya que consta de más de una gnb (en este caso 2 como se puede apreciar en la figura 4.7).

Particularmente los ues se asocian a la gnb que tengan más cercana. En la etapa de inicialización se descubren las posibles gnbs y en base a la señal que reciben de cada una elige a cuál asociarse por cercanía. Sin embargo podría configurarse de otra manera por ejemplo X cantidad asociados a gnb1 y los restantes a la gnb2.

Las gnbs se conectaban entre ellas mediante su *gate x2* y usando la aplicación *x2App*. Sin esta conexión entre ellas todos los ues deberían estar asociados a la misma gnb, esto se debe a que se tiene un único vector de ues en este caso y las gnbs se comunican entre sí para mandar tráfico desde un objeto del vector hacia otro que este asociado a la otra gnb. Por ejemplo, si se tiene ue[0] asociado a

la gnb2, ue[1] asociado a la gnb1 y ue[0] quiere mandar un mensaje a ue[1]; la gnb2 cuando reciba dicho mensaje va a mandarlo a través de su *gate x2App* ya que identifica el vector ue[...] pero no está asociada a ue[1].

Si se quisiera desconectar las gnbs, lo que se debería hacer es crear dos grupos de ues, que podrían ser ue1[...] y ue2[...]. Donde todos los ues de cada grupo se asocien a la misma gnb, por ejemplo ue1[...] con gnb1 y ue2[...] con la gnb2.

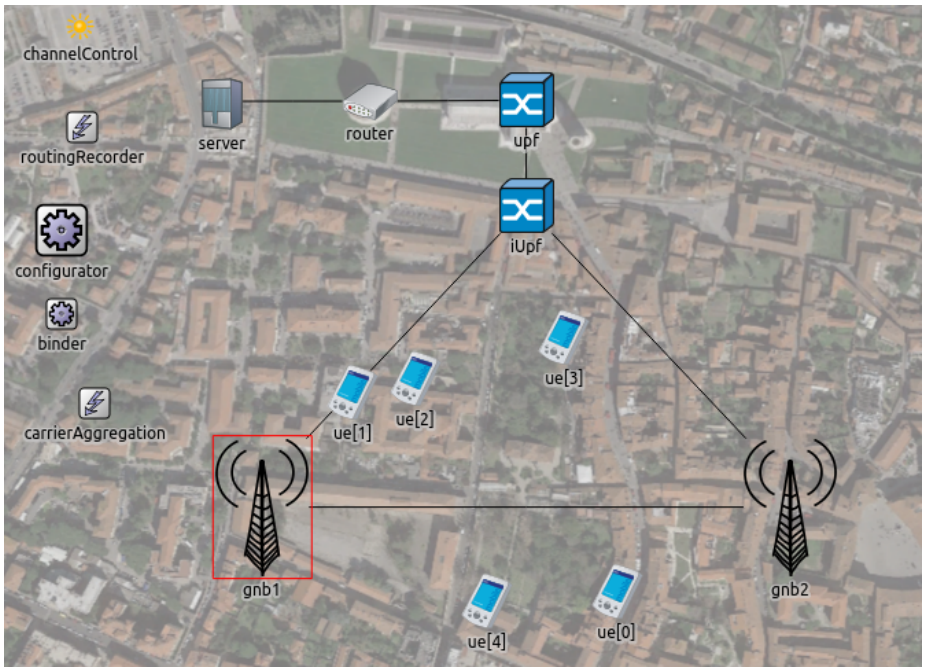

Figura 4.7: Topología 3 - Comunicación de n ues a *server* mediante dos gnbs.

A continuación se explica brevemente qué cosas se podían configurar en las distintas topologías y cuáles se usaron para simular.

Para cada topología se probaron distintos perfiles de tráfico como se menciona anteriormente, algunos de ellos fueron:

- VoIP (*Voice over Internet Protocol*): Como su nombre lo indica son mensajes de voz. En la práctica funciona convirtiendo la voz en datos digitales mediante un codec (codificador-decodificador), los cuales se envían como paquetes de datos a través de la red. En el extremo receptor, estos datos se decodifican de nuevo en señales de audio para ser reproducidas. Los momentos de habla y de silencio se manejan mediante técnicas de detección de actividad de voz y silencio. Para simular este tipo de tráfico, en Simu5G los tiempos de habla y de silencios se definen utilizando distribuciones de Weibull. En la sección 5.2.2 se detalla más de esto.

- CBR (*Constant Bit Rate*): Refiere a la transmisión de paquetes de X cantidad de bits mandados cada cierto tiempo definido. En la práctica se utiliza para aplicaciones donde se requiere una velocidad de transmisión constante, por ejemplo para transmisión de video en tiempo real.

- Burst: Refiere a enviar una gran cantidad de paquetes en ráfagas de tiempo cortas. En la práctica sucede como un fenómeno en redes de comunicación donde se produce un aumento repentino y temporal en el volumen de datos transmitidos.

Otro parámetro que se podía variar fue el tamaño de los paquetes a ser enviados y el tiempo entre que se envía un paquete y otro (esto último también dependía del tipo de tráfico que se mandaba).

Además se probó tráfico *uplink* (desde ue hacia gnb), *downlink* (desde gnb hacia ue) y ambos a la vez dentro de la misma simulación. A su vez el tráfico podía ser de ues hacia ues, de ues hacia *server* o de *server* hacia ues dependiendo del caso.

Con respecto a la configuración de la gnb se observó que se podían variar los algoritmos de *scheduling* tanto para *uplink* como *downlink*. Algunos de los algoritmos que se probaron fueron los siguientes:

- *Round Robin* (RR): Se asigna a cada dispositivo una ranura de tiempo para transmitir datos y una vez finalice ese tiempo, transmite el siguiente dispositivo que se encuentre en la cola de espera. Es un algoritmo equitativo pero muy simple y no necesariamente el óptimo dependiendo de las características de la red.

- *Proportional fairness* (PF): Busca lograr un compromiso entre maximizar el *throughput* de la red y ser justo con todos los usuarios. El n-esimo usuario recibe una métrica m_n que se calcula siguiendo la siguiente fórmula:

  $$m_n = \frac{I_n}{T_n}$$

  Donde I_n es el *throughput* instantáneo y T_n es el *throughput* medio del n-esimo usuario. De esta manera se asigna recursos al usuario con mayor métrica.

  El sentido de la justicia esta gracias a que utiliza la media de transmisiones pasadas, por lo que si un usuario no transmitió por un largo tiempo aumenta su prioridad.

- *Max C/I*: Asigna recursos priorizando usuarios por mayor C/I, o sea relación entre la capacidad de transmisión e interferencia que experimenta. Este algoritmo optimiza los recursos pero no es muy justo porque puede dejar afuera a usuarios en zonas más lejanas a la radiobase.

No se ha indagado demasiado pero también se observó que se pueden configurar distintos métodos de propagación por aire:

- *Free Space Model*: Sirve para predecir la atenuación de una señal electromagnética mientras se propaga a través del espacio libre. Se utiliza especialmente en situaciones donde no hay obstrucciones significativas entre el transmisor y el receptor, como en comunicaciones de radio o enlaces de microondas de largo alcance. Este es el método de propagación que se utiliza por defecto en las simulaciones realizadas.

- *Two Ray Ground Model*: Sirve para modelar la propagación de ondas electromagnéticas, especialmente en entornos donde hay una superficie terrestre que puede reflejar la señal.

También se observaron qué estadísticas se podían sacar y cuales resultan más interesantes de monitorear. En ese sentido se eligieron estadísticas suficientemente generales (que aplicaban tanto al dominio de las redes móviles como al dominio de las redes ópticas) para poder observar tanto ambos dominios en conjunto como separados. Las estadísticas elegidas fueron el *delay end-to-end* y *throughput end-to-end*, dándole más relevancia al primero.

El *delay end-to-end* (o latencia) es el tiempo que tarda un paquete en viajar desde su origen hasta su destino. Un *delay* bajo es crucial para el rendimiento de aplicaciones en tiempo real. Ayuda a garantizar que la red sea lo suficientemente rápida para las aplicaciones críticas y para mejorar la experiencia del usuario.

A su vez, el *throughput* refiere a la cantidad efectiva de datos que se pueden transmitir a través de una red en un período de tiempo determinado. Es una medida del desempeño real de la red, influenciada por el *delay*, la pérdida de paquetes y otros factores de la red. A diferencia del ancho de banda, que es una medida teórica del máximo de datos que puede manejar una red, el *throughput* indica cuánto está siendo efectivamente procesado y entregado. Esto lo convierte en un indicador más práctico y realista del rendimiento de la red.

En conclusión, al considerar tanto el *delay end-to-end* como el *throughput*, se obtiene una comprensión más completa del rendimiento de la red, permitiendo una mejor evaluación de las topologías multidominio. Por otro lado se tomarán estadísticas de cuántos paquetes se envían versus cuántos paquetes llegan al destino para obtener un mejor análisis de la red, en caso de ser necesario. En el capítulo 5 se realizan las simulaciones con la herramienta terminada y se explica en mayor detalle sobre las configuraciones y estadísticas que se obtienen. Además en los trabajos futuros 7.2.4 se nombran otras estadísticas más específicas de cada dominio y/o tipo de tráfico para realizar un análisis más exhaustivo.

Tras haberse mostrado las topologías que comprenden el dominio móvil, en la siguiente subsección se hablará de la red óptica que fue utilizada. Es importante aclarar que la red óptica fue desarrollada en paralelo a este proyecto por un grupo reducido de integrantes del proyecto de investigación CSIC, por esta razón no está muy consolidada ni testeada en comparación a la red móvil, que tiene años desde su implementación y muchos contribuyentes.

4.1.4. Red Óptica

A continuación se detalla la implementación de cada módulo, mensaje y comportamiento de la red óptica desarrollada. Así mismo una explicación de cómo utilizarlos y sus parámetros correspondientes.

Mensajes implementados

- **InitialMessage:** Es un mensaje creado con el objetivo de insertar tráfico en la red, por el módulo llamado *TrafficGenerator*. En el caso de querer simular una red únicamente con elementos ópticos, es necesario tener mensajes que den comienzo al intercambio de información. Cuenta con los siguientes parámetros:
  - *source*: El identificador del nodo donde se origina el mensaje.
  - *target*: El identificador del nodo hacia donde se pretende enviar el mensaje.
  - *throughput*: La capacidad requerida por el transmisor.
  - *size*: El tamaño del mensaje enviado en Mb.
- **MessageGenerated:** Es un mensaje creado con el objetivo de agregar algunos parámetros que son recibidos en el transmisor óptico.

  Este mensaje cuenta con los mismos parámetros que los de *InitialMessage* (obtenidos de este) más los siguientes parámetros nuevos
  - *wavelength*: La longitud de onda de la señal donde está "viajando" el mensaje.
  - *bandwidth*: El ancho de banda de la señal transmitida.
  - *power*: La variación de la potencia de envió del mensaje.
  - *bitrate*: Tasa de transmisión de bits.
- **MessageFiber:** Al igual que con el mensaje anterior, se necesita tener en cuenta otros parámetros para tener información de otros elementos de la red. En este caso particular, *MessageFiber* es creado para tener en cuenta el paso de los mensajes a través de una fibra óptica y los efectos que puede tener esto sobre la calidad de la información.

  Este mensaje cuenta con los mismos parámetros que los de *MessageGenerated* (obtenidos de este) más los siguientes parámetros nuevos:
  - *dispersion*: Dispersión cromática simplificada.
  - *kerrEffect*: Efecto de Kerr no lineal simplificado.

  Con estos parámetros (y los de *MessageGenerated* con los que también cuenta *MessageFiber*), el mensaje ya puede viajar a través de la totalidad de la red. Algunos parámetros se van actualizando a medida que pasan a través de varios bloques (como la potencia que lógicamente se verá afectada) y de esa forma posible tener los datos que en principio son necesarios en la red óptica.

Clases implementadas

Se presentan en esta sección algunas clases implementadas. Estos son algunos mecanismos requeridos por módulos y para lograr que cierta información se mantenga de forma externa a ellos pero accesible, se implementaron clases.

- **SwitchMatrix** Esta clase fue implementada para mantener la información de la tabla de ruteo de los *OpticalSwitches* y de los ROADMS. Por esta razón esta clase permite cargar la tabla de rutas de los *switches*, agregar y eliminar entradas a las mismas y devolver el puerto de salida para un *switch* dados el puerto de entrada y la longitud de onda de la señal. La tabla consiste de un puerto de salida para cada pareja [puertoEntrada, *Wavelength*].

  De esta forma, a la hora de agregar una entrada a la tabla, esta clase hace un chequeo de consistencia. Esto implica verificar que si ya está utilizada la longitud de onda en el puerto de salida que se desea agregar, o si ya existe una asignación de salida para la pareja [puertoEntrada, *Wavelength*] con la que se desea mapear. En caso de que no se cumplan esas condiciones, el mapeo entre [puertoEntrada, *Wavelength*] y el puerto de salida puede hacerse de forma correcta.

  De manera similar, para eliminar un mapeo, esta clase chequea que dicho mapeo exista.

  En *SwitchMatrix* también se mantiene la tabla de ruteo de los ROADMs. Para esto, como los ROADMs tienen un funcionamiento similar a los OpticalSwitches (excepto porque pueden recibir y enviar a componentes eléctricos), se hace provecho del chequeo de consistencia para los . En este caso las tablas de rutas son más simples, según el destino del mensaje se obtiene un puerto de salida del ROADM. Esto se puede implementar porque también se hace uso de la tabla de rutas del *OpticalSwitch*.

  Al inicializar la tabla de un ROADM, la tabla de *OpticalSwitch* que también tiene dentro debe ser consistente y cuando se desea agregar un mapeo de destino con un puerto de salida, también se debe enviar la longitud de onda del mensaje. De esta forma, antes de agregar el mapeo en la tabla de rutas del ROADM, se intenta de agregar en la tabla de rutas del *OpticalSwitch*. Si esto último es posible se agregaen la tabla del ROADM (y si no es posible no se hace el mapeo).

  De igual manera, para eliminar un mapeo de la tabla de rutas del ROADM, se elimina antes de la tabla del *switch* óptico. Para obtener el puerto de salida basta con saber el destino del mensaje.

- **SwitchMatrixFlexi** Esta clase es muy similar a *SwitchMatrix* pero pensada para funcionar en el caso de tener una red *FlexiGrid*. Esto implica que las funciones que tiene son las mismas pero en todas se agrega un parámetro de entrada que es el ancho de banda de la señal.

  Como en una red *FlexiGrid* los anchos de banda de las señales no son fijos, a la hora de agregar un mapeo, no alcanza con chequear lo que se chequeaba en *SwitchMatrix* sino que también se debe chequear si la señal

entra en el pedazo de espectro que se desea colocar. Para esto, según la longitud de onda (que equivale a una frecuencia central si se pasa a Hz) y el ancho de banda se puede saber si colisiona o no con la señal que quedaría a su izquierda, con la que quedaría a su derecha, con ambas o con ninguna. En caso de que pasen los controles de *SwitchMatrix* y no colisione con la señal de la derecha ni con la de la izquierda, el mapeo puede hacerse.

Para eliminar un mapeo, debe verificarse que el ancho de banda también coincida (además de que exista el mapeo). De la misma forma, para obtener el puerto de salida se necesita el puerto de entrada, la longitud de onda y el ancho de banda de la señal.

Módulos implementados

En esta subsección se presentan todos los bloques de forma individual con sus respectivos detalles.

- **TrafficGenerator:** Este bloque genera tráfico para permitir una simulación completamente óptica. Como los elementos ópticos en general envían luz, si no se tienen componentes eléctricos, debe haber algún elemento con la capacidad de generar tráfico y permitir que viajen mensajes a través de la red. Él mismo irá seguido de un *OpticalTransmitter* que será quien envíe la información generada por este bloque. Tiene un único puerto de salida ya que no recibe información (es la fuente de información).

  Cuenta con los parámetros:

  - *messageInterval*: Cada cuanto tiempo en segundos se envían mensajes.
  - *throughput*: La capacidad de envío del generador en Mbps.
  - *size*: El tamaño de la información que envía el generador en Mb.

  Cuando se inicia una simulación que tiene únicamente bloques ópticos, este bloque es el que inicia el intercambio de mensajes. Los parámetros que tiene el bloque los debe recibir como entrada en el archivo de inicialización. Con estos y otros parámetros que obtiene desde datos de la red, crea un *InitialMessage* que viajará al bloque transmisor.

- **OpticalTransmitter:** Este bloque recibe datos desde otro bloque anterior (que genere tráfico o que lo reciba) y se encarga de moverlo hacia otros módulos (usualmente a través de un *OpicalFiber*) que permitirán que los datos continúen su camino por la red óptica.

  Cuenta con puertos de entrada (*"in"*) y puertos de salida (*"out"*). La cantidad de puertos no está limitada en la definición del bloque, por lo que a la hora de definir una nueva topología, el usuario puede agregar según los requerimientos que tenga.

  Los parámetros con los que cuenta este bloque son los siguientes:

  - *wavelength*: La longitud de onda en nm de la señal (luz) que enviará.
  - *bandwidth*: El ancho de banda en GHz que tendrá la señal enviada. Este parámetro está pensado principalmente para que se utilice en el

caso de una red Flexi, razón por la cual cuanta con un valor por defecto de 50GHz.

- *minPower*: La mínima potencia de transmisión en dBm.
- *maxPower*: La máxima potencia de transmisión en dBm.
- *launchPowerVariation*: La variación de la potencia de envío en dB.
- *bit_rate*: La tasa de envío de bits utilizada por el transmisor en Gbps.

Este bloque se conecta en general con un generador de tráfico. Esto quiere decir que recibirá mensajes de *InitialMessage*. Según los parámetros que reciba en el archivo de inicialización en este bloque se creará un *Message-Generated* que se enviará a través de una fibra al bloque que esté en el otro extremo de esta.

- **OpticalFiber:** Este bloque simula una fibra óptica. En *Optical Network*se optó por utilizar un módulo en lugar de un canal para simular una fibra para simplificar algunas operaciones y la obtención de algunos parámetros. *OpticalFiber* es utilizada para conectar algunos dispositivos ópticos como *switches* , transmisores y receptores.

  Los parámetros con los que cuenta este bloque son los siguientes:

  - *distance*: Distancia de la fibra en km.
  - *alpha*: Coeficiente de atenuación en db/km (en realidad, los parámetros que permite OMNeT++ son limitados y se usa dB).
  - *D*: Parámetro de dispersión cromática en ps/nm/km (por la misma razón que antes se usa ps).
  - *gamma*: Coeficiente de Kerr no lineal en W^{-1}/km (por la misma razón que antes se usa km).

  OpticalFiber recibe mensajes desde un *transmitter* o desde otro elemento óptico. Si recibe mensaje desde un *transmitter*, a partir de los parámetros que reciba en el archivo de inicialización y de los parámetros de *Message-Generated* crea un *MessageFiber* que se encaminará por la red. Si recibe un parámetro desde otro elemento óptico suele recibir un *MessageFiber*. En ambos casos actualiza los parámetros necesarios y lo encamina como el mismo tipo por la red.

- **OpticalSwitch** Este bloque simula *switch* óptico. El mismo cuenta con los siguientes parámetros:

  - *numInputs*: Indica la cantidad de puertos de entrada con los que cuenta el switch.
  - *wavelengthToOutputGate*: Tabla de rutas del switch. Se recibe un string de la forma *InputGate wavelength OutputGate; InputGate wavelength OutputGate; InputGate wavelength OutputGate; ....*
    Por ejemplo, "0 1550 0; 0 1350 4;1 1550 3" indica que si ingresa un haz de luz por el puerto 0 con longitud de onda 1550nm, tiene puerto de salida 0; si ingresa un haz de luz por el puerto 0 con longitud de onda 1350nm, tiene puerto de salida 4 y que si ingresa un haz de luz por el puerto 1 con longitud de onda 1550nm, tiene puerto de salida 3.

Este módulo debe recibir en el archivo de inicialización el string *wavelengthToOutputGate* para poder inicializar su tabla de rutas.

Tendrá tantas entradas como lo indique su parámetro *numInputs* y tantas salidas como indique el usuario en la topología que se defina.

En cada entrada y en cada salida tendrá conectada una fibra para poder enviar y recibir los mensajes de los bloques correspondientes. Por esta razón, el *switch* recibe y envía *MessageFiber*.

Todo *switch* invoca a la clase *SwitchMatrix* que guarda la tabla de ruteo, chequea la consistencia y devuelve el puerto de salida en caso de que se solicite. De esta forma, cuando un *switch* recibe un mensaje desde una fibra, según el puerto por donde ingrese y la longitud de onda del mensaje, se obtendrá un puerto de salida (o -1). Si el puerto es un puerto existente se envía el mensaje y si no se elimina.

- **OpticalSwitchFlexi** En este módulo se simula un *switch* óptico para una red *FlexiGrid*. El funcionamiento es muy similar a *OpticalSwitch* pero se tiene en cuenta el ancho de banda de cada señal pues este valor es variable. Cuenta con los mismos parámetros y agrega también un parámetro de **numOutputs** para indicar la cantidad de salidas de este bloque.

