DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

PROYECTO FIN DE CARRERA

INGENIERÍA DE TELECOMUNICACIÓN

DESARROLLO DE UN MÓDULO DE ANÁLISIS DE MAPAS SHAPEFILE PARA UN SIMULADOR DE REDES PRIME

AUTOR: ALBERTO MARTÍNEZ SÁNCHEZ

TUTOR: DR. GREGORIO LÓPEZ LÓPEZ

CO-DIRECTOR: DR. JAVIER MATANZA DOMINGO

LEGANÉS, FEBRERO DE 2016

TÍTULO: DESARROLLO DE UN MÓDULO DE ANÁLISIS DE MAPAS SHAPEFILE PARA UN SIMULADOR DE REDES PRIME.
AUTOR: ALBERTO MARTÍNEZ SÁNCHEZ
TUTOR: DR. GREGORIO LÓPEZ LÓPEZ
CO-DIRECTOR: DR. JAVIER MATANZA DOMINGO

EL TRIBUNAL

PRESIDENTE: DR. JOSÉ IGNACIO MORENO NOVELLA

VOCAL: DR. FRANCISCO JAVIER HERRAIZ MARTÍNEZ

SECRETARIO: DRA. FLORINA ALMENARES MENDOZA

REALIZADO EL ACTO DE DEFENSA Y LECTURA DEL PROYECTO FIN DE CARRERA EL DÍA 19 DE FEBRERO DE 2016 EN LEGANÉS, EN LA ESCUELA POLITÉCNICA SUPERIOR DE LA UNIVERSIDAD CARLOS III DE MADRID, ACUERDA OTORGARLE LA CALIFICACIÓN DE:

VOCAL

SECRETARIO PRESIDENTE

A mis padres

Agradecimientos

A Gregorio, algo más que un tutor, por su apoyo, dedicación y paciencia, por haber confiado en mí, por sus ánimos, por su optimismo y por ser el mejor guía en este largo camino. Él, junto a Javier y Miguel, han hecho posible que este proyecto salga adelante. Mil gracias a los tres.

A mi hermano José Ignacio, por transmitir siempre esa mezcla de felicidad y locura. Gracias por haber sido la mejor referencia, por el cariño que siempre me has dado y porque ambos sabemos que siempre podremos contar el uno con el otro.

A Celia, la última pieza del rompecabezas de mi vida, la parte que me complementa, la que me escucha y comprende, la única que siempre me hace sonreír. Gracias por creer en mí, por tu apoyo en los malos momentos, por tus consejos y porque imaginar un futuro junto a ti ha sido la mayor motivación. Para nosotros todos los días son y serán el principio…

A mis padres, José y Dolores, porque cuando me sentí perdido siempre encontré refugio en sus brazos. Nunca podré encontrar palabras para agradecer todo lo que me habéis dado. Os quiero.

“Si se quiere ascender por cuestas empinadas,
es necesario al principio andar despacio”
William Shakespeare (1564-1616)

Resumen

PRIME es un estándar de comunicaciones PLC de Banda Estrecha ampliamente utilizado en la última milla de los despliegues de infraestructuras avanzadas de medición en España (Iberdrola, Unión Fenosa) y con proyección internacional. Las herramientas de simulación son especialmente importantes para la planificación y evaluación de este tipo de redes, permitiendo minimizar riesgos, tiempo y costes. El objetivo de este Proyecto Fin de Carrera es desarrollar un módulo *software* que procese mapas de redes de distribución eléctrica en formato Shapefile e integrarlo en una aplicación Web para simulación de redes PRIME, permitiendo la simulación de redes PRIME desplegadas en campo.

El presente Proyecto Fin de Carrera ha sido desarrollado dentro del ámbito del proyecto de investigación nacional OSIRIS (Optimización de la Supervisión Inteligente de la Red de Distribución), financiado por el Ministerio de Economía y Competitividad y liderado por Unión Fenosa Distribución (tercera distribuidora eléctrica a nivel nacional).

Palabras clave

Comunicaciones PLC de Banda Estrecha; Django Web Framework; Infraestructuras Avanzadas de Medición; OMNeT++; PRIME; Python; Red Eléctrica Inteligente; Shapefile

Abstract

Narrowband-PLC (NB-PLC) technologies are winning momentum in the last mile of currently deployed Advanced Metering Infrastructures (AMI). PoweRline Intelligent Metering Evolution (PRIME) stands as a promising NB-PLC technology out of the available ones. PRIME presents high penetration in Spain, since it is being deployed by major Spanish Distribution System Operators (DSO) such as Iberdrola or Unión Fenosa (as a matter of fact, by 2018 there will be around 15M PRIME-compliant smart meters deployed only in Spain due to the Spanish directive IET/290/2012). In addition, the new version of the standard (v1.4) expands its focus worldwide, including frequency bands for the American and Asia Pacific markets.

Network simulation tools are especially important for PRIME networks, since they dramatically reduce the time and cost associated to making decisions on the planning and evaluation of this kind of networks. The main objective of this Thesis is to develop a software module that allows processing maps of actual power distribution networks in Shapefile format and to integrate it in a Web-based PRIME network simulator, thus enabling the simulations of PRIME networks already deployed in the field.

This Thesis has been developed within the scope of the national research project OSIRIS (Optimization of the Distribution Network Intelligent Monitoring), funded by the Spanish Ministry of Economy and Competiveness and led by Unión Fenosa Distribución (third company in the Spanish electricity distribution market).

Keywords

Advanced Metering Infrastructure (AMI); Django Web Framework; Narrowband-PLC (NB-PLC); OMNeT++; PoweRline Intelligent Metering Evolution (PRIME); Python; Shapefile; Smart Grids

Índice general

Índice de figuras

Índice de tablas

Acrónimos

AMI	*Advanced Metering Infrastructure*
ARQ	*Automatic Repeat reQuest*
BBDD	Bases de Datos
BER	*Bit Error Rate*
CAD	Computer Aided Design
CFP	*Contention Free Period*
COSEM	*COmpanion Specification for Energy Metering*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
DLMS	*Device Language Message Specification*
DPSK	*Differential Phase Shift Keying*
DSO	*Distribution System Operator*
ESRI	*Environmental Systems Research Institute*
FEC	*Forward Error Correction*
GML	*Geography Markup Language*
GNU	*General Public License*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
ITU-T	*Telecommunication Standardization Sector of the International Telecommunications Union*
LLC	*Logical Link Control*
LV	*Low Voltage*
MAC	*Medium Access Control*
MTU	*Maximum Transmission Unit*

MTV	Model-Template-View
MV	*Medium Voltage*
MVC	Modelo-Vista-Controlador
M2M	*Machine-to-Machine*
NB-PLC	*Narrowband Power Line Communications*
OFDM	*Orthogonal Frequency Division Multiplexing*
OSGP	*Open Smart Grid Protocol*
OSIRIS	Optimización de la Supervisión Inteligente de la Red de Distribución
PER	*Packet Error Rate*
PFC	Proyecto Fin de Carrera
PHY	*PHYsical*
PLC	*Power Line Communications*
PMBOK	*Project Management Body of Knowledge*
PMI	*Project Management Institute*
PRIME	*PoweRline Intelligent Metering Evolution*
SCP	*Shared Contention Period*
SIG	Sistemas de Información Geográfica
SNR	*Signal-to-Noise-Ratio*
TIC	Tecnologías de la Información y la Comunicación
VMDS	*Version Manager Data Store*
WS	*Windows Size*

Capítulo 1

Introducción

El primer capítulo de este Proyecto Fin de Carrera está dedicado a proporcionar una visión global del mismo. Comenzaremos por presentar los diferentes aspectos que han motivado su desarrollo y los objetivos que se pretenden alcanzar en él. Finalmente se describe como han sido estructurados los diferentes contenidos del presente documento.

1.1 Motivación

En la actualidad existen pocos sectores donde el uso de las Tecnologías de la Información y la Comunicación (TIC) no esté presente y el sector eléctrico no es la excepción, en gran parte por las denominadas *Smart Grids*. Estas redes inteligentes son las responsables de integrar de forma eficiente a todos los actores conectados a ella. De esta forma, se trata de asegurar un sistema energético sostenible y eficiente, con bajas pérdidas y altos niveles de calidad y seguridad [1].

Estas redes suponen una gran revolución desde el punto de vista de la distribución y el consumo de energía ya que proporcionan una base tecnológica para el panorama energético que aún está por llegar. La irrupción de las energías renovables no sólo está cobrando más protagonismo sino que está cambiando los flujos de energía. Los usuarios ahora no sólo consumen, sino que también producen electricidad a través de la misma red, por lo que éstas adquieren un carácter bidireccional con los *Smart Meters* como protagonistas.

Las redes de comunicaciones *Machine-to-Machine* (M2M) son de especial importancia para las *Smart Grids* ya que proporcionan la conectividad necesaria para transmitir la información entre los extremos de la red [2]. Uno de sus principales propósitos es permitir la gestión de los *Smart Meters* reduciendo la latencia, aumentando la disponibilidad y reduciendo los costes de despliegue y operación. Desde el punto de vista técnico, las tecnologías *Power Line Communications* (PLC) son especialmente atractivas para la comunicación directa con los *Smart Meters*, en lo que se conoce como la última milla de las Infraestructuras de Medición Avanzada (AMI).

Cuando hablamos de las tecnologías PLC nos referimos a los cables de baja tensión convencionales utilizados como medio de comunicación. Esta solución presenta muchos beneficios para las distribuidoras eléctricas (DSO o *Distribution System Operator*), como, por ejemplo, que el cable eléctrico ya está desplegado. Así, las tecnologías PLC de Banda Estrecha (NB-PLC o *Narrowband Power Line Communications*) están siendo ampliamente utilizadas para la comunicación entre los *Smart Meters* y los concentradores de datos (típicamente localizados en los centros de transformación) en los despliegues de AMI actuales [3].

Entre las distintas opciones NB-PLC disponibles en el mercado destaca la tecnología *PoweRline Intelligent Metering Evolution* (PRIME), por ser hoy en día un estándar PLC maduro, consolidado en España y con proyección internacional. Cuenta con un gran número de equipos y sistemas certificados de diferentes fabricantes que ya ha permitido el despliegue de más de 10 millones de *Smart Meters* PRIME por parte de DSOs en Europa y fuera de Europa (p.ej., Brasil o Australia). Además, en 2018 habrá alrededor de 15 millones de *Smart Meters* PRIME desplegados sólo en España, debido al cumplimiento de la directiva IET/290/2012, y la nueva versión del estándar extiende su alcance a nivel mundial.

El protocolo PRIME ha sido definido por la *PRIME Alliance* [4] y sus capas PHY y MAC han sido recientemente aceptadas como estándar por la ITU-T [5]. Cuenta con unas especificaciones exhaustivas y detalladas que no están sujetas a derechos de

propiedad intelectual, es decir, PRIME es una solución económicamente eficiente y sin barreras que beneficia a las compañías eléctricas y a sus usuarios.

Sin embargo, además de los beneficios que aportan las redes NB-PLC, en general, y PRIME, en particular, también presentan una serie de retos tecnológicos que pueden generar dudas respecto al rendimiento de sistemas basados en ellas. Este hecho remarca la importancia de las herramientas de simulación en este tipo de entornos para planificar, evaluar y tomar decisiones minimizando riesgos, tiempo y costes [6].

De entre los simuladores de redes PRIME disponibles en el estado del arte destaca simPRIME, desarrollado como parte de la tesis doctoral de Javier Matanza Domingo [7]. Este simulador persigue los objetivos propuestos haciendo uso de herramientas como Matlab y OMNeT++, por lo que ha despertado el interés por parte de distribuidoras eléctricas como Unión Fenosa o Iberdrola.

Sin embargo, la larga curva de aprendizaje que requiere la utilización de este tipo de simuladores supone un obstáculo para ser usado en entornos de producción. Esta dificultad de manejo motivó el desarrollo de una aplicación Web en un Proyecto Fin de Carrera previo para simplificar la interacción con el simulador, intentando aumentar así su impacto en la industria.

Dicha aplicación Web permite simular ciertas topologías lógicas bajo ciertas condiciones de la red de comunicaciones, determinadas por la probabilidad de error de bit (BER) entre distintos niveles lógicos. Sin embargo, las topologías lógicas varían con el tiempo en una red PRIME real y las BER dependen de la modulación utilizada y de la relación señal a ruido (SNR), que a su vez depende de factores relacionados con la topología física de la red de distribución eléctrica, tales como las impedancias o las distancias entre nodos.

El objetivo de este Proyecto Fin de Carrera es precisamente desarrollar un módulo *software* que permita procesar información geográfica de redes de distribución eléctrica desplegadas en campo e integrarlo en la aplicación Web existente, permitiendo la simulación de redes PRIME en operación. De este modo, la aplicación permitirá evaluar problemas de redes PRIME en operación así como diferentes soluciones, reduciendo drásticamente el tiempo y el coste de solventarlos. Shapefile [8] es el formato elegido para la representación de la información geográfica debido a su gran aceptación en este tipo de entornos (p.ej., es el formato utilizado por Unión Fenosa para la representación geográfica de sus redes de distribución eléctrica).

Cabe destacar que el presente Proyecto Fin de Carrera ha sido desarrollado dentro del ámbito del proyecto de investigación nacional OSIRIS (Optimización de la Supervisión Inteligente de la Red de Distribución) [9], financiado por el Ministerio de Economía y Competitividad y liderado por Unión Fenosa Distribución (tercera distribuidora eléctrica a nivel nacional), lo que demuestra su alineación con las demandas actuales de este sector de la industria.

1.2 Objetivos

El objetivo principal de este Proyecto Fin de Carrera es permitir la simulación de redes PRIME desplegadas en campo a partir de información geográfica de las redes de distribución eléctrica subyacentes recogida en formato Shapefile. Este formato describe completamente la topología física de dichas redes por lo que se propone su procesamiento para que el simulador de redes simPRIME proporcione resultados que permitan evaluar problemas de redes PRIME en operación así como diferentes soluciones, reduciendo drásticamente el tiempo y el coste de solventarlos.

Para conseguir este objetivo global, se plantean los siguientes subobjetivos:

- Planificación y diseño de las etapas en las que se dividirá el presente Proyecto Fin de Carrera.
- Estudio de los Sistemas de Información Geográfica (SIG) y en concreto del formato Shapefile.
- Estudio de la aplicación Web existente con el fin de obtener una panorámica de cada uno de los módulos que la forman.
- Desarrollo de un módulo *software* que permita simular redes PRIME a partir de la información recogida en archivos Shapefile.
- Integración del módulo *software* desarrollado en la aplicación Web original con la mayor transparencia posible.
- Validación del módulo desarrollado de forma individual y posteriormente de forma global en el ámbito de la aplicación Web.
- Documentación del desarrollo realizado.

1.3 Estructura de la memoria

La organización de los contenidos del presente Proyecto Fin de Carrera se detalla a continuación:

- **Capítulo 1: Introducción**

El primer capítulo de este Proyecto Fin de Carrera está dedicado a proporcionar una visión global del mismo. En este capítulo se presentan los diferentes aspectos que han motivado el desarrollo de este PFC, los objetivos que se pretenden alcanzar en él y la estructura del presente documento.

- **Capítulo 2: Estado del Arte**

El propósito de este capítulo es obtener una visión global de todas aquellas tecnologías y herramientas relacionadas con el presente Proyecto Fin de Carrera. Así, se revisan las tecnologías NB-PLC para presentar más detalladamente las redes PRIME y así comprender la importancia de la existencia de simuladores de redes de comunicaciones en este ámbito. Seguidamente, se presenta el funcionamiento del simulador simPRIME. A continuación se presenta un análisis de diferentes opciones disponibles para llevar a cabo el desarrollo *software* propuesto. El capítulo finaliza con una introducción a los Sistemas de Información Geográfica (SIG) donde los Shapefiles son protagonistas.

- **Capítulo 3: Diseño**

En este capítulo se describe el proceso de diseño y la metodología de trabajo llevada a cabo en el presente proyecto. Además, se proporciona una visión global de la aplicación con el fin de contextualizar cada uno de los módulos explicados detalladamente en el próximo capítulo.

- **Capítulo 4: Desarrollo**

El objetivo de este capítulo es tomar como referencia el diseño presentado en el capítulo anterior para presentar los aspectos más técnicos del desarrollo. Se detalla la implementación de cada uno de los módulos en el orden en que fueron desarrollados para permitir comprender el proceso de forma clara y sencilla

- **Capítulo 5: Validación**

Este capítulo detalla las diferentes validaciones realizadas para comprobar el correcto funcionamiento del módulo *software* desarrollado. En primer lugar se desarrolla un entorno de pruebas sencillo bajo una topología lineal para posteriormente realizar el mismo estudio bajo planos de redes de distribución eléctrica reales proporcionados por Unión Fenosa. Finalmente se valida la correcta integración del módulo *software* desarrollado en la aplicación Web existente.

- **Capítulo 6: Conclusiones y Líneas de Trabajo Futuras**

En este capítulo final, se exponen las conclusiones obtenidas tras la ejecución del presente Proyecto Fin de Carrera así como posibles trabajos futuros.

- **Anexo I: Modelado de la función de transferencia de canal**

En este anexo se detalla el proceso mediante el cual se obtiene la matriz de atenuaciones.

- **Anexo II: Presupuesto**

Este anexo presenta la planificación y fases de desarrollo del Proyecto Fin de Carrera así como el presupuesto del mismo, desglosándolo en los correspondientes costes de material y costes de recursos humanos.

Capítulo 2

Estado del arte

El propósito de este capítulo es obtener una visión global de todas aquellas tecnologías y herramientas relacionadas con el presente Proyecto Fin de Carrera. Así, se revisan las tecnologías NB-PLC para presentar más detalladamente las redes PRIME y así comprender la importancia de la existencia de simuladores de redes de comunicaciones en este ámbito. Seguidamente, se presenta el funcionamiento del simulador simPRIME. A continuación se presenta un análisis de diferentes opciones disponibles para llevar a cabo el desarrollo *software* propuesto. El capítulo finaliza con una introducción a los Sistemas de Información Geográfica (SIG) donde los Shapefiles son protagonistas.

2.1 Tecnologías PLC de Banda Estrecha

2.1.1 Visión global

Las redes de comunicaciones M2M son de especial importancia para las *Smart Grids* en tanto en cuanto son clave para permitir la monitorización y control en casi tiempo real del elevado número de puntos de consumo y de generación previstos. Por lo tanto, deben cumplir requisitos muy exigentes desde el punto técnico (p.ej., baja latencia, alta disponibilidad) [10], pero también desde el económico (bajos costes de despliegue y operación) [3].

Precisamente, las tecnologías PLC son especialmente relevantes en este tipo de entornos porque representan una solución de compromiso entre ambas perspectivas. Así, por ejemplo, sólo la posibilidad de poder aprovechar el cableado de baja tensión convencional supone una reducción en los costes de despliegue que para otro tipo de tecnologías es imposible alcanzar. Sin embargo, este tipo de medio es poco amigable desde el punto de vista de comunicaciones dado que no fue diseñado para la transmisión de datos sino para la transmisión de potencia, lo cual supone grandes retos para su implementación.

Las posibilidades y los beneficios que deja entrever esta tecnología han motivado que las redes NB-PLC estén siendo ampliamente desplegadas en la última milla de los despliegues AMI actuales, como interfaz entre los *Smart Meters* y los concentradores de datos (típicamente localizados en los centros de transformación). Ante esta situación, son varias las opciones que encontramos en el mercado en forma de estándares:

- *Meters and More* es una tecnología cuya especificación viene liderada por el grupo Enel, aunque la especificación de sus capas inferiores también ha sido aceptada como estándar por IEC/CENELEC (CLC TS 50568-4). Presenta sus mayores niveles de penetración en mercados donde opera el grupo ENEL, tales como Italia (el 100% de los contadores italianos usan esta tecnología actualmente) o España (aproximadamente la mitad del parque de contadores inteligentes españoles utilizarán esta tecnología en 2018). Funciona en la banda CENELEC-A. Utiliza una única portadora, destacando por su robustez, aunque alcanza tasas de transmisión de datos bajas (9,6 kbps).
- *Open Smart Grid Protocol* (OSGP) es una tecnología promovida por el fabricante norteamericano Echelon, aunque sus capas inferiores están también estandarizadas por el IEC (IEC 14908.1). Presenta sus mayores tasas de despliegue en los países nórdicos y Rusia. Funciona en la banda CENELEC-A. Se trata de una tecnología monoportadora que alcanza tasas de transmisión de datos de hasta 3,6 kbps.
- CX1 es una tecnología promovida por Siemens y cuyas capas inferiores están siendo estandarizadas también por IEC/CENELEC (CLC TS 50590). Actualmente se está desplegando en Austria. Funciona en la banda CENELEC-A. Utiliza una modulación multiportadora adaptativa que le permite alcanzar tasas de hasta 64 kbps.

- G3 es una tecnología cuya especificación lidera la distribuidora EDF y el fabricante de chipsets Maxim, aunque sus capas PHY y de enlace también han sido publicadas como estándar por la ITU-T recientemente (ITU-T G9903 [11]). Presenta sus mayores niveles de penetración en mercados donde opera EDF, como Francia. Inicialmente definida para que funcionase en la banda CENELEC-A, ha sido extendida recientemente para poder ser utilizada en el mercado americano (FCC) y asiático (ARIB). Se trata de una tecnología multiportadora que alcanza tasas de hasta 34 kbps y que ha destacado desde sus inicios por ser especialmente robusta.
- Como ya se ha comentado, *PoweRline Intelligent Metering Evolution* (PRIME) es una tecnología promovida por la *PRIME Alliance*, liderada por distribuidoras eléctricas españolas como Iberdrola y Unión Fenosa y por fabricantes de chipsets como *Texas Instruments*. Sus capas PHY y de enlace también han sido publicadas como estándar por la ITU-T (ITU-T G.9904) [5]. Al igual que G3, fue inicialmente definida para que funcionase en la banda CENELEC-A y ha sido extendida recientemente para poder ser utilizada en el mercado americano (FCC) y asiático (ARIB). Es una tecnología multiportadora que puede llegar a alcanzar tasas de hasta 128,6 kbps (en su versión para la CENELEC-A).
- G.hnem [12] se trata de un esfuerzo de la ITU-T por homogeneizar G3 y PRIME, aunque su implementación es computacionalmente más compleja que las anteriores. Su utilización actualmente es baja.
- IEEE 1901.2 es la especificación NB-PLC del IEEE. Su utilización actualmente también es baja.

Entre las distintas opciones NB-PLC disponibles en el mercado, destaca la tecnología PRIME por ser hoy en día un estándar PLC maduro, consolidado en España y con proyección mundial. Cuenta con un gran número de equipos y sistemas certificados de diferentes fabricantes que ya ha permitido el despliegue de más de 10 millones de *Smart Meters* PRIME por parte de proveedores y servicios públicos en Europa, Brasil y Australia.

Ya que el simulador que ha motivado la realización del presente Proyecto Fin de Carrera se centra en esta tecnología, el próximo apartado está dedicado a ella, resumiendo brevemente sus principales detalles técnicos.

2.1.2 PRIME

El protocolo PRIME ha sido definido por la *PRIME Alliance* [4], cuyo objetivo es establecer un conjunto de estándares públicos. Cuenta con unas especificaciones exhaustivas y detalladas sin estar sujetos a derechos de propiedad intelectual, es decir, PRIME es una solución económicamente eficiente y sin barreras que beneficia a las compañías eléctricas y a sus usuarios.

Es una tecnología NB-PLC de segunda generación donde su versión 1.3.6 de la especificación de las capas PHY, MAC y de Convergencia ha sido aceptada como estándar por la ITU-T desde 2012 [5]. La versión 1.4 expande el espectro de frecuencia utilizado para poder operar en los mercados americano y asiático e incluye algunas

funciones para incrementar la robustez de la comunicación a nivel de la capa PHY y MAC.

Desde una perspectiva de la capa PHY, PRIME opera en la banda de 41-89 kHz (v1.3.6) o de 3-500 kHz (v1.4), usando la modulación OFDM para hacer un uso más eficiente del espectro. Los dispositivos PRIME pueden usar las modulaciones DPSK, DQPSK o D8PSK. Además, permite la utilización de FEC para recuperarse ante posibles errores introducidos por el canal. A nivel de capa física, la velocidad de transmisión en redes PRIME puede ir desde los 5.4 kbps hasta los 1028.8 kbps, dependiendo de la combinación de modulación digital y FEC utilizadas (también conocido como modo de comunicación).

A nivel MAC, se definen dos tipos de nodos: Base y de Servicio. Sólo se permite un Nodo Base por cada red PRIME, ya que actúa de coordinador. En terminología AMI, el Nodo Base es conocido como concentrador de datos, o simplemente concentrador. Aunque los Nodos de Servicio suelen ser *Smart Meters*, dependiendo de las necesidades de la red, también pueden funcionar como *switches*. El objetivo principal de los *switches* es incrementar el rango de la señal en el cable mediante un mecanismo de repetición, mitigando así los efectos del ruido y la atenuación. Así, la topología física de este tipo de redes es en *bus*, mientras que la topología lógica tiene típicamente forma de árbol. Un aspecto a considerar es que la posición de estos *switches* puede influir en el comportamiento de la red, como se discute en [13].

En relación a los mecanismos MAC, aunque el estándar define un periodo con contienda (SCP) y un periodo libre de contienda (CFP), actualmente sólo el SCP está implementado por los fabricantes. A pesar de ello, se está investigando sobre los beneficios potenciales de usar CFP para nuevos servicios de *Smart Grids* [14]. En el SCP se utiliza CSMA/CA como técnica de acceso al medio.

La capa de Control de Enlace Lógico (LLC), perteneciente a la capa de Convergencia, es responsable de manejar las conexiones lógicas. Identifica cada transacción con un número de identificación y realiza los procesos de control de flujo. El control de flujo está implementado en PRIME mediante el establecimiento de una Unidad Máxima de Transmisión (MTU) y usando un procedimiento de ventana deslizante. La MTU define la longitud en *bytes* del mayor paquete de datos que puede encapsular el nivel MAC. En caso de que la aplicación intente enviar un mensaje mayor, la capa LLC fragmenta el mensaje en varios paquetes, ninguno de los cuales será mayor que la MTU. Cada uno de estos paquetes está etiquetado con un identificador para que la parte receptora pueda reensamblarlos.

Con respecto al procedimiento de ventana deslizante, PRIME establece diferentes valores permitidos para el Tamaño de Ventana (WS). El valor del WS puede jugar un rol importante en el rendimiento de la red en términos de latencia. Además, los dispositivos PRIME pueden implementar o no capacidades de Repetición de Solicitudes (ARQ) para asegurar la correcta recepción de todos los mensajes. Dado que los parámetros ARQ son negociados en la fase de conexión, este proceso funciona extremo a extremo entre el transmisor y el receptor final, siendo por tanto los *switches* transparentes al mismo.

En la capa de aplicación, DLMS/COSEM es el estándar utilizado sobre todas las tecnologías NB-PLC disponibles en el mercado. En concreto, COSEM (IEC 62056-61/62) es un perfil del protocolo de aplicación DLMS (IEC 62056-53) especialmente diseñado para la medición de energía. Como tal, DLMS/COSEM incluye modelos de datos para representar parámetros comunes relacionados con la energía junto con un protocolo de comunicación diseñado para transmitir este tipo de información.

2.2 Simuladores de redes de comunicaciones

Esta sección analiza algunas de las soluciones disponibles en el mercado para desarrollar herramientas de simulación de redes de comunicaciones. Dentro del ámbito de los simuladores de redes de comunicaciones propiamente dichos, esta sección se centra en los siguientes por tratarse de los tres con más aceptación en el mercado:

- Riverbed Modeler (anteriormente, OPNET) [15].
- NS3 [16].
- OMNeT++ [17].

Sin embargo, también pueden desarrollarse herramientas de simulación en base a *software* de cálculo (p.ej., Microsoft Excel o Matlab) o en lenguajes de scripting (p. ej., Python), dependiendo de las características de las tecnologías de comunicaciones a simular.

2.2.1 Riverbed Modeler

Riverbed Modeler (antes conocido como OPNET Modeler), producto actualmente perteneciente a Riverbed, es un potente simulador de redes de comunicaciones basado en eventos discretos que incluye modelos muy precisos de tecnologías de comunicaciones como, por ejemplo, las móviles (2G, 3G). Sin embargo, no existe modelo aún para redes PLC, aunque podría desarrollarse partiendo de los módulos ya implementados (p. ej., Ethernet). El lenguaje de programación que se utiliza en este simulador es C++.

La versión comercial de Riverbed Modeler requiere del pago de una licencia que incluye la utilización del simulador así como soporte técnico y formación. En cualquier caso, las universidades y centros de investigación actualmente tienen la posibilidad de obtener una licencia de pruebas de 6 meses, que puede renovarse siempre y cuando se hagan públicos los resultados de investigación obtenidos.

2.2.2 NS3

NS3 es la última versión del popular simulador de redes de comunicaciones NS2. NS2 y NS3 son bastante comunes en entornos académicos, llegando a ser casi de uso obligatorio para que los resultados obtenidos al simular determinados protocolos tengan validez dentro de la comunidad investigadora (p.ej., para protocolos del *Internet Engineering Task Force* - IETF).

Existe una implementación de PLC para NS3 [18] que podría utilizarse como referencia y adaptarse a PRIME.

NS es software libre que se ofrece bajo la versión 2 de la *General Public License* (GNU).

2.2.3 OMNeT++

OMNeT++ es un simulador de redes de comunicaciones modular basado en eventos discretos. En OMNeT++ existen dos tipos de módulos:

- Simples: que representan elementos atómicos;
- Compuestos: compuestos – valga la redundancia – por una combinación de módulos simples.

En OMNeT++ se manejan tres tipos de ficheros:

- .cpp: definen funcionalidad, utilizando C++ como lenguaje de programación;
- .ned: definen la topología de red;
- .ini: configuran las simulaciones a realizar (parámetros de entrada, parámetros de salida, duración, etc.).

No se ha encontrado implementación oficial de PRIME para OMNeT++. Sin embargo, como ya se ha comentado, en [7] se presenta la versión inicial de simPRIME, una implementación de PRIME para OMNeT++ desarrollada a partir de la implementación de Ethernet disponible en el INET Framework.

OMNeT++ se distribuye bajo *Academic Public License*, que no permite su uso comercial. La versión comercial de OMNeT++ es OMNEST.

2.3 Simuladores de redes PRIME

En este mismo capítulo hemos presentado las principales tecnologías NB-PLC y las características de PRIME en profundidad, escenario cuyos retos tecnológicos aún suponen demasiada incertidumbre. Este hecho remarca la importancia de las herramientas de simulación en este tipo de entornos para planificar, evaluar y tomar decisiones minimizando tiempo, costes y riesgos.

2.3.1 Visión global

Dado el aumento de la popularidad de las redes PRIME, una amplia cantidad de herramientas de simulación están siendo desarrolladas para evitar el alto coste y la complejidad de los laboratorios de pruebas o de los despliegues en campo. Prueba de ello son los numerosos trabajos de investigación que se han llevado a cabo recientemente en este sentido:

- [19] presenta, por ejemplo, un análisis interesante sobre el rendimiento disponible en redes PLC multi-salto. Sin embargo, no se considera ruido en el canal, por lo que la comunicación no presenta errores.
- [20] propone un método para abstraer la capa PHY de las simulaciones a través de curvas de PER vs SNR. Sin embargo, esas curvas se calculan para un tamaño de paquete fijo, lo que hace que los resultados no sean realistas.
- [21] presenta un simulador de redes PRIME basado en OMNeT++. Sin embargo, este simulador no considera la capa de aplicación, sino que se fija exclusivamente en probabilidades de error. Este tipo de estudios pueden tener interés, pero los DSO se fijan en otro tipo de parámetros representativos del rendimiento de la red, tales como la latencia.
- [22] presenta también un simulador de redes PRIME basado en OMNeT++. Dicho simulador se utiliza concretamente para simular la telecarga de firmware en redes PRIME. Al igual que [33], abstrae la capa PHY de las simulaciones a través de curvas de PER vs SNR. En este caso también se toma un tamaño de paquete fijo para calcular la PER pero además es el máximo posible, lo que penaliza al tráfico de control, que se suele caracterizar por el reducido tamaño de sus paquetes.

El simulador de redes PRIME en el que se centra este PFC (simPRIME) utiliza MATLAB para abstraerle el nivel PHY a OMNeT++, pero trabaja a nivel de BER y cubre no sólo hasta el nivel de enlace sino hasta el de aplicación. El siguiente apartado resume brevemente sus principales detalles de implementación.

