ETS
Ingeniería de
Telecomunicación

Universidad
Politécnica
de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNCACIÓN

Grado en Ingeniería Telemática

Simulación realista de comunicaciones IoT en entornos urbanos

TRABAJO FIN DE GRADO

Autor:
Andrés Ruz Nieto

Directores:
José Santa Lozano
Esteban Egea López

CARTAGENA, 2021

Agradecimientos

Me gustaría agradecer a José Santa Lozano y a Esteban Egea López, directores de este proyecto, toda la ayuda que me han dado en la realización de este Trabajo Fin de Grado y por introducirme en el mundo de la investigación. Gracias a ellos, el trabajo ha obtenido financiación del Ministerio de Transición Ecológica y Reto Demográfico, a través del proyecto MECANO (PGE-MOVES-SING-2019-000104), y del Ministerio de Ciencia e Innovación, a través del programa Ramón y Cajal (RYC-2017-23823) y la red Go2Edge (RED2018-102585-T). También me gustaría dar las gracias a José María Molina García-Pardo, por prestarnos el material necesario para realizar las medidas de precisión.

A mis compañeros del grupo GIRTEL, Jesús y Ramón, por crear un clima de buen rollo en el laboratorio a pesar de no poder vernos mucho por el Covid y por haber realizado el despliegue de la red LoRaWAN, que he podido aprovechar para obtener datos sobre la cobertura.

A mis amigos, por esas largas sesiones de estudio en *pizarra*, ayudándonos y apoyándonos en todo, tanto dentro como fuera de la Universidad. Por todas las BBQ que hemos hecho para desconectar cuando acabábamos los exámenes. Por esos días desde la mañana hasta la noche de pandemia en Discord. Por todos esos momentos en la 1.4 organizando la Teleco LAN Party. Sin vosotros todo hubiera sido distinto.

A Isabel Costa Gómez, por estar ahí siempre que la he necesitado, y por darme tanta caña sabiendo todo el potencial que podía sacar de mí, eres la hermana mayor que no tengo.

A mi familia, en especial a mis padres, por darme todo lo que tengo, por tener tanta paciencia y por apoyarme en cada uno de los pasos y decisiones que he tomado. A mi abuelo Andrés, por todos esos acertijos y problemas que me ponía de pequeño, y, porque aunque no esté aquí para leer esto, sé que hubiera estado y está muy orgulloso de mí.

Muchas gracias a todos.

Resumen

Los entornos de ciudad inteligente requieren el tratamiento de los datos recogidos por un gran número de dispositivos o sensores, donde las tecnologías del Internet de las Cosas (*Internet of Things* - IoT) juegan un papel fundamental. Los requerimientos en cuanto a comunicaciones de estos sistemas complejos hacen que las *Low-Power Wide-Area Networks* (LP-WAN) sean cada vez más utilizadas para la interconexión inalámbrica de los mencionados sensores. Estas redes permiten comunicaciones de bajo consumo energético a grandes distancias. Para ahorrar costes de instalación y mantenimiento, además de posibles fallos en el sistema de la red desplegada, que puede extenderse entre decenas o centenas de nodos y decenas de kilómetros cuadrados, deberíamos analizar previamente el funcionamiento y el rendimiento de dicha red. El principal objetivo de las herramientas de planificación de redes más utilizadas es dar una aproximación de la cobertura, mientras que las herramientas de simulación de red ofrecen información con un mayor grado de detalle sobre el rendimiento de esta. Sin embargo, los *frameworks* actuales de simulación incluyen modelos de propagación limitados, basados en mediciones empíricas y estadísticas, que no consideran las particularidades que puede tener cada escenario, como bien pueden ser los edificios, la vegetación o la elevación del terreno. Esto es fundamental en ciudades, ya que, si no se tienen en cuenta las propiedades del entorno, los resultados pueden ser incorrectos o muy distantes de la realidad, como se podrá ver en los próximos capítulos. En esta línea, este trabajo fin de grado presenta una solución que incluye un simulador de red para medir el rendimiento de las comunicaciones LPWAN en un entorno preciso, un motor 3D para la construcción de este entorno, y una herramienta de trazado de rayos que mejora el modelo de propagación, que ofrece resultados realistas sobre el rendimiento esperado cuando se utiliza la tecnología *Long-Range Wide-Area Network* (LoRaWAN). Hemos validado el correcto funcionamiento de la solución tomando como referencia una campaña de pruebas en la ciudad de Cartagena. Además, los resultados obtenidos con nuestro framework de simulación se han comparado con los datos obtenidos con modelos de propagación conocidos en la literatura, como son Okumura-Hata o Log-Distance. Los resultados indican que nuestra aproximación se ajusta con precisión a los valores experimentales, superando a estos modelos en entornos urbanos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

1.1. Motivación

En la actualidad, la tecnología tiende hacia la creación de escenarios inteligentes dentro de entornos urbanos, las llamadas ciudades inteligentes, pero también rurales, como los propios despliegues automatizados de cultivos. Esto implica la instalación de diversos dispositivos, bien para monitorizar distintos parámetros de interés (ambientales, de volumen de tráfico, etc.) o bien para la activación/desactivación de otros elementos dentro de los despliegues de Internet de las Cosas. Estos sensores podrían estar conectados mediante cables a pasarelas locales, pero las implementaciones en espacios grandes están limitadas por el alto coste que tendría la infraestructura. Debido a ello, se opta por hacer uso de sensores inalámbricos. Hasta hace poco, estas redes se implementaban empleando comunicaciones de corto alcance como Bluetooth, Bluetooth de baja energía (BLE) o ZigBee. Sin embargo, este tipo de tecnologías necesita una pasarela para recopilar datos para finalmente llegar a una red cableada.

Cuando el espacio que hay que cubrir es muy grande, es más recomendable el uso de redes de baja potencia y área amplia (*Low Power Wide Area Networks* - LP-WAN). Las de uso más generalizado son Sigfox, IoT de banda estrecha (NB-IoT) y LoRa. Las dos primeras requieren de una suscripción a un operador. LoRa trabaja en bandas de frecuencias libres, lo que la ha hecho muy popular en la comunidad de investigadores y en despliegues industriales. LoRaWAN se encarga de proporcionar una capa de acceso al medio físico LoRa, ofreciendo la gestión del flujo de datos, seguridad y conexión a la red. En el siguiente capítulo se profundizará más en esta tecnología.

Para la planificación de este tipo de redes podemos encontrar software variado que

nos ayudan a realizar el diseño de la red, pero todos ellos usan modelos de propagación empíricos que podrían no adaptarse correctamente a diferentes entornos de forma automática.

1.2. Propuesta - Hipótesis

Proponemos un framework de simulación LoRaWAN realista con modelado 3D del entorno y empleando trazado de rayos. Dicho framework debe tener en cuenta la configuración del entorno sobre el que se ejecuta la simulación, y también realizar aproximaciones que se ajusten a los escenarios reales sin tener que añadir parámetros extra de configuración que, en ocasiones, pueden ser complicados de calcular o requerir la realización medidas reales para insertar en el modelo.

1.3. Innovación

El framework que se propone en el trabajo incluye la técnica de trazado de rayos para detectar los obstáculos y permitir la reflexión de estos rayos en el resto del entorno. Esta técnica no se ha empleado hasta la fecha para crear soluciones de simulación eficaces para redes LPWAN con LoRaWAN, como podemos apreciar en la literatura sobre simulaciones LoRa (Sección 2.1).

El framework incluirá:

- Modelado 3D de las edificaciones de una ciudad.
- Modelos digitales de elevación del terreno.
- Visualizador del entorno 3D completo.
- Simulador de red capaz de realizar un trazado de rayos para obtener de forma realista el alcance de una comunicación.

1.4. Estructura del trabajo

A continuación se resume el contenido de los capítulos de este trabajo.

- Capítulo 1. Introducción

  Es el capítulo actual, en él se concreta cual es la motivación de realizar este proyecto, la propuesta que se realiza para resolver el problema existente y la innovación aportada.

- Capítulo 2. Estado del arte

  En este capítulo, se pueden ver las diferentes ramas de investigación relacionadas con el trabajo y los diferentes simuladores de red que se han ido desarrollado para la tecnología LoRa.

- Capítulo 3. Tecnologías

  En el Capítulo 3 se realiza, detalladamente, una explicación sobre las distintas tecnologías y software que se han utilizado para el desarrollo del trabajo. Se versa sobre algunas de las especificaciones de LoRaWAN y LoRa, y se describe la utilidad de los distintos programas empleados.

- Capítulo 4. Arquitectura de la solución

  Se realiza una explicación del trabajo realizado para la creación del framework propuesto, acompañando la explicación con imágenes y diagramas del proceso seguido.

- Capítulo 5. Entorno de simulación

  En el Capítulo 5, se explica cómo se ha modelado el entorno que se cargará en el simulador y los diferentes modelos de propagación empleados a la hora de simular.

- Capítulo 6. Despliegues de red usados

  Se presentan los dos despliegues que se han realizado, uno de ellos es despliegue de red LoRaWAN real en la ciudad de Cartagena, que será sobre la que posteriormente se obtengan los datos para ajustar el modelo Log-Distance. El segundo despliegue, realizado con instrumentos de precisión, será empleado para obtener los datos empíricos para realizar la comparación con los datos simulados.

- Capítulo 7. Evaluación

  En este capítulo, se describe el proceso que se ha seguido para configurar la simulación, montaje de la geometría del entorno (edificios y topografía) y modelado de la red en OMNeT++, y se realiza una breve comparación entre el entorno 3D que se obtiene y el entorno real. A continuación se realiza una comparativa entre los datos obtenidos en la realidad mediante mediciones y los obtenidos mediante la simulación.

- Capítulo 8. Conclusiones

  Se muestra la conclusión final del trabajo y se nombran lineas de investigación futuras.

CAPÍTULO 2

ESTADO DEL ARTE

2.1. Estudios previos

A pesar de que LoRaWAN es una de las tecnologías LPWAN más conocidas desde su aparición en 2015, no hay muchos trabajos relacionados con simulaciones realistas de despliegues de red que utilicen esta tecnología.

La primera etapa de simuladores capaces de trabajar con LoRaWAN surgió en 2018 como podemos leer en [1]. Una gran parte de los simuladores revisados en el artículo citado anteriormente no incluyen soporte MAC, es decir, estaban principalmente enfocados en la capa física de LoRa, o no consideran características importantes como el tráfico descendente (del gateaway hacia el nodo final), que puede influir en el rendimiento esperado debido a la ocupación del canal de los mensajes de confirmación de recepción. La gran importancia de este tema hace que los desarrolladores actualicen los simuladores tan solo un año después [2]. En esta referencia se puede apreciar cómo se incluye el soporte MAC y cómo se consideran modelos más precisos, y nuevas características de análisis de los canales, como la limitación del ciclo de trabajo o el consumo de energía.

Revisando la evolución de estos simuladores en la literatura podemos observar las mejoras alcanzadas en la precisión de la simulación LoRaWAN. Uno de los primeros y más nombrados es el simulador LoRaSim [3], basado en SimPy, una plataforma de simulación en Python, que empieza a incluir capacidades limitadas centradas principalmente en el canal físico. En [4], se implementa en LoRaSim la capacidad de evaluar comunicaciones direccionales y múltiples GWs. Posteriormente en [5] se realiza una extensión adicional, ampliando el soporte de LoRaWAN MAC y evaluando el tráfico de

los enlaces descendentes.

Los módulos de NS-3 para la simulación de LoRaWAN también están documentados en la literatura. En [6] se presenta un simulador NS-3 para evaluar las capacidades de acceso al medio de LoRaWAN comparándolo con un esquema ALOHA común. Este trabajo fue ampliado más tarde en [7] con un análisis sobre el impacto de los mensajes de confirmación descendentes en el rendimiento general de la red. El módulo de NS-3 presentado en [8] se centra en las interferencias debidas a la proliferación de nodos finales, y presenta un modelo de error desarrollado en MATLAB, que se importa en el simulador para estudiar la escalabilidad LoRaWAN.

El simulador LoRaFREE [9] comparte el mismo objetivo de mejorar el modelado de errores, considerando las interferencias entre canales y partiendo del trabajo en realizado con LoRaSim. El desarrollo mostrado en [10] sigue la misma aproximación, pero ahora amplía las funcionalidades de LoRaSim con información precisa de consumo de energía en los nodos finales. El trabajo reciente [11] desarrolla un simulador en Python desde cero que considera particularmete la modulación de la señal de LoRa para obtener una evaluación más precisa.

Más tarde, el simulador LoRaWANSim [12] ofrece la posibilidad de configurar distintos parámetros para las capas físicas y MAC, aunque no está integrado con un entorno de simulación de red, lo que limita su uso. Por eso, sus prestaciones son inferiores a las de aquellos módulos integrados en frameworks, como Simpy y, sobre todo, NS-3.

Recientemente han aparecido otras versiones de simuladores LoRaWAN como FloRa [13], el cual está desarrollado como un módulo de OMNeT++, este framework, considera las capacidades MAC de LoRaWAN e implementa capacidades de adaptación de la tasa de datos (*Adaptive Data Rate* - ADR). Actualmente este entorno está ganando mucha popularidad entre los grupos de trabajo que usan simuladores LoRaWAN. En [14] se usa FloRa para obtener resultados de simulaciones de un dron controlado a través de LoRaWAN. En el trabajo reciente [15] los autores añaden a FloRa características de seguridad para evaluar la calidad del servicio de implementaciones de redes IoT.

FloRa es en la actualidad uno de los frameworks más conocidos sobre comunicaciones LoRa, debido a su flexibilidad e integridad. Se pueden recrear despliegues de LoRaWAN dentro de un simulador de red extendido en el mundo académico. Sin embargo, en ninguno de los simuladores anteriormente citados, ni en este último, podemos encontrar un modelo de propagación preciso que consideren la configuración del entorno sobre el cual se está simulando.

