



Simulación de redes ad-hoc con enlaces PLC e inalámbricos para verificación de calidad de servicio en redes comunitarias

Diego Alexis Romero Rincón

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica
Bogotá, D.C., Colombia
2021

Simulación de redes ad-hoc con enlaces PLC e inalámbricos para verificación de calidad de servicio en redes comunitarias

Diego Alexis Romero Rincón

Tesis presentada como requisito parcial para optar al título de:
Magister en Ingeniería Electrónica

Director:

Ph.D. Carlos Iván Camargo Bareño

Codirector:

Ph.D. Jorge Eduardo Ortiz Triviño

Línea de Investigación:

Redes de Telecomunicaciones Dinámicas y Lenguajes de Programación Distribuidos

Grupo de Investigación:

TLÖN - Grupo de Investigación en Redes de Telecomunicaciones Dinámicas y Lenguajes de Programación Distribuidos

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica
Bogotá, D.C., Colombia
2021

Resumen

Uno de los trabajos más importantes en el desarrollo de las telecomunicaciones es el que aborda los aspectos físicos de los canales de comunicación. Por esta razón, la presente investigación, plantea la simulación, mediante el software de eventos discretos NS3, del canal de comunicación compuesto por el cableado eléctrico, denominado comúnmente Power Line Communication, al interior de una vivienda unifamiliar.

El escenario de simulación se enfoca en redes ad-hoc y concretamente en redes mesh, las cuales poseen estructuras fijas sobre las que se interconectan dispositivos con posibilidad de movilidad. La aplicación específica que se busca evaluar es la de las redes comunitarias, que son infraestructuras de comunicación construidas y mantenidas por los usuarios finales, lo que permite la apropiación de la tecnología por parte de los usuarios finales.

El presente trabajo demuestra la utilidad de los canales de Power Line Communication para la implementación de estas redes tomando como parámetro variable la potencia aplicada para realizar la transmisión sobre la red eléctrica, en un escenario de simulación sobre el software de eventos discretos ns-3.

Palabras clave: Comunicación por línea de potencia, redes ad-hoc, red mallada, red comunitaria, ns-3, simulación..

Abstract

Ad-hoc network simulation with PLC and wireless links for verification of quality of service in community networks.

One of the central works in the development of telecommunications is that which addresses the physical aspects of communication channels. For this reason, the present research work proposes the simulation, using the ns-3 discrete event software, of the communication channel made up of electrical wiring, commonly called Power Line Communication, inside a single-family home.

The simulation scenario focuses on ad-hoc networks and specifically mesh networks that have fixed structures on which devices with the possibility of mobility are interconnected. The specific application to be evaluated is that of community internet networks, which are communication infrastructures built and maintained by end users, which allows the appropriation of technology.

This work demonstrates the usefulness of Power Line Communication channels for the implementation of these networks, taking as a variable parameter the power applied to carry out transmission over the electrical network, over an simulation scenario on the ns-3 software.

Keywords: Power Line Communication, ad-hoc network, mesh network, community network, ns-3, in home, simulation..

Contenido

| | |
|---|----|
| Resumen | v |
| Lista de figuras | ix |
| Lista de tablas | 1 |
| 1. Introducción | 2 |
| 1.1. Objetivo General | 3 |
| 1.2. Objetivos Específicos | 3 |
| 2. Redes Ad-Hoc | 4 |
| 2.1. Redes Móviles Inalámbricas Ad-Hoc | 6 |
| 2.2. Protocolos de Enrutamiento | 8 |
| 2.2.1. Protocolos Proactivos | 10 |
| 2.2.2. Protocolos Reactivos | 13 |
| 2.3. Redes Mesh | 16 |
| 2.3.1. Arquitectura de las Redes Mesh | 17 |
| 2.3.2. Retos en el diseño de redes Mesh | 19 |
| 2.3.3. Redes Comunitarias de Internet | 20 |
| 2.4. Modelo de Tráfico de Audio y Video | 21 |
| 3. Power Line Communication | 23 |
| 3.1. Modelo de Canal en PLC | 24 |
| 3.2. Simulación de canales PLC | 32 |
| 4. Simulación de redes PLC e inalámbricas utilizando NS3 | 36 |
| 4.1. Escenarios de simulación | 36 |
| 4.2. Escenario PLC | 38 |
| 4.3. Escenario Mixto PLC-WiFi | 41 |
| 4.3.1. Primer Escenario Mixto PLC-WiFi | 41 |
| 4.3.2. Segundo Escenario Mixto PLC-WiFi | 42 |
| 4.3.3. Tercer Escenario Mixto PLC-WiFi | 42 |
| 5. Resultados y análisis | 43 |
| 5.1. Resultados y análisis escenario PLC | 43 |

| | |
|---|-----------|
| 5.2. Resultados y análisis en escenario mixto PLC-WiFi | 46 |
| 5.2.1. Resultados y análisis primer escenario mixto PLC-WiFi | 47 |
| 5.2.2. Resultados y análisis segundo escenario mixto PLC-WiFi | 49 |
| 5.2.3. Resultados y análisis tercer escenario mixto PLC-WiFi | 49 |
| 6. Conclusiones y recomendaciones | 57 |
| 6.1. Conclusiones | 57 |
| 6.2. Recomendaciones | 58 |
| A. Anexo: Código de NS3 para la comparación de protocolos de enrutamiento en canal PLC | 60 |
| B. Anexo: Código de NS3 para la comparación de protocolos de enrutamiento en canal inalámbrico | 68 |
| C. Anexo: Código de NS3 para la implementación de primer escenario de simulación mixto PLC-WiFi | 75 |
| D. Anexo: Código de NS3 para la implementación de segundo escenario de simulación mixto PLC-WiFi | 82 |
| E. Anexo: Código de NS3 para la implementación de tercer escenario de simulación mixto PLC-WiFi | 90 |
| Bibliografía | 97 |

Lista de Figuras

| | |
|---|----|
| 2-1. Diferentes topologías en redes de comunicación | 4 |
| 2-2. Redes ad hoc de único y múltiple salto. | 5 |
| 2-3. Diferencias entre red móvil tradicional y red móvil ad-hoc | 7 |
| 2-4. Representación en grafos de una red ad-hoc | 9 |
| 2-5. Estructura básica de los mensajes OLSR. [Barbeau and Kranakis, 2007] | 12 |
| 2-6. Propagación de un mensaje de requerimiento DSR en una red. [Ilyas, 2003] . . | 14 |
| 2-7. Ruta de un mensaje de respuesta DSR en una red. [Ilyas, 2003] | 15 |
| 2-8. Propagación de un mensaje de requerimiento AODV en una red. [Ilyas, 2003] | 16 |
| 2-9. Ruta de un mensaje de respuesta AODV en una red. [Ilyas, 2003] | 17 |
| 2-10. Arquitectura típica de una red mesh.[Leung, 2008] | 18 |
| | |
| 3-1. frequency and time selective PLC channel.png | 26 |
| 3-2. Ejemplo para la determinación de caminos en una red PLC con una derivación A y una discontinuidad C. [Ferreira et al., 2010] | 27 |
| 3-3. Segmento de línea y los parámetros distribuidos que lo representan.[Lutz Lampe, 2016] | 29 |
| 3-4. Modelo de dos líneas concatenadas.[Group et al., 2006] | 32 |
| 3-5. Descripción gráfica del módulo PLC para NS3.[Zarate-Ceballos, 2021] | 34 |
| | |
| 4-1. Plano eléctrico de vivienda unifamiliar.[Zarate-Ceballos, 2021] | 37 |
| 4-2. Diagrama de nodos eléctricos de acuerdo a plano.[Zarate-Ceballos, 2021] . . | 38 |
| 4-3. Diagrama de nodos que componen el backbone de comunicación. | 39 |
| 4-4. Diagrama de enlaces para escenario PLC. | 39 |
| 4-5. Definición de tráfico on-off en el software NS3. | 40 |
| 4-6. Diagrama de enlaces para escenario WiFi. | 40 |
| 4-7. Primer escenario de simulación mixto PLC-WiFi. | 41 |
| 4-8. Segundo escenario de simulación mixto PLC-WiFi. | 42 |
| 4-9. Tercer escenario de simulación mixto PLC-WiFi. | 42 |
| | |
| 5-1. Resultados de simulación para throughput y goodput en red PLC con proto- colo de enrutamiento DSDV | 45 |
| 5-2. Resultados de simulación para throughput y goodput en red PLC con proto- colo de enrutamiento OLSR | 46 |
| 5-3. Resultados de simulación para throughput y goodput en red PLC con proto- colo de enrutamiento AODV | 47 |

| | |
|---|----|
| 5-4. Resultados de simulación para throughput y goodput en red PLC con protocolo de enrutamiento DSR | 49 |
| 5-5. Comparación de resultados de simulación para throughput en red PLC con protocolos de enrutamiento DSDV, OLSR, AODV y DSR | 52 |
| 5-6. Comparación de resultados de simulación para goodput en red PLC con protocolos de enrutamiento DSDV, OLSR, AODV y DSR | 52 |
| 5-7. Comparación de resultados de simulación para throughput y goodput en canales PLC y Wifi con protocolo de enrutamiento DSDV | 53 |
| 5-8. Comparación de resultados de simulación para throughput y goodput en canales PLC y Wifi con protocolo de enrutamiento OLSR | 53 |
| 5-9. Comparación de resultados de simulación para throughput y goodput en canales PLC y Wifi con protocolo de enrutamiento AODV | 53 |
| 5-10. Comparación de resultados de simulación para throughput y goodput en canales PLC y Wifi con protocolo de enrutamiento DSR | 54 |
| 5-11. Comparación de resultados de simulación para throughput en canales mixtos PLC - Wifi con protocolos de enrutamiento DSDV, OLSR y AODV en primer escenario | 54 |
| 5-12. Comparación de resultados de simulación para throughput en canales mixtos PLC - Wifi con protocolos de enrutamiento DSDV, OLSR y AODV en segundo escenario | 55 |
| 5-13. Comparación de resultados de simulación para throughput en canales mixtos PLC - Wifi con protocolos de enrutamiento DSDV, OLSR y AODV en tercer escenario | 56 |

Lista de Tablas

| | |
|---|----|
| 2-1. Diferencias entre las redes Ad Hoc y Mesh [Zhang et al., 2007] | 19 |
| 5-1. Datos de calidad de servicio obtenidos en red PLC. | 44 |
| 5-2. Datos de calidad de servicio obtenidos en red mixta PLC-WiFi en el primer escenario de simulación. | 48 |
| 5-3. Datos de calidad de servicio obtenidos en red mixta PLC-WiFi en el segundo escenario de simulación. | 50 |
| 5-4. Datos de calidad de servicio obtenidos en red mixta PLC-WiFi en el tercer escenario de simulación. | 51 |

1. Introducción

El presente trabajo de investigación parte de la búsqueda de alternativas de comunicación en países como Colombia donde existen amplias regiones que no han sido interconectadas a las redes de comunicaciones y que no han tenido un acercamiento a nuevas tecnologías, indispensables para el desarrollo de los territorios. En este sentido las redes comunitaria, redes construidas de forma autónoma por las comunidades usuarias de la tecnología, se constituyen en una alternativa atractiva, no sólo porque no dependen de empresas tradicionales de comunicación que muchas veces no tienen un interés concreto en instalar redes en estas regiones, sino también porque permite la apropiación de la tecnología por parte de las comunidades, quienes participan de manera activa en el diseño, planeación, construcción y sostenimiento de la red.

Esta forma de comunicación se fundamenta en las redes denominadas mesh, un tipo particular de redes ad-hoc o redes descentralizadas, las cuales se configuran automáticamente de acuerdo con los cambios en su topología, por lo que no requieren de una infraestructura fija, lo que las hace fácilmente escalables. Debido a su carácter descentralizado, las redes ad-hoc y las redes mesh como un caso particular de éstas, permiten no sólo ser usadas como una forma de conexión a Internet, sino que también la implementación de múltiples aplicaciones que no requieren una conectividad a una red mayor, habilitando así un canal de comunicación interno para la comunidad que no requiere la gestión de empresas u organizaciones externas.

Estas redes han sido implementadas tradicionalmente usando canales inalámbricos, sin embargo, las características propias de este tipo de comunicación presentan limitaciones cuando se intentan realizar enlaces a largas distancias o en escenarios donde la geografía no permita una visión directa entre emisor y receptor. Es por esto por lo que la búsqueda de nuevos canales de comunicación se hace esencial en la implementación y extensión de estas redes.

El cableado eléctrico, presente en la mayor parte del territorio nacional, no ha sido diseñado para la transmisión de información, sin embargo, diversos avances en ingeniería han permitido su uso de manera eficiente para este fin. Esta tecnología ha sido denominada comúnmente como Power Line Communication o PLC. Es por esto por lo que el presente trabajo investigativo plantea la simulación de este canal en un escenario delimitado al interior de una vivienda unifamiliar como posible medio de comunicación para implementar y extender las redes mesh en su aplicación para redes comunitarias.

Se plantean, entonces, diferentes escenarios de simulación en los que se intentará demostrar la usabilidad de este canal como medio de comunicación para redes comunitarias. El primero de ellos plantea una comunicación completamente realizada sobre el cableado eléctrico sobre el cual se simularán diferentes protocolos de enrutamiento, fundamentales en el funcionamiento de redes ad-hoc y redes mesh. Los resultados serán comparados con un escenario completamente inalámbrico, teniendo en cuenta que éste ha sido el enlace comúnmente usado para la implementación de estas redes. Igualmente se probará la influencia que tiene la potencia usada para realizar la transmisión sobre los enlaces de comunicación.

Los otros escenarios de simulación plantean una interfaz mixta, es decir que se componen de enlaces PLC e inalámbricos y se probará también la influencia que tienen los protocolos de enrutamiento sobre los enlaces, así como la potencia usada para la transmisión sobre el cableado eléctrico.

Teniendo en cuenta que uno de los usos más comunes para los enlaces de comunicación modernos es la transmisión de audio y video y a la versatilidad que tiene esta aplicación en ámbitos tan diversos como la educación, el entretenimiento o la vigilancia, el presente trabajo se orienta a la verificación de parámetros de calidad de servicio específicos para transmisión de audio y video por demanda, esperando obtener resultados que permitan implementaciones futuras de redes comunitarias sobre este canal.

1.1. Objetivo General

Simular redes ad-hoc con tecnologías PLC e inalámbricas en diferentes topologías para verificación de parámetros de calidad de servicio en redes comunitarias.

1.2. Objetivos Específicos

1. Verificar la eficiencia de las técnicas de enrutamiento reactivas y proactivas en canal PLC con múltiples nodos mediante simulación.
2. Comparar los resultados obtenidos con simulaciones de redes ad-hoc inalámbricas para su aplicación en redes comunitarias.
3. Evaluar los resultados de simulación en tres escenarios de prueba diferentes con enlaces PLC e inalámbricos, verificando parámetros de calidad de servicio para audio y video en las simulaciones.

2. Redes Ad-Hoc

Una red de comunicación es un conjunto de entidades o dispositivos que intercambian entre si datos o información. Si hablamos específicamente de una red de computadores que se comunican entre sí, podrían clasificarse de acuerdo con su tamaño (LAN, MAN, WAN), según el canal de comunicación que utilizan para transmitir información (cableada, inalámbrica, fibra óptica) o por la topología o distribución física e interconexión que poseen los elementos de la red entre si (bus, anillo, estrella, malla) tal como se ilustra en la Figura 2-1.

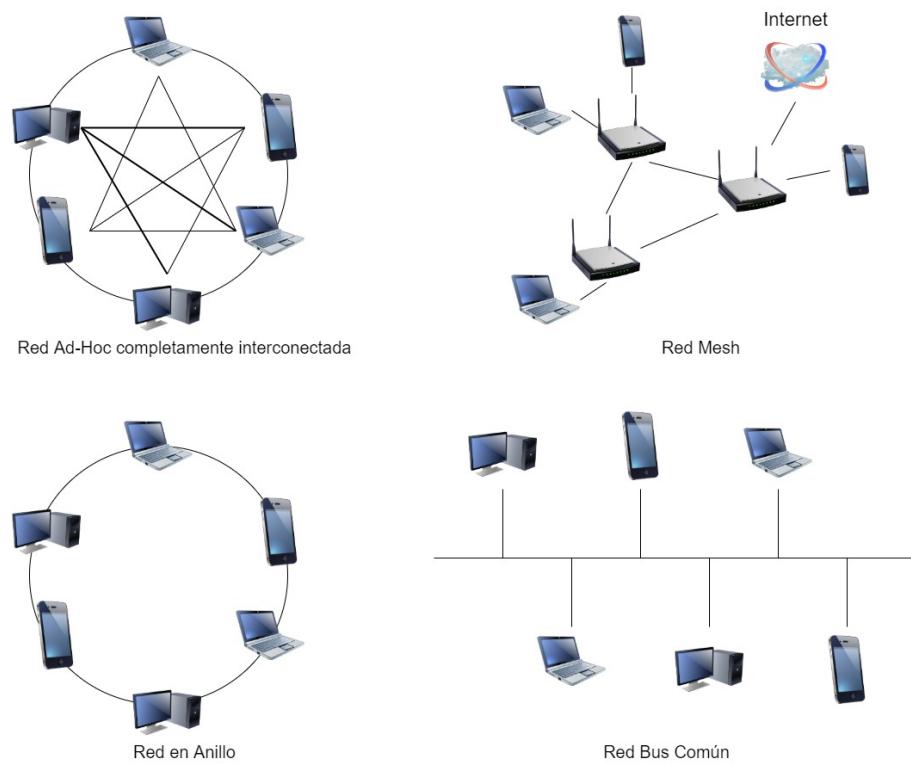


Figura 2-1.: Diferentes topologías en redes de comunicación

Dentro de las posibles redes de comunicación podemos encontrar las ad hoc (locución latina que significa “para esto”) que hace referencia a una red que tiene una configuración específica en un momento de tiempo y que puede cambiar y adaptarse de acuerdo con las necesidades de los participantes de la comunicación. Las redes ad-hoc son redes de comunicaciones descentralizadas, auto-organizables, donde cada uno de los dispositivos o nodos puede ser usado

como emisor, receptor o relevo para transmitir información y alcanzar así dispositivos que pueden no encontrarse disponibles para una comunicación directa entre dos nodos emisor-receptor. Dado que este tipo de redes no dependen de una estructura fija poseen ciertas ventajas, tales como su economía, la escalabilidad, al permitir el ingreso y salida de nuevos nodos y el permitir su movilidad dentro de la red. [Hekmat, 2006]

Por su capacidad adaptativa y su no dependencia a estructuras fijas de comunicación, la mayor utilidad de las redes ad-hoc se encuentra en las redes inalámbricas con dispositivos móviles. De esta forma, al contrario de las comunicaciones basadas en una infraestructura fija, el costo de alcanzar nuevos usuarios a distancias mayores se reduce ostensiblemente, dado que no es necesario el montaje de nuevas antenas. En aplicaciones tradicionales las estructuras fijas han sido de gran ayuda para la interoperabilidad de las redes de comunicación, sin embargo, en usos actuales donde los dispositivos pueden encontrarse en constante movimiento o en zonas distantes no interconectadas su funcionalidad puede verse fuertemente limitada.