2.3.2 simPRIME

Como ya se ha comentado, el núcleo del simulador simPRIME fue desarrollado como parte de la tesis doctoral de Javier Matanza [7]. Este simulador combina

MATLAB y OMNeT++ para modelar los efectos de la capa PHY y de las capas superiores (MAC, LLC y aplicación) respectivamente, como ilustra la parte derecha de la Figura 1. Como también muestra la Figura 1, y en consonancia con lo comentado en la sección 2.1.2, la implementación de las capas PHY, MAC y LLC se ajusta a la especificación PRIME, mientras que la capa de aplicación se modela en base a DLMS/COSEM.

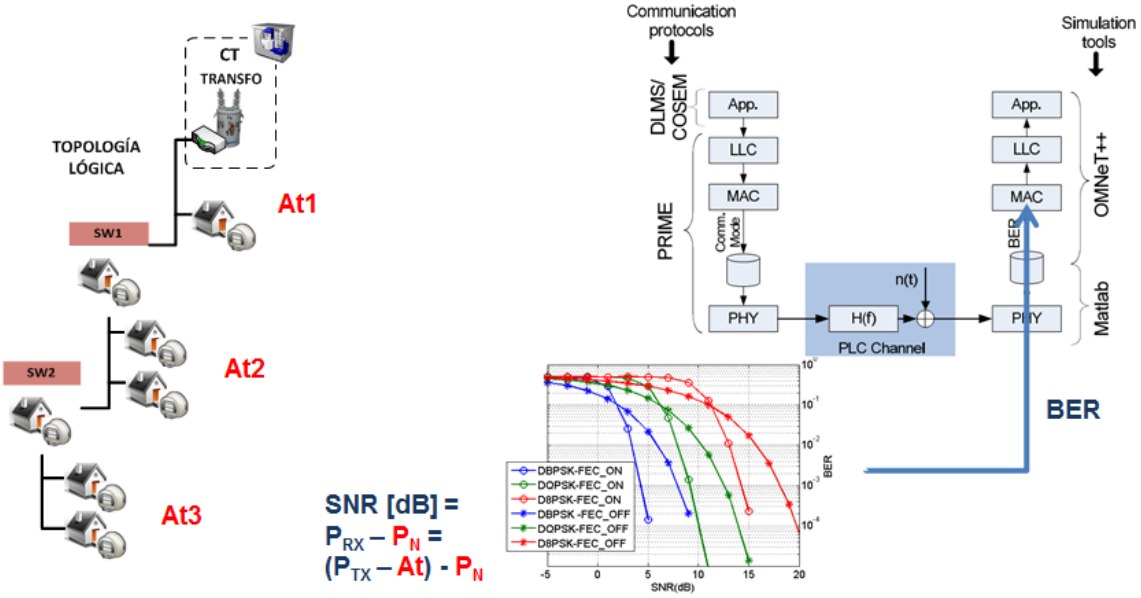


Figura 1: Esquema de la arquitectura de SimPRIME.

La Figura 1 también ilustra la interacción entre MATLAB y OMNeT++. Como puede observarse, conocidos la potencia de transmisión y el nivel de ruido base, cuando un nodo envía un mensaje a otro, se calcula la potencia recibida en base a una matriz de atenuaciones que contiene la atenuación entre cada par de nodos. Con la potencia recibida y la potencia de ruido base se obtiene la SNR, tal y como muestra la ecuación incluida en la Figura 1. Conocidos la SNR y la constelación utilizada, se obtiene la BER mediante las curvas SNR frente a BER que también muestra la Figura 1. Dichas curvas se calculan *a priori* en base al estándar usando para ello MATLAB. OMNeT++ utiliza el valor de BER resultante para decidir si el mensaje recibido contiene errores (y por tanto debe ser descartado) o por el contrario fue recibido correctamente (y puede ser procesado por las capas superiores).

La versión original del simulador ha sido extendida permitiendo simular topologías lógicas fijadas *a priori*, como la que aparece en la parte izquierda de la Figura 1. Esta nueva característica permite simular topologías lógicas reales obtenidas a partir del procesamiento del fichero de topología estándar que proporcionan los concentradores (S11.xml). En esta misma línea, el simulador original también ha sido extendido para que las atenuaciones entre cada par de nodos no sólo puedan obtenerse en base a teoría de líneas de transmisión [23], sino que también puedan obtenerse a partir de determinadas BER fijadas para cada par de nodos de la red dependiendo del nivel lógico en el que se encuentren. En la Figura 1, por ejemplo, se propone asumir una BER para todos los nodos que se encuentran en un mismo nivel de la topología lógica (dando lugar a *At1*), otra entre los nodos que están a un nivel de distancia (dando lugar a *At2*) y otra entre los nodos que están a más de un nivel de distancia (dando lugar a *At3*).

Sin embargo, como se ha mencionado en el capítulo 1, la topología lógica de una red PRIME varía con el tiempo y la BER depende de factores relacionados con la topología física de la red, por lo que se hace necesario un módulo *software* como el que representa el objetivo de este PFC para poder simular escenarios reales desplegados en campo.

2.4 Revisión de tecnologías de desarrollo

2.4.1 Python

Python es un lenguaje de programación multiparadigma, fácil de aprender, poderoso y con una semántica dinámica. Su enfoque es simple pero efectivo, del mismo modo que sus estructuras de datos son eficientes y de un alto nivel. Está basado en el paradigma de programación orientada a objetos. Se trata además de un lenguaje interpretado [24].

Una de sus principales características es la reducción del coste de mantenimiento del *software*, algo que es posible gracias a la legibilidad del código. Por otro lado, si lo comparamos con un lenguaje de gran aceptación como Java, hay que apuntar que un código en Python puede ser entre 3 y 5 veces más corto que el primero [25].

Otra de las características de Python, es que suele ser una de las opciones de mayor utilidad por parte de los desarrolladores para implementar tanto *frontend* como *backend*, en gran parte debido a la amplia comunidad Python que existe en la actualidad. En su sitio Web podemos encontrar a nuestra disposición el intérprete de Python para las principales plataformas y la biblioteca estándar en forma binaria. Además, se encuentran también múltiples enlaces a módulos hechos por terceros, programas y herramientas que facilitan la mayoría de las implementaciones.

Uno de los problemas que encontramos al hablar de Python son las incompatibilidades que se dan entre las versiones 2.7 y 3. El problema radica en que la versión 3 no incluye compatibilidad con algunos elementos de otras versiones anteriores, por lo que sólo se recomienda su uso cuando no se den dichas incompatibilidades. No obstante, la aceptación por la versión 3 ha sido tan alta que finalmente terminará desbancando a las versiones anteriores [26].

2.4.2 Django

Django es un *Web framework* para Python, mantenido por la *Django Software Foundation* [27], orientado al desarrollo de aplicaciones Web utilizando el patrón MVC (Modelo-Vista-Controlador). Está diseñado para facilitar la implementación y el despliegue de aplicaciones, facilitando la interactuación del sistema con la base de datos. Es fácilmente escalable y ofrece por defecto características de seguridad como protección contra inyecciones SQL o un sistema de gestión de cuentas de usuario [28].

Este *framework* induce a un diseño modular de aplicaciones para fomentar la reutilización del código. Además, incluye un sistema de gestión (django-admin o manage.py) que permite crear aplicaciones y módulos, usuarios de administración, gestionar la internacionalización del sistema, lanzar un servidor Web de desarrollo y gestionar la base de datos de forma simple y eficiente.

Aunque Django sigue un patrón de diseño MVC, según los desarrolladores del *framework*, esa terminología es debatible y prefieren referirse a un patrón de diseño Modelo-Plantilla-Vista (MTV o *Model-Template-View*), correspondiéndose los componentes vista y controlador del patrón MVC a los componentes plantilla y vista respectivamente (el componente modelo no varía de nombre) [29].

2.4.3 Celery

Es un conjunto de herramientas que nos permite trabajar fácilmente con múltiples servicios de tal forma que se pueden lanzar servicios viéndolos como tareas [30]. Además, ofrece una gran compatibilidad con aplicaciones escritas en Python y Django.

Las tareas en proyectos complejos son de suma importancia ya que reduce el esfuerzo de la aplicación en realizar actividades en un solo hilo, pudiendo generar varios que realicen tareas de manera asíncrona y de forma eficiente. La cola de colas de tareas asíncronas está basada en paso de mensajes distribuidos, centrándose en una operación en tiempo real pero admitiendo también la programación de las mismas. Las unidades de ejecución, llamadas tareas, se ejecutan simultáneamente en un único o varios servidores de trabajo utilizando el multiprocesamiento.

2.4.4 RabbitMQ

Es un software de negociación de mensajes de código abierto que entra dentro de la categoría de *middleware* de mensajería. Cuando hablamos de comunicación asíncrona, hablamos de que el emisor y el receptor no tienen que estar conectados de manera simultánea a la cola para que llegue a darse la transmisión del mensaje. Este procedimiento es muy utilizado en aplicaciones Web que exigen la ejecución simultánea de tareas pesadas en segundo plano. Lo que hace es ejecutar dichos procesos pesados mientras continua interactuando con el usuario, gracias a que la interfaz Web se comunica con el *backend* que los ejecuta [31].

2.4.5 PostgreSQL

Es un potente sistema de base de datos objeto-relacional de código abierto que permite que, mientras un proceso escribe en una tabla, otros accedan a la misma sin necesidad de bloqueos. De igual modo, implementa características como control transaccional y cumple con el estándar ANSI-SQL:2008 [32].

2.5 Sistemas de Información Geográfica

Durante siglos hemos utilizado los tradicionales mapas de papel para representar nuestro espacio y así poder gestionar, organizar y tomar decisiones sobre el territorio. Con la rápida evolución de los Sistemas de Información Geográfica (SIG) y su tecnología asociada, los SIG han revolucionado el mundo de la cartografía, del análisis espacial, de la planificación y de la gestión del territorio.

2.5.1 Una visión global de los SIG

Se entiende por sistema de información a la conjunción de información con herramientas informáticas, es decir, con programas informáticos o *software*. Si el objeto concreto de un sistema de información es la obtención de datos relacionados con el espacio físico, entonces estaremos hablando de un Sistema de Información Geográfica.

Así pues, un SIG es un *software* específico que permite a los usuarios crear consultas interactivas, integrar, analizar y representar de una forma eficiente cualquier tipo de información geográfica referenciada asociada a un territorio, conectando mapas con bases de datos [33].

El uso de este tipo de sistemas facilita la visualización de los datos obtenidos en un mapa con el fin de reflejar y relacionar fenómenos geográficos de cualquier tipo, desde mapas de carreteras hasta sistemas de redes eléctricas.

La mayoría de los elementos que existen en la naturaleza pueden ser representados mediante formas geométricas (puntos, líneas o polígonos, esto es, vectores) o mediante celdillas con información (*raster*). Son formas intuitivas y versátiles de ilustrar el espacio que ayudan a comprender mejor los elementos objeto de estudio según su naturaleza.

En función de la forma que utilicen para representar el espacio de la que hacen uso, podemos clasificar los SIG en dos grandes modelos o formatos como se ilustran en la Figura 2:

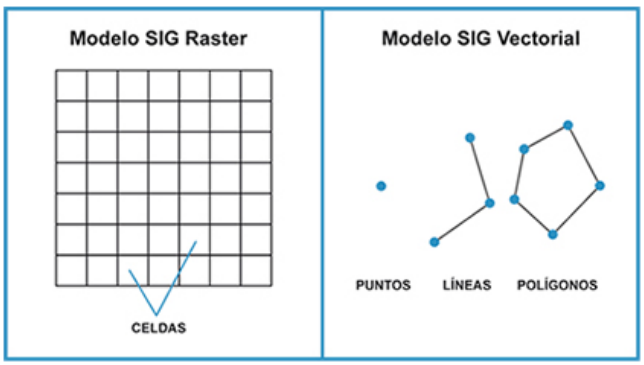


Figura 2: Modelo raster y modelo vectorial.

La elección de un modelo u otro dependerá de si las propiedades topológicas son importantes para el análisis. Sí es así, el modelo de datos vectorial es la mejor opción, pero su estructura de datos, aunque muy precisa, es mucho más compleja y esto puede ralentizar el proceso de análisis. Por ello, si el análisis que nos interesa no requiere acudir a las propiedades topológicas, es mucho más rápido, sencillo y eficaz el uso del formato *raster*.

También es más fácil decantarse por una estructura de datos vectorial cuando hay que reflejar más de un atributo en un mismo espacio. Usar un formato *raster* nos obligaría a crear una capa distinta para cada atributo.

2.5.2 Funcionamiento de los SIG

Los SIG operan como una base de datos geográfica asociada a los objetos existentes en un mapa digital, y dan respuesta a las consultas interactivas de los usuarios analizando y relacionando diferentes tipos de información con una sola localización geográfica. Básicamente, el funcionamiento de un SIG pasa por las siguientes fases:

- Entrada de la información en el sistema, ya sea digital o pendiente de digitalización.
- Almacenamiento y actualización de las bases de datos geográficamente, es decir, georreferenciar la información mediante coordenadas geográficas de latitud y longitud.
- Análisis e interpretación de los datos georreferenciados.
- Salida de la información en forma de productos diferentes, que dependerán de las necesidades del usuario.

Los SIG, como se muestra en la Figura 3, separan la información en capas temáticas y las almacena de forma independiente, haciendo más rápida y sencilla la tarea final de relacionar la información existente para la obtención de resultados.

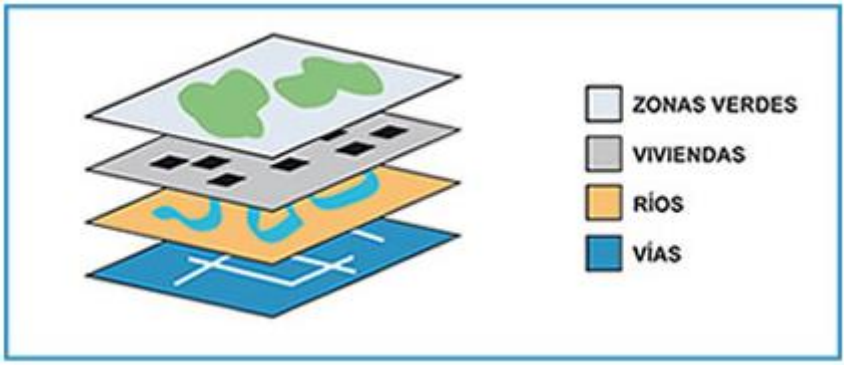


Figura 3: Modelo de información en capas.

No obstante, en función del formato que se utilice estas formas de separar o dividir la información se llevará a cabo de una forma diferente. A continuación, se presentan los principales formatos de almacenamiento [34]:

Ficheros CAD (Microstation y AutoCAD)

Los conocidos como ficheros *Computer Aided Design* (CAD) no sólo almacenan datos geográficos, sino que son el formato de almacenamiento de las herramientas de diseño de arquitectos, ingenieros y diseñadores. Las más conocidas y usadas son AutoCAD y Microstation.

Desde finales de los ochenta, las herramientas CAD no sólo permiten diseñar geometrías, sino que también permiten ubicarlas en el territorio georeferenciándolas. De esta manera, sus ficheros pueden almacenar datos ubicados sobre el territorio.

Cada herramienta CAD tiene un formato propio que normalmente cambia de una versión de la herramienta a otra. Los formatos de archivo CAD más habituales son:

- AutoCAD DXF. Es un formato de archivo CAD diseñado por Autodesk de cara a facilitar la interoperabilidad de AutoCAD con otros programas.
- DWG. Es un formato de archivo que permite guardar datos de diseños en dos y tres dimensiones y es el formato nativo e interno de trabajo de AutoCAD.
- DGN. Es un formato de archivo CAD con las mismas capacidades que DWG, pero para la herramienta Microstation de Bentley.

General Electric Smallworld

Smallworld, propiedad de General Electric, está basado en dos tecnologías:

- Se ejecuta sobre el lenguaje Magik, un lenguaje orientado a objetos que se ejecuta sobre una máquina virtual que interpreta el código fuente.
- El formato de almacenamiento está basado en *Version Manager Data Store* (VMDS), que es un formato propietario de General Electric diseñado para ser una base de datos relacional orientada a versiones.

El formato de Smallworld permite almacenar puntos, líneas, polígonos y *rasters* y tiene amplias capacidades topológicas. Quizá el hecho más característico de este formato es que trata todos los elementos como objetos, no como simples datos. La diferencia radica en el hecho de que los objetos, aparte de sus propiedades, tienen métodos para transformarlos, enlazarlos o, en general, para trabajar con ellos.

ESRI Shapefile

Shapefile es uno de los formatos más populares y sobresalientes. Fue diseñado y lo mantiene el *Environmental Systems Research Institute* (ESRI) como formato interoperable y de intercambio de información entre las herramientas de ESRI y otras herramientas SIG. Un Shapefile se almacena en varios ficheros, pero su contenido se puede entender como una tabla de una base de datos relacional.

Los datos de un shapefile se almacenan en una combinación de varios ficheros:

- Fichero .shp. Contiene las geometrías de los elementos.
- Fichero .shx. Contiene el índice de los elementos.
- Fichero .dbf. Contiene los atributos alfanuméricos de los elementos. Son estos datos los que se pueden considerar como una tabla de base de datos relacional.

- Fichero .prj. Contiene el sistema de coordenadas del Shapefile.
- Fichero .sbn. Contiene los índices espaciales, si los hay.

De todos éstos sólo son obligatorios los ficheros .shp, .shx y .dbf. Los otros ficheros son opcionales. Al no ser una base de datos relacional propiamente dicha, tiene algunas limitaciones:

- No permite modelos de almacenamiento avanzados que incluyan topología y LRS.
- No permite contener valores nulos.
- No permite el acceso concurrente para la modificación de los datos.

A pesar de las limitaciones, es uno de los formatos más extendidos debido a su simplicidad y su formato abierto.

Geomedia Warehouse

En las herramientas de Geomedia se accede de la misma manera a todos los almacenes de datos SIG, a excepción de los formatos *raster*. Entendemos por almacén Geomedia (Geomedia Warehouse) el conjunto de formatos propios de Geomedia para el almacenamiento de datos. Geomedia también soporta formatos a otras compañías, como el formato Shapefile, pero no los podemos considerar almacenes propios de la herramienta.

GML

Geography Markup Language (GML) es una gramática XML definida por OGC para almacenar y transmitir información geográfica. GML se ha establecido como el estándar para el intercambio de información geográfica en las transacciones por Internet. Una de las claves del éxito de GML, aparte del hecho de ser un estándar OGC, es la habilidad de integrar todos los tipos de datos geográficos, desde objetos discretos o información vectorial hasta *rasters* o coberturas.

GML define dos conceptos principales. Por una parte, las entidades son objetos que representan una realidad física, por ejemplo, un edificio, una carretera o un árbol. Por otra parte, los objetos geométricos definen una localización o región. Muchas herramientas SIG hacen converger estos dos conceptos en uno solo donde las entidades tienen un objeto geométrico que las ubica en el mundo. GML es más flexible y permite que una entidad no tenga varios objetos geométricos o ninguno que la representen.

ESRI ArcSDE

ESRI es una de las compañías líder en el mercado del SIG, sobre todo para administraciones públicas y gestión ambiental. El paquete de herramientas SIG de ESRI se denomina ArcGIS y está formado por una decena de productos con el prefijo Arc. Hay varias herramientas dentro de ArcGIS. Las principales son:

- ArcMAP: herramienta de visualización y análisis de datos geográficos.
- ArcCatalog: herramienta de tratamiento y organización de datos geográficos.
- ArcIMS: servidor de mapas por internet.

- ArcGIS Server: servidor de mapas y procesamiento geográfico por internet.
- ArcSDE: conector del resto de herramientas con almacenes sobre bases de datos ralacionales.

PostGIS

PostGIS es el primer desarrollo de código libre para el almacenamiento de geometrías sobre una base de datos relacional. Añade el soporte para datos geográficos sobre el SGBD PostgreSQL.

PostGIS sigue las especificaciones de OGC y ha sido certificado para cumplir el estándar *simple feature type*. Ha sido desarrollado por Refractions Research como proyecto de código libre y todavía está en evolución. Aunque no tiene la potencia de otras opciones comerciales, tiene las funcionalidades principales de almacenamiento y es una opción más que válida para muchos proyectos.

Oracle Spatial

La apuesta de Oracle por los SIG ha sido muy fuerte y se ha situado rápidamente entre uno de los formatos de almacenamiento más escalable, fiable y de alto rendimiento de los disponibles.

Una de las ventajas de este formato es que guarda los datos geográficos como un tipo de campo más y extiende SQL para poder realizar consultas espaciales sobre esta información.

2.5.3 Formato Shapefile

Aunque ya ha sido presentado el formato de los archivos Shapefiles, es imprescindible profundizar en sus detalles más técnicos debido a la importancia que suponen para el presente PFC. Como hemos comentado con anterioridad, toda la información de las redes PRIME desplegadas en campo que queremos simular en el mismo está contenida en este tipo de archivos.

Siguiendo la descripción técnica facilitada por ESRI [8], sabemos que los Shapefiles consisten en un conjunto de tres archivos elementales capaces de definir formas como puntos, líneas y áreas:

- *Main File* (.shp): Es un archivo con registros de longitud variable que describe una forma con la lista de sus vértices.
- *Index File* (.shx): Cada registro contiene un *offset* correspondiente a un registro del *Main File* para establecer la relación.
- *dBASE Table* (.dbf): Contiene diferentes características de cada uno de los registros, es decir, se vuelve a establecer una relación uno-a-uno.

Existe una conveniencia de nombres donde los tres archivos deben tener el mismo prefijo. Además, este prefijo debe empezar por un número o por una letra y el resto de caracteres debe contener también letras, números y/o los caracteres "_" y "-".

Otro dato importante son los tipos numéricos utilizados, existiendo el tipo *Integer* (4 Bytes con signo) y el tipo *Double* (8 Bytes con signo).

2.5.3.1 Organización del Main File

Este tipo de archivo contiene una cabecera de longitud fija seguida de varios registros de longitud variable como se ilustra en la Figura 4. Además, cada registro se compone de una cabecera de longitud fija seguida del contenido del registro con una longitud variable:

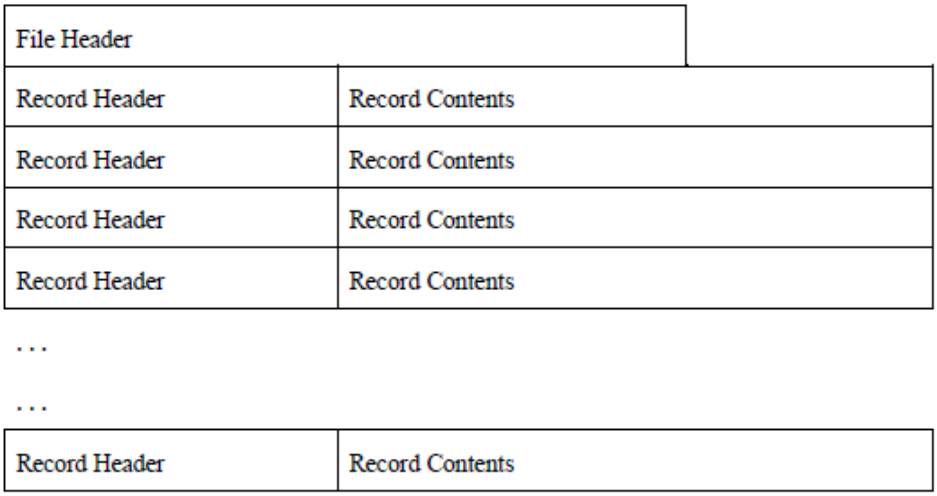


Figura 4: Organización del Main File.

Orden de los bytes:

Todos los contenidos del Shapefile se dividen en 2 categorías:

- Datos relacionados:
  - Contenido de los registros del *Main File*
  - Descripción de los campos de la cabecera del *Main File*
- Gestión de archivos relacionados:
  - Longitud del archivo y de los registros
  - *Offset* de los registros, etc.

Los enteros y los números en coma flotante que definen los campos de la cabecera y los contenidos de los registros están en *Little Endian*. El resto del archivo está en *Big Endian* como veremos a continuación.

Cabecera del Main File:

Esta cabecera tiene una longitud fija de 100 *Bytes*. La Figura 5 muestra sus campos con sus posiciones en bytes, valor, tipo y orden de byte. La posición mostrada es respecto del comienzo del archivo.

Position	Field	Value	Type	Byte Order
Byte 0	File Code	9994	Integer	Big
Byte 4	Unused	0	Integer	Big
Byte 8	Unused	0	Integer	Big
Byte 12	Unused	0	Integer	Big
Byte 16	Unused	0	Integer	Big
Byte 20	Unused	0	Integer	Big
Byte 24	File Length	File Length	Integer	Big
Byte 28	Version	1000	Integer	Little
Byte 32	Shape Type	Shape Type	Integer	Little
Byte 36	Bounding Box	Xmin	Double	Little
Byte 44	Bounding Box	Ymin	Double	Little
Byte 52	Bounding Box	Xmax	Double	Little
Byte 60	Bounding Box	Ymax	Double	Little
Byte 68*	Bounding Box	Zmin	Double	Little
Byte 76*	Bounding Box	Zmax	Double	Little
Byte 84*	Bounding Box	Mmin	Double	Little
Byte 92*	Bounding Box	Mmax	Double	Little

* Unused, with value 0.0, if not Measured or Z type

Figura 5: Descripción de la cabecera del Main File

El valor de *File Length* es la longitud total del archivo en palabras de 16 bits. Es decir, son 50 palabras de 16 bits de la cabecera más la longitud de los registros.

Todos las formas *non-Null* en el Shapefile deben tener el mismo *shape type*. Los valores que puede tomar en función del tipo son los mostrados en la Figura 6. Los valores no especificados (2, 4,6,.., hasta 33) están reservados para usos futuros.

Actualmente, como dijimos anteriormente, todas las formas especificadas en un mismo archivo tienen el mismo tipo pero en el futuro quizá contengan más de un tipo.

Value	Shape Type
0	Null Shape
1	Point
3	PolyLine
5	Polygon
8	MultiPoint
11	PointZ
13	PolyLineZ
15	PolygonZ
18	MultiPointZ
21	PointM
23	PolyLineM
25	PolygonM
28	MultiPointM
31	MultiPatch

Figura 6: Valores del campo Shape Type.

Cabecera de los registros:

La cabecera de cada registro tiene una longitud fija de 8 Bytes y almacena el "número de registro" y la "longitud del contenido del registro" como se detalla en la Figura 7. La longitud reflejada es la longitud del contenido del registro medida en palabras de 16 bits. Es decir, cada registro tiene 4 palabras de 16 bits que representan los campos de "número de registro" y "longitud del contenido" más el número de palabras indicado en el campo "longitud del contenido".

Como último apunte, hay que tener en cuenta que el número de registro empieza siempre en 1.

Position	Field	Value	Type	Byte Order
Byte 0	Record Number	Record Number	Integer	Big
Byte 4	Content Length	Content Length	Integer	Big

Figura 7: Descripción de la cabecera de los registros del Main File

2.5.3.2 Contenido de los registros del *Main File*

El contenido de los registros consiste en la especificación del tipo de forma (*Shape Type*) del registro seguido de los datos geométricos. La longitud del contenido del registro depende del número de partes y vértices que tenga la forma. A continuación describiremos los tipos de forma que son de interés para el presente proyecto y cómo son estructurados.

Point:

Consiste en un par de coordenadas X e Y de tipo *Double* como se muestra en la Figura 8.

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	1	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little

Figura 8: Contenido de un registro de tipo Point.

Multipoint:

Como se muestra en la Figura 9 representa un conjunto de puntos indicando sus coordenadas X, Y además de indicar cuáles son sus delimitadores Xmin, Xmax, Ymin, Ymax (Box).

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	8	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little

Figura 9: Contenido de un registro de tipo Multipoint.

Polyline:

Es un conjunto de vértices ordenados que consisten en una o más partes donde una parte es una secuencia de dos o más puntos conectados. Estas partes quizá estén o no conectadas a otras partes o también pueden tener intersecciones con otras partes. Un ejemplo de su estructura es el que se muestra en la Figura 10.

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	3	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Note: X = 44 + 4 * NumParts					

Figura 10: Contenido de un registro de tipo Polyline.

2.5.3.3 Organización del *Index File*

El *Index File* (.shx) contiene 100 Bytes de cabecera seguidos de diferentes registros con una longitud fija de 8 Bytes como se ilustra en la Figura 11.

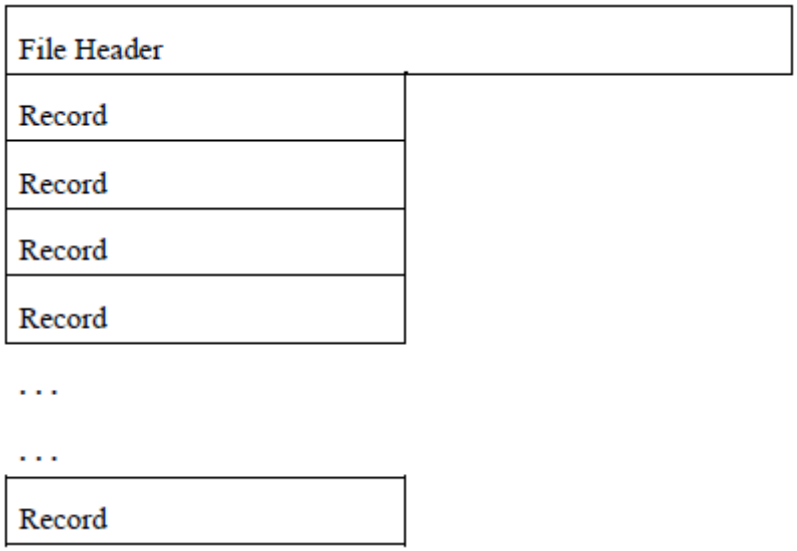


Figura 11: Organización del Index File.

Cabecera del *Index File*:

La cabecera del *Index File* es idéntica en organización a la del *Main File* descrita con anterioridad. En esta ocasión el campo *File Length* vuelve a medirse en palabras de 16 bits por lo que tendrá las 50 palabras correspondientes a la cabecera más 4 veces el número de registros que tenga.

Índice de registros:

El registro i-ésimo almacena el *offset* y la longitud del i-ésimo registro del *Main File*. El *offset* en el *Main File* es el número de palabras de 16 bits desde el principio del archivo hasta el primer *Byte* de la cabecera del registro. El *Content Length* es idéntico al almacenado en la cabecera del registro del *Main File*, tal y como muestra la Figura 12.

Position	Field	Value	Type	Byte Order
Byte 0	Offset	Offset	Integer	Big
Byte 4	Content Length	Content Length	Integer	Big

Figura 12. Descripción del índice de registros.

2.5.3.4 Organización del *dBase File*

El *dBASE File* (.dbf) contiene cualquier atributo deseado o enlaces a otras tablas para poder unirlas. Su formato es el estándar utilizado en muchas aplicaciones de Microsoft y DOS donde cualquiera de sus campos puede ser representado en una tabla. Existen tres requisitos:

- El nombre debe tener el mismo prefijo que el *Main File* y el *Index File*.
- La tabla debe contener un registro por cada característica de la forma.
- El orden de los registros debe ser igual al orden de las formas en el *Main File*.
- El valor del año en la cabecera de la *dBASE* debe ser a partir de 1900.

Capítulo 3

Diseño

En este capítulo se describe el proceso de diseño y la metodología de trabajo seguida en el presente proyecto. Además, se proporciona una visión global de la aplicación con el fin de contextualizar cada uno de los módulos explicados detalladamente en el capítulo 4. Para ello, se presenta en primer lugar la metodología establecida por el *Project Management Institute* (PMI), cuyos principios nos obligan a entrar en detalle en los requisitos y en la planificación del desarrollo del proyecto. Finalmente trataremos de buscar un enlace con los aspectos más técnicos del proyecto presentando las principales decisiones de diseño que se han tomado.

3.1 Metodología

Por definición, un proyecto es una planificación que consiste en el conjunto de actividades interrelacionadas y coordinadas con el fin de alcanzar unos objetivos específicos dentro de unos límites de tiempo, calidad y presupuesto. En función de esos objetivos, hay múltiples caminos para llegar a ellos por lo que puede aumentar el grado de complejidad de su desarrollo. Este hecho nos lleva a seguir los pasos definidos por el *Project Management Institute* (PMI) para un proyecto de desarrollo de *software*.