Los escenarios urbanos están ganando protagonismo en estos últimos años en el campo de redes de sensores, como pueden ser los despliegues en ciudades inteligentes. Las condiciones urbanas suponen un desafío a la hora de realizar simulaciones debido a la cantidad de construcciones existentes. La orografía del terreno también puede afectar a estos escenarios. En la literatura frecuente hay varios trabajos que

emplean modelos de propagación realistas para LoRaWAN. En [16] se puede apreciar cómo el modelo de propagación Okumura-Hata es capaz de obtener buenos resultados para distancias urbanas limitadas, aunque se obtienen resultados más precisos con herramientas de planificación que utilizan el Modelo de Terreno Irregular con Obstrucciones (Irregular Terrain With Obstructions Model - ITWOM), que tiene en cuenta la elevación del terreno. El trabajo [17] indica que los modelos comunes Okumura-Hata y Log-Normal son propensos a fallar bajo configuraciones urbanas con construcciones y/o una vegetación densa. El modelo Okumura-Hata frente al COST Hata para escenarios de comunicación LoRaWAN punto a punto, sin un estación base convencional colocada en altura. Aunque la comunicación todavía es posible, la presencia de obstáculos en estas condiciones afecta en gran medida al rendimiento.

En [18] se evalúan los modelos Okumura-Hata, Log-Distance, SUI y Ericsson, en donde se obtiene que el primero es el más preciso para áreas urbanas. En [19] se obtiene, mediante regresión, un modelo de pérdida de propagación que proporciona mejores resultados que los modelos empíricos comunes, tanto para interiores como para exteriores, aunque está particularizado para áreas concretas. La propagación en interiores de las comunicaciones LoRaWAN se estudia en [20] proponiendo un modelo que supera al de distancia logarítmica común (Log-Distance) al fusionar una solución de red neuronal con el modelo COST231. Los mismos autores mejoraron sus resultados en [21] utilizando lógica difusa basada en redes adaptativas, que reduce la complejidad y mejora la eficiencia. Si se utiliza una herramienta de planificación que emplea trazado de rayos [22] se obtienen resultados muy precisos, lo que enfatiza las virtudes de este método cuando se modela el entorno.

De los diferentes trabajos que evalúan modelos de propagación, destaca el uso de los modelos Okumura-Hata y Log-Distance para LoRaWAN, que también se puede ver claramente en los principales simuladores presentados anteriormente [3, 6, 9, 10, 12]. De hecho, Okumura-Hata aparece como un buen modelo empírico para esta tecnología de comunicación cuando se configuran los parámetros adecuados. Otros trabajos obtienen, mediante regresión, modelos de pérdidas adaptados a escenarios particulares. Sin embargo, cuando se dispone de un modelo 3D del escenario a evaluar, el trazado de rayos es más preciso. En este trabajo fin de grado se presenta el enfoque para integrar el trazado de rayos en un simulador de red, resolviendo la generación del modelo 3D mediante la automatización de la extracción de datos 3D de fuentes abiertas. Los resultados se comparan con los modelos Okumura-Hata y Log-Distance ya que son muy conocidos en la literatura y son una buena referencia para contrastar datos.

2.2. Herramientas de simulación

En la actualidad existen diversas herramientas que permiten simular redes, como por ejemplo:

- Cloud-RF [23]: Es un servicio para el modelado de propagación, dispone de datos de terreno, diferentes patrones de radiación de antenas y modelos de propagación.
- Atoll LPWA [24]: Atoll es una plataforma que permite el diseño y optimización de diferentes tecnologías inalámbricas. Es compatible con los últimos avances tecnológicos como Massive MIMO, beamforming y la propagación de ondas milimetricas utilizadas en 5G. Incluye un módulo específico para diseño de redes LPWAN. Su coste no es visible en su página web.
- S_IoT [25]: Esta herramienta ofrece un enfoque integral para las simulaciones de IoT con métodos y análisis adaptados a las limitaciones de LPWAN. Basado en el modelo de predicción de múltiples trayectos. S_IoT trabaja con cualquier tipo de planificación de redes IoT aprovechando los datos GIS que posee la empresa Siradel (propietarios de esta herramienta).
- Radio Mobile [26]: Software gratuito desarrollado por Roger Coudé que simula la propagación de radio frecuencias. Los datos del terreno los obtiene de fuentes abiertas como OpenStreetMap. Su uso no está dedicado a LoRaWAN y su documentación es muy escasa.

Como se puede ver, no existen simuladores de red que tengan en cuenta, no solo el terreno, sino también las construcciones existentes en un entorno. En este aspecto, este proyecto introduce una gran innovación a la hora de realizar simulaciones realistas.

CAPÍTULO 3

TECNOLOGÍAS

A continuación se presentan las tecnologías empleadas para la realización de este proyecto.

3.1. LoRa y LoRaWAN

Tanto el término LoRa como LoRaWAN [27] se utilizan para para referirse a la tecnología LoRaWAN, sin embargo, LoRa se ocupa de la capa física, mientras que LoRaWAN proporciona el protocolo MAC sobre el cual se envían los mensajes a nivel de aplicación.

LoRa es un tipo de modulación basada en la técnica Chirp de Espectro Ensanchado (*Chirp Spread Spectrum* - CSS) creado por Semtech. CSS junto con la corrección eficiente de errores mediante técnicas de corrección de errores hacia delante (*Forward Error Correction* - FEC) hace que LoRa una tecnología muy robusta frente a interferencias y permite transmisiones de largo alcance.

Los principales parámetros que definen las comunicaciones LoRa son:

- Ancho de banda (*Bandwidth* - BW): Determina el rango de frecuencias empleadas en la transmisión. Si se aumenta el BW, también aumentará la velocidad de transmisión de datos, pero la sensibilidad será menor. Normalmente se emplean anchos de banda de 125KHz, 250KHz y 500KHz.
- Factor de propagación (*Spreading Factor* - SF): Determina la proporción de los símbolos frente a ”Chirps” empleados, lo que indica el ”esfuerzo” que realiza la

modulación para representar un dato. Los factores de propagación elevados incrementan la sensibilidad y el rango de transmisión, reduciendo la tasa de datos y aumentando el tiempo en el cual el paquete está en el aire.

- Frecuencia de portadora (*Carrier Frequency* - CF): Es la frecuencia central utilizada y esta depende de la región donde se encuentre el dispositivo, siendo de 902-928MHz en Estados Unidos y 863-870Mhz en Europa. Estas bandas no necesitan ningún tipo de licencia para hacer uso de ellas, lo que facilita los despliegues de redes para la industria o investigaciones.
- Tasa de codificación (*Coding Rate* - CR): Define qué modo de FEC se va a emplear. Los modos soportados son 4/5, 4/6, 4/7 y 4/8. Conforme aumenta el CR, se proporciona una protección extra, pero al igual que en el SF, el tiempo de transmisión será mayor ya que el paquete es más grande.

Como resultado obtenemos un medio de comunicación robusto y energéticamente eficiente, lo que permite llegar a dispositivos a varios kilómetros del gateaway (GW) empleando potencias de transmisión que varían de 2dBm a 20dBm pero a velocidades de transmisión limitadas (usualmente se transmite a una velocidad de 19.2Kbps). Esto cumple con los requisitos de comunicación de muchos dispositivos, sensores y despliegues de IoT.

LoRaWAN trabaja sobre LoRa, y define la estructura de los paquetes, el acceso al medio, la gestión del flujo de datos y su seguridad. Fue definida por LoRa Alliance y cubre la ruta desde los dispositivos finales hasta lo servidores de aplicación. El acceso al medio está basado en el mecanismo ALOHA, con el objetivo de reducir la complejidad del protocolo.

Los dispositivos finales envían paquetes con los datos a los LoRa GWs, que se encargarán de redirigir a estos hacia el servidor LoRa (*LoRa Server* - LS). Por lo tanto, desde el punto de vista de LoRaWAN, la comunicación es extremo a extremo, es decir, desde los dispositivos final hasta los servidores LoRa. En este punto se lleva a cabo la gestión del flujo. A pesar de que la comunicación inalámbrica entre los nodos y GWs está basado en LoRa, la conexión entre GW y LS se considera un enlace TCP/IP sobre tecnologías de comunicación más comunes, como pueden ser un despliegue WAN, WiFi, 5G o Ethernet. Los paquetes de datos se envían finalmente a los servidores de aplicación, la seguridad también es extremo a extremo y se proporciona mediante el ”Estándar de Cifrado Avanzado” (*Advanced Encryption Standard* - AES).

En LoRaWAN existen tres tipos de nodos finales: A, B y C. La clase A es la más común y permiten la comunicación bidireccional entre nodos, pero el tráfico de enlace descendente se permitirá solamente después de una transmisión de enlace ascendente mediante la habilitación dos ventanas de recepción. La clase B proviene de la clase A y añade una ventana de recepción adicional en programada en un tiempo determinado, para ello el dispositivo se deberá sincronizar con el gateaway haciendo uso de *beacons*. Por último, la clase C mantiene abiertas las ventanas de recepción de forma continua,

salvo cuando se está realizando una transmisión. Debido a ello la energía consumida aumentará de la clase A a la C pero la latencia disminuirá. Es importante saber que LoRaWAN define un limite máximo del ciclo de trabajo del 1%, es decir, un dispositivo sólo puede ocupar el 1% del tiempo del canal indicado por el SF. La adaptación de la tasa de datos (*Adaptive Data Rate* - ADR), permite la asignación dinámica del SF que se utiliza desde el LS, con el objetivo de mantener un equilibrio entre la ocupación del canal, la tasa de datos general y la vida útil de la batería del nodo final, ya que lo más normal es que estos no puedan estar conectados a la red eléctrica debido a su situación geográfica.

3.2. OMNeT++

OMNeT++ [28] es un *framework* de simulación extensible, modular y basado en componentes de C++ que principalmente se usa para construir simuladores de red. El término ”Red” se entiende en un sentido más amplio que incluye redes de comunicación cableadas o inalámbricas, redes en un chip, redes de colas, etc. Otros *frameworks* desarrollados como proyectos independientes, proporcionan funcionalidades específicas como soportar redes de sensores, redes ad-hoc inalámbricas, protocolos de internet, modelado del rendimiento y redes ópticas. OMNeT++ ofrece un IDE basado en Eclipse, un entorno de tiempo de ejecución gráfico y otras herramientas. Existen extensiones para simulaciones en tiempo real, emulación de redes, integración de bases de datos o de SystemC entre muchas funciones. OMNeT++ es distribuido bajo la *Academic Public License*.

A pesar de que OMNeT++ no fue creado para ser un simulador de red, ha ganado mucha popularidad como plataforma de simulación de red en la comunidad científica, así como en entornos industriales y agrupa a una gran comunidad de usuarios.

OMNeT++ proporciona una arquitectura de componentes para los modelos. Los componentes, también llamados módulos, se programan en C++ y son ensamblados con otros originando componentes y modelos superiores empleando un lenguaje de alto nivel (NED). La reutilización de los modelos es gratuita. OMNeT++ tiene un gran soporte de interfaz de usuario y, debido a su arquitectura modular, el kernel de simulación puede integrarse fácilmente en otras aplicaciones.

OMNeT++ está compuesto por:

- Librería de simulación (C++).
- Lenguaje de descripción de la topología NED.
- IDE basado en la plataforma Eclipse.
- GUI en tiempo de ejecución para simulaciones interactivas (Qtenv)

- Interfaz de línea de comandos para la ejecución de la simulación (Cmdenv).
- Utilidades (por ejemplo, herramienta de creación de archivos make).
- Documentación, ejemplos de simulación.

En todos estos años, investigadores de diversas áreas han escrito innumerables modelos de simulación y *frameworks*: colas, modelado de recursos, protocolos de Internet, redes inalámbricas, LAN conmutadas, redes punto a punto, transmisión de imagen y sonido, redes de malla, redes de sensores inalámbricos, redes vehiculares, redes ópticas y muchas más. La mayoría de estos *frameworks* son de código abierto y se desarrollan como proyectos independientes siguiendo sus propios ciclos de lanzamiento.

3.3. Framework INET

Framework INET [29] puede considerarse la biblioteca de modelos de protocolo estándar de OMNeT++ y contiene modelos para la pila de Internet y muchos otros protocolos y componentes. Este *framework* se mantiene por el mismo equipo de OMNeT++ y utilizan parches y modelos aportados por miembros de la comunidad. Otros *frameworks* toman INET como base y lo extienden en direcciones específicas, como redes vehiculares (Veins, CoRE), redes superpuestas o punto a punto (OverSim), LTE (SimuLTE) o LoRa (FLoRa).

3.4. FloRa

FloRa [13] es un *framework* para realizar simulaciones punto a punto para redes LoRa. Este módulo funciona sobre OMNeT++ y también utiliza componentes del *framework* INET. FloRa permite la creación de redes LoRaWAN con módulos para nodos LoRa, GW y un servidor LoRa. La lógica de la aplicación se puede implementar como módulos independientes que están conectados con el servidor LoRa. El servidor LoRa y los nodos admiten la gestión dinámica de sus parámetros a través de la adaptación de la tasa de datos. Finalmente, las estadísticas de cada nodo, como puede ser el consumo de energía, paquetes enviados, recibidos y perdidos, potencia recibida, relación señal-ruido, entre otras, se recopilan para que el usuario pueda consultarlas.

Sus características son:

- Modelo preciso de la capa física de LoRa (incluidas las colisiones y el efecto de captura).
- Simulaciones con una (o más) gateaway en la red.

- Simulaciones de extremo a extremo, incluido el modelado preciso de la red de retorno.
- Estadísticas de consumo energético en red.

3.5. Unity

La plataforma Unity [30] se emplea para la creación de de espacios 2D, 3D, realidad virtual o realidad aumentada para videojuegos u otras simulaciones. Soporta multitud de plataformas y es utilizado en todo el mundo. Permite el empleo de distintos lenguajes de programación como Boo, JS o C# y se puede integrar con Visual Studio y MonoDevelop. La versión gratuita incluye todas las funciones necesarias para la realización del proyecto.