Dentro de las redes de comunicación ad-hoc, la más simple es la denominada “single-hop” o de un sólo salto. En estas redes los nodos que se encuentran dentro del rango de alcance mutuo pueden establecer una comunicación directa de forma automática. Las redes “multi-hop” o de múltiples saltos (llamadas también Mobile Ad hoc Networks o MANETS), en cambio, permiten alcanzar nodos que no se encuentran en el rango de comunicación directa, usando otros nodos cercanos como relevo, es decir dispositivos donde la información salta hasta alcanzar su destino final, tal como se muestra en la figura 2-2. [Basagni et al., 2004].

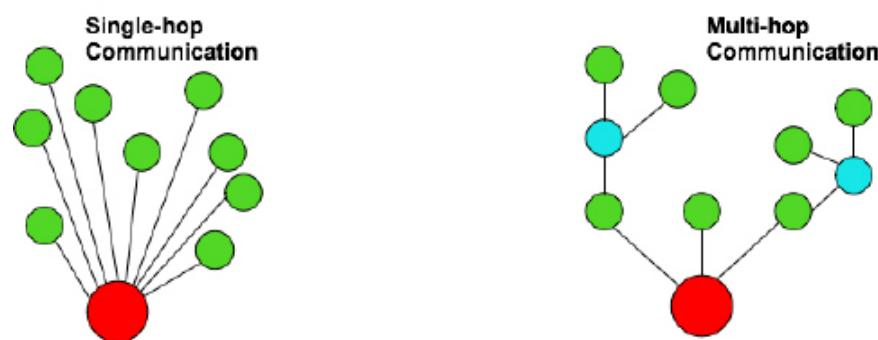


Figura 2-2.: Redes ad hoc de único y múltiple salto.

2.1. Redes Móviles Inalámbricas Ad-Hoc

El uso del espectro electromagnético para realizar transmisiones inalámbricas implica la utilización de tecnologías, protocolos y estándares que permitan la compatibilidad de comunicación entre dispositivos y el uso eficiente del espectro mismo. Dentro de los principales estándares y especificaciones encontramos el Bluetooth que permite establecer redes de corto alcance utilizando la banda de 2.56 GHz que inicialmente se encontraba incluida dentro de las bandas usadas con fines industriales, científicos y médicos pero que, después de la saturación de otros canales, ha sido utilizada de forma amplia en aplicaciones de comunicaciones debido a que no requiere una licencia para su utilización. La tecnología Bluetooth permite establecer redes de hasta 8 dispositivos en un rango de distancias entre 10 y 100 metros con potencias de 1 mW en rangos omnidireccionales. [Sarkar et al., 2016].

A pesar del uso generalizado de la tecnología Bluetooth en aplicaciones de audio y datos para corta distancia, el uso del canal inalámbrico evolucionó a estándares de mayor alcance, versatilidad y seguridad. Los estándares IEEE 802.11 e IEEE 802.16 (wireless fidelity y worldwide interoperability for microwave access, respectivamente).

De esta forma se logró escalar la velocidad de la transmisión sobre el canal, así como la distancia efectiva del envío de información, pasando de 1 Mbps con una distancia de menos de 10 metros, en la tecnología Bluetooth, a picos de hasta 54 MBps con distancias que pueden alcanzar los 50 metros en el estándar 802.11. [Sarkar et al., 2016].

Sin embargo, esta evolución no ha representado la desaparición total de ninguna tecnología, ni su reemplazo en las aplicaciones prácticas de redes de comunicaciones. Mas bien ha orientado un nuevo enfoque en el que los dispositivos pueden aprovechar lo mejor de cada protocolo de comunicación, haciendo necesario que los aparatos tengan ahora la necesidad de poseer más de una interfaz, permitiendo enviar y recibir información de forma constante a través de varios medios y en varios protocolos diferentes.

Dentro de la gran gama de posibilidades que subyacen en las transmisiones inalámbricas, no sólo se encuentra la de incrementar las velocidades o las distancias de transmisión, sino que también puede aprovecharse este canal para crear redes con topologías diversas y, aprovechando la gran ventaja que implica no estar atado a un cableado, redes que cambian continuamente con dispositivos que se están moviendo en un espacio determinado.

Este tipo de transmisión en las comunicaciones da pie a la conformación de redes que, no solo permiten la interconexión entre dispositivos, sino también un cambio constante en su topología, ya que los dispositivos pueden encontrarse en movimiento y salir o entrar constantemente del alcance de otros nodos de la red. Por tal motivo se hizo necesario pensar en

aparatos que contuvieran en sus características más de una interfaz de red, es decir que se pudieran comunicar de múltiples formas e inclusive por múltiples canales y, de igual forma, en redes de comunicación que pudieran configurarse de manera rápida y automática a los cambios que sufre la red debido a la movilidad de los nodos.

Dada esta perspectiva de redes dinámicas y con movilidad, surge el enfoque de redes móviles ad-hoc. Dadas las condiciones dinámicas y auto - adaptables que poseen este tipo de redes, encontramos en su aplicación con dispositivos no fijos algunas ventajas respecto a las redes de comunicación móviles tradicionales. En la figura 2-3 puede observarse la diferencia en estas dos estructuras de comunicación. Mientras en la red móvil tradicional, es necesaria una red de comunicación fija que actúa como troncal para el paso de la información, permitiendo enlazar dispositivos que se encuentran aledaños a alguna antena fija, en las redes móviles ad-hoc no es necesaria infraestructura adicional a los mismos dispositivos finales de comunicación, lo que implica que, al desplazarse éstos en un espacio geográfico, la red podrá desplazarse también, adaptándose a los cambios propios del movimiento de los nodos. [Hekmat, 2006]

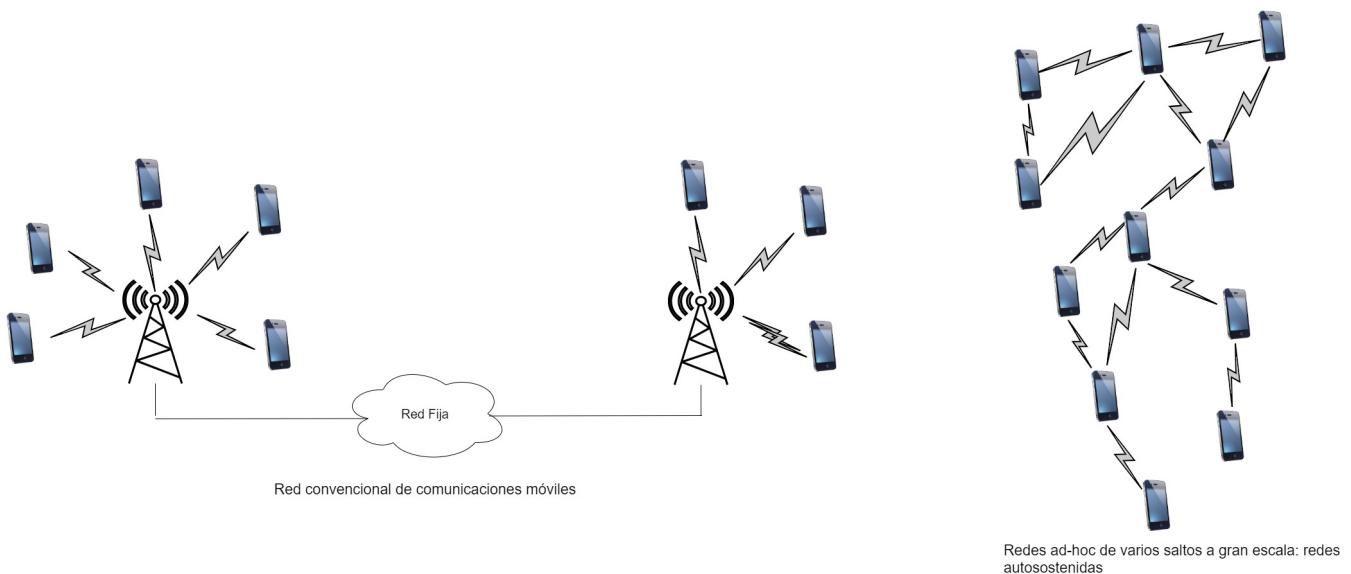


Figura 2-3.: Diferencias entre red móvil tradicional y red móvil ad-hoc

Dependiendo de la aplicación, existen ciertas ventajas al implementar este tipo de redes de comunicación. Entre las principales se encuentran su flexibilidad a la hora de establecer la red, ya que, bajo las configuraciones adecuadas en los dispositivos, al agregar a la red un nuevo dispositivo, este será detectado automáticamente e incluido en la comunicación. Otro factor importante para el uso de este tipo de redes es su bajo costo, dado que no se requiere una infraestructura central, por lo que puede ser usada en situaciones en las que una red de

comunicación tradicional es inviable. De igual forma, dada la auto adaptación de las redes ad-hoc, poseen una gran escalabilidad, lo que permite partir de algunos cuantos nodos y llegar a cientos o incluso miles de ellos en redes que abarcan grandes extensiones geográficas (en este caso es importante mencionar la red desarrollada en España por la fundación Guifi que hoy reúne casi 37 mil nodos [Guifi.net,]).

Sin embargo, lograr este tipo de comunicaciones ha implicado nuevos retos en el desarrollo de ingeniería, con el fin de lograr transmisiones estables y a velocidades que cumplan los requisitos de las aplicaciones contemporáneas. Uno de estos retos es el paso de información entre nodos que no se encuentran comunicados directamente. Dado que no existe una infraestructura fija para realizar esta transmisión, deben entonces aprovecharse los recursos de los otros nodos que la componen para que actúen como relevos en la comunicación. Existen múltiples formas de resolver este problema, los protocolos cooperativos, por ejemplo, realizan una transmisión del mensaje de manera abierta (broadcast) a todos los nodos que se encuentran cerca da él y estos repiten a su vez el procedimiento hasta alcanzar el nodo destino, el cual reconstruye el mensaje a partir de todos los mensajes recibidos.[Nosratinia et al., 2004] [Laneman et al., 2004] Sin embargo la opción más usada es la de enrutamiento mediante saltos a través de rutas preestablecidas, donde los mensajes saltan entre nodos hasta llegar al destino. La forma en la que se realizan estos saltos entre nodos para alcanzar un destinatario final de un mensaje se denomina enrutamiento y existen múltiples formas y protocolos diseñados para este fin.

2.2. Protocolos de Enrutamiento

El enrutamiento es el proceso que permite llevar paquetes de información entre dispositivos para alcanzar un destinatario determinado. Esto se hace necesario en las redes ad-hoc dado que, al no existir una infraestructura fija para establecer la comunicación, es preciso usar todos los dispositivos de la red como relevos para llevar de manera exitosa un mensaje a su destino final.

Este proceso permite la comunicación a distancias más grandes de las que se podría establecer con un solo par de dispositivos con su capacidad y alcance máximo, teniendo en cuenta que al utilizar nodos adicionales que actúan como intermediarios, se extiende el alcance de la comunicación efectiva, utilizando saltos entre nodos para alcanzar el destinatario final.

Este tipo de redes puede ser modelado como un grafo $G = (V, E)$, en el que $V = 1, \dots, n$ sería el conjunto de vértices o nodos que confirman dicha red y E corresponde a los bordes o enlaces que conforman los grafos, es decir cada uno de los enlaces que se observan en la figura 2-4. En el caso concreto de las redes móviles ad-hoc, V correspondería a los dispositivos móviles

que permiten enrutar la información y E a los enlaces inalámbricos que se conforman entre los nodos físicamente adyacentes. Estos enlaces se establecen entre dos nodos cualquiera de la red, cuándo la distancia física entre ellos es menor a un cierto valor que será definido por la tecnología utilizada para la transmisión y las condiciones mismas del ambiente donde ésta se realiza. [Ilyas, 2003]

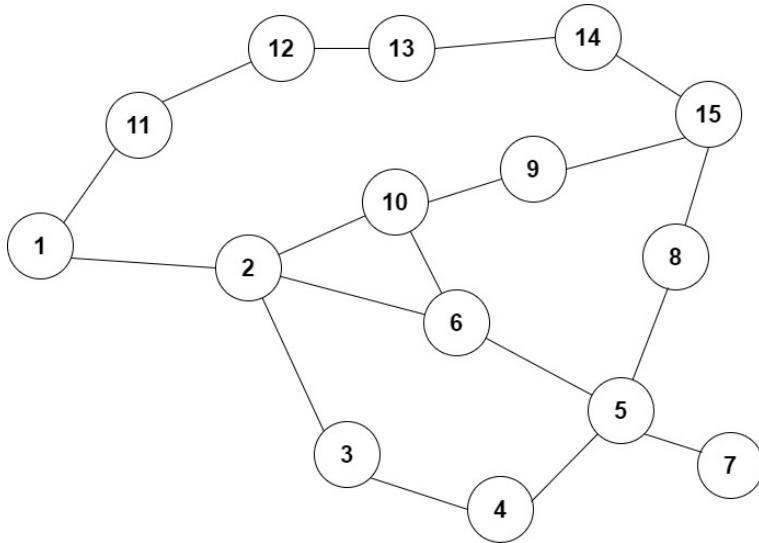


Figura 2-4.: Representación en grafos de una red ad-hoc

Dada la movilidad aleatoria de los nodos, la red se reconfigura constantemente, creándose nuevos enlaces y eliminándose otros debido a la distancia física que los separa. Esto causa cambios en el conjunto E de enlaces, ocasionando cambios topológicos constantes en la red. Bajo estas circunstancias cambiantes, el reto para realizar una transmisión desde un nodo cualquiera hasta otro de la red se fundamenta en encontrar un camino que incluirá una cantidad de saltos que tendrán que efectuar los paquetes de información, antes de encontrar su destino. [Ilyas, 2003]

El camino que seguirá un paquete determinado entre el emisor y el receptor se calcula de manera distribuida sobre la red, utilizando también un algoritmo que permite comprobar los enlaces constantemente para verificar los cambios topológicos y evitar usar caminos que ya no se encuentre disponibles físicamente para la transmisión. De igual forma es común encontrar más de un camino posible para enviar el mensaje desde el emisor hasta el receptor, por lo que también debe existir un algoritmo que permita encontrar la mejor ruta posible, utilizando métricas como la calidad del enlace, el tiempo de latencia o la distancia física más corta entre el emisor y el receptor.

De esta forma, un nodo que actúa como relevo en la comunicación y que recibe un mensa-

je, debe retransmitirlo al siguiente nodo presupuestado en el camino que conecta el emisor con el receptor. Una forma tradicional de realizar esta tarea es a través de una tabla de enrutamiento, que es accesible a todos los nodos y que registra continuamente las rutas más eficientes bajo los parámetros necesarios. Otra opción utilizada para este tipo de enrutamientos consiste en establecer la ruta desde el origen y agregarla al encabezado de cada paquete, a este método se le denomina “enrutamiento desde la fuente”.

El cálculo del camino que tomará la información desde el emisor hasta el receptor es un factor determinante en el funcionamiento de las redes ad-hoc, dado su carácter cambiante, por lo que suelen encontrarse dos métodos fundamentales para realizar esta función. [Ilyas, 2003]

Uno de ellos es el denominado enrutamiento proactivo o enrutamiento basado en tablas. La palabra “proactivo” hace referencia a la característica propia de este método que permite calcular todos los caminos posibles que tomará un paquete, independientemente de si se va a usar o no en la aplicación práctica. Dentro de estos protocolos encontramos algoritmos de vectores de distancia (DV) en el que un nodo envía a todos los nodos adyacentes la tabla completa de enrutamiento que se fundamenta únicamente en la distancia entre nodos y los algoritmos de estado de enlace (LS), en los que se detectan los vecinos de cada uno de los nodos y la distancia a la que se encuentran, construyéndose tablas que permiten establecer un mapa de la topología de la red y, en últimas, calcular los mejores caminos por los que se transmitirán los paquetes de datos.

Al otro grupo de métodos usados en el enrutamiento de paquetes en redes ad-hoc se les denomina protocolos reactivos o protocolos por demanda. En estos métodos el camino que tomará un paquete se calcula sólo cuando es requerido, por lo que la red, incluyendo las actividades de enrutamiento, estará completamente inactiva hasta que alguno de los nodos que la componen se disponga a enviar información.

2.2.1. Protocolos Proactivos

La característica principal de este tipo de enrutamiento es que calcula una tabla de rutas posibles para el envío de información, actualizándola de forma periódica para garantizar la actualización de la información a pesar de los cambios topológicos que pueda presentar la red debido a la movilidad y a la conexión y desconexión de nodos.

Dado que todos los nodos de la red poseen esta tabla actualizada de caminos, al iniciarse el envío de un paquete de datos no hay demora en su cálculo, lo que implica un menor tiempo de transmisión o una disminución de la latencia.

La implementación de este protocolo requiere de la utilización de algunos mecanismos que garanticen un funcionamiento adecuado. Una de ellas es incrementar la información almacenada en cada nodo sobre la topología de la red, lo cual permitirá al algoritmo calcular las rutas de manera más eficiente. De igual forma es necesario variar de forma dinámica la cantidad de información almacenada sobre la red y la frecuencia con que se calculan las tablas. Esto debido a que al incrementarse la red se puede aumentar de manera excesiva la cantidad de información sobre su topología, lo cual puede superar la capacidad de almacenamiento determinada por cada nodo para esta tarea y, de igual forma, podría incrementarse el tiempo usado para difundir y consolidar el contenido de cada tabla. [Ilyas, 2003]

Existen múltiples métodos de enrutamiento proactivo que han sido desarrollados para diversas aplicaciones, sin embargo algunos de los más utilizados son el DSDV (Destination Sequenced Distance Vector) y el OLSR (Optimized Link-State Routing), los cuales serán descritos a continuación.

DSDV

El esquema de enrutamiento por vector de distancia secuenciado por destino o DSDV de sus siglas en inglés (Destination Sequenced Distance Vector). En este protocolo, cada nodo que compone la red funciona como un enrutador, es decir que cada nodo participa de manera activa en el enrutamiento a través de la construcción y difusión de las tablas de enrutamiento. Estas tablas contienen las direcciones de destino de los nodos, la distancia más corta para alcanzar otro nodo de la red, y el nodo siguiente para reenviar la información. [Barbeau and Kranakis, 2007]

Igualmente la tabla contiene de forma general, en el encabezado de los paquetes, la dirección de la red y la dirección del hardware del nodo que transmite estas tablas de enrutamiento. [Sarkar et al., 2016]

Este protocolo de enrutamiento es eficiente en la creación y detección de rutas para la información, por lo que el tiempo que tarde en encontrarlas es bajo. Igualmente, el método garantiza la ausencia total de bucles de enrutamiento, fenómeno que sucede cuando hay errores en el mismo y se crean caminos cerrados en la ruta de la información, por lo que el mensaje queda circulando infinitamente en este bucle. Para evitar este tipo de errores, puede ser utilizado el método de horizonte dividido, método que se basa en prohibir que un nodo envíe la ruta a través de la misma interfaz por la que adquirió dicha ruta. [Barbeau and Kranakis, 2007]

Un ejemplo claro de este tipo de errores es cuando tenemos tres nodos interconectados por enlaces punto a punto de la forma A-B-C, la tabla dirá que la ruta más corta de A a C es pasando a través de B, pero si la conexión B-C se pierde momentáneamente, B deberá tomar

el camino de A y este a su vez devolverá el mensaje a B porque es la ruta más corta y el mensaje quedará rebotando infinitamente. La solución a este problema propuesta por este método es evitar que B considere una ruta a través de A, dado que a través de este nodo se creó la ruta más cercana hacia C. [Goralski, 2002]

Por otro lado, este método de enrutamiento también genera algunos inconvenientes en el uso de la red, dado que debe enviar mensajes continuamente para construir y mantener actualizada la tabla de enrutamiento, lo que utiliza energía y ancho de banda aun cuando la red no esté siendo utilizada en absoluto.