PMI es una organización internacional no lucrativa cuyo objetivo es convertir la gerencia de proyectos en una actividad indispensable para obtener resultados en cualquier actividad de negocio. En la práctica, es un grupo de profesionales de la gerencia de proyectos que se dedican a promover el desarrollo del conocimiento y las competencias básicas para el ejercicio profesional.

El PMI define la gestión de proyectos como la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades de un proyecto para satisfacer sus requisitos. Para que la gestión de proyectos sea satisfactoria es imprescindible partir de una planificación coherente y que permita alcanzar los objetivos del proyecto optimizando la asignación y coste de recursos.

El más famoso y reconocido producto del PMI es el *Project Management Body of Knowledge* (PMBOK). Como su nombre sugiere, describe un conjunto de conocimientos y de prácticas aplicables a cualquier situación que se quiera formular. No debe entenderse como una metodología *per se*, sino como una guía de estándares internacionales para que los profesionales puedan adaptar a cada caso y contexto particular los procesos reconocidos como buenas prácticas por el PMI.

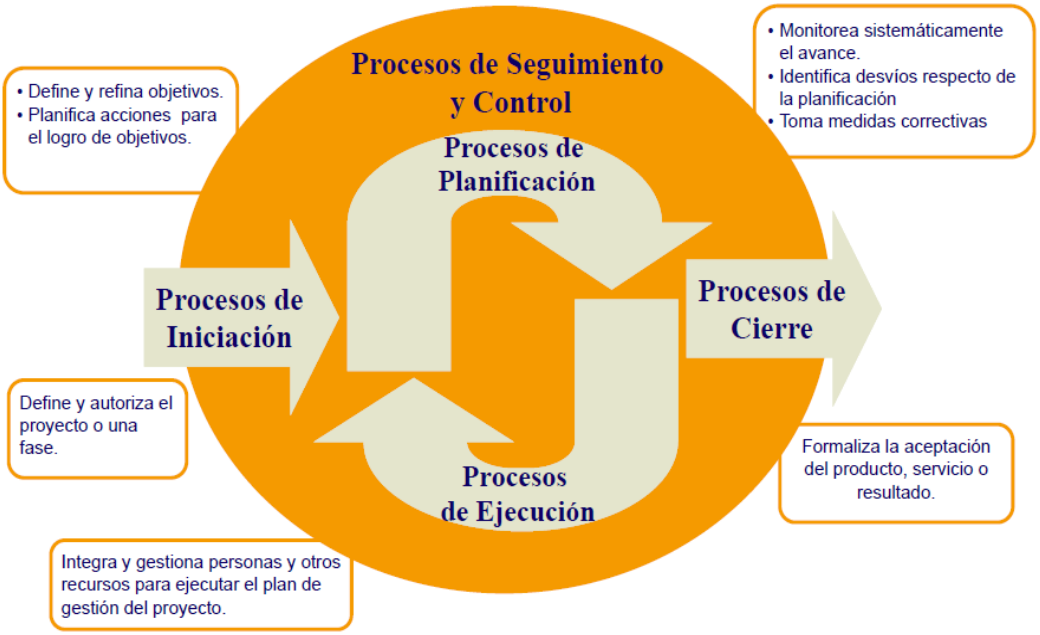


Figura 13: Grupos de procesos PMI.

Como se resume en la Figura 13, la importancia del PMBOK es que provee un marco de referencia formal para desarrollar proyectos, guiando y orientando sobre la forma de avanzar en los procesos y pasos necesarios para la construcción de resultados y alcanzar los objetivos. A diferencia de los modelos tradicionales, es una metodología ágil definida por un grupo de procesos aplicables a diferentes áreas de conocimiento, mostradas en la Figura 14.

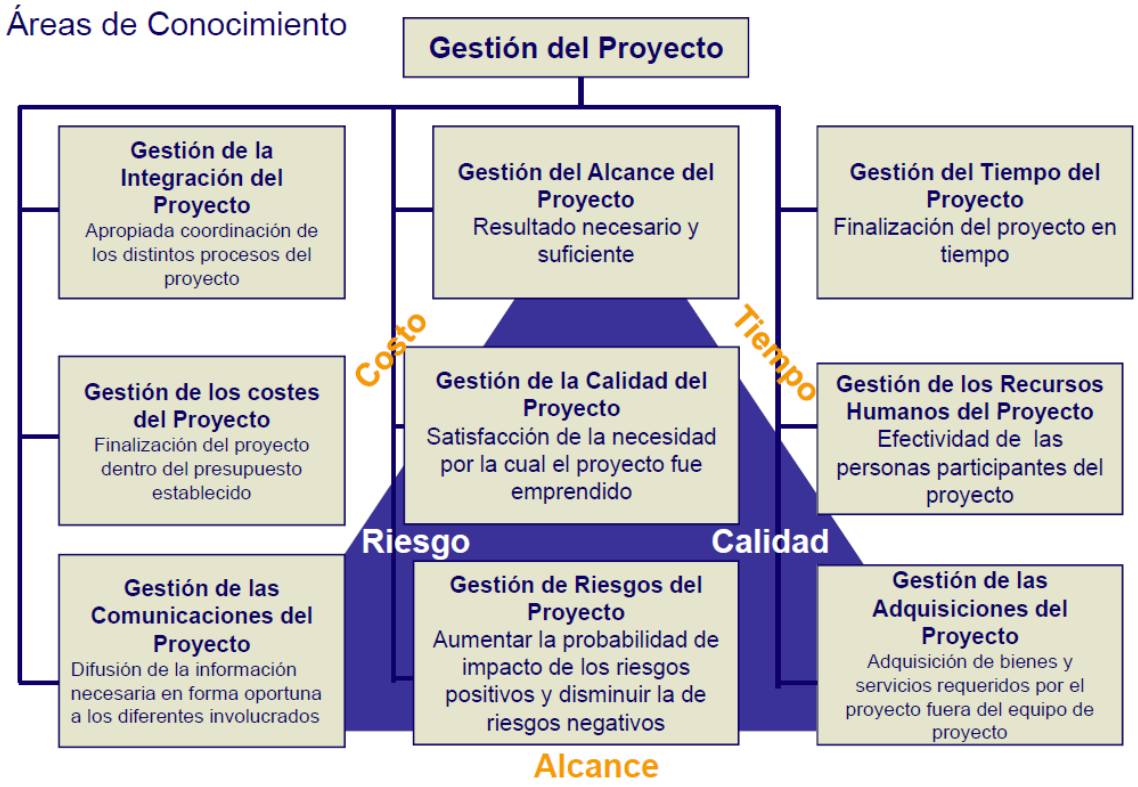


Figura 14: Áreas de conocimiento PMI.

De la confluencia de los procesos centrales y de las áreas de conocimiento se crea una matriz de procesos que representan las directrices de la metodología PMI [35]. Esta matriz de procesos ha sido el punto de partida de este proyecto, donde el alcance del proyecto y el tiempo del proyecto han sido las áreas de conocimiento principales. Si tratamos de concretar un poco más, todo el desarrollo del proyecto se ha regido por el cumplimiento de una serie de requisitos y de una planificación, hecho que nos obliga a entrar más en detalle en estos dos aspectos en las próximas secciones.

3.2 Requisitos

Los requisitos que definen este proyecto vienen inicialmente establecidos por los antecedentes que ya fueron presentados en el capítulo 1. El objetivo final es la incorporación de un módulo *software* de análisis de mapas en formato Shapefile en un simulador de redes PRIME ya implementado y mejorado mediante una aplicación Web, por lo que partimos de los requisitos de dicho entorno [36]. Éste, como podemos ver en la Figura 15, consta de unas entradas, unas salidas y un diseño que nos obligan a adaptarnos a una aplicación cerrada.

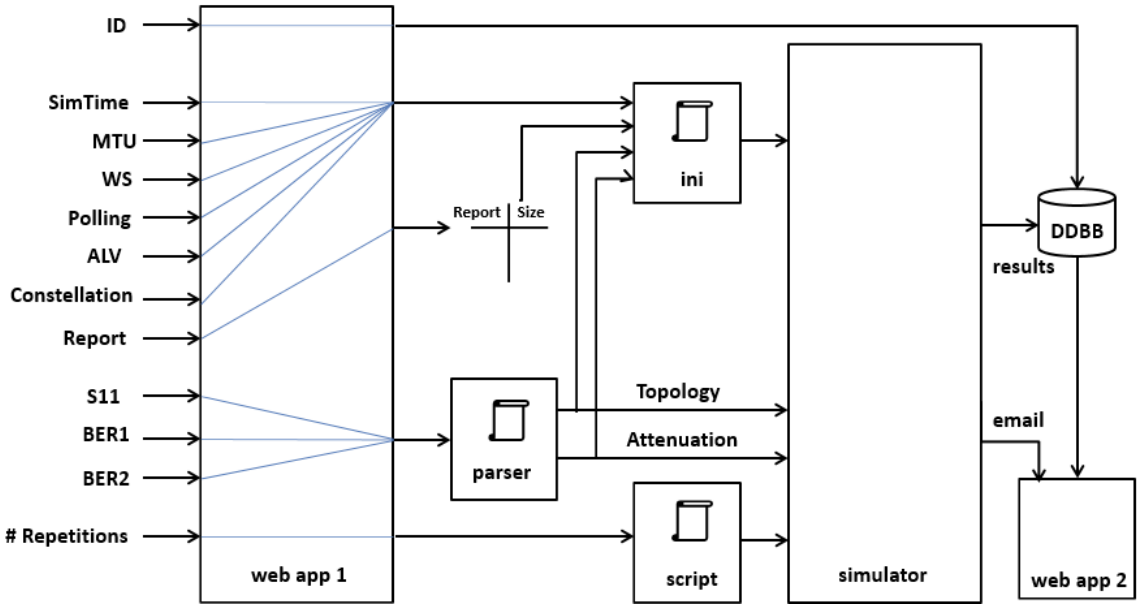


Figura 15: Aplicación Web y simPRIME.

Entre los requisitos cumplidos por dicha aplicación, debemos destacar uno de ellos: "El sistema deberá generar la matriz de atenuaciones de los pares de nodos con el formato requerido por el simulador". Esta matriz de atenuaciones representa la atenuación entre cada contador con el resto de contadores y con el concentrador (situado en el centro de transformación), es decir, depende de la topología de la red en estudio.

La aplicación de la que partimos calcula todas las atenuaciones a partir de los campos Constelación, BER1 y BER2 proporcionados por el usuario. Combinando esta información, se obtiene la SNR y, conocidas la potencia transmitida y la del ruido de fondo, a su vez se obtiene la atenuación. Se sigue este proceso ya que la simulación se lleva a cabo sobre una topología lógica fija, justo lo contrario al caso que nos ocupa.

Por lo tanto, ya que los Shapefile definen una topología física real, se hace necesario realizar el cálculo de la matriz de atenuaciones a partir de dichos planos, siendo éste el requisito fundamental del presente proyecto. Además, ya que esta funcionalidad debe estar integrada en la aplicación existente, tenemos una serie de requisitos que se hacen imprescindibles para la correcta implementación del sistema:

Requisito I: Utilización de Python y Django como herramientas de desarrollo.

La aplicación web de la que partimos está desarrollada bajo el entorno Django que, como dijimos en el cápitulo 2, es un *Web framework* escrito en Python. Aunque técnicamente no es imprescindible, la utilización de Python para el cálculo de la matriz de atenuaciones se toma como requisito debido a que facilita la integración del código desarrollado y su posterior mantenimiento. Esta tarea se podría llevar a cabo en otros lenguajes de programación como C ya que existen módulos que permiten la integración de aplicaciones escritas en C sobre entornos Python [37].

En cuanto a la utilización de Django, ésta es imprescindible ya que de lo contrario habría que rehacer la aplicación web bajo otro tipo de *framework.* Además, Django cuenta con una gran sencillez y versatilidad que permite modificar ciertas partes de su código con gran simplicidad.

Requisito II: La matriz de atenuaciones debe ser calculada en base a los Shapefile facilitados.

Como hemos comentado anteriormente, el hecho de que nuestra aplicación consista en la simulación de redes PRIME a partir de unos Shapefiles hace que sea fundamental el cálculo de la matriz de atenuaciones a partir de estos.

Para ello, debemos ser capaces de extraer la información necesaria como los nodos que forman el plano, los materiales utilizados, la longitud de sus tramos, etc, tomando como referencia los archivos facilitados por Unión Fenosa.

Requisito III: El cálculo de la matriz de atenuaciones debe tener en cuenta únicamente contadores inteligentes.

En la realidad, las redes PRIME funcionan con contadores inteligentes que permiten la conexión permanente con concentrador situado en el centro de transformación. Ya que los Shapefiles corresponden a planos de instalaciones reales, hay casos en los que las viviendas tienen aún contadores tradicionales. Esta situación condiciona el cálculo de la matriz de atenuaciones ya que hay que diferenciar ambos tipos de contadores, pues pueden presentar diferentes impedancias y, además, los tradicionales no se tienen en cuenta a la hora de simular la red PRIME.

La dificultad de este requisito es que los Shapefile no contienen esta información por lo que se hace necesario buscar otra forma de extraer estos datos.

Requisito IV: El nuevo módulo debe ser integrado correctamente en la aplicación existente.

Este requisito es evidente ya que de cara al usuario la aplicación simplemente debe ganar una funcionalidad que no condicione las que ya existían.

3.3 Planificación

Como se ha visto en la sección 3.1, la metodología utilizada para el desarrollo de este proyecto se ha basado en el cumplimiento de unos requisitos y de una planificación. Esta última fue acordada con coherencia como punto de partida, definiéndose y secuenciándose las principales fases del proyecto con sus correspondientes tareas y realizándose una estimación inicial de la duración de cada una de estas fases. En el Anexo II se detalla dicha planificación indicándose la duración real de dichas fases.

3.4 Diseño

Como hemos comentado en varias ocasiones, el objetivo de este proyecto es la incorporación de una nueva funcionalidad a un simulador Web de redes PRIME ya desarrollado. Esta situación limita nuestras posibilidades de diseño ya que debemos adaptarnos al esquema presentado en la Figura 15. No obstante, estas limitaciones solamente afectan a la integración del nuevo módulo en la aplicación Web y no al desarrollo del propio módulo. Es decir, el proceso principal de diseño se ha focalizado en el desarrollo del tratamiento de los Shapefiles para la generación de la matriz de atenuaciones.

La Fgura 16 presenta un diagrama de la aplicación Web junto con el simulador y el nuevo módulo incorporado, con el objetivo de obtener una vision global del sistema y de la localización del nuevo módulo dentro del mismo. Ya que uno de los requisitos era la correcta integración de la nueva funcionalidad, se ha tratado de replicar el diseño existente para trabajar con los Shapefile de forma similar a como se hace con el archivo de topologías en formato XML (S11.XML) que recibe de entrada la aplicación original.

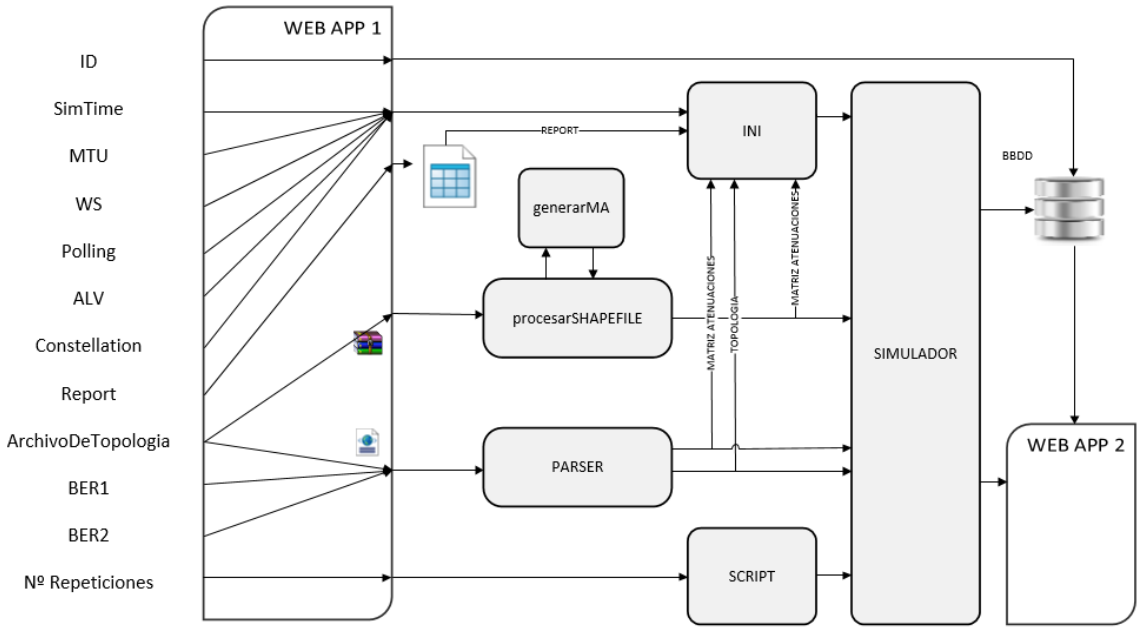


Figura 16: Aplicación Web, simPRIME y módulo para procesar Shapefiles.

Para ello, el primer paso es diferenciar el tipo de archivo de entrada en la aplicación ya que si es un .ZIP asumiremos que contiene Shapefiles en su interior y comenzaremos el proceso para la generación de la matriz de atenuaciones. En el caso contrario, se seguirá el proceso que ya habia sido desarrollado en la aplicación original. No obstante, a continuación se enumeran las decisiones de diseño tomadas para la correcta integración del nuevo módulo, asi como los motivos que nos han llevado a elegirlas.

3.4.1 Integración del nuevo módulo

Decisión I. Diferenciación del tipo de archivo de topología

Como es lógico, para realizar el estudio de los Shapefiles es necesario que éstos archivos sean facilitados. El problema es que, como vimos en el capitulo 2, un Shapefile es un conjunto de tres o mas archivos interrelacionados entre sí por lo que deben ser subidos a la aplicación como un conjunto.

Por otro lado, los archivos que proporciona Unión Fenosa para cada plano, siempre estan compuestos de un archivo Excel y varios Shapefiles: gdotramo_baja, gdoacom, gdoct y gdoapoyo_baja. Este hecho propició la decición de incluir todos estos archivos dentro de un mismo .ZIP.

Por lo tanto, diferenciar el tipo de archivo de entrada definirá el proceso que realizará la aplicación.

Decisión II. Generación de datos necesarios para el simulador

El objetivo principal de la aplicación original de la que partimos es actuar como interfaz gráfica del simulador de redes PRIME simPRIME. Para ello, en su desarrollo fue vital conocer aquellas entradas necesarias que permitieran realizar la simulación, por lo que, a efectos prácticos, dicho simulador puede considerarse como el módulo que ilustra la Figura 17.

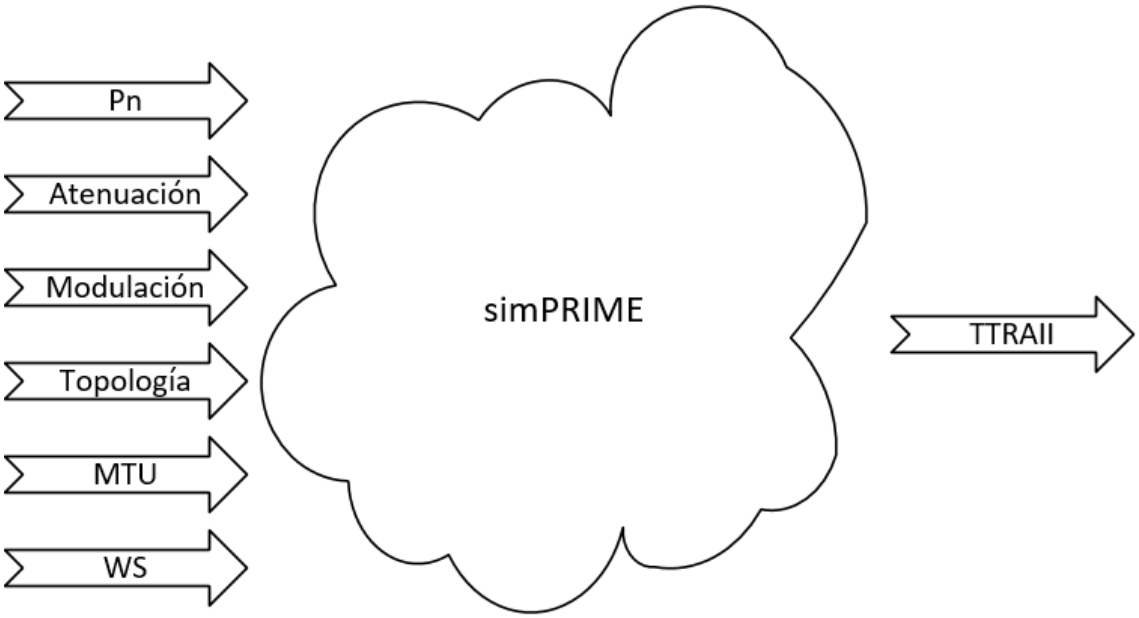


Figura 17: Resumen gráfico del simulador simPRIME.

En la aplicación original, todas las entradas del simulador son generadas mediante el módulo “PARSER” y directamente introducidas por el usuario. Ya que el simulador esta basado en OMNeT++, estas entradas se utilizan para establecer la configuración del archivo .INI necesario para el correcto arranque y funcionamiento del mismo.

Todas estas condiciones necesitan ser cumplidas tambien cuando queremos iniciar una simulación a partir de unos Shapefiles. Ya que en este caso el proceso es diferente pero se quería seguir la misma filosofia de diseño, se decidió introducir el módulo "procesarSHAPEFILE" como alternativa del módulo "PARSER". De esta forma, el nuevo módulo será el que actúe como enlace entre la entradas de la aplicación y las entradas del simulador.

Decisión III. Generación de la matriz de atenuaciones

Ya que la generación de la matriz de atenuaciones a partir de los Shapefiles es una tarea compleja, se decidió introducir un nuevo módulo, denominado "generarMA" (ver Figura 16), que se encargará única y exclusivamente de esta función. Puesto que la matriz de atenuaciones es una de las entradas del simulador, desde "procesarSHAPEFILE" se realiza la petición para la generación de dicha matriz para posteriormente dar el formato correcto a los resultados proporcionados por "generarMA".

3.4.2 Construcción del nuevo módulo

Como dijimos al principio de esta sección, nuestras tareas principales tanto de diseño como de desarrollo se han centrado en cómo procesar los Shapefiles adecuadamente. Aunque aquí hemos presentado primero las decisiones referentes a la integración con la plataforma existente, en el desarrollo real primero se diseñó e implementó todo lo relativo a la generacion de la matriz de atenuaciones. El orden presentado en este documento tiene la intención de facilitar la comprensión del desarrollo realizado.

A continuación se presentan las deciones tomadas para la correcta generación de la matriz de atenuaciones a partir del archivo .ZIP facilitado.

Decision IV. Adquisición de los archivos necesarios para el proceso

Como se ha indicado anteriormennte, para la generación de la matriz de atenuaciones partimos de un archivo .ZIP facilitado por el usuario. Este hecho nos obliga a extraer los archivos que se encuentran en su interior para poder procesar los Shapefiles.

Otro factor a tener en cuenta es el hecho de que estos archivos, una vez procesados, son innecesarios por lo que su almacenaje es inútil. Por lo tanto, estos serán extraidos en una carpeta temporal que será eliminada tras la creación de la matriz de atenuaciones.

Decision V. Procesamiento de los Shapefiles para obtener la información.

Una vez extraídos los Shapefiles comprimidos en el .ZIP, el siguiente paso es obtener la información contenida en ellos, procesarla y almacenarla durante la ejecución. Para ello hay que tener en cuenta como se estructura y qué contiene cada Shapefile ya que de entrada son un montón de bits ininteligibles. No obstante, esto no

supone ningún problema ya que las caracteristicas de este tipo de ficheros fueron vistas en el capitulo 2.

Ya que los Shapefiles facilitados contienen un conjunto de nodos que forman tramos al unirse y estos tramos forman grafos al conectarse, se tomó la decisión de crear los objectos Nodo, Tramo y Grafo. Es decir, la información se ha estructurado utilizando el potencial de Python como lenguaje de programación orientado a objetos.

Además, ya que cada plano en estudio contiene varios Grafos que a su vez involucran a los Tramos y a los Nodos, se hace necesario almacenar cada uno de estos objectos. Para ello, como se muestra en la Figura 18, se tomó la decisión de identificarlos y contenerlos en vectores de tal forma que pudiera accederse a cada objeto si fuera necesario.

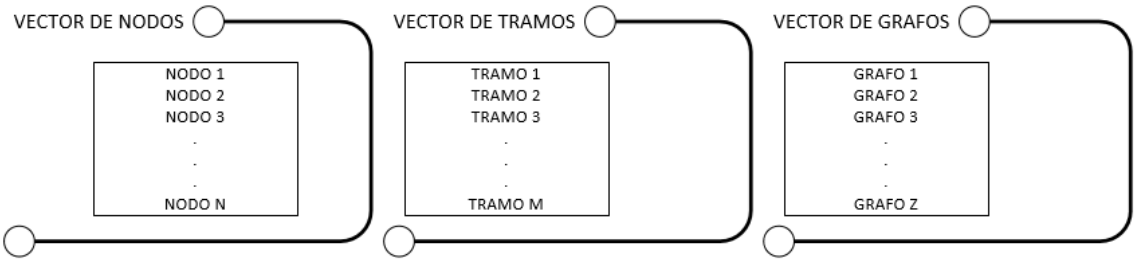


Figura 18: Estructura de la información obtenida a partir de los Shapefiles.

Decisión VI. Procesamiento de archivo Excel para obtener el tipo de contador.

Uno de los requisitos nos imponía con lógica que la matriz de atenuaciones únicamente debe calcularse entre contadores inteligentes. Ya que esta información no esta contenida en los Shapefiles pero si en un archivo Excel tambien proporcionado, se hizo necesaria su incorporación al archivo .ZIP que se indicó anteriormente.

Por lo tanto, una vez extraído dicho Excel en la carpeta temporal sobre la que trabajamos, accemos a su información con el fin de asignar a cada nodo contenido en el vector de nodos el número de contadores inteligentes y no inteligentes que contiene.

Decisión VII. Generación de Shapefiles del escenario en estudio

Los Shapefiles facilitados cuentan con diferentes planos y estos a su vez se dividen en lo que se denomina como trafos. Estos trafos son un conjunto de tramos con un identificador particular relacionado con el identificador del plano. Es decir, un Shapefile puede contener los planos PL01 y PL02 donde el primero de ellos está formado por los trafos PL01-1 y PL01-7, y el segundo por PL02-2 y PL02-5.

Esta forma de organizar los planos ocasionó que el procedimiento para analizar la información se realizara trafo o trafo, o lo que en nuestro caso es igual a grafo a grafo. Esta correspondencia entre trafo y grafo fue un punto de inflexión del diseño ya que se pasaba de tratar el plano completo a tratar sólo partes de él.

Como veremos en el capítulo siguiente, este hecho, unido a la posibilidad de que hubiese bucles, hizo necesario la implementación de una función que permitiese generar un Shapefile del grafo en estudio. Es decir, se tomó la decisión de tener la posibilidad

de generar un Shapefile de un cierto grafo como herramienta para poder visualizar únicamente las caracteristicas de éste.

Decisión VIII. Generar una estructura de arbol con los datos recopilados

Una vez estructurada en forma de grafo toda la información recogida en los Shapefiles, debemos procesarla para obtener la matriz de atenuaciones buscada. Ya que para calcular las atenuaciones entre contadores se hace necesario recorrer el camino que los separa, la estructura de grafo no es suficiente por la posibilidad de existir bucles. Es decir, se tomó la decisión de estructurar la información en forma de árbol ya que con los grafos podría haber varios caminos para llegar de un nodo a otro y esto, aunque físicamente pueda ocurrir en una red eléctrica de distribución, no ocurre realmente en explotación (típicamente los bucles no existen realmente ya que se utilizan relés).

Para poder estructurar nuestros datos como un árbol, se hizo necesaria la introducción de ciertos parámetros para que cada nodo pudiera relacionarse ordenadamente con el resto. Además, para evitar la existencia de bucles, se utilizó el Algoritmo de Dijkstra [38], proceso que veremos en el próximo cápitulo.

Como vemos en la Figura 19, el proceso de pasar de grafo a árbol no sólo tiene la ventaja de la eliminación de los de bucles, sino que también se pueden simplificar los árboles quedándonos únicamente con los nodos relevantes.

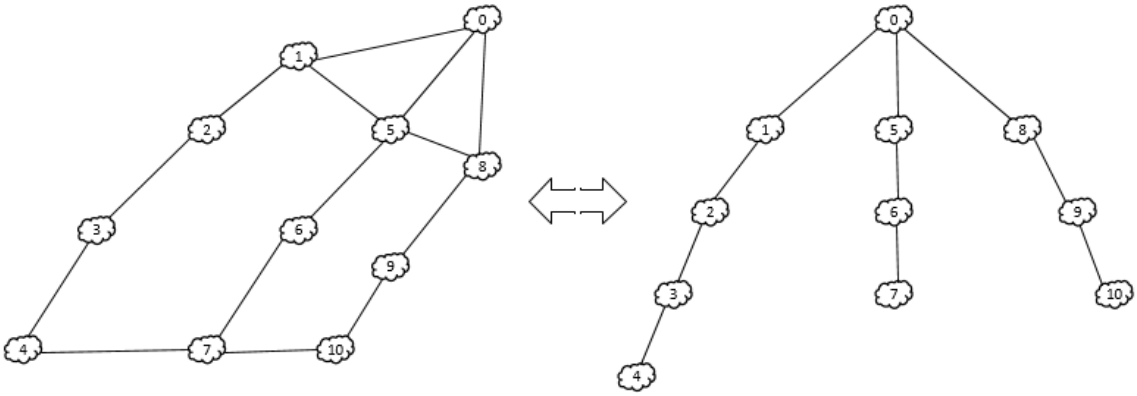


Figura 19: Grafo frente a árbol.

Decision IX. Calculo de impedancias de cada rama del arbol

Llegados a este punto, ya estamos en condiciones de comenzar a preparar el escenario que nos permitirá realizar el calculo de atenuaciones. Para ello es necesario recorrer el árbol inicialemente de abajo a arriba y finalizar recorriéndolo de arriba abajo, según se detalla en el Anexo I.

Recorriendo cada rama del árbol podremos calcular la impedancia que supone avanzar de un Nodo a otro teniendo en cuenta que el comportamiento físico del cable es el de una linea de transmisión. De esta forma, guardando cada una de estas impedancias, podremos calcular posteriormente la atenuación que supone ir de cierto nodo a otro siguiendo el algoritmo detallado en el Anexo I.

Decisión X. Calculo de la matriz de atenuaciones

Finalmente, una vez asignadas las impedancias de cada rama en ambos sentidos, tenemos lo necesario para realizar el calculo de la matriz de atenuaciones. Éste es uno de los pasos críticos junto con el calculo de impedancias por lo que se buscó la simplicidad mas que la eficiencia del código.

Como hemos comentado anteriormente, el desarrollo se ha llevado a cabo estudiando cada trafo por separado, por lo que llegados a este punto, tuvimos que integrar todos los trafos en uno. Sólo de esta manera era posible relacionarlos y obtener la matriz de atenuaciones por lo que tuvimos que conectarlos con el único Nodo que tienen en común todas las ramas: el centro de transformación (i.e., el concentrador).