  Otra diferencia importante con respecto al bloque se *OpticalSwitch* es que el parámetro para cargar la tabla de rutas se inicializa con un string de la siguiente manera: *InputGate wavelength bandwidth OutputGate; InputGate wavelength bandwidth OutputGate;....*

  Un ejemplo de esto es "0 1550 75 0; 0 1350 50 4;1 1550 100 3" que indica que si ingresa una señal por el puerto 0 con longitud de onda 1550 y ancho de banda 75GHz debe salir por el puerto de salida 0; si llega una señal por el puerto 0 con longitud de onda 1350 y ancho de banda 50GHz, debe salir por el puerto de salida 4, etc. Al enviar la tabla de rutas de esta forma, en *OpticalSwitchFlexi* se puede hacer correctamente el mapeo de [puertoEntrada, *Wavelength*, *Bandwidth*] con un puerto de salida.

  Vale la pena aclarar que este *switch* utiliza las funciones de la clase *SwitchMatrixFlexi*, por lo que se tiene en cuenta el ancho de banda de cada señal para agregar, eliminar y obtener culaquier mapeo.

- **OpticalReceiver** Este bloque simula el comportamiento de un receptor óptico. Al igual que *TrafficGenerator*, está pensado para ser utilizado en el caso en que la red es completamente óptica. En consecuencia, este bloque está pensado para recibir mensajes desde una fibra (del tipo *MessageFiber*). Cuenta con un único parámetro de *sensitivity* para indicar la sensibilidad del receptor en dBm. Esto quiere decir que si la potencia del mensaje recibido es mayor o igual al parámetro del receptor, el mensaje podrá ser interpretado correctamente. En caso contrario, el mensaje será eliminado. Como este bloque está pensado para ser usado en redes únicamente ópticas, no enviará los mensajes recibidos hacia ningún otro bloque, sino que actuará como "sumidero". En consecuencia, *OpticalReceiver* tendrá tantas entradas como desee el usuario pero no cuenta con ninguna salida.

Dentro de este módulo, si la potencia del mensaje recibido es mayor o igual a la sensibilidad del receptor, se calculan algunos parámetros relevantes para la red óptica. En particular se calcula el *delay* que sufrió el bloque a través de toda la red óptica y el *throughput* efectivo.

Para dejar más en claro cómo es la red óptica implementada se muestra un esquema en la figura 4.8. En esta imagen se ve como el módulo *trafficGenerator* genera un mensaje que es enviado hacia el *transmitter*, donde comienza la red óptica. El *transmitter* está conectado a un *switch* mediante fibra óptica y el *switch* a su vez está conectado a través de fibra óptica al *receiver*.

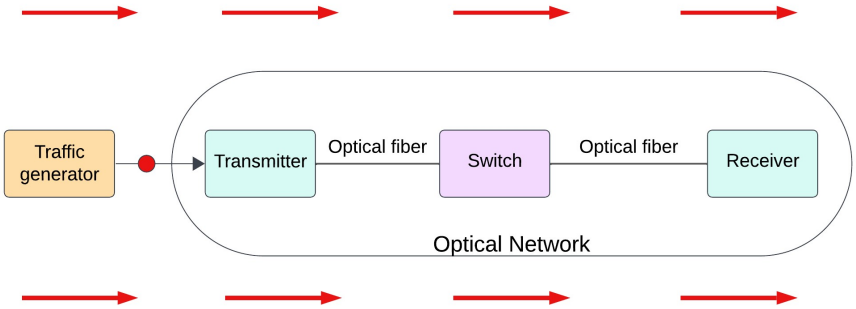

Figura 4.8: Red óptica simple.

Con estos módulos se puede crear redes ópticas más complejas, por ejemplo donde existan más de un *transmitter* y más de un *receiver*. En ese caso el *switch* es el encargado de mandar el mensaje por la fibra óptica correspondiente para que llegue al destino. En la figura 4.9 se observa un ejemplo de una red óptica un poco más compleja. En esta imagen se muestra exclusivamente la red óptica, asumiendo que llega tráfico desde afuera hacia los *transmitters* correspondientes.

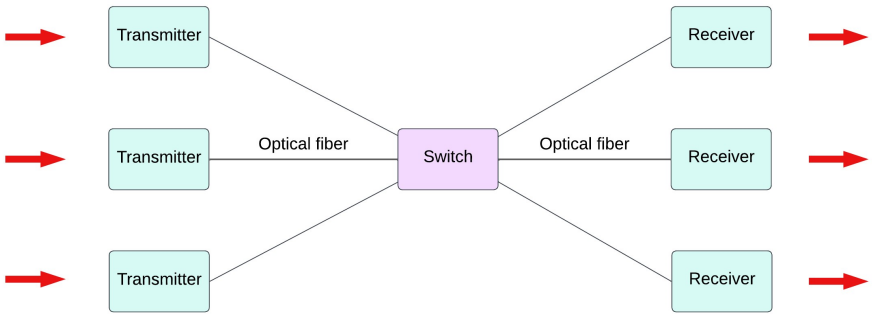

Figura 4.9: Red óptica compleja.

En la figura 4.10 se observa el diagrama de secuencia del sistema del simu-

lador de red óptica. Se aprecia el flujo de mensajes comenzando por el módulo traffic generator. El *transmitter* recibe el mensaje y manda un mensaje message generated hacia la fibra óptica. La fibra a su vez al recibir el mensaje crea un message fiber, el cual es transportado por el *switch* y por otra red óptica hasta llegar el *receiver*.

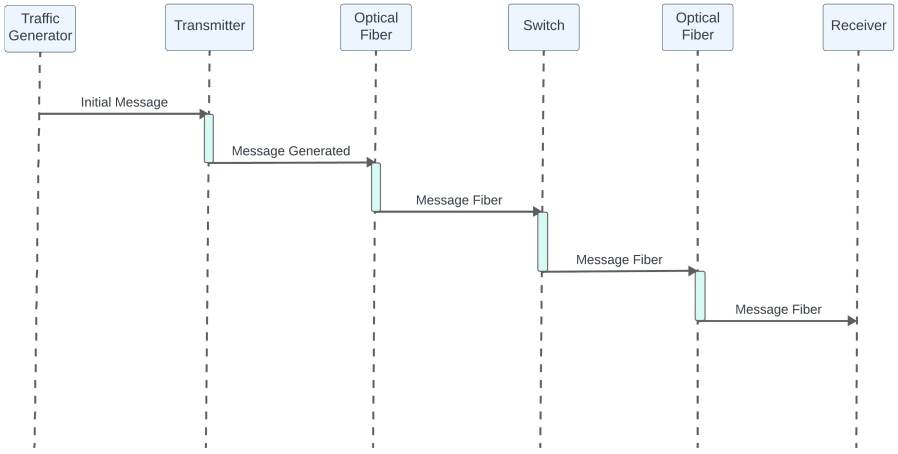

Figura 4.10: DSS red óptica.

A continuación se pasará a explicar todo el proceso de cómo se desarrolló la herramienta de simulación óptico-móvil, teniendo en cuenta los desafíos que conllevo.

4.2. Desarrollo de convergencia

En esta sección se explica la parte más importante del proyecto de grado, el desarrollo de la convergencia entre el mundo óptico y móvil. El resultado de esta sección es la herramienta terminada, que será muy útil para el proyecto CSIC y para el desarrollo de las telecomunicaciones en general, al ser la primera plataforma de simulación óptico-móvil *open source*. Para ello fue necesario todo el estudio previo de ambos dominios y familiarización con la herramienta. A su vez sin ingenio ni creatividad no hubiera sido posible solucionar algunos inconvenientes que surgieron en el camino de la convergencia.

La primer pregunta que se plateó fue: ¿Cómo hacer la conexión entre los dos mundos? Por un lado se tiene la red móvil y por otro la red óptica, ambas funcionando pero la red óptica contaba con un generador de tráfico, que debía ser reemplazado por el tráfico real de la gnb. La conexión directa desde la gnb hacia la red óptica a través del *transmitter* evidentemente no funcionaba. Por esto era necesario la implementación de un nodo nuevo entre ellos para lograr esta conexión.

Surgieron dos ideas del posible rol de este nodo, las cuales involucran distintas técnicas de programación. La primera fue que el nodo tenga un rol de gnb extendido con mejoras para poder comunicarse con la red óptica, o sea el nodo sería una clase derivada de la clase gnb lo que implica que tendría toda la lógica de la comunicación por radio con los ues y a su vez este nodo tenía funcionalidades extra que le permitía comunicarse con la red óptica. Esta idea seguía la filosofía de OMNeT++ al usar la propiedad herencia de la programación orientada a objetos para que el nuevo nodo tenga todas las propiedades de la gnb. Sin embargo luego de algunas pruebas esta idea fue descartada debido a la gran complejidad de la gnb de la cual no todo se ha estudiado en gran profundidad.

La segunda idea era que el nuevo nodo cumpla el rol de traductor entre la gnb y la red óptica. Para ello se creó un nodo que inicialmente podía comunicarse con la red óptica y mandar mensajes a través de ella, conectado un nodo al *transmitter* y otro nodo al *receiver*. El segundo paso fue lograr que el nodo pueda recibir correctamente los mensajes de la red móvil y enviarlos por la red óptica para que llegue al otro lado de la red móvil sin perjuicios ni modificaciones. Esta fue la solución optada, pero surgieron algunas preguntas en el proceso: ¿Cómo se lograba realmente la traducción del mundo móvil al mundo óptico? ¿Dónde cortar exactamente la red móvil para conectar el nuevo nodo junto con la red óptica? ¿El corte iba a ser universal o dependiente de las características de la topología? No es lo mismo pensar en la topología 1 4.6 que en la topología 3 4.7, la cual presenta más gnbs. A continuación se dará la respuesta a todas estas preguntas, que se fueron resolviendo prácticamente en paralelo.

4.2.1. Clase Nodo

En esta sección se desarrolla que comportamiento tiene la clase nodo para lograr cumplir ese rol de traductor entre los dos mundos. Primero se mencionará qué atributos y parámetros tiene, y cuál es la función de cada uno.

Atributos:

- *id* : identificador numérico del nodo en la red.
- *trafficGenerator:* booleano que indica si el nodo genera tráfico o no.
- *numMessages:* cantidad de mensajes que pasaron por ese nodo.

Parámetros:

- *iat (Inter-Arrival Time):* Tiempo entre llegada de eventos sucesivos. Se utiliza como timer para generar tráfico (en caso de ser *trafficGenerator*).
- *trafficGenerator:* misma definición que el atributo, es un parámetro también para que el usuario defina en el archivo de configuración si desea que genere o no tráfico.
- *propagationDelay:* Tiempo de retardo en mandar un paquete. Todos los mensajes que manda el nodo por su *gate* de salida se mandan con este retardo.

Como se mencionó anteriormente, primero se busco que fuera compatible con la red óptica, por esta razón se creo el parámetro *trafficGenerator*. Si ese parámetro es *true*, el nodo se inicializa enviandose un mensaje a si mismo cada cierto tiempo (parámetro *iat*) que indica que debe enviar un nuevo mensaje por su *gate* de salida. De esta manera se conectó un nodo (que sea generador de tráfico) al trasmisor de la red óptica y otro al receptor y se logró que el tráfico fluyera desde un nodo hacia el otro.

Ahora para la conexión con la red móvil el primer paso fue resolver la traducción de los mensajes provenientes de la gnb para que pasaran por la red óptica sin problema. Para ello la solución encontrada fue crear un nuevo mensaje llamado *Container*, este mensaje debía ser compatible con los mensajes que recibía la red óptica pero a su vez debía guardar el mensaje original de la gnb. Este mensaje será enviado por el nodo inicial y una vez fluya por toda la red óptica, lo recibirá el nodo receptor. A continuación se detalla con más exactitud las características de este nuevo mensaje y algunas leves modificiaciones que sufrió la red óptica para lograr pasar este mensaje.

4.2.2. Mensaje Container

Este mensaje cuenta con los siguientes atributos:

- *source*: Es el identificador del nodo que crea el mensaje.
- *target*: Es el identificador del *receiver* al cual esta dirigido el mensaje una vez que termine su recorrido por la red óptica.
- *size*: Es el tamaño en Mb del mensaje original enviado por la gnb contenido en este mensaje.
- *msg*: Es el mensaje original enviado por la gnb.

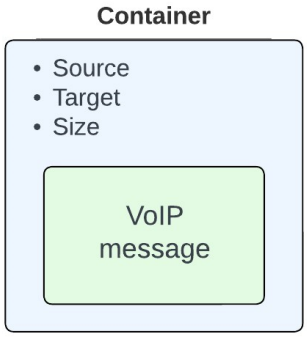

Figura 4.11: Message Container.

Los primeros tres atributos están exclusivamente porque son necesarios dentro de la red óptica. Mientras tanto el último atributo que contiene el mensaje

original enviado por la gnb, es el más importante ya que gracias a este, el mensaje *Container* puede viajar por la red óptica con el mensaje encapsulado y una vez salga de la red óptica, este mensaje se puede recuperar sin sufrir ningún daño.

En la figura 4.11 se ve ilustrado como es el mensaje *Container*. En este caso, el mensaje original a pasar por la red óptica es de tipo VoIP. Este mensaje es encapsulado por el mensaje *Container* y de esta manera puede pasar sin problemas por la red óptica.

Además se describen los cambios realizados sobre el simulador de red óptica base. Entre ellos se tienen:

- Se modificó el comportamiento de la clase *OpticalTransmitter* para que pueda recibir este tipo de mensaje y extraiga su información.
- Se modificaron los mensajes *MessageFiber* y *MessageGenerated* para que guarden un atributo más que sería este mensaje encapsulado.
- Se modificó el comportamiento de la clase *OpticalReceiver* para que una vez reciba el mensaje de la fibra óptica, si este cumple que la potencia de señal es la suficiente, lo envie al nodo receptor. Para ello a su vez fue necesario agregarle una *gate* de salida *"out"* que esté conectada al nodo receptor.

Ahora, ¿Cómo hace uso el nodo traductor de este nuevo mensaje *Container*? Primero es importante aclarar que son necesarios dos nodos traductores por cada flujo de red óptica. Uno va a estar conectado al *transmitter* por su entrada *"in"* y otro al receptor por su salida *"out"*.

El primer nodo va a estar conectado a su vez con la red móvil, la cual enviará un mensaje a través del mismo. Este nodo va crear un nuevo mensaje *Container* donde guardará el mensaje original de la red móvil. De esta manera realiza la traducción del mundo móvil al mundo óptico.

Una vez que el mensaje atraviese toda la red óptica, el nodo receptor va a recibir este mensaje *Container* y va a extraer el mensaje original de la red móvil. Este mensaje no recibió ningún cambio y va a ser enviado para afuera de la red óptica. De esta manera el pasaje por la red óptca es totalmente invisible a los ojos de la red móvil y así se logra la traducción del mundo óptico al mundo móvil. Por lo que el nodo cumple el rol de traductor bilateral. En la figura 4.12 se aprecia el flujo.

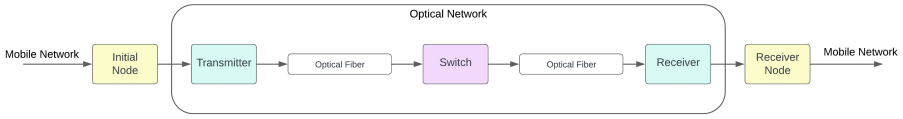

Figura 4.12: Nodos traductores unidos con la red óptica.

Es importante aclarar que en una simulación de la topología creada, para que los mensajes fluyan de manera correcta y a su vez para una mejor comprensión

del flujo por parte del usuario, es necesario usar la nomenclatura adecuada en cada uno de estos módulos. Los detalles más específicos de la nomenclatura se explicarán en el anexo D para los lectores que deseen conocerlos. Sin embargo es necesario saber que la simulación es sensible a los nombres de los módulos para que funcione correctamente.

Siguiendo la lista de preguntas planteadas, fue necesario resolver en que parte se cortaba la red móvil para poner la red óptica. Para ello se realizaron pensaron varias ideas, partiendo de las distintas topologías que se crearon en el inicio.

4.2.3. Convergencia partiendo de topología 1

Se decidió empezar con la topología más simple, donde solo existía una gnb (figura 4.6), para luego ir sumando complejidad de a medida que se avanzaba en la creación de la herramienta. Se pensaron distintos cortes de la red original para poder colocar la red óptica, entre ellos están:

- Entre el *router* y el *server*: Este corte fue descartado dado que tanto el *router* como el *server* no son componentes esenciales en las redes móviles. Por lo tanto se requeriría otro enfoque en caso de desear probar una topología alternativa que no incluyera *router* ni *server*. De todos modos se probó y se identificó que el *router* no estaba configurado para reconocer el nodo al que estaba conectado, lo que resultaba en la eliminación de los mensajes dirigidos al *server*.
- Entre la gnb y el iUpf: Este corte tampoco resultaría beneficioso dado que la gnb no podría determinar la ubicación del upf debido a la presencia de módulos ajenos en su camino (el nodo). Es importante destacar que la gnb está diseñada para enviar paquetes cuyo destino no conoce (como las ues fuera de su alcance) al upf para su correcto enrutamiento. Por más que existe un upf intermedio (iUpf), es parte lo esperado en un red 5G donde pueden haber muchos upfs conectados entre sí. La gnb debe estar conectada a un upf (ya sea upf o iUpf) para mandar mensajes cuyo destino no conoce, por esta razón no funcionaría el corte.
- Entre el iUpf y el upf: Para este corte aplica el mismo razonamiento que en el corte anterior. La gnb debe poder reconocer la ubicación del upf que va a enrutar el paquete. En este caso el objeto ajeno a la red 5G está entre el upf y iUpf, impidiendo que el iUpf conozca el camino al upf, ya que no conoce el funcionamiento del nodo. Por lo que en el momento de inicialización de los módulos la gnb tampoco va a encontrar el camino al upf indicado, ya que si bien está conectada el iUpf, nada asegura que enviar el paquete por ese camino va a llegar al upf correcto que enrute al paquete.

Dado todos estos impedimentos el corte más razonable es colocar la red óptica entre upf y el *router*. De esta manera la gnb iba a reconocer el camino hacia el upf y se podría replicar también con otras topologías que no tengan *router* y/o *server* al conectarlo luego del upf enrutador. También tiene la ventaja de dejar el núcleo 5GCN unido (formado por iUpf y upf).

A su vez como el *router* ya no pertenece al mundo de la radio, la upf se conecta con el *router* a través de su *gate filterGate*. Esto sugiere que es un buen lugar para colocar la red óptica. Este corte fue implementado y logró un resultado exitoso transmitiendo el mensaje hacia el otro lado. En la figura 4.13 se puede observar el primer paso de este trabajo.

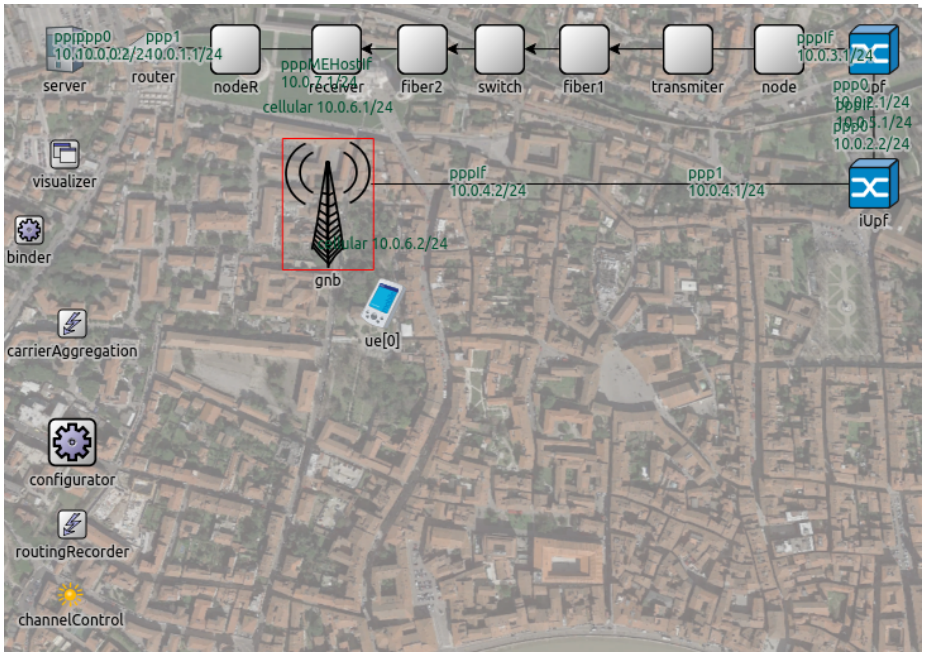

Figura 4.13: Convergencia entre red móvil y red óptica partiendo de topología 1.

Ahora que se logró la conexión desde un ue hacia un *server*, se probará conexiones de ues a ues conectados a distintas antenas.

Esto aumenta la complejidad dado que la red óptica debería estar en el camino entre las 2 gnbs. Lo que quiere decir que el punto de corte debe ser totalmente transparente a la red móvil.