3.6. Veneris

Veneris [31] es un framework empleado para simular tráfico y no realiza ninguna simulación relacionada con la red o la comunicación. Se utiliza para simular el tráfico rodado con movilidad realista. Para simular redes vehiculares con este framework, con el mismo generaríamos la simulación de tráfico para posteriormente enviar los resultados a OMNeT++, que sería el encargado de simular la red.

Veneris es un conjunto de componentes de Unity que proporcionan una simulación de red de carreteras realista en un entorno 3D interactivo:

- Los componentes del vehículo incluyen un modelo de la dinámica de este.
- Los componentes del constructor se utilizan para generar los elementos del escenario: carreteras, intersecciones, semáforos o edificios.
- Los componentes de comunicación implementan la comunicación con los módulos de simulación.
- Otros componentes modelan el comportamiento de los vehículos en las carreteras, intersecciones y las interacciones con otros vehículos.

Los gestores manejan diferentes aspectos de la simulación a nivel global. Con Veneris se puede crear un escenario de simulación a partir de datos de mapas del mundo real de OpenStreetMap y ejecutar una simulación de tráfico en él. El objetivo de Veneris es apoyar la simulación de redes vehiculares realistas junto con otras herramientas como OMNeT ++ que proporciona la simulación de red, y Opal, un simulador del canal

físico de comunicación que soporta muti-trayecto con trazado de rayos en GPU. Para este proyecto se usará Veneris para generar el modelo 3D de la ciudad y, en líneas futuras, servirá para poder mover los nodos en el entorno.

3.7. Opal

Opal [32] es un simulador de propagación de radio frecuencias basado en el trazado de rayos (*Ray-Tracing*) en GPU. Es complementario a los otros dos módulos (Veneris y módulo de Veneris para OMNeT++), es decir, se puede utilizar con cualquiera de ellos o con ninguno. Puede usarse como simulador de propagación independiente en C++.

Se puede usar directamente como canal de comunicación en OMNeT++. En este caso, en el archivo de mallas 3D también habrá que añadir una representación de los edificios en su escenario para que Opal pueda calcular las reflexiones de los rayos en el entorno. Para ello se tendrá que generar un escenario, guardar las mallas en archivos de texto y cargarlas en OMNeT++ para, finalmente, usarlos en la simulación.

Opal también se puede utilizar directamente en la simulación de Veneris pero, en este caso, no se implementa una pila de protocolos adicional y sólo se podrá transmitir y obtener la potencia recibida. A pesar de esto, sigue siendo útil para hacer una caracterización electromagnética, por ejemplo, para simular una cobertura celular.

3.8. Herramientas GIS

Para la realización del modelo 3D del entorno con el modelado digital de la elevación del terreno se han empleado varias herramientas de geoprocesamiento. Se emplearán las siguientes herramientas y su único uso será el de convertir el relieve a un formato que se pueda importar en Unity:

- QGIS [33]: Es un sistema de información geográfica Open Source, donde se puede visualizar, gestionar, editar y analizar datos georeferenciados.

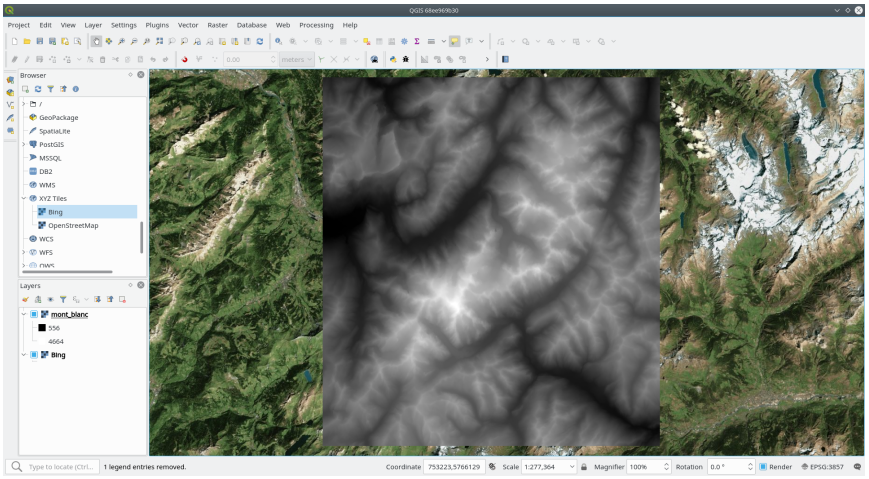

Figura 3.1: QGIS

- dem2mesh [34]: Gracias a este software programado en C++ podemos convertir una imagen .tif, que contiene información del relieve en un archivo *Polygon File Format*.

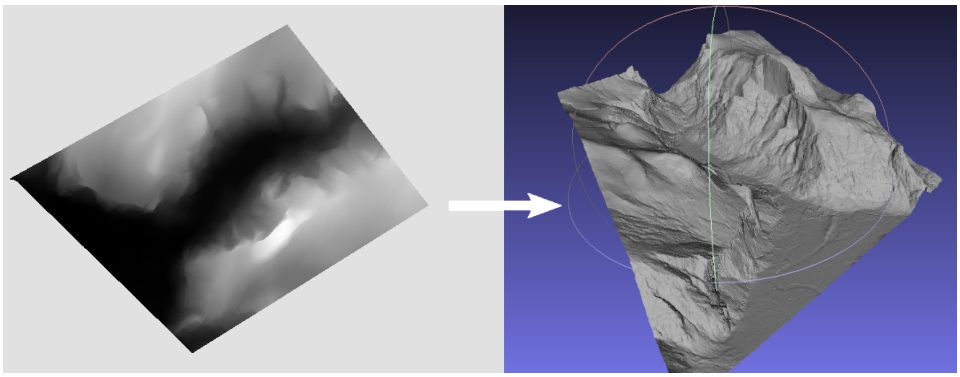

Figura 3.2: dem2mesh

- MeshLab [35]: Este último programa, proporciona herramientas de procesado, limpieza, renderizado y conversión de mallas. Se encargará del último paso antes de importar la malla desde Unity.

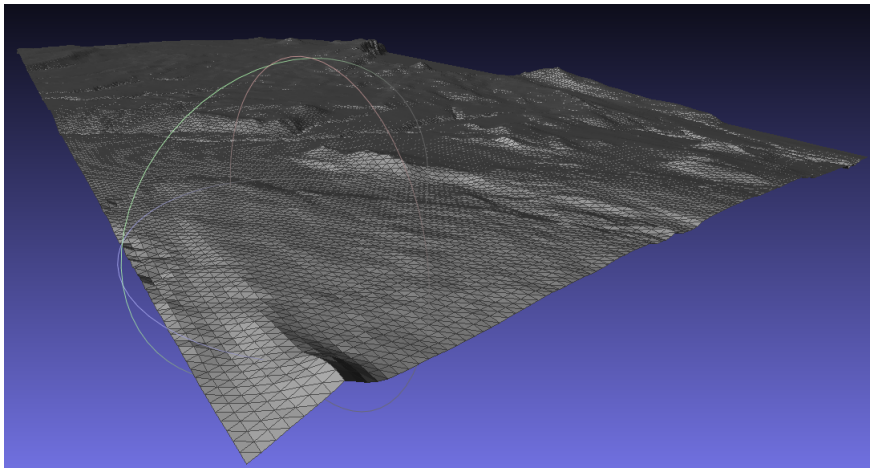

Figura 3.3: MeshLab

CAPÍTULO 4

ARQUITECTURA DE LA SOLUCIÓN

4.1. Escenario objeto de estudio

Este proyecto se enfoca hacia la evaluación de un escenario de red LoRaWAN. Estas se están expandiendo en muchos entornos, desde la ciudad a los pueblos, pasando por industrias y campos de cultivo, con el fin de monitorizar distintos parámetros de interés o controlar distintos dispositivos. Las redes LoRaWAN están formadas por distintos elementos que se pueden ver resumidamente en la Figura 4.1.

Esta hace referencia a una red LoRaWAN desplegada en una ciudad. Estás redes están formadas por GWs, nodos, un servidor LoRa, donde se recopilan los datos y servidores de aplicación, donde se ejecutan aplicaciones para tratar y visualizar los datos recogidos. Además, existen multitud de nodos para supervisar distintos parámetros o activar otros dispositivos. Por ejemplo, en una ciudad se podría hacer un control del tráfico en distintas zonas con el fin de ofrecer al usuario información sobre por qué calles ir para ahorrar tiempo, monitorizar la contaminación acústica o del aire, ver el estado de llenado de contenedores o incluso encender y apagar el riego de los jardines, entre otras muchas utilidades. Este trabajo fin de grado se centra en un entorno urbano donde se pueden encontrar sensores de tráfico, humedad, contaminación, de capacidad de contenedores o dispositivos para activar riego y luces.

Planificar un despliegue de red en una ciudad es complicado porque no se puede obtener directamente el rendimiento de la red esperado en estos entornos sin haberlos probado experimentalmente, ya que hay que tener en cuenta que existen árboles, edificios, y otro tipo de estructuras u objetos que pueden degradar la señal a la hora de establecer una comunicación.

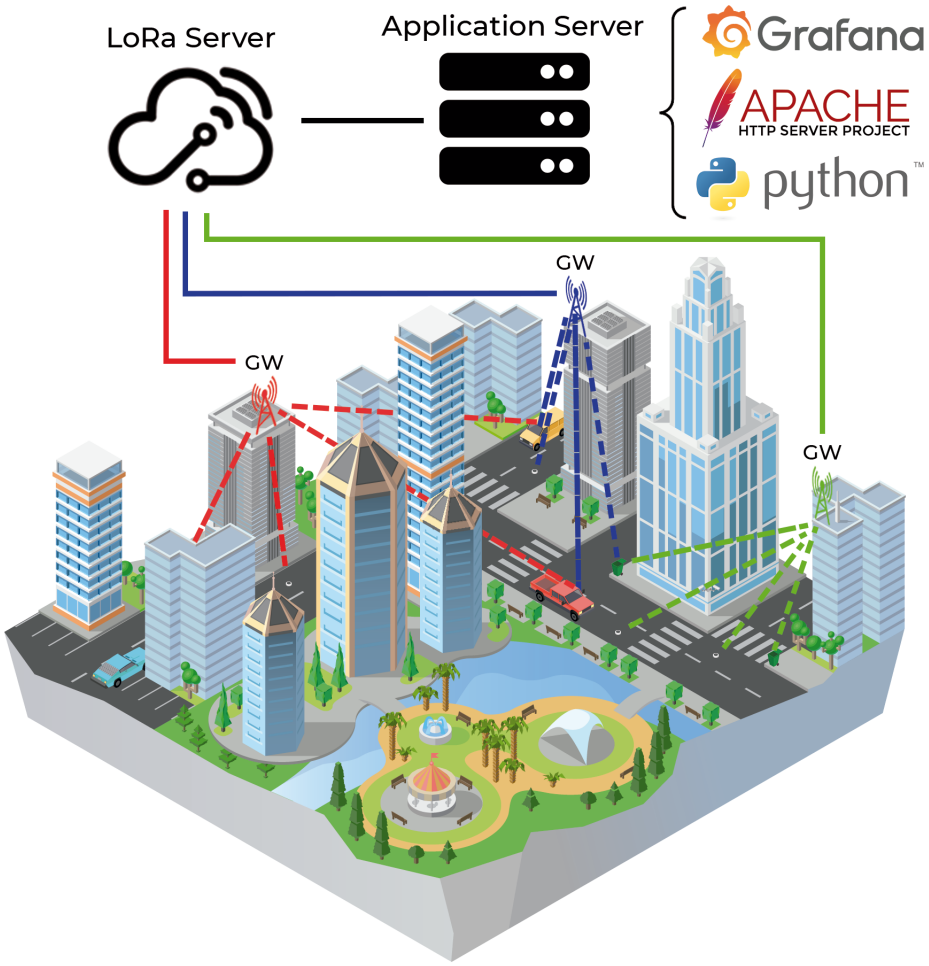

Figura 4.1: Gráfico simplificado de red LoRaWAN

4.2. Modelo general de la solución

En La Figura 4.2 se muestra la solución propuesta que presenta este proyecto. Para ello se hace uso de software dedicado para cada parte del desarrollo. Estos diferentes programas se complementan entre si para, finalmente, poder realizar una simulación de comunicaciones realista en un entorno urbano.