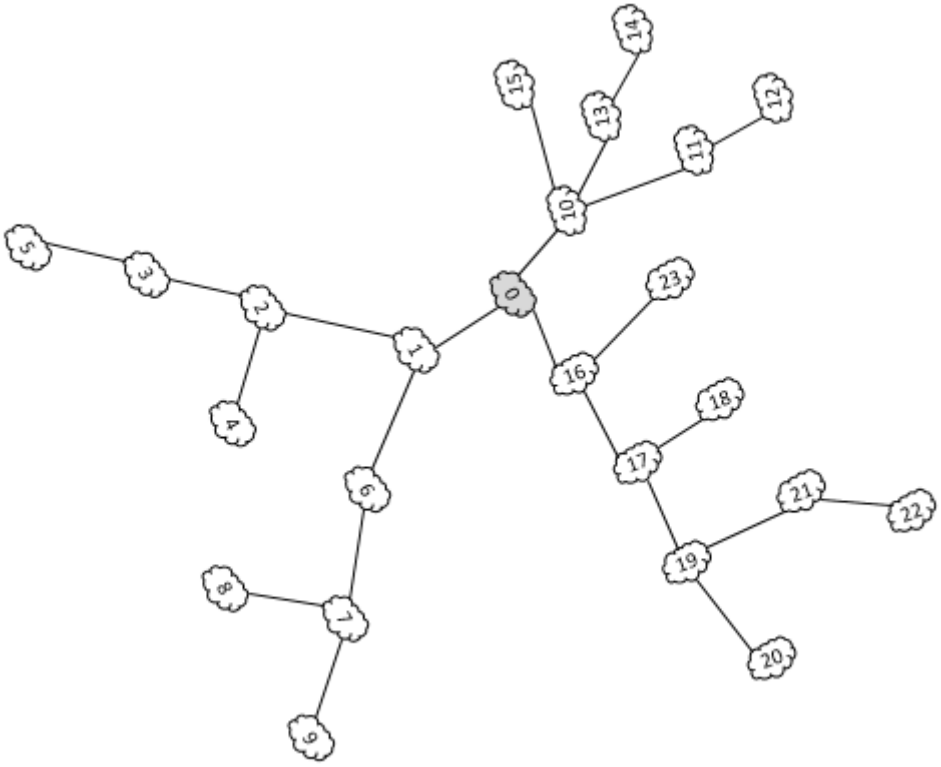


Figura 20: Árbol de trafos.

Aunque según la Figura 20 esta idea es sencilla, su implementación es compleja y laboriosa por lo que su explicación ocupará gran parte del proximo cápitulo.

3.4.3 Diagrama de bloques del nuevo módulo

Como comentábamos en la introducción de este capítulo, el objetivo de esta última sección es mostrar una visión global del proyecto para facilitar la comprensión de los aspectos más técnicos del proyecto, abordados en el próximo capítulo. Aunque ya hemos introducido ciertos detalles relevantes, parece imprescindible presentar un diagrama que pueda facilitar la comprensión del desarrollo llevado a cabo.

La Figura 21 ilustra en forma de diagrama cada módulo y/o función que han permitido la generación de la matriz de atenuaciones. Además, se intenta mostrar como

interactúa cada módulo con los demás con el fin último de orientar al lector en cada una de las secciones del Capítulo 4.

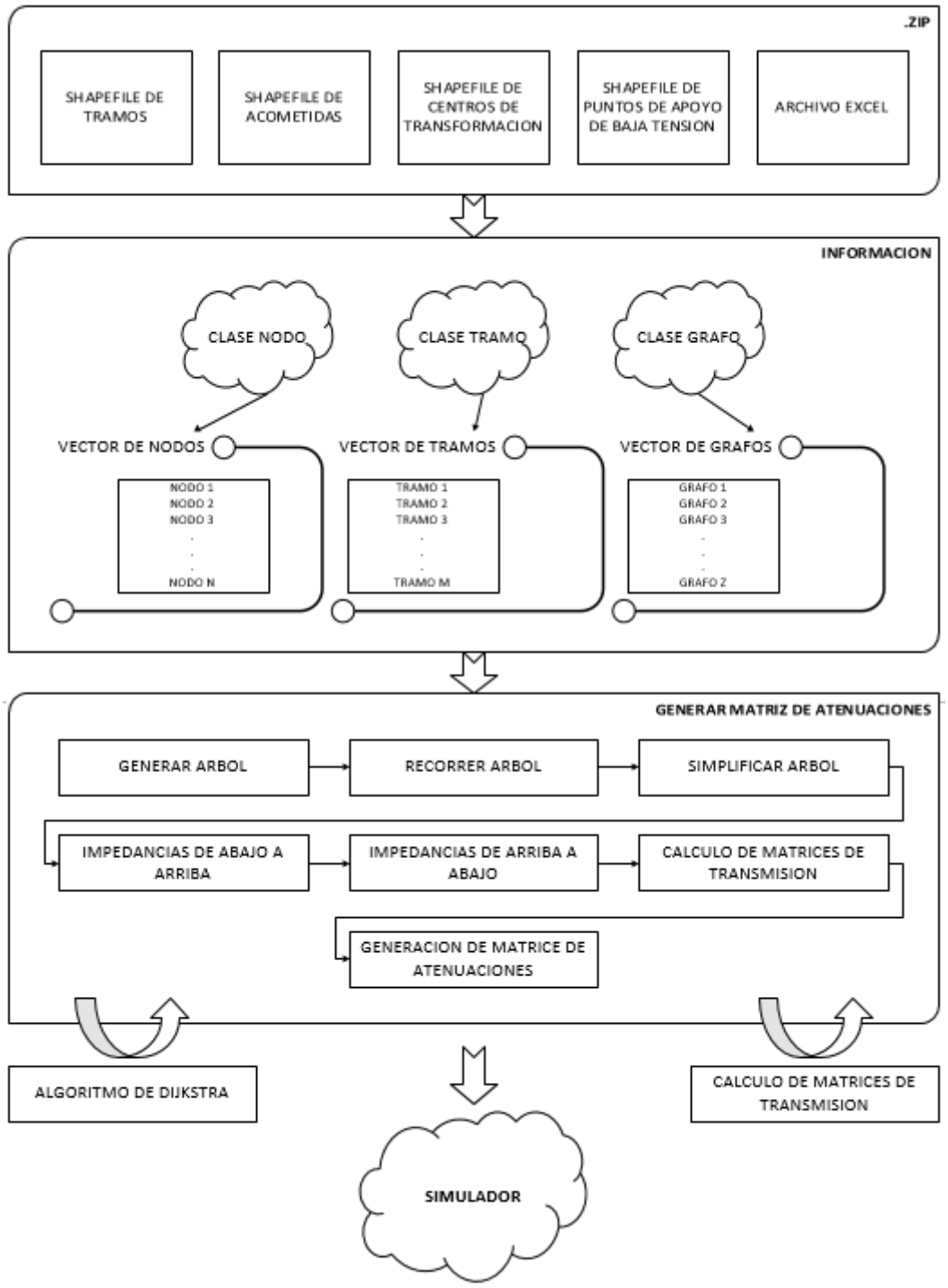


Figura 21: Diagrama del proceso de generación de la matriz de atenuaciones.

Capítulo 4

Desarrollo

El objetivo de este capítulo es presentar los aspectos más técnicos del desarrollo, tomando como referencia el diseño presentado en el capítulo anterior. Así, en este capítulo se detalla la implementación de cada uno de los módulos en el orden que fueron desarrollados para permitir al lector comprender el proceso de forma clara y sencilla.

4.1 Generación de matriz de atenuaciones

Partiendo de las indicaciones presentadas en el capítulo anterior y utilizando como referencia el diagrama de la Figura 21, nos disponemos a detallar cada uno de los pasos seguidos durante el desarrollo para la generación de la matriz de atenuaciones.

4.1.1 Adquisición de los archivos necesarios para el proceso

Cuando realizamos la petición de generar la matriz de atenuaciones ("GenerarMA") desde el módulo "procesarSHAPEFILES", éste nos proporciona el nombre del ZIP que contiene los archivos necesarios para el desarrollo y la ruta de la carpeta donde permanecerán estos archivos durante la ejecución. Es decir, la extracción del ZIP ya ha sido realizada cuando comienza la ejecución de "GenerarMA".

No obstante, al comienzo del desarrollo, este módulo funcionaba de forma autónoma. Al no existir aún integración con la aplicación Web, la función "GenerarMA" recibía la ruta donde se encontraba el ZIP y se realizaba la extracción. Este planteamiento forzaba a que el ZIP tuviera un formato cerrado como el de la Figura 22 para así evitar factores de error.

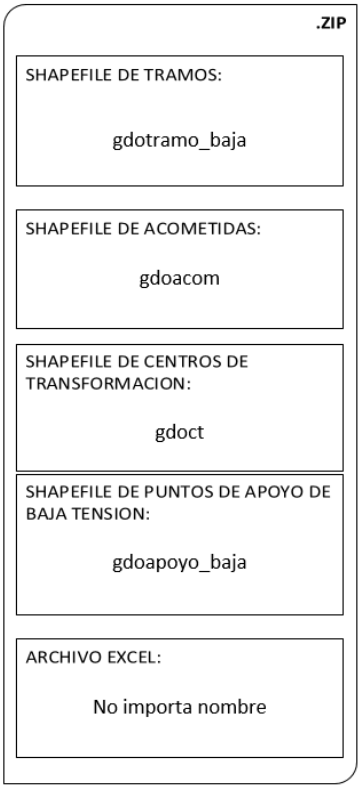


Figura 22: Formato del archivo de entrada.

Además de respetar este formato, el nombre del ZIP debe ser el código del plano que se quiere estudiar. Este hecho es importante ya que dentro de un mismo Shapefile puede haber varios planos y usaremos este nombre para centrar el estudio sobre el plano adecuado sin una intervención extra por parte del usuario.

Una vez conocido el formato del ZIP, la extracción se lleva a cabo gracias al módulo de Python Zipfile [39]. Este módulo proporciona herramientas para crear, leer, escribir, añadir y listar un archivo ZIP. Por lo tanto, ya que conocemos la ruta del archivo comprimido, simplemente debemos guardar su nombre, leer su contenido con este módulo y depositarlo en la carpeta de destino elegida.

Debemos recordar que los archivos se depositan sobre una carpeta temporal ya que son útiles únicamente durante la ejecución del código. Para ello, simplemente hacemos uso de las características que proporciona el módulo del sistema de Python "os" para obtener la ruta actual de trabajo y así poder crear la carpeta que denominamos "TEMPORAL".

Al realizar la integración, proceso ilustrado en la Figura 23, fue cuando se pasó a realizar el proceso de extracción inmediatamente después de subir el archivo a la plataforma. Ya que la aplicación está desarrollada en Django y éste está desarrollado en Python, el proceso de extracción es similar al explicado anteriormente, sólo cambia el lugar del código donde se realiza. Por este motivo, el módulo "GenerarMA" simplemente necesita conocer el nombre del ZIP y el lugar donde se ha extraído su contenido.

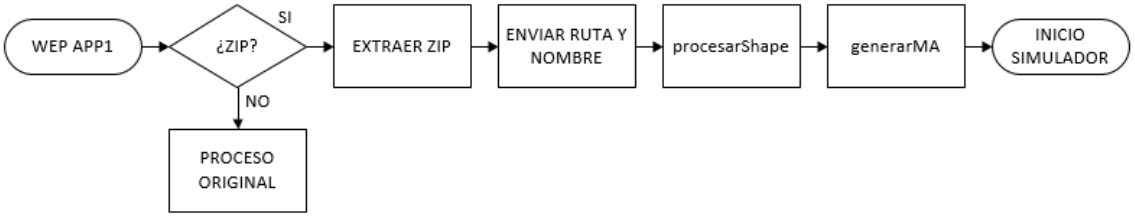


Figura 23: Explicación del proceso de extracción.

Finalmente, una vez extraídos y localizados los archivos necesarios podemos comenzar con el proceso de lectura de los Shapefiles y del archivo Excel.

4.1.2 Procesamiento de los Shapefiles

Como vimos en el capítulo 2, los Shapefiles cuentan con numerosos datos que son relevantes para el desarrollo de este proyecto. Estos datos, aunque con una estructura muy bien definida, están representados en formato binario, es decir, leer esta información desde Python representa una tarea ardua. Este hecho, unido a la gran comunidad de Python formada en los últimos años, nos hizo plantearnos la existencia de un módulo que pudiera facilitarnos esta labor.

El resultado de esta búsqueda fue el módulo Python Shapefile [40]. Éste proporciona la capacidad de leer y escribir archivos shp, shx y dbf de todo tipo de geometrías. No obstante, la familiarización con las funciones de este módulo no es una

tarea inmediata ya que es imprescindible conocer cómo se estructuran los Shapefiles y cómo son interpretados por el citado módulo.

Como vimos anteriormente, los planos que Unión Fenosa ha facilitado siempre están compuestos por un Shapefile de tipo Polyline y dos o tres de tipo Point (dependiendo de si hay o no puntos de apoyo de baja tensión). Esto se traduce en que debemos leer cada archivo de forma individual en función de su formato para posteriormente hacer que esta información converja a una única estructura.

Ya que el elemento principal de los Shapefiles son los puntos que lo forman, se tomó la decisión de definir en primer lugar el concepto Nodo. Para este desarrollo, un nodo es el elemento principal que representa un punto en el espacio con capacidad de interconectarse con otros nodos y, además, guarda información relevante del lugar que representa. Por tanto, este elemento está presente en todos los Shapefiles y la información que contengan es la que definirá su función y/o emplazamiento.

Para los Shapefiles de tipo *Point* con la representación en forma de nodo es suficiente, pero el archivo de tipo *Polyline* contiene puntos interconectados por lo que se hace necesario definir la estructura Tramo. Ésta consiste en la conexión ordenada de varios nodos de tal forma que la unión de dos o más nodos forman segmentos interconectados como vemos en la Figura 24.

Ya que para los Shapefiles de tipo *Polyline* cada registro corresponde con un tramo y el conjunto de ciertos registros con el mismo identificador están conectados, definimos esta interconexión como Grafo. Es decir, un grafo engloba toda la información contenida en los archivos Shapefiles facilitados, como también muestra la Figura 24.

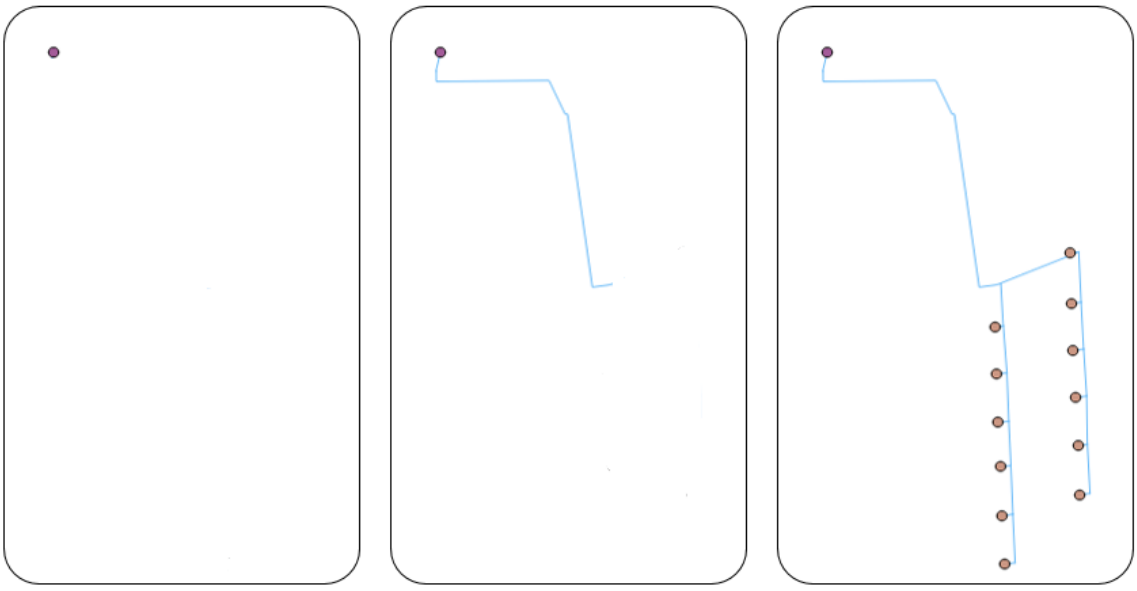


Figura 24: De izquierda a derecha, Nodo, Tramo y Grafo respectivamente.

Tras la definición de estos conceptos y aprovechando la capacidad de programación orientada a objetos que brinda Python, fue evidente la definición de las clases Nodo, Tramo y Grafo. Como vemos en la Figura 25, para crear estas clases es necesario establecer ciertos parámetros leídos directamente desde los Shapefiles.

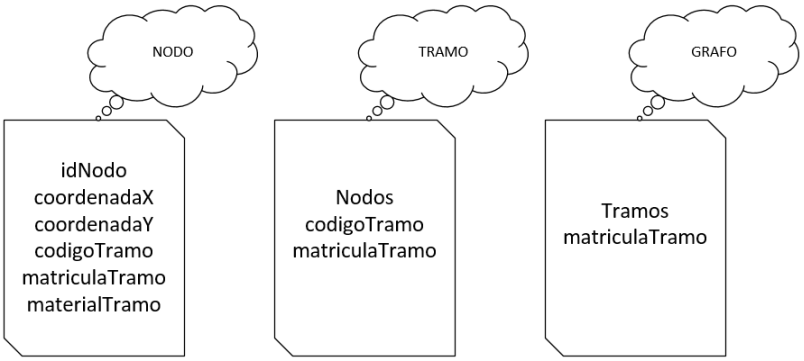


Figura 25: Clases Nodo, Tramo y Grafo

Aunque los nombres de los parámetros son bastante intuitivos, a continuación definimos lo que representan:

- idNodo: es un número entero positivo utilizado para identificar cada nodo.
- coordenadaX: coordenada X en metros del nodo.
- coordenanadaY: coordenada Y en metros del nodo.
- codigoTramo: identificador de un tramo concreto.
- matriculaTramo: identificador del grafo. Respetamos el nombre del .dbf.
- materialTramo: material utilizado en los cables de cierto tramo.
- Nodos: conjunto de nodos que forman el tramo.
- Tramos: Conjunto de tramos que forman el grafo.

Como ya hemos comentado, el objeto Nodo es el elemento básico de un Grafo y es el que mayor información tiene. Además de contener los parámetros que acabamos de ver, tiene ciertos parámetros que se establecen en función de si es una acometida, un centro de transformación o un punto de apoyo de baja tensión. La definición de estas características, como vemos en la Figura 26 viene dada por la lectura del Shapefile correspondiente.

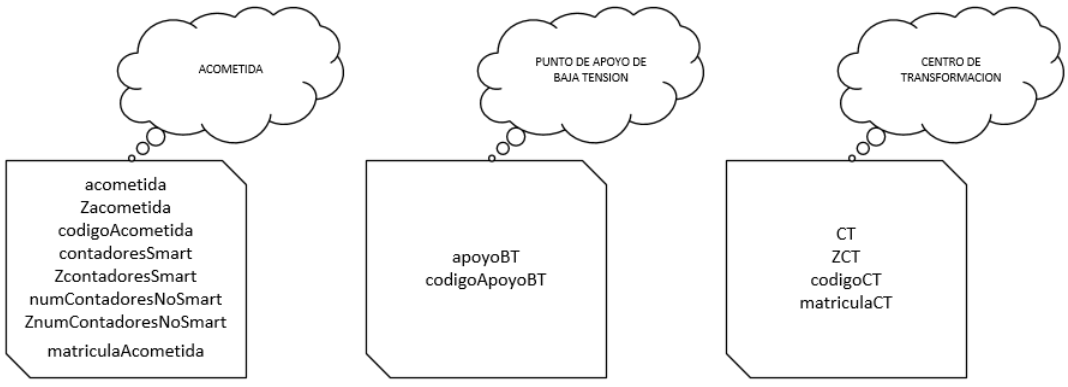


Figura 26: Parámetros Acometida, Centro de Transformación y Puntos de Apoyo.

Todos estos parámetros se "activarán" en función de la pertenencia del nodo al Shapefile en estudio. Es decir, si uno de los nodos pertenecientes al Shapefile de acometidas tiene las mismas coordenadas que un nodo ya leído en el Shapefile de tramos, éste se establecerá exclusivamente como nodo acometida. No obstante, esta explicación se hace más sencilla presentando en detalle cada uno de estos nuevos parámetros:

- acometida: si este valor está activo entonces el nodo es una acometida y nunca será un centro de transformación o un punto de apoyo.
- Zacometida: las acometidas contienen uno o más contadores con su respectiva impedancia. Este parámetro corresponde al paralelo de todas esas impedancias.
- codigoAcometida: es el identificador de la acometida según el Shapefile.
- contadoresSmart: vector con los identificadores de cada contador inteligente.
- ZcontadoresSmart: Impedancia de cada contador inteligente según [41].
- numContadoresNoSmart: Numero de contadores no inteligentes.
- ZnumContadoresNoSmart: impedancia que supone el paralelo de las impedancias de los contadores no inteligentes.
- matriculaAcometida: identificador alternativo de la acometida según el Shapefile. Éste, como veremos en el apartado siguiente, se utilizará para buscar la información relativa a contadores en el archivo Excel.
- apoyoBT: si este valor está activo entonces el nodo es un punto de apoyo de baja tensión y nunca será un centro de transformación o una acometida.
- codigoApoyoBT: éste es identificador del punto de apoyo de baja tensión según el Shapefile.
- CT: si este valor está activo entonces el nodo es un centro de transformación y nunca será una acometida o un punto de apoyo.
- ZCT: es la impedancia del centro de transformación modelada como una resistencia en serie con un condensador y ambos en paralelo con una bobina. Su valores son establecidos con $R_T = 8\ \Omega$, $L_T = 25\ \mu H$ y $C_T = 48\ nF$ para la resistencia, la bobina y el condensador respectivamente según [42].
- codigoCT: es el identificador del centro de transformación según el Shapefile.
- matriculaCT: identificador alternativo del centro de transformación según el Shapefile.

Tras haber establecido los parámetros pertinentes, como vimos en la Figura 18, toda esta información se almacenan en vectores de Nodo, Tramo y Grafo. Esta decisión se tomó para poder acceder en cualquier punto de la ejecución a toda la información leída de los Shapefiles. Esta situación ha sido de gran provecho sobre todo en el caso del vector de Nodos ya que constantemente se realizan búsquedas a partir del "idNodo" para conseguir ciertas caracteristicas de un nodo concreto.

Finalmente, a modo de conclusión de este apartado, se presenta la Figura 27 para describir de forma sencilla y concisa el procesamiento de los Shapefiles. No obstante, este proceso es realizado en una sola ocasión al inicio de la ejecución.

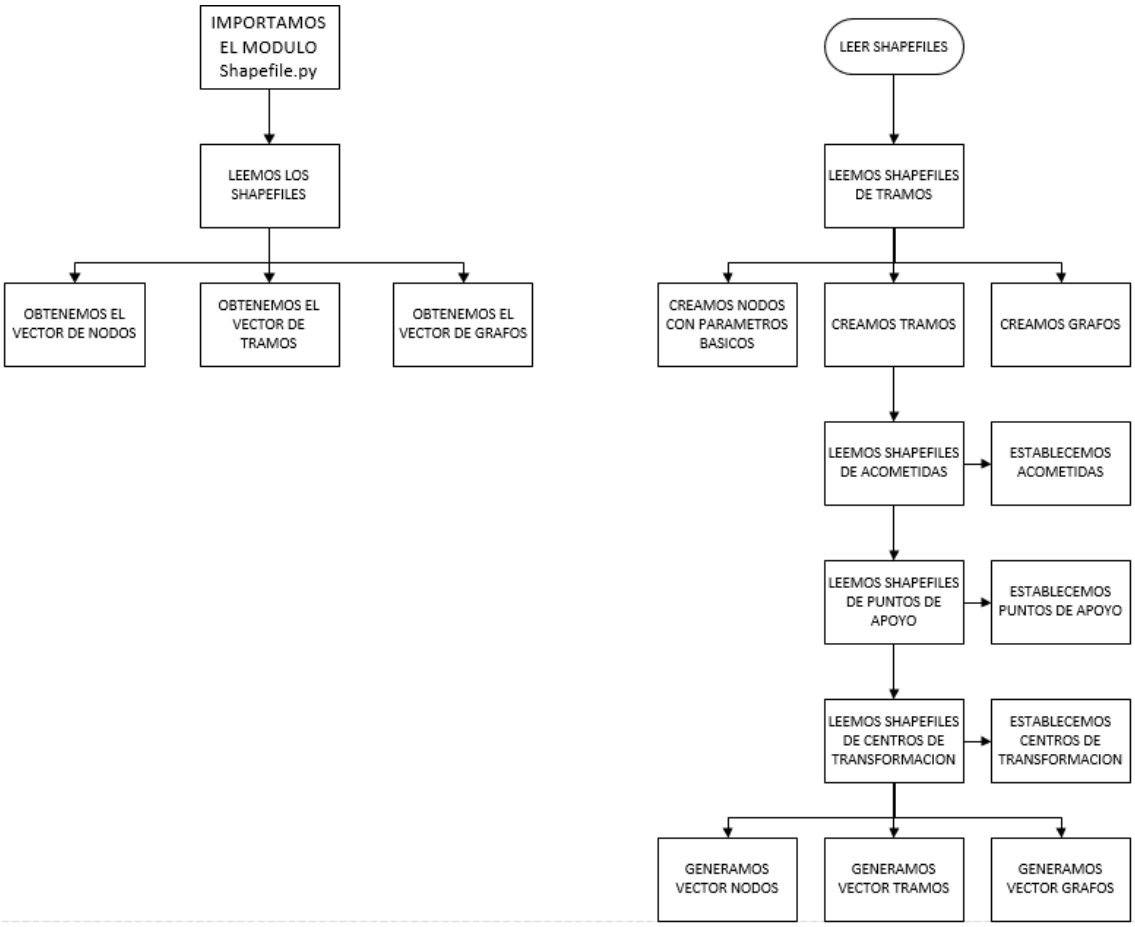


Figura 27: Procesamiento de Shapefiles.

4.1.3 Procesamiento del archivo Excel

Como vimos en el capítulo 2, la redes PRIME implican una conexión permanente entre el centro de transformación y los *Smart Meters* o contadores inteligentes. Es decir, se debe establecer una diferencia entre contadores inteligentes y contadores tradicionales, ya que estos últimos no tienen capacidad de comunicación.

Los Shapefiles contienen mucha información pero nada relevante a los contadores que hay en cada acometida. Sin embargo, Unión Fenosa facilitó también un archivo Excel que contiene información detallada sobre los contadores. Por lo tanto, es imprescindible leer el contenido de este archivo para enlazar la información necesaria con los Shapefiles ya interpretados.

El primer paso, al igual que en el apartado anterior, fue buscar la existencia de un módulo para Python que facilitara este trabajo. El resultado fue el módulo Openpyxl [43], el cual permite leer y escribir archivos Excel con extensión .xlsx.

Ya que la información que debemos leer de este archivo Excel sólo involucra a los contadores, el procedimiento seguido es que, cada vez que se establezca a un nodo el papel de acometida, se leerá el Excel para comprobar qué contadores existen en dicha acometida. Para ello, según la matrícula de la acometida indicada en el Shapefile

("matriculaAcometida") podremos saber cuántos contadores inteligentes y tradicionales hay en la acometida así como su correspondiente valor identificativo.

Tras conocer la información contenida en el Excel ya tenemos todos los parámetros relativos a una acometida vistos en la sección anterior, salvo las impedancias. Éstas son generadas de forma aleatoria de acuerdo a una distribución uniforme [0,5] +j[-5,5] Ω tal y como se propone en [41]. No obstante, para los contadores tradicionales, sólo es relevante la impedancia en paralelo de las impedancias individuales de cada uno, siendo éste el valor guardado en "ZnumContadoresNoSmart".

Aunque los pasos de este proceso son sencillos, existieron grandes complicaciones debido a los formatos de los archivos Excel facilitados. No todos contaban con el mismo formato de nombres ni de datos, hecho que ocasionó diversas modificaciones para conseguir que este módulo fuera lo más universal posible.

4.1.4 Relación entre Grafos y Trafos

Una vez obtenida toda la información necesaria contenida en los archivos facilitados, el siguiente objetivo es recorrer los grafos para calcular las atenuaciones entre acometidas en una primera instancia y posteriormente entre contadores inteligentes. Esto nos llevó a analizar los planos y vimos que éstos contenían bucles, es decir, para ir de una acometida a otra había varios caminos posibles.

Para resolver este problema pasamos a analizar aún más en profundidad dichos planos y vimos que estaban divididos en trafos que aparentemente no tenían bucles. Este fue el motivo por el cual establecimos la relación trafo-grafo y rechazamos la opción de tratar todo el plano como si de un único grafo se tratara.

Ya que el tratamiento de los datos es más sencillo cuando se visualizan, decidimos utilizar las posibilidades de escritura de Shapefiles que proporciona el modulo Shapefile.py. De esta forma, proporcionando la ruta de destino, a partir de un grafo somos capaces de generar un Shapefile que pueda ser representado. Aunque se pueden escribir tantos datos como se desee en la base de datos del Shapefile, simplemente optamos por escribir los requisitos mínimos para poder representar el grafo utilizando la aplicación QGIS [44]. La Figura 28 muestra la representación obtenida de una de las redes de distribución proporcionadas en formato Shapefile.

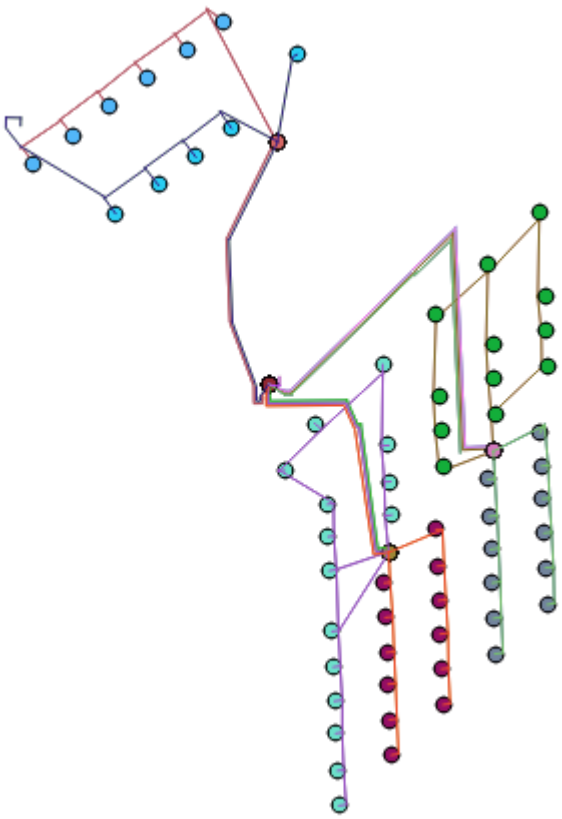


Figura 28: Representación gráfica de uno de los mapas en formato Shapefile proporcionados.

Esa representación permitió que nos cerciorámos de que realizando la división por trafos, aún existían bucles en los Shapefiles que Unión Fenosa había proporcionado hasta el momento, por lo que tuvimos que buscar la solución que veremos en detalle en el próximo apartado.

4.1.5 Generación de Árboles a partir de Grafos

Para solucionar los problemas que surgieron cuando queríamos recorrer los grafos, hubo que plantear una solución que nos permitiera eliminar los bucles. Para ello primero pensamos en el sentido físico que caracterizaba estos escenarios. En este sentido si bien es cierto que la topología física de una red de distribución puede presentar bucles, también es cierto que la red en explotación no los presenta (realmente se utilizan relés que los evitan inicialmente y que se cierran para seguir dando servicio frente a algún fallo puntual). Asimismo, Unión Fenosa también identificó que había mapas erróneos, es decir, que había mapas que presentaban ramas unidas que en la realidad no lo estaban.

Ante esta situación, basándonos en el protocolo *Spannning Tree* [45], tomamos la iniciativa de seleccionar los caminos más cortos desde el Centro de Transformación (o desde el punto de apoyo de baja tensión si lo hubiera) hasta cada una de las acometidas. De esta manera se conseguía un grafo sin bucles con el centro de transformación (o el punto de apoyo de baja tensión) como nodo raíz.

Esta denominación de "nodo raíz" hizo que nos replanteáramos la estructura de grafo llegando a la conclusión de que sería más útil y sencillo continuar con una estructura en forma de árbol. Es decir, convertimos los grafos en árboles obteniendo un resultado similar al de la Figura 29.

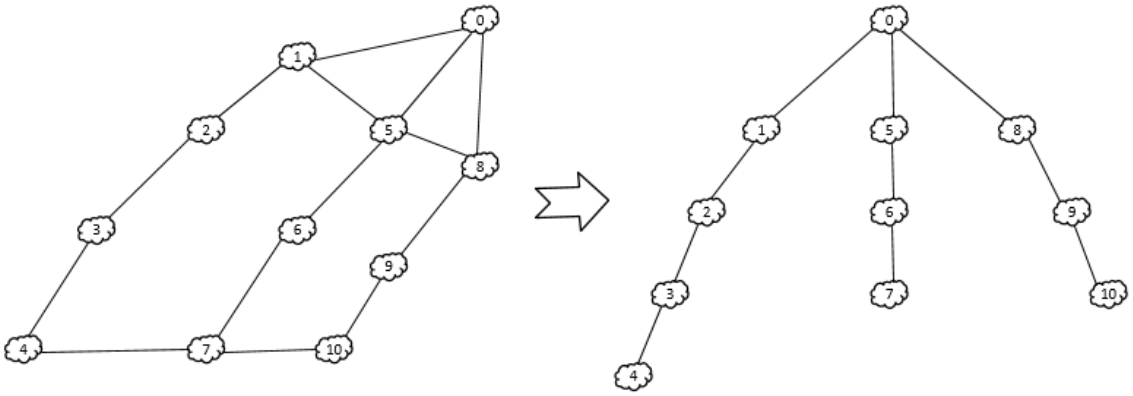


Figura 29: Paso de grafo a árbol.