OLSR

En este protocolo de enrutamiento todos los enlaces que se calculan son de tipo simétrico, es decir que se utilizan las mismas rutas en la vía de envío y en la de recepción de la información. En este método también se utilizan tablas que se almacenan en cada uno de los nodos y que contienen la dirección de destino de cada nodo, la dirección del nodo siguiente del relevo para alcanzar el destino y una dirección de interfaz que permite identificar el enlace por el cual se realizará la comunicación con el siguiente nodo, teniendo en cuenta que cada uno puede tener varias interfaces de red.

Al igual que en otros métodos basados en tablas, éstas deben ser actualizadas de manera constante para adecuarse a los cambios en la topología de la red, por esta razón, cuando esta aumenta de tamaño, la tabla va creciendo en la cantidad de ingresos, esto puede ocasionar problemas en el uso de ancho de banda y en el almacenamiento de cada nodo, implicando problemas al escalar la red, por lo que esta técnica se aplica únicamente en redes pequeñas.

El protocolo OLSR utiliza datagramas UDP para enviar y recibir los mensajes de control que permiten crear y actualizar las tablas. Dado que los mensajes UDP funcionan sobre la capa de transporte de red o la capa 4 del modelo OSI, este método puede implementarse como un proceso de usuario que se comunica directamente con el kernel del sistema operativo. [Barbeau and Kranakis, 2007]

El formato del mensaje estándar del protocolo OLSR puede observarse en la figura 2-5. Dada la simplicidad de estos mensajes, un único datagrama de UDP puede contener varios mensajes. Existe un encabezado común a todos los mensajes que contiene la longitud total de la secuencia de mensajes y cada mensaje, a su vez, posee un encabezado que contiene el tipo de mensaje, su tamaño y la dirección de origen. [Barbeau and Kranakis, 2007]



Figura 2-5.: Estructura básica de los mensajes OLSR. [Barbeau and Kranakis, 2007]

2.2.2. Protocolos Reactivos

Una dificultad común que debe afrontarse al resolver problemas computacionales es el sobrecosto (overhead), es decir el gasto excesivo de recursos que puede generarse debido a los métodos usados. Los protocolos de enrutamiento también recaen en sobrecostos al tener que mantener actualizadas las rutas posibles de envío de información y los cambios topológicos que se generen por el movimiento de los nodos, se debe usar canal disponible (el espectro electromagnético, en el caso de las comunicaciones inalámbricas) para enviar mensajes que permitan realizar este mapeo de nodos. Con el fin de reducir al máximo este sobrecosto, necesario para mantener actualizada la topología de la red y encontrar rutas adecuadas para el envío de paquetes, los protocolos reactivos realizan la transmisión de estos mensajes de control sólo cuando se requiere una comunicación, es decir, sólo se actualizan los cambios en la red cuando uno de los nodos que pertenece a esta se dispone a enviar información.

En este tipo de protocolos se utilizan ciclos de consultas y respuestas que se propagan por los nodos hasta alcanzar el destino requerido, conformando en el proceso un mapa de la red que será utilizado para encontrar el camino adecuado para la información, sin embargo los protocolos reactivos no realizan un mapeo de la red en cada transmisión, dado que un camino encontrado en un momento determinado puede tener validez para un instante de tiempo más o menos largo, lo que permitiría el envío de varios paquetes antes de realizar un nuevo descubrimiento de rutas.

A continuación, se describen dos protocolos de enrutamiento reactivos usados tradicionalmente en redes MANET, específicamente los protocolos DSR (Dynamic Source Routing) y AODV (Ad Hoc On Demand Distance Vector).

DSR

El enrutamiento de origen dinámico o DSR por sus siglas en inglés, es un protocolo reactivo comúnmente usado en redes móviles ad hoc que establece las rutas cada vez que se requiere una comunicación, por lo que puede ser dividido en dos componentes fundamentales: El descubrimiento de rutas para el envío de información y el mantenimiento de estas rutas. [Sarkar et al., 2016]

El descubrimiento de rutas se establece mediante mensajes de requerimiento de ruta (RREQ

o Route Request) y las respuestas a estos mensajes (RREP – Route Reply). Estos mensajes se originan en el nodo emisor de la transmisión en el momento en el que éste requiere iniciar el envío de información y se propaga a través de la red, acumulando la información de los caminos.

La figura 2-6 muestra cómo, a partir del nodo emisor, se van acumulando las rutas a medida que los mensajes RREQ y RREP se propagan a través de la red hasta llegar al nodo receptor. El descubrimiento inicia cuando el emisor desea enviar un mensaje y no tienen almacenado en su caché la ruta para llegar a él. En ese momento se envía el paquete RREQ que incluye la dirección del remitente, la dirección de destino, un número único que identifica el requerimiento y un registro de la ruta. Al recibir un mensaje de requerimiento, un nodo que no es el destinatario del mensaje puede hacer dos cosas: Si el nodo intermedio tiene almacenada una ruta en su memoria, devolverá al nodo de origen un mensaje RREP en el que se concatena la ruta acumulada desde el origen con la ruta almacenada hasta el destino; Si el nodo ya recibió ese mensaje a través de otro nodo, la otra opción que se toma es descartar el paquete y, por último, si el nodo no ha recibido el mensaje ni posee una ruta almacenada, agrega su identificación en el campo de la ruta y transmite el mensaje a todos los nodos contiguos. [Ilyas, 2003]

Cuando el nodo destinatario finalmente recibe el mensaje, devuelve otro RREQ al nodo emisor a través de la ruta encontrada mediante el procedimiento acumulativo, tal como se muestra en la figura 2-7. En este mensaje se encuentra entonces la ruta que será usada tanto para el envío del mensaje original desde el emisor hasta el receptor, como la ruta inversa en caso de que el nodo destino quiera, a su vez, enviar mensajes al emisor. [Ilyas, 2003]

AODV

El protocolo de enrutamiento ad hoc por vector de distancia por demanda o AODV por sus siglas en inglés utiliza un número de secuencia para reemplazar las rutas calculadas con anterioridad y evitar bucles en el enrutamiento. De igual forma, este protocolo utiliza un método similar para descubrir nuevas rutas, es decir a través de la propagación de mensajes RREQ y RREP, sin embargo, se diferencia del protocolo anterior debido a que la ruta que se descubre desde el emisor hasta el nodo destinatario no se almacena en los encabezados de los paquetes, sino que se almacena de forma local en los nodos.

Al igual que en el protocolo DSR, el descubrimiento de las rutas inicia cuando un nodo desea enviar datos a otro de la red y no posee información de un camino para realizarlo. En ese momento el mensaje RREQ se propaga a los nodos vecinos, que a su vez enviarán un mensaje por el camino inverso hacia el origen, grabando la dirección del nodo adyacente del que recibió el mensaje por primera vez. De igual forma, cuando un paquete RREQ se envía

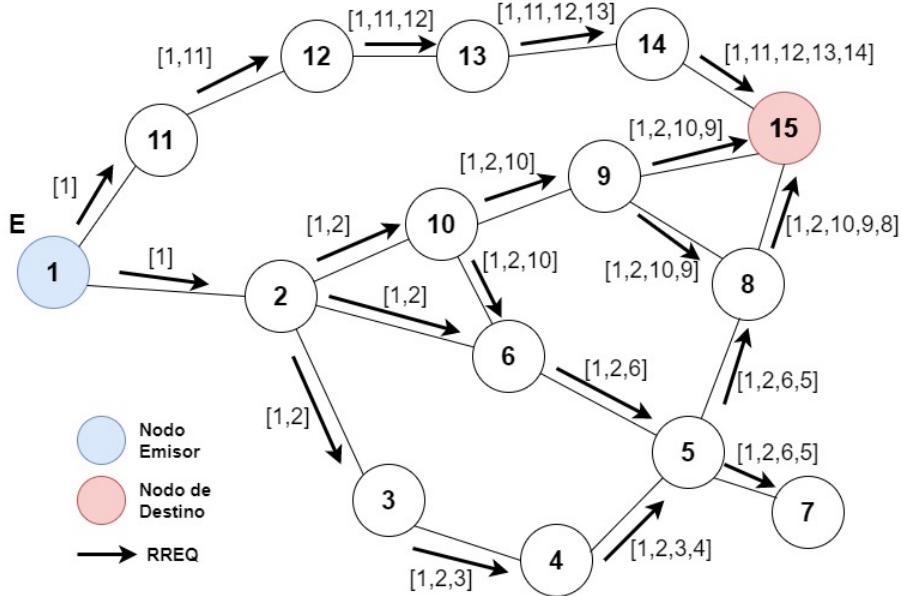


Figura 2-6.: Propagación de un mensaje de requerimiento DSR en una red. [Ilyas, 2003]

hacia el destino desde un nodo, éste establece automáticamente la ruta inversa de todos los nodos hasta llegar al nodo emisor y las otras rutas inversas se eliminan después de un tiempo determinado. El proceso de propagación del mensaje RREQ y los mensajes inversos en uno y otro sentido se encuentran ilustrados en las figuras 2-8 y 2-9. [Ilyas, 2003]

Una vez determinados los caminos, el protocolo se encarga de mantener estas rutas utilizando mensajes de reconocimiento que se envían periódicamente. Si el mensaje falla en alguno de los caminos planteados el nodo envía a los adyacentes un mensaje denominado “Respuesta de Ruta no Solicitado” para que desactiven y dejen de usar esta ruta y estos nodos envían a su vez el mismo mensaje a los vecinos hasta completar la totalidad de la red. Cuando esta información llega al nodo emisor del mensaje original, el cual desencadenó el descubrimiento de rutas, éste envía un nuevo requerimiento de rutas para iniciar nuevamente el proceso de descubrimiento.

2.3. Redes Mesh

Las redes mesh o redes malladas, se basan en estructuras de comunicación híbridas en las que se combinan las propiedades de las redes tradicionales basadas en infraestructura generalmente estática y las redes ad hoc móviles. Este tipo de redes representa una gran oportunidad en diversas áreas de aplicación, con ellas pueden construirse múltiples aplicaciones como redes de banda ancha para el hogar, redes comunitarias, manejo coordinado de redes, sistemas

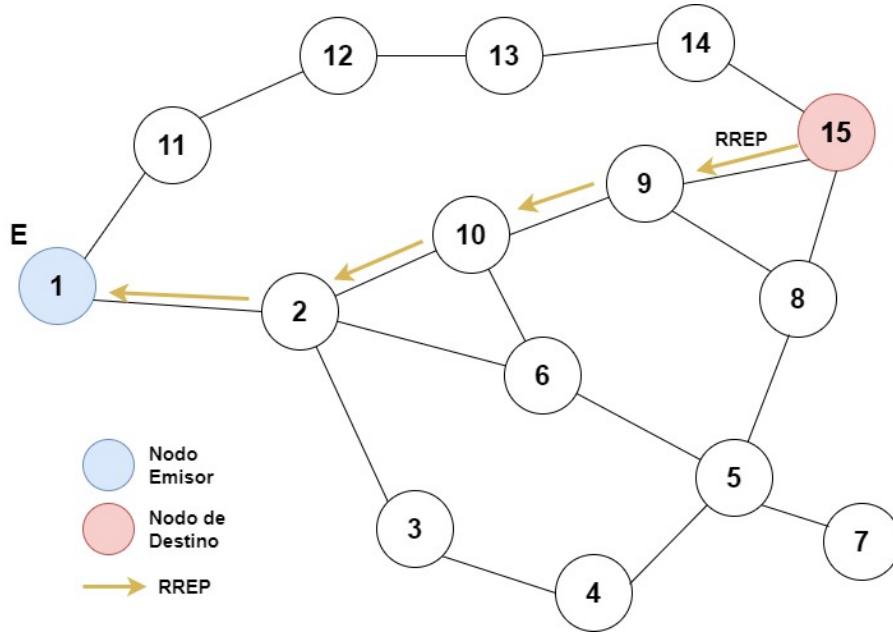


Figura 2-7.: Ruta de un mensaje de respuesta DSR en una red. [Ilyas, 2003]

inteligentes de transporte, Smart Grids, IoT, entre otras muchas. [Leung, 2008]

La perspectiva ad hoc que plantean las redes mesh, permite un mayor dinamismo en contraste con las redes de comunicación tradicionales basadas en infraestructuras fijas. La capacidad de auto organización de estas redes representa ventajas evidentes para los usuarios finales de las redes, dentro de las cuales se tienen el bajo costo en su instalación y las redes robustas que pueden lograrse a medida que crece. De igual forma, las tecnologías usadas para la implementación de este tipo de redes permiten utilizar múltiples bandas del espectro electromagnético e incluso múltiples canales de comunicación, lo cual facilita la implementación de las redes y mejora el alcance que pueden lograr.

2.3.1. Arquitectura de las Redes Mesh

La arquitectura de este tipo de redes plantea la utilización de diferentes tipos de dispositivos, tal como se observa en la figura 2-10. Se evidencia que existe una infraestructura fija, conformada por enrutadores y gateways o puertas de enlace, y una estructura móvil, que se encuentra conformada por dispositivos conectados a los enrutadores o a los gateways y que también pueden establecer comunicación entre sí. [Leung, 2008]

La infraestructura fija actúa como el “backbone” de comunicación, es decir, brinda una estructura base para realizar la transmisión de información y alcanzar eventualmente al usuario

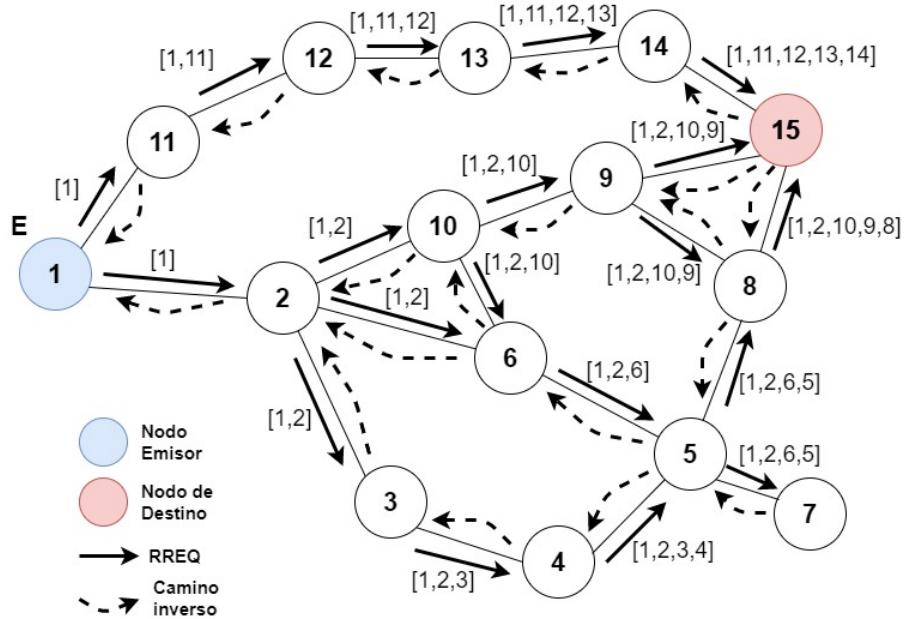


Figura 2-8.: Propagación de un mensaje de requerimiento AODV en una red. [Ilyas, 2003]

final. La diferencia fundamental que encontramos respecto a las redes tradicionales es que estos enrutadores no sólo permiten la conexión a la Internet o a una red de comunicación más amplia, sino que, dada su condición de red ad hoc, permite conectarse entre sí a todos los dispositivos que pertenecen a la red.

En algunas aplicaciones prácticas, los enrutadores pueden actuar de manera indistinta como Gateways al interior de la red, por lo que, al obtener una conexión a internet, ésta será compartida a los demás usuarios a través de la comunicación descentralizada que se establece entre los enrutadores. Estos enrutadores, debido a su carácter fijo, están conectados generalmente a la red eléctrica, por lo que no existen restricciones amplias de energía, lo que permite utilizar múltiples interfaces de comunicación, generalmente las bandas de 2.5 y 5 GHz.

De igual forma es común encontrar que los dispositivos finales se conectan directamente a alguno de los enrutadores pero que no se conectan entre sí para formar una red ad hoc. Esto se debe a que la implementación de conexiones entre los dispositivos implicaría la instalación de software o configuraciones adicionales, por lo que la perspectiva tradicional de estas redes suele considerar que los usuarios móviles se desplacen entre las zonas de cobertura de cada enrutador para realizar su conexión a la red.

Existen claras diferencias entre las redes mesh y las redes ad hoc planteadas con anterioridad, dado que éstas últimas presentan un comportamiento altamente dinámico en su topología

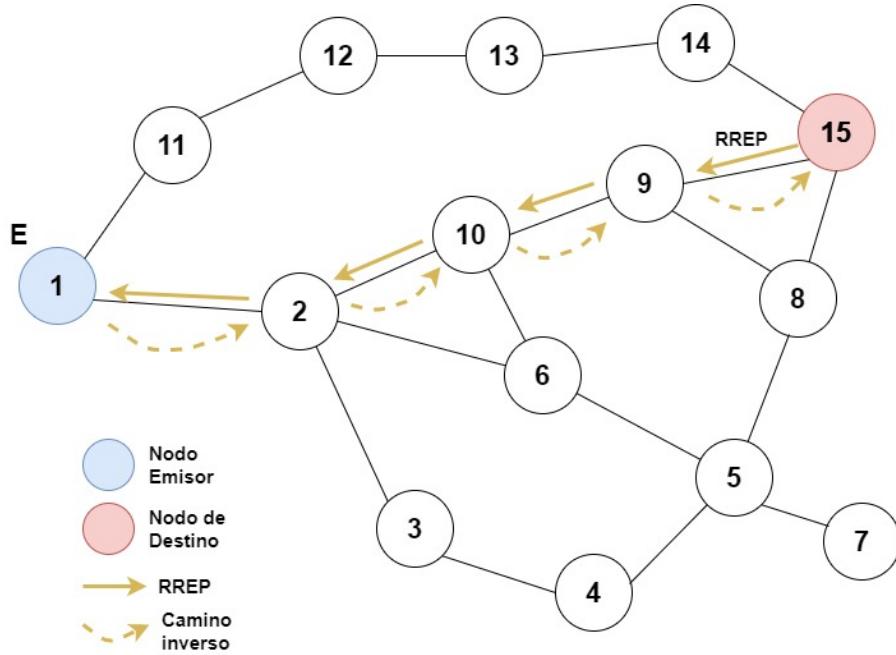


Figura 2-9.: Ruta de un mensaje de respuesta AODV en una red. [Ilyas, 2003]

de acuerdo con la posible movilidad de todos sus nodos. En el caso de las redes mesh, se tiene una disposición más o menos fija de los nodos que actúan como enrutadores o Gateway, por lo que existen diferencias fundamentales entre ambas estructuras de red. La tabla 2-1 muestra las diferencias más importantes que pueden presentarse entre ambas perspectivas de red.