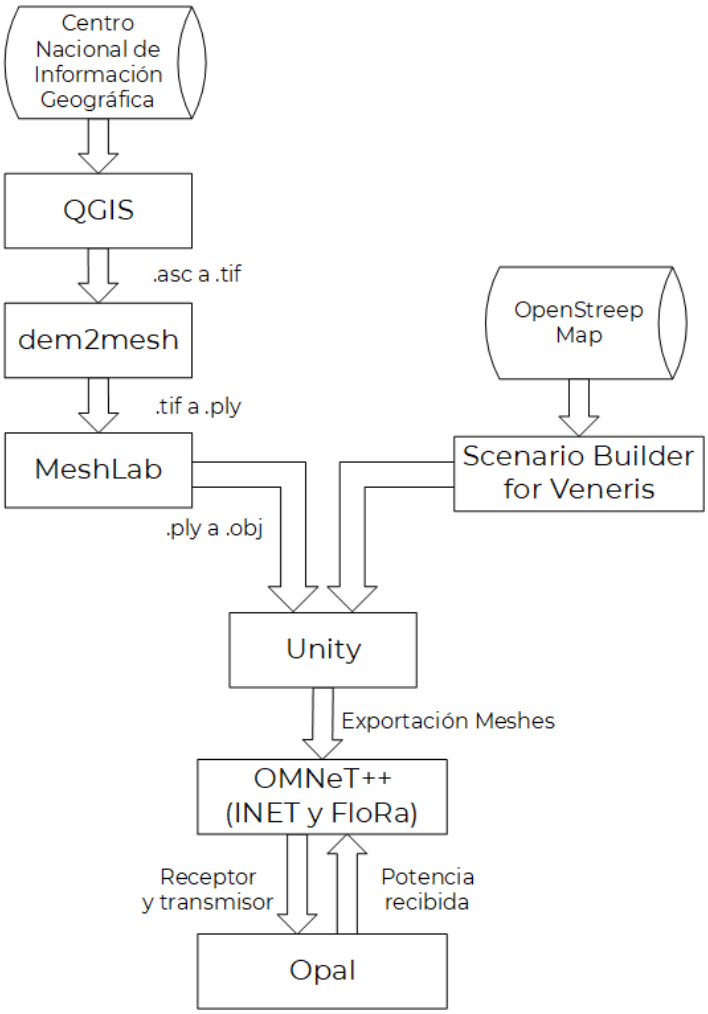

Figura 4.2: Diagrama de software empleado en la arquitectura de la solución

Gracias al Centro Nacional de Información Geográfica (CNIG), se pueden obtener modelos digitales de elevación de toda la superficie de España. Mediante el software QGIS se recorta la zona del terreno deseada que será exportada en formato .tif, que se empleado para almacenar imágenes de mapa de bits. Esta imagen está en escala de grises. Posteriormente este archivo se transforma en un .ply (*Polygon File Format*) y, empleando MeshLab, se obtendrá el un archivo en formato .obj (formato de definición geométrica) el cual se puede importar desde Unity.

La información de la posición geográfica de cada estructura de la ciudad se obtiene de OpenStreepMap y el "Scenario Builder" de Veneris se encargará de importar esta información en Unity.

Posteriormente, las mallas que forman el modelo 3D se exportan en archivos de texto. Cada malla genera cuatro archivos, incluyendo los vértices de ésta, los triángulos que la forman, la matriz de transformación que moverá la malla a su posición, ya

que todos los modelos se generan en la posición de origen para después colocarlo en su lugar, y el último archivo, que contendrá información sobre las propiedades del material de construcción de las estructuras (permitividad relativa (Ecuación 4.1) y conductividad (Ecuación 4.2)), que se pueden expresar mediante las siguientes fórmulas [36]:

$$\eta' = af^b \tag{4.1}$$

$$\sigma = cf^d \tag{4.2}$$

Donde f es la frecuencia en GHz.

Material	Parte real de la permitividad relativa		Conductividad S/m	
	a	b	c	d
Hormigón	5.31	0	0.0326	0.8095
Ladrillos	3.75	0	0.038	0
Metal	1	0	10^7	0
Suelo muy seco	3	0	0.00015	2.52
Suelo seco	15	-0.1	0.035	1.63
Suelo húmedo	30	-0.4	0.15	1.30

Tabla 4.1: Propiedades de los materiales empleados en Opal

La ubicación de estas mallas se pasará como parámetro al simulador en OMNeT++, el cual, a pesar de no tener una interfaz gráfica, generará el modelo de la ciudad. Empleando INET y FloRa se simulará una red LoRaWAN. Cada vez que se genere una transmisión entre nodo y gateaway, Opal calculará las reflexiones de las ondas en los edificios modelados para, finalmente, devolver la potencia total recibida.

4.3. Implementación del sistema

Para realizar llamadas a Opal se han incluido modificaciones en el código de FloRa. A continuación se puede observar un diagrama que contiene las clases que han sido creadas para el correcto funcionamiento del simulador con el modelo de propagación de Ray-Tracing. Posteriormente se explicará detalladamente qué implementa cada clase y para qué se emplea.

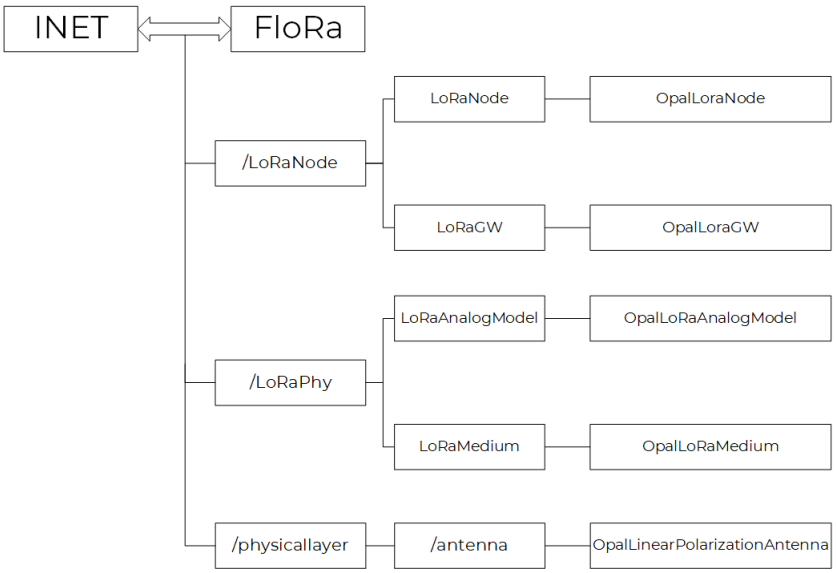

Figura 4.3: Esquema de funciones para la integración de FloRa con Opal

- OpalLoRaNode y OpalLoRaGW: Son dos módulos de OMNeT++ (.ned) los cuáles extienden de LoRaNode y LoRaGW respetivamente. A cada uno de ellos se les han añadido dos parámetros: ”id”, que es un número entero que identifica al nodo o al gateaway en Opal, y ”receptionRadius”, que es un número decimal que indica al simulador el radio de la esfera de recepción de las antenas.
- OpalLoRaAnalogModel: Deriva de la clase LoRaAnalogModel, como se puede ver en el diagrama, y se divide en un archivo de cabecera (.h), un archivo de implementación (.cpp) y un módulo de OMNeT++(.ned). Esta clase se encarga de llamar a la función que calculará la potencia recibida a través del trazado de rayos que proporciona Opal, el cual también se encargará de añadir las ganancias de las antenas a esta potencia.
- OpalLoRaMedium: Deriva de la clase LoRaMedium y está formado por tres archivos, al igual que la clase anterior. Esta clase se encarga de toda la comunicación con Opal y se implementa una nueva funcionalidad para acelerar las simulaciones con trazado de rayos si los nodos están posicionados de forma estática. Se crea una matriz de vectores de potencia que incluyen el número de identificación del emisor, el del receptor y la potencia. Si hemos indicado que la simulación es estática, la primera vez que se emita entre un emisor y un receptor determinado, se calculará la potencia recibida a través de Ray-Tracing. Posteriormente, cuando se tenga que realizar una nueva comunicación entre esta pareja de emisor-receptor, simplemente se tendrá que consultar este vector, facilitando así la ejecución de este simulador en ordenadores sin Ray-Tracing por hardware. Actualmente sólo admiten trazado de rayos con aceleración por hardware las tarjetas de la gama RTX de NVIDIA y a partir de la gama RX 6000 de AMD.
- OpalLinearPolarizationAntenna: Deriva de AntennaBase se encarga de obtener la ganancia de transmisión/recepción de la antena mediante la lectura de un archi-

vo de texto donde se encuentran las ganancias. Mediante los parámetros que se pueden ver en la figura 4.4 se configura el simulador para que pueda leer correctamente el archivo de ganancias ya que no todos los diagramas de radiación de las antenas son iguales de precisos.

```
double initAzimuth=default(0.0);
double azimuthDelta = default(1.0);
double endAzimuth=default(360.0);
double initElevation=default(0.0);
double elevationDelta = default(1.0);
double endElevation=default(180.0);
bool useDecimalDegreeDelta= default(false);
```

Figura 4.4: Parámetros de configuración de OpalLinearPolarizationAntenna

CAPÍTULO 5

ENTORNO DE SIMULACIÓN

5.1. Modelos de propagación considerados

Los modelos de propagación Log-Distance y Hata-Okumura se emplean para obtener valores de simulación de referencia, con el fin comparar los datos obtenidos con el trazado de rayos.

5.1.1. Log-Distance

Está basado en el cálculo de la potencia utilizando la distancia logarítmica [37] entre emisor y receptor, a partir de un "offset" y un término aleatorio. El modelo en dB viene dado por la siguiente fórmula (Ecuación 5.1).

$$[P_L(d)]\,dB = [P_L(d_0)]\,dB + 10\,n\,log_{10}\left(\frac{d}{d_0}\right) + \chi_\sigma,\ d_0 \leq d \tag{5.1}$$

Donde d es la distancia entre transmisor y receptor en metros, n es el exponente de pérdidas, $P_L(d_0)$ es la pérdida a una distancia de referencia d_0 y χ es el desvanecimiento representado como una variable aleatoria con una distribución normal $N(0, \sigma^2)$. Todos estos valores son calculados en el Capítulo 7.

5.1.2. Okumura-Hata

Sobre este modelo se habla por primera vez en el trabajo realizado por Y. Okumura en [38]. Aquí se presenta un modelo de propagación basado en la fórmula de pérdidas en espacio libre (*Free Space Loss*), pero considerando la ganancia de frecuencia, antena y propagación debido a las diferentes alturas de instalación de los dispositivos, además de una ganancia de corrección para simular las pérdidas ocasionadas por las características del entorno. M. Hata mejora este modelo en [39], nombrando al modelo como Okumura-Hata, considerando la reflexión, difracción y dispersión de las señales e incluye explícitamente las alturas de las antenas nodo y estación base como se puede ver en las siguiente fórmulas.

$$[P_{Lurban}]\, dB = A - a\,(h_{ed}) + B\; log_{10}(d) \tag{5.2}$$

$$a(h_{ed}) = 0{,}8 + (1{,}1\; log_{10}(f) - 0{,}7)\; h_{ed} - 1{,}56\; log_{10}(f) \tag{5.3}$$

$$A = 69{,}55 + 26{,}16\; log_{10}(f) - 13{,}82\; log_{10}(h_{gw}) \tag{5.4}$$

$$B = 44{,}9 - 6{,}55\; log_{10}(h_{gw}) \tag{5.5}$$

La Ecuación 5.2 proporciona las pérdidas en entornos urbanos, donde los factores A y B vienen dados por las Ecuaciones 5.4 y 5.5 respectivamente y el parámetro $a(h_{ed})$ empleado en pequeñas y medianas ciudades se calcula a partir de la Ecuación 5.3. f es la frecuencia portadora en MHz, h_{gw} y h_{ed} son las alturas de las antenas GW y nodo dadas en metros respectivamente, y d es la distancia entre ellos en kilométros. En este caso, f adopta un valor de 868 MHz y el resto de parámetros se obtienen en puntos concretos dentro de la simulación.

5.2. Modelado del entorno

Como se ha indicado anteriormente, se emplea Unity para obtener el modelado del entorno en 3D para llevar a cabo la simulación de forma realista. Para la construcción de los edificios en dicho software, se utiliza ”Veneris Simulator”. En el constructor de escenarios [40] se selecciona la zona que se desea simular (Figura 5.2); en este caso se escoge una parte de la ciudad de Cartagena. Pulsando sobre ”Generate Scenario” se descargará un archivo comprimido (.zip) que contendrá toda la información necesaria

para construir los edificios y carreteras en Unity. Para este proyecto solo interesan las edificaciones, ya que más tarde se generará la topografía sobre la que se ubicarán estas.

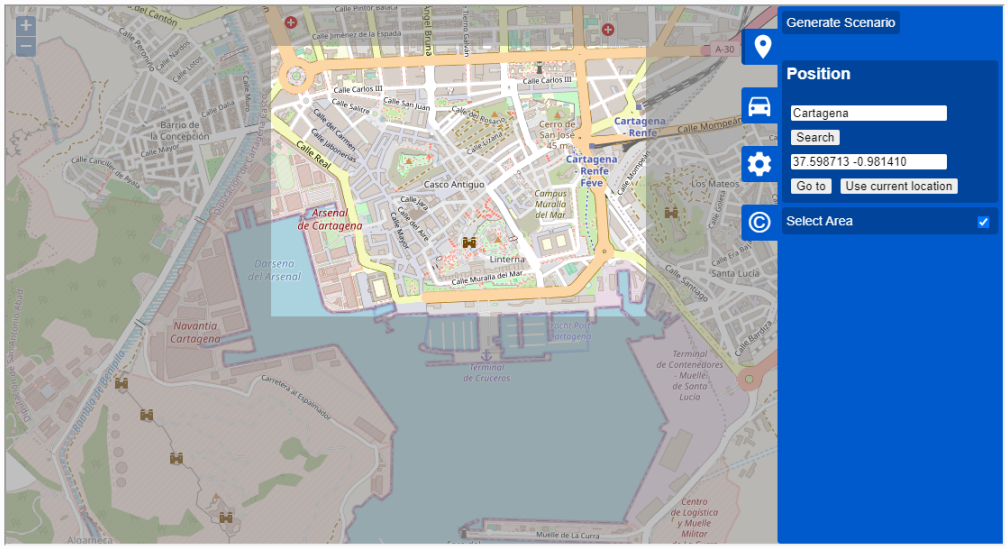

Figura 5.1: Veneris Scenario Builder

En cuanto a la elevación del terreno, se hará uso de la información ofrecida por el Centro Nacional de Información Geográfica (CNIG) [41], en concreto se empleará la sección MDT02-ETRS89-HU30-0977-2-COB2 del Modelo Digital del Terreno - MDT02.