Llevar a cabo esta transformación fue una ardua tarea que obligó a la generación de nuevos parámetros que permitieran establecer padres e hijos y no sólo nodo siguiente y anterior. Aunque un nodo dado sólo puede tener un padre, puede tener varios hijos por lo que se definierón los siguientes parametros:

- padre: es el nodo padre del nodo actual.
- distanciaPadre: distancia en metros desde el nodo actual al nodo padre.
- hijo: vector que contiene los nodos hijos del nodo actual.
- distanciaHijo: vector de distancias en metros desde el nodo actual a cada uno de los nodos hijos. Este vector y el anterior tienen el mismo orden.

Tras la definición de estos nuevos parámetros, el siguiente paso se resumía en establecer sus valores en función de la posición que cada nodo ocupaba en el grafo teniendo en cuenta el camino más corto que mencionábamos antes. Finalmente, no sólo establecemos la estructura árbol sino que los analizamos y conseguimos simplificarlos, tal y como ilustra la Figura 30.

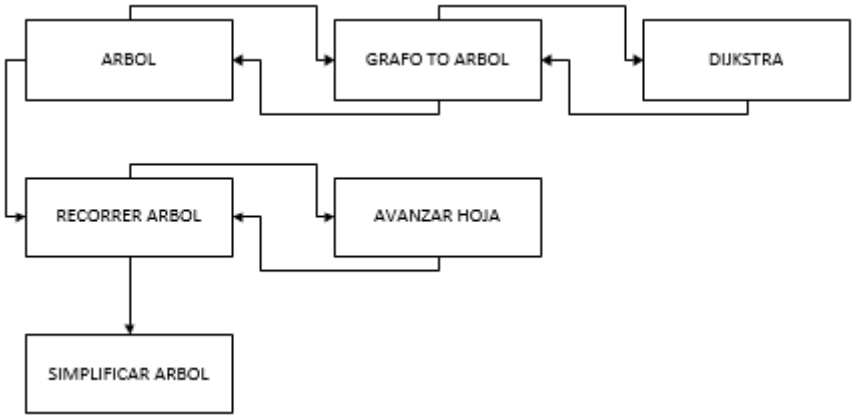


Figura 30: Proceso de creación y simplificación de árboles.

Ya que este desarrollo puede ser confuso, a continuación se explica más detalladamente lo que hace cada uno de los módulos involucrados:

dijkstra:

Para conseguir el camino más corto entre el nodo raíz y cada acometida es necesario aplicar el algoritmo de Dijkstra implementado en esta función. El resultado que obtenemos tras su ejecución es un vector con el identificador de cada nodo que debemos recorrer de forma ordenada o infinito en el caso de no existir conexión con el nodo raíz.

El resultado que obtenemos con esta función es perfecto para el caso que nos ocupa pero, dadas las exigencias del algoritmo, encontramos una complicación: los parámetros de entrada. Ya que el algoritmo debe analizar cada uno de los caminos posibles, es necesario proporcionar en cada ejecución un vector de aristas que, como vemos en la Figura 31, contenga todo el escenario en estudio.

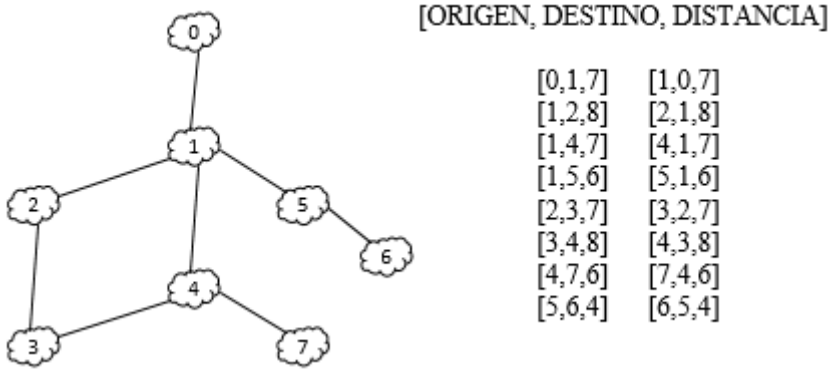


Figura 31: Grafo como vector de aristas.

Por lo tanto, esta función nos proporciona el camino más corto entre dos puntos dado un vector de aristas, el punto origen y el punto destino.

grafoToArbol:

Dadas las exigencias de los parámetros de entrada de la función anterior, es imprescindible representar el Grafo del que partimos como un conjunto de aristas por lo que es el primer proceso que llevamos a cabo en esta nueva función.

Una vez creado el vector de aristas y utilizando la función Dijkstra, podremos calcular el camino más corto entre cada acometida y el Nodo raíz. Es decir, obtenemos las diferentes rutas que definen el camino desde el nodo a raíz a cada acometida. Como vemos en la Figura 32, el árbol queda perfectamente definido interpretando en conjunto todas estas rutas.

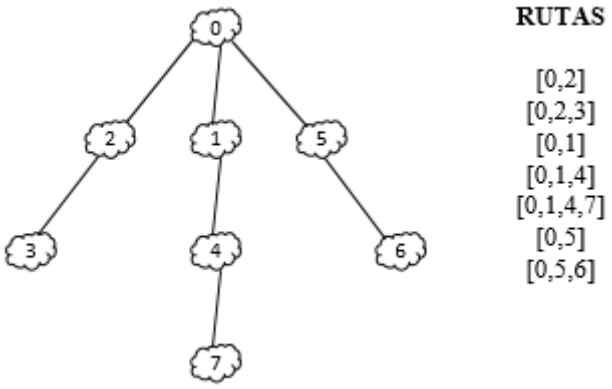


Figura 32: Grafo como vector de rutas.

Es importante aclarar que el Nodo raiz siempre será un punto de apoyo de baja tensión salvo que éste no exista y pase a serlo el centro de transformación. Éste último caso es el mas común pero dado que la matriz de atenuaciones involucra el centro de transformación, en esta misma función creamos una ruta que une el centro de transformación con el punto de apoyo de baja tensión cuando éste último esté presente. Es decir, todo el árbol queda completamente conectado salvo que el grafo original contenga desconexiones, caso en el que la ejecución se detiene y muestra un mensaje de error.

Finalmente, tras todos estos matices, sólo nos falta eliminar aquellas rutas redundantes como la 1-2-3 cuando ya tenemos la ruta 1-2-3-4-5 y devolver a la función soliticitante el vector de rutas resultante.

árbol:

El primer paso para transformar nuestro grafo en un árbol es obtener el vector de rutas que lo defina. Usando la función anterior tenemos este primer requisito superado por lo que únicamente falta establecer los parametros que definen el árbol para cada nodo como vemos en la Figura 33.

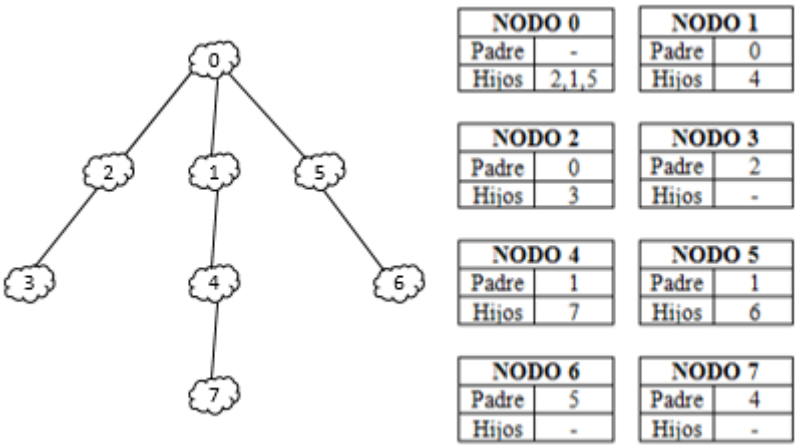


Figura 33: Establecer padres e hijos

Recorriendo cada una de estas rutas, ya que están ordenadas, simplemente debemos ir estableciendo padres e hijos. Además, en el caso de haber utilizado un punto de apoyo de baja tensión como Nodo raiz, separamos la ruta que lo une con el centro de

transformación para finalmente devolver el árbol y el enlace entre el centro de transformación y el punto de apoyo.

avanzarHoja:

Ya que las funciones recursivas son ciertamente problematicas en Python, la solución más apropiada para recorrer el árbol desde los nodos hoja (no tienen hijos) hasta el nodo raiz (no tienen padre) es implentar una función que "escale" por el árbol en cada llamada. Es decir, una función que cada vez que se ejecuta dada una posición actual, avance hasta la siguiente posición devolviendo esta situación como la actual.

Para realizar este algoritmo, utilizamos la función "avanzarHoja" cuyos parametros de entrada son los nodos hoja desde los que partimos y el identificador del nodo raiz. Su funcionamiento consiste en avanzar desde los nodos hoja facilitados hasta los nodos padre de al menos 2 hijos que se encuentrán mas cerca en el recorrido del árbol. Finalmente serán devueltos estos nodos padre de tal forma que puedan ser útilizados en la proxima ejecución según lo solicite la función "recorrerÁrbol".

recorrerArbol:

Como acabamos de comentar, somos capaces de recorrer el árbol llamando en numerosas ocasiones a la función "avanzarHoja" desde otra funcion que gestione el momento de detener estas llamadas y guarde los resultados. Éste es el motivo por el cual fue implementada la función "recorrerArbol", es decir, es la encargada de enviar peticiones a "avanzarHoja" para "escalar" por el árbol, almacenar la información generada y detener el ciclo de peticiones.

En la Figura 34 se ilustra cómo se recorre el árbol haciendo uso de este algoritmo basado en el establecimiento de hojas. Es decir, partimos de los nodos hoja reales y en cada iteración los nuevos nodos hoja seran aquellos que tengan 2 o más hijos de tal forma que vamos "eliminando" el recorrido ya visitado.

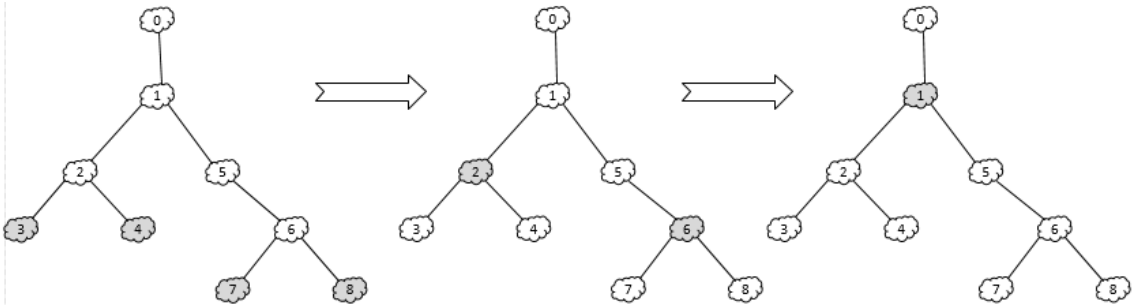


Figura 34: Recorrido por el árbol basado en el avance entre hojas.

Finalmente devolvemos los identificadores de los que han sido nodos hoja en cada una de las iteraciónes.

simplificarArbol:

Finalmente, con los datos generados, somos capaces de simplificar el árbol sin perder la información que nos insteresa. A grandes rasgos, nuestro objetivo es eliminar aquellos nodos cuya función era relevante en los Shapefiles para realizar un cambio de dirección en los tramos. Llegados a este punto sólo nos son de interes características

como el material y la distancia del enlace, si sufre un cambio de dirección no nos aporta información para este proyecto.

Lo primero a lo que nos enfrentamos es al estudio de cada una de las ramas que forman el Árbol de forma individual. Para ello, ya que contamos con los distintos identificadores de cada nodo origen en el escalado del árbol ("idNodosHojaRecorridos"), establecemos origenes y destinos en cada iteración para así poder estudiar cada rama de forma independiente, como se muestra en la Figura 35.

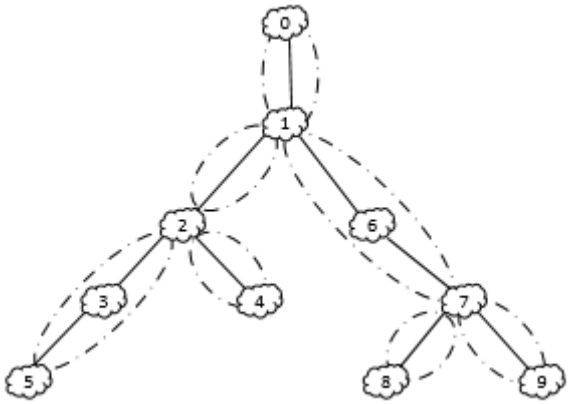


Figura 35: Estudio individual de las ramas.

Una vez obtenida la capacidad de estudiar cada rama individualmente realizaremos el mismo estudio en cada una de ellas para intentar simplificar el numero de nodos y enlaces que la forman. No obstante, es importante mencionar las caracteristicas con las que nos podremos encontrar:

- La primera de ellas es el caso del cambio de material en el enlace. Los Shapefiles contemplan esta situación como la unión de dos tramos diferentes con materiales diferentes. Es decir, existe un nodo con las mismas coordenadas en ambos tramos. La solución adoptada para el caso de nuestros árboles, ha sido respetar la existencia de ambos nodos pero fijar un enlace virtual con un material que denominamos "X" y una distancia de 0 metros. Ante esta situación, recorriendo el árbol somos capaces de reconocer cuando hay un emplame y, aunque ahora no se utiliza este potencial, se podría contemplar la opción de sumar unas perdidas por cada cambio de material.
- Como ya hemos comentado en números ocasiones, nos podemos encontrar con nodos acometida, centro de transformación y puntos de apoyo de baja tensión.
- Otra situación diferente se da cuando nos encontramos ante un nodo con varios hijos. Es decir, un nodo del que "cuelga" al menos una rama.
- Aunque normalmente son las acometidas, tambien podemos encontrar nodos hoja "puros". Es decir, nodos que no tienen hijos.
- Finalmente llegamos a la situación que queremos eliminar, nodos que se encuentran en medio de la rama donde su enlace anterior y posterior tienen el mismo material. Sólo actuaban como cambio de dirección en los Shapefiles.

Por lo tanto, en el estudio de cada rama, analizaremos la última situación enumerada para obtener un árbol simplificado como el de la Figura 36. Es decir, suprimimos aquellos nodos con el mismo material en sus dos enlaces y creamos un único enlace entre los nodos vecinos del nodo eliminado teniendo en cuenta la suma de las distancias de los enlaces suprimidos y su material.

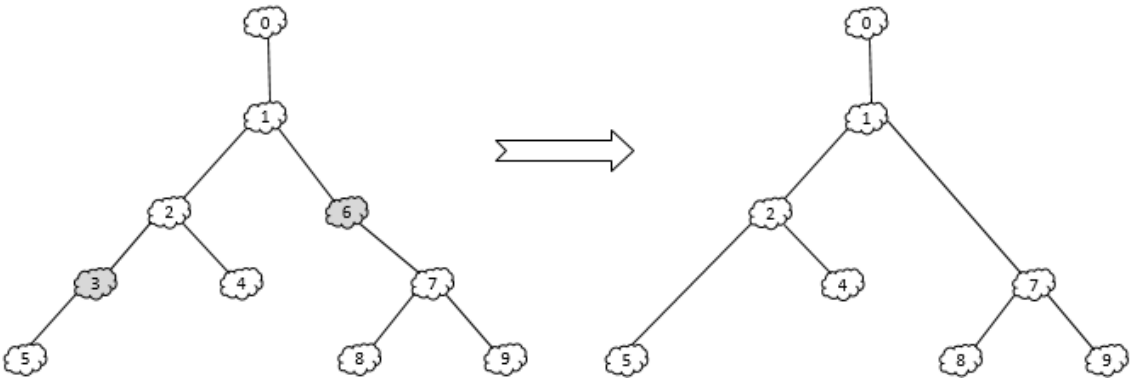


Figura 36: Simplificación de árboles.

Para concluir este proceso, aprovechamos el hecho de recorrer cada rama para crear un vector de aristas actualizado. Éste nos será de gran utilidad en los apartados siguientes para poder volver a utilizar el algoritmo de Dijkstra.

4.1.6 Calculo de impedancias

Una vez generada la topología en forma de árbol, se nos plantea qué metodología seguir para realizar el cálculo de la matriz de atenuaciones objetivo. Para ello, llevaremos a cabo un estudio basado en la función de transferencia a lo largo de cada uno de los enlaces que forman el árbol como se explica detalladamente en el Anexo I. No obstante, para poder implementar este algoritmo, debemos recorrer el árbol de abajo a arriba y luego de arriba a abajo para calcular las impedancias de entrada que verá cada uno de los nodos en cada una de sus ramas.

Antes de comenzar el proceso del cálculo de impedancias es imprescindible recordar que hemos estudiado cada una de las ramas que "cuelgan" del centro de transformación como si fueran árboles independientes. Este hecho plantea el escenario presente en la Figura 37, donde volveremos a estudiar en conjunto todos estos subarboles.

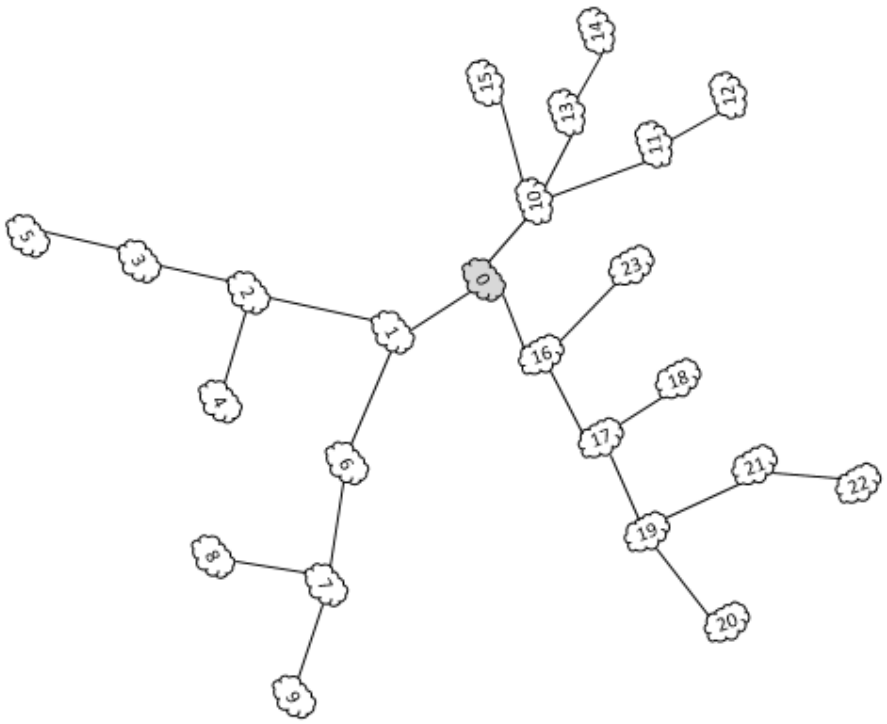


Figura 37: Árbol de árboles.

No obstante, como puede observarse en la Figura 38, para la primera parte del calculo de impedancias continuaremos tratando cada subárbol por separado. Es cuándo pasemos a realizar los calculos de impedancias de arriba abajo cuando realicemos la convergencia.

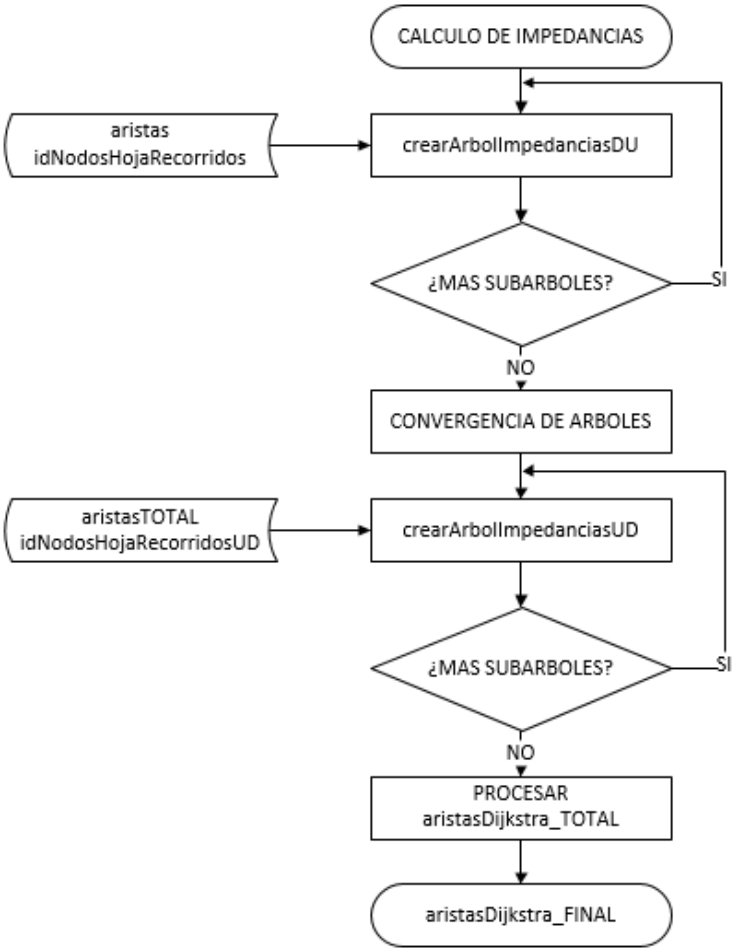


Figura 38: Cálculo de impedancias del árbol.

Por lo tanto, nuestra primera tarea es realizar el calculo de impedancias de arriba abajo utilizando la función "crearArbolImpedanciasDU" pero antes nos vimos obligados a añadir nuevos parametros a la clase Nodo que nos permitiera almacenar los resultados que no interesan de cada cálculo:

- nodoArbolImpedancias: para evitar confusiones con nodos suprimidos en el proceso de simplificación del árbol, se utiliza esta variable para establecer los nodos correspondientes como miembros del árbol cuyas impedancias calcularemos.
- padreArbolImpedancias: es el nodo padre del nodo en estudio.
- distanciaPadreArbolImpedancias: distancia del nodo en estudio a su padre.
- hijoArbolImpedancias: vector que contiene cada uno de los hijos del nodo en estudio. Siempre en referencia al árbol para el calculo de impedancias.
- distanciaHijoArbolImpedancias: vector con las distancias a cada nodo de los detallados en la variable anterior.
- impedanciaEnlacePadre: impedancia del enlace que nos une con el padre
- impedanciaEnlaceHijo: vector de las impedancias de los enlaces que nos une con cada uno de los hijos.
- impedanciaRamaHijo: vector con la impedancia a la entrada de la rama de cada uno de los hijos.

- impedanciaRamaPadre: impedancia a la entrada de la rama que se dirige hacia el nodo padre.
- materialEnlacePadre: modelo del cable utilizado en el enlace con el padre.
- materialEnlaceHijo: vector con el modelo de los cables que unen al nodo en estudio con cada uno de sus hijos.
- idHijo: identificador de cada uno de sus hijo utilizado para tener cada elemento referido a los hijos de forma ordenada.

Una vez presentados los nuevos parametros procedemos a explicar más detalladamente cada uno de los módulos involucrados en este nuevo proceso:

crearArbolImpedanciasDU:

El propósito de esta función es recorrer el árbol desde sus nodos hoja hasta su nodo raíz para calcular las impedancias de entrada de cada una de las ramas hija de cada uno de los nodos. Es decir, recorremos el árbol de abajo a arriba para calcular el valor de "impedanciaRamaHijo" de cada uno de los nodos.

Para ello, aunque anteriormente realizamos una tarea similar, contruimos lo que denominamos "Árbol de impedancias", es decir, establecemos los parametros de padres e hijos recientemente detallados para formar un árbol con solo aquellos elementos de interés para el cálculo de impedancias. Seguidamente, procemos de forma similar a la utilizada en "simplificarArbol" para obtener origenes y destinos. Es decir, utilizamos el vector "idNodosHojaRecorridos" para poder estudiar de abajo a arriba cada rama de forma independiente.

Una vez capacitados para estudiar cada una de las ramas de forma independiente, realizaremos el calculo de impedancias de cada una siguiendo el método detallado en la Figura 39.

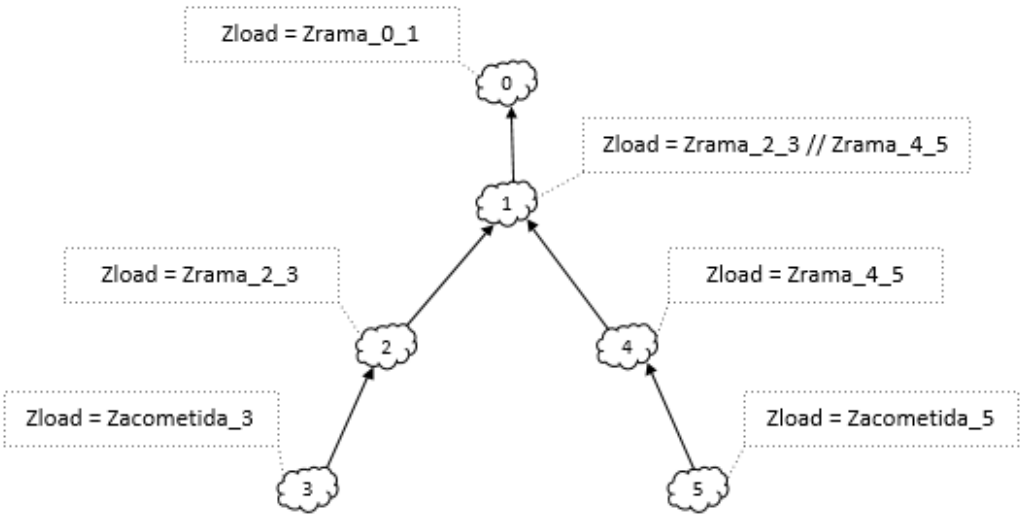


Figura 39: Cálculo de impedancias de abajo a arriba.

Durante todo este proceso vamos calculando las impedancias de forma acumulativa hasta llegar a un nodo con al menos dos hijos. Es decir, la impedancia de entrada de una rama inferior es la impedancia de carga de la impedancia de rama

superior. Seguidamente, cuando llegamos al nodo con varios hijos, la impedancia de carga de su rama superior será la impedancia en paralelo de todas las impedancias de entrada de cada uno de los hijos.

Es importante comentar que el cálculo de impedancias de entrada de cada enlace se realiza utilizando las expresiones de la impedancia de entrada en una linea de transmision descritas en el Anexo I, donde sus factores más determinantes son la distancia, la frecuencia de trabajo y el material del enlace. Este último definirá las propiedades físicas de los cables que permiten calcular los parámetros por unidad de longitud R, L, G y C.

Por lo tanto, igual que ocurría en la sección anterior, durante esta fase del cálculo de impedancias tendremos que tener en cuenta cada una de las situaciones ante las que nos podemos encontrar:

- Acometida sin hijos: éste es el caso más general y suele ser el principio habitual de este algoritmo. Establecemos como impedancia de carga de la rama inmediatamente superior la impedancia de la acometida.
- Acometida en medio de una rama: es un caso excepcional en el que la acometida tiene un enlace padre y un enlace hijo. Ya que las acometidas son el protagonista principal para el cálculo de atenuaciones, calculamos la impedancia de su única rama hija que a su vez será la impedancia de carga de la rama padre.
- Cambios de medio: ante esta situación debemos proceder de forma análoga al caso anterior, calcular la impedancia de entrada de su rama hija que, en este caso, será la impedancia de carga de la rama superior tras haber realizado el cambio de medio.
- Nodos hoja que no son acometidas: aunque es un caso muy poco habitual puede darse la situación de existir hojas sin acometidas. Ante esta situación interpretamos que estamos ante un enlace terminado en circuito abierto, es decir, una impedancia de un valor idealmente infinito como impedancia de carga.
- Nodos con varios hijos: en este caso simplemente calculamos la que será la impedancia de carga del enlace superior como el paralelo de las impedancias de entrada de cada una de las ramas hijas.
- Enlace de punto de apoyo de baja tensión con centro de transformación: como dijimos en secciones anteriores, puede darse el caso remoto de que nuestro nodo raíz original sea un punto de apoyo de baja tensión por lo que tendremos que realizar el cálculo de impedancias en la rama de enlace con el centro de transformación de manera similar a los casos anteriores.

Tras el cálculo de las impedancias de entrada de cada uno de los enlaces hijos en todos los subárboles que cuelgan del centro de transformación, habremos generado los vectores “idNodosHojaRecorridos_UD_TOTAL” y “aristasTOTAL” que serán de utilidad para realizar el cálculo de las impedancias de las ramas padre.

Convergencia:

Para completar el cálculo de impedancias, tendremos que recorrer el árbol de arriba a abajo calculando el valor de la impedancia de entrada de la rama padre de cada uno de los nodos. Para ello procederemos de manera muy similar a la llevada a cabo en el caso anterior pero en este caso tenemos que tener en cuenta un factor importante: la impedancia que supone el resto de subárboles al subárbol en estudio. Para ello hemos ideado, como se ilustra en la Figura 37, un metodo que nos permite unir cada uno de los subárboles mediante el centro de transformación (igual que ocurre en los Shapefiles).

Ya que el centro de transformación de cada subarbol en realidad es el mismo (solo se diferencia por el idNodo por motivos prácticos), simplemente creamos un "Nodo Central virtual" que sirva de enlace entre ellos sólo a efectos de que todo quede conectado, como se ilustra en la Figura 40.

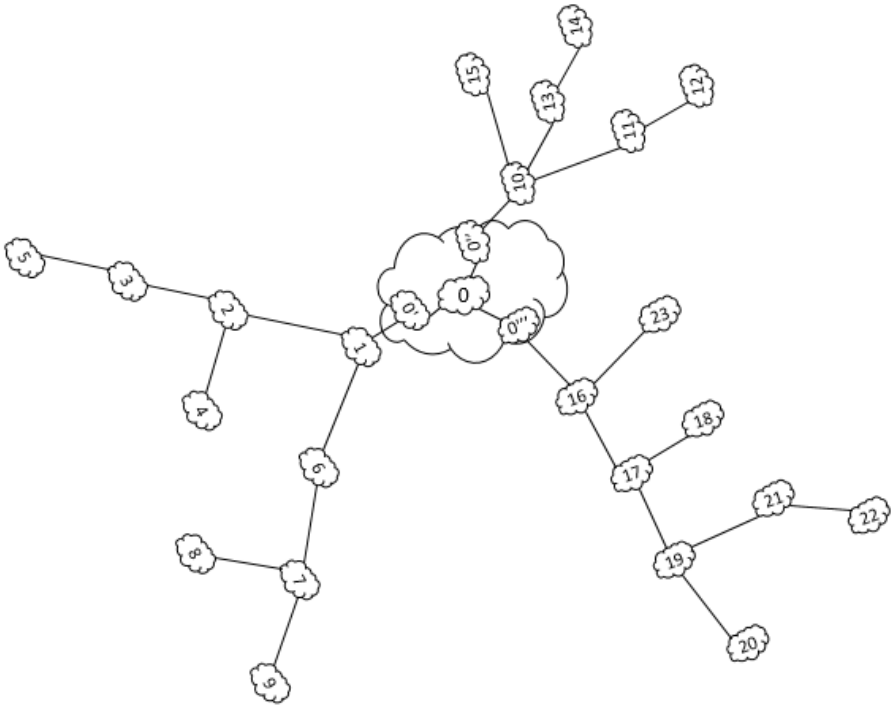


Figura 40: Nodo central virtual.

crearArbolImpedanciasUD

En esta nueva función, como hemos comentado previamente, el objetivo principal es calcular el valor de la variable "impedanciaRamaPadre" de cada uno de los nodos que forman el árbol en estudio. Para ello debemos recorrer el árbol de arriba a abajo y así realizar el cálculo de impedancias de forma acumulativa.