4.2.4. Convergencia partiendo de topología 3

Recordando la topología 3 en la figura 4.7 se puede ver que esta presenta 2 gnbs pero las mismas están conectadas entre ellas. Por lo que el primer paso es eliminar la conexión directa de ambas gnbs y colocar upfs en el medio que ya se sabe que se pueden conectar correctamente con la red óptica. A su vez, se eliminan de la topología el *router* y *server* dado que ahora interesa una conexión de ue a ue.

La nueva topología se ve como en la figura 4.14. Antes de empezar a incorporar los módulos de la red óptica se realizaron distintas simulaciones para

ver si funcionaba correctamente y se probó con distintas cantidades de upfs. El resultado obtenido fue que el upf central debe conocer a ambas gnbs (y las gnbs a dicho upf) para que fluyan correctamente los mensajes. A su vez se necesita un iUpf conectado a cada gnb.

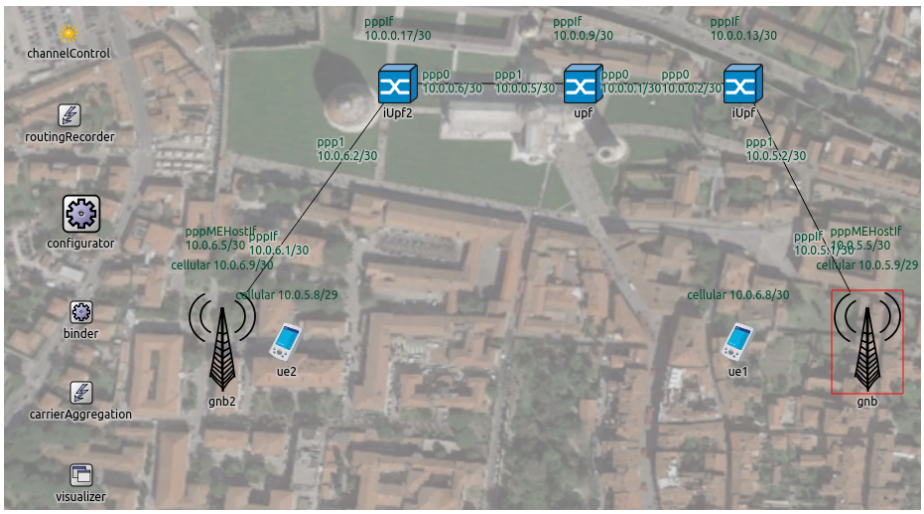

Figura 4.14: Topología 3 modificada.

Se incorporó la red óptica entre el upf y el iUpf2 dado que la gnb que envía tráfico debe tener el camino a la upf sin módulos que no conoce en el camino. Sin embargo surgieron algunos inconvenientes para lograr un flujo de mensajes de ue a ue.

En primer lugar cuando llegaba el mensaje al upf para ser enrutado, este reconocía que el destino final era un ue y no lo mandaba por la salida conectada al nodo. Esto se debe a que el upf se conectó con el nodo a través de su salida *filterGate*, la cual es una salida especial que se utiliza cuando el destino del paquete esta fuera del mundo de la radio, por eso cuando el destino era el *server* funcionaba correctamente. Así que se cambió la conexión con el nodo por su otra *gate* de salida *pppg*.

Si bien se acercaba a la solución, este cambio no fue suficiente, dado que el upf no tenía el destino en su tabla de ruteo y devolvía el paquete por donde venía. El configurador por su lado, colocaba direcciones IP de la misma subred a las distintas gnbs, lo cual era erroneo. Por lo que se pusieron direcciones IP específicas a cada gnb para que pertenezcan a subredes distintas y se añadió manualmente rutas estáticas en la tabla de ruteo del upf que indicarán que si el destino es la IP de la otra gnb, la interfaz de salida debía ser la que estaba conectada al nodo. Estas rutas debían ser lo más específicas posibles, dado que de otra manera el configurador creaba rutas aún más específicas hacia el otro lado.

Finalmente se consiguió una topología completa con conexión de ue a ue a

través de una red óptica. La misma se puede apreciar en la figura 4.15.

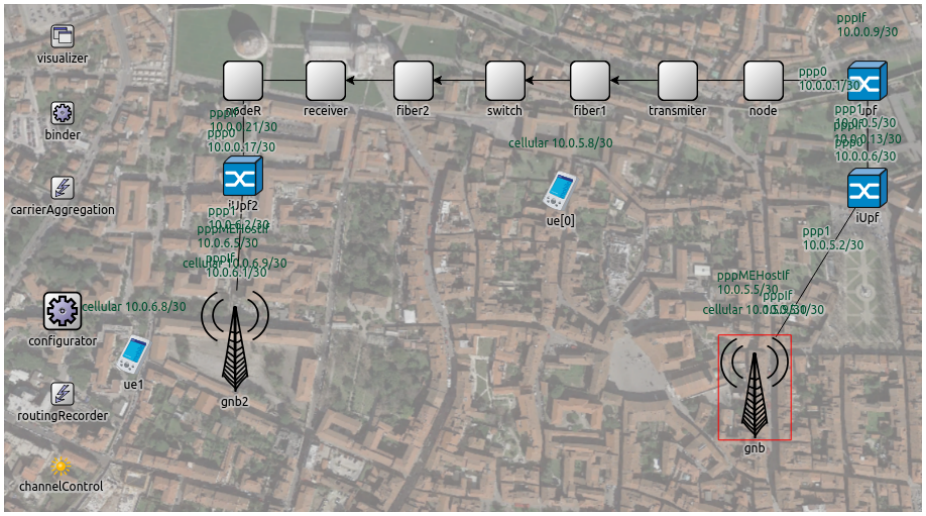

Figura 4.15: Topología 3 con red óptica incorporada.

Hasta ahora solo se han enviado mensajes desde ue[0] hasta ue1. El siguiente paso es lograr una topología donde los mensajes fluyan de manera bilateral.

4.2.5. Topología bilateral

Dado que la red óptica esta diseñada para que los mensajes fluyan de un origen a un destino determinado, impidiendo el flujo contrario, nuevamente surgen interrogantes: ¿Cómo agregar un nuevo camino óptico en el sentido contrario? Para responder esta pregunta se pensaron varias opciones.

La primera idea y también la más intuitiva era conectar un segundo upf con el iUpf2, luego conectar toda la red óptica con nodos incluidos y finalmente el último nodo con el iUpf. Sin embargo lo más intuitivo no siempre es lo correcto, ya que cada gnb tiene un único upf asignado y por ende no puede enviar mensajes a través de otro.

Al conectar la gnb2 a un nuevo upf2, el flujo de mensajes original se perdía ya que la upf desechaba todos los paquetes con destino a gnb2. A su vez, la gnb2 tampoco podía enviar mensajes a través de la upf2 ya que la otra gnb no estaba conectada al upf2, por lo que esta idea fue descartada.

La solución encontrada fue añadiendo las siguientes conexiones y módulos:

- Una nueva conexión desde iUpf2 hasta upf.
- Un nuevo flujo de red óptica y nodos donde el nodo inicial iba conectado con la upf y el nodo receptor conectado al iUpf.

De esta manera para enviar mensajes de una gnb a la otra la upf tiene dos caminos: una siguiendo la red óptica y otra casi directa. Por esta razón no es

muy intuitiva la solución y a su vez se debe configurar la tabla de ruteo de la upf nuevamente para que siempre mande los mensajes que van destinados a una gnb por el camino óptico correspondiente. La topología terminada se ve como en la figura 4.16.

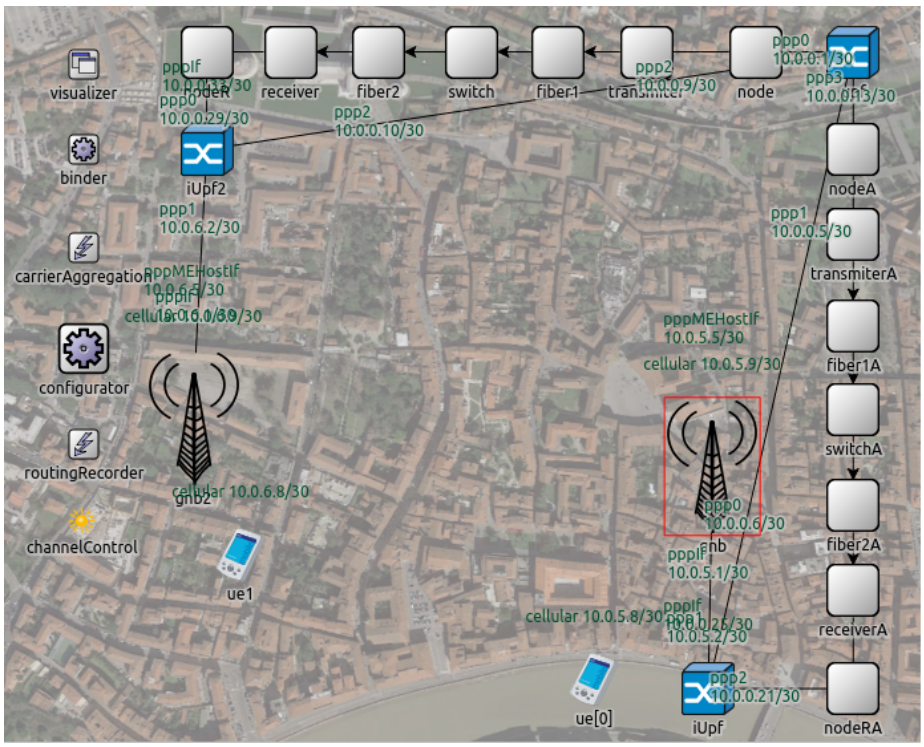

Figura 4.16: Topología bilateral: Las gnbs se comunican a través de dos flujos de redes ópticas.

Finalmente sólo queda realizar distintos experimentos variando la cantidad de ues, cantidad de gnbs, tipo de tráfico, etc. Los resultados obtenidos de estos experimentos serán de gran utilidad para continuar avanzando en la unión de redes ópticas y redes 5G.

Hasta ahora se explicó como fue la creación la solución para la convergencia de la red móvil y la red óptica. Sin embargo, no se ha explicado aún como se da el flujo de eventos una vez comience la simulación.

4.2.6. Flujo de eventos

A continuación se resume el flujo básico de eventos que se dan en una simulación de la topología creada, donde un ue manda tráfico a otro ue a través de la red óptica.

Para comenzar, en la etapa de inicialización, los ues y las gnbs de la topología se mandan mensajes mutuamente (ues con gnbs) para determinar la lejanía de cada gnb con cada ue. Luego de este saludo se asocian cada ue a la gnb que tengan más próxima. Cada tanto también se mandan mensajes de actualización ($feedback\_pkt$).

En un momento dado el ue encargado de generar tráfico manda un mensaje con destino al ue receptor. La gnb recibe dicho mensaje y como no conoce la dirección IP del destino (ya que el ue receptor no pertenece a los ues que tiene asociado) lo manda a la upf, para eso atraviesa primero iUpf.

La Upf reconoce que el mensaje es para ella y deja que una capa más alta determine el destino real. En este momento se determina el destino final del mensaje y que para llegar dicho destino debe mandarlo a la gnb2 (por ser la radiobase asociada al ue destino del mensaje). La upf se fija en su tabla de ruteo para dónde debe enviar el mensaje con destino a gnb2 y encuentra la interfaz de salida conectada con el nodo.

En este momento el mensaje sale del dominio móvil para ser procesado por el nodo y entrar al dominio óptico. Una vez que el mensaje llega al nodo, este primero chequea el tipo del mensaje para corroborar si viene desde la red óptica o la red móvil. Luego el nodo crea un mensaje *Container* en donde encapsula el mensaje que le llegó y guarda algunos datos de control para la red óptica, entre ellos la fuente (él mismo), el destino (receptor adecuado) y tamaño del mensaje original encapsulado. Este mensaje es enviado hacia el transmisor óptico, donde oficialmente inicia la red óptica.

El *transmitter* recibe el mensaje *Container* y en base a este crea un nuevo mensaje *MessageGenerated*. Este mensaje incluye los mismos atributos, incluyendo el mensaje original encapsulado, y además algunos extra del *transmitter* (entre ellos *wavelength*, *bandwidth*, *power*, *bit rate*). Este mensaje es enviado a la fibra óptica.

Una vez el mensaje llega al final de la red óptica (*receiver*), este evalúa si la potencia de señal es mayor a la sensibilidad y en ese caso lo envía al nodo receptor.

El nodo receptor se encarga de hacer el decapsulamiento adecuado para mandar el mensaje original hacia la gnb2 y que llegue correctamente al destino. Cuando el mensaje sale desde el nodo receptor hacia la gnb2 vuelve a entrar en el dominio móvil. La red óptica en el medio de la comunicación es totalmente invisible para la red móvil y viceversa.

En las figuras 4.18 4.17 se observa el diagrama de secuencia del sistema del flujo de una simulación donde el ue desea mandar mensajes VoIP a ueR. Para simplificar el diagrama se utiliza una única entidad para representar la red óptica.

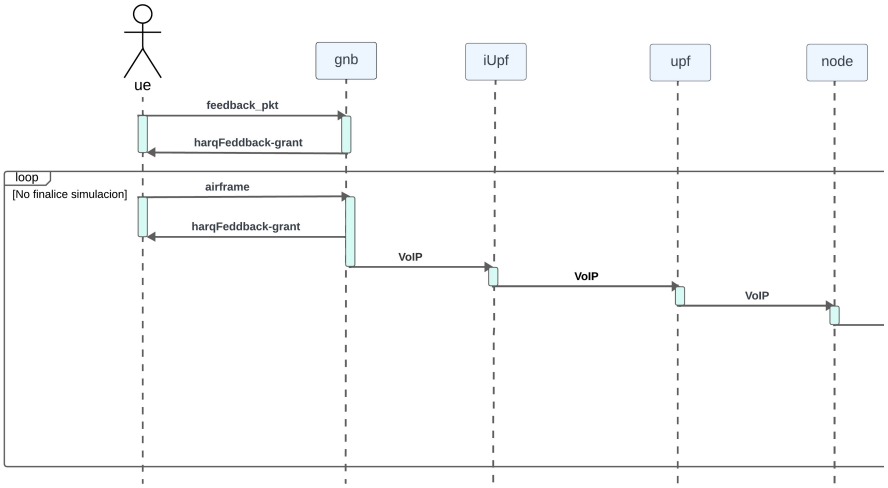

Figura 4.17: DSS simulación parte 1.

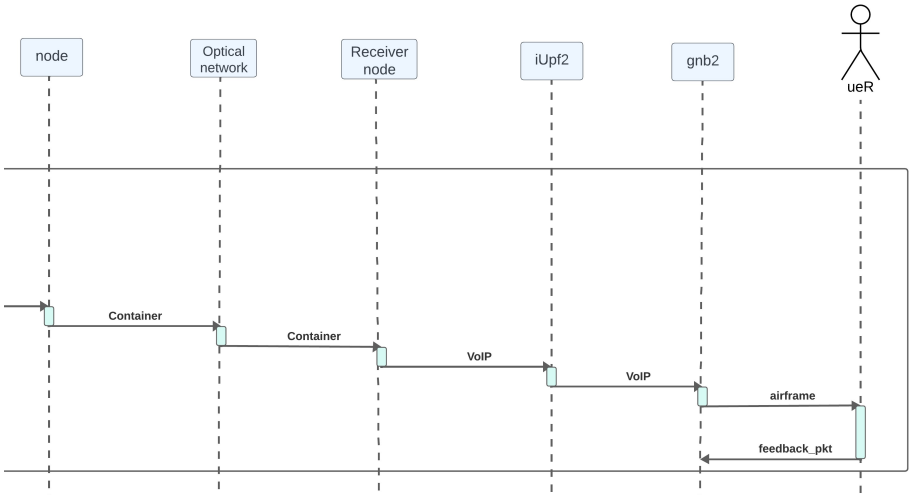

Figura 4.18: DSS simulación parte 2.

Capítulo 5

Experimentación

En este capítulo se presentarán las simulaciones realizadas con la herramienta creada. Incluye a su vez una guía de cómo correr una simulación, que detalles hay que tener en cuenta, qué cosas son parametrizables y algunos experimentos realizados en distintas topologías junto con sus resultados.

5.1. Cómo correr una simulación

Topología y configuración

Lo primero que se necesita es tener una topología definida, donde sus módulos tengan sentido para lograr una corrida exitosa. Luego se continúa creando un archivo de configuración .ini, en donde se darán características específicas de la simulación a realizar.

En la figura 5.1 se muestra un esquema de una topología simplificada de ejemplo para explicar cuales son los pasos a seguir.

1. El primer paso es definir los módulos de la topología y sus respectivos nombres. Uno de los cuidados a tener es que muchos módulos tienen un nombre predeterminado para que funcionen correctamente o necesitan de otro módulo con un nombre predeterminado. En las configuraciones específicas 5.2.2 se verá un ejemplo.

   Otro cuidado a tener es cómo modelar cuando se necesitan muchas instancias de un módulo, es posible crearlas separadas o crear un módulo vector. Para tomar esta decisión es necesario pensar si se tiene el número de instancias de ese módulo a crear o es un número variable. Por ejemplo en la figura se mezclan ambas técnicas para modelar los ues (creando dos instancias de módulos vectores). Se definió de esta manera dado que la cantidad de ues es variable, por lo que sería util tener un vector de ues, pero por otro lado interesa diferenciar los ues conectados a distintas gnbs para diferenciar distintos grupos de comportamiento. De esta manera se

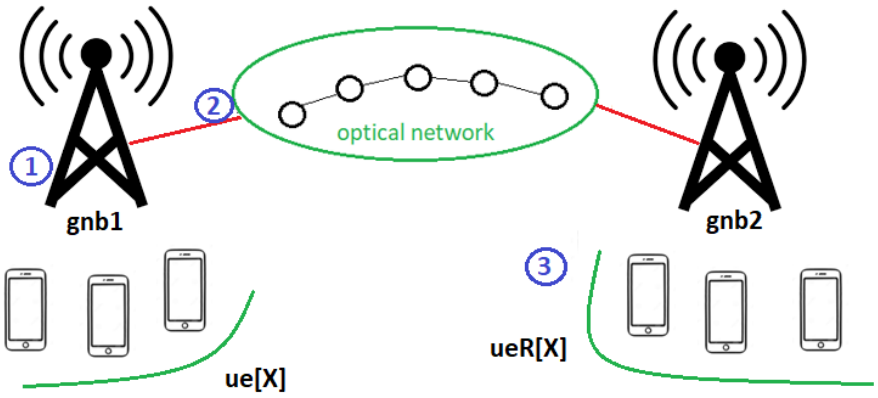

Figura 5.1: Creación de una topología: qué se debe definir y qué cuidados se deben que tener.

tienen un vector de ues conectados a la gnb1 (ue[X]) y otro vector de ues conectado a la gnb2 (ueR[X]).

2. El segundo paso es definir correctamente las conexiones entre los distintos módulos. Por ejemplo en la figura la gnb1 está conectada a la red óptica y la red óptica a su vez está conectada con la gnb2. Las conexiones dentro de la red óptica también deben ser definidas.
3. El tercer paso es definir cuales son los parámetros de la topología. En la figura se observan dos vectores de ues de tamaño variable "X", por lo tanto "X" es un parámetro de la topología, al que se debe asignarle uno o varios valores en el archivo de configuración .ini. Se pueden definir todos los parámetros que sean necesarios y también se les puede definir un valor por defecto, en ese caso si no se específica su valor en el .ini se tomará el valor por defecto.

Luego de terminar la topología es necesario seguir con el archivo de configuración. Normalmente se le llama omnetpp.ini y tiene una estructura como la de la figura 5.2, la cual se explica a continuación.

Se pueden crear muchos tipos de configuraciones dentro de un mismo archivo .ini, cada uno de ellos se representa entre corchetes rectos como se muestra en 5.2 (*Config1* y *Config2*). Algunas configuraciones pueden ser extensiones de otras. Una práctica usual es crear una configuración lo más genérica posible y luego crear más configuraciones que extiendan de esta.

Se presentarán los ítems mínimos necesarios para cada configuración, que son los que indica la figura. Cada configuración debe tener especificado cual es la topología que se va a simular y cual es el tiempo de simulación.