Las principales diferencias que podemos encontrar en las redes mesh respecto a las ad hoc se fundamentan en la limitación que plantea poseer una estructura fija para el enrutamiento, lo que deriva en diferencias en el desempeño de los protocolos usados para este fin [Zhang et al., 2007]. Cabe considerar también, que los canales planteados en estos escenarios generalmente son inalámbricos, sin embargo, esto no niega la posibilidad de estructurar enlaces entre nodos a través de otros medios.

2.3.2. Retos en el diseño de redes Mesh

Al igual que en las redes ad hoc, existen dificultades a la hora de realizar la implementación de redes mesh, desde la capa física hasta la capa de aplicación. Sin embargo, existen factores fundamentales que permiten analizar y encontrar mejoras a la hora de diseñar aplicaciones y protocolos para este tipo de redes.

Tabla 2-1.: Diferencias entre las redes Ad Hoc y Mesh [Zhang et al., 2007].

| Aspecto | Redes Móviles Ad Hoc | Redes Mesh |
|-----------------------------------|----------------------------------|---|
| Topología de la red | Altamente Dinámica | Relativamente estática |
| Movilidad de los nodos de relevo | Media o alta | Baja |
| Restricción de energía | Alta | Baja |
| Características de la aplicación | Temporal | Semipermanente o permanente |
| Requerimientos de infraestructura | Sin infraestructura | Infraestructura parcial o totalmente fija |
| Retransmisión de mensajes | Se realiza por los nodos móviles | Se realiza por nodos fijos |
| Desempeño del enrutamiento | Completamente distribuido | Completa o parcialmente distribuido con enrutamiento basado en tablas |
| Implementación | Fácil de implementar | Se requiere alguna planeación |
| Característica del tráfico | Tráfico típico de usuarios | Tráfico típico de usuario y tráfico de sensores |

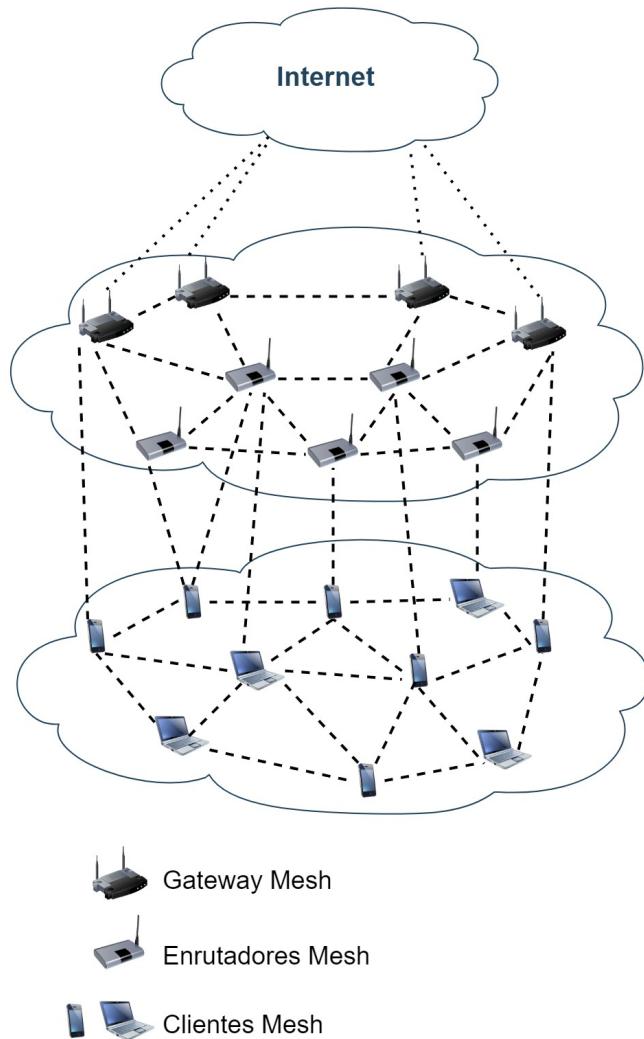


Figura 2-10.: Arquitectura típica de una red mesh.[Leung, 2008]

Uno de los campos de investigación y mejora en redes mesh es la que concierne a la transmisión a través de un canal determinado, generalmente el espectro electromagnético. En este sentido existen áreas activas de investigación y desarrollo en antenas, aplicaciones de radio cognitiva y el uso de múltiples bandas del espectro electromagnético. La interoperabilidad es otro factor esencial a la hora de implementar este tipo de redes, esto quiere decir que es necesario garantizar al usuario final que las redes implementadas bajo una tecnología o a través de un canal de comunicación sea compatible con otras redes, con el fin de asegurar que la red sea completamente escalable. Por último, pero no menos importante, es necesario garantizar que la transmisión de información y las aplicaciones generadas para estas redes sean completamente seguras y evitar negaciones del servicio, ataques o intrusiones al interior de la red. Esto plantea un reto considerable en relación con las redes tradicionales dada la falta de centralidad de las configuraciones mesh. [Leung, 2008]

Otro reto técnico que se generan a la hora de diseñar las redes mesh es la limitación en la tasa de transferencia efectiva de información que plantea tener múltiples saltos en la comunicación. Esta característica, fundamental en la medición de la calidad del servicio, se degrada rápidamente al aumentar la cantidad de saltos requeridos para alcanzar un nodo de destino. [Zhang et al., 2007]

En cuanto a la confiabilidad de la red, en general las redes ad hoc poseen mejores desempeños que las redes tradicionales, dada la creación de múltiples rutas para alcanzar un destino, lo que disminuye la probabilidad una falla en la comunicación. Sin embargo, es de considerar que muchas de las aplicaciones reales que utilizan esta topología, trabajan de forma inalámbrica sobre bandas del espectro electromagnético que son de uso abierto, esto implica generalmente una gran cantidad de mensajes sobre el mismo medio y, en consecuencia, aumenta la probabilidad de enfrentar fallas durante la recepción de la información. Por esta razón es importante el uso de varias bandas del espectro o incluso otras interfaces de comunicación que permitan mitigar este problema.

2.3.3. Redes Comunitarias de Internet

Las redes comunitarias consideran un escenario de aplicación de las redes mesh en el que se establecen enlaces de comunicación descentralizados, en el que los usuarios finales de la red son quienes participan en el diseño, planeación implementación y mantenimiento de manera autónoma y, generalmente, autogestionada. El uso de este tipo de redes ha sido variado dependiendo de los territorios de aplicación y de las necesidades propias de las comunidades, desde su uso como infraestructura interna de comunicación para transmitir información entre los usuarios de una comunidad, hasta su uso en el servicio de última milla para las empresas de telecomunicaciones.

Diversas organizaciones se han esforzado por crear este tipo de redes o por divulgar la información de su construcción. En Latinoamérica la Asociación Civil AlterMundo [AlterMundi,] con sede en Argentina realizó la puesta en funcionamiento de una red autogestionada de servicios de telecomunicaciones en la población San José de la Quintana, la cual se ha extendido a localidades cercanas. Este modelo se ha difundido a diferentes partes del mundo y ha permitido la conectividad de Internet a regiones alejadas de centros urbanos que, bien sea por su baja densidad poblacional o por su situación geográfica, no son considerados por las empresas de telecomunicaciones como opciones viables de inversión.

Este tipo de redes poseen ciertas características que, más que requerimientos, consisten en acuerdos tácitos entre usuarios y creadores que brindan posibilidades de expansión, interco-

nectividad y confianza en la transmisión. Algunas de estas características son las siguientes: [Guifi.net, 2017]

- Libre uso: puede ser utilizada por sus participantes para ofrecer y recibir cualquier tipo de servicio que no afecte su buen funcionamiento.
- Neutralidad: no inspecciona ni modifica los flujos de datos dentro de la red más allá de lo necesario para su operación.
- Libre interconexión: permite, de forma libre y gratuita, el flujo de datos con otras redes que respeten las mismas condiciones.
- Libre tránsito: provee a otras redes libres acceso a las redes con las que mantiene acuerdos voluntarios de libre interconexión. [Guifi.net, 2017]

Diversas experiencias al rededor del mundo con este tipo de redes han dejado registros fundamentales en su implementación. Se destacan los trabajos realizados en [Butler, 2013] [Johnson et al., 2007] que se han configurado, no sólo como una guía práctica para la construcción de redes comunitarias, sino también como una herramienta fundamental en la apropiación de este tipo de tecnologías por las comunidades en diversos lugares del planeta.

2.4. Modelo de Tráfico de Audio y Video

Dada la complejidad y el carácter estocástico del flujo de información en una red de comunicación, el obtener un modelo que permita recrear y predecir el tráfico de datos ha sido un problema fundamental en el diseño de redes de telecomunicaciones.

Uno de los modelos más simples, pero a la vez más poderosos en este intento de predicción del tráfico, ha sido el denominado on-off. [ANDRÉS PARRA LEÓN, ELKIN M. PIEDRAHITA, 2011] Los flujos de datos que se encuentran en una red tan amplia como internet varían dependiendo de la aplicación y de los requerimientos de los usuarios, sin embargo existen características en común de acuerdo a diversos estudios realizados con el fin de caracterizar estos flujos. Las dos principales son la alta variabilidad que significa que siempre pueden ocurrir cambios repentinos sobre el tráfico, por tal motivo es posible utilizar distribuciones de probabilidad como Pareto o Weibull con el fin de generar valores con varianzas y una variabilidad alta. La otra característica fundamental de este tipo de tráfico es la autosimilitud, característica que consiste en que una porción del tráfico generado posee rasgos similares a la muestra total del mismo. [Antoine Varet, 2014]

El modelo de tráfico on-off, consiste en modelar el flujo de información en dos momentos únicos: un estado activo o encendido y otro inactivo o apagado. Durante el tiempo activo o encendido el tráfico se asume como constante y los momentos de estado inactivo o apagado se consideran de flujo cero en el sistema.

Este modelo ha sido utilizado en múltiples estudios y ha demostrado poseer características muy similares a las del tráfico real. Uno de estos trabajos se presenta en [Campo-muñoz et al., 2017] donde se captura el tráfico real de una red con un servicio de audio y video por demanda, para luego obtener un modelo estadístico de la red usando el tráfico on-off y usándolo posteriormente en simulaciones basadas en el software de simulación de eventos discretos NS3.

3. Power Line Communication

Buena parte del trabajo de transmisión de información se lleva a cabo sobre el canal físico de comunicación. El espectro electromagnético es considerado el medio más común de transmisión física para la transmisión de redes ad hoc o redes mesh, sin embargo, esto no niega la posibilidad de establecer comunicaciones en otros canales. Cada uno de los canales usados para la transmisión posee características disímiles que deben ser abordadas para lograr comunicaciones eficientes a través de diferentes técnicas de modulación, multiplexación y acceso al medio físico.

Por esta razón el trabajo en ingeniería se ha orientado hacia la exploración de diferentes canales físicos que permitan mejorar la calidad de las transmisiones. Desde guías de onda de diferentes formas, hasta fibras ópticas que han permitido incrementar las velocidades y disminuir notablemente las latencias, diversas áreas de investigación han hecho avances notables que han permitido incluso aprovechar al máximo el espectro electromagnético y brindar mejores prestaciones en la movilidad de los dispositivos de comunicación.

En esta búsqueda constante de nuevos canales de comunicación, se han considerado algunos que no fueron diseñados originalmente para tal fin, pero que han mostrado un desempeño satisfactorio en la transmisión de información y han permitido extender, tanto en su alcance como en sus aplicaciones, las redes de comunicación. Uno de estos canales son las redes de transmisión eléctrica que ha sido explorado desde finales de los años 80 y principios de los 90 como canal de transmisión de información de mediciones de las redes eléctricas. [Lutz Lampe, 2016]

Con el tiempo se han incrementado las aplicaciones que han intentado usar este canal como medio de comunicación a medida que se incrementaban las frecuencias de transmisión y, como consecuencia, las velocidades que se podrían alcanzar, sin embargo, a principio de los 90s las frecuencias posibles sobre el canal se encontraban aún por debajo de los 3KHz, por lo que las transmisiones eran aún lentas.

Este escenario empieza a tener cambios drásticos a finales de los años 90 cuando pudieron desarrollarse aplicaciones en bandas que iban desde los 1.8 MHz y los 250 MHz. Estos nuevos desarrollos permitieron realizar transmisiones del orden de los cientos de Mpbs, lo que empieza a posibilitar la conexión de internet a través de este canal. A esta tecnología se

le ha denominado comúnmente como Power Line Communication o PLC. A pesar de estos avances en velocidades de transmisión las aplicaciones en la denominada “banda estrecha” que utiliza anchos de banda reducidos para transmitir información ha tenido un auge considerable a principios de este siglo, debido a las posibles aplicaciones que puede tener en redes eléctricas inteligentes y en IoT. [Lutz Lampe, 2016]

Dada la estructura topológica de las redes de distribución eléctrica existe una división general en redes de alta, media y baja tensión. Todas ellas posibilitan la transmisión de información, ya sea como troncales de información, como canal de última milla o incluso como extensor de las transmisiones en escenarios dentro del hogar, sin embargo, sólo esta última ha tenido un desarrollo comercial más amplio, debido a las dificultades técnicas y normativas que implica trabajar sobre redes eléctricas. Sin embargo diversos trabajos han planteado la posibilidad de usar este canal como medio de acceso a redes de comunicación para territorios que no han sido interconectadas por las redes comerciales tradicionales [Anatory et al., 2004] y también para aplicaciones modernas como las denominadas Smart Grids y el IoT. [Dib et al., 2018]

Es así como diversas entidades internacionales han enfocado su trabajo en el desarrollo de protocolos y estándares que permiten, no sólo la utilización de este canal de comunicación, sino también la creación de dispositivos funcionales y compatibles que aprovechan de manera eficiente los canales PLC.[Mengi et al., 2016]

3.1. Modelo de Canal en PLC

El gran reto que siempre ha presentado el uso del cableado eléctrico en la transmisión de información es el hecho de que es un canal que no ha sido diseñado para este fin, sino con la única intención de transmitir y distribuir energía eléctrica desde los puntos de generación hasta los usuarios finales. Debido a que esta distribución se realiza a bajas frecuencias (50 y 60 Hz), el uso del canal para transmitir información, incluso a bajas velocidades, implica la inyección de señales de alta frecuencia que no fueron consideradas en el diseño de las redes.

Es por esto por lo que el uso de canales PLC (Power Line Communication) como medio de comunicación, implica grandes desafíos que surgen como propios del medio y de su diseño original en la transmisión de energía. El primer gran reto al que se debe enfrentar es la obtención de un modelo apropiado del canal que permita sustentar un trabajo de desarrollo de técnicas y protocolos para efectuar una comunicación eficiente. Por esta razón, los primeros esfuerzos estructurados para usar este canal se encontraron orientados a la obtención de parámetros útiles para este modelo del canal, usando para tal fin mediciones realizadas en múltiples redes eléctricas. Esto sin embargo puede ser una tarea demasiado amplia si se tiene en cuenta que las redes eléctricas difieren entre sí entre países e incluso, en casos específicos como el de Colombia, entre regiones internas dado que influye la forma en la que las personas

construyen las redes internas de sus viviendas.

Por las razones ya mencionadas con anterioridad, los desarrollos posteriores fueron orientados en la creación de un modelo determinista que permitirá caracterizar el canal PLC en sus diferentes componentes de alta, media y baja tensión, sin importar las diferencias topológicas o de diseño de la red eléctrica.

Partiendo de los estudios iniciales sobre las redes PLC, en los que se realizaron múltiples mediciones en diferentes circunstancias y topologías, pudieron observarse y validarse ciertas características de las redes eléctricas que orientaron el trabajo del desarrollo de un modelo determinista. En este sentido pudo encontrarse, a través de la medición de parámetros temporales y frecuenciales, que los canales PLC son selectivos tanto en tiempo como en frecuencia.

En términos frecuenciales, esto quiere decir que, en algunas frecuencias concretas, el canal atenúa las señales en mayor medida que en otras y, además, que estas atenuaciones también cambian en el tiempo, considerando la naturaleza cíclica de la señal eléctrica y los constantes cambios de los dispositivos que se conectan en ellas. Esto implica que las transmisiones a través de estos canales deben considerar un cambio constante de bandas, evitando las frecuencias atenuadas en diferentes instantes de tiempo, dada la naturaleza aleatoria que implica la conexión y desconexión de dispositivos a la red.

En cuanto a los cambios temporales encontrados en la red eléctrica, se presenta una fuerte relación entre la característica periódica de la red, por lo que muchos de los cambios medidos de forma práctica poseían una periodicidad asociada a la de la red (50Hz o 60Hz dependiendo de la región) con un periodo igual a la mitad del de las señales de transmisión eléctrica.

La imagen **3-1** tomada de [Ferreira et al., 2010] muestra la medición de la ganancia de señales aplicadas sobre un canal PLC. Allí pueden observarse, tanto las variaciones temporales como frecuenciales mencionadas anteriormente. Si se analiza el eje frecuencial es fácil notar que existen regiones puntuales en las que la atenuación de las señales aplicadas es casi total. De igual forma, pasando a un análisis sobre el eje temporal, se evidencia que estas atenuaciones poseen una naturaleza cíclica propia de los cambios periódicos de la red eléctrica.