En esta ocasión, a diferencia de cuando íbamos de abajo a arriba, es importante conocer la rama por la que vamos avanzando ya que en ciertos nodos habrá que tener en cuenta la impedancia de rama del resto de ramas hijas. Este matiz se ilustra en la Figura 41 donde en los nodos 1 y 2 debemos tener en cuenta la impedancia de las ramas hijas que no forman parte del recorrido.

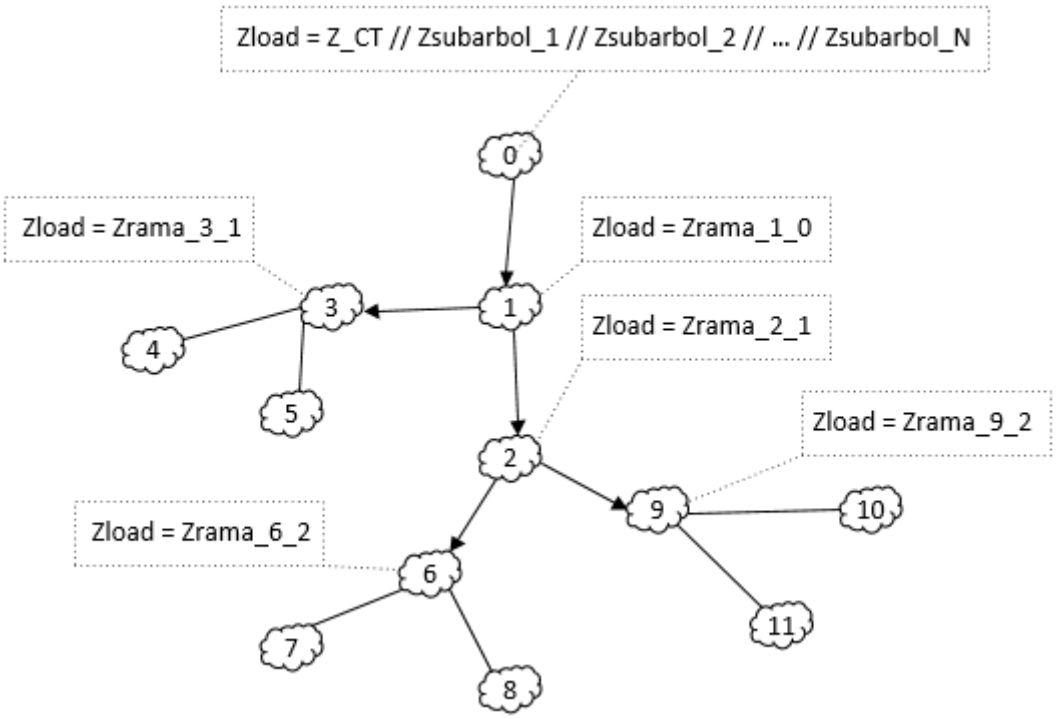


Figura 41: Calculo de impedancias de arriba a abajo

La solución tomada, ya que contamos con los vectores "aristas" e "idNodosHojaRecorridos", ha sido establecer siempre como origen el centro de transformación y calcular, utilizando el algoritmo de Dijkstra, la ruta hacia cada nodo hoja. De esta forma tratamos de forma independiente cada uno de los posibles caminos conociendo siempre nuestra localización actual sobre ellos. No obstante, como ocurría en el caso inverso, debemos tener en cuenta las opciones que nos podemos encontrar a lo largo del recorrido:

- El nodo actual es el centro de transformación: es el principio de cada una de las rutas por lo que establecemos como impedancia de carga la impedancia del centro de transformación en paralelo con las impedancias "hacia abajo" del resto de árboles.
- Nodo hoja: es el final del recorrido por lo que simplemente se establece el valor de la impedancia de entrada de la rama padre del nodo en estudio.
- Nodo con varios hijos: éste es el caso más delicado de este proceso ya que debemos tener en cuenta hacia dónde vamos. De esta forma comenzamos calculando la impedancia de entrada de la rama padre del nodo actual para, seguidamente, calcular la nueva impedancia de carga como el paralelo de la impedancia de entrada de la rama padre que acabamos de calcular con la impedancia de las ramas hijas excepto la siguiente rama de nuestro recorrido.
- Cambios de medio: en este caso, calculamos la impedancia de entrada de la rama padre para asignarle este mismo valor a la impedancia de carga de este mismo nodo pero sobre el nuevo medio.

Tras el cálculo y el establecimiento de las impedancias de la rama padre de cada uno de los nodos, generamos el vector "aristasDijkstraTOTAL" compuesto por las aristas de todo el conjunto de subárboles.

Procesamiento de aristas:

Para finalizar este proceso preparatorio para el cálculo de la matriz de atenuaciones, sólo nos falta añadir al vector "aristasDijkstraTOTAL" las aristas que unen el "nodo central virtual" con cada subárbol como vimos en la Figura 40. Con esta incorporación tendremos finalmente el vector "aristasDijkstraFINAL" que será de gran utilidad en el próximo apartado.

4.1.7 Calculo de la matriz de atenuaciones

Llegados a este último punto de la generación de la matriz de atenuaciones debemos tener en cuenta que todo el proceso que estamos detallando no sólo será ejecutado una vez antes de generar la matriz de atenuaciones buscada. Ya que hay ciertos parámetros aleatorios y la frecuencia de trabajo se encuentra situada dentro de la banda de PRIME, realizaremos varias iteraciones para impedancias diferentes y calculando la atenuación media en toda la banda para cada iteración. De esta manera, la matriz de atenuaciones final representara más estrictamente el escenario que los Shapefiles originales planteaban.

Una vez preparado el escenario completo entramos de lleno en el cálculo de las diferentes atenuaciones que definirán la matriz buscada. Para ello continuaremos con el desarrollo del algoritmo presentado en el Anexo I, según el cual nuestro siguiente paso consistirá en el cálculo de las diferentes matrices de transmisión involucradas en la ruta en estudio y su posterior procesamiento para calcular la atenuación.

Como vemos en la Figura 42, el proceso que seguiremos se basa en la continua petición por parte de "crearMatrizAtenuaciones" a "calcularMatricesTransmision" para obtener la matriz de transmisión resultante entre acometidas. Tras la obtencion de esta matriz, se procesa para calcular la atenuación extremo a extremo de la ruta en estudio y así finalmente ser incorporada a la matriz de atenuaciones.

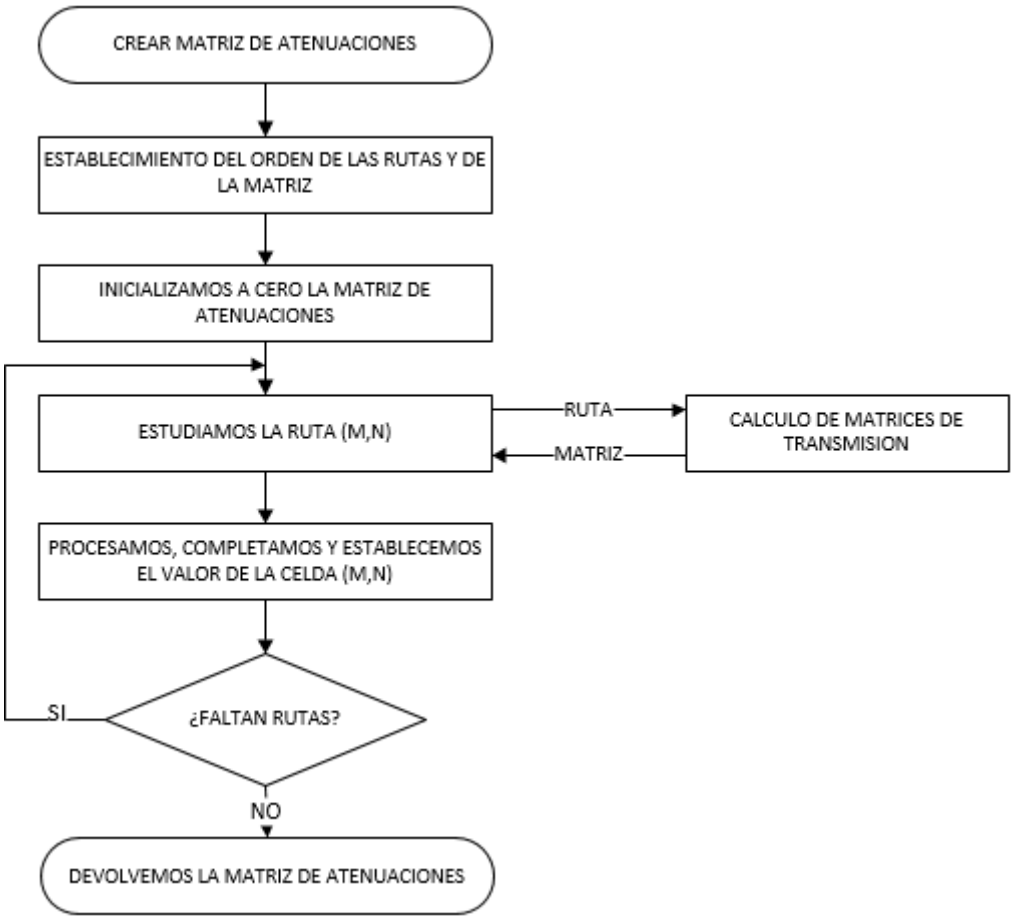


Figura 42: Proceso de creación de la matriz de atenuaciones.

Como podemos intuir, este es un proceso largo, complejo y delicado por lo que en su desarrollo se optó por la claridad de su código mas que por eficiencia. Es decir, la implementación de los módulos que detallaremos a continuación podría haber ocupado menos lineas de código pero dada su criticidad, se optó por la simplicidad.

calcularMatricesTransmision:

Las matrices de transmisión, como se indica en el Anexo I, se calculan integramente en base a parámetros relativos a los modelos de cables y a las impedancias calculadas en el apartado anterior. Es decir, tenemos todo lo necesario para proceder a su calculo.

En esta función particular nos encargaremos, dada una ruta del subarbol, en calcular la multiplicación de todas las matrices de transmisión involucradas. Como vemos en la Figura 43, en este proceso debemos tener en cuenta dos tipos de matrices: la matriz de transmisión T_L y T_{LL}, donde la primera de ellas hace alusión al camino recorrido y la segunda a las ramas que quedan sin recorrer.

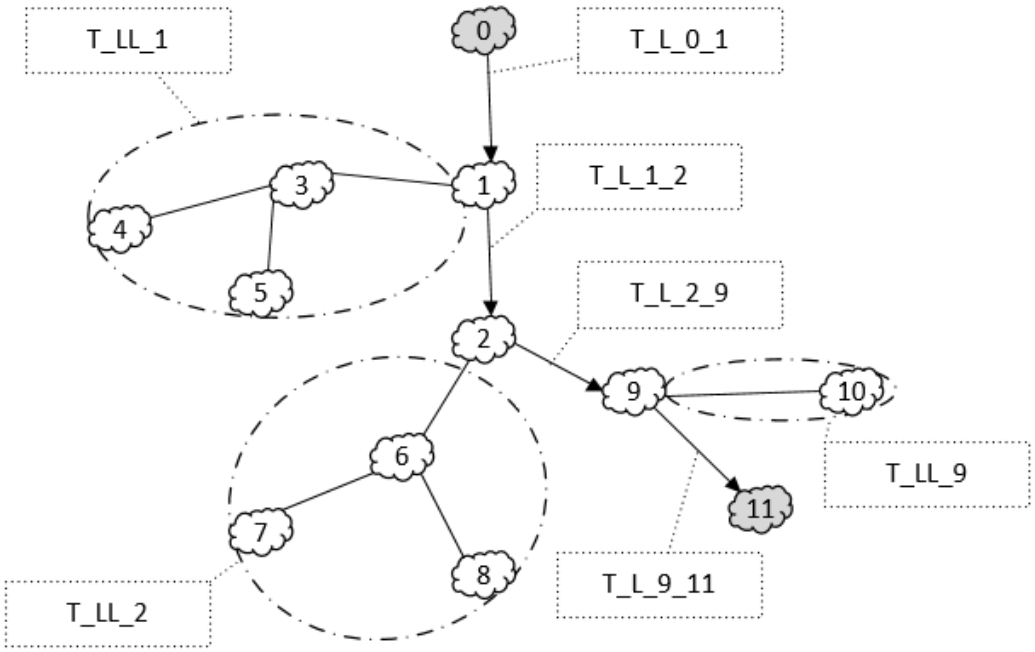


Figura 43: Matrices de transmisión

Una vez más, la mayor complicación radica en conocer en todo momento la localización que ocupamos en el árbol en el momento actual ya que ahora es relevante de dónde venimos y hacia dónde vamos. Ádemas, un factor importante es que la multiplicación de las matrices de atenuación debe hacerse en el orden correcto ya que, dadas las reglas básicas del álgebra, el orden de las matrices importa para multiplicarlas.

Sin embargo, áun siendo un proceso tan delicado, procedemos de una manera muy similar a casos anteriores donde la situación del nodo en estudio y la de sus vecinos definirán las diferentes situaciones con las que nos podemos encontrar:

- Nodo con varios hijos: es la situación más común y el procedimiento a seguir dependerá de si venimos y/o vamos a una rama padre y de si venimos y/o vamos a una rama hijo. Para cualquiera de los casos posibles se calculará la T_L correspondiente al enlace anterior de la ruta y la T_{LL} correspondiente a la impedancia en paralelo de todas las ramas no pertenecientes a la ruta.
- Cambios de medio: la primera opción que se plantea es que sea un simple empalme en medio de una rama por lo que simplemente se procede al calculo de la T_L correspondiente al enlace anterior. La otra opción es que estemos ante un nodo con varios hijos tambien por lo que se procederá de forma similar al caso anterior.
- Último nodo de la ruta: cuando estamos en el último tramo de la ruta simplemente se calcula la T_L correspondiente a este último enlace.

Todas estas matrices que han sido introducidades en un vector de forma ordenada, finalmente son multiplicadas en orden para devolver la matriz “matrizTransmisionTOTAL” a la función solicitante, es decir, a “crearMatrizAtenuaciones”.

crearMatrizAtenuaciones:

Tras un largo proceso, finalmente nos disponemos a detallar la última función involucrada en la generación de la matriz de atenuaciones. Como ya hemos comentado, ocupa gran cantidad de código Python pero el proceso de implementación es fácilmente resumible. El primer paso, como vimos en la Figura 42, crea un orden para definir la matriz de atenuaciones y las rutas, es decir, un vector que identifique las filas/columnasde la matriz y que permita generar las diferentes rutas como se ilustra en la Figura 44.

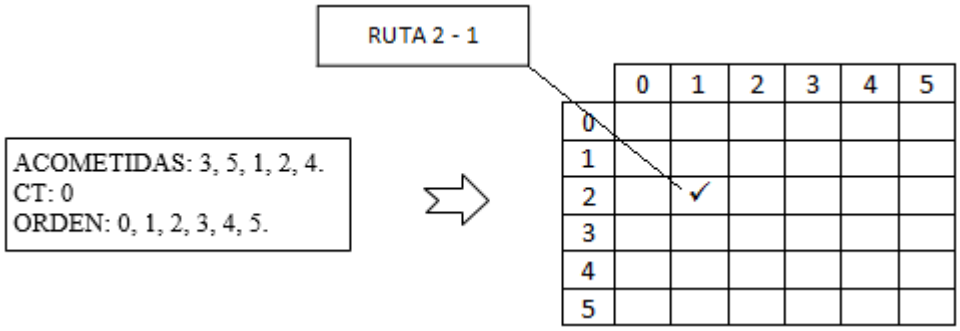


Figura 44: Orden de las rutas y de la matriz de atenuaciones.

Una vez definido el orden en función del centro de transformación y de los contadores inteligentes involucrados, inicializamos la matriz de atenuaciones a cero y creamos un vector de rutas donde se guarda cada una de las rutas que estudiaremos para calcular su atenuación.

En función del origen y el destino de la ruta, aplicaremos el calculo de atenuaciones en base a las siguientes situaciones:

- Atenuación entre contadores inteligentes de una misma acometida:
  - Atenuación de un contador inteligente con él mismo.
  - Atenuación de un contador con otro de la misma acometida.
- Atenuación del centro de transformación con él mismo.
- La ruta pasa por el centro de transformación.
- La ruta no pasa por el centro de transformación:
  - De un contador de una acometida a otro de otra acometida.
  - De un contador al centro de transformación.
  - Del centro de transformación a un contador.

Como se puede deducir fácilmente, el caso mas complejo y comprometido es el de aquellas rutas que pasan por el centro de transformación, es decir, partimos de una acometida de cierto subarbol para finalizar en la acometida de otro subarbol. Ya que es la situación con mayor nivel de dificultad, a continuación la explicaremos en detalle de tal forma que sirva de comprensión para el resto de los casos.

Para la correcta implementación del cálculo de atenuaciones en una ruta que pasa por el centro de transformación, lo primero es detallar el escenario que se nos presenta para analizarlo y poder aplicar las herramientas con las que contamos. Como vemos en la Figura 45, podemos dividir la ruta entre dos: una ruta desde la acometida

origen hasta el centro de transformacion y otra desde centro de transformacion a la otra acometida.

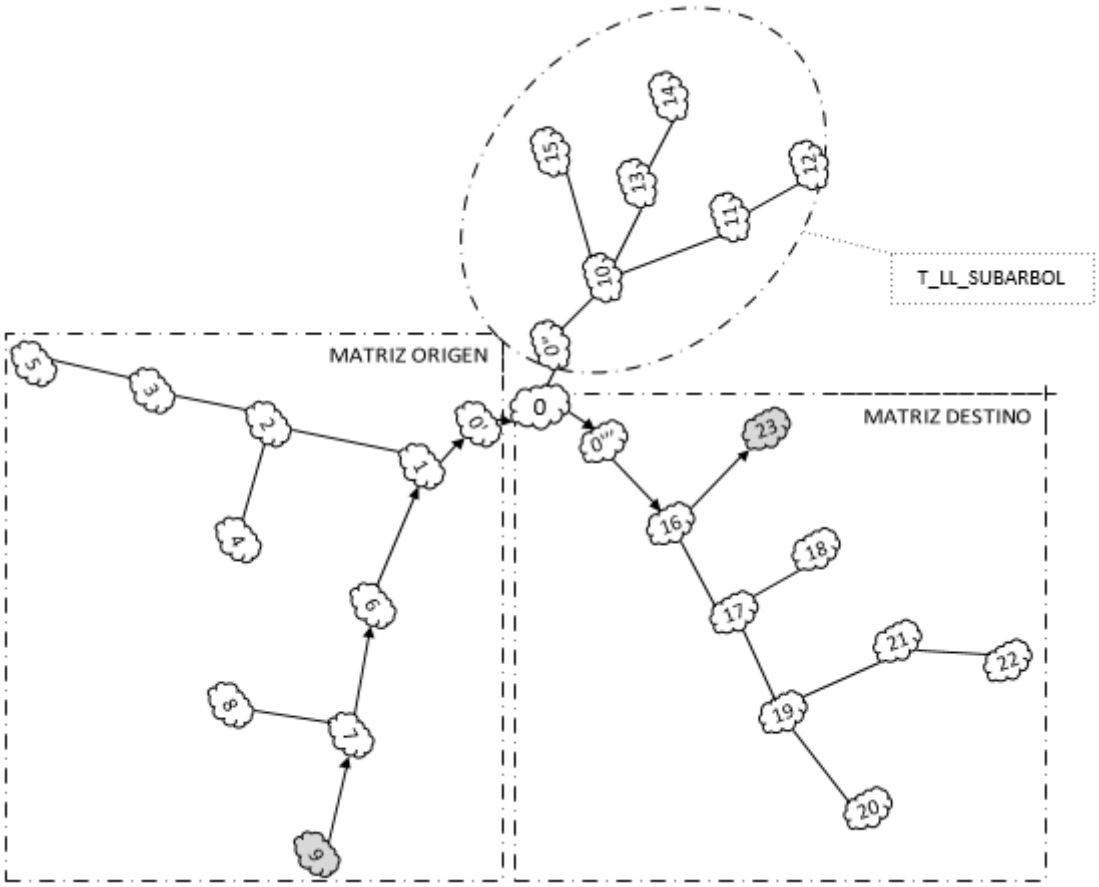


Figura 45: Cálculo de la atenuación de una ruta.

Ante esta situación, utilizando la función "calcularMatricesTransmision", calculamos la matriz de transmisión resultante tanto para la primera como para la segunda ruta. Sólo nos falta calcular las matrices de transmisión T_{LL} en la acometida origen, en el centro de transformación y en la acometida destino:

- Calculo de T_{LL} en el centro de transformación: En este caso simplemente tenemos que tener en cuenta las impedancias de entrada de cada uno de los subarboles no involucrados en la ruta actual. Calculando el paralelo de estas impedancias simplemente calculamos T_{LL}.
- Calculo de T_{LL} en la acometida origen/destino: En esta situación debemos calcular el paralelo de las impedancias del resto de contadores inteligentes, de los contadores no inteligentes y de las ramas hijo de la acometida para posteriormente calcular T_{LL}. Obviamente se tendrán en cuenta estas impedancias siempre que estén presentes en el nodo en estudio.

Finalmente calculamos la matriz de transmisión final multiplicando de forma ordenada cada una de las calculadas anteriormente. De esta manera, utilizando las fórmulas que se encuentran al final del Anexo I ((12) y (13)), obtenemos la atenuación buscada.

Siguiendo este proceso para cada una de las rutas y teniendo en cuenta que la atenuación de un contador con él mismo es idealmente infinita, simplemente colocamos el valor calculado en la posición correcta de la matriz de atenuaciones. En la Figura 46 se ilustra un ejemplo de la matriz de atenuaciones resultante.

```
-1000|-10.636730|-21.234199|-31.837398|-42.631976|-51.468252|
-3.271602|-1000|-10.754898|-21.358097|-32.152675|-40.988951|
-12.703006|-9.588833|-1000|-10.819276|-21.613854|-30.450130|
-23.417184|-20.303011|-10.930254|-1000|-11.011589|-19.847865|
-34.020923|-30.906751|-21.533994|-10.820751|-1000|-9.053811|
-44.528532|-41.414359|-32.041602|-21.328359|-10.725143|-1000|
```

Figura 46: Ejemplo de la matriz de atenuaciones. Atenuaciones en dB.

4.2 Integración con la aplicación Web

Tras haber desarrollado el código encargado de generar la matriz de atenuaciones, el siguiente obstáculo al que nos enfrentamos es a la integración del nuevo módulo en la plataforma web que fue presentada en el capítulo anterior.

Uno de los requisitos fundamentales de este proyecto es añadir la nueva funcionalidad al entorno web existente con total transparencia para el usuario. Es decir, el usuario debe encontrarse el mismo entorno pero con la nueva capacidad de simular Shapefile.

4.2.1 Tratamiento del archivo de entrada

La plataforma web, como ya hemos comentado, va a mantener todos los aspectos tanto visuales como funcionales de cara al usuario. La única diferencia, tras esta actualización, es que el archivo de topología puede ser un archivo.ZIP y no sólo un .XML. Por lo tanto, el usuario visualmente sólo comprobará que en aquellos lugares en los que se hacía mención al archivo S11 ahora pasa a ser denominado "Archivo de topología".

Con el fin de aprovechar todo el desarrollo existente y mantener los datos ya simulados, para la incorporación del nuevo módulo se hace imprescindible no modificar la base de datos. Este requisito se traduce en la no modificación de los modelos Django existentes por lo que la diferenciación del tipo de entrada se lleva a cabo en el archivo de vistas como vemos en la Figura 47.

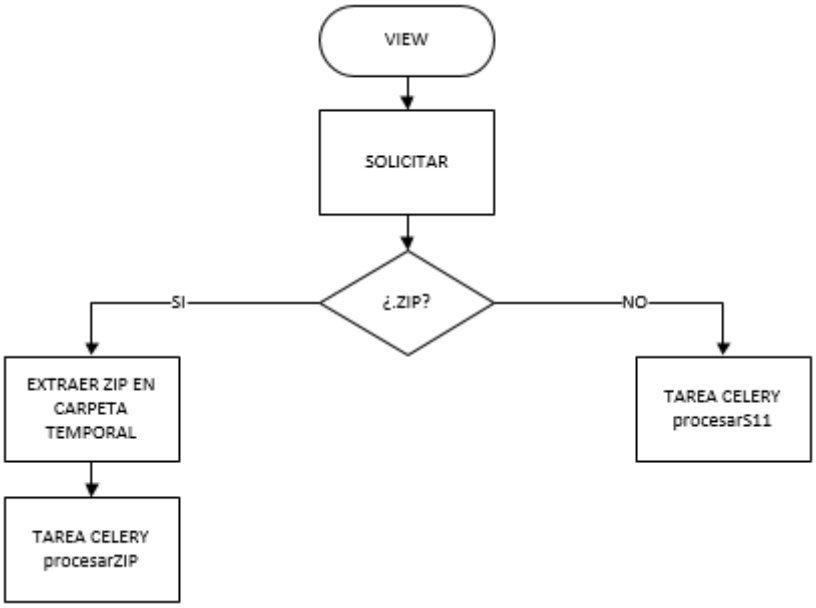


Figura 47: Modificación del archivo views.py.

Por lo tanto, el punto de partida para el estudio de los Shapefiles es descomprimir el archivo .ZIP en la carpeta temporal donde serán procesados lo planos y proporcionar a la tarea "procesarZIP" la ruta de dicha carpeta y el nombre del .ZIP.

4.2.2 Generación de la matriz de atenuaciones

Llegados a este punto, la intervención por parte del usuario ya ha finalizado, con lo que es el propio código Python el que procesará el archivo Excel y los Shapefiles extraídos. Para ello se ha iniciado la tarea Celery "procesarZIP" que, como vemos en la Figura 48, se encarga de llamar a la función "procesarShapefile" encargada de generar la matriz de atenuaciones y guardarla en la BBDD para poder ser utilizada por el simulador posteriormente.

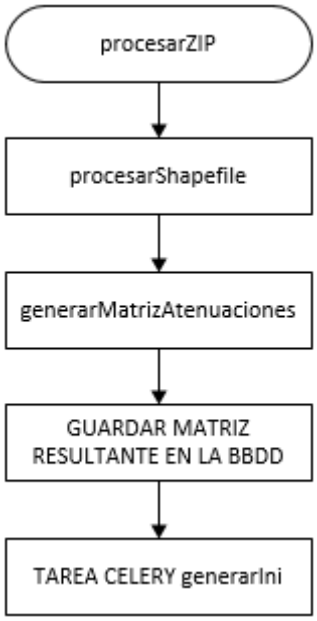


Figura 48: Proceso para generar la matriz de atenuaciones en Django.

De nuevo, todo este proceso ha sido desarrollado intentando imitar cada uno de los pasos que se siguen en el procesamiento del archivo .XML que ya se realizaba en la aplicación original. Esta decisión de diseño aporta sencillez en el la integración y nos garantiza una ejecución correcta que nos proporciona los datos necesarios para iniciar la tarea "generaINI" en base a la topologia dada en los Shapefiles.

4.2.3 Preparación de los datos necesarios para el simulador

Finalmente, ya que simPRIME es una herramienta basada en OMNeT++, se requiere un archivo de configuración .INI donde se establece cómo debe comportarse dicho simulador. Aunque éste tendrá una configuración similar a la detallada en [36], existe una variable que definirá una simulación basada en los Shapefiles facilitados o una simulación en función de los datos del XML: "fixedArch".

Como vemos en la Figura 49, si la variable "fixedArch" se establece con un valor "false" la simulación recreará el escenario descrito por la matriz de atenuaciones generada a partir de los Shapefiles. Es decir, esta variable le dirá al simulador si la topología de la red en estudio es fija y estudiará el escenario presentador por el archivo .XML o la topología no es fija y simulara el escenario dado por los Shapefiles.

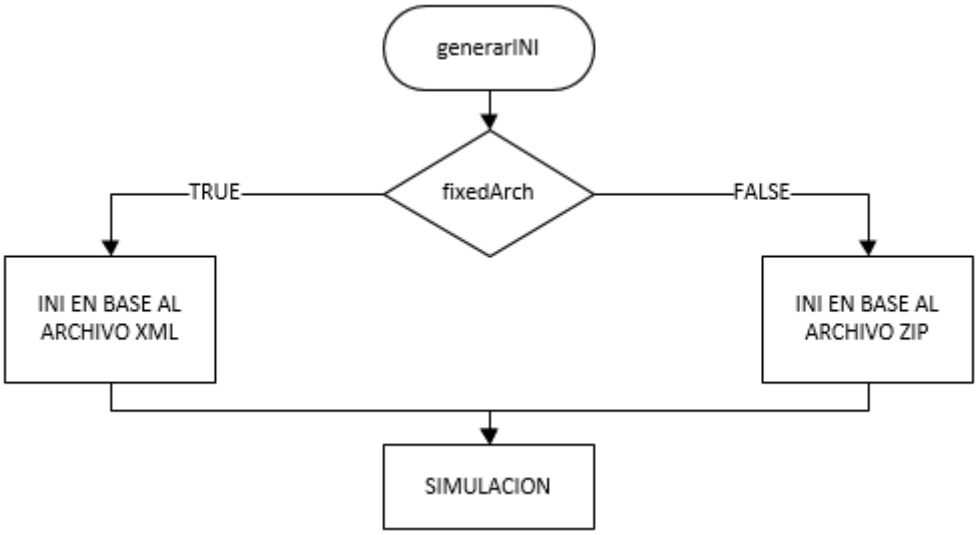


Figura 49: Generación del archivo .INI.

Tras esta distinción, el resto de parametros se establecerán de forma idéntica a como se hacía en la aplicación original para así generar el archico .INI buscado. Con este archivo de configuración, finalmente iniciaremos la tarea de simulación con el simulador simPRIME y simplemente habrá que esperar a que los resultados sean presentados de forma idéntica a como lo eran antes de la incorporación del nuevo módulo.

Capítulo 5

Validación

Este capítulo detalla las diferentes validaciones realizadas para comprobar el correcto funcionamiento del módulo desarrollado. En primer lugar se desarrolla un entorno de pruebas sencillo bajo una topología lineal para validar el correcto funcionamiento del algoritmo de cálculo de la matriz de atenuaciones implementado. Posteriormente se estudia el cálculo de matrices de atenuaciones para una serie de planos en formato Shapefile de redes de distribución eléctrica reales proporcionados por Unión Fenosa. Finalmente, se valida la correcta integración del módulo desarrollado en la aplicación Web.

5.1 Validación con una topología lineal

El desarrollo llevado a cabo ha sido una tarea larga y compleja por lo que la probabilidad de haber cometido errores en su implementación no es despreciable. De esta forma, se hace imprescindible la realización de diferentes pruebas que validen el correcto funcionamiento del *software* desarrollado.

Ya que nos enfrentamos ante un escenario nunca implementado, las posibilidades de encontrar literatura que nos sirva de referencia son remotas. No obstante, en trabajos como el visto en [7] se ilustra la forma de diseñar escenarios sencillos para su posterior cálculo de la matriz de atenuaciones. Bajo estas circunstancias, nuestro primer entorno de validación consistirá en el diseño de una red con una topología lineal como la que se muestra en la Figura 50, para posteriormente ejecutar el código Python desarrollado y comparar los resultados con los obtenidos al de ejecutar el código Matlab validado en [7].

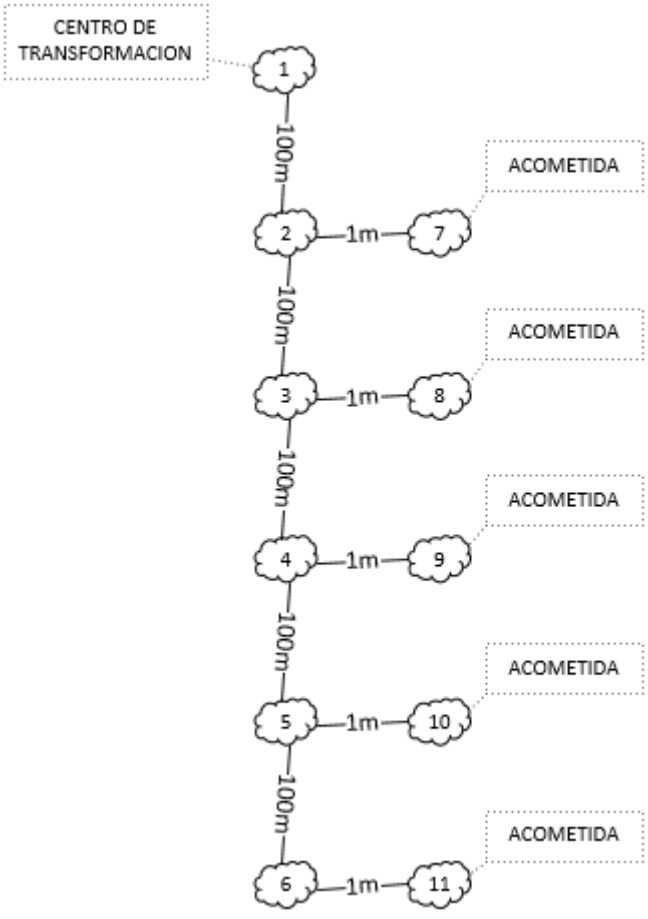


Figura 50: Entorno de validación con topología lineal.