Luego se deberá indicar un valor para cada parámetro obligatorio de los módulos que comprenden la topología y de la topología misma. Parámetros

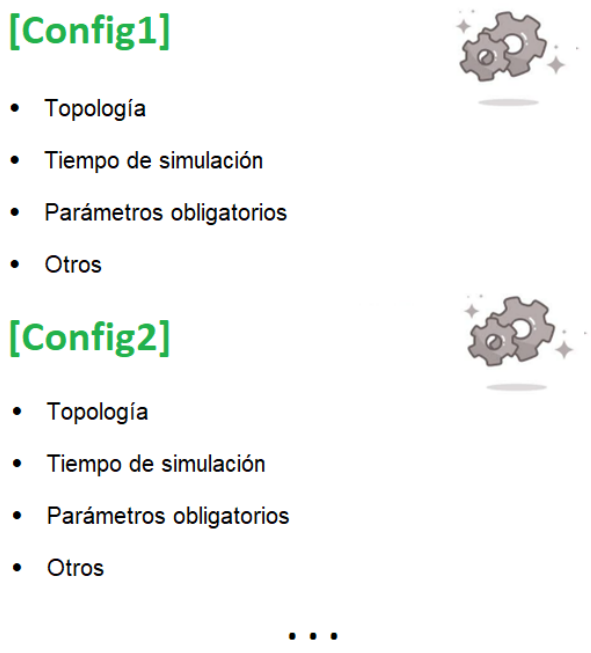

Figura 5.2: Creación del archivo de configuración: estructura principal.

obligatorios refieren a aquellos que no tienen valores por defecto y que si o si necesitan un valor para que todo funcione de la manera esperada.

Para ello es de suma relevancia identificar cuales son los parámetros obligatorios de cada módulo. Muchos módulos tienen parámetros comunes, los cuales se pueden agrupar para no escribir el mismo parámetro cada vez por cada módulo (a menos que se quiera un valor distinto y en ese caso si se deben escribir cada uno por separado). Un ejemplo de parámetros comunes a muchos módulos son los que definen el guardado de estadísticas y el tipo de estadísticas. En la sección 5.2.1 se explica en mayor profundidad cómo agrupar parámetros comunes a muchos módulos.

También es posible asignar un rango de valores a los parámetros en vez de un único valor. Esto permite decidir el valor concreto del parámetro un instante antes del comienzo de la simulación y probar muchas combinaciones de distintos valores de parámetros sin tener que crear una nueva configuración específica. Así mismo, en caso de que quede algún parámetro obligatorio sin definir al momento de correr la simulación también se pedirá el valor específico de ese parámetro.

Más adelante se explicará en detalle todas las configuraciones, parámetros y topologías usadas para el objetivo específico de este proyecto. Una vez finalizados la topología y el archivo .ini OMNeT++ permite correr la simulación por consola o utilizando el simulador gráfico.

5.1.1. Corrida por consola

Por lo general, se ejecuta desde la consola cuando se busca una ejecución más rápida y obtener estadísticas. Para hacerlo, es necesario conocer algunos comandos clave:

```
opp_runall < nombre de archivo .ini >
```

Este comando señala la intención de ejecutar una simulación del archivo especificado. Sin embargo, existen numerosos otros parámetros que se pueden especificar. Para explorar todas las opciones de comandos disponibles, se puede ejecutar opp_runall -h.

A su vez los comandos más usados son:

- `-j <cantidad de procesos en paralelo>`
- `-b <cantidad de batches>`
- `./< nombre del archivo ejecutable>`
- `-u <corrida por consola o por interfaz grafica>`
- `-c <nombre de la configuracin>`
- `-r <numero de instancia de parmetros>`
- `-n < archivos .ned que utiliza la simulacin >`
- `-l < bibliotecas que utiliza la simulacin >`

El comando "Cmdenv" señala que la simulación se ejecuta mediante la consola, mientras que "Qtenv" indica que se ejecuta a través de una interfaz gráfica. Puede resultar tedioso tener que especificar todos los archivos .ned que se utilizan. Una alternativa es configurarlos adecuadamente en la ruta (*path*). A continuación se muestra un ejemplo práctico de una corrida por consola:

```
opp_runall -j1 -b1 ./run -u Cmdenv -n
    ↪ :../src:../../simu5g/emulation:../../simu5g/simulations:../..
    ↪ /simu5g/src:../../inet4.5/examples:../../inet4.5/showcases:..
    ↪ /../inet4.5/src:../../inet4.5/tests/validation:../../
    ↪ inet4.5/tests/networks:../../inet4.5/tutorials -l
    ↪ ../../simu5g/src/simu5g -l ../../inet4.5/src/INET
    ↪ omnetpp.ini -c VoIP-DL -r 0
```

Donde la simulación tiene las siguientes características:

- Un único proceso y un batch (-j1 y -b1).
- El nombre del ejecutable es run.

- Se corre por consola sin abrir el simulador grafico (-u Cmdenv).
- Se utilizan muchas redes .ned (por ejemplo .:../src:../../simu5g/emulation). Se puede observar que entre ellas se separan por dos puntos (:).
- Se utillizan 2 bibliotecas, Simu5G y Inet (-l ../../simu5g/src/simu5g -l ../../inet4.5/src/INET).
- El archivo a correr se llama omnetpp.ini.
- La configuracion a correr se llama VoIP-DL (-c VoIP-DL).
- Se corre la primera opción de los parámetros a elegir (-r 0).

5.1.2. Corrida utilizando el simulador gráfico

Esta ejecución lleva considerablemente más tiempo que la realizada por consola, ya que implica cargar el simulador gráfico con todas sus configuraciones visuales. Sin embargo, la ventaja que ofrece es una representación visual detallada de cada interacción entre módulos, mostrando cada mensaje y evento con precisión. Por supuesto se pueden deshabilitar estas opciones para lograr acelerar la finalización de la simulación.

El procedimiento para correr una simulación es haciendo click derecho en el archivo .ini, presionando en la opción “Run As” y luego seleccionar “OMNeT++ Simulation”. La otra alternativa es definir configuraciones específicas para esa simulación. También existe la posibilidad de debuguear cambiando el segundo paso por presionar la opción “Debug As” .

Seguidamente se inicia el visualizador gráfico y se abre un *pop-up* donde se elige la configuración a correr y los valores de los parámetros que se definieron como rangos de valores. En el ejemplo de la figura 5.3 se seleccionó la configuración “UL-MultiUE” y los parámetros definidos por rango son *numUE* y *TipoT*, que refieren al número de ues y tipo de tráfico respectivamente. La cantidad de ues tenía rango definido entre (1,5,15,50) y el tipo de tráfico entre (VoIP, CBR y Burst), combinando todas las posibilidades se tienen 12 opciones posibles y en este caso se elige la sexta.

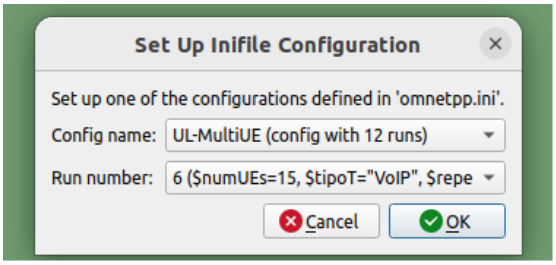

Figura 5.3: Pop-up de elección de configuración para correr simulaciones.

A continuación se explican algunos elementos claves del entorno gráfico de ejecución, los mismos se representan en la figura 5.4. Mientras que algunos

mecanismos pueden ser intuitivos, otros son más complejos y podrían requerir una descripción más detallada.

1. Botón *play*: Se presiona para correr la simulación.
2. Botón avance rápido: Se presiona para acelerar la simulación, sin dejar de mostrar los eventos que suceden gráficamente. A medida que se incrementa el tiempo de simulación, los eventos iniciales se reinician para evitar retrasos en la ejecución.
3. Segundo botón avance rápido: Se presiona para acelerar aún más la simulación. Al presionarlo, se dejan de mostrar los eventos gráficamente, permitiendo que la simulación avance a una velocidad aún mayor.
4. Botón pausa: Se presiona para pausar la simulación.
5. Botón *rebuild*: Se presiona para reconstruir la topología desde cero para ejecutar una simulación.
6. Botón *recorder*: Se presiona para guardar todo lo impreso en la consola durante la simulación en un archivo con extensión .elog.
7. Botón *debug*: Se presiona para correr la simulación en modo *debug*.
8. Barra de aceleración: Permite ajustar manualmente la velocidad de la simulación. Aunque no alcanza la misma aceleración que los botones dedicados, ofrece un control más preciso sobre la velocidad de la ejecución.
9. Pestaña de visualización de consolas: En esta pestaña se puede visualizar la consola de eventos o la consola de logs. En la consola de eventos (que muestra actualmente en la imagen) se ven únicamente los eventos asociados a envíos de mensajes. Incluye el tiempo exacto de envío, emisor, receptor, tipo de mensaje y más información como por ejemplo tamaño del mensaje. La consola de logs muestra todos los eventos ocurridos con sus respectivos logs hasta nivel INFO (a menos que se ejecute en modo DEBUG). Brinda información mucho más detallada.
10. Botón consola de logs: Se presiona para mostrar la consola de logs en la pestaña de visualización. Por otro lado el botón más próximo a la izquierda sirve para cambiar nuevamente a la consola de eventos.
11. Botón de filtro: Al presionarlo brinda la posibilidad de filtrar el contenido que se muestra en la pestaña de visualización por módulos o submódulos.
12. Indicador de tiempo: Muestra el tiempo de simulación.
13. Indicador de evento: Muestra el último evento de la simulación actual.

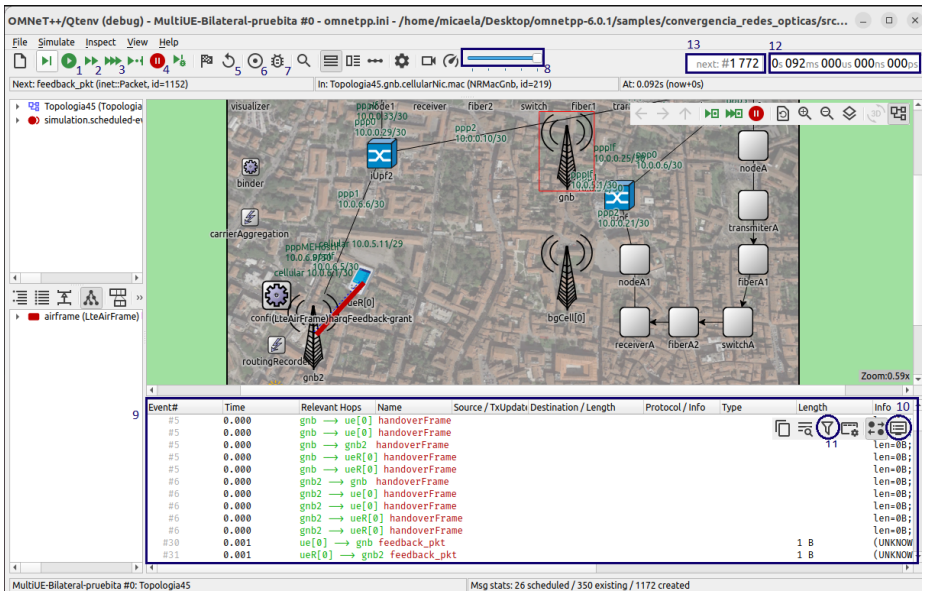

Figura 5.4: Entorno gráfico de simulación.

5.2. Preparación para simulaciones

5.2.1. Cómo funciona el archivo de configuración

Ahora que se han explicado las reglas generales, se procederá al análisis del caso concreto. Particularmente se realizaron pruebas con la topología que conectaba ues con el *server* 5.5 y la que conectaba ues con ues 5.6, ambas con una red óptica en el medio.

En un archivo de configuración generalmente se utilizan los asteriscos (*) como comodines para especificar múltiples entradas que comparten un patrón común. Se pueden colocar ningún asterisco, uno o dos dependiendo del caso.

Por ejemplo:

- Ningún asterisco: Indica que la configuración se aplica solo al módulo o parámetro específico indicado. En este caso se indica la red que se simulará.

  ```
  network = Topologia4
  ```

- Un asterisco: Indica que la configuración se aplica a todos los módulos directamente bajo la jerarquía especificada. El siguiente ejemplo aplica para todos los módulos llamados ue que tengan como submódulo o parámetro *MacCellId* en el nivel específico de la jerarquía. A su vez como hay un

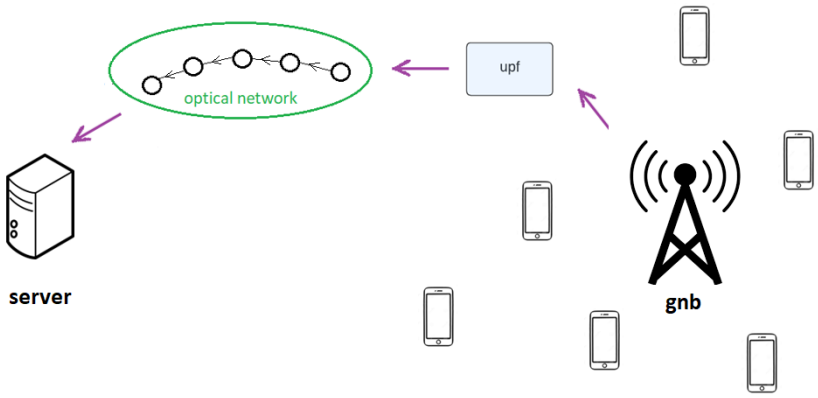

Figura 5.5: Esquema de topología que conecta ues a *server* a través de una red óptica.

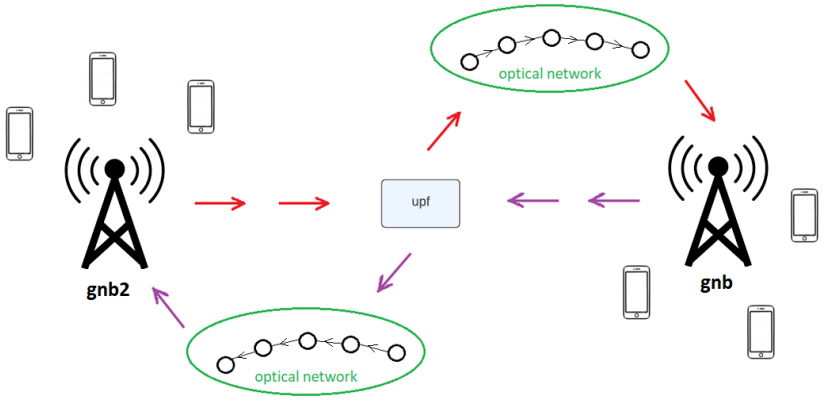

Figura 5.6: Esquema de topología que conecta ues a ues a través de dos redes ópticas.

asterísco en la índice del arreglo, significa que aplica para cualquier posición del vector ue.

```
*.ue[*].macCellId = 0
```

- Dos asteriscos: Indican que la configuración se aplica a todos los módulos, independientemente de su nivel en la jerarquía. En el ejemplos significa que cualquier módulo o submódulo que tenga como parámetro *downlink_interference* se verá afectado.

```
**.downlink_interference = false
```

En la siguiente sección se explicará concretamente los detalles de la topología y de la configuración para lograr una corrida exitosa.

5.2.2. Preparación de Topología

Al igual que se le asigna parámetros a los módulos también se le asigna a las topologías. Los parámetros creados en este caso fueron:

- *numUe*: Número de ues por cada gnb.
- *tamanioPaquete*: Tamaño de paquetes que fluyen por la red.
- *frecuenciaEnvio*: Frecuencia de envío entre cada paquete.
- *tiposTraficos* : Tipos de tráfico de los paquetes.

Estos parámetros se definieron en rangos para crear distintas combinaciones de los mismos, lo cual resulta de utilidad cuando se quieren correr muchas simulaciones a la vez.

Los rangos se definen de la siguiente manera dentro del archivo .ini:

```
*.numUe = ${numUEs=1, 10, 20, 50}
*.tamanioPaquete = ${tamanio= 300, 5000,10000}
*.frecuenciaEnvio = ${frecuencia = 0.02s, 0.01s, 0.005s}
*.tiposTraficos = ${tipoT = "VoIP" , "Cbr", "Burst"}
```

Conexiones entre módulos

Es importante tener en cuenta que cuando se definen las conexiones es importante evitar la constante modificación en su orden ya que esto podría alterar el comportamiento de la simulación. Una vez que se ha establecido el orden de las conexiones, cualquier modificación debe considerar las rutas estáticas definidas, puesto que algunos *gates* son vectores, lo que significa que al alterar el orden se modifica el índice y, por consiguiente, la interfaz asociada.

Por ejemplo si en la parte de conexiones de la topología se escribe:

```
upf.pppg++ <---> node.in
upf.pppg++ <---> iUpf.pppg++
```

Entonces se tendrán unas conexiones como en la parte superior de la figura 5.7. Sin embargo si se invierte el orden escribiendo:

```
upf.pppg++ <---> iUpf.pppg++
upf.pppg++ <---> node.in
```

Se obtendrán unas conexiones como las de la parte inferior de la figura 5.7. A pesar de parecer un aspecto simple, teniendo en cuenta que hay que configurar las rutas del upf para que mande el tráfico a través del nodo (y por ende colocar la interfaz que conecta con el nodo), si no se tiene cuidado en este proceso existe el riesgo de enviar el tráfico por la ruta incorrecta.

Entre otros temas es relevante tener en cuenta que en la topología 4.13 el upf se conecta a nodo con su *gate filterGate* ya que el destino es el *server* y

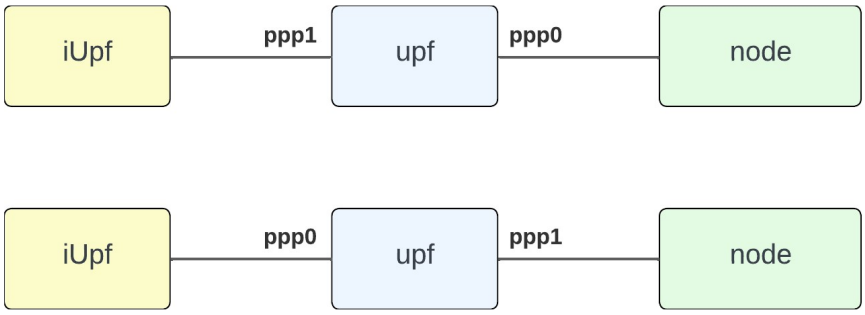

Figura 5.7: Cambio de interfaces por distinto orden de conexiones.

esta fuera del mundo de la radio. Por otro lado para la topología 4.16 la upf debe conectarse al nodo a través de su *gate pppg*. En la figura 5.8 se muestra un esquema de ambas topologías, recalcando la importancia de las conexiones.

Para finalizar, una consideración relevante de la topología de ues a ues implica que las dos gnbs deben estar conectadas al mismo upf. De otro modo la comunicación entre ellas no será posible. A continuación se detallan con mucha mayor granularidad configuraciones posibles del ini. para poder realizar simulaciones lo más completas posibles

Configuraciones generales:

A continuación se brinda una lista con algunos parámetros generales necesarios para las simulaciones realizadas:

- *network*: indica la topología.
- *sim-time-limit*: indica el tiempo de simulación en segundos (ej: 1s).
- *mobility*: parámetro que define el área donde se pueden mover los objetos de la simulación.
- *vector-recording*: parámetro que define si se guardan las estadísticas del tipo vector de él o los elementos que matchen (existe análogamente un parámetro para cada tipo de estadística también).

Direcciones IP y rutas:

Para configurar las direcciones IP se debe escribir todas las instrucciones en un xml que es pasado al parámetro *config* del módulo configurator (encargado de esta tarea). Como ya se mencionó fue necesario asignar direcciones IPs específicas a las distintas gnbs y añadir las rutas en las tablas de ruteo. En el anexo C se detalla mejor cuales son las configuraciones específicas para los lectores que estén interesados.

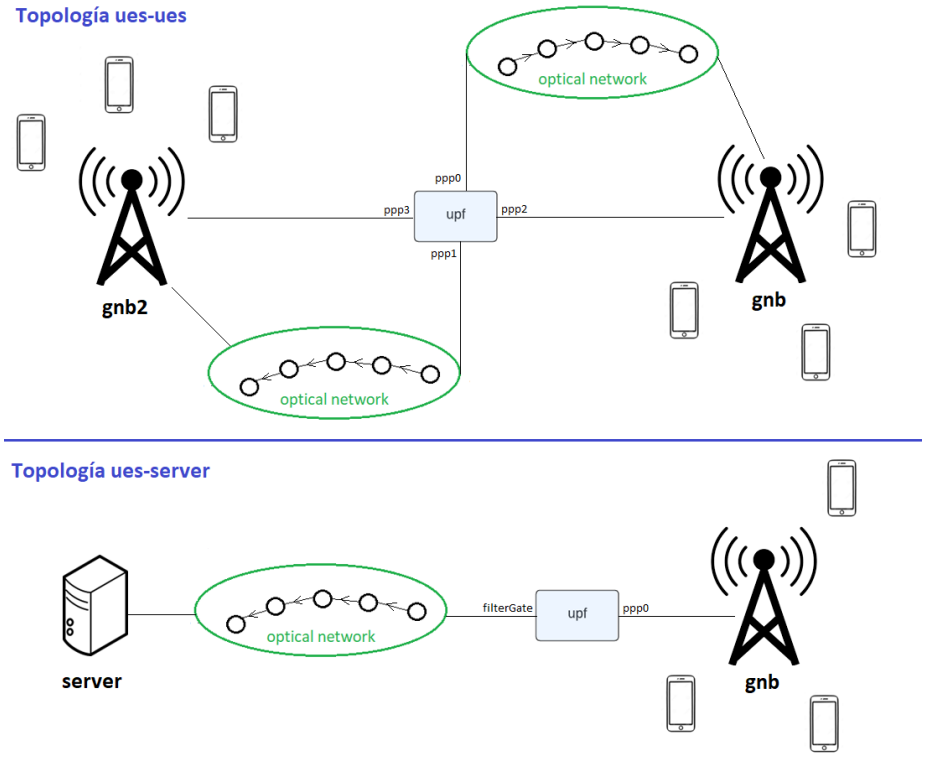

Figura 5.8: Esquema simplificado que indica las conexiones importantes a tener en cuenta del upf en las topologías que van a tener simulaciones.