Figura 5.2: Centro Nacional de Información Geográfica

El CNIG ofrece estos archivos en formato ASCII matriz ESRI, por lo que antes de poder ser insertado en Unity, este tendrá que ser tratado para hacerlo compatible con el

motor gráfico. Además se seleccionará la parte que interesa simular para no cargar toda la topografía que ha sido descargada. Para ello se hará uso del software libre QGIS, donde se importará el archivo descargado para, a continuación, recortar la zona deseada y exportarla a formato .tif. Después el archivo se pasa de formato .tif a formato .ply, haciendo uso de "dem2mesh" y finalmente empleando el software MeshLab, se podrá convertir este último archivo a uno con formato .obj, el cual puede ser leído por Unity.

Finalmente se obtendrá un escenario como el que se puede ver a continuación:

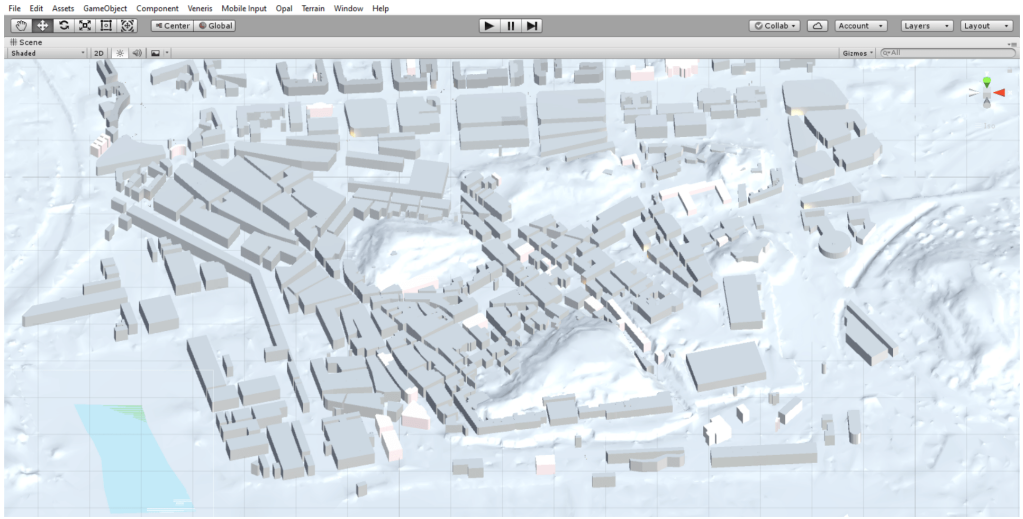

Figura 5.3: Modelado 3D de Cartagena

5.3. Ray-Tracing

El trazado de rayos (*Ray-Tracing*) es un conjunto ordenado de operaciones sistemáticas cuyo objetivo es calcular la incidencia de un rayo en un superficie y simular su efecto sobre esta (reflexión, refracción y difracción). En los videojuegos esta tecnología se emplea para mejorar la calidad de los reflejos y las sombras de las iluminaciones.

En el caso de este trabajo, se usa *Ray-Tracing* con el fin de simular el comportamiento real de una onda al estar en un entorno donde se puede encontrar con distintos obstáculos y obtener la potencia real que se recibe en la antena, de este proceso se encarga Opal.

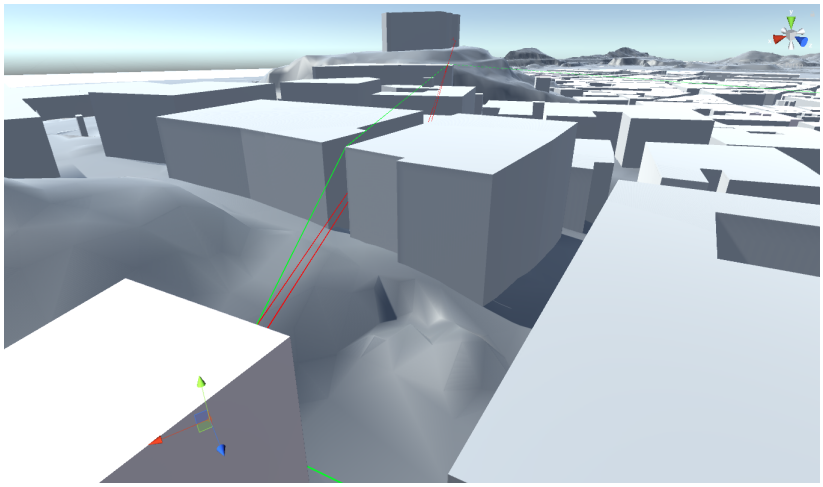

Figura 5.4: Visualización de los rayos en Unity

Opal, en este caso, se ha configurado para realizar simulaciones empleando tanto reflexiones como difracciones, las reflexiones siempre están activadas, pero para que Opal calcule las difracciones, esta opción se habilita de la siguiente forma. El archivo *edges.txt* contiene un listado con todos los *edges* que contiene la ruta, Opal leerá ese archivo y los cargará, esto servirá para calcular las difracciones sobre las estructuras.

```
**.OpalLoRaRadioMedium.useDiffraction = true
**.OpalLoRaRadioMedium.loadEdgesFromFile = true
**.OpalLoRaRadioMedium.edgesPath = "/home/anrunie/Documentos/GIRTEL/meshes"
**.OpalLoRaRadioMedium.edgesFileList = "edges.txt"
```

Figura 5.5: Código que habilita el cálculo de difracciones

También hay que pasarle a Opal el directorio donde se encuentran las mallas de las estructuras y del suelo. Se sigue el mismo proceso que para los *edges*, Opal leerá el archivo *names.txt* y cargará cada malla con sus propiedades. Sin mallas no se pueden calcular las reflexiones.

```
**.OpalLoRaRadioMedium.loadMeshesFromFile = true
**.OpalLoRaRadioMedium.meshesPath = "/home/anrunie/Documentos/GIRTEL/meshes"
**.OpalLoRaRadioMedium.meshesFileList = "names.txt"
```

Figura 5.6: Código que habilita la carga de mallas en Opal

Para configurar las ganancias de las antenas hay que usar el tipo de antena "*OpalLinearPolarizationAntenna*". También hay que habilitar la carga de las ganancias para las antenas.

```
**.LoRaGWNic.radio.antennaType = "OpalLinearPolarizationAntenna"
**.LoRaNic.radio.antennaType = "OpalLinearPolarizationAntenna"
**.OpalLoRaRadioMedium.useAntennaGain = true
**.OpalLoRaRadioMedium.antennaGainFilePaths = "/home/anrunie/Documentos/GIRTEL/gainsPath.txt"
*.loRaNodes[*].LoRaNic.radio.antenna.useAntennaGain = true
*.loRaNodes[*].LoRaNic.radio.antenna.gainFilePath = "/home/anrunie/Documentos/GIRTEL/gain11467.txt"
*.loRaGW[*].LoRaGWNic.radio.antenna.useAntennaGain = true
*.loRaGW[*].LoRaGWNic.radio.antenna.gainFilePath = "/home/anrunie/Documentos/GIRTEL/gain17514.txt"
```

Figura 5.7: Código que habilita la carga las ganancias para las antenas

El radio de recepción de las antenas está configurado a 1m, también se ha configurado Opal para que el número máximo de reflexiones sea de 30.

CAPÍTULO 6

DESPLIEGUES DE RED USADOS

6.1. Despliegue de red LoRaWAN real

Se ha aprovechado un despliegue de una red LPWAN en Cartagena con LoRaWAN desarrollado en el grupo de investigación GIRTEL de la UPCT. Esta red servirá, dentro de un proyecto de investigación en curso, para monitorizar tanto la contaminación, a través de sensores montados en patinetes eléctricos, como el tráfico en distintas zonas de la ciudad, mediante sensores de paso instalados en el asfalto.

El GW (Figura 6.1) está instalado en la azotea del Edificio de Laboratorios de Investigación de la Universidad Politécnica de Cartagena. Este GW está conectado mediante Ethernet al servidor LoRa, donde se recogen los datos para su tratamiento. El valor de potencia recogido se empleará para entrenar al modelo Log-Distance.

Figura 6.1: Gateaway MTCDDTIP-266A

Se están realizando pruebas con el fin de obtener datos empíricos sobre la cobertura que puede dar un GW, para ello, se hace uso de un tester LoRaWAN modelo *Adeunis LoRaWAN 868 Field Test*.

Figura 6.2: Adeunis LoRaWAN 868 Field Test

Dicho *tester* permite una configuración para realizar mediciones periódicas de la calidad del canal (potencia de enlace ascendente y descendente, SNIR, etc), además de otros parámetros que se pueden ver en la Figura 6.3.

Bytes	1	2	3 to 6				7 to 10				11	12	13	14-15		16	17
Description	Status	Temperature	GPS Latitude				GPS Longitude				GPS Quality	UL Counter	DL Counter	Battery level		RSSI	SNR
Example	BF	1B	45	15	96	90	00	53	45	00	27	20	20	0F	C9	52	07

Figura 6.3: Datos de la trama del tester LoRaWAN

En las Tablas 6.1 y 6.2 se pueden observar tanto la configuración del GW como del tester:

Parámetro	Valor
Altura	45m
Ganancia	8dBi
Cable	LMR-400 - 1m
Modelo Gateaway	MTCDDTIP-266A

Tabla 6.1: Datos del gateaway usado

Parámetro	Valor
BW	125KHz
SF	12
CR	4/5
Potencia	25mW ≈14dBm

Tabla 6.2: Parámetros de configuración del *tester* LoRaWAN

6.2. Despliegue de precisión

Para realizar comparaciones con el framework desarrollado en este trabajo fin de grado, se ha realizado un pequeño despliegue con instrumentos de alta precisión.

Se ha colocado un generador de señales (Rohde&Schwarz SMB100A) en la azotea del Edificio de Laboratorios de Investigación de la Universidad Politécnica de Cartagena (Figura 6.4). El generador se ha configurado para transmitir una señal a 0.025W (14dBm) de potencia en la frecuencia en la que hacemos uso de LoRa, es decir 868MHz (Figura 6.5).

Figura 6.4: Ubicación del gateaway

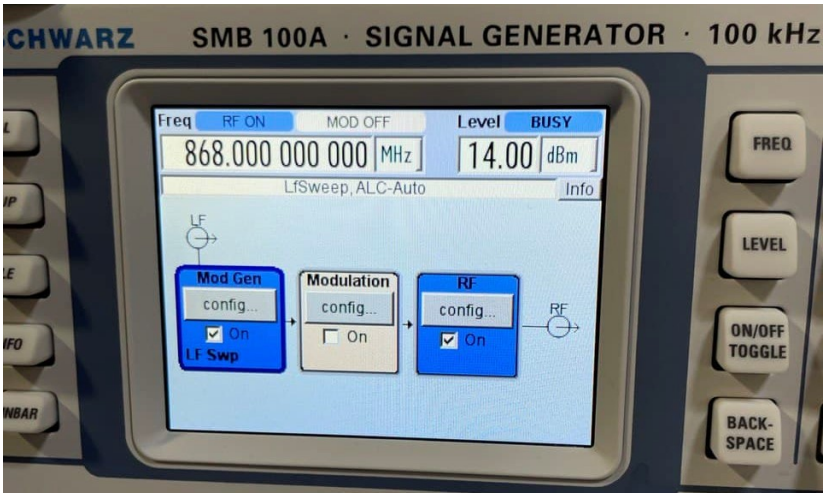

Figura 6.5: Generador de señales

Figura 6.6: Montaje en la azotea del ELDI

Para comprobar la potencia recibida en los puntos que se verán en el siguiente capítulo, se emplea un analizador de espectros Rohde&Schwarz FSH3 (Figura 6.7)

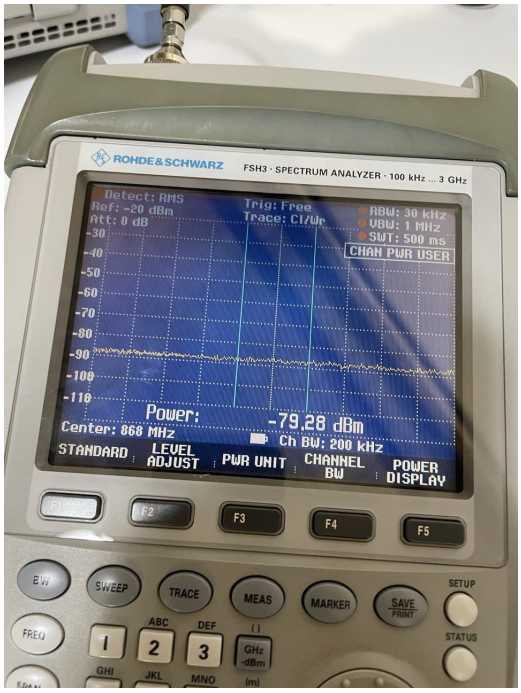

Figura 6.7: Analizador de espectros

El analizador de espectros está configurado para visualizar la frecuencia a la que emite el generador de espectros (868MHz) con un ancho de banda de 200kHz.

Se han empleado dos antenas omnidireccionales con polarización vertical (*QOM-SL-0.8-40-K-SG-L*), las cuáles soportan frecuencias desde 0.8 hasta 40 GHz con una ganancia de -2.2 a 6.9 dBi.

Figura 6.8: Antena empleada en el despliegue

CAPÍTULO 7

EVALUACIÓN

7.1. Descripción de las pruebas

Se ha realizado una pequeña campaña de recogida de datos en forma de intensidad de señal recibida en las inmediaciones del despliegue, considerando el escenario objeto de referencia en el trabajo fin de grado (Despliegue 6.2). Los puntos donde se han realizado medidas se pueden ver en la Figura 7.1. Como se puede observar, se han tomado algunas medidas en puntos donde no hay linea de visión directa con la antena. Con esto podremos ver si el trazado de rayos funciona correctamente.