A partir de lo dicho anteriormente puede concluirse que existen dos grandes perspectivas en el modelamiento de los canales PLC. Uno de ellos es el análisis temporal en el que el canal de comunicación se considera un canal multi camino (multi-path), perspectiva muy común en el análisis de redes inalámbricas debido a la multiplicidad de caminos y rebotes que puede encontrar una señal antes de llegar al receptor. En el caso concreto de los canales PLC, este fenómeno se ocasiona debido a que, al aplicar señales de alta frecuencia sobre la red, los

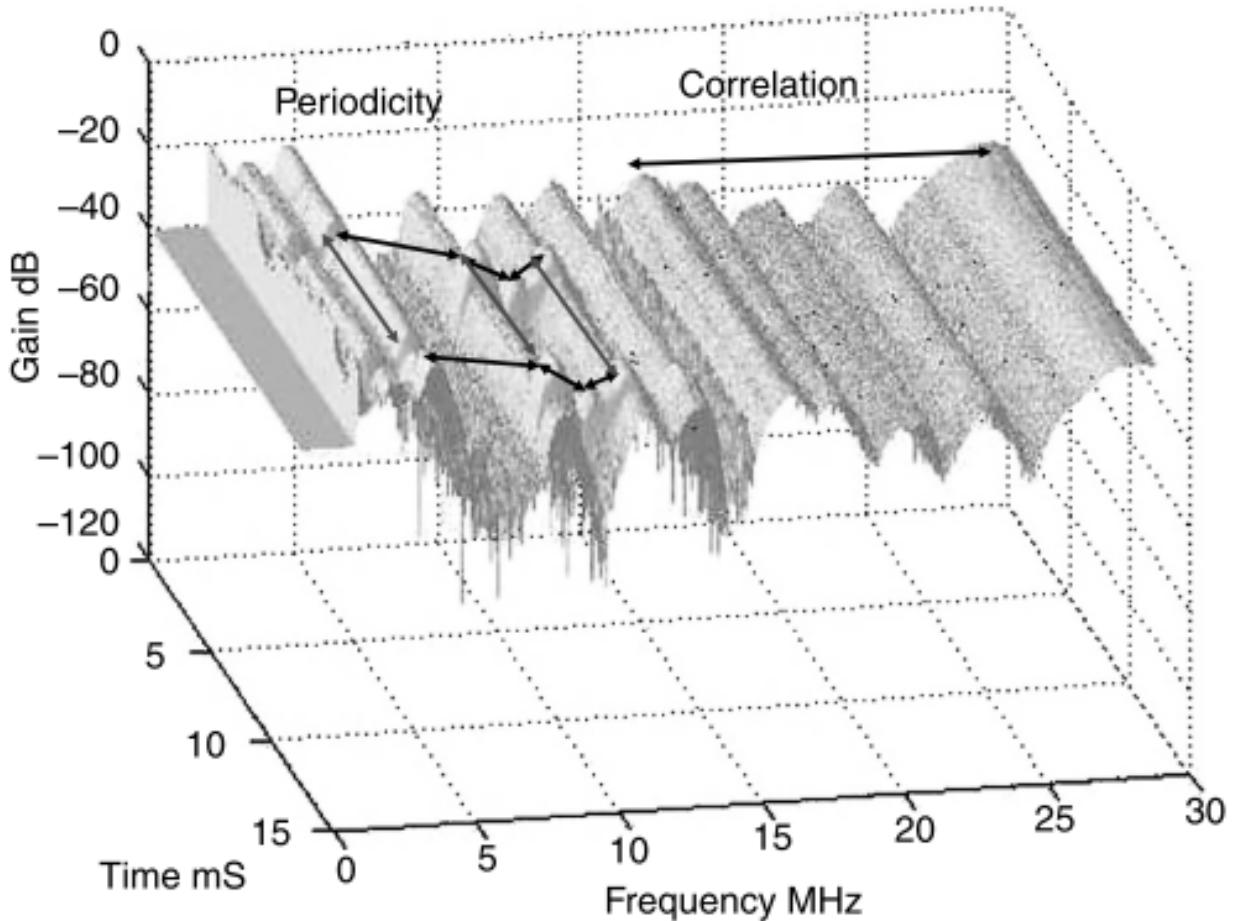


Figura 3-1.: Variaciones en tiempo y frecuencia medidas sobre el canal PLC [Ferreira et al., 2010]

nodos y derivaciones propios de la misma, ocasionan reflexiones de la señal, causando un fenómeno de múltiples caminos. [Ferreira et al., 2010]

$$H(f) = \sum_{i=1}^N g_i e^{-j2\pi f \tau_i} e^{-\alpha(f)\ell_i} \quad (3-1)$$

Desde el punto de visto temporal el canal puede ser modelado mediante una función de transferencia que se representa por la suma de las funciones que representan a cada uno de los caminos posibles que tomará la señal debido a las múltiples reflexiones. La ecuación 3-1 tomada de [Lutz Lampe, 2016] nos muestra la función de transferencia que se obtiene mediante este modelo matemático, donde g_i es un número complejo que depende de la topología del enlace, $\alpha(f)$ corresponde al coeficiente de atenuación que tiene en cuenta el efecto piel y las pérdidas dieléctricas, τ_i es el retardo asociado al camino i -ésimo, ℓ_i representa la longitud

de cada uno de estos caminos y N corresponde al número de caminos suficientemente largos para ser considerados. [Ferreira et al., 2010]

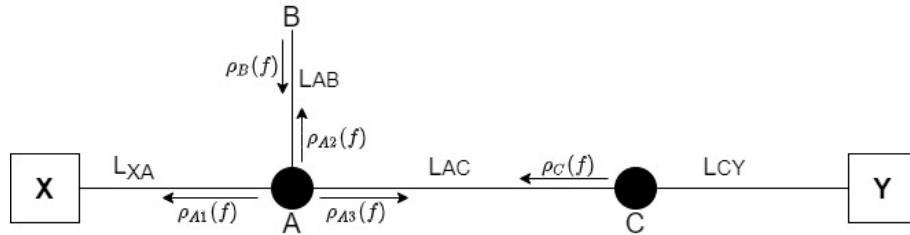


Figura 3-2.: Ejemplo para la determinación de caminos en una red PLC con una derivación A y una discontinuidad C. [Ferreira et al., 2010]

Tomemos como ejemplo el canal eléctrico mostrado en la figura 3-2, que presenta un emisor X y un receptor Y, así como una derivación en el punto A y una discontinuidad en el punto C. Tanto la derivación como la discontinuidad se describen mediante coeficientes de reflexión. Esta configuración ha sido planteada en múltiples publicaciones como [He et al., 2004] [Meng et al., 2004] [Zimmermann and Dostert, 2002] [Ferreira et al., 2010] [J. Anatomy and N. Theethayi, 2008].

Si se asume por un momento que no existe la discontinuidad en C, entonces la distancia entre la derivación y el receptor estará dada por la ruta denominada L_{AY} . De este modo la transmisión seguirá una línea directa desde X hasta A y luego hasta Y ($X \rightarrow A \rightarrow Y$) y se generarán un número infinito de rutas secundarias debido a las ondas que se reflejan entre los puntos A y B una cantidad de veces i . Es decir que cuando i es igual a 1 el camino sería $X \rightarrow A \rightarrow B \rightarrow A \rightarrow Y$, y cuando i es igual a dos se generarían más reflexiones de la forma $X \rightarrow A \rightarrow B \rightarrow A \rightarrow B \rightarrow A \rightarrow Y$, y de esta forma, en cada nueva consideración se incrementarían las señales reflejadas. Las ponderaciones g_i y las longitudes de los caminos ℓ_i se encuentran determinadas por las siguientes expresiones: [Ferreira et al., 2010]

$$\begin{aligned} \text{Camino directo } (i = 0) : & \left\{ \begin{array}{l} \ell_0 = L_{XA} + L_{AY} \\ g_0 = 1 + \rho_{A1} \end{array} \right. \\ \text{Caminos secundarios } (i > 0) : & \left\{ \begin{array}{l} \ell_i = L_{XA} + 2iL_{AB} + L_{AY} \\ g_i = (1 + \rho_{A1})(1 + \rho_{A2})(\rho_B \rho_{A2})^{i-1} \rho_B \end{array} \right. \end{aligned} \quad (3-2)$$

Si incluimos ahora la discontinuidad ubicada en el punto C, el escenario se complica aún más. Las características específicas de esta discontinuidad no necesitan ser descritas en su totalidad, dado que está completamente caracterizada por el coeficiente de reflexión ρ_B . por

lo que puede ser una derivación, una interconexión entre dos secciones de cable o cualquier otra discontinuidad posible en una red eléctrica.

Agregando esta nueva discontinuidad se presentan tres tipos de rutas que llegan al receptor Y. Las que llegan de una cantidad i de reflexiones entre A y B, las que llegan luego de una cantidad j de reflexiones entre A y C y, por último, las que llegan luego de i reflexiones entre A y B y luego j reflexiones entre A y C. Todos los caminos por los que viajan estas reflexiones, así como los coeficientes de reflexión pueden ser calculados de la siguiente manera. [Lutz Lampe, 2016]

$$\begin{aligned}
 \text{Camino directo } (i = 0) : & \left\{ \begin{array}{l} \ell_0 = L_{XA} + L_{AC} + L_{CY} \\ g_0 = (1 + \rho_{A1})(1 + \rho_C), \end{array} \right. \\
 \text{Caminos secundarios} & \quad \text{de tipo 1} (i > 0) : \left\{ \begin{array}{l} \ell_i = L_{XA} + 2iL_{AB} + L_{AC} + L_{CY} \\ g_i = (1 + \rho_{A1})(1 + \rho_{A2})(\rho_B \rho_{A2})^{i-1} \rho_B (1 + \rho_C) \end{array} \right. \\
 \text{Caminos secundarios} & \quad \text{de tipo 2} (j > 0) : \left\{ \begin{array}{l} \ell_i = L_{XA} + (2j + 1)L_{AC} + L_{CY} \\ g_i = (1 + \rho_{A1})(\rho_C \rho_{A3})^j (1 + \rho_C), \end{array} \right. \\
 \text{Caminos secundarios} & \quad \text{de tipo 3} (i, j > 0) : \left\{ \begin{array}{l} \ell_i = L_{XA} + 2iL_{AB} + (2j + 1)L_{AC} + L_{CY} \\ g_i = (1 + \rho_{A1})(\rho_B \rho_{A2})^{i-1} \rho_B (1 + \rho_{A2})(\rho_C \rho_{A3})^j \rho_B (1 + \rho_C). \end{array} \right. \tag{3-3}
 \end{aligned}$$

Esto muestra que al agregar tan sólo una discontinuidad, la complejidad del modelo aumenta considerablemente y al incluir en el modelo características propias de la red eléctrica como derivaciones o interrupciones, el procesamiento del problema, incluso con herramientas computacionales, puede hacerse tan complejo que dificultaría obtener resultados.

La otra perspectiva que ha sido utilizada para el modelamiento de canales PLC de baja tensión ha sido el análisis a través de la teoría de líneas de transmisión, la cual parte de un conocimiento detallado de la topología de la red y de las características del cableado con el cual fue construida dicha red. Este enfoque ha sido utilizado para el análisis de otros medios de comunicación como el cableado de cobre de dos hilos, propio de las comunicaciones DSL [Group et al., 2006], sin embargo su análisis puede ser extendido a canales PLC, aun cuando se trate de más de dos hilos.

Este análisis parte de considerar fragmentos infinitesimales de la línea de transmisión, tal como se muestra en la figura 3-3, el cual puede ser caracterizado de forma completa usando 4 parámetros distribuidos: Un valor resistivo “R” y un valor inductivo “L” por unidad de

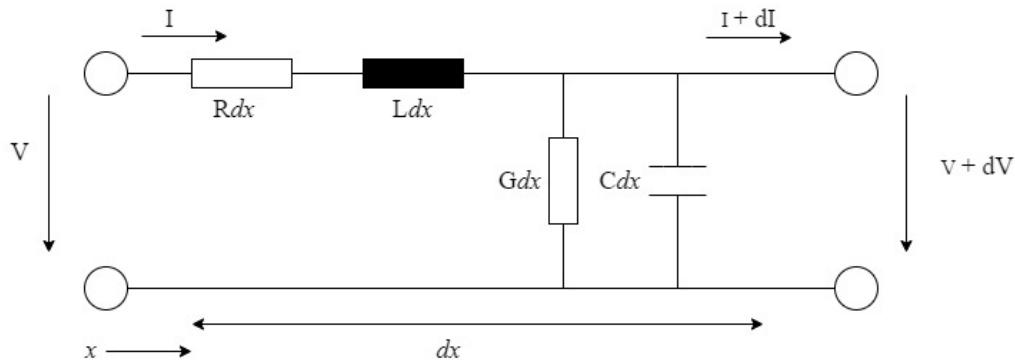


Figura 3-3.: Segmento de línea y los parámetros distribuidos que lo representan. [Lutz Lampe, 2016]

longitud que se muestran serie al circuito de entrada y un valor de conductancia “G” y un valor de capacitancia “C” que se ubican en paralelo a la entrada de tensión del segmento infinitesimal. A partir de un análisis circuital del segmento infinitesimal, podemos obtener las siguientes ecuaciones que corresponden a los componentes infinitesimales de corriente y tensión. [Group et al., 2006] [Lutz Lampe, 2016] [Orfanidis, 2008]

$$dV = -(R + jL\omega)Idx \quad dI = -(G + jC\omega)Vdx \quad (3-4)$$

$$\frac{dV}{dx} = -(R + jL\omega)I \quad \frac{dI}{dx} = -(G + jC\omega)V \quad (3-5)$$

Derivando con respecto a la longitud estas dos expresiones, podremos encontrar dos ecuaciones diferenciales de segundo orden que relational la tensión y la corriente, de la siguiente forma. [Group et al., 2006]

$$\begin{aligned} \frac{d^2V}{dx^2} &= -(R + jL\omega)\frac{dI}{dx} = (R + jL\omega)(G + jC\omega)V \\ \frac{d^2I}{dx^2} &= -(G + jC\omega)\frac{dV}{dx} = (R + jL\omega)(G + jC\omega)I. \end{aligned} \quad (3-6)$$

Definimos ahora el factor γ de la siguiente forma: [Group et al., 2006]

$$\gamma = \alpha + j\beta = \sqrt{(R + jL\omega)(G + jC\omega)} \quad (3-7)$$

Usando este parámetro, podremos reescribir las ecuaciones diferenciales de la siguiente manera: [Group et al., 2006]

$$\frac{d^2V}{dx^2} = \gamma^2 V \quad \frac{d^2I}{dx^2} = \gamma^2 I \quad (3-8)$$

Estas ecuaciones son más conocidas como las ecuaciones del telegrafista. Si pensamos ahora en una línea de longitud l que se encuentra terminada con una impedancia Z_L y alimentada por una fuente de tensión V_S con impedancia interna Z_S , la solución a estas ecuaciones a una distancia x de la fuente será la suma de dos tensiones y corrientes, viajando en direcciones opuestas, de la siguiente manera: [Group et al., 2006]

$$\begin{aligned} V(x) &= V_0^+ e^{-\gamma x} + V_0^- e^{\gamma x} \\ I(x) &= I_0^+ e^{-\gamma x} + I_0^- e^{\gamma x} \end{aligned} \quad (3-9)$$

Las señales de tensión y corriente que viajan con atenuación dada por $e^{-\gamma x}$ corresponden a las ondas incidentes y las que poseen una atenuación $e^{\gamma x}$ corresponden a las ondas reflejadas. La constante γ se denomina la constante de propagación, su parte real es llamada constante de atenuación y su parte imaginaria la constante de fase. [Group et al., 2006]

La impedancia característica de la línea (Z_0) se define como la carga que al ser aplicada a la línea hace que la impedancia en cualquier punto de la misma sea exactamente igual a ella. Este parámetro es importante ya que permite, a través de su valor, definir el comportamiento de la línea. Esta impedancia se encuentra definida a partir de los parámetros distribuidos de la línea de la siguiente manera. [Group et al., 2006]

$$Z_0 = \sqrt{\frac{R + jL\omega}{G + jC\omega}} \quad (3-10)$$

De esta forma, mediante el uso de la impedancia característica, podríamos reescribir las ecuaciones 3-9, utilizando una notación matricial, obteniendo la siguiente ecuación. [Group et al., 2006]

$$\begin{bmatrix} V(x) \\ I(x) \end{bmatrix} = \begin{bmatrix} V_0^+ e^{-\gamma x} + V_0^- e^{\gamma x} \\ \frac{V_0^+}{Z_0} e^{-\gamma x} - \frac{V_0^-}{Z_0} e^{\gamma x} \end{bmatrix} \quad (3-11)$$

Evaluando ahora las condiciones de frontera de estas ecuaciones podemos evaluarlas en $x = l$ y en $x = 0$, obtendremos las dos siguientes ecuaciones: [Group et al., 2006]

$$V_0^+ = \frac{1}{2} (V(l) + Z_0 I(l)) e^{+\gamma l} \quad V_0^- = \frac{1}{2} (V(l) - Z_0 I(l)) e^{-\gamma l} \quad (3-12)$$

$$V(0) = V_0^+ + V_0^- \quad I(0) = \frac{V_0^+}{Z_0} - \frac{V_0^-}{Z_0} \quad (3-13)$$

Y si combinamos estas dos expresiones, podremos obtener una relación entre las corrientes y las tensiones a ambos lados de la línea. [Group et al., 2006]

$$\begin{bmatrix} V(0) \\ I(0) \end{bmatrix} = \begin{bmatrix} \cosh(\gamma l) & Z_0 \sinh(\gamma l) \\ \frac{1}{Z_0} \sinh(\gamma l) & \cosh(\gamma l) \end{bmatrix} \begin{bmatrix} V(l) \\ I(l) \end{bmatrix} \quad (3-14)$$

Esta expresión ha sido fundamental en la simulación de líneas de transmisión o la concatenación de estas mismas líneas, como bien pueden ser modelados los canales PLC. Igualmente permite, a través de la relación entre las dos ecuaciones, obtener una impedancia de entrada de la línea Z_i . [Group et al., 2006]

$$Z_i = \frac{V(0)}{I(0)} = Z_0 \frac{Z_L \cosh(\gamma l) + Z_0 \sinh(\gamma l)}{Z_L \sinh(\gamma l) + Z_0 \cosh(\gamma l)} = Z_0 \frac{Z_L + Z_0 \tanh(\gamma l)}{Z_0 + Z_L \tanh(\gamma l)} \quad (3-15)$$

En la expresión presentada en 3-14 se hace evidente que una línea de transmisión uniforme en un segmento puede modelarse como una red de dos puertos, donde se aplica por un lado

una tensión V_1 y una corriente I_1 y se obtiene por el otro una tensión y una corriente V_2 e I_2 . A este modelo de dos puertos se le denomina ABCD, por la forma matricial en la que puede expresarse de la siguiente manera. [Group et al., 2006]

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} \quad (3-16)$$

Donde

$$\begin{aligned} A &= \left. \frac{V_1}{V_2} \right|_{I_2=0} = \cosh(\gamma l) & B &= \left. \frac{V_1}{I_2} \right|_{V_2=0} = Z_0 \sinh(\gamma l) \\ C &= \left. \frac{I_1}{V_2} \right|_{I_2=0} = \frac{1}{Z_0} \sinh(\gamma l) & D &= \left. \frac{I_1}{I_2} \right|_{V_2=0} = \cosh(\gamma l) \end{aligned} \quad (3-17)$$

Los parámetros A, B, C y D, no dependen de la impedancia de entrada o de salida de la línea, por lo que este modelo es bastante útil a la hora de modelar y simular líneas de transmisión que se encuentran concatenadas una tras otra, tal como se muestra en la figura 3-4.