Cabe destacar que en la topología de ues a ues se añadieron las direcciones IPs y rutas pensadas para un máximo de 100 ues por cada gnb. Esto implica que si se añaden más gnbs o ues hay que escalar las configuraciones de direcciones IPs para que funcione de manera correcta.

Configuraciones básicas de las gnbs

La posición inicial se indica con el parámetro *mobility.initialX* y *mobility.initialY* en metros. Es de suma relevancia configurar una posición específica y tenerla presente a la hora de configuración de la posición de los ues, ya que va a influir directamente en las perdidas de señal en las transmisiones *uplink* y *downlink*.

Se puede indicar algoritmos de *scheduling* con el parámetro *cellularNic.mac.schedulingDisciplineUl* y *cellularNic.mac.schedulingDisciplineDl* para tráfico *uplink* y *downlink* respectivamente. Las opciones disponibles son *Round Robin*(RR), *Proportional Fairness* (PF), *Best Fit* , *Max C/I* y algunas

variantes de este último. Se optó por utilizar PF pero se puede variar.

Un parámetro importante es el *gateway* , que indica el nombre del módulo upf en la topología. Por defecto este parámetro tiene el valor "upf" por lo que no es necesario ponerlo. Pero si el nombre del upf es otro, se debe especificar para todas las gnbs que estén conectadas a dicho upf.

También esta disponible el parámetro *cellularNic.bgTrafficGenerator[0].numBgUes* que indica la cantidad generadores de tráfico que se desea, luego hay que configurar cada uno como un módulo aparte.

Configuraciones básicas de los ues

Para cada ue debe tener la siguiente configuración:

```
*.ue[*].macCellId = 0
*.ue[*].masterId = 0
*.ue[*].nrMacCellId = 1
*.ue[*].nrMasterId = 1
```

Esta permite que el *NIC* de cada celular se conecte a la correspondiente gnb.

Luego para que cada ue se asocie a la gnb más cercana se coloca el parámetro *dynamicCellAssociation* en *true*. Opcionalmente se puede habilitar el *handover* con el parámetro *enableHandover*.

También tiene parámetro la posición inicial al igual que las gnbs y en este caso se configuro de manera uniforme y así si se aumenta el número de ues no se tiene que poner una posición fija a cada uno. En la topología de ues a ues se tienen dos grupos de ues y cada uno tiene una posición inicial uniforme pero dentro de cierto rango asi todos los ues se conectan con gnb y ueRs con gnb2.

Además se pueden añadir parámetros de movilidad como el tipo de movilidad y velocidad, por ejemplo:

```
*.ue[*].mobility.typename = "LinearMobility"
*.ue[*].mobility.speed = uniform(600mps,1300.89mps)
```

Cómo mandar mensajes

Sin duda esta es la parte más importante y donde más se ha trabajado para la creación de las distintas simulaciones. Los módulos que han enviado o recibido mensajes trabajados son los ues y *servers*, estos se configuran de la misma manera que se detallará a continuación.

Primero se debe configurar cuantas aplicaciones tiene cada módulo, una por cada servicio. Por ejemplo si un ue quiere mandar y recibir mensajes debe tener al menos dos aplicaciones (una para enviar y otra para recibir). La cantidad de aplicaciones se define mediante el parámetro *numApps*.

Las aplicaciones que se usaron para la simulación utilizaban UDP como protocolo de transporte. Dado que en las aplicaciones que se usaron para simular se priorizó una mayor velocidad sobre la pérdida de paquetes se decidió que usar UDP era la mejor opción, sin embargo se podría cambiar para que se utilice TCP como protocolo de transporte.

Lo siguiente es definir el tipo de aplicación (que indica además el tipo de tráfico) y otros parámetros que dependen de si es emisor o receptor.

Entre los distintos tipos de tráfico se crearon simulaciones centradas en VoIP y CBR, aunque también se probó otros como Burst. Una aplicación de un tipo de tráfico X receptora, solo recibe de ese tráfico X. A continuación se muestra un ejemplo de una pareja de aplicaciones emisora y receptora de tráfico VoIP para observar qué otros parámetros se deben configurar. Cuando la aplicación es receptora de mensajes bastaría solo definir el puerto donde recibirá los mensajes:

```
*.ue[*].app[*].typename = "VoIPReceiver"
*.ue[*].app[*].localPort = 3000
```

Sin embargo si la aplicación es emisora de mensajes necesita más parámetros como los siguientes:

```
*.ue1.app[*].typename = "VoIPSender"
*.ue1.app[*].PacketSize = 300
*.ue1.app[*].destAddress = "ue[0]"
*.ue1.app[*].localPort = 3088
*.ue1.app[*].destPort = 3000
*.ue1.app[*].startTime = uniform(0s,0.02s)
*.ue1[*].app[*].sampling_time = 0.02s
```

Los parámetros definidos indican tipo, tamaño de paquetes en bytes, nombre del módulo destino, puerto local, puerto destino, el tiempo de simulación donde inicia a mandar mensajes y cada cuanto tiempo los manda respectivamente.

Los parámetros pueden variar dependiendo del tipo de aplicación. En caso de la aplicación VoIP que tiene tiempos de habla y de silencios se pueden configurar algunos parámetros más. La duración del habla y del silencio depende de la distribución de Weibull, la cual es una distribución de probabilidad continua que se utiliza normalmente para representar tiempos de vida, tiempos de fallo y otros fenómenos relacionados con el tiempo. Tiene los parámetros *scale* y *shape* que determinan la anchura y forma de la distribución tanto para el habla como para el silencio (*scaletalk*, *shapetalk*, *scalesil*, *shapesil*). Si *shape* es mayor a 1 la distribución es ascendente, si es menor a 1 es desendente y si es igual a 1 es exponencial. En este caso los valores de *scale* y *shape* son los dejados por defecto:

```
shape_talk = default(0.824);
scale_talk = default(1.423);
shape_sil = default(1.089);
scale_sil = default(0.899);
```

Por otro lado es importante tener en cuenta que si la aplicación es VoIP, la frecuencia del envío de mensajes será valida únicamente si se esta en período de habla. Por ejemplo si tiene 3 segundos de duración de habla y un *sampling time* de 0,5s entonces se mandarán 6 paquetes cada 0,5 segundos y luego iniciará un silencio.

Volviendo a los parámetros de la parte superior de la aplicación emisora, estos mismos parámetros sirven también si el tráfico es CBR, únicamente se debe cambiar el tipo de aplicación a "CbrReceiver" y "CbrReceiver" respectivamente. Si la aplicación es de tipo "CbrReceiver" a su vez se puede configurar un parámetro de tiempo de finalización de envío de mensajes llamado *finishTime*.

Para otras aplicaciones se tienen otros parámetros, por ejemplo para la aplicación Burst existe el párametro *burstSize* que indica el número de paquetes por ráfaga.

Configuración de channelControl y carrierAgregation

En este caso no se ha modificado nada de lo que está por defecto pero se deja constancia por si se quisiera variar algún parámetro en el futuro. Con respecto al carrierAgregation se tiene:

- *carrierFrequency*: Por defecto esta en 2GHz.
- *numBands*: Tamaño del espectro.
- *numerologyIndex* : Índice de numerología.
- *useTdd*: Por defecto esta en *false* pero en caso de que sea *true* se pueden configurar los dos parámetros inferiores.
- *tddNumSymbolsDl*: número de símbolos para *downlink*.
- *tddNumSymbolsUl*: análogo pero para *uplink*.

Por otro lado del channelControl se tiene:

- *pMax*: Máxima potencia de envío usada en la topología en mW.
- *carrierFrequency*: Frecuencia portadora base de todos los canales.
- *alpha*: Coeficiente de pérdida de caminos.
- *numChannels*: Número de canales de radio.
- *propagationModel*: Método de propagación por aire.
  De este último se tienen entre algunos modelos:
  - *FreeSpaceModel*.
  - *TwoRayGroundModel*.
  - *RiceModel*.
  - *RayleighModel*.
  - *NakagamiModel*.
  - *LogNormalShadowingModel*.

Configuraciones de la red óptica

A continuación se muestran los parámetros utilizados para la red óptica. Como ya se explicó cada uno en capítulos anteriores simplemente se muestra cuales son los valores que se usaron en cada parámetro para las simulaciones.

Transmisor óptico:

```
**.wavelength = 1550nm
**.minPower = -10dBm
**.maxPower = 10dBm
**.launchPowerVariation = 20dB
**.bit_rate = 10Gbps
```

Fibra óptica:

```
**.distance = 100km
**.alpha_db = 0.2dB
**.D = 16ps
**.gamma = 0.001km
```

Switch:

```
**.wavelengthRoutingTable = "1550 0"
```

Receptor:

```
**.sensitivity = -200dBm
```

Nodo:

```
*.node*.iat = 0s
*.node*.propagation_delay = 0.001s
*.node*.trafficGenerator = false
```

El retardo de propagación de cada módulo de la red óptica no se contempla como parámetro pero es importante mencionarlo. El retardo de la fibra óptica es muy complejo de calcular, por eso se utilizó una aproximación de 0,0005s. Este valor fue determinado teniendo en cuenta el largo de la fibra (100km) y la velocidad de la luz c (300000km/s). Sin embargo como sería demasiado ideal ese retardo y es claro que existen deficiencias en la fibra se aumentó levemente para que sea más realista. Es importante tener en cuenta que si se quiere hacer simulaciones con fibras de otros largos se debe cambiar el valor en el código de *OpticalFiber*.

Por otro lado el retardo del *transmitter* y receptor es de 0,000001 segundos dado que es despreciable en comparación al de la fibra. El retardo del nodo es un poco mayor dado que debe hacer la traducción, pero al ser un parámetro se puede modificar fácilmente al correr simulaciones.

En este momento se cuenta con los recursos necesarios para crear distintas simulaciones, pero antes es necesario analizar que estadísticas se van a capturar. En la siguiente sección se hablará de las estadísticas que fueron más importantes para este proyecto y de cómo se crearon en determinados módulos para conseguirlas.

5.3. Creación de estadísticas

Las estadísticas que más interesan son

- *Delay*.
- *Throughput*.
- Cantidad de paquetes perdidos.

De estas estadísticas resulta interesante visualizarlas a nivel *end-to-end* y de cada segmento (red óptica y red 5G). Hablando puntualmente del *delay*, se captura el *delay end-to-end*. Esto es el tiempo que demora un mensaje desde que es emitido por la fuente hasta que llega a su destino final. Por otro lado también se captura el *delay* en la red óptica, o sea el tiempo desde que el mensaje entra en la red óptica hasta el tiempo que sale. Con estos dos valores se puede deducir el *delay* del segmento de la red 5G.

Respecto al *throughput* también se busca el *thorughput end-to-end* pero de igual manera interesa el *throughput* a nivel de aplicación y en la red óptica. Análogamente se puede calcular el *throughput* a nivel de red 5G.

A su vez se llevará la cuenta de los paquetes que llegaron éxitosamente, los que se perdieron en el camino y los que pasaron por la red óptica.

5.3.1. Estadística de la red óptica

Las estadísticas creadas de la red óptica son enviadas por el módulo *receiver*, por ser el último módulo de la red óptica. El mismo calculaba el *delay* y *throughput end-to-end* de la red óptica y los emitía por cada vez que le llegaba un nuevo mensaje en su función *handlemessage()*.

Para calcular el *delay*, se resta el momento en que se recibió el mensaje en la simulación del tiempo en que se creó el mensaje *Container*, ya que este último marca el instante en que el mensaje entra en la red óptica.

Para el cálculo del *throughput*), se toma el tamaño del mensaje recibido, se convierte a bits y luego se divide por el *delay* calculado previamente.

Ambas métricas generadas incluyen el máximo, mínimo, promedio, recuento (cantidad de veces que se registró) y un vector (necesario para la representación gráfica).

5.3.2. Estadísticas *end-to-end*

Respecto a las estadísticas *end-to-end*, no todas estaban implementadas de forma correcta en las distintas aplicaciones. Por ejemplo en la aplicación de VoIP calculaba el *throughput* a nivel de aplicación pero no el *throughput end-to-end*. Tampoco calculaba correctamente el *delay end-to-end*, dado que mandaba todos los valores obtenidos al final de la simulación y no en el momento que llegaba cada mensaje, por lo que no se apreciaba muy bien la gráfica final.

Por estos motivos se modificó levemente la implementación de la aplicación para poder guardar estas estadísticas. El *delay* se calcula en la función *handlemessage()* cada vez que llega un nuevo mensaje. Se calcula manera similar a la

fibra óptica pero con el mensaje original enviado por el ue, teniendo la precaución de que el tiempo de creación que lleva el mensaje sea efectivamente cuando se creó para no cometer errores en los cálculos. Para el *throughput end-to-end* se utiliza el *delay* creado como denominador y el tamaño del mensaje que se recibió en bit como numerador. En el anexo E se detalla más específicamente el código modificado para recrear estas estadísticas.

A continuación se presenta cada una de las simulaciones creadas.

5.4. Simulaciones

Para mostrar y evaluar la *performance* de la herramienta de simulación *end-to-end* multidominio tecnológico, se presentarán a continuación tres experimentos distintos que se corresponden con tres simulaciones. Cada simulación tendrá distintas características lo que hace que sea más interesante la visualización de distintas estadísticas dependiendo del caso. La primer simulación se realizó en la topología que conectaba ues con el *server* y las otras dos en la topología que conectaba ues con ues.

5.4.1. Simulación 1

Como se mencionó anteriormente esta simulación se dio en la topología de ues a *server* 4.13. La simulación consta de 15 ues que mandan distinto tipo de tráfico. Al inicio de la simulación el único ue que manda tráfico es ue[0] y manda mensajes de tipo VoIP. Luego de que ya pasaron aproximadamente 0,2 s, otros 14 ues empiezan a mandar tráfico CBR y paran en el tiempo de simulación 1,5s. El ue que manda tráfico VoIP nunca para de mandar mensajes hasta que termina la simulación.

En esta simulación puede resultar interesante ver el *delay* de ambos tipos de tráfico (VoIP y CBR) y qué pasa cuando se llena la red de mensajes CBR que antes no estaban.

Se mencionan algunas características de la simulación en particular:

- La simulación se detiene a los 15s.
- Todos los ues mandan tráfico con destino al *server* y es el *server* quien nos dará las estadísticas correspondientes.
- Todos los paquetes enviados son de 300B y se mandan con una separación de 0,02 s (tanto VoIP como CBR).
- Las características de la red óptica son las mismas que se mencionaron en la sección anterior.

En esta simulación se le da prioridad al análisis del *delay* pero también se observa el *throughput* y la cantidad de mensajes que llegaron efectivamente al servidor para entender mejor que está sucediendo.

Graficando el *delay* en el *server* nos queda la siguiente gráfica 5.9. Se utiliza una gráfica de puntos para visualizar mejor en que momentos de la simulación efectivamente llegó un mensaje al *server* y cuánto demoró desde que fue creado.

A su vez también se compara con el *delay* en la red óptica, que refiere al tiempo desde que el mensaje entra a la red óptica hasta que sale.

Observando la gráfica lo primero que se ve es que ni el *delay* VoIP ni el *delay* de la red óptica incrementan demasiado y se mantienen en valores casi constantes. Por otro lado el *delay* del CBR es un poco mayor (manteniéndose en un rango entre los 0,015s y 0,030s) pero también varía un poco más. Como se puede apreciar la mayoría de los mensajes CBR tienen *delay* 0,017s aproximadamente y únicamente unos pocos puntos demoran un poco más. Sin embargo ningún *delay* incrementa drásticamente, ni siquiera en el período que se mandan mucho más tráfico, a partir del segundo 0,2 hasta el segundo 1,5 de la simulación aproximadamente (dado que inicia el tráfico CBR de 14 ues).

Otra particularidad bastante notoria es que hay tiempos en la simulación donde no llega ningún mensaje VoIP hacia el *server* por más que se transmita tráfico durante toda la simulación. Esto se da por la característica de la aplicación VoIP que genera tiempos de habla y tiempos de silencio siguiendo una distribución específica. Por lo que en los tiempos de silencio, que se deciden dinámicamente, no se genera tráfico. Como a partir del segundo 1,5 el único tráfico que llega al *server* es VoIP, estos silencio se ven reflejados también en la red óptica.

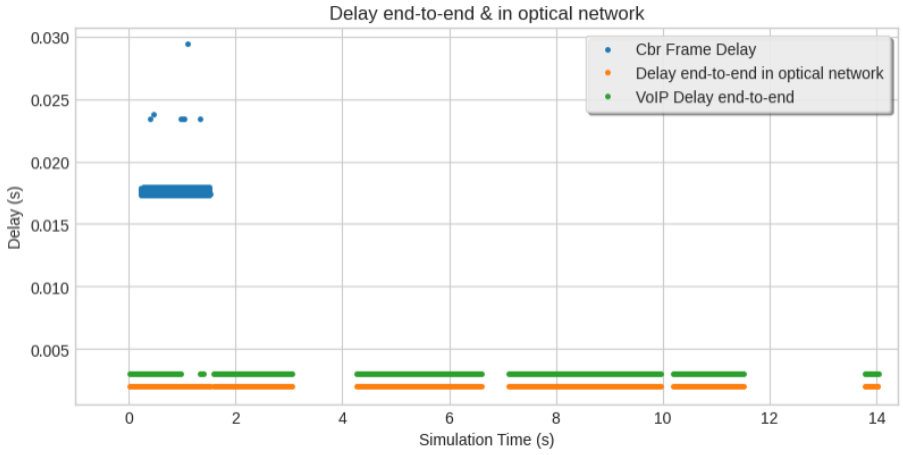

Figura 5.9: *Delay* - Primera simulación.

A continuación se dejan algunos valores claves del *delay*. Dado que en el caso de VoIP y en la red óptica son casi constantes, se dará únicamente un valor. Mientras que el CBR se presentará máximo, mínimo y promedio.

- *Delay* VoIP: 0,003003s.
- *Delay* red óptica: 0,002001s.
- Mínimo *delay* CBR: 0,017365s.
- Máximo *delay* CBR: 0,029388s.

- Promedio *delay* CBR: 0,017693s.

Se observa ahora la cantidad de mensajes que son enviados y los que realmente llegan.

- Cantidad de mensajes VoIP enviados por ue[0] : 467.
- Cantidad de mensajes CBR enviados por cada ue (desde ue[1] hasta ue[14]): Entre 63 y 65 dependiendo del ue.
- Cantidad de mensajes que pasaron por la red óptica: 1368.
- Cantidad de mensajes VoIP recibidos en *server*: 467.
- Cantidad de mensajes CBR recibidos por *server*: 901.

De estos valores se puede llegar a dos conclusiones. La primera es que por la red óptica pasaron únicamente los mensajes VoIP y CBR destinados al *server*, ya que la suma de los mensajes recibidos por el *server* VoIP y CBR es igual a la cantidad de mensajes que pasaron por el receptor de la red óptica. La otra conclusión y positiva también es que la mayoría de los mensajes llegan correctamente, concretamente los mensajes VoIP llegaron todos los que fueron enviados.

A continuación se analiza el *throughput* VoIP tanto a nivel *end-to-end* como a nivel de aplicación. En la figura 5.10 se observa el *throughput end-to-end*. Como se puede apreciar es constante de igual manera que lo es la gráfica de *delay*, también manteniendo sus períodos de silencio. Esto se da porque al no variar el tamaño de paquetes (300B) ni la frecuencia de envío de los mismos (0,02s) esta gráfica depende únicamente de la variación del *delay* y como ya se sabe es constante.

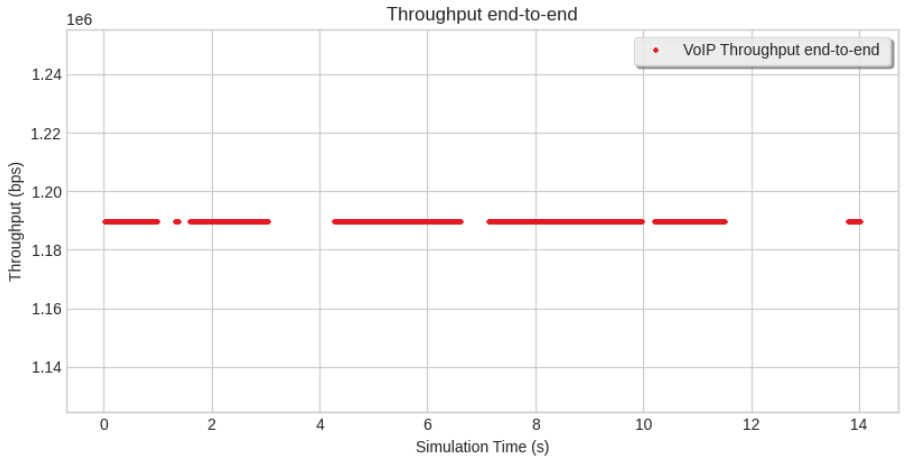

Figura 5.10: *Throughput end-to-end* - Primera Simulación.