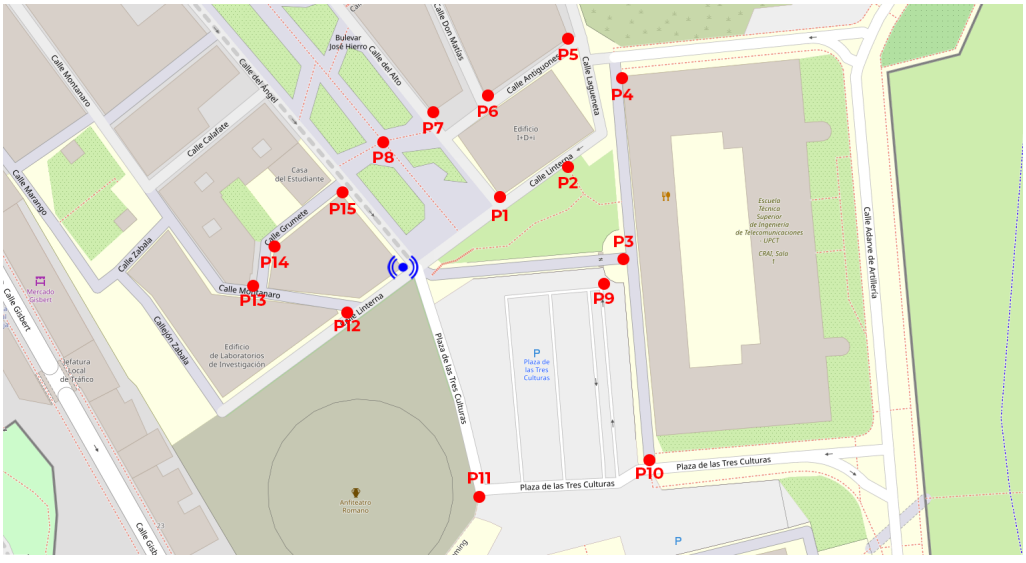

Figura 7.1: Mapa de posiciones donde se ha realizado el despliegue de pruebas

Para la realización de este trabajo, esta recogida de datos era necesaria con el fin de poder validar los resultados obtenidos con el framework. Si este sistema tuviera una precisión adecuada, a la hora de realizar una despliegue real se podría ahorrar un estudio exhaustivo de campo, ahorrando así tiempo y dinero.

En Unity, se coloca manualmente cada punto para obtener las coordenadas que se pasan a OMNeT++ en el archivo de inicialización de la simulación (Tabla 7.1).

PUNTO	X (m)	Y (m)	Z (m)
Emisor	1547,14	40,69	620,8
P1	1585,34	29	644,6
P2	1607,4	29	653,4
P3	1618,5	29	629,3
P4 (NLOS)	1625,9	29	693,29
P5 (NLOS)	1608,2	28,6	702,8
P6 (NLOS)	1578,9	24,5	681
P7	1560,2	23,1	674,5
P8	1547,9	21,82	666,1
P9	1620,13	22,88	620,94
P10	1632,9	21,67	560,55
P11	1572,9	21,67	560,55
P12 (NLOS)	1528,8	25	605
P13 (NLOS)	1498,8	25	611,9
P14(NLOS)	1504,2	24,1	626,723
P15 (NLOS)	1528,32	19,94	646,39

Tabla 7.1: Coordenadas de los puntos empleados en la simulación

Los puntos marcados con NLOS son puntos donde no existe visión directa con la antena que se encuentra conectada al generador de señales.

A continuación se realizan dos archivos de inicialización de la simulación, uno de ellos está preparado para ejecutar Opal y el segundo simulará tanto con el modelo Okumura-Hata como con el modelo Log-Distance. Estos archivos se pueden ver en el Anexo I y II.

Para el análisis de datos, se realizarán 20 simulaciones con la configuración de Opal y en cada simulación se moverá el receptor 30 centímetros en cada eje de forma aleatoria y, posteriormente, se realizará la media de las potencias recibidas. Este proceso no se realiza en las simulaciones con los otros modelos porque el resultado esperado es determinista y en el modelo no se considera el entorno de transmisión. En el caso del trazado de rayos, el número de haces generados y la presencia de edificios hace

necesario considerar un valor medio sobre pequeños desplazamientos del nodo, para así evitar considerar valores atípicos.

Como se puede ver en los archivos de inicialización de la simulación, hay 15 GWs y un solo nodo, esto es debido a que las medidas experimentales se han tomado de esa forma, ya que el emisor (nodo) no se podía mover de su posición.

Para obtener los parámetros adecuados para el ajuste el modelo Log-Distance, se han empleado los datos recogidos de la red LoRaWAN desplegada por el GIRTEL. Se han empleado diversos scripts en Python para procesar los datos guardados en una base de datos InfluxDB. Se han recogido 490 medidas abarcando la superficie que se puede ver en la Figura 7.2

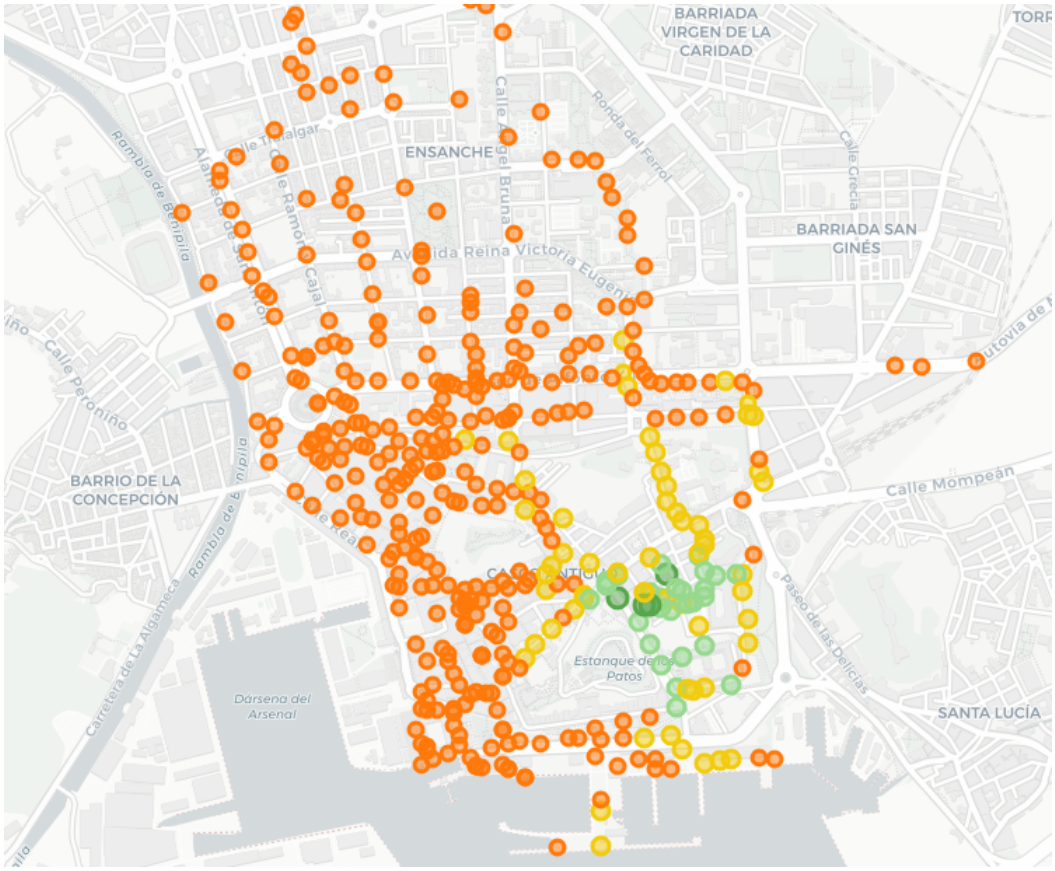

Figura 7.2: Datos recogidos en InfluxDB

La posición GPS se guarda en coordenadas geográficas (EPSG:4326 - WGS84), por lo que para insertar los nodos de forma automática en la simulación se deberá realizar una conversión de estar coordenadas al Sistema de coordenadas universal transversal de Mercator (UTM). Este sistema, en vez de emplear latitud y longitud, expresa sus magnitudes en metros, en concreto se realizará una conversión a EPSG:32630.

Todo este proceso se realiza mediante un *script* en Python siguiendo los siguientes pasos:

1. Las coordenadas geográficas se guardan en formato DMS (grados, minutos, segundos) por lo que se tendrá que convertir a formato DD (decimal). Para ello se han realizado tres funciones. (Figura 7.3)

```
def dms2dd(degrees, minutes, direction):
    dd = float(degrees) + float(minutes)/60;
    if direction == 'E' or direction == 'N':
        dd *= 1
    else :
        dd *= -1
    return dd;

def dd2dms(deg):
    d = int(deg)
    md = abs(deg - d) * 60
    m = int(md)
    sd = (md - m) * 60
    return [d, m, sd]

def parse_dms(dms):
    parts = re.split('[^\d\w]', dms)
    lat = dms2dd(parts[1], float(parts[2] + '.' + parts[3]) , parts[0])
    return (lat)
```

Figura 7.3: Funciones de conversión DMS a DD

2. Una vez convertidas las coordenadas a DD, la librería GeoPandas se encargará de obtener las distancias UTM. Posteriormente se aplicará un offset (-676768.14, -4162885.72) para poder introducir los nodos en Unity para detectar errores de ubicación de los nodos. Este proceso también genera un archivo que contiene las posiciones x e y de cada nodo. (Figura 7.4)

```
s = gpd.GeoSeries([Point(x,y) for x, y in zip(latlon['Longitude'], latlon['Latitude'])])
geo_df = gpd.GeoDataFrame(latlon[['Time']], geometry=s)
geo_df.set_crs(epsg=4326, inplace=True)
geo_df = geo_df.to_crs(epsg=32630)
```

Figura 7.4: Código de conversión DMS a UTM

3. Unity leerá el archivo con las posiciones de los nodos y los dibujará en el modelo 3D de la ciudad. Además de ello imprimirá por pantalla los nodos que se encuentran en el interior de estructuras y la altura de los nodos.

4. Ahora se volverá a ejecutar el mismo *script* pero añadiendo los nodos que se desean eliminar y las alturas del resto de nodos (solo se tendrá que copiar la lista que se imprime por pantalla en Unity). (Figura 7.5)

```
alturas = ['ALTURAS DE LOS NODOS']
geo.drop(['NODOS A ELIMINAR'], inplace=True)
geo.reset_index(inplace = True)
```

Figura 7.5: Código de eliminación de nodos erróneos e inserción de alturas

5. A continuación calcula la distancia de cada nodo con el GW. La distancia y la potencia recibida se exportan a un archivo csv.

6. El script [42] lee el archivo csv e imprimirá por pantalla los datos que necesitamos para realizar la aproximación más adecuada del modelo Log-Distance para este caso. (Figura 7.6)

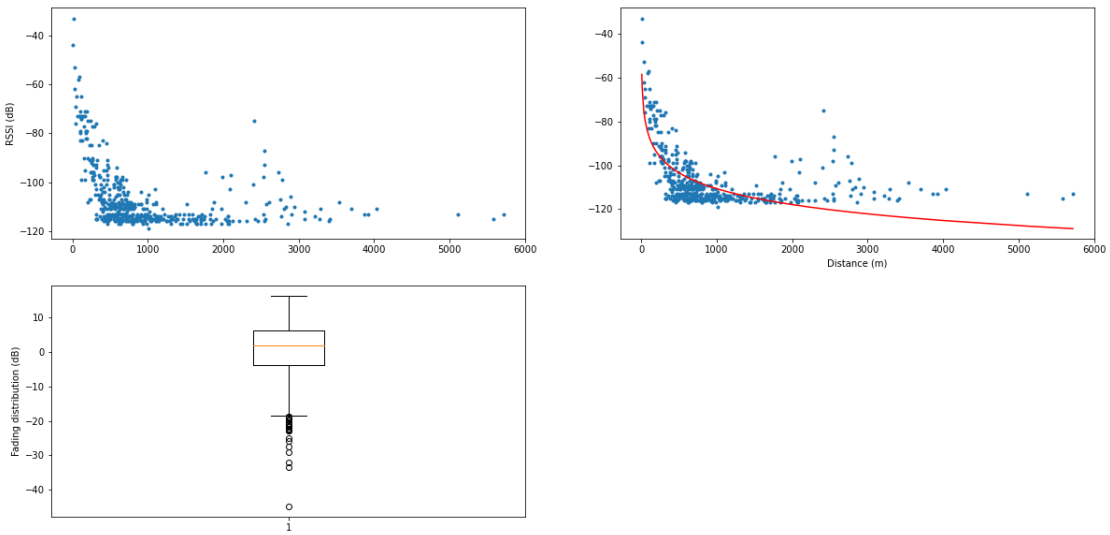

Figura 7.6: Gráficas obtenidas mediante el script de aproximación para Log-Distance

n	$P_L(d_0)$	d_0	χ_σ
2.41	-90.85 dBm	150.54 m	$N(14.32, 8.868)$

Tabla 7.2: Parámetros para Log-Distance

7.2. Resultados empíricos

En cada posición de la Figura 7.1 se toman cincos medidas para posteriormente realizar una media, de donde se obtienen los siguientes valores de potencia para cada punto:

PUNTO	MEDIDA EMPÍRICA (dBm)
P1	-54,3
P2	-56,4
P3	-60,8
P4 (NLOS)	-77,8
P5 (NLOS)	-76,6
P6 (NLOS)	-67,5
P7	-61,4
P8	-77,5
P9	-61,3
P10	-66,8
P11	-61,6
P12 (NLOS)	-75,4
P13 (NLOS)	-75,4
P14(NLOS)	-78,2
P15 (NLOS)	-74,3

Tabla 7.3: Medidas empíricas

7.3. Resultados en la simulación

Después de realizar las simulaciones y realizar los cálculos de promedio para los valores obtenidos mediante Ray-Tracing, se obtienen los siguientes valores (Tabla 7.4), donde RMSE es el error cuadrático medio calculado mediante la Ecuación 7.1

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}\Big(P_i - O_i\Big)^2}{n}} \tag{7.1}$$

PUNTO	EXPERIMENTAL	LOG-DISTANCE	DIF	OKUMURA-HATA	DIF	RAY-TRACING	DIF
P1	-54,3	-78,88	24,58	-66	11,7	-56,18	1,88
P2	-56,4	-83,09	26,69	-72,65	16,25	-61,43	5,03
P3	-60,8	-83,58	22,78	-73,42	12,62	-59,31	1,49
P4 (NLOS)	-77,8	-87,67	9,87	-79,89	2,09	-83,82	6,02
P5 (NLOS)	-76,6	-87,2	10,6	-79,15	2,55	-82,16	5,56
P6 (NLOS)	-67,5	-83,16	15,66	-72,76	5,26	-70,18	2,68
P7	-61,4	-81,2	19,8	-69,66	8,26	-66,36	4,96
P8	-77,5	-79,45	1,95	-66,89	10,61	-67,56	9,94
P9	-61,3	-83,91	22,61	-73,94	12,64	-71,42	10,12
P10	-66,8	-87,56	20,76	-79,71	12,91	-59,97	6,83
P11	-61,6	-82,9	21,3	-72,34	10,74	-61,06	0,54
P12 (NLOS)	-75,4	-73,89	1,51	-58,11	17,29	-75,15	0,25
P13 (NLOS)	-75,4	-79,97	4,57	-67,72	7,68	-90,2	14,8
P14 (NLOS)	-78,2	-78,86	0,66	-65,97	12,23	-86,29	8,09
P15 (NLOS)	-74,3	-73,55	0,75	-57,58	16,72	-67,9	6,4
		PROMEDIO	13,61		10,64		5,64
		RMSE	16,55		11,57		6,85

Tabla 7.4: Comparación de medidas

Log-Distance es el modelo más simple, ya que está basado principalmente en la propagación de señales en espacio abierto. A pesar de que está adaptado mediante aproximaciones al escenario usado, mediante los parámetros vistos en la Tabla 7.2, este modelo no es capaz de calcular el nivel de señal ante la presencia de obstrucciones como edificios.