Ya que nuestro propósito es comparar los resultados obtenidos tras la ejecución de códigos diferentes, nos interesa que los resultados sean deterministas, es decir, siempre debemos obtener los mismos resultados independientemente del momento en el que el código sea ejecutado. Por lo tanto, todos los parámetros que forman el escenario son idénticos para ambos códigos (p.ej., se fijan unos valores determinados para las

impedancias), de tal forma que las matrices de atenuaciones resultantes idealmente deben ser iguales.

Una vez implementados los escenarios de forma idéntica, procedemos a la ejecución de ambos códigos obteniendo dos matrices de atenuaciones diferentes como se muestra en la Figura 51.

```
-1000|-10.635696|-21.232246|-31.834512|-42.628308|-51.463640|
-3.271510|-1000|-10.753864|-21.356129|-32.149925|-40.985257|
-12.702031|-9.587834|-1000|-10.818229|-21.612025|-30.447357|
-23.415381|-20.301184|-10.929313|-1000|-11.010691|-19.846023|
-34.018197|-30.904000|-21.532130|-10.819711|-1000|-9.052754|
-44.524883|-41.410686|-32.038815|-21.326397|-10.724108|-1000|

-1000|-10.636730|-21.234199|-31.837398|-42.631976|-51.468252|
-3.271602|-1000|-10.754898|-21.358097|-32.152675|-40.988951|
-12.703006|-9.588833|-1000|-10.819276|-21.613854|-30.450130|
-23.417184|-20.303011|-10.930254|-1000|-11.011589|-19.847865|
-34.020923|-30.906751|-21.533994|-10.820751|-1000|-9.053811|
-44.528532|-41.414359|-32.041602|-21.328359|-10.725143|-1000|
```

Figura 51: Matriz resultante en Matlab y en Python respectivamente.

Se observa que los resultados obtenidos son muy parecidos, existiendo ligeras diferencias que se atribuyen a los diferentes métodos de redondeo de las dos plataformas utilizadas (i.e., Python y Matlab). La Figura 52 ilustra estas diferencias en unidades naturales, observándose que son del orden de 10^{-5}.

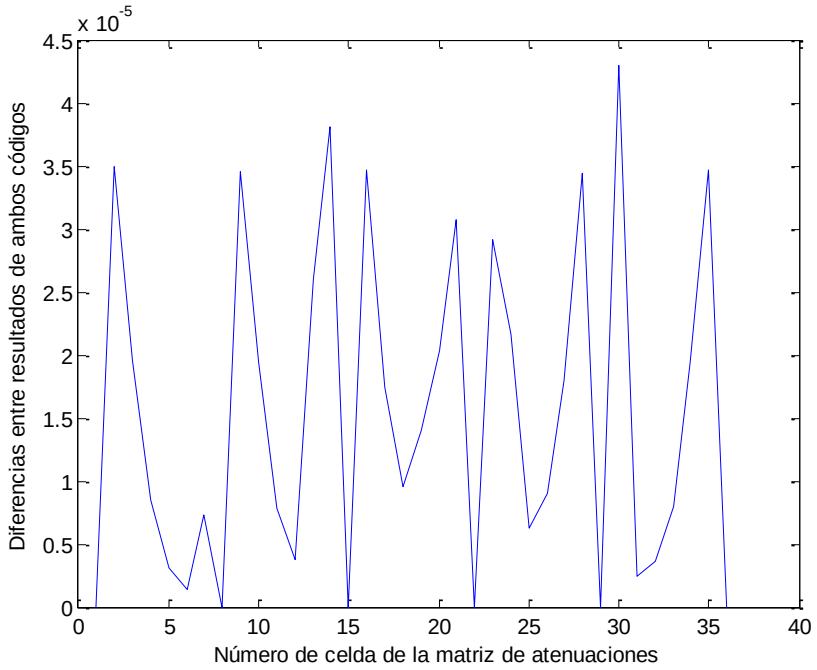


Figura 52. Diferencia entre ambas matrices de atenuación en unidades naturales.

5.2 Validación con mapas Shapefile reales (topología en árbol)

Una vez comprobado el correcto funcionamiento del algoritmo para el cálculo de la matriz de atenuaciones para un escenario sencillo, el siguiente paso es comprobar que efectivamente es capaz de procesar la información contenida en los Shapefiles facilitados. Es decir, debemos validar que a partir de un Shapefile concreto junto con el archivo Excel que lo complementa el código desarrollado genera una matriz de atenuaciones correcta.

Para este nuevo proceso contamos con 5 mapas diferentes facilitados por Unión Fenosa a lo largo del proyecto por lo que hemos podido comprobar el funcionamiento del módulo desarrollado para 5 escenarios diferentes:

- **Escenario 1**: Este primer conjunto de planos fue facilitado principalmente para la familiarización con el formato Shapefile. Como se presentó en el capítulo 2, el formato Shapefile contempla múltiples opciones para la creación de mapas por lo que tuvieron mucha utilidad al comienzo de la etapa de desarrollo. No obstante, entre los archivos que proporcionaron para este escenario no facilitaron el archivo Excel necesario para interpretar los contadores de interés. Por lo tanto, bajo este escenario no se ha podido realizar ningún tipo de validación ya que no contamos con los archivos de entrada necesarios. La Figura 53 muestra este escenario.

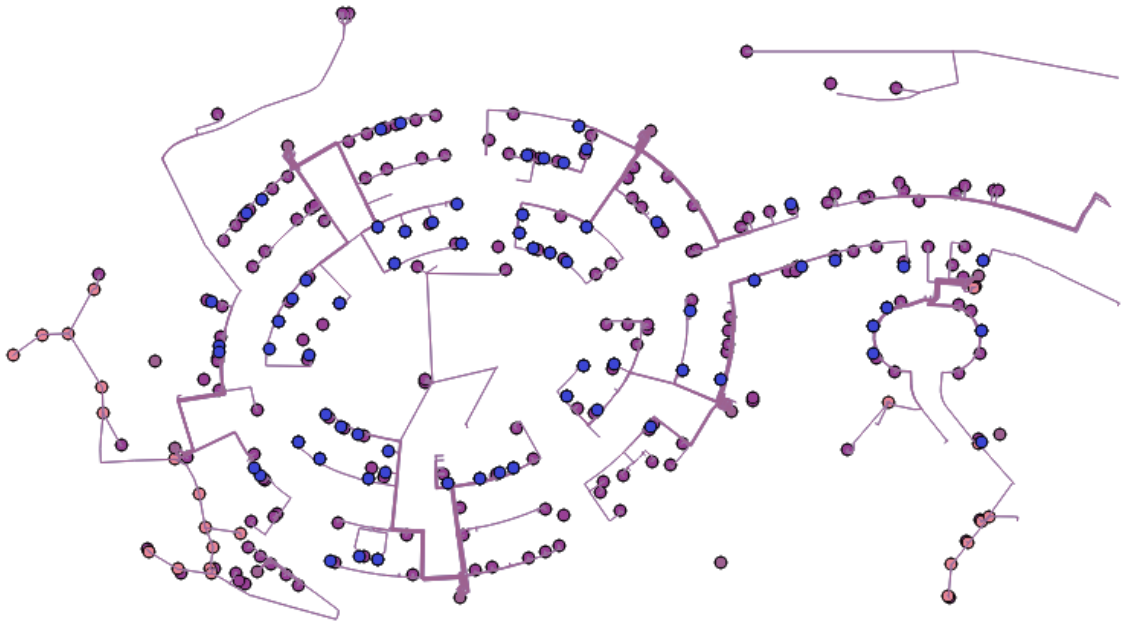


Figura 53.Escenario 1

- **Escenario 2**: En este nuevo caso contamos con todos los archivos necesarios para realizar una validación completa, es decir, generar la matriz de atenuaciones a partir de los Shapefiles y del Excel. No obstante, el resultado de

la ejecución fue satisfactorio pero no se generó la matriz buscada, debido a que existen desconexiones en los planos, por lo que se informa de tal error tras la ejecución. Por lo tanto, este escenario permite validar el correcto funcionamiento del sistema ante desconexiones en los cables representados en los Shapefiles. La Figura 54 muestra este escenario.

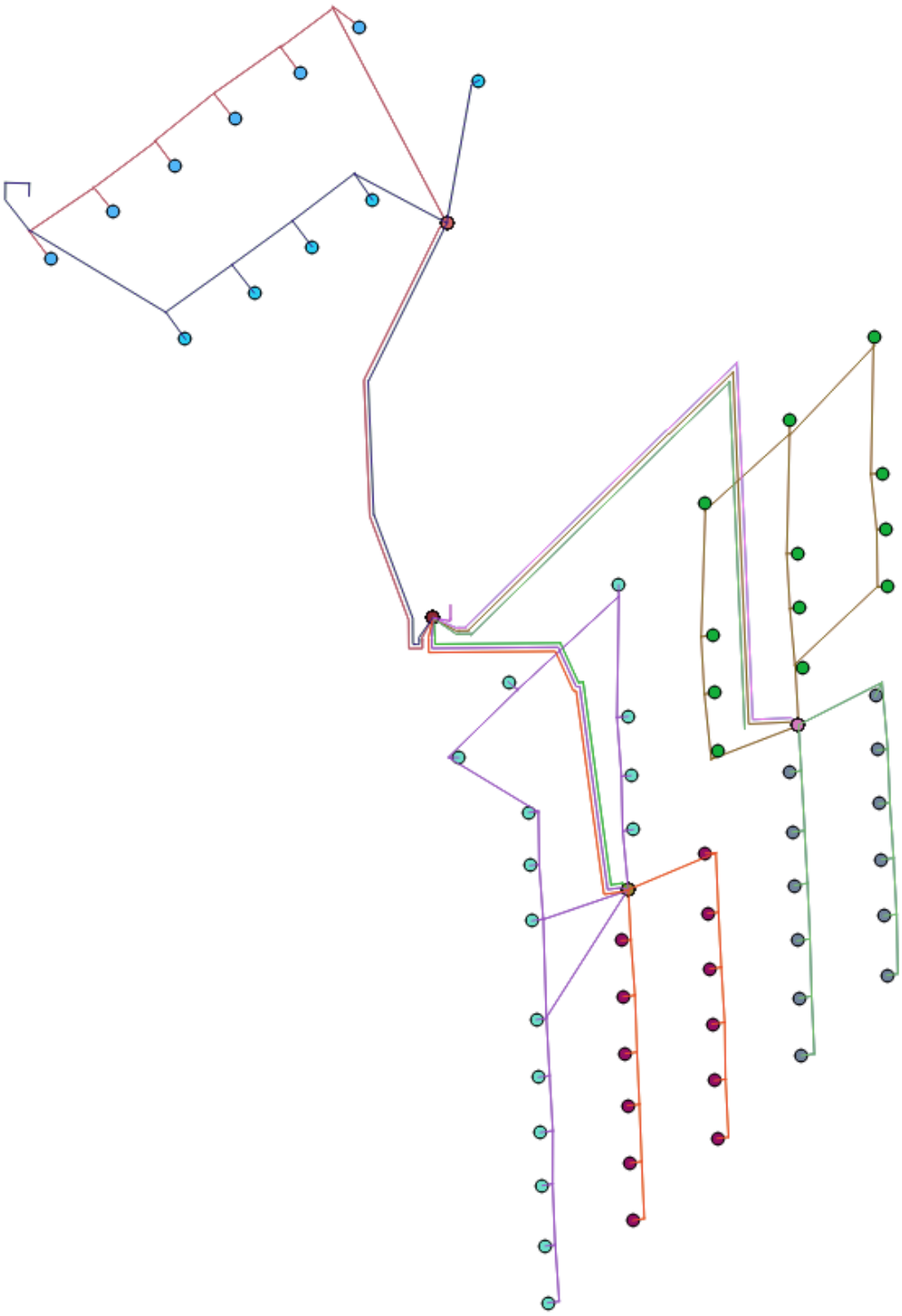


Figura 54.Escenario 2

- **Escenario 3**: En esta ocasión nos enfrentamos nuevamente a un conjunto de archivos completo que permite realizar la validación del módulo *software* desarrollado. Tras esperar unos minutos a que la ejecución finalizara, pudimos comprobar cómo la matriz de atenuaciones fue generada. Como es lógico, se procedió al análisis de los resultados y vimos que este escenario cuenta con cierta peculiaridad: contiene una distribución de conexiones que permiten la existencia de bucles. Por lo tanto, este escenario permite validar que el módulo desarrollado es capaz de analizar la red y evitar la existencia de bucles para generar la matriz buscada. La Figura 55 ilustra este escenario.

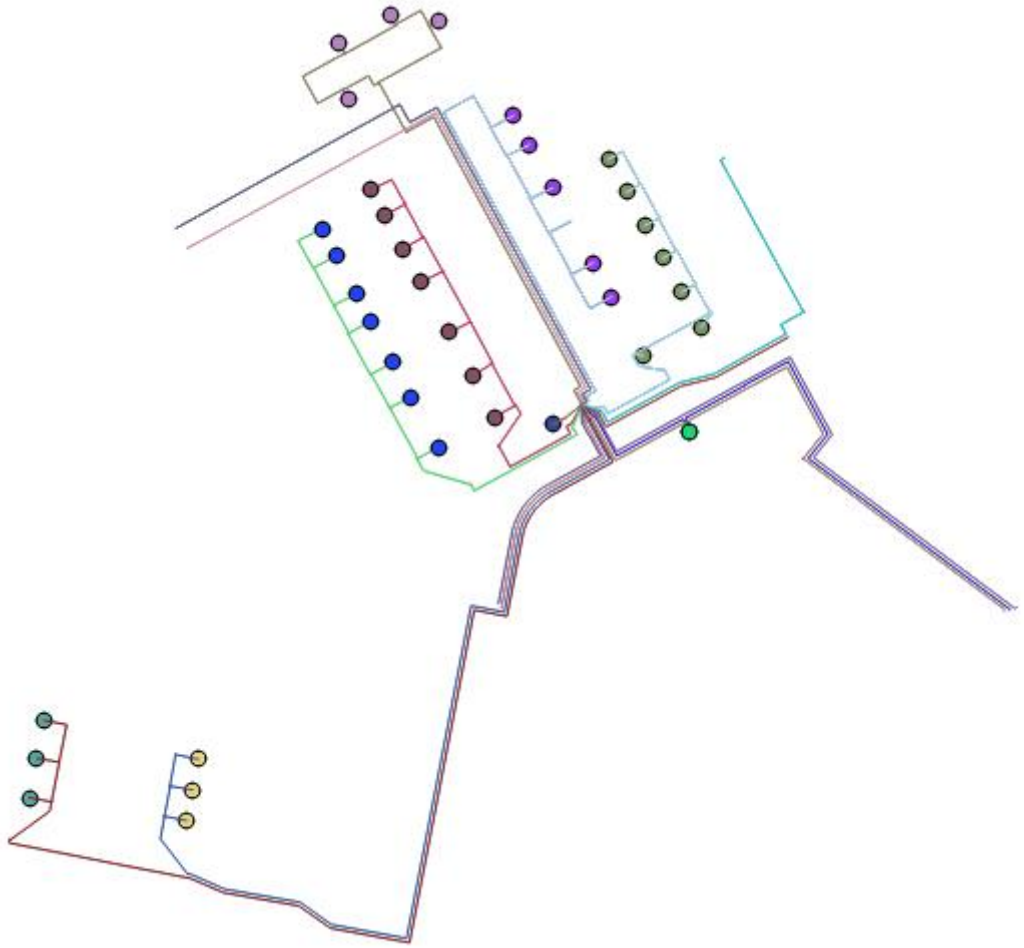


Figura 55.Escenario 3

- **Escenario 4**: Una vez más, contamos con los archivos necesarios y la ejecución es capaz de finalizar correctamente. En esta ocasión, la estructura del cableado no cuenta con la posible existencia de bucles pero si con una gran cantidad de contadores en cada acometida (dado que se trata de bloques de edificios) que lo hacen de gran interés. Como dato extrapolable al caso anterior, se comprueba que los valores de la matriz resultante son razonables en cuanto a valor y distribución. La Figura 56 muestra este escenario.

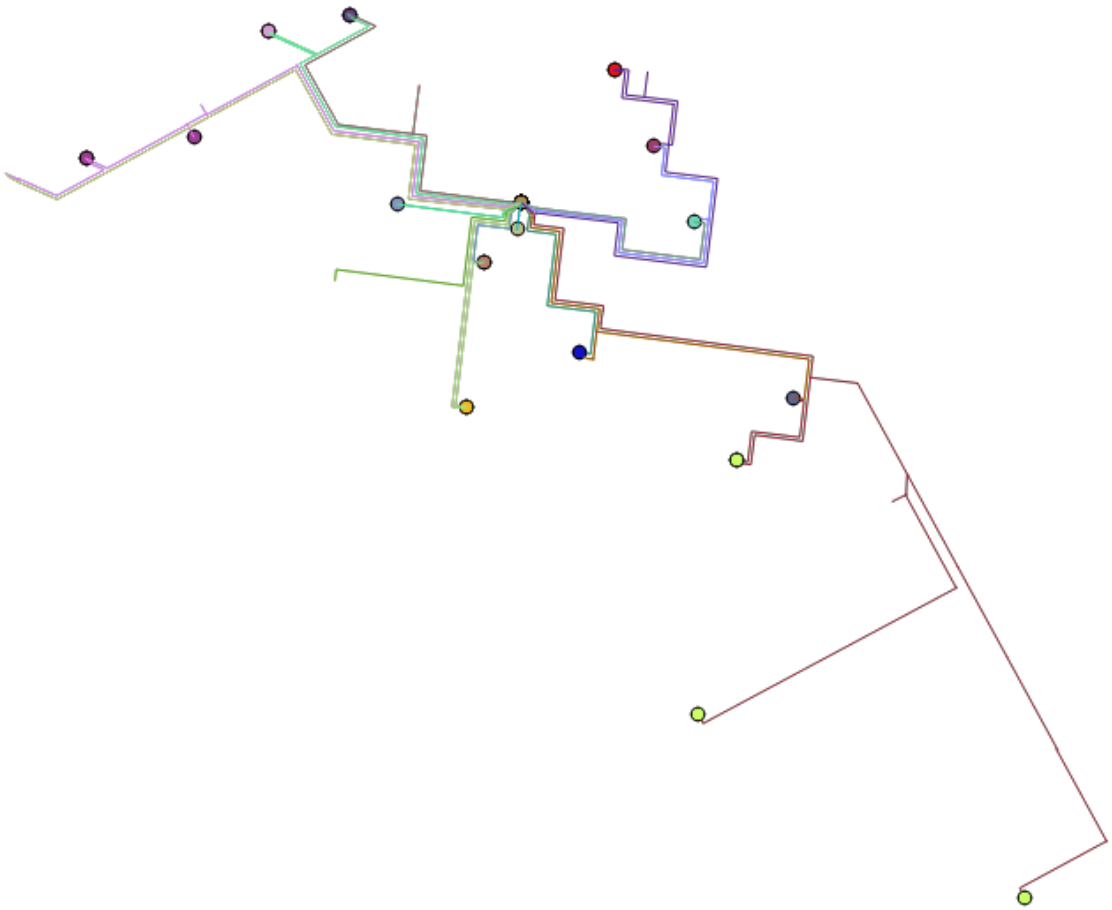


Figura 56.Escenario 4

- **Escenario 5**: Finalmente contamos con un caso ciertamente contrario al anterior ya que, en esta ocasión, estamos ante planos con viviendas unifamiliares, es decir, en cada acometida sólo hay un contador. Volvemos a obtener una matriz de atenuaciones con unos resultados razonables que permiten dar por finalizado este proceso de validación. La Figura 57 ilustra la topología física de este escenario.

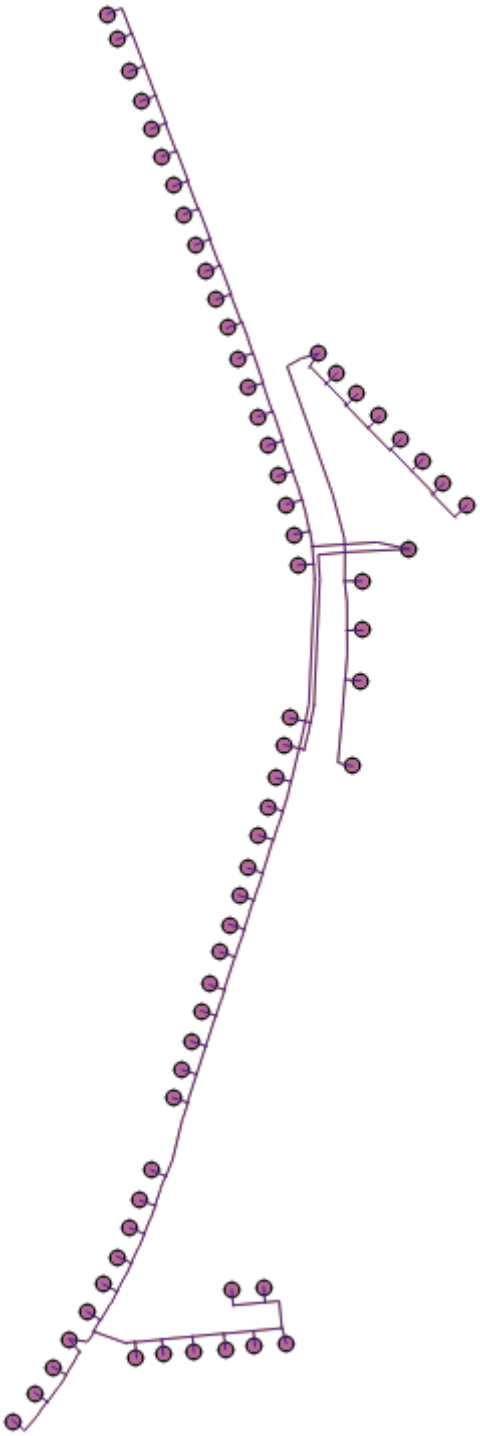


Figura 57.Escenario 5

5.3 Validación global

Una vez validado que el módulo *software* desarrollado genera matrices de atenuación correctas a partir de mapas en formato Shapefile, se procedió a validar la correcta integración de dicho módulo en la aplicación Web basada en Django.

Esta prueba consiste básicamente en configurar y lanzar una simulación desde la aplicación Web proporcionando como entrada un fichero ZIP con el plano de una red de distribución eléctrica en formato Shapefile y el fichero EXCEL adicional, en lugar del fichero XML de topología lógica (S11.XML), y comprobar que dicha simulación se ejecuta correctamente.

Estas simulaciones se ejecutaron en el servidor del grupo de investigación del Departamento de Ingeniería Telemática en el que se ha desarrollado este PFC, que presenta las siguientes características:

- Microprocesador: 2 Intel Xeon de 2 núcleos cada uno a 3.2 GHz
- RAM: 8 GB
- Disco: 500 GB
- Sistema Operativo: Ubuntu Server de 64 bits

En primer lugar se probó a simular el Escenario 4 (610 contadores inteligentes), pero esta opción se descartó debido a que se comprobó que después de un día de ejecución el progreso de la simulación no había pasado del 1%.

Por lo tanto, se decidió ejecutar esta prueba con el Escenario 5 (63 contadores inteligentes) y se comprobó que las simulaciones se ejecutaron con éxito, obteniéndose la gráfica asociada al tiempo necesario para obtener los informes de consumo diarios S02 de todos los contadores de la red PRIME después de que el concentrador envíe solicitud a tal efecto (TTRAll) que se muestra en la Figura 58.

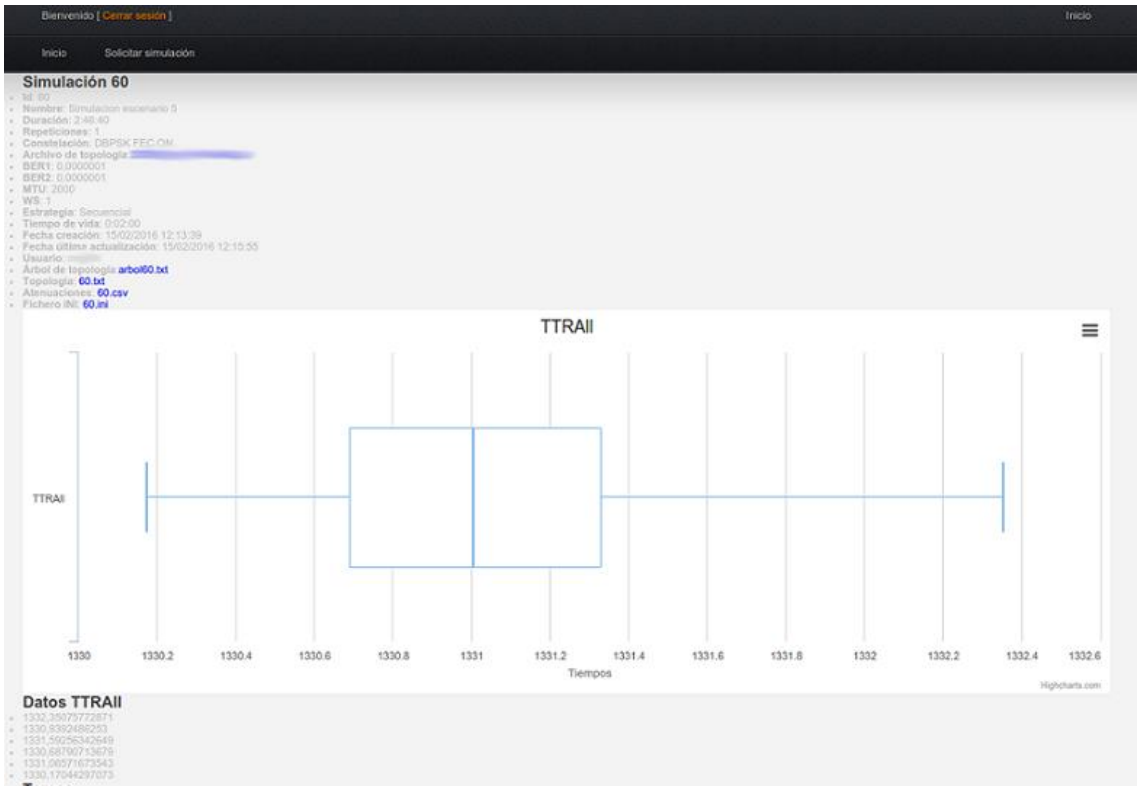


Figura 58.Resultado de la simulación del Escenario 5 desde la aplicación Web

Capítulo 6

Conclusiones y líneas de trabajo futuras

En este capítulo se exponen las conclusiones obtenidas tras la ejecución del presente Proyecto Fin de Carrera. Además, se proponen una serie de mejoras que podrían ayudar a mejorar el sistema en un futuro.

6.1 Conclusiones

En el presente Proyecto Fin de Carrera se ha diseñado, desarrollado y validado un módulo *software* capaz de procesar mapas de redes de distribución eléctrica en formato Shapefile y se ha integrado en una aplicación web para simulación de redes PRIME, permitiendo la simulación de redes PRIME desplegadas en campo. En consecuencia, la nueva versión del simulador web permitirá evaluar problemas existentes en redes PRIME en operación así como diferentes soluciones, reduciendo drásticamente el tiempo y el coste de solventarlos.

El trabajo ha sido desarrollado dentro del ámbito del proyecto de investigación nacional OSIRIS (Optimización de la Supervisión Inteligente de la Red de Distribución), financiado por el Ministerio de Economía y Competitividad y liderado por Unión Fenosa Distribución (tercera distribuidora eléctrica a nivel nacional), lo que demuestra su alineación con las demandas actuales de este sector de la industria. En concreto, Unión Fenosa ha mostrado especial interés en la posibilidad de simular redes PRIME a partir de mapas en formato Shapefile, participando en el proyecto mediante la distribución de archivos que permitieran el correcto desarrollo de la aplicación.

El punto de partida para la realización de este Proyecto Fin de Carrera ha sido la adquisición de conocimientos relativos a las tecnologías PLC de banda estrecha y, particularmente, a las redes PRIME y a los medios existentes para simularlas, como el simulador simPRIME. Una vez situados dentro del ámbito que nos ocupa, procedimos con el estudio de los SIG para posteriormente estudiar en detalle el formato Shapefile.

Con la adquisición de estos conocimientos, junto al repaso de tecnologías de desarrollo como Python y Django, pudimos adentrarnos en la parte de diseño. El objetivo prioritario era añadir una nueva funcionalidad a una aplicación web ya implementada de la forma más transparente para el usuario, es decir, sin condicionar las funcionalidades con que ésta ya contaba. Esta situación supuso un reto desde el punto de vista de diseño, ya que no sólo debíamos desarrollar el nuevo módulo sino también integrarlo, por lo que tuvimos que estudiar en detalle cada una de las partes de la aplicación web de la que partíamos.

Una vez decidida la forma de abordar la integración de nuestro módulo con la aplicación web, comenzamos con la parte de desarrollo para poder procesar los mapas facilitados por Unión Fenosa y obtener los datos necesarios para proceder a su simulación. Los Shapefiles proporcionados contaban con características muy variadas por lo que la complejidad de este desarrollo aumentó considerablemente debido a la continua búsqueda de un código lo más universal posible. Asimismo, la adaptación del método para el cálculo de la función de transferencia del canal descrito en el Anexo I a los mapas Shapefile facilitados también supuso un reto de complejidad considerable.

Una vez finalizada la etapa de desarrollo, fue imprescindible encontrar un método que nos permitiera comprobar que los resultados obtenidos eran correctos. Para ello contábamos con una pequeña aplicación ya validada que proporcionaba unos resultados con la misma estructura que los nuestros. El inconveniente es que los escenarios que era

capaz de analizar eran únicamente lineales por lo que tuvimos que diseñar un entorno simplificado que nos permitiera validar nuestro módulo.

Los resultados fueron satisfactorios por lo que continuamos la validación con el análisis de los Shapefiles facilitados para finalmente comprobar el correcto funcionamiento de la integración del módulo *software* desarrollado en la aplicación Web existente. Es decir, con el módulo correctamente integrado en la aplicación web, procedimos a validar el sistema global introduciendo uno de los mapas en formato Shapefile, junto con la información adicional necesaria, para que fuera procesado y simulado, obteniéndose gráfica asociada al tiempo necesario para obtener los informes de consumo diarios S02 de todos los contadores de la red PRIME después de que el concentrador envíe solicitud a tal efecto (TTRAll).

Con el fin de enriquecer el presente Proyecto Fin de Carrera, el código Python desarrollado ha sido ampliamente comentado y documentado. No obstante, para comprender y reproducir el proceso seguido, es imprescindible contar con ciertos conocimientos técnicos tanto desde el punto de vista del desarrollo *software* como del de análisis de líneas de transmisión.

6.2 Líneas de trabajo futuras

El módulo *software* desarrollado ha proporcionado una nueva funcionalidad a la aplicación web original, lo cual ha supuesto una mejora considerable para el simulador de redes simPRIME. No obstante, existe un amplio abanico de posibilidades para seguir enriqueciendo esta aplicación.

En la mayoría de los casos, cuando se desarrolla una mejora en una aplicación es consecuencia directa de haber encontrado un problema y su correspondiente solución. En este caso, cuando se desarrolló la primera versión de la aplicación web, fue fruto de la complejidad que el uso de un simulador como OMNeT++ suponía. Ahora, al haber añadido la posibilidad de simular redes PRIME desplegadas en campo, que pueden llegar a involucra un número de nodos muy elevado (hasta 800 – 1000 contadores inteligentes), ha surgido un nuevo inconveniente: la falta de recursos computacionales. Éste, evidentemente, es un problema de escalabilidad de la aplicación, pero existe una solución concreta con un gran potencial.

Por su naturaleza basada en la virtualización, la escalabilidad ofrecida por los entornos *Cloud* es una de las soluciones más adoptadas en la actualidad. Empresas como Amazon [46] y Microsoft [47] han invertido gran cantidad de recursos para conseguir unos resultados lo suficientemente atractivos como para captar la atención de todo el sector de las TIC.