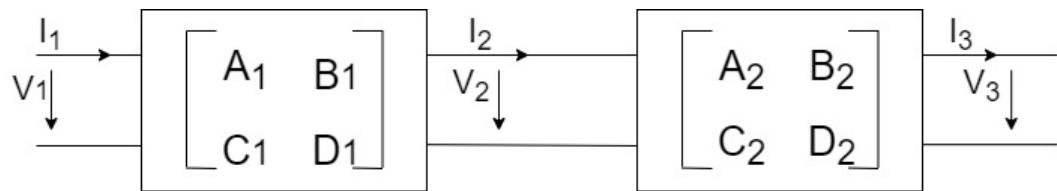


Figura 3-4.: Modelo de dos líneas concatenadas.[Group et al., 2006]

De igual forma, este análisis puede extenderse a una serie de n segmentos de líneas, cada una caracterizada por su propia matriz ABCD, obteniéndose lo siguiente. [Group et al., 2006]

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \prod_{i=1}^n \begin{bmatrix} A_i & B_i \\ C_i & D_i \end{bmatrix} \begin{bmatrix} V_{n+1} \\ I_{n+1} \end{bmatrix} \quad (3-18)$$

3.2. Simulación de canales PLC

Dadas estas formas de determinar un modelo exacto de los canales PLC, el trabajo ahora se enfoca en sintetizar computacionalmente estos modelos para simular y predecir la forma en la que el canal puede transmitir la información. Dentro de estos trabajos, que han permitido trasladar los modelos matemáticos a herramientas de software, permitiendo predecir el comportamiento de canal PLC, se encuentra el desarrollado por [Fariba Aalamifar, 2013], en el que se construyó un módulo para simulación de redes de comunicación en el software de eventos discretos NS3. [Fariba Aalamifar,]

Este software se fundamenta en la teoría de líneas de transmisión mencionada con anterioridad para modelar los canales PLC y calcular su función de transferencia. Igualmente, la herramienta de software desarrollada contiene funcionalidades que permiten agregar al modelo diferentes tipos de ruido e impedancias asociadas a cada uno de los nodos de la red, ayudando también a la interconexión con las funcionalidades y abstracciones propias del simulador NS3.

El módulo puede dividirse en cuatro partes fundamentales. El primer módulo denominado “Elementos de Red y Malla” (Grid and Network Elements), en el que el usuario puede crear diferentes topologías de red y establecer enlaces entre los nodos. Esta abstracción del módulo requiere la definición de un rango de frecuencias y una resolución para luego ejecutar los cálculos de la simulación.

Cada uno de los nodos generados en la red PLC puede ser visto como un vértice de un grafo y puede, a su vez, desempeñar múltiples funciones. El módulo permite agregar una impedancia al nodo PLC, esta impedancia puede ser constante (modelada como un número complejo que puede variar durante la simulación mediante la programación de eventos), selectiva en frecuencia (modelada en un vector de números complejos en el que cada posición corresponde a un valor de frecuencia específico o tres parámetros de un circuito resonante), selectiva en tiempo (modelada por un vector de números complejos en el que cada posición corresponde a un instante específico del periodo principal) y, por último, selectiva en tiempo y en frecuencia que combina las dos propiedades de los modos anteriores.

Los nodos también pueden ser usados como componentes activos de la comunicación, siendo asignados como transmisores o receptores. En este sentido, el nodo tendrá una interfaz que permitirá el uso de protocolos como el TCP/IP.

Con el fin de crear un escenario de simulación más realista, el software también permite que los nodos se utilicen como fuente de ruido. Se implementan varias funciones para modelar el ruido blanco, el ruido de color, el ruido impulsivo establecido por el usuario y el ruido

impulsivo de tipo aleatorio.

Finalmente, el software permite que los nodos se utilicen como uniones donde múltiples vértices del grafo convergen o bifurcan.

Tal como se aprecia en la figura 3-5, otra funcionalidad de la abstracción del módulo “Grid and Network” es la de creación de aristas, las cuales se entienden como los enlaces entre los nodos. Este enlace es modelado por el software a través de una red de dos puertos caracterizada por una matriz ABCD, los elementos pueden ser fijos, selectivos en tiempo o selectivos en frecuencia. El módulo incluye tres modelos diferentes de cables utilizados en instalaciones eléctricas: los cables de cuatro secciones NAYY 150SE y NAYY 50SE y el cable de tres secciones AL3X95XLPE.

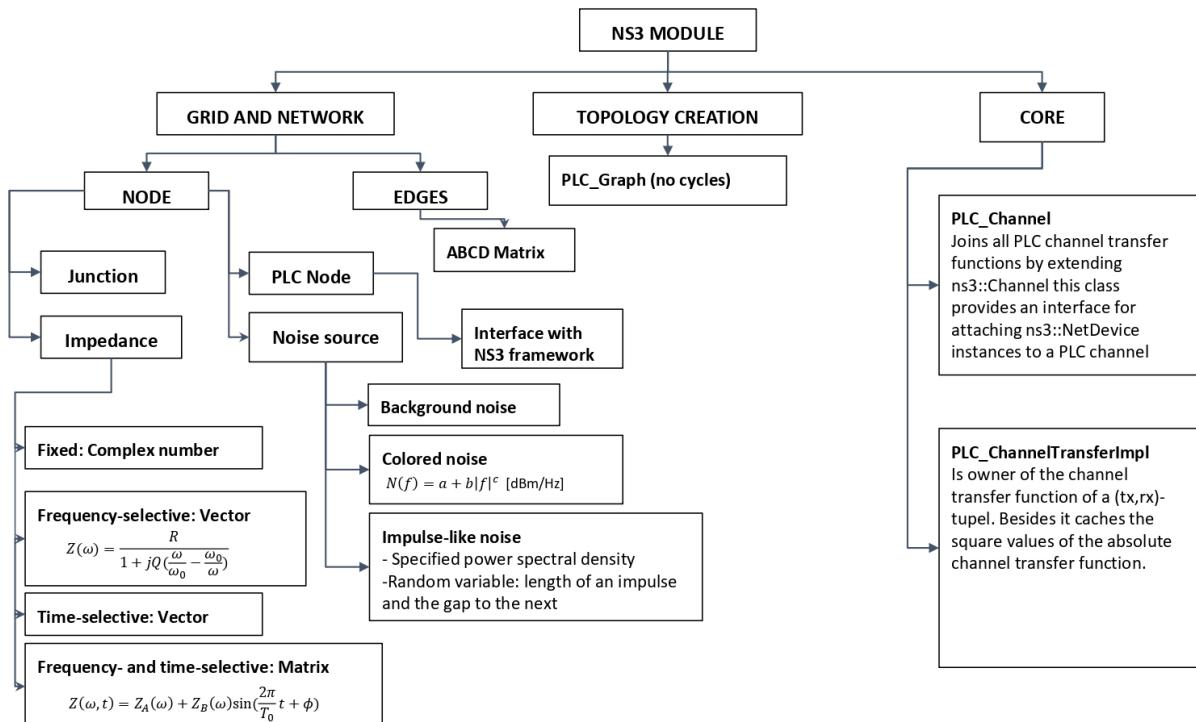


Figura 3-5.: Descripción gráfica del módulo PLC para NS3.[Zarate-Ceballos, 2021]

La siguiente abstracción del módulo es la denominada **Creación de topología** (Topology Creation). En esta abstracción, las topologías están formadas por nodos y los enlaces entre ellos. Para la creación de topologías el módulo permite un número arbitrario de nodos y enlaces, sin embargo, la única limitación es que no posean ningún ciclo cerrado, lo cual se adecua a la realidad de las redes eléctricas tradicionales, que nunca poseen este tipo de topologías. El software permite también el cálculo de la función de transferencia entre cualquier par de

nodos de PLC en la topología, así como la relación señal-ruido y la densidad espectral de potencia.

Finalmente encontramos el módulo de núcleo o **core**. Este módulo se divide en dos partes principales. La primera es la clase de canal PLC o **PLC_Channel** que permite vincular todas las funciones de transferencia de los canales del PLC y también extiende la clase **Channel** de NS3, permitiendo agregar elementos de la clase **NetDevice** a los nodos del PLC. La segunda es la clase **PLC_ChannelTransferImpl** que calcula el canal de transmisión utilizando la teoría de la línea de transmisión.[Zarate-Ceballos, 2021] [Fariba Aalamifar, 2013]

4. Simulación de redes PLC e inalámbricas utilizando NS3

Hasta el momento se han planteado los fundamentos teóricos, que permiten realizar el análisis de escenarios experimentales concretos para verificar la eficiencia de los canales de comunicación PLC, permitiendo posibles aplicaciones futuras en la implementación de redes descentralizadas y, específicamente, redes comunitarias de internet.

Por tal motivo se plantean a continuación escenarios de simulación implementados y ejecutados en el software de simulación de eventos discretos NS3, en las cuales se hace uso del módulo [Fariba Alamifar,] desarrollado en [Fariba Alamifar, 2013]. Tal como fue descrito con anterioridad, este módulo presenta dentro de sus características la implementación de cableados usados en algunos países para la construcción de redes eléctricas internas en viviendas y edificaciones.

Para tal fin se plantea un escenario concreto en el interior de una vivienda unifamiliar sobre el cual se montarán dispositivos de comunicación sobre la red eléctrica que permitirán simular la transmisión de tráfico de datos de video con el fin de verificar su utilidad en este tipo de redes.

Con el fin de realizar una ponderación de la calidad de servicio se evaluarán en estas simulaciones dos parámetros fundamentales: El throughput y el goodput. El primero hace referencia a la tasa exitosa de transmisión de mensajes en unidad de tiempo, es decir la cantidad de paquetes netos que son recibidos de manera exitosa por el receptor de la comunicación, el segundo responde a la tasa de transmisión exitosa, pero en los paquetes útiles, es decir que en este parámetro se descartan todos los paquetes o bits correspondientes a encabezados y protocolos de comunicación.

4.1. Escenarios de simulación

El escenario principal de simulación se desarrolla en el interior de una vivienda unifamiliar sobre el cual se plantea un cableado eléctrico que suministra energía a todos los espacios habitacionales. En la imagen 4-1 puede observarse el bosquejo del plano arquitectónico de

dicha vivienda, así como el cableado que allí se encuentra, incluyendo todos los puntos de conexión e iluminación, al igual que los interruptores que la componen. Pueden distinguirse dos secciones dentro del plano, una de ellas es el circuito principal, que parte de la caja de distribución de la casa, donde se encuentran los seccionadores y las protecciones eléctricas, y se distribuye por las zonas centrales de la vivienda. De este circuito principal se derivan cableados o circuitos secundarios que permiten alcanzar todos los rincones de la vivienda para lograr la distribución de la energía eléctrica.

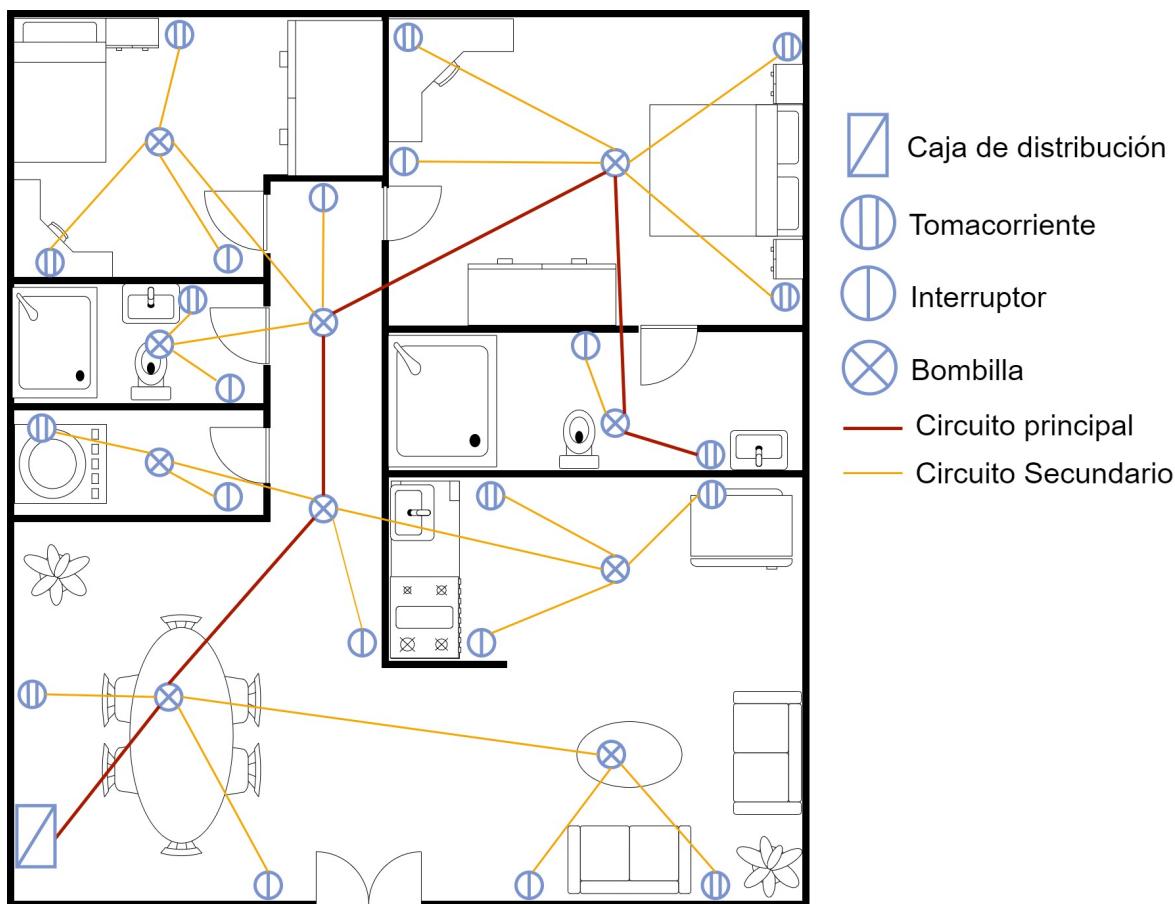


Figura 4-1.: Plano eléctrico de vivienda unifamiliar.[Zarate-Ceballos, 2021]

La figura 4-2 muestra el mismo escenario, pero extendiendo el plano y ubicándolo en dos dimensiones, por lo que todos los nodos y derivaciones se encuentran con las distancias originales que plantearía su disposición tridimensional, esto con el fin de plantear escenarios de simulación más sencillos en su implementación, pero que respeten las dimensiones originales.

Dadas las características del cableado eléctrico de la casa, existen derivaciones sobre las que no se conectan dispositivos de ninguna índole en ciertos momentos, por lo que el escenario

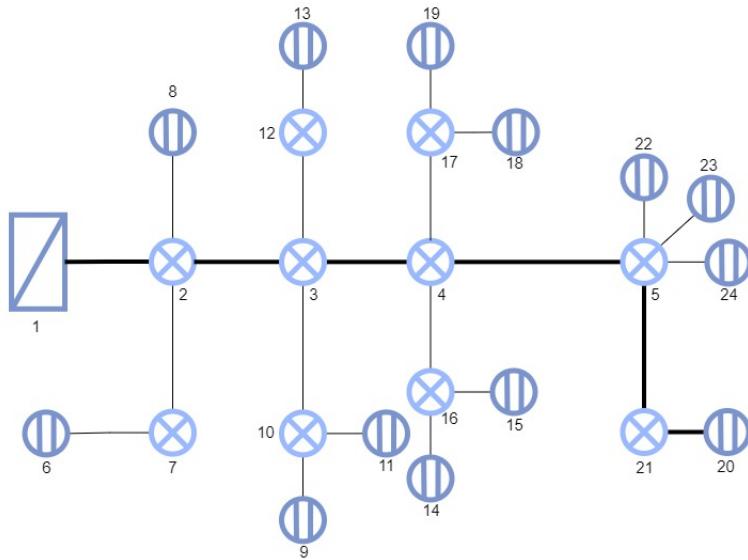


Figura 4-2.: Diagrama de nodos eléctricos de acuerdo a plano.[Zarate-Ceballos, 2021]

de simulación se verá reducido al circuito principal que es el de mayor extensión, esto con el fin de comprobar el efecto de la mayor longitud posible sobre la comunicación. En la imagen **4-3** puede apreciarse dicho canal, que será usado como escenario básico para estructurar las simulaciones.

A partir de este escenario básico se estructurarán varias simulaciones, primero sobre un canal netamente PLC que será usado para la transmisión de un flujo de datos entre dos nodos. Para este fin se utilizará un tráfico on-off, cuyo modelo estadístico en los tiempos de encendido y apagado fue obtenido en [Campo-muñoz et al., 2017] a partir la generación de tráfico de video bajo demanda. En este trabajo se usaron tres escenarios reales con videos de diferentes duraciones y se realizó una adquisición de tráfico mediante la herramienta Wireshark, lo que permitió hacer una análisis comparativo y, en últimas, determinar una función de distribución de probabilidad para este tipo de tráfico.

4.2. Escenario PLC

Este escenario utilizará enlaces de comunicación únicamente a través del cableado eléctrico, esto con el fin de verificar este canal como escenario viable de comunicación al interior de la vivienda. Igualmente, dado que los escenarios de redes comunitarias se plantean como redes ad-hoc, específicamente como redes mesh, es indispensable la implementación de protocolos de enrutamiento para su funcionamiento. Por tal motivo estos escenarios se enmarcarán en la prueba de diferentes protocolos de enrutamiento, tanto proactivos como reactivos, usados en

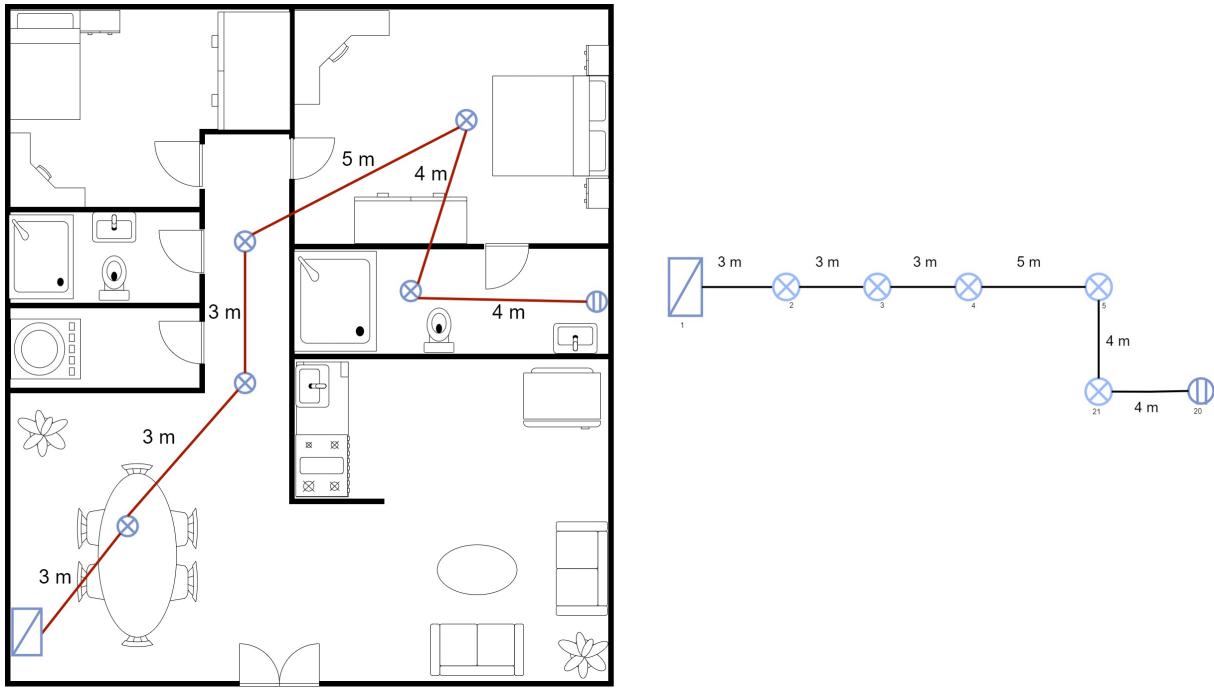


Figura 4-3.: Diagrama de nodos que componen el backbone de comunicación.

las redes ad-hoc y, posterior a esto, utilizar un escenario estándar con una topología idéntica, pero usando enlaces WiFi, con el fin de compararlos y determinar si existen diferencias en la utilización de este canal como backbone en una red comunitaria.