Ahora, ¿qué pasa con el *throughput* a nivel de aplicación? Se puede observar en la gráfica 5.11. Primero es importante aclarar como se calcula este *throughput*.

Cada vez que llega un nuevo mensaje a la aplicación VoIP receptora, este suma los bytes del mensaje al total de bytes recibidos hasta el momento y con este número calcula el *throughput* utilizando el tiempo de simulación sin contar el tiempo de *warm up* (preparación de los módulos). Esto significa que se emite una estadística cada vez que llega un nuevo mensaje y el valor emitido es la cantidad de bytes recibidos hasta el momento sobre el tiempo que pasó desde que se terminaron de inicializar todos los módulos.

Como se puede observar esta gráfica empieza bastante constante por unos segundos y luego varía bastante. Esto se debe a los silencios provocados, ya que a nivel de aplicación afecta al variar la frecuencia con la que llegan los mensajes. En condiciones de habla los mensajes llegan siempre a la misma frecuencia (0,02s), pero a nivel de aplicación cuando se da un silencio es como si la frecuencia aumentará drásticamente y luego vuelve a su ritmo. Por eso se dan saltos tan grandes en la gráfica y si se observa la gráfica del *delay* se puede confirmar que los saltos se dan prácticamente al mismo tiempo que empiezan los silencios.

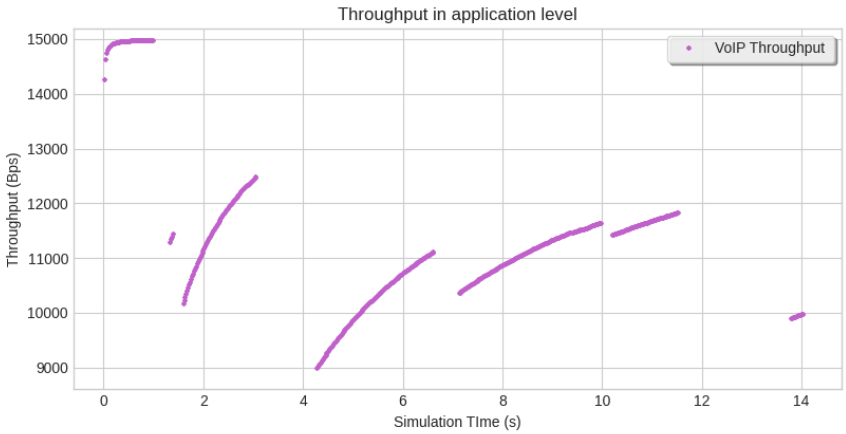

Figura 5.11: *Throughput* a nivel de aplicación - Primera Simulación.

Se dejan algunos valores claves del *throughput* :

- *Throughput end-to-end* VoIP: 799213 bps.
- Mayor *throughput* a nivel de aplicación VoIP : 14984 Bps.
- Menor *throughput* a nivel de aplicación VoIP: 8988 Bps.
- Promedio *throughput* a nivel de aplicación VoIP: 11392 Bps.

5.4.2. Simulación 2

Esta simulación se realiza en la topología donde se envían mensajes de ues a ues 4.16.

La simulación consta un total de 40 ues, de los cuales 20 pertenecen al módulo vector ueR[] y los otros 20 al módulo vector ue[]. Los ues emisores de tráfico son los ue[0..19] y mandan únicamente tráfico VoIP, pero en este caso se va a variar el la frecuencia con la que se envían los paquetes.

Al inicio de la simulación el único ue que manda tráfico es ue[0]. Manda paquetes de 300B, con una frecuencia de 0,01s y con destino ueR[0]. Luego de que ya pasaron aproximadamente 2,2s, otros 4 ues (ue[1..4]) empiezan a mandar tráfico hacia ueR[0] pero con paquetes de 600B y a una frecuencia de 0,005s. O sea el doble de rápido. A su vez a los 5,4s de simulación otros 5 ues (ue[5..9]) empiezan a mandar tráfico hacia ueR[0] también de 600B pero disminuyendo aún más la frecuencia entre mensajes a 0,001s.

Finalmente a los 10,6s de simulación los 10 ues restantes empiezan a mandar paquetes de 600B con frecuencia de 0,0005s al ueR receptor que corresponda su índice en el vector. Por ejemplo ue[10] manda a ueR[10]. La idea de este tráfico es congestionar la red.

En esta simulación se busca visualizar principalmente como varía el *throughput* en ueR[0]. A su vez como también se aumenta la cantidad de ues que envían tráfico por lo que se seguirá analizando el *delay* y otras medidas.

Se mencionan algunas características de la simulación en particular:

- La simulación se detiene a los 25s.
- El único tráfico que se envía es VoIP.
- Los ueR que reciben tráfico son ueR[0] y ueR[10..19]. Los demás no tienen ninguna aplicación.
- Las características de la red óptica son las mismas que se mencionaron en la sección anterior.

Respecto a las cantidades de paquetes enviados y recibidas se tienen las siguientes medidas:

- Cantidad de paquetes enviados por ue[0]: 1616.
- Cantidad de paquetes enviados por ue[1...4] : Entre 3000 y 3500 por ue.
- Cantidad de paquetes enviados por ue[5...9] : Entre 10000 y 18000 por ue.
- Cantidad de paquetes enviados por ue[10...19] : Entre 15000 y 23000 por ue.
- Cantidad de paquetes recibidos por ueR[0]: 52949.
- Cantidad de paquetes que pasan por la red óptica: 172026.

Al realizar los cálculos pertinentes (la suma de la mínima cantidad de paquetes enviados por cada ue), se puede deducir que una parte de los paquetes enviados no llegaron a su destino. Por ejemplo en ueR[0] tendrían que haber llegado un mínimo de 63616 paquetes. En la red óptica ocurre algo similar ya que

deberían haber pasado al menos 213616 considerando todo el tráfico de la red. Esto ocurrió porque al disminuir tanto las frecuencias de envío de paquetes, la red se lleno de muchos más paquetes de los que podía procesar. Por este motivo muchos paquetes seguramente hayan sido descartados.

A continuación se visualizan el *delay* y *throughput* en ueR[0] en las gráficas 5.12 y 5.13 respectivamente. Cabe aclarar que se han dividido las gráficas en franjas para poder visualizar mejor en qué momento se aumenta el tráfico. En total se distinguen 4 franjas y las mismas indican:

1. Franja 0 - 2,2 segundos: Red únicamente con tráfico de ue[0], enviando paquetes VoIP cada 0,01s con destino ueR[0].
2. Franja 2,2 - 5,4 segundos: Se incrementa la cantidad de ues, se añade el tráfico de 4 ues enviando paquetes VoIP cada 0,005s con destino ueR[0].
3. Franja 5,4 - 10,6 segundos: Se incrementa nuevamente la cantidad de ues, se añade el tráfico de 5 ues más que envían paquetes VoIP cada 0,001s con destino ueR[0].
4. Franja 10,6 - 25 segundos: Se incrementa el tráfico en la red pero no para ueR[0] (pues los nuevos ues envían paquetes a otros destinos). Se añade el tráfico de 10 ues que envían paquetes VoIP cada 0,0005s con destinos distintos a ueR[0].

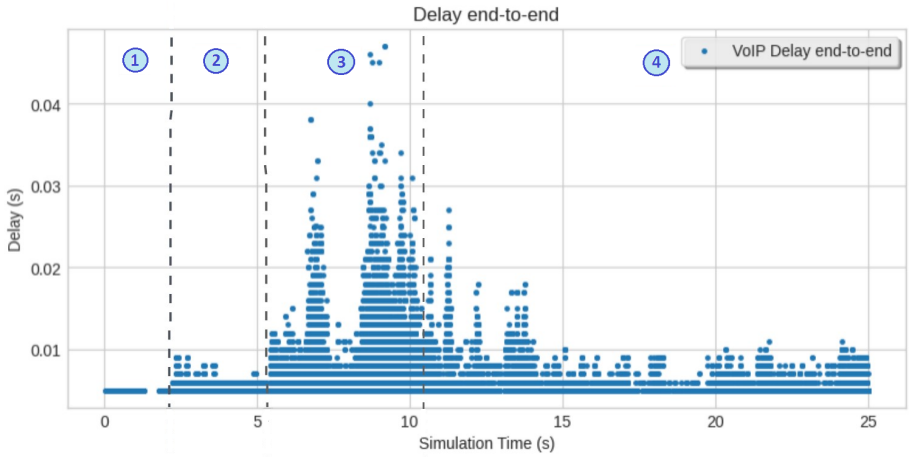

Figura 5.12: *Delay end-to-end* en ueR[0] - Segunda Simulación.

De la gráfica 5.12 se nota que el *delay* empieza en valores bajos pero en la primera franja pero luego aumenta levemente en la segunda franja, pues empiezan a llegar más paquetes y con menores frecuencias. Cuando comienza la 3ra franja varía mucho más, alcanzado incluso valores 4 veces más grandes que en la primera franja. Esto se debe a que recibe mucho más tráfico que antes. Finalmente en la cuarta franja el *delay* baja levemente, pero el tráfico que recibe

ueR[0] no ha cambiado. Lo interesante de ver es que en esta franja aumenta el tráfico en la red pero no en ueR[0], por lo que sería intuitivo pensar que el *delay* se mantuviera mayor o igual a la tercera franja. Sin embargo, lo que realmente ocurre es que como la red se satura se empiezan a perder muchos paquetes, por lo que llegan menos paquetes a ueR[0] y en la gráfica da la impresión de que el *delay* es menor. Efectivamente los paquetes que llegan al destino llegan con menor *delay*, pero muchos otros paquetes se pierden.

Se dejan algunos valores claves del *delay*:

- Mínimo *delay* VoIP: 0,004986s.
- Máximo *delay* VoIP: 0,046996s.
- Promedio *delay* VoIP: 0,006368s.

Ahora se analiza el *throughput end-to-end* en la figura 5.13. Como se puede observar, en la primera franja el *throughput* de mantiene constante. A partir de la segunda franja empieza a variar levemente dado el aumento de tráfico, en esta franja recibe paquetes de 300B y 600B algunos con frecuencia de 0,01s y otros con frecuencia 0,005s.

Un gran cambio se visualiza en la tercera franja dado que varía mucho más, llegando a valores mucho más bajos que antes. En esta franja se aumenta el tráfico de paquetes con una frecuencia de 0,001s, o sea se reciben 5 veces más paquetes que la franja anterior considerando un mismo período de tiempo.

En la última etapa ocurre lo mismo que se aprecia en la gráfica del *delay*. Como se pierden muchos paquetes, da la impresión de que se estabiliza la el *throughput* pero llegan muchos menos paquetes y por eso se visualizan menos puntos en la gráfica.

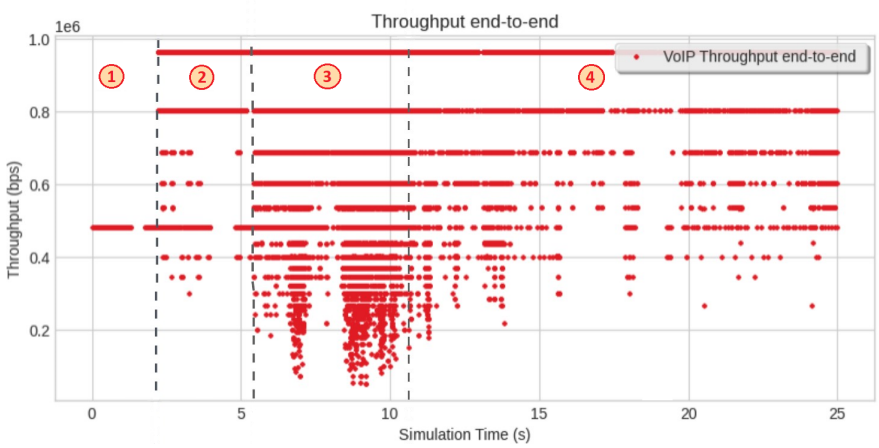

Figura 5.13: *Throughput end-to-end* en ueR[0] - Segunda Simulación.

Por otro lado en la gráfica 5.14 se visualiza el *throughput* en la red óptica, el cual tiene valores levemente mayores porque además de que el *delay* es menor,

también recibió muchos más paquetes del ue[10] hasta el ue[19] que no estaban contemplado en las estadísticas del ueR[0]. A su vez, no varía tanto como el *throughput end-to-end* dado que el *delay* en la fibra óptica es mucho menor que en las redes móviles y las frecuencias usadas no afectaron el resultado, los dos valores distintos que se visualizan de deben al la diferencia entre los tamaños de paquetes (300B y 600B).

Es interesante de ver que la línea con el valor inferior representa un único tráfico (paquetes de 300B enviados por ue[0]), por lo que se pueden apreciar cuando son los períodos de silencio característicos de VoIP. Mientras que en la línea superior que corresponde a los demás tráficos seguramente el período de silencio de un aplicación es interferido con el período de habla de otra y por ello no se ve una línea descontinuada.

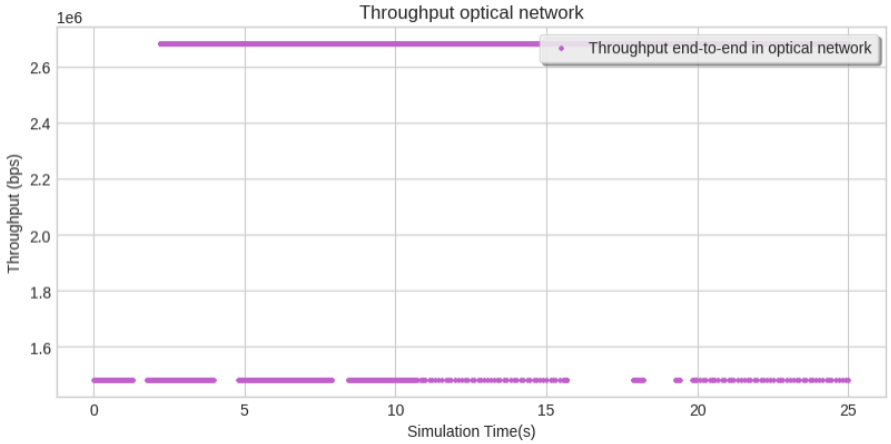

Figura 5.14: Throuhput optical network - Segunda Simulación.

A continuación se muestran algunos valores claves del *throughput end-to-end* y de la red óptica:

- Máximo *throughput* VoIP ueR[0]: 962603 bps.
- Mínimo *throughput* VoIP ueR[0]: 51067 bps.
- Promedio *throughput* ueR[0]: 824533 bps.
- Máximo *throughput* red óptica: 2682658 bps.
- Mínimo *throughput* red óptica: 1483258 bps.
- Promedio *throughput* red óptica: 2673274 bps.

5.4.3. Simulación 3

Esta es la última simulación y al igual que la anterior se uso la topología 4.16. En este caso la mayor diferencia va a ser que se tiene tráfico de los dos lados (se utilizan los dos flujos de redes ópticas).

La simulación consta de 20 ues en total. 10 de ellos pertenecen al módulo vector ue[] y los otros 10 al módulo vector ueR[]. Todos los ues mandan y reciben tráfico, lo que quiere decir que tienen una aplicación emisora y otra receptora. La única diferencia va a ser que los ues pertenecientes al módulo ue[] mandan tráfico CBR, mientras que los ues que pertenecen a ueR[] mandan tráfico VoIP.

En esta simulación puede resultar interesante ver el *delay* de ambos tipos de tráfico (VoIP y CBR) en iguales condiciones y ver si se presentan algunas diferencias entre los dos flujos de redes ópticas.

Se mencionan algunas características de la simulación en particular:

- La simulación se detiene a los 25s.
- Todos los ues pertenecientes a ue[] mandan tráfico con destino al ueR[] correspondiente a su nombre y viceversa.
- Todos los paquetes enviados son de 300B y se mandan con una separación de 0,02 s (tanto VoIP como CBR).
- Las características de las redes ópticas son iguales entre ellas y son las mismas que se mencionaron en la sección anterior. La única diferencia es que una dirige tráfico hacia la gnb y otra hacia la gnb2.

En esta simulación se le da prioridad al análisis del *delay* pero no se analizará el *throughput* dado que al no variar el tamaño de los paquetes ni la frecuencia la gráfica va a ser inversamente proporcional al *delay* como en la primera simulación.

Se estudian particularmente la pareja ue[0] y ueR[0] que se mandan mensajes mutuamente. Con respecto a la cantidad de paquetes enviados y recibidos se tienen los siguientes valores:

- Cantidad de paquetes CBR enviados por ue[0]: 1250.
- Cantidad de paquetes VoIP enviados por ueR[0]: 739.
- Cantidad de paquetes VoIP que llegaron a ue[0]: 738.
- Cantidad de paquetes CBR que llegaron a ueR[0]: 1249 .
- Cantidad de paquetes CBR que pasaron por la red óptica en dirección a gnb2 : 12490.
- Cantidad de paquetes VoIP que pasaron por la red óptica en dirección a gnb: 8570.

Como se puede ver en la gráfica 5.15 en iguales condiciones el *delay* del VoIP es levemente menor al del CBR, característica que se mantuvo en la primera simulación también. A pesar de ello, ambos retardos se mantienen casi constantes.

Si se compara nuevamente con la primera simulación, se puede observar que ambos retardos (tanto VoIP como CBR) son levemente mayores. Esto es interesante porque en realidad los flujos de red óptica están menos congestionados ya que en la primera simulación el tráfico era generado por 15 ues , mientras en cada red óptica de esta simulación el tráfico es generado por 10 ues (conservando el mismo tamaño de paquetes y frecuencia de envío). El aumento en el retardo se da porque en esta simulación el tráfico fluye bidireccionalmente, por lo que aunque a los distintos flujos de redes ópticas no les afecte, a las gnbs y upfs sí. Estos componentes deben manejar tanto aplicaciones *uplink* como *downlink*. Esto se puede confirmar observando los valores de *delay* de ambas redes ópticas más abajo, que son exactamente iguales entre sí y a su vez iguales a antiguas simulaciones.

Otro dato interesante es que se está observando el tráfico generado por un ue (tráfico CBR generado por ue[0] y tráfico VoIP generado por ueR[0]). Pero si se tomará otra pareja de ues que se manden mensajes mutuamente los valores se mantienen prácticamente iguales, variando únicamente el dibujo de la gráfica por los silencios que difieren en cada aplicación VoIP.

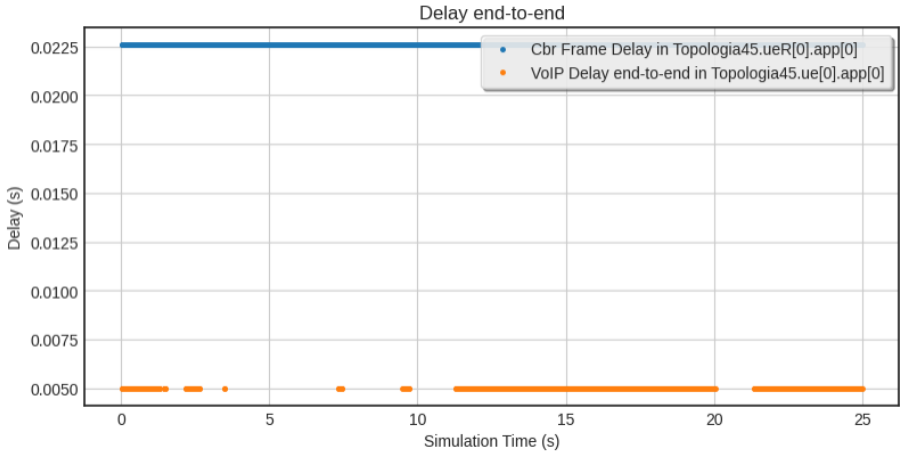

Figura 5.15: *Delay end-to-end* en ue[0] y ueR[0].

Se muestran a continuación algunos valores clave del *delay* end-to-end:

- *Delay* VoIP:0,004996s.
- *Delay* CBR: 0,022601s.
- Mínimo *delay* CBR:0,077601s.
- *Delay* red óptica en dirección a gnb2: 0,002001s.
- *Delay* red óptica en dirección a gnb: 0,002001s.

No se gráfica el *delay* en ninguna de las redes ópticas ya que es constante y similar al de la primera simulación.

En el siguiente capítulo se contará brevemente cuales fueron las distintas etapas que tuvo este proyecto de grado, que se hizo en cada etapa y un registro de esfuerzos. A su vez también incluye actividades de gestión que comprenden el proyecto CSIC y la organización que se tuvo en general.