Okumura-Hata es un modelo más completo, donde se incorporan las características para adaptarlo a las condiciones de simulación, como es el caso de entornos urbanos para ciudades de un determinado perfil. Además, considera la información sobre la altura del transmisor y receptor. Todo esto hace que los resultados se acerquen más a las medidas reales que en el caso de Log-Distance. Sin embargo, la presencia de edificios en puntos concretos presenta un problema para el modelo, pues trata igual todos los puntos a una determinada distancia del emisor.

La solución propuesta considera la geometría del escenario de pruebas, ofreciendo valores mucho más cercanos a la realidad. La determinación de la pérdida de señal se realiza considerando los obstáculos en el camino de visión directa. Para ello, se generan rayos sintéticos que realizan un trazado sobre el escenario 3D. Se puede ver que los resultados medios mejoran en gran medida los obtenidos con modelos tradicionales, obteniendo una media de error de 5.64 dBm, y un RMSE de 6.85 dBm. Estos valores

disminuyen un 50 % el error cometido en el caso de Okumura-Hata, y en un 65 % con respecto a los obtenidos con Log-Distance.

CAPÍTULO 8

CONCLUSIONES

En este trabajo se presenta un sistema para realizar simulaciones con tecnología LoRaWAN en entornos urbanos de forma realista mediante la integración del framework FloRa de OMNeT++, empleado para la simulación de redes LoRa, con Opal, un simulador de propagación de radio frecuencias basado en el trazado de rayos (Ray-Tracing) en GPU. También se ha realizado un pequeño despliegue para realizar medidas experimentales para comparar estos valores con los obtenidos mediante simulación.

Según los resultados obtenidos, se puede observar que el trazado de rayos tiene un gran potencial en este campo, ya que se pueden realizar simulaciones que tengan en cuenta el entorno sobre el que se ejecutan para así obtener valores lo más cercanos a la realidad posible, en términos generales, el framework tiene una mejora de hasta un 50 % en el mejor caso de los modelos que se pueden ver en la literatura.

Se va a seguir trabajando sobre este framework para ajustarlo aún más a la realidad y poder realizar simulaciones de cientos de nodos simultáneamente.

A su vez se permitirá que los nodos se muevan empleando el framework Veneris, simulando así una unidad a bordo de un vehículo, esto se empleará para simular la cobertura en redes vehiculares.

BIBLIOGRAFÍA

[1] Jetmir Haxhibeqiri y col. «A Survey of LoRaWAN for IoT: From Technology to Application». En: *Sensors* 18.11 (2018). ISSN: 1424-8220. DOI: 10.3390/s18113995. URL: https://www.mdpi.com/1424-8220/18/11/3995.

[2] J. M. Marais, A. M. Abu-Mahfouz y G. P. Hancke. «A Review of LoRaWAN Simulators: Design Requirements and Limitations». En: *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*. 2019, págs. 1-6. DOI: 10.1109/IMITEC45504.2019.9015882.

[3] Martin C. Bor y col. «Do LoRa Low-Power Wide-Area Networks Scale?» En: *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. MSWiM '16. Malta, Malta: Association for Computing Machinery, 2016, págs. 59-67. ISBN: 9781450345026. DOI: 10.1145/2988287.2989163. URL: https://doi.org/10.1145/2988287.2989163.

[4] Thiemo Voigt y col. «Mitigating Inter-Network Interference in LoRa Networks». En: *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*. EWSN '17. Uppsala, Sweden: Junction Publishing, 2017, págs. 323-328. ISBN: 9780994988614.

[5] A. Pop y col. «Does Bidirectional Traffic Do More Harm Than Good in LoRaWAN Based LPWA Networks?» En: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 2017, págs. 1-6. DOI: 10.1109/GLOCOM.2017.8254509.

[6] D. Magrin, M. Centenaro y L. Vangelista. «Performance evaluation of LoRa networks in a smart city scenario». En: *2017 IEEE International Conference on Communications (ICC)*. 2017, págs. 1-7. DOI: 10.1109/ICC.2017.7996384.

[7] M. Capuzzo, D. Magrin y A. Zanella. «Confirmed traffic in LoRaWAN: Pitfalls and countermeasures». En: *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. 2018, págs. 1-7. DOI: 10.23919/MedHocNet.2018.8407095.

[8] F. Van den Abeele y col. «Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3». En: *IEEE Internet of Things Journal* 4.6 (2017), págs. 2186-2198. DOI: 10.1109/JIOT.2017.2768498.

[9] K. Q. Abdelfadeel y col. «$FREE$—Fine-Grained Scheduling for Reliable and Energy-Efficient Data Collection in LoRaWAN». En: *IEEE Internet of Things Journal* 7.1 (2020), págs. 669-683. DOI: 10.1109/JIOT.2019.2949918.

[10] G. Callebaut, G. Ottoy y L. van der Perre. «Cross-Layer Framework and Optimization for Efficient Use of the Energy Budget of IoT Nodes». En: *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. 2019, págs. 1-6. DOI: 10.1109/WCNC.2019.8885739.

[11] K. P. Maurya y A. A. Kherani. «A Modular Hybrid Simulator for LoRaWAN». En: *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. 2020, págs. 1-4. DOI: 10.1109/ANTS50601.2020.9342777.

[12] Riccardo Marini y col. «LoRaWANSim: A Flexible Simulator for LoRaWAN Networks». En: *Sensors* 21.3 (2021). ISSN: 1424-8220. DOI: 10.3390/s21030695. URL: https://www.mdpi.com/1424-8220/21/3/695.

[13] M. Slabicki, G. Premsankar y M. Di Francesco. «Adaptive configuration of lora networks for dense IoT deployments». En: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. 2018, págs. 1-9. DOI: 10.1109/NOMS.2018.8406255.

[14] Jorge Gallego-Madrid y col. «Enhancing Extensive and Remote LoRa Deployments through MEC-Powered Drone Gateways». En: *Sensors* 20.15 (2020). ISSN: 1424-8220. DOI: 10.3390/s20154109. URL: https://www.mdpi.com/1424-8220/20/15/4109.

[15] Jose-Manuel Martinez-Caro y Maria-Dolores Cano. «A novel holistic approach for performance evaluation in Internet of Things». En: *International Journal of Communication Systems* 34.2 (2021). e4454 dac.4454, e4454. DOI: https://doi.org/10.1002/dac.4454. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/dac.4454. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4454.

[16] N. S. Bezerra y col. «Propagation Model Evaluation for LoRaWAN: Planning Tool Versus Real Case Scenario». En: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. 2019, págs. 1-6. DOI: 10.1109/WF-IoT.2019.8767299.

[17] N. A. B. Masadan, M. H. Habaebi y S. H. Yusoff. «LoRa LPWAN Propagation Channel Modelling in IIUM Campus». En: *2018 7th International Conference on Computer and Communication Engineering (ICCCE)*. 2018, págs. 14-19. DOI: 10.1109/ICCCE.2018.8539327.

[18] O. Dieng, C. Pham y O. Thiare. «Comparing and Adapting Propagation Models for LoRa Networks». En: *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 2020, págs. 1-7. DOI: 10.1109/WiMob50308.2020.9253410.

[19] R. El Chall, S. Lahoud y M. El Helou. «LoRaWAN Network: Radio Propagation Models and Performance Evaluation in Various Environments in Lebanon». En: *IEEE Internet of Things Journal* 6.2 (2019), págs. 2366-2378. DOI: 10.1109/JIOT.2019.2906838.

[20] Salaheddin Hosseinzadeh y col. «A Neural Network Propagation Model for LoRaWAN and Critical Analysis with Real-World Measurements». En: *Big Data and Cognitive Computing* 1.1 (2017). ISSN: 2504-2289. DOI: 10.3390/bdcc1010007. URL: https://www.mdpi.com/2504-2289/1/1/7.

[21] Salaheddin Hosseinzadeh y col. «An Adaptive Neuro-Fuzzy Propagation Model for LoRaWAN». En: *Applied System Innovation* 2.1 (2019). ISSN: 2571-5577. DOI: 10.3390/asi2010010. URL: https://www.mdpi.com/2571-5577/2/1/10.

[22] Paula Fraga-Lamas y col. «Design and Experimental Validation of a LoRaWAN Fog Computing Based Architecture for IoT Enabled Smart Campus Applications». En: *Sensors* 19.15 (2019). ISSN: 1424-8220. DOI: 10.3390/s19153287. URL: https://www.mdpi.com/1424-8220/19/15/3287.

[23] Cloud-RF. *Cloud-RF*. URL: https://cloudrf.com/.

[24] Atoll. *Atoll LPWA/IoT*. URL: https://www.forsk.com/atoll-lpwaiot.

[25] Siradel. *S_IoT*. URL: https://www.siradel.com/solutions/software/s_iot/.

[26] Roger Coudé. *Radio Mobile*. URL: http://www.ve2dbe.com/english1.html.

[27] Fei Gu y col. «Survey of the low power wide area network technologies». En: *Journal of Network and Computer Applications* 149 (2020), pág. 102459. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2019.102459. URL: https://www.sciencedirect.com/science/article/pii/S1084804519303194.

[28] OMNeT++. URL: https://omnetpp.org/.

[29] Framework INET. URL: https://inet.omnetpp.org/.

[30] Unity. URL: https://unity.com/.

[31] Veneris. URL: http://pcacribia.upct.es/veneris/.

[32] Opal. URL: https://gitlab.com/esteban.egea/opal.

[33] QGis. URL: https://www.qgis.org/.

[34] Dem2Mesh. URL: https://github.com/OpenDroneMap/dem2mesh.

[35] MeshLab. URL: https://www.meshlab.net/.

[36] UIT-R. *UIT-R P.2040-1*. URL: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.2040-1-201507-I!!PDF-S.pdf.

[37] Andreas F. Molisch. *Wireless Communications*. Wiley, 2010.

[38] Y. Okumura. «Field strength and its variability in VHF and UHF land-mobile radio service». En: *Rev. Electr. Commun. Lab.* 16 (1968), págs. 825-873. URL: https://ci.nii.ac.jp/naid/10010001461/en/.

[39] M. Hata. «Empirical formula for propagation loss in land mobile radio services». En: *IEEE Transactions on Vehicular Technology* 29.3 (1980), págs. 317-325. DOI: 10.1109/T-VT.1980.23859.

[40] Esteban Egea-Lopez. *Veneris Scenario Builder*. URL: http://pcacribia.upct.es/veneris/scenario.

[41] CNIG. *CNIG*. URL: https://centrodedescargas.cnig.es/CentroDescargas/index.jsp.

[42] Salaheddin Hosseinzadeh. *Log-distance propagation analysis + opotimization and data analysis*. URL: https://es.mathworks.com/matlabcentral/fileexchange/70097-python-matlab-optimization-for-log-distance-model-example?s_tid=prof_contriblnk&s_tid=mwa_osa_a.