La figura 4-4 muestra el escenario concreto sobre el que se trabajará la simulación. Los nodos denotados con los números 0 y 20 actuarán como emisor y receptor, respectivamente. El cable que une todos los enlaces eléctricos es el cable AL3X95XLPE modelado a través de la matriz ABCD y que se encuentra incluido en la librería realizada en [Fariba Aalamifar, 2013] [Fariba Aalamifar,].

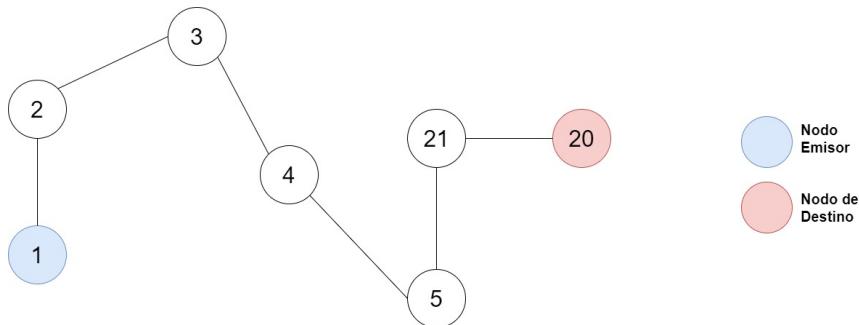


Figura 4-4.: Diagrama de enlaces para escenario PLC.

Con el fin de evaluar la calidad del enlace generado, se creará un tráfico on/off de acuerdo a los parámetros de audio y video encontrados en [Campo-muñoz et al., 2017], el cual se transmitirá entre los nodos 0 y 20, usando un canal PLC compuesto por un total de 8 nodos. La definición de estos parámetros puede observarse en la figura 4-5. Los nodos que no están siendo utilizados implícitamente para la comunicación estarán asociados a una impedancia constante de 50 Ohms, valor que se mantendrá constante en todos los escenarios planteados. La simulación será probada con dos protocolos de enrutamiento proactivos (DSDV y OLSR) y dos protocolos reactivos (AODV y DSR), en condiciones idénticas. Se usará una tasa de transmisión relativamente baja de 512 kb/s que no llegue a la saturación del canal con el fin de enfocar los esfuerzos de simulación en variar la potencia de la transmisión. Esta potencia se variará a través de la densidad espectral de potencia, es decir la potencia que se usa sobre cada una de las frecuencias de transmisión, por lo que se realizará una variación desde $1 \times 10^{-15} W$ hasta $5 \times 10^{-3} W$ en la que se obtendrán como resultado parámetros de throughput y goodput.

```
OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address ());
onoff1.SetAttribute("OffTime", StringValue ("ns3::LogNormalRandomVariable[Mu=0.4026,Sigma=0.0352]"));
onoff1.SetAttribute("OnTime", StringValue ("ns3::WeibullRandomVariable[Shape=10.2063,Scale=57480.9]"));
onoff1.SetAttribute("MaxBytes", UintegerValue(10989173));
```

Figura 4-5.: Definición de tráfico on-off en el software NS3.

Por último, se realizará una topología idéntica a la planteada en el escenario PLC, con nodos totalmente fijos que se encontrarán ubicados a las mismas distancias entre sí, pero que se enlazarán mediante el protocolo WiFi 802.11 en modo ad-hoc en el que se utiliza DS-CDMA (acceso múltiple por división de código en secuencia directa) como método de codificación. Este escenario final permitirá realizar una comparación entre el desempeño de la red PLC y una red típica ad-hoc en condiciones similares y evaluar su desempeño para la construcción de redes mesh en aplicaciones de redes comunitarias. La topología con los enlaces WiFi se muestra en la figura 4-6. Los códigos completos de la simulación de este escenario se encuentran consignados en los Anexos A y B del presente documento.

4.3. Escenario Mixto PLC-WiFi

Se plantea también la realización de tres escenarios de simulación que contendrán nodos con interfaz mixta PLC e inalámbrica, sobre los cuales se probarán únicamente el parámetro de calidad de servicio de Throughput. Esto debido a que se implementarán protocolos distintos para los enlaces, debido a que son canales de comunicación distintos, por lo que es conveniente conocer la tasa neta de transmisión exitosa de paquetes en todo el enlace.

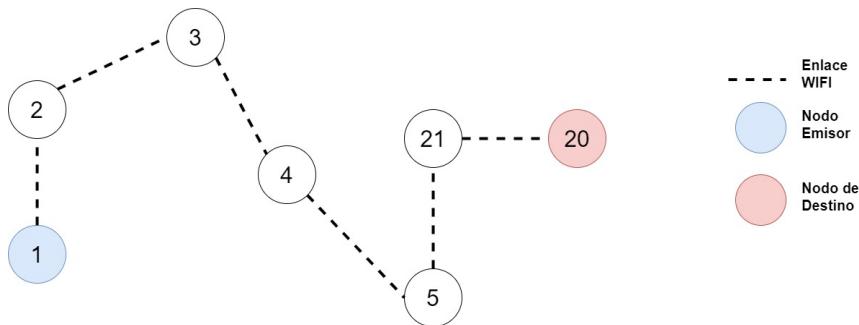


Figura 4-6.: Diagrama de enlaces para escenario WiFi.

En estos tres escenarios se utilizará el mismo backbone de comunicación sobre la red PLC, agregando en cada uno nodos WiFi y, por último, criterios de movilidad en los nodos, lo que permitirá conocer de manera más realista las consecuencias que puede tener sobre la red mesh los enlaces mixtos. De manera idéntica se planteará como variable de la simulación la densidad espectral de potencia de la transmisión sobre los canales PLC que estará variando, de manera idéntica al escenario experimental anterior, en un rango de $1 \times 10^{-15} W$ a $5 \times 10^{-3} W$.

4.3.1. Primer Escenario Mixto PLC-WiFi

Este escenario de simulación se compone del backbone PLC ya usado anteriormente, al cual se agrega un nuevo nodo que poseerá una interfaz WiFi que también se encontrará montada sobre el nodo 1 de la red PLC, de esta forma se creará un enlace en el que el nuevo nodo actuará como una estación y el nodo 1 de la red PLC como una Access Point a la red PLC. Al nodo wifi se le asigna un modelo estático de movilidad, es decir estará completamente inmóvil durante toda la simulación. La figura 4-7 muestra el escenario concreto de simulación. El código completo de esta simulación se encuentra consignado en el Anexo C del presente documento.

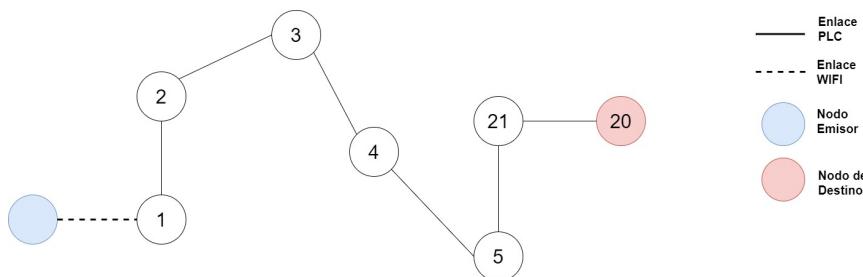


Figura 4-7.: Primer escenario de simulación mixto PLC-WiFi.

4.3.2. Segundo Escenario Mixto PLC-WiFi

Este escenario, mostrado en la figura 4-8, posee el backbone PLC de los escenarios anteriores y se enlazan a los nodos 1 y 20, dos nodos adicionales que poseen una interfaz WiFi. Estos nodos se encuentran estáticos con respecto a los nodos PLC en todo momento de la simulación y los resultados obtenidos de calidad de servicio se orientarán a la obtención del Throughput promedio de todos los enlaces. El código completo de esta simulación se encuentra consignado en el Anexo D.

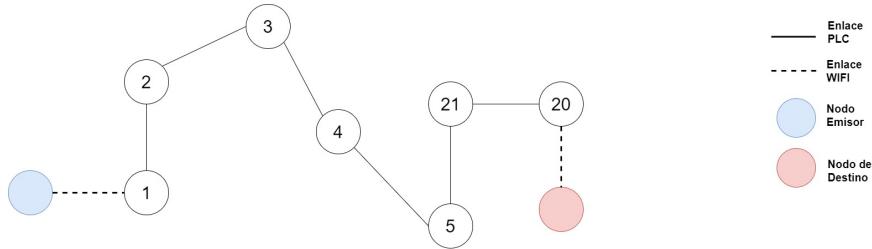


Figura 4-8.: Segundo escenario de simulación mixto PLC-WiFi.

4.3.3. Tercer Escenario Mixto PLC-WiFi

Por último, se presenta un escenario en el que se conectan dos nodos adicionales en los extremos del canal PLC, a diferencia del caso anterior, estos nodos poseen un modelo de movilidad aleatoria que se desplaza a 3 metros por segundo y se detiene durante 0.4 segundos para iniciar su desplazamiento en una ruta que se selecciona de forma aleatoria. Este desplazamiento se realiza sobre un espacio cuadrado de 10 m de lado. La figura 4-9 muestra la representación gráfica de este escenario de simulación.

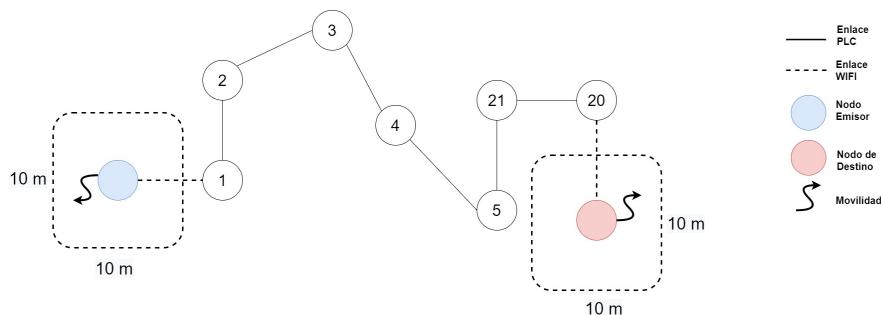


Figura 4-9.: Tercer escenario de simulación mixto PLC-WiFi.

5. Resultados y análisis

A continuación, se presentan los resultados obtenidos de los escenarios de simulación a través del software de simulación NS3, en los que se verifican parámetros de calidad de servicio en redes con canales PLC, inalámbricos y mixtos con un tráfico de datos on-off caracterizado para audio y video por demanda, tal como se especificó en el capítulo anterior. En todos los escenarios de simulación se realiza un barrido sobre la densidad de potencia espectral, asumiendo que se transmitirá con una potencia constante sobre todo el ancho de banda con el fin de verificar su efecto sobre la calidad de servicio en los enlaces.

Los resultados se encuentran dispuestos en la misma forma que los escenarios de simulación presentados con anterioridad, por lo que se iniciará con la configuración PLC, analizando los parámetros de calidad de servicio Throughput y Goodput de los enlaces y se compararán con la red inalámbrica WiFi de topología idéntica con el fin de verificar los cambios causados por el cambio del canal. Posteriormente se presentan los escenarios de simulación mixtos sobre los que se verifica la calidad servicio del enlace de comunicación. Luego se muestran los resultados obtenidos en las simulaciones mixtas sobre los tres escenarios planteados verificando el parámetro de Throughput que serán fundamentales para evaluar la viabilidad de este tipo de redes como complemento al canal inalámbrico en la implementación de redes Mesh.

5.1. Resultados y análisis escenario PLC

Se realiza la topología mostrada en la figura 4-4, en la que se utiliza un canal netamente PLC para transmitir información entre los nodos notados con la numeración 1 y 20. Se inicia con una potencia de $1 \times 10^{-15} W$ y se realiza un barrido logarítmico hasta llegar a los $5 \times 10^{-3} W$, registrándose en cada una de las corridas de simulación el valor del throughput y del goodput en todos los enlaces de la red. Se realiza este procedimiento usando cada uno de los protocolos de enrutamiento proactivos (DSDV y OLSR) y reactivos (AODV y DSR). Los resultados obtenidos se muestran en la tabla 5-1.

Las imágenes 5-1 y 5-2 muestran los resultados gráficos de los parámetros de calidad de servicio de throughput y goodput sobre los dos protocolos de enrutamiento proactivos DSDV y OLSR. Respecto al desempeño del protocolo DSDV puede apreciarse que el parámetro

Tabla 5-1.: Datos de calidad de servicio obtenidos en red PLC.

| Densidad espectral de potencia (Watts) | PLC | | | | | | | |
|--|-------------------|----------------|-------------------|----------------|-------------------|----------------|-------------------|----------------|
| | OLSR | | AODV | | DSDV | | DSR | |
| | Throughput (kbps) | Goodput (kbps) |
| 1.00E-15 | 24.18 | 0.00 | 50.08 | 8.64 | 4.61 | 8.70 | 0.33 | 0.00 |
| 5E-15 | 24.18 | 0.00 | 50.08 | 8.64 | 4.61 | 8.70 | 0.33 | 0.00 |
| 1E-14 | 24.18 | 0.00 | 50.08 | 8.64 | 4.61 | 8.70 | 0.33 | 0.00 |
| 5.00E-14 | 24.18 | 0.00 | 50.08 | 8.64 | 4.61 | 8.70 | 0.33 | 0.00 |
| 1E-13 | 25.32 | 0.00 | 50.08 | 8.64 | 5.69 | 8.70 | 0.33 | 0.00 |
| 5E-13 | 46.52 | 1.61 | 52.14 | 8.64 | 20.44 | 8.64 | 0.60 | 0.00 |
| 1E-12 | 48.18 | 8.69 | 49.94 | 8.67 | 26.53 | 8.69 | 3.84 | 0.65 |
| 5E-12 | 51.83 | 8.69 | 49.94 | 8.67 | 31.32 | 8.70 | 0.55 | 0.00 |
| 1E-11 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 5E-11 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 1E-10 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 5E-10 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 1E-09 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 5E-09 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.00000001 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.00000005 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.0000001 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.0000005 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.000001 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.000005 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.00001 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.00005 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.0001 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.0005 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.001 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.005 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |
| 0.005 | 49.87 | 8.70 | 49.94 | 8.67 | 31.31 | 8.70 | 4.44 | 5.24 |

throughput presenta un pico de caída sobre los $5 \times 10^{-13} W$ que puede deberse a un fenómeno resonante propio del modelo usado para simular el canal, además de esto se observa que el goodput aumenta a medida que se aumenta la potencia de transmisión, lo cual es coherente con un escenario realista. Sin embargo, el escenario del throughput plantea algunas diferencias, ya que decrece ligeramente al aumentar la potencia, lo que puede deberse a que la topología de la red nunca se modifica y los saltos para llegar al destino son idénticos, por lo que los datos necesarios para actualizar las tablas y mantener las rutas disminuyen ligeramente. Respecto al protocolo OLSR, dado que los datos de actualización de la tabla de enrutamiento son constantemente renovados y no se evalúa nunca la calidad del enlace, los mensajes se incrementan al aumentar la potencia y al alcanzarse nuevos nodos, por esta razón, tanto el throughput como el goodput se incrementan al incrementar la potencia de transmisión.

Por otro lado, el desempeño de los protocolos reactivos AODV y DSR, pueden observarse en las figuras 5-3 y 5-4, respectivamente. En el caso del protocolo AODV se observa también un ligero decaimiento en el throughput al aumentar la potencia de la transmisión, lo cual puede deberse a la forma en la que se realiza la construcción de las rutas, dado que la información de éstas no se almacena en los encabezados de los mensajes enviados, sino que se almacena localmente en cada nodo, lo que hace que al determinarse las rutas, la cantidad de información disminuya a menos que los caminos cambien, lo cual no ocurre en este escenario, ya que los nodos se encuentran fijos. En el caso del protocolo DSR, donde las rutas se almacenan en los encabezados de los paquetes enviados el resultado de la simulación es consecuente, dado que, al descubrir las rutas del enlace, se incrementa el tamaño de los paquetes hasta que todas son establecidas, momento en el que el tamaño es estabiliza.

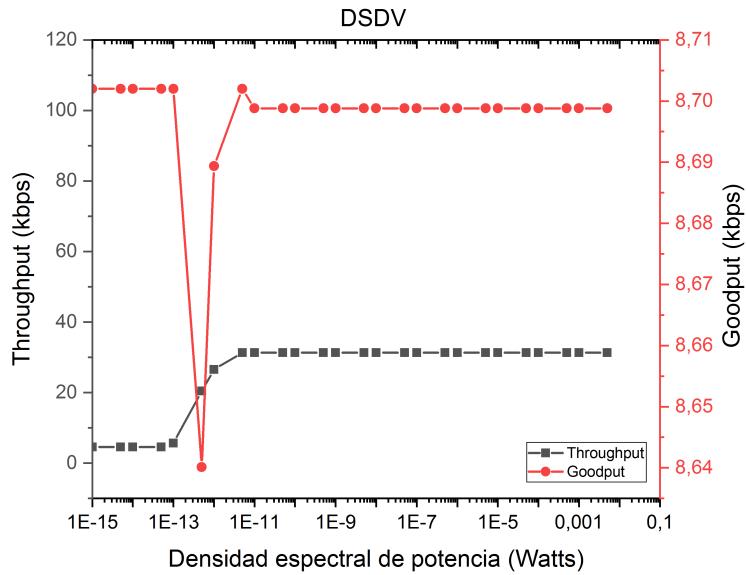


Figura 5-1.: Resultados de simulación para throughput y goodput en red PLC con protocolo de enrutamiento DSDV

Las imágenes 5-5 y 5-4, permiten observar el desempeño comparativo de todos los protocolos de enrutamiento en el canal PLC para los parámetros de Throughput y Goodput, respectivamente. Puede observarse que el desempeño de los protocolos OLSR y AODV presentan mejores prestaciones al mantener una tasa exitosa de transmisión neta, sin embargo, en la tasa de transmisión de paquetes útiles, al aumentar la potencia, las diferencias entre los protocolos OLSR, AODV y DSDV no son tan notorias. El escenario DSR es el único que presenta resultados drásticamente diferentes respecto a los demás, siendo claramente el de peor desempeño en este escenario.