Capítulo 6

Ingeniería de Software

Como ya se mencionó este proyecto de grado es parte del proyecto CSIC Convergencia entre redes 5G/6G y redes ópticas: un enfoque holístico. Por este motivo la organización fue significativamente superior debido a la magnitud del proyecto. A continuación se detallará la metodología que se uso así como las herramientas utilizadas para la organización y comunicación dentro del proyecto CSIC.

6.1. Gestión del proyecto

Como medio de comunicación primario se utilizó la herramienta de Mattermost con los principales integrantes de CSIC. Había un canal específico para todo lo que sea relacionado a este proyecto, así como otros que estaban relacionados a otros proyectos o temas. Se utilizó el mail como medio de comunicación secundario pero con mucha menos frecuencia y en general cuando se necesitaba notificar también a personas del proyecto CSIC que no pertenecían al canal de Mattermost.

También se realizaron reuniones por Zoom con cierta periodicidad dependiendo de la etapa del proyecto. A veces fueron semanales, otras bi-semanales y en otros casos una vez al mes. Las reuniones tenían distintas temáticas, por ejemplo reuniones de seguimientos de avances, reuniones de planificación de próximos pasos, seminarios de aprendizaje (reuniones donde se presentaba un tema o herramienta), reuniones donde se explicaban dudas o problemas, etc.

Para organización del proyecto CSIC se utilizó un Notion, en cual se tenía todos los *links* de información centralizados, entre ellos:

- *Link* al canal de Mattermost.
- *Link* a ownCloud, el cual contenía información de ambos dominios, propuesta, etc.
- *Link* al GitLab del proyecto [2].

- *Links* a distintas herramientas de simulación y sus respectivas documentaciones.

A su vez se tenía toda la información de las reuniones que se realizaron en el transcurso de este proyecto. Por ejemplo *links* de las reuniones, grabación de las mismas y archivos adjuntos en caso de ser utilizados.

En el ownCloud se manejaban distintas carpetas por ejemplo con archivos de la propuesta CSIC, bibliografía para el dominio de redes ópticas y 5G, archivos utilizados en los seminarios, cronogramas, etc.

6.2. Gestión del esfuerzo

Ahora se hará énfasis en las etapas y actividades que se tuvieron a lo largo de la duración del proyecto. Este proyecto de grado comenzó en el mes de marzo de 2023 y culminó en mayo de 2024, por lo que duró 15 meses. Por otro lado el proyecto CSIC empezó formalmente a mediados de abril y se prevé una duración de dos años aproximadamente. A continuación se describen todas las etapas con sus respectivas duraciones

En la primera etapa del proyecto se tenía como objetivo entender los dos dominios específicos (redes 5g y redes ópticas). Para ello se dedicó alrededor de 3 meses de estudio dado que eran temas nuevos, que no se dan en la currícula de la carrera ingeniería en computación. En esta etapa se realizaron reuniones semanales para consultar dudas a los tutores, expertos en los dominios.

A fines del mes de abril, se empezaron a realizar seminarios semanales de distintos temas donde participaron todos o la mayoría de los integrantes del proyecto CSIC. El objetivo de estos seminarios era que cada persona hable del tema donde era experto (ya que habían personas que sabían mucho uno de los dos dominios pero no tanto el otro por ejemplo). Estos seminarios fueron extremadamente enriquecedores para la adquisición de conocimientos de ambos dominios necesarios para poder realizar este proyecto.

A partir de mediados de mayo aproximadamente comenzó la segunda etapa del proyecto, que se trataba de investigar herramientas de simulación. Los seminarios siguieron en paralelo a esta etapa y pasarón a ser exposiciones de las herramientas de simulación estudiadas. De mí lado investigué especialmente OMNeT++ y GNPy, de los cuales solo presenté la herramienta GNPy en uno de los seminarios ya que otro participante presentó OMNeT++. La última herramienta de simulación que fue presentada fue ns-3, presentada por el otro proyecto de grado. Tanto la segunda etapa como los seminarios terminaron con el comienzo de junio.

Luego del estudio de las distintas herramientas de simulación, se optó en este proyecto por usar OMNeT++ para crear la infraestructura. La tercera etapa consistió en un estudio más a fondo de la herramienta OMNeT++ y particularmente de Simu5G. Aquí se desarrollaron las tres topologías que luegos sirvieron de base para la herramienta óptico-móvil. Esta etapa implicó un conocimiento profundo de los principales módulos de Simu5G, cómo interactuan, qué parámetros se pueden variar para las simulaciones, qué estadísticas se pueden sacar y

cómo crear nuevas estadísticas. Esta etapa duró aproximadamente tres meses, empezando en julio y terminando en octubre.

Ahora llega la cuarta etapa, la más larga y tediosa, la implementación de la herramienta óptico-móvil. Como se menciona anteriormente en 4 el camino para llegar a la herramienta fue bastante empedrado, pues por más que se haya dedicado un buen tiempo al estudio de Simu5G, para lograr la convergencia fue necesario una comprensión mucho más profunda. Esta etapa comenzó en octubre y termino en marzo.

Finalmente en la quinta y última etapa se ejecutaron muchas simulaciones variando determinados parámetros para crear escenarios interesantes y capturar estádisticas. Esta etapa consistió en la experimentación y evaluación de la herramienta para llegar a las conclusiones. Incluyó la implementacion de algunas estadísticas que no estaban.

Capítulo 7

Conclusiones y Trabajo Futuro

7.1. Conclusiones

Este proyecto se enmarca en una línea de investigación que surge en respuesta a las demandas crecientes de la sociedad por una conectividad mejorada, velocidad superior y menor latencia, tal como se evidencia en los casos de uso del 5G. La creación de una herramienta óptico-móvil de código abierto, capaz de realizar diversas simulaciones, representa el primer paso hacia la mejora potencial de las comunicaciones.

Los resultados obtenidos de este proyecto contribuyen al avance de la investigación y desarrollo tecnológico en redes móviles y ópticas. Se ha logrado desarrollar una herramienta de simulación de redes óptico-móviles que abre nuevas posibilidades para su estudio exhaustivo, proporcionando estadísticas precisas sobre las redes simuladas. Los principales resultados de este proyecto de grado se resumen en el artículo científico [23] que fue aceptado para ser presentado en la *International Conference on Transparent Optical Networks 2024*.

Es crucial destacar que la herramienta desarrollada tiene aplicaciones prácticas en la comprensión del comportamiento de las redes bajo diferentes sistemas de asignación de recursos y en la emulación de diversas arquitecturas de red. Estos fueron los principales motivos que impulsaron el desarrollo de la herramienta. Su utilidad se extiende a la evaluación de algoritmos de asignación de recursos en un enfoque multitecnológico, lo que puede mejorar el rendimiento de las redes óptico-móviles en escenarios reales.

A partir de las simulaciones realizadas, se puede concluir que se obtuvieron resultados positivos al incorporar una red óptica en la infraestructura de una red móvil 5G. El tráfico de mensajes recorrió de un extremo a otro sin contratiempos y de forma imperceptible para la red móvil. Sin duda alguna las simulaciones se podrían complejizar para lograr resultados más fieles a la realidad. Es fundamental continuar invirtiendo tiempo en esta herramienta para comprender

mejor cómo sería una topología real basada en la convergencia de redes ópticas y móviles.

Como se ha mencionado el proyecto forma parte de un esfuerzo más amplio de investigación y desarrollo que involucra a diversos actores, desde estudiantes hasta expertos en el campo, incluyendo también instituciones interesadas como Antel y DINATEL, los cuales han manifestado su apoyo e interés. Esta colaboración interdisciplinaria no solo enriqueció el proyecto, sino que también aumenta su potencial impacto en la sociedad y el futuro de las comunicaciones.

Se ha enfatizado la importancia de la documentación precisa y clara del trabajo realizado, brindando los conocimientos adquiridos a quienes siguen por este camino dentro del proyecto CSIC. A su vez, cumpliendo el objetivo de facilitar la comprensión y el uso de la herramienta por parte de una amplia audiencia. La disponibilidad de la herramienta en un repositorio público promueve su accesibilidad y fomenta la colaboración, permitiendo que más personas se beneficien del conocimiento generado y contribuyan a su mejora continua.

En resumen, este proyecto ha logrado alcanzar sus objetivos específicos y sienta las bases para futuras investigaciones y desarrollos en el campo de las redes óptico-móviles, con el potencial de impactar positivamente en diversos aspectos de la sociedad moderna. A continuación se presentan los trabajos futuros hacia la herramienta creada.

7.2. Trabajo futuro

Si bien se logró una gran conformidad con la creación de la herramienta y los resultados obtenidos de las simulaciones, quedaron muchos temas a investigar y modificaciones para mejorar la herramienta de simulación. En esta sección se detallarán todos los problemas encontrados que podrían tener una mejor solución pero que no se pudieron implementar en el tiempo abarcado.

7.2.1. Ruteo estático

En el proceso de la creación de la herramienta se conocieron más los módulos de Simu5G y INET. En particular faltó estudio al configurador de INET, para detectar como funcionaba a detalle en la asignación de rutas estáticas a las tablas de ruteo de los módulos. Ya que el mismo añadía rutas evitando que los mensajes se propaguen a través del nodo (y por ende la fibra óptica).

Para solucionar este problema se insertaron rutas manualmente en las tablas de ruteo, pero estas rutas debían ser lo más específicas posibles ya que el configurador creaba rutas en sentido contrario para las mismas entradas. Esta solución no es escalable y actualmente el simulador funciona para un máximo de 100 ues por gnb, para añadir más ues se precisa añadir más rutas manualmente.

Una mejor solución sería un estudio del configurador de INET a fondo para entender por qué se comporta de esta manera y evaluar la posibilidad de remediar el problema. Otra mejor solución posible sería implementar un nuevo

sistema de asignación de rutas estáticas que se ajuste a las necesidades requeridas para no depender del funcionamiento del configurador de INET. También esta la posibilidad de una solución híbrida que se base en modificar el funcionamiento del configurador de INET para que se ajuste a las necesidades existentes.

7.2.2. Fragmentación en las simulaciones

En la última etapa del proyecto donde se realizaron distintas simulaciones para obtener distintas estadísticas, se observó que la herramienta no funciona tan bien cuando se mandan paquetes de tamaños grandes (Mayores a 2MB). Estos problemas se daban porque el paquete era fragmentado y parecía que cuando llegaba al destino no era posible la reconstrucción del paquete original.

Por otro lado los canales se llenaban y la simulación se detenía a menos que se modificará el código para que no se manden mensajes hasta que los canales se vacién. Teniendo en cuenta que esto provocaba un aumento inmenso en el *delay* de todas las simulaciones y que no era una buena solución al problema, se optó por evitar mandar paquetes de tamaño tan grandes para no recurrir a la fragmentación.

De esta manera queda pendiente una investigación más extensa de como se maneja la fragmentación en INET y Simu5G para entender como solucionar el problema de raiz.

7.2.3. Comunicación entre las gnbs

Como se ha aclarado en capítulos anteriores, para que la conexión *end-to-end* entre dos gnbs se da únicamente si ambas están conectadas al mismo upf. Esto sucede ya que el upf es quien enruta los paquetes de una gnb a otra y cada gnb tiene asociado un upf al que le envía los paquetes que no correspondan a los ues de su dominio.

Sin embargo si se quiere hacer una topología con más de dos gnbs que se comuniquen entre sí a través de una red óptica, se tiene el problema de que absolutamente todos los paquetes pasarán por el upf, congestionando ese punto en la red y provocando un cuello de botella. Por lo que no es escalable al aumentar el número de gnbs interconectadas.

Queda como pendiente una investigación más profunda de ambos módulos (gnb y upf) para poder encontrar una solución que permita la escalabilidad. Quizás sea necesario modificar el código de Simu5G para permitir que las gnbs puedan asociarse a más de un upf y así no sobrecargar uno solo.

7.2.4. Recopilación de estadísticas

Este proyecto se ha centrado principalmente en visualizar el *delay* y *throughput end-to-end* en distintas simulaciones, dandole más prioridad al *delay* e intentando mostrar resultados interesantes. Sin embargo existen muchas estadísticas que se podrían capturar y que sin duda enriquecerían la herramienta de simulación óptico-móvil.

Entre las estadísticas que se podrían obtener se encuentran:

- *Frame Loss*: Perdida de trama, refiere a cuando una trama de datos es perdida o descartada durante su transmisión a través de la red.
- *Tail Drop Loss*: Pérdida por descarte de cola, refiere a la pérdida que se da por el módulo receptor recibe más datos de los que puede procesar y transmitir el *buffer*.
- *Playout Delay:* Retraso de reproducción del módulo receptor.
- *MOS signal* en VoIP (*Mean Opinion Score*): Mide la calidad de la experiencia de usuario en las comunicaciones VoIP.
- *Jitter*: Variación en el retardo de la llegada de los paquetes de datos en una red
- % Total de perdida de paquetes.
- Cantidad de bytes recibidos.
- OSNR (Dentro de la red óptica).

7.2.5. Topologías y simulaciones

Si bien se exploraron distintas configuraciones y distintas topologías, quedaron aún muchos parámetros para explorar y algunas topologías de las que no se realizaron muchas pruebas.

Dentro de las posibles topologías para hacer simulaciones falto realizar pruebas con:

- Topologías con más de dos gnbs conectadas entre sí. Por ejemplo tres gnbs conectadas a través de tres flujos de redes ópticas.
- Topologías con 4 o más gnbs pero que solo estén conectadas de a parejas. Podría ser un análisis interesante visualizar si hay diferencias entre mismo tráfico en las distintas parejas.
- Topologías con distintas cantidad de ues por cada gnb.
- Topologías que deriven de NSA y puedan incluir comunicación de una gnb con un enb a través de una red óptica.

Con respecto a las posibles configuraciones que no se exploraron y que quedan para un futuro se tienen:

- Variar los distintos métodos de propagación por aire.
- Variar los algoritmos de *scheduling* de las gnbs.
- Configurar uno o más background cell para que generen interferencias en las simulaciones.
- Utilizar otros tipos de tráfico distinto a VoIP y CBR (por ejemplo Burst).
- Configurar numerología y espectro.
- Poner los ues en movimiento e incluso permitir que sea posible que cambien de gnb en tiempo de simulación.
- Variar las configuraciones de la red óptica, por ejemplo el largo de la fibra óptica, la sensibilidad del *receiver*, etc.

7.2.6. Explotación de la herramienta

Hasta ahora solo se ha hablado de posibles mejoras de la herramienta creada, sin embargo es importante mencionar también los siguientes pasos para sacarle la mayor utilidad a la herramienta. Por temas de tiempo no se pudo avanzar más en el trabajo futuro que se presenta a continuación, pero se deja documentado para que se puede continuar desarrollando sobre el mismo.

Es de gran utilidad ejecutar muchas simulaciones, variando las topologías y parámetros para guardar sus estadísticas y con ellas crear un gran dataset. Dicho dataset contendría información de la topología en la que se simuló, las características de la simulación (ej. cuanto tráfico, tipo de tráfico, etc.) y a su vez las estadísticas resultantes de esa simulación. La generación del dataset permitiría entrenar algoritmos de inteligencia artificial para que predigan si se van a cumplir ciertos requerimientos de la red dado las características de la topología y la demanda de tráfico. Para el entrenamiento del algortimo se utilizarían técnicas de machine learning, como podría ser (siguiendo una metodología de aprendizaje supervisado) dividiendo el dataset en dos, permitiendo que una parte se utilice para entrenamiento y otra para testeo del algoritmo.

Simplemente para un mayor entendimiento se explica un ejemplo de un posible caso de uso. Un requerimiento podría ser que el *delay* no supere los 0,02s. Para ello se entrena un algoritmo para predecir el *delay* basándose en el dataset creado. El algoritmo recibiría como dato de entrada una topología y las demandas de tráfico. La salida del algoritmo sería el *delay* previsto para las características de la topología y la demanda, y en base a eso se podría determinar si cumple el requerimiento inicial o no.

Existen muchas posibilidades de algoritmos que se pueden crear, podrían ser más simples y dado un valor predecir si se va a cumplir o no (respuesta booleana), o un poco más complejos y predecir el valor que va a tener si se simula una topología con las características dadas como entrada. A su vez se podrían tener más de un requerimiento a cumplir (por ejemplo baja latencia y a su vez gran volumen de datos), por lo que se podrían hacer más de un algoritmo dependiendo del requerimiento que se quiera determinar si se cumple o no.

Bibliografía

[1] Omnet++: Discrete event simulator. https://omnetpp.org/. Accessed: 2024-03-20.

[2] Convergencia redes ópticas y 5g. https://gitlab.fing.edu.uy/mobile-optical-networks/convergencia-redes-opticas-y-5g. Accessed: 2024-06-17.

[3] Omnetpp: cmessage. https://doc.omnetpp.org/omnetpp/api/classomnetpp_1_1cMessage.html, 2022. Accessed: 2024-03-20.

[4] Omnetpp: csimplemodule. https://doc.omnetpp.org/omnetpp/api/classomnetpp_1_1cSimpleModule.html#a5584b421899bb435d1ef83def8c61453, 2022. Accessed: 2024-03-20.

[5] Inet. https://inet.omnetpp.org/, 2022. Accessed: 2024-03-20.

[6] E. Ahlman, S. Parkvall, and J. Skold. *5G NR: The Next Generation Wireless Access Technology*. Elsevier Science, 2018. ISBN 9780128143230.

[7] Jaeuk Baek, Jimin Bae, Yongjae Kim, Jinteak Lim, Eunhye Park, Jaehyeok Lee, Gyujae Lee, Sang Ik Han, Chol Chu, and Youngnam Han. 5g k-simulator of flexible, open, modular (fom) structure and web-based 5g k-simplatform. In *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–4, 2019. doi: 10.1109/CCNC.2019.8651775.

[8] Nicola Baldo, Manuel Requena-Esteso, Marco Miozzo, and Raymond Kwan. An open source model for the simulation of lte handover scenarios and algorithms in ns-3. In *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, MSWiM '13, page 289–298, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450323536. doi: 10.1145/2507924.2507940. URL https://doi.org/10.1145/2507924.2507940.

[9] Ingmar Baumgart, Bernhard Heep, and Stephan Krause. Oversim: A flexible overlay network simulation framework. In *2007 IEEE Global Internet Symposium*, pages 79–84, 2007. doi: 10.1109/GI.2007.4301435.

[10] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[11] Lucas R. Costa and André C. Drummond. ONS – optical network simulator. //ons-simulator.com, 2021.

[12] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[13] Felipe Falcón, Gonzalo España, and Danilo Bórquez-Paredes. Flex net sim: A lightly manual, 2021.

[14] Patricia Hernández and Gonzalo Carro. *Uso del Espectro Radioeléctrico en Uruguay y Oportunidades para el Uso de Radio Cognitiva*. Tesis de grado, Universidad de la Republica, Facultad de Ingeniería, Noviembre 2016. URL https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/20119.

[15] Lucas Inglés. Aprendizaje profundo para la asignación de recursos en redes 5g. Tesis de maestría, Universidad de la Republica, Facultad de Ingeniería, Marzo 2023. URL https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/36721.

[16] IpLook. User plane function(upf). https://es.iplook.com/productos/5gc/upf.html. Accessed: 2024-05-10.

[17] ITU. Recomendación uit-r m.2083-0. concepción de las imt – marco y objetivos generales del futuro desarrollo de las imt para 2020 y en adelante. https://www.itu.int/rec/R-REC-M.2083/es, 2015. Accessed: 2024-03-20.

[18] Florian Kaltenberger, Guy de Souza, Raymond Knopp, and Hongzhi Wang. The openairinterface 5g new radio implementation: Current status and roadmap. In *International ITG Workshop on Smart Antennas*, pages 1–5, 2019. URL https://api.semanticscholar.org/CorpusID:160022106.

[19] Florian Kaltenberger, Aloizio P. Silva, Abhimanyu Gosain, Luhan Wang, and Tien-Thinh Nguyen. Openairinterface: Democratizing innovation in the 5g era. *Computer Networks*, 176:107284, 2020. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2020.107284. URL https://www.sciencedirect.com/science/article/pii/S1389128619314410.

[20] Yongjae Kim, Jimin Bae, Jinteak Lim, Eunhye Park, Jaeuk Baek, Sang Ik Han, Chol Chu, and Youngnam Han. 5g k-simulator: 5g system simulator for performance evaluation. In *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–2, 2018. doi: 10.1109/DySPAN.2018.8610404.

[21] Jan Kundrát, Gert Grammeland, Alessio Ferrari, Mark Filer, Esther Le Rouzic, , Bruno Correia, Karthikeyan Balasubramanian, Yawei Yin, Gabriele Galimberti, and Vittorio Curri. Gnpy: an open source planning tool

for open optical networks. pages 1–6, 05 2020. doi: 10.23919/ONDM48393.2020.9133027.

[22] Víctor López and Luis Velasco. *Elastic Optical: Networks Architectures, Technologies, and Control*. Springer, 2016. ISBN 978-3-319-30173-0.

[23] M. Rodriguez L. Inglés A. Castro M. Pirri, C. Rattaro. Integrating optical and mobile networks: A comprehensive end-to-end simulation platform. *24th ICTON 2024 (International Conference on Transparent Optical Networks)*, 2024.