La principal ventaja de apostar por un entorno *Cloud* es la capacidad de adaptarse a cada necesidad según varíe la demanda de sus recursos. Es decir, se monitorizan los recursos que se están consumiendo para reservarlos o liberarlos conforme varíe la carga de trabajo. Por lo tanto, estamos ante una nueva línea de trabajo que podría ofrecer un valor añadido a la aplicación web desarrollada.

No obstante, la migración a la nube es una tarea compleja que requiere el estudio de las diferentes tecnologías y propuestas existentes en el mercado [48] para iniciar un proceso crítico para cualquier aplicación.

En cuanto al algoritmo para el cálculo de la matriz de atenuaciones, queda como trabajo futuro el tener en cuenta las pérdidas debidas a cambios de material en los cables eléctricos, tal y como se indicó en el apartado 4.1.5.

Referencias

[1] International Energy Agency, “Technology Roadmaps: Smart Grids”, **2011**.

[2] G. López López, “Contribution to Machine-to-Machine Architectures for Smart Grids”. Tesis Doctoral. Universidad Carlos III de Madrid, **2014**.

[3] G. López, J. I. Moreno, H. Amaris, F. Salazar, “Paving the Road toward Smart Grid through Large-Scale Advanced Metering Infrastructures”, *Electric Power Systems Research*, **2014**. DOI: 10.1016/j.epsr.2014.05.006.

[4] Página web oficial de la PRIME Alliance: http://www.prime-alliance.org/. [Último acceso: 12/02/2016]

[5] ITU-T Standard G.9904, “Narrowband orthogonal frecuency division multiplexing power line comunication transceivers for PRIME networks”, **2012**.

[6] M. Seijo, G. López, J.I. Moreno, J. Matanza, F. Martín, C. Martínez, "Herramientas TIC para la mejora del rendimiento y la planificación de redes PLC para Smart Grids", *XII Jornadas de Ingeniería Telemática - JITEL 2015*. Palma de Mallorca, Spain, **2015**.

[7] J. Matanza Domingo, “Improvements in the PLC Systems for Smart Grids Environments”, Tesis Doctoral, Universidad Pontificia de Comillas, **2013**.

[8] ESRI, “Shapefile Technical Description”, An ESRI White Paper, **1998**.

[9] Página web oficial del proyecto OSIRIS: http://www.unionfenosadistribucion.com/es/redes+inteligentes/investigacion+y+desarrollo/nacionales/1297262412251/osiris.html [Último acceso: 12/02/2016]

[10] G. López, P. Moura, V. Custodio, J. I. Moreno, “Modeling the Neighborhood Area Networks of the Smart Grid”, *IEEE ICC* 2012, Ottawa, Canada, **2012**.

[11] ITU-T Standard G.9903, “Narrowband orthogonal frecuency division multiplexing power line comunication transceivers for G3-PLC networks”, **2012**.

[12] V. Oksman y J. Zhang, "G.hnem: the new ITU_T standard on narrowband plc technology", *IEEE Communications Magazine*, vol. 49, nº 12, pp. 36-44, **2011**.

[13] E. Alonso, J. Matanza, C. Rodriguez-Morcillo y S. Alexandres, "A switch promotion algorithm for improving PRIME PLC network latency", *18th IEEE International Symposium on Power Line Communications*, **2014** doi:10.1109/ISPLC.2014.6812350.

[14] A. Sendin, I. Urrutia, M. Garai, T. Arzuaga y N. Uribe, "Narrowband PLC for LV Smart Grid Services", *18th IEEE International Symposium on Power Line Communications*, **2014** doi:10.1109/ISPLC.2014.6812376.

[15] Página web oficial de Riverbed modeler: http://www.riverbed.com/products/steelcentral/steelcentral-riverbed-modeler.html [Último acceso: 12/02/2016]

[16] Página web oficial de NS-3: https://www.nsnam.org/overview/what-is-ns-3/ [Último acceso: 12/02/2016].

[17] A. Varga y R. Horning, "An overview of the OMNeT++ simulation environment", *Proceedings of the Simutools08 1st International Conference on Simulation tools and Techniques for Communications, Networks and Systems & Workshops*, Bruselas, **2008**.

[18] Simulador PLC basado en NS3 [En línea]. Disponible: http://www.ece.ubc.ca/~faribaa/User_Guide.pdf. [Último acceso: 12/02/2016]

[19] M.-S. Kim, D.-M. Son, Y.-B. Ko, Y.-H. Kim, "A simulation study of the plc-mac performance using network simulator-2", *IEEE Symposium on Power Line Communications and Its Applications (ISPLC)*, **2008**, doi:10.1109/ISPLC.2008.4510406.

[20] A. Sanz, P. Piero, D. Montoro, J. Garcia, "High-accuracy distributed simulation environment for PRIME networks analysis and improvement" , *IEEE Symposium on Power Line Communications and Its Applications (ISPLC)*, **2012**, doi:10.1109/ISPLC.2012.6201298.

[21] A. Gogic, A. Mahmutbegovic, D. Borovina, I. Cavdar, N. Suljanovic, "Simulation of the narrow-band plc system implementing PRIME standard", *IEEE Energy Conference* (ENERGYCON), **2014**, doi:10.1109/ENERGYCON.2014.6850624.

[22] J. Larrañaga, J. Legarda, I. Urrutia, A. Sendin, "An experimentally validated PRIME subnetwork simulation model for utility applications", *IEEE Symposium on Power Line Communications and Its Applications (ISPLC)*, **2015**.

[23] O. G. Hooijen, "A channel model for the residential power circuit used as a digital communications medium", *IEEE Transactions on Electromagnetic Compatibility* vol. 40 no. 4, **1998**, doi:10.1109/15.736218.

[24] "What is Python? Executive Summary | Python.org" [En línea]. Disponible: https://www.python.org/doc/essays/blurb/. [Último acceso: 12/02/2016]

[25] "Comparing Python to Other Languages | Python.org" [En línea]. Disponible: https://www.python.org/doc/essays/comparisons/ [Último acceso: 12/02/2016]

[26] "Python2 or Python3 - Python Wiki" [En línea]. Disponible: https://wiki.python.org/moin/Python2orPython3 [Último acceso: 12/02/2016]

[27] "About the Django Software Foundation | Django" [En línea]. Disponible: https://www.djangoproject.com/foundation/. [Último acceso: 12/02/2016]

[28] "Django Overview | Django" [En línea]. Disponible: https://www.djangoproject.com/start/overview/. [Último acceso: 12/02/2016]

[29] "Django appears to be a MVC framework, but you call the Controller the "view", and the View the "template". How come you don't use the standard names? | FAQ: General | Django documentation | Django" [En línea]. Disponible: https://docs.djangoproject.com/en/1.8/faq/general/#django-appears-to-be-a-mvc-

framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names [Último acceso: 12/02/2016]
[30] "Celery - Distributed Task Queue - Celery 3.1.18 documentation" [En línea]. Disponible: http://docs.celeryproject.org/en/latest/ [Último acceso: 12/02/2016]
[31] "RabbitMQ - Messaging that just works" [En línea]. Disponible: http://www.rabbitmq.com/ [Último acceso: 12/02/2016].
[32] "PostgreSQL: About" [En línea]. Disponible: http://www.postgresql.org/about/. [Último acceso: 12/02/2016]
[33] "Sistemas de Información Geográfica, tipos y aplicaciones empresariales" [En línea]. Disponible: http://sig.cea.es/SIG [Último acceso: 12/02/2016]
[34] A.Pérez, "Introducción a los Sistemas de Información Geográfica y Geotelemática – Primera Edición", **2011**.
[35] Project Management Institute (PMI), "A Guide to the Project Management Body of Knowledge (PMBOK Guide) - Fifth Edition", **2013**.
[36] A. Ferré, "Aplicación web para la automatización y visualización de simuladores de redes PRIME", Proyecto Fin de Carrera, Universidad Carlos III de Madrid, **2015**.
[37] "Cython C-Extensions for Python" [En línea]. Disponible: http://cython.org/ [Último acceso: 12/02/2016]
[38] "Dijkstra shortest path algorithm based on python heap implementation" [En línea]. Disponible: https://gist.github.com/kachayev/5990802 [Último acceso: 12/02/2016]
[39] "zipfile - Work with ZIP archives" [En línea]. Disponible: https://docs.python.org/3/library/zipfile.html [Último acceso: 12/02/2016]
[40] "pyshp 1.2.3 - Pure Python read/write support for ESRI Shapefile format" [En línea]. Disponible: https://pypi.python.org/pypi/pyshp [Último acceso: 12/02/2016]
[41] L. Lampe and A. Vinck, "On cooperative coding for narrow band PLC networks", *AEU – International Journal of Electronics and Communications*, vol. 65, pp. 681-687, **2011**.
[42] K. Doster, M. Zimmermann, T. Waldeck and M. Arzberger, "Fundamental Properties of the Low Voltage Power Distribution Grid Used as a Data Channel", *European Transactions on Telecommunications*, vol. 11, pp. 297-306, **2000**.
[43] "openpyxl 2.3.3 – A Python library to read/write Excel 2010 xlsx/xlsm files" [En línea]. Disponible: https://pypi.python.org/pypi/openpyxl [Último acceso: 12/02/2016]
[44] "QGIS - Un Sistema de Información Geográfica libre y de Código Abierto" [En línea]. Disponible: http://www.qgis.org/es/site/ [Último acceso: 12/02/2016]
[45] "Introducción y configuración del Spanning Tree Protocol (STP) en los switches Catalyst" [En línea]. Disponible: http://www.cisco.com/cisco/web/support/LA/7/73/73037_5.html [Último acceso: 12/02/2016]
[46] "Amazon Web Services" [En línea] Disponible: http://aws.amazon.com [Último acceso: 12/02/2016]
[47] "La Nube de la nueva empresa" [En línea] Disponible: https://azure.microsoft.com/es-es/ [Último acceso: 12/02/2016]
[48] "IaaS, PaaS, SaaS, Nubes Privadas y Públicas e ITaaS" [En línea] Disponible: http://blogs.technet.com/b/davidcervigon/archive/2010/11/21/iaas-paas-saas-nubes-privadas-y-p-250-blicas-e-itaas.aspx [Último acceso: 12/02/2016]

ANEXO I

Modelado de la función de transferencia del canal

En este anexo se detalla el proceso mediante el cual se obtiene la matriz de atenuaciones, tal cual se describe en la sección 3.3.1 de [7].

I.I Cálculo de la función de transferencia

La mayor atenuación de la señal en una red PLC es debida a las pérdidas en el medio de transmisión (cable) y a las terminaciones (cargas). Un cable puede ser modelado de forma precisa utilizando sus parámetros por unidad de longitud. Estos son la resistencia, la capacidad, la inductancia y la admitancia por unidad de longitud (R, C, L y G respectivamente). Estos parámetros se utilizan para calcular la impedancia característica del cable (Z_C) y la constante de propagación (γ):

$$Z_C = \sqrt{\frac{R+j\omega L}{G+j\omega C}} \quad (1)$$

$$\gamma = \alpha + j\beta = \sqrt{(R + j\omega L).(G + j\omega C)} \quad (2)$$

Estos parámetros también son conocidos como parámetros de línea de transmisión. En las ecuaciones anteriores α y β representan los cambios en la atenuación y en la fase por unidad de longitud respectivamente medidos en Np/m y rad/m.

Alternativamente, la matriz de transmisión de una red de dos puertos describe la relación entre los valores de tensión y corriente de ambos puertos. La Figura 59 ilustra una red de dos puertos definida por sus parámetros ABCD. La ecuación que vemos a continuación define la relación entre los pares (U_1, I_1) y (U_2, I_2) utilizando la matriz de transmisión considerando la dirección de las corrientes y de las tensiones como se representa en la Figura 59.

$$\begin{bmatrix} U_1 \\ I_1 \end{bmatrix} = [T] \begin{bmatrix} U_2 \\ I_2 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} U_2 \\ I_2 \end{bmatrix} \quad (3)$$

Todos los parámetros ABCD pueden ser calculados basándonos en los parámetros de línea de distribución (Z_C y γ) y la longitud de los cables (l). Como vemos en la ecuación anterior, pueden ser usados para relacionar las tensiones de entrada y salida.

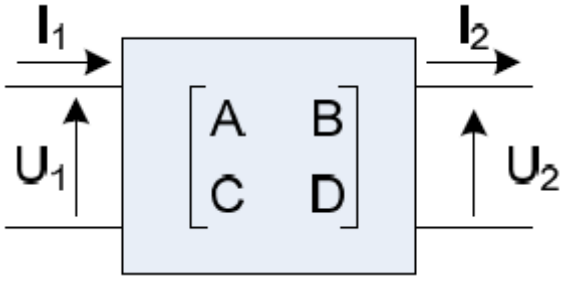


Figura 59: Descripción de una matriz de transmisión de dos puertos.

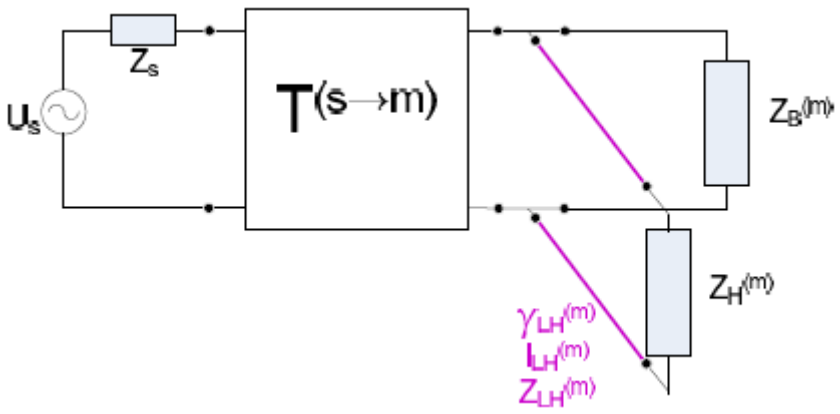


Figura 60: Interconexión del transformador MV/LV con el usuario m-ésimo.

Dada la definición del modelo de matriz de transmisión anterior, el circuito genérico que interconecta el transformador MV/LV con cualquier usuario (m) es representado en la Figura 60. En esta figura, los cables de distribución entre el transformador y el usuario "m-esimo" son modelados por la matriz de transmisión $T^{s\to m}$ mientras que el resto de la red es vista como una impedancia en paralelo $Z_B^{(m)}$. Adicionalmente, la impedancia de entrada del usuario $Z_H^{(m)}$ no está conectada directamente a la línea de distribución principal, por lo que el cable del bucle local (referido por los subíndices LH) también debe ser tenido en cuenta. El proceso a seguir para calcular la función de transferencia ($H^{(s\to m)}(f)$) se detalla en los párrafos siguientes.

En primer lugar, vamos a comenzar con la impedancia $Z_B^{(m)}$, la cual se debe al resto de elementos situados "hacía abajo" del nodo de interés. Con el fin de calcular este valor, se sigue un proceso iterativo comenzando en la impedancia localizada en la posición más alejada del transformador (asumimos que es el nodo localizado en la posición N). La Figura 61 muestra el esquema genérico de la línea de distribución principal y el bucle local para cada usuario.

Empezando por el nodo situado en la posición N, la impedancia vista a la entrada de su bucle local puede ser calculada con la ecuación que calcula la impedancia de entrada de una línea de transmisión cargada:

$$Z_{LL}^{(N)} = Z_{LH}^{(N)} \cdot \frac{Z_H^{(N)} + Z_{LH}^{(N)} . \tanh(\varphi_{LH}^{(N)})}{Z_{LH}^{(N)} + Z_H^{(N)} . \tanh(\varphi_{LH}^{(N)})} \quad (4)$$

donde $Z_{LH}^{(N)}$ representa la impedancia característica de la línea de transmisión del bucle local; $Z_H^{(N)}$, como ya hemos mencionado, es la impedancia de entrada del usuario; y $\varphi_{LH}^{(N)} = \gamma_{LH}^{(N)} . l_{LH}^{(N)}$ representa la propagación a través del bucle local.

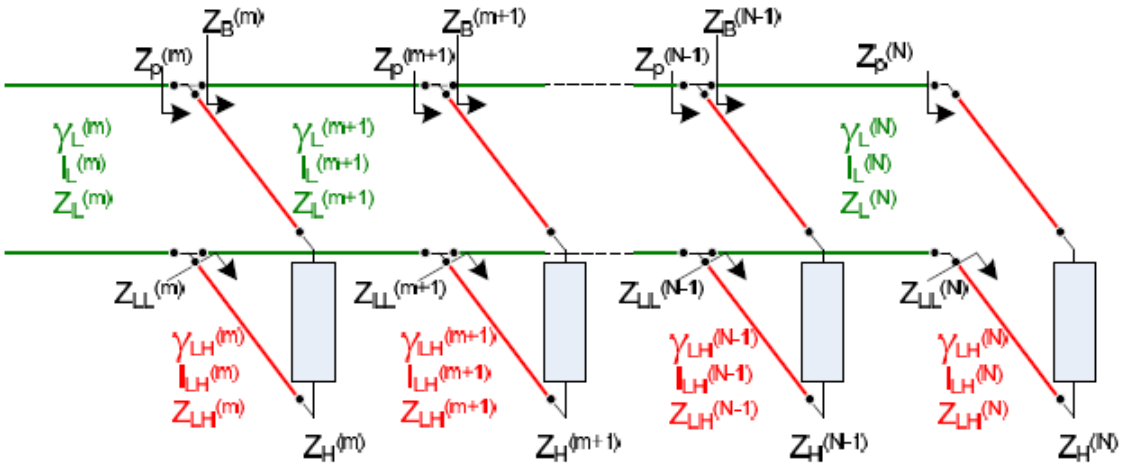


Figura 61: Segmento genérico de red como ejemplo de cálculo impedancia "hacia abajo".

Cuando nos movemos "hacia arriba", el mismo calculo puede ser usado para el nodo N-1. Sin embargo, en este punto, la impedancia de entrada de su bucle local está en paralelo con la impedancia vista "hacia abajo" $Z_B^{(N-1)}$. De forma análoga a la ecuación anterior, esto puede ser calculado como:

$$Z_B^{(N-1)} = Z_L^{(N)} \cdot \frac{Z_{LL}^{(N)} + Z_L^{(N)} \cdot \tanh(\varphi_L^{(N)})}{Z_L^{(N)} + Z_{LL}^{(N)} \cdot \tanh(\varphi_L^{(N)})} \quad (5)$$

Como se muestra en la Figura 61, tanto $Z_B^{(N-1)}$ como $Z_{LL}^{(N-1)}$ se sitúan en paralelo por lo que su valor combinado (expresado con $Z_P^{(N-1)}$) se calcula como:

$$Z_P^{(N-1)} = \frac{Z_B^{(N-1)} \cdot Z_{LL}^{(N-1)}}{Z_B^{(N-1)} + Z_{LL}^{(N-1)}} \quad (6)$$

Para cualquier nodo genérico situado en la posición m, la ecuación anterior puede ser redefinida como:

$$Z_B^{(m)} = Z_L^{(m+1)} \cdot \frac{Z_P^{(m+1)} + Z_L^{(m+1)} \cdot \tanh(\varphi_L^{(m+1)})}{Z_L^{(m+1)} + Z_P^{(N)} \cdot \tanh(\varphi_L^{(m+1)})} \quad (7)$$

$$Z_{LL}^{(m)} = Z_{LH}^{(m)} \cdot \frac{Z_H^{(m)} + Z_{LH}^{(m)} \cdot \tanh(\varphi_{LH}^{(m)})}{Z_{LH}^{(m)} + Z_H^{(m)} \cdot \tanh(\varphi_{LH}^{(m)})} \quad (8)$$

$$Z_P^{(m)} = \frac{Z_B^{(m)} \cdot Z_{LL}^{(m)}}{Z_B^{(m)} + Z_{LL}^{(m)}} \quad (9)$$

Una vez calculada $Z_B^{(m)}$, es el momento de calcular la matriz de transmisión desde el transformador al usuario $T^{s \to m}$. Ahora, comenzamos a calcular desde los nodos situados en las posiciones más cercanas al transformador. La Figura 62 muestra un esquemático del circuito que existe desde el transformador MV/LV a un usuario genérico situado en la posición m.

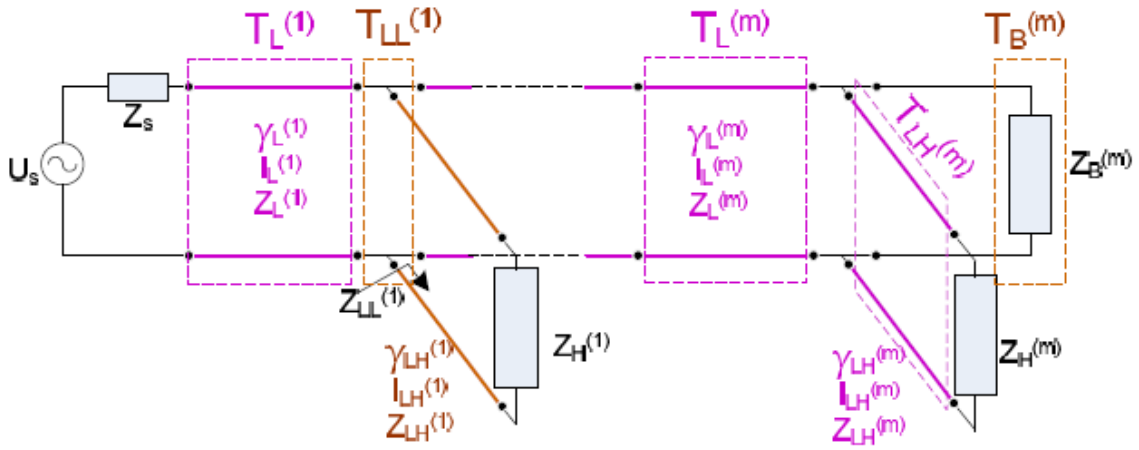


Figura 62: Segmento genérico de red como ejemplo de cálculo de matriz de transmisión

Como podemos ver, los diferentes segmentos pueden ser modelados por sus respectivas matrices de transmisión. El primer segmento es la línea de distribución que hay desde el transformador al primero bucle local. La matriz de transmisión puede ser calculada como:

$$[T_L^1] = \begin{bmatrix} \cosh\left(\varphi_L^{(1)}\right) & Z_L^{(1)} . \sinh\left(\varphi_L^{(1)}\right) \\ \frac{1}{Z_L^{(1)}} . \sinh\left(\varphi_L^{(1)}\right) & \cosh\left(\varphi_L^{(1)}\right) \end{bmatrix} (10)$$

El siguiente segmento es el que corresponde a un bucle local que pertenece a un usuario diferente de m. En este caso, la matriz de transmisión es aquella que en el modelo es una impedancia colocada en paralelo:

$$[T_{LL}^1] = \begin{bmatrix} 1 & 0 \\ \frac{1}{Z_{LL}^{(1)}} & 1 \end{bmatrix} (11)$$

donde $Z_{LL}^{(1)}$ puede ser calculada como la impedancia de entrada en una línea de transmisión cargada. Este mismo proceso se itera "hacia abajo" hasta la derivación "m-esima".

Una vez calculados $Z_B^{(m)}$ y $T_{s\to m}$, el esquema mostrado en la Figura 60 se ha terminado. Ahora, con el fin de alcanzar la carga $Z_H^{(m)}$, la matriz de transmisión de la impedancia en paralelo $T_B^{(m)}$ (ver Figura 63) y la matriz de transmisión del bucle local $T_{LH}^{(m)}$ deben ser incluidos en los cálculos.

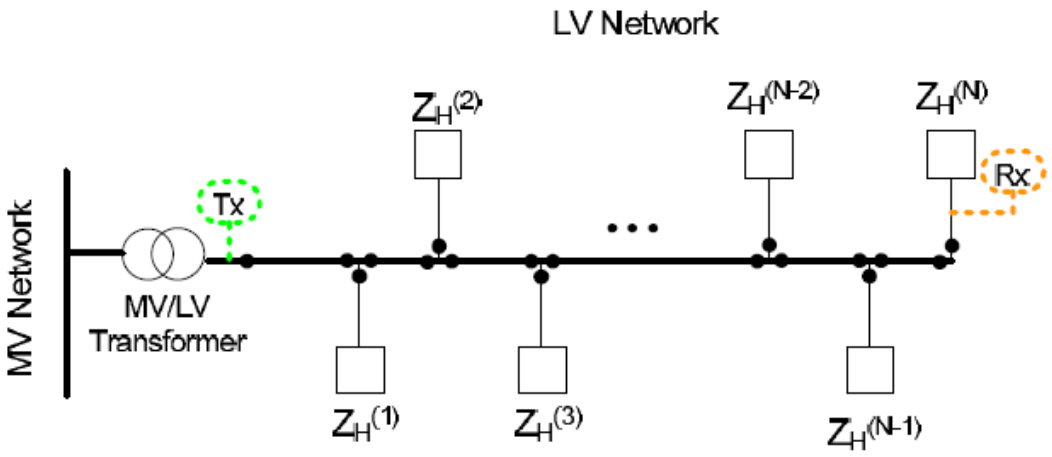


Figura 63: Escenario genérico de transmisión simplificado.

La matriz de transmisión genérica puede ser calculada como la multiplicación de cada una de las matrices de transmisión con las que se ha modelado cada segmento de la red. En el caso del nodo "m-esimo" esto lleva a:

$$T^{(s\to m)} = T_L^{(1)} \cdot T_{LL}^{(1)} \cdot T_L^{(2)} \cdot T_{LL}^{(2)} \dots T_L^{(m-1)} \cdot T_{LL}^{(m-1)} \cdot T_L^{(m)} \cdot T_B^{(m)} \cdot T_{LH}^{(m)} = \begin{bmatrix} A^{(m)} & B^{(m)} \\ C^{(m)} & D^{(m)} \end{bmatrix} \quad (12)$$

Con el fin de obtener la función de transferencia buscada desde la fuente al usuario "m-esimo", los siguientes cálculos pueden ser realizados teniendo en cuenta la definición de la ecuación de los parámetros ABCD presentada al principio de este mismo Anexo:

$$T^{(s\to m)}(f) = \frac{U_m(f)}{U_s(f)} = \frac{Z_H^{(m)}(f)}{A^{(m)}(f) Z_H^{(m)}(f) + B^{(m)}(f)} \quad (13)$$

ANEXO II

Presupuesto

Este anexo presenta la planificación y fases de desarrollo del presente Proyecto Fin de Carrera así como el presupuesto del mismo, desglosándolo en los correspondientes costes de material y costes de recursos humanos.

.

II.I. Planificación

A continuación se resumen las principales fases del proceso de desarrollo llevado a cabo en el presente proyecto, identificándose las tareas más significativas en cada una de ellas:

Fase I. Análisis del estado del arte

- Obtener una visión global de las tecnologías PLC de Banda Ancha haciendo hincapié en las redes PRIME.
- Análisis de los diferentes simuladores de redes que hay en el mercado.
- Estudio de los simuladores de redes PRIME y más concretamente de simPRIME.
- Revisión de las diferentes tecnologías de desarrollo que serán utilizadas.
- Análisis de los sistemas de geolocalización en general y de los Shapefiles en particular.

Fase II. Diseño de la aplicación

- Análisis de la aplicación web que sirve de plataforma para el simulador simPRIME y será la base de nuestra aplicación.
- Definir las vías de desarrollo óptimas para nuestro proyecto.
- Considerar la forma más apropiada para la integración del módulo desarrollado con la aplicación web original.

Fase II. Desarrollo de la aplicación

- Adquisición de datos relevantes de los Shapefiles en estudio.
- Calculo de la matriz de atenuaciones.
- Integración del módulo en la aplicación web del simulador.

Fase IV. Validación

- Diseñar un conjunto de pruebas que permitan comprobar el cálculo correcto de la matriz de atenuaciones.
- Realización de diferentes pruebas con diferentes entradas sobre el sistema completo.

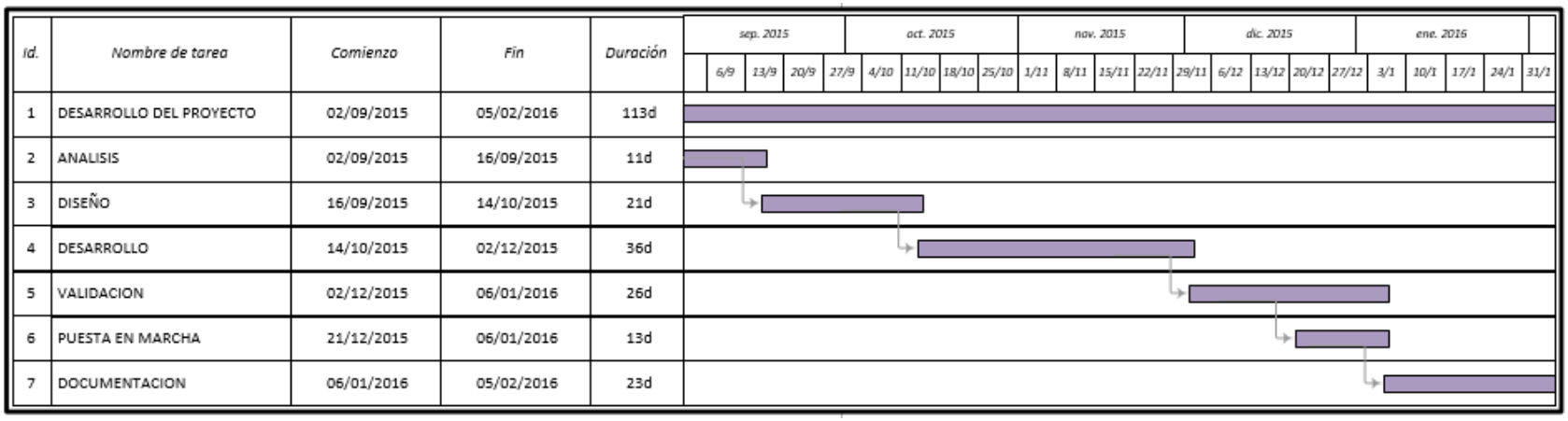
Fase V. Despliegue y puesta en marcha

- Integración final del módulo sobre la aplicación que está actualmente en producción.

Fase VI. Documentación

- Generar un documento que describa el trabajo realizado.
- Generar un juego de transparencias que permita presentar dicho trabajo.

La planificación de la ejecución de estas fases se muestra bajo un diagrama de Gantt en la Figura 64.



Id.	Nombre de tarea	Comienzo	Fin	Duración
1	DESARROLLO DEL PROYECTO	02/09/2015	05/02/2016	113d
2	ANALISIS	02/09/2015	16/09/2015	11d
3	DISEÑO	16/09/2015	14/10/2015	21d
4	DESARROLLO	14/10/2015	02/12/2015	36d
5	VALIDACION	02/12/2015	06/01/2016	26d
6	PUESTA EN MARCHA	21/12/2015	06/01/2016	13d
7	DOCUMENTACION	06/01/2016	05/02/2016	23d

Figura 64: Diagrama de Gantt.

II.II Costes de material

Para la realización de este Proyecto Fin de Carrera se ha requerido el uso de un ordenador portátil y de un monitor. Se considera que ambos equipos tienen una vida útil de 48 meses cuyos cálculos de amortización a 6 meses son presentados en la Tabla 1.

Por otro lado, todo el *software* utilizado ha sido gratuito ya que aquellas aplicaciones que son de pago han sido utilizadas en su versión de prueba.

Concepto	Unidades	Coste/Unidad	Amortización (%)	Total
Ordenador	1	750€	12.5	93.75€
Monitor	1	250€	12.5	31.25€
Total				**125€**

Tabla 1: Costes de material.

II.III Costes de recursos humanos

Para el cálculo de costes de recursos humanos se considera la suma de horas que han sido empleadas para la implementación del proyecto. De esta forma, como se muestra en la Tabla 2, este coste vendrá dado por el coste por persona y hora.

Tarea	Horas trabajadas	Coste/Hora	Total
Análisis	35		
Diseño	50		
Desarrollo	300		
Integración	40	40€	
Validación	25		
Documentación	140		
Total	**590**		**23600€**

Tabla 2: Costes de recursos humanos.

II.IV Costes totales

Como muestra la Tabla 3, la suma de los dos tipos de costes vistos indica que el coste total presupuestado asciende a veintitrés mil setecientos veinticinco Euros.

Concepto	**Total (€)**
Costes de material	125
Costes de recursos humanos	23600
Total	**23725**

Tabla 3: Costes totales.