La siguiente comparación se realiza respecto a un escenario de simulación donde los enlaces PLC son reemplazados por enlaces WiFi ad-hoc, manteniendo los protocolos de enrutamiento. En la misma forma y con el fin de mantener un escenario de comparación, se realiza una variación en la potencia de transmisión de los nodos WiFi, sin embargo, dada la cercanía de los enlaces, las mismas variaciones de potencia asignadas al escenario PLC hacen que los parámetros de calidad de servicio se mantengan estables. Sin embargo, se realiza la comparación gráfica de las respuestas para cada uno de los protocolos en los canales PLC e inalámbrico con el fin de evidenciar las posibles diferencias en su comportamiento al cambiar el canal. Las figuras 5-7, 5-8, 5-9 y 5-10 muestran los resultados de simulación comparativos entre el canal PLC e inalámbrico para los protocolos DSDV, OLSR, AODV y DSR, respectivamente.

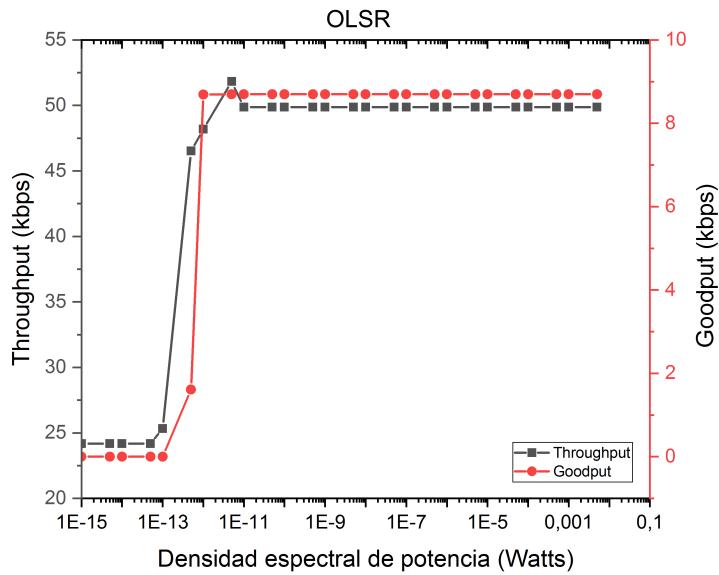


Figura 5-2.: Resultados de simulación para throughput y goodput en red PLC con protocolo de enrutamiento OLSR

Puede observarse que en la mayor parte de los escenarios planteados el desempeño, medido en la calidad del servicio a través del throughput y el goodput, es similar en ambos canales de comunicación e incluso, algunos de ellos, son mejores sobre el canal PLC. Aun cuando se tienen resultados en los que un parámetro puede ser mejor en el escenario WiFi, el otro demuestra tener un resultado mejor sobre el canal PLC, por lo que haciendo un balance del desempeño podría decirse que los resultados son similares en ambos medios de transmisión.

5.2. Resultados y análisis en escenario mixto PLC-WiFi

Planteados ya los escenarios con enlaces completamente PLC, se realizará el análisis de los resultados obtenidos en simulaciones para redes con canales mixtos PLC y WiFi, teniendo en cuenta que la aplicación directa sobre redes mesh y en específico de redes comunitarias sería la extensión de redes implementadas en canales inalámbricos.

Se presentan a continuación los resultados para los tres escenarios mixtos de simulación descritos con anterioridad, realizando, al igual que en los casos anteriores, variaciones sobre la densidad de espectral de potencia aplicada por cada nodo PLC para transmitir la información.

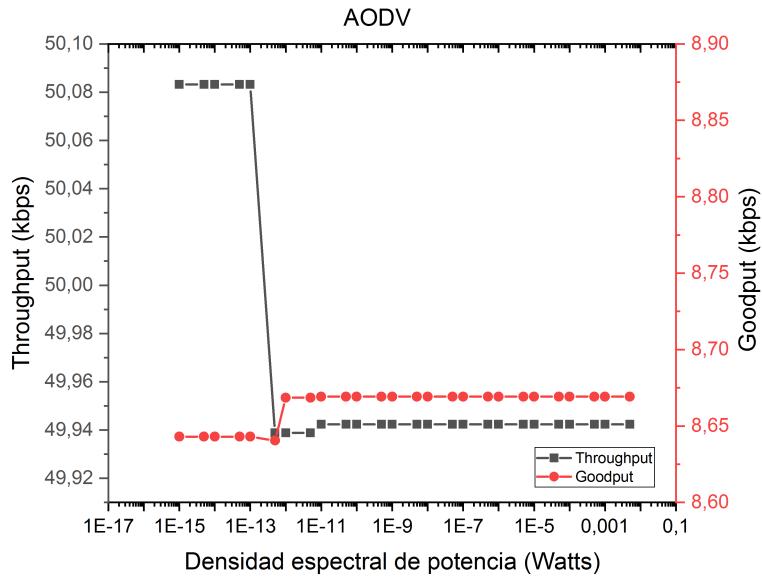


Figura 5-3.: Resultados de simulación para throughput y goodput en red PLC con protocolo de enrutamiento AODV

5.2.1. Resultados y análisis primer escenario mixto PLC-WiFi

En la tabla 5-2 se observan los resultados obtenidos para el throughput en el primer escenario de simulación mixto donde se utiliza un nodo adicional vinculado al primer nodo PLC mediante un enlace WiFi. Dadas las bajas prestaciones que se establecieron en el escenario anterior para el protocolo de enrutamiento DSR, éste será descartado de los próximos escenarios, por lo que se muestran únicamente los resultados para los protocolos DSDV, OLSR y AODV.

La imagen 5-11 muestra los resultados gráficos de esta simulación, permitiendo realizar una comparación más sencilla entre los protocolos de enrutamiento planteados. Puede observarse que el protocolo AODV se mantiene constante en la tasa exitosa de transferencia neta durante todo el barrido de potencia realizado, lo cual puede ser benéfico si se busca optimizar el manejo de potencia del dispositivo, sin embargo, dado que los canales PLC son usados esencialmente para transmitir energía, este no sería un problema latente en este tipo de dispositivos de comunicación. Por otro lado, se evidencia que al aumentar la potencia del enlace el escenario de simulación donde se utiliza el protocolo de enrutamiento OLSR resulta ser el que presenta mejores prestaciones en la calidad del servicio.

Tabla 5-2.: Datos de calidad de servicio obtenidos en red mixta PLC-WiFi en el primer escenario de simulación.

| Densidad espectral de potencia (Watts) | Throughput (kbps) | | |
|---|--------------------------|-------------|-------------|
| | OLSR | AODV | DSDV |
| 1,00E-15 | 30,50 | 54,47 | 3,83 |
| 5E-15 | 30,50 | 54,47 | 3,83 |
| 1E-14 | 30,50 | 54,47 | 3,83 |
| 5,00E-14 | 30,50 | 54,47 | 3,83 |
| 1E-13 | 31,99 | 54,45 | 5,88 |
| 5E-13 | 67,07 | 54,39 | 24,93 |
| 1E-12 | 58,04 | 54,41 | 30,85 |
| 5E-12 | 61,03 | 54,36 | 35,33 |
| 1E-11 | 57,73 | 54,36 | 35,33 |
| 5E-11 | 57,73 | 54,36 | 35,33 |
| 1E-10 | 57,73 | 54,36 | 35,33 |
| 5E-10 | 57,73 | 54,36 | 35,33 |
| 1E-09 | 57,73 | 54,36 | 35,33 |
| 5E-09 | 57,73 | 54,36 | 35,33 |
| 0,00000001 | 57,73 | 54,36 | 35,33 |
| 0,00000005 | 57,73 | 54,36 | 35,33 |
| 0,0000001 | 57,73 | 54,36 | 35,33 |
| 0,0000005 | 57,73 | 54,36 | 35,33 |
| 0,000001 | 57,73 | 54,36 | 35,33 |
| 0,000005 | 57,73 | 54,36 | 35,33 |
| 0,00001 | 57,73 | 54,36 | 35,33 |
| 0,00005 | 57,73 | 54,36 | 35,33 |
| 0,0001 | 57,73 | 54,36 | 35,33 |
| 0,0005 | 57,73 | 54,36 | 35,33 |
| 0,001 | 57,73 | 54,36 | 35,33 |
| 0,005 | 57,73 | 54,36 | 35,33 |

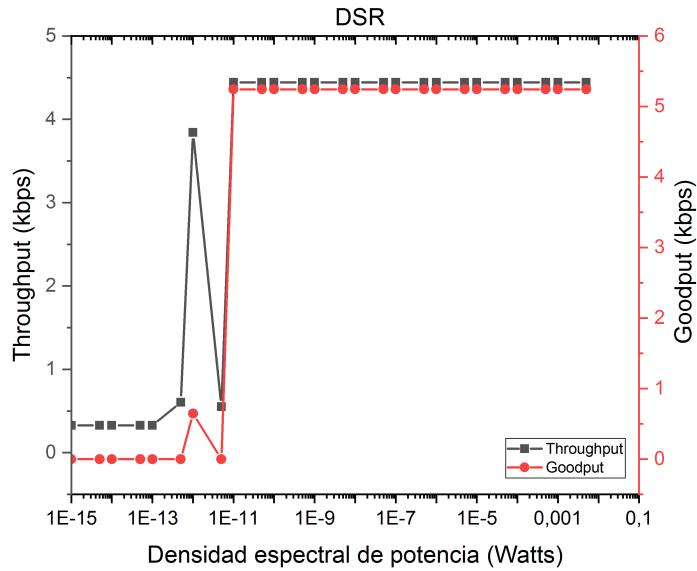


Figura 5-4.: Resultados de simulación para throughput y goodput en red PLC con protocolo de enrutamiento DSR

5.2.2. Resultados y análisis segundo escenario mixto PLC-WiFi

Se analizará ahora el segundo escenario de simulación mixto planteado, en el cual se agregará un nodo WiFi que actuará como receptor del tráfico de datos generado. En este escenario ambos nodos se encuentran fijos respecto al backbone PLC. En la tabla 5-3 pueden observarse los datos resultado de la simulación en este escenario. De igual forma, la figura 5-12 muestra el resultado gráfico de la simulación, allí puede observarse un resultado similar al escenario uno en el que el protocolo de enrutamiento OLSR presenta un desempeño superior a medida que la densidad espectral de potencia se aumenta.

5.2.3. Resultados y análisis tercer escenario mixto PLC-WiFi

Este escenario de simulación presenta una única modificación respecto al anterior, el parámetro de movilidad agregado a los nodos WiFi, sobre un espacio de 10 metros cuadrados en torno a los nodos PLC. La tabla 5-4 muestra los datos obtenidos de esta simulación. El resultado se encuentra graficado en la 5-13, sin embargo se evidencia que lo obtenido en la simulación no difiere del planteado en el segundo escenario de simulación.

Tabla 5-3.: Datos de calidad de servicio obtenidos en red mixta PLC-WiFi en el segundo escenario de simulación.

| Densidad espectral de potencia (Watts) | Throughput (kbps) | | |
|---|--------------------------|-------------|-------------|
| | OLSR | AODV | DSDV |
| 1,00E-15 | 35,28 | 59,18 | 4,39 |
| 5E-15 | 35,28 | 59,18 | 4,39 |
| 1E-14 | 35,28 | 59,18 | 4,39 |
| 5,00E-14 | 35,28 | 59,18 | 4,39 |
| 1E-13 | 36,61 | 59,14 | 6,18 |
| 5E-13 | 72,43 | 59,12 | 23,18 |
| 1E-12 | 71,25 | 59,11 | 32,65 |
| 5E-12 | 75,85 | 59,05 | 38,25 |
| 1E-11 | 72,50 | 59,05 | 38,25 |
| 5E-11 | 72,50 | 59,05 | 38,25 |
| 1E-10 | 72,50 | 59,05 | 38,25 |
| 5E-10 | 72,50 | 59,05 | 38,25 |
| 1E-09 | 72,50 | 59,05 | 38,25 |
| 5E-09 | 72,50 | 59,05 | 38,25 |
| 0,00000001 | 72,50 | 59,05 | 38,25 |
| 0,00000005 | 72,50 | 59,05 | 38,25 |
| 0,0000001 | 72,50 | 59,05 | 38,25 |
| 0,0000005 | 72,50 | 59,05 | 38,25 |
| 0,000001 | 72,50 | 59,05 | 38,25 |
| 0,000005 | 72,50 | 59,05 | 38,25 |
| 0,00001 | 72,50 | 59,05 | 38,25 |
| 0,00005 | 72,50 | 59,05 | 38,25 |
| 0,0001 | 72,50 | 59,05 | 38,25 |
| 0,0005 | 72,50 | 59,05 | 38,25 |
| 0,001 | 72,50 | 59,05 | 38,25 |
| 0,005 | 72,50 | 59,05 | 38,25 |

Tabla 5-4.: Datos de calidad de servicio obtenidos en red mixta PLC-WiFi en el tercer escenario de simulación.

| Densidad espectral de potencia (Watts) | Throughput (kbps) | | |
|---|--------------------------|-------------|-------------|
| | OLSR | AODV | DSDV |
| 1,00E-15 | 35,28 | 59,18 | 4,39 |
| 5E-15 | 35,28 | 59,18 | 4,39 |
| 1E-14 | 35,28 | 59,18 | 4,39 |
| 5,00E-14 | 35,28 | 59,18 | 4,39 |
| 1E-13 | 36,61 | 59,14 | 6,18 |
| 5E-13 | 72,43 | 59,12 | 23,18 |
| 1E-12 | 71,25 | 59,11 | 32,65 |
| 5E-12 | 75,85 | 59,05 | 38,25 |
| 1E-11 | 72,50 | 59,05 | 38,25 |
| 5E-11 | 72,50 | 59,05 | 38,25 |
| 1E-10 | 72,50 | 59,05 | 38,25 |
| 5E-10 | 72,50 | 59,05 | 38,25 |
| 1E-09 | 72,50 | 59,05 | 38,25 |
| 5E-09 | 72,50 | 59,05 | 38,25 |
| 0,00000001 | 72,50 | 59,05 | 38,25 |
| 0,00000005 | 72,50 | 59,05 | 38,25 |
| 0,0000001 | 72,50 | 59,05 | 38,25 |
| 0,0000005 | 72,50 | 59,05 | 38,25 |
| 0,000001 | 72,50 | 59,05 | 38,25 |
| 0,000005 | 72,50 | 59,05 | 38,25 |
| 0,00001 | 72,50 | 59,05 | 38,25 |
| 0,00005 | 72,50 | 59,05 | 38,25 |
| 0,0001 | 72,50 | 59,05 | 38,25 |
| 0,0005 | 72,50 | 59,05 | 38,25 |
| 0,001 | 72,50 | 59,05 | 38,25 |
| 0,005 | 72,50 | 59,05 | 38,25 |

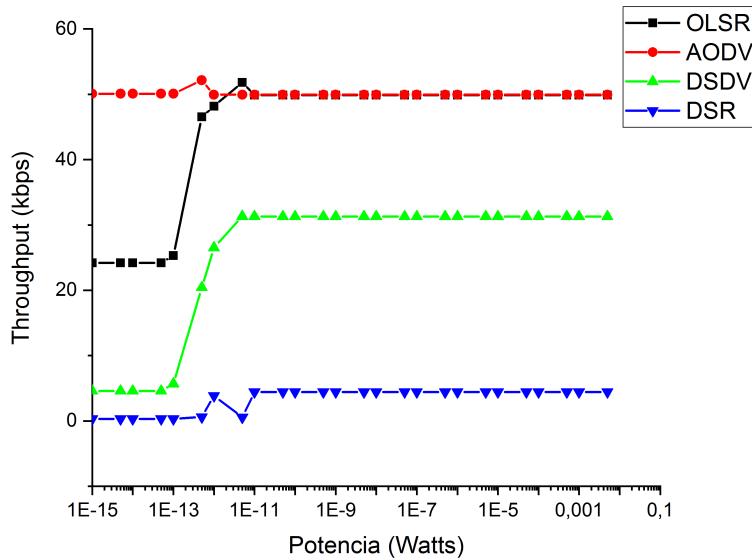


Figura 5-5.: Comparación de resultados de simulación para throughput en red PLC con protocolos de enrutamiento DSDV, OLSR, AODV y DSR

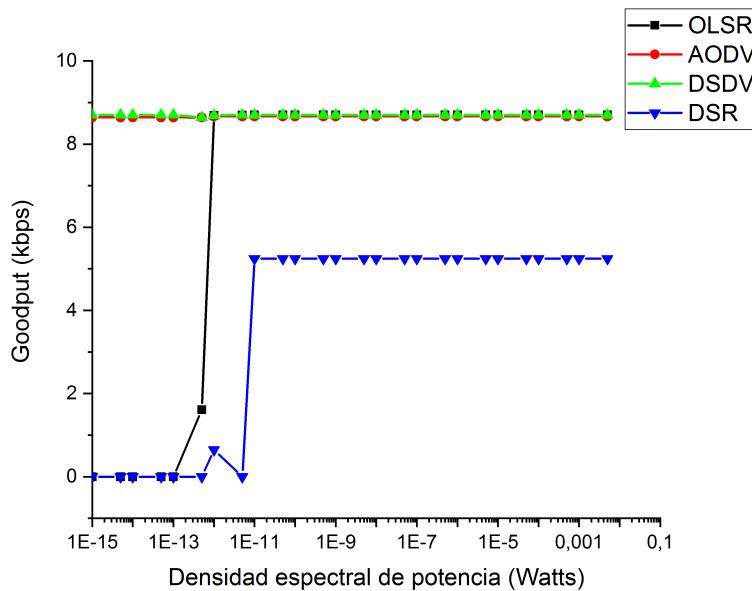


Figura 5-6.: Comparación de resultados de simulación para goodput en red PLC con protocolos de enrutamiento DSDV, OLSR, AODV y DSR

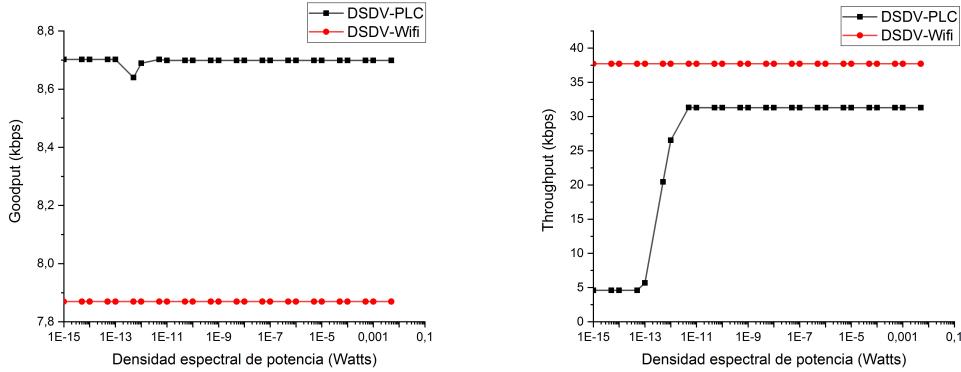