[24] Martin Klaus Müller, Fjolla Ademaj, Thomas Dittrich, Agnes Fastenbauer, Blanca Ramos Elbal, Armand Nabavi, Lukas Nagel, Stefan Schwarz, and Markus Rupp. Flexible multi-node simulation of cellular mobile communications: the vienna 5g system level simulator. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):17, 2018. URL https://api.semanticscholar.org/CorpusID:256237007.

[25] Giovanni Nardini, Dario Sabella, Giovanni Stea, Purvi Thakkar, and Antonio Virdis. Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks. *IEEE Access*, 8:181176–181191, 2020. doi: 10.1109/ACCESS.2020.3028550.

[26] Giovanni Nardini, Giovanni Stea, Antonio Virdis, and Dario Sabella. Simu5g: A system-level simulator for 5g networks. In *International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, 2020. URL https://api.semanticscholar.org/CorpusID:220847264.

[27] Natale Patriciello, Sandra Lagen, Biljana Bojovic, and Lorenza Giupponi. An e2e simulator for 5g nr networks. *Simulation Modelling Practice and Theory*, 96:101933, 2019. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2019.101933. URL https://www.sciencedirect.com/science/article/pii/S1569190X19300589.

[28] Wilder Peña, Walter Piastri, and Pablo Vázquez. *Implementación de una Maqueta de Pruebas y Desarrollo de una Red 5G Stand Alone Completa.* Tesis de grado, Universidad de la Republica, Facultad de Ingeniería, Noviembre 2023. URL https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/42953.

[29] C. T. Politi, C. Matrakidis, and A. Stavdas. *A Tutorial on Physical-Layer Impairments in Optical Networks*, pages 5–29. Springer US, Boston, MA, 2013. ISBN 978-1-4614-5671-1. doi: 10.1007/978-1-4614-5671-1_2. URL https://doi.org/10.1007/978-1-4614-5671-1_2.

[30] Stefan Pratschner, Bashar Tahir, Ljiljana Marijanović, Mariam Mussbah, Kiril Kirev, Ronald Nissel, Stefan Schwarz, and Markus Rupp. Versatile

mobile communications simulation: the vienna 5g link level simulator. *EURASIP Journal on Wireless Communications and Networking*, 2018(1):226, 2018. URL https://api.semanticscholar.org/CorpusID:52832376.

[31] Rattaro, Pereyra, and Belzarena. Py5chesim: a 5g multi-slice cell capacity simulator. pages 2–7, 2021.

[32] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, 2011. doi: 10.1109/TMC.2010.133.

[33] Antonio Virdis, Giovanni Stea, and Giovanni Nardini. Simulte - a modular system-level simulator for lte/lte-a networks based on omnet++. In *2014 4th International Conference On Simulation And Modeling Methodologies, Technologies And Applications (SIMULTECH)*, pages 59–70, 2014. doi: 10.5220/0005040000590070.

[34] Syed Muhammad Asad Zaidi, Marvin Manalastas, Hasan Farooq, and Ali Shariq Imran. Syntheticnet: A 3gpp compliant simulator for ai enabled 5g and beyond. *IEEE Access*, 8:82938–82950, 2020. URL https://api.semanticscholar.org/CorpusID:218650643.

Anexo A

¿Cómo funciona el acceso al medio?

Los sistemas analógicos no pueden transmitir muchos datos por segundo, si se quisiera una gran transmisión de datos se necesitarían sistemas digitales. En los sistemas digitales se busca transmitir bits por la magnitud física, para ello se realiza un mapeo de bit a una amplitud (representado por un símbolo). La cantidad se símbolos que se necesita es 2^N, siendo N la cantidad de bits que se quiere transmitir. Los símbolos se representan como puntos en constelaciones. Una representación gráfica en A.1.

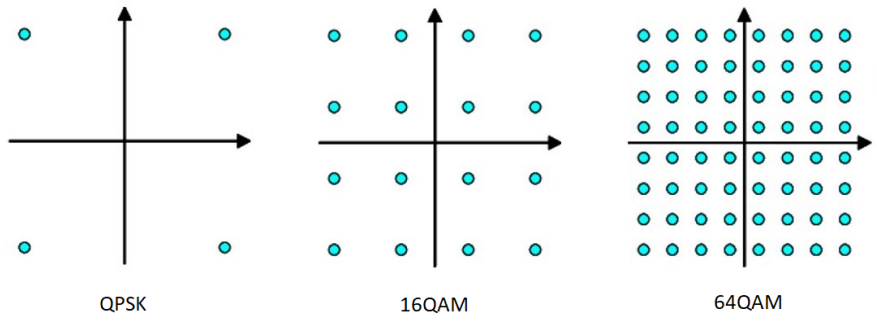

Figura A.1: Constelaciones.

LTE transmite cadena de 4 bits, o sea necesita 16 símbolos. Los 16 símbolos se representan como puntos en una constelación 16QAM. El código Gray indica que punto en la constelación corresponde a cada cadena de 4 bits.

Cada símbolo se forma como combinación lineal de dos señales analógicas que se representan en el eje horizontal y vertical. El eje horizontal es :

$\phi_1(t) = cos(2\pi f_c t)\Pi(\frac{t-T/2}{T})$ mientras que el eje vertical es

$\phi_2(t) = sen(2\pi f_c t)\Pi(\frac{t-T/2}{T})$ donde f_c es la frecuencia de la portadora y T

es el cuadro de tiempo en el que se va a transmitir una señal (símbolo). Luego de este tiempo se empieza a transmitir el siguiente fragmento (símbolo).

Como $\phi_1(t)$ y $\phi_2(t)$ son senos y cosenos, su combinación lineal también lo es y el resultado es un pulso que se transmite durante el cuadro de tiempo T. Los distintos símbolos tienen distinta amplitud y al introducir ruido a veces el receptor de estos pulsos puede confundir los símbolos. Mientras la constelación tenga más puntos más aumenta la tasa de error, pues los puntos están más próximos y con el ruido se puede variar más el pulso que llega al receptor.

Los errores se puede dar por diversas razones:

- Pérdidas de espacio libre: Ocurre porque la energía de la señal se dispersa en un área cada vez mayor a medida que se aleja de la fuente.
- Ensombramiento: Se produce cuando existen obstáculos físicos entre el transmisor y el receptor, lo cual hace que parte de la señal se absorba por el obstáculo, otra se refleje y otra se difracta.
- Desvanecimiento por multicamino: Ocurre cuando una señal se propaga por múltiples rutas entre el transmisor y el receptor.
- Desvanecimiento por efecto Dopler: Se da cuando el transmisor o receptor está en movimiento y se producen cambios en la frecuencia debido al movimiento.

Anexo B

Rangos de frecuencia utilizados

Frecuencias de banda baja (sub 2Ghz): Se utilizan en 4G. Algunas bandas importantes son: 700MHZ 700 MHz 3.3-3.8 GHz (*C-band*) 3.4-3.7 GHz (n77, n78)

Esto las hace ideales para la cobertura en áreas extensas urbanas y suburbanas. Las más altas son 600 y 700 MHz, bandas n71 y n28. Máximo de 20 MHz channel bandwidth.

Frecuencias medias (3-6 GHz): Capacidad de cobertura. Así como también una tasa de datos más alta debido a un mayor bandwidth. Los rangos más altos interesados globalmente son 3300-4200 MHz, lo que 3GPP llamó n77 y n78. Channel bandwidth hasta 100 MHz. Hasta 200 MHz por operador podría ser asignado. En Europa hay variantes de este rango, y Japón.

Frecuencias altas (sobre 24 GHz) En particular interesa el rango 24.25 - 29.5 GHz, llamadas n257 y n258. Channel bandwidth hasta 400 MHz, con incluso mayor bandwith posible con la agregación de portadoras.

NR se desplegará en las bandas existentes IMT y las futuras bandas que serán identificadas por WRC o por cuerpos regionales.

Anexo C

Configuración de las direcciones IP y rutas

Como se mencionó anteriormente, se tuvo que configurar a manualmente las direcciones IPs y rutas para que los mensajes se mandaran por el camino correcto. Aquí se explicará de manera más detallada como configurar el módulo *Ipv4NetworkConfigurator*, encargado de asignar direcciones IP y rutas estáticas.

El configurador tiene un parámetro llamando *config*, que consiste en un xml con todas las rutas, direcciones IP y otras configuraciones. Este xml puede ser escrito en el mismo archivo .ini, pero si es de un largo muy extenso conviene hacerlo en un xml aparte de la siguiente manera:

```
*.configurator.config = xmldoc("./config.xml")
```

El archivo xml debe contener :

- Distintas subredes para cada gnb y celulares próximos. Por ejemplo: gnb con subred 10.0.5.x y gnb2 con subred 10.0.6.x. Ambos con mascara 255.255.255.0.
- Opcionalmente la dirección IP específica de los celulares teniendo en cuenta que gnb tienen más próxima, de otra manera asigna a todos los ues una misma subred.
- Una ruta con la interfaz de salida correcta en la tabla de ruteo del upf para cada:
  - Subred de cada gnb (ej 10.0.6.x).
  - Cada gnb específica (ej 10.0.6.8).
  - Opcionalmenta cada ue específico (ej 10.0.6.9).

El upf manda los paquetes a la gnb asociada al ue destino, por lo que asignar rutas específicas con direcciones IP de los ues no es necesario. Sin embargo es necesario añadir una ruta específica por cada interfaz de cada gnb. Osea una dirección IP destino de las interfaces

- *pppIf.*

- *pppMEHostIf*.
- *cellular*.

Si se deja una ruta con destino con mascara de red, el configurador crea una ruta más específica que lo mande por otro lado.

Para configurar las direcciones IP se debe crear una etiqueta *interface*. En el atributo *hosts* se coloca el nombre del módulo y en el atributo *address* la dirección IP deseada. Si es una dirección IP específica se debe colocar en el atributo *names* el nombre de la interfaz. Por otro lado si se quiere asignar un rango de direcciones IP al módulo para todas sus interfaces en vez de colocar el atributo *names* se coloca el atributo *netmask* con la mascara correspondiente. Es importante aclarar que no se permite poner distinto rango de direcciones IP a distintos objetos de son parte del mismo módulo vector. Por ejemplo ue[0] y ue[1] deben permanecer a la misma subred.

Para configurar una ruta estática se debe crear una etiqueta *route*. La misma debe tener el atributo *hosts* que indica el nombre del módulo donde se va a añadir la nueva ruta a su tabla de ruteo. En el atributo *destination* y *netmask* se indica la dirección IP destino y la mascara a aplicar del paquete que viene a ser enrutado. En el atributo *gateway* se indica la dirección IP de la interfaz de salida donde sebe ser enviado el paquete y a su vez en el atributo *interface* se indica el nombre de la interfaz correspondiente.

El xml *config* de esta topología se ve de la siguiente manera:

```
<config>
    <interface hosts="gnb" address="10.0.5.x"
        ↪ netmask="255.255.255.x"/>

    <interface hosts="gnb2" address="10.0.6.x"
        ↪ netmask="255.255.255.x"/>

    <interface hosts="**" address="10.x.x.x" netmask="255.x.x.x"/>

    <route hosts="upf" destination="10.0.6.0"
        ↪ netmask="255.255.255.0" gateway="10.0.0.2"
        ↪ interface="ppp0" metric="0"/>

    <route hosts="upf" destination="10.0.6.9"
        ↪ netmask="255.255.255.255" gateway="10.0.0.2"
        ↪ interface="ppp0" metric="0"/>

    <route hosts="upf" destination="10.0.6.8"
        ↪ netmask="255.255.255.255" gateway="10.0.0.2"
        ↪ interface="ppp0" metric="0"/>

    <route hosts="upf" destination="10.0.6.1"
        ↪ netmask="255.255.255.255" gateway="10.0.0.2"
        ↪ interface="ppp0" metric="0"/>
```

```
    <route hosts="upf" destination="10.0.6.5"
        ↪ netmask="255.255.255.255" gateway="10.0.0.2"
        ↪ interface="ppp0" metric="0"/>

    <route hosts="upf" destination="10.0.5.0"
        ↪ netmask="255.255.255.0" gateway="10.0.0.13"
        ↪ interface="ppp3" metric="0"/>

    <route hosts="upf" destination="10.0.5.1"
        ↪ netmask="255.255.255.255" gateway="10.0.0.13"
        ↪ interface="ppp3" metric="0"/>

    ...
</config>
```

En el caso de la asignación de direcciones IP a la gnbs se realiza a través de un rango y utilizando una mascara. Esto sucede porque su interfaz *cellular* debe tener suficientes IP para conectar con todas las interfaces *cellular* de los respectivos ues y por ende a menos que se escriba una IP específica por cada ue (lo cual es engorroso y no es preciso dado que la cantidad de ues varia), esta es la manera más sencilla.

Por otro lado la tercera línea del xml indica que asigne todas las direcciones IP de los elementos de la red en la subred 10.x.x.x. Si no fuera necesario la configuración manual de rutas y direcciones IP, esta sería la única línea que debería estar en el xml.

Es importante aclarar que como la cantidad de ue son variables y la asignación de IPs depende de ello, hasta el momento de la simulación no se sabe con exactitud cual es la IP específica que tiene cada gnb. Este xml funciona hasta para incrementar el número de celulares en 100 por gnb (200 en total). Sin embargo si se desean añadir más celulares es posible que se deban añadir más rutas.

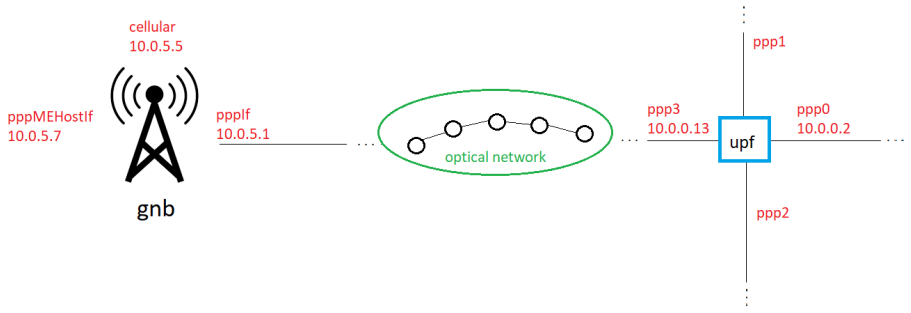

Figura C.1: Direcciones IPs de los módulos.

Tabla de ruteo upf		
subred	**gateway**	**interface**
10.0.6.0/24	10.0.0.2	ppp0
10.0.6.1/32	10.0.0.2	ppp0
10.0.6.9/32	10.0.0.2	ppp0
10.0.6.8/32	10.0.0.2	ppp0
10.0.6.5/32	10.0.0.2	ppp0
10.0.5.0/24	10.0.0.13	ppp3
10.0.5.1/32	10.0.0.13	ppp3

Figura C.2: Tabla de ruteo del upf según archivo xml.

Anexo D

Nomenclatura

Para que la herramienta funcione de manera adecuada sigue una nomenclatura específica La misma cuenta con las siguientes sugerencias y restricciones:

- El nodo inicial que es el encargado de mandar mensajes al *transmitter* debe tener de nombre "nodeX", siendo X cualquier combinación de caracteres que se desee para identificar a ese flujo, incluso puede ser vacío.
- El *OpticalReceiver* que va a recibir el mensaje enviado por dicho nodo luego de cruzar toda la red óptica, debe llamarse como "receiverX", siendo X la misma combinación de caracteres que se utilizó en el nombre del nodo.
- Se sugiere fuertemente que los nombres de los demás módulos involucrados en el mismo flujo empiecen con su nombre correspondiente y terminen en la misma combinación de caracteres "X".
- En caso de existir más de un flujo, los módulos de un distinto flujo deberán terminar en "Y", siendo "Y" una combinación de caracteres distinta de "X". De esta manera se le facilita al usuario le visualizacion de las distintas subredes conectadas dentro de una topología más grande.

La justificación de las primeras restricciones se de porque el nodo cuando crea el mensaje *Container* debe asignarle el identificador del receptor correspondiente y para ello se basa en el nombre de él mismo.

A continuación en la imagen D.1 se presentará un ejemplo de dos flujos con una correcta nomenclatura. En el primer flujo "X" es vacio y en el segundo "X" es el string "A".

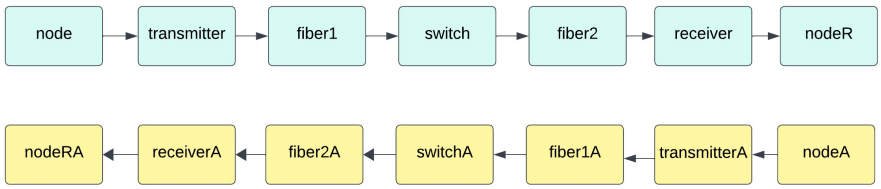

Figura D.1: Flujos con correcta nomenclatura.

Anexo E

Estadísticas en VoIP

Como se menciona anteriormente, fue necesario la creación de nuevas estadísticas en la aplicación VoIP (implementada en Simu5G) para capturar el correcto *delay* y *throughput*. Al ser modificado el código original del *framework*, es necesario dejar constancia de cuales fueron las modificaciones específicas para poder recrear estas estadísticas.

En este caso como todas las estadísticas fueron tomadas en el receptor de la aplicación, solo se modifico el VoIPReceiver. Sin embargo el mismo cuenta con tres archivos (.h, .cc y .ned) de los cuales se muestran los cambios específicos a continuación.

E.1. Cambios en VoIPReceiver.h

Se agrega dos señales a emitir en la definición de la clase como atributos. Estas corresponden respectivamente al *delay end-to-end* y al *throughput end-to-end*.

En el código inferior se visualiza como sería la definición de la clase, haciendo énfasis únicamente en los nuevos atributos de la clase que van a corresponder con las señales a emitir.

```
class VoIPReceiver : public omnetpp::cSimpleModule
{
...
    omnetpp::simsignal_t voIPDelayEndToEnd;
    omnetpp::simsignal_t voIPThroughputEndToEnd;

...
}
```

E.2. Cambios en VoIPReceiver.cc

En este archivo se define la lógica de la clase y por ende cómo y cuándo se emiten estas nuevas señales. Para ello se modificó la función *initialize()*, en donde se inicializan todos los atributos de la clase. En esta función se modificó el código para indicar que el valor de los nuevos atributos corresponde a la señal que será emitida.

```
void VoIPReceiver::initialize(int stage)
{
...
    voIPDelayEndToEnd= registerSignal("voIPDelayEndToEnd");
    voIPThroughputEndToEnd =
        ↪ registerSignal("voIPThroughputEndToEnd");
...
}
```

A continuación se debe emitir ambas señales de *delay* y *throughput* en el momento que sea adecuado. Como son estadísticas *end-to-end* lo correcto sería enviar una señal de *delay* y una de *throughput* cada vez que llegue un nuevo mensaje a la aplicación receptora.

Por eso mismo, se modificó la función *habndleMessage()* para que emita estas dos señales. Para el cálculo de ambas señales se utilizan los datos del paquete que acaba de ser recibido, específicamente el tiempo en el que se creó el mismo y su tamaño. Luego de que se calculen el *delay* y el *throughput*, se emiten cada una por su respectiva señal. A continuación se aprecia el fragmento de código modificado.

```
void VoIPReceiver::handleMessage(cMessage *msg)
{
    ...

    Packet* pPacket = check_and_cast<Packet*>(msg);

    ...

    double delay = (simTime() - pPacket->getCreationTime()).dbl();

    double throughput =
        ↪ static_cast<double>(pPacket->getBitLength()/delay);


    emit(voIPDelayEndToEnd, delay );

    emit(voIPThroughputEndToEnd, throughput );
```

```
    ...
}
```

E.3. Cambios en VoIPReceiver.ned

Finalmente en el archivo .ned se indican las nuevas señales a capturar y que valores que se capturaron de las mismas se van a guardar para la generación de estadísticas. En este archivo también se especifican detalles como el tipo de datos que se espera de la señal recibida, la unidad de la estadística, el título de la estadística, entre otros.

A su vez dentro de los valores que fueron capturados interesan :

- Máximo, mínimo y promedio como valores escalares, para tener algunos valores clave de la simulación.
- Vector y histograma como valores vectoriales, los cuales permiten graficar todo el contenido que nos llega.

A continuación se muestra el fragmento de código añadido.

```
simple VoIPReceiver like IApp
{
...

        @signal[voIPDelayEndToEnd] (type="double");
        @statistic[voIPDelayEndToEnd](title="VoIP Delay
            ↪ end-to-end"; unit="s"; source="voIPDelayEndToEnd";
            ↪ interpolationmode=none;
            ↪ record=max,min,mean,vector,timeavg, histogram);

        @signal[voIPThroughputEndToEnd](type="double");
        @statistic[voIPThroughputEndToEnd](title="VoIP Throughput
            ↪ end-to-end "; unit="bps";
            ↪ source="voIPThroughputEndToEnd";
            ↪ interpolationmode=none;
            ↪ record=max,min,mean,vector,timeavg, histogram);
...
}
```