Anexos

ANEXO A

ANEXO I: CONFIGURACIÓN PARA OPAL

```
[General]
network = CartagenaOPAL
rng-class = "cMersenneTwister"
repeat = 20

**.result-recording-modes = all
**.numberOfGateways = 15
**.loRaGW[*].numUdpApps = 1
**.loRaGW[*].packetForwarder.localPort = 2000
**.loRaGW[*].packetForwarder.destPort = 1000
**.loRaGW[*].packetForwarder.destAddresses = "networkServer"
**.loRaGW[*].packetForwarder.indexNumber = 0
**.loRaGW[*].receptionRadius = 1
**.LoRaGWNic.radio.antennaType = "OpalLinearPolarizationAntenna"
**.LoRaNic.radio.antennaType = "OpalLinearPolarizationAntenna"


**.LoRaGWNic.radio.iAmGateway = true
**.loRaGW[*].**.initFromDisplayString = false

**.loRaGW[0].**.initialX = 1585.34m  + uniform(-0.3m, 0.3m)
**.loRaGW[0].**.initialY = 29m  + uniform(-0.3m, 0.3m)
**.loRaGW[0].**.initialZ = 644.6m  + uniform(-0.3m, 0.3m)

**.loRaGW[1].**.initialX = 1607.4m  + uniform(-0.3m, 0.3m)
**.loRaGW[1].**.initialY = 29m + uniform(-0.3m, 0.3m)
**.loRaGW[1].**.initialZ = 653.4m  + uniform(-0.3m, 0.3m)

**.loRaGW[2].**.initialX = 1618.5m  + uniform(-0.3m, 0.3m)
**.loRaGW[2].**.initialY = 29m + uniform(-0.3m, 0.3m)
**.loRaGW[2].**.initialZ = 629.3m  + uniform(-0.3m, 0.3m)

**.loRaGW[3].**.initialX = 1625.9m  + uniform(-0.3m, 0.3m)
**.loRaGW[3].**.initialY = 29m + uniform(-0.3m, 0.3m)
**.loRaGW[3].**.initialZ = 693.29m  + uniform(-0.3m, 0.3m)
```

```
**.loRaGW[4].**.initialX = 1608.2m  + uniform(-0.3m, 0.3m)
**.loRaGW[4].**.initialY = 28.6m + uniform(-0.3m, 0.3m)
**.loRaGW[4].**.initialZ = 702.8m  + uniform(-0.3m, 0.3m)

**.loRaGW[5].**.initialX = 1578.9m  + uniform(-0.3m, 0.3m)
**.loRaGW[5].**.initialY = 24.5m + uniform(-0.3m, 0.3m)
**.loRaGW[5].**.initialZ = 681m  + uniform(-0.3m, 0.3m)

**.loRaGW[6].**.initialX = 1560.2m  + uniform(-0.3m, 0.3m)
**.loRaGW[6].**.initialY = 23.1m + uniform(-0.3m, 0.3m)
**.loRaGW[6].**.initialZ = 674.5m  + uniform(-0.3m, 0.3m)

**.loRaGW[7].**.initialX = 1547.9m  + uniform(-0.3m, 0.3m)
**.loRaGW[7].**.initialY = 21.82m + uniform(-0.3m, 0.3m)
**.loRaGW[7].**.initialZ = 666.1m  + uniform(-0.3m, 0.3m)

**.loRaGW[8].**.initialX = 1620.13m  + uniform(-0.3m, 0.3m)
**.loRaGW[8].**.initialY = 22.88m + uniform(-0.3m, 0.3m)
**.loRaGW[8].**.initialZ = 620.94m  + uniform(-0.3m, 0.3m)

**.loRaGW[9].**.initialX = 1632.9m  + uniform(-0.3m, 0.3m)
**.loRaGW[9].**.initialY = 21.67m + uniform(-0.3m, 0.3m)
**.loRaGW[9].**.initialZ = 560.55m  + uniform(-0.3m, 0.3m)

**.loRaGW[10].**.initialX = 1572.9m  + uniform(-0.3m, 0.3m)
**.loRaGW[10].**.initialY = 21.67m + uniform(-0.3m, 0.3m)
**.loRaGW[10].**.initialZ = 560.55m  + uniform(-0.3m, 0.3m)

**.loRaGW[11].**.initialX = 1528.8m  + uniform(-0.3m, 0.3m)
**.loRaGW[11].**.initialY = 25m + uniform(-0.3m, 0.3m)
**.loRaGW[11].**.initialZ = 605m  + uniform(-0.3m, 0.3m)

**.loRaGW[12].**.initialX = 1498.8m  + uniform(-0.3m, 0.3m)
**.loRaGW[12].**.initialY = 25m + uniform(-0.3m, 0.3m)
**.loRaGW[12].**.initialZ = 611.9m  + uniform(-0.3m, 0.3m)

**.loRaGW[13].**.initialX = 1504.2m  + uniform(-0.3m, 0.3m)
**.loRaGW[13].**.initialY = 24.1m + uniform(-0.3m, 0.3m)
**.loRaGW[13].**.initialZ = 626.723m  + uniform(-0.3m, 0.3m)

**.loRaGW[14].**.initialX = 1528.32m  + uniform(-0.3m, 0.3m)
**.loRaGW[14].**.initialY = 19.94m + uniform(-0.3m, 0.3m)
**.loRaGW[14].**.initialZ = 646.39m  + uniform(-0.3m, 0.3m)


**.networkServer.numUdpApps = 1
**.networkServer.**.evaluateADRinServer = false
**.networkServer.udpApp[0].typename = "NetworkServerApp"
**.networkServer.udpApp[0].destAddresses = "loRaGW[*]"
**.networkServer.udpApp[0].destPort = 2000
**.networkServer.udpApp[0].localPort = 1000
**.networkServer.udpApp[0].adrMethod = \$\{"avg"\}

**.numberOfPacketsToSend = 1
sim-time-limit = 1d
warmup-period = 3h
simtime-resolution = -11
```

```
**.timeToFirstPacket = exponential(100s)
**.timeToNextPacket = exponential(100s)
**.alohaChannelModel = false

**.numberOfNodes = 1

**.loRaNodes[*].**.initFromDisplayString = false
**.loRaNodes[*].**.evaluateADRinNode = false
**.loRaNodes[*].**initialLoRaSF = 12
**.loRaNodes[*].**initialLoRaBW = 125 kHz
**.loRaNodes[*].**initialLoRaCR = 5
**.loRaNodes[*].**initialLoRaTP = 14 dBm

**.loRaNodes[0].**.initialX = 1547.14m
**.loRaNodes[0].**.initialY = 40.69m
**.loRaNodes[0].**.initialZ = 620.8m

**.loRaNodes[*].LoRaNic.radio.energyConsumerType = "LoRaEnergyConsumer"
**.loRaNodes[*].**.energySourceModule = "IdealEpEnergyStorage"
**.loRaNodes[*].LoRaNic.radio.energyConsumer.configFile = xmldoc("
    energyConsumptionParameters.xml")

**.sigma = 0
**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 3000m
**.constraintAreaMaxY = 3000m
**.constraintAreaMaxZ = 3000m
CartagenaOPAL.**.radio.separateTransmissionParts = false
CartagenaOPAL.**.radio.separateReceptionParts = false

**.delayer.config = xmldoc("cloudDelays.xml")
**.radio.radioMediumModule = "OpalLoRaRadioMedium"
**.minInterferenceTime = 0s
**.displayAddresses = false

**.OpalLoRaRadioMedium.carrierFrequency = 868MHz
**.mediumLimitCache.carrierFrequency = 868MHz
**.radio.carrierFrequency = 868MHz

**.OpalLoRaRadioMedium.maxNumberOfReflections = 30
**.OpalLoRaRadioMedium.enableFastMath = false
**.OpalLoRaRadioMedium.useExactSpeedOfLight = true
**.OpalLoRaRadioMedium.loadMeshesFromFile = true
**.OpalLoRaRadioMedium.meshesPath = "/home/anrunie/Documentos/TFG/meshes"
**.OpalLoRaRadioMedium.meshesFileList = "names.txt"
**.OpalLoRaRadioMedium.azimuthDelta = 0.1
**.OpalLoRaRadioMedium.elevationDelta = 0.1
**.OpalLoRaRadioMedium.receptionRadius = 1m
**.OpalLoRaRadioMedium.isStatic = true

**.OpalLoRaRadioMedium.useDiffraction = true
**.OpalLoRaRadioMedium.loadEdgesFromFile = true
**.OpalLoRaRadioMedium.edgesPath = "/home/anrunie/Documentos/TFG/meshes"

**.OpalLoRaRadioMedium.edgesFileList = "edges.txt"
**.OpalLoRaRadioMedium.useAntennaGain = true
**.OpalLoRaRadioMedium.antennaGainFilePaths = "/home/anrunie/Documentos/TFG/gainsPath.
```

```
    txt"
*.loRaNodes[*].LoRaNic.radio.antenna.useAntennaGain = true
*.loRaNodes[*].LoRaNic.radio.antenna.gainFilePath = "/home/anrunie/Documentos/TFG/
    gain11467.txt"
*.loRaGW[*].LoRaGWNic.radio.antenna.useAntennaGain = true
*.loRaGW[*].LoRaGWNic.radio.antenna.gainFilePath = "/home/anrunie/Documentos/TFG/
    gain17514.txt"
```

ANEXO B

ANEXO II: CONFIGURACIÓN PARA OKUMURA-HATA Y LOG-DISTANCE

```
[General]
network = Cartagena
rng-class = "cMersenneTwister"
repeat = 1

**.result-recording-modes = all
**.numberOfGateways = 15
**.loRaGW[*].numUdpApps = 1
**.loRaGW[*].packetForwarder.localPort = 2000
**.loRaGW[*].packetForwarder.destPort = 1000
**.loRaGW[*].packetForwarder.destAddresses = "networkServer"
**.loRaGW[*].packetForwarder.indexNumber = 0

**.LoRaGWNic.radio.antennaType = "IsotropicAntenna"
**.LoRaNic.radio.antennaType = "IsotropicAntenna"

**.LoRaGWNic.radio.iAmGateway = true
**.loRaGW[*].**.initFromDisplayString = false

**.loRaGW[0].**.initialX = 1585.34m
**.loRaGW[0].**.initialY = 29m
**.loRaGW[0].**.initialZ = 644.6m

**.loRaGW[1].**.initialX = 1607.4m
**.loRaGW[1].**.initialY = 29m
**.loRaGW[1].**.initialZ = 653.4m

**.loRaGW[2].**.initialX = 1618.5m
**.loRaGW[2].**.initialY = 29m
**.loRaGW[2].**.initialZ = 629.3m

**.loRaGW[3].**.initialX = 1625.9m
**.loRaGW[3].**.initialY = 29m
**.loRaGW[3].**.initialZ = 693.29m
```

```
**.loRaGW[4].**.initialX = 1608.2m
**.loRaGW[4].**.initialY = 28.6m
**.loRaGW[4].**.initialZ = 702.8m

**.loRaGW[5].**.initialX = 1578.9m
**.loRaGW[5].**.initialY = 24.5m
**.loRaGW[5].**.initialZ = 681m

**.loRaGW[6].**.initialX = 1560.2m
**.loRaGW[6].**.initialY = 23.1m
**.loRaGW[6].**.initialZ = 674.5m

**.loRaGW[7].**.initialX = 1547.9m
**.loRaGW[7].**.initialY = 21.82m
**.loRaGW[7].**.initialZ = 666.1m

**.loRaGW[8].**.initialX = 1620.13m
**.loRaGW[8].**.initialY = 22.88m
**.loRaGW[8].**.initialZ = 620.94m

**.loRaGW[9].**.initialX = 1632.9m
**.loRaGW[9].**.initialY = 21.67m
**.loRaGW[9].**.initialZ = 560.55m

**.loRaGW[10].**.initialX = 1572.9m
**.loRaGW[10].**.initialY = 21.67m
**.loRaGW[10].**.initialZ = 560.55m

**.loRaGW[11].**.initialX = 1528.8m
**.loRaGW[11].**.initialY = 25m
**.loRaGW[11].**.initialZ = 605m

**.loRaGW[12].**.initialX = 1498.8m
**.loRaGW[12].**.initialY = 25m
**.loRaGW[12].**.initialZ = 611.9m

**.loRaGW[13].**.initialX = 1504.2m
**.loRaGW[13].**.initialY = 24.1m
**.loRaGW[13].**.initialZ = 626.723m

**.loRaGW[14].**.initialX = 1528.23m
**.loRaGW[14].**.initialY = 20.94m
**.loRaGW[14].**.initialZ = 626.52m


**.networkServer.numUdpApps = 1
**.networkServer.**.evaluateADRinServer = false
**.networkServer.udpApp[0].typename = "NetworkServerApp"
**.networkServer.udpApp[0].destAddresses = "loRaGW[*]"
**.networkServer.udpApp[0].destPort = 2000
**.networkServer.udpApp[0].localPort = 1000
**.networkServer.udpApp[0].adrMethod = \$\{"avg"\}

**.numberOfPacketsToSend = 1 #0 means infinite number of packets
sim-time-limit = 1d
warmup-period = 3h
simtime-resolution = -11
```

```
**.timeToFirstPacket = exponential(100s)
**.timeToNextPacket = exponential(100s)
**.alohaChannelModel = false

**.numberOfNodes = 1

**.loRaNodes[*].**.initFromDisplayString = false
**.loRaNodes[*].**.evaluateADRinNode = false
**.loRaNodes[*].**initialLoRaSF = 12
**.loRaNodes[*].**initialLoRaBW = 125 kHz
**.loRaNodes[*].**initialLoRaCR = 5
**.loRaNodes[*].**initialLoRaTP = 14 dBm

**.loRaNodes[0].**.initialX = 1547.14m
**.loRaNodes[0].**.initialY = 40.69m
**.loRaNodes[0].**.initialZ = 620.8m

**.loRaNodes[*].LoRaNic.radio.energyConsumerType = "LoRaEnergyConsumer"
**.loRaNodes[*].**.energySourceModule = "IdealEpEnergyStorage"
**.loRaNodes[*].LoRaNic.radio.energyConsumer.configFile = xmldoc("
    energyConsumptionParameters.xml")

**.sigma = 0
**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 3000m
**.constraintAreaMaxY = 3000m
**.constraintAreaMaxZ = 3000m
Cartagena.**.radio.separateTransmissionParts = false
Cartagena.**.radio.separateReceptionParts = false

**.delayer.config = xmldoc("cloudDelays.xml")
**.radio.radioMediumModule = "LoRaMedium"
**.minInterferenceTime = 0s
**.displayAddresses = false

**.LoRaMedium.analogModelType = "LoRaAnalogModel"
**.LoRaMedium.pathLossType = \$\{pathlosstype="LoRaLogNormalShadowing", "HataOkumura"\}
**.radio.carrierFrequency = 868MHz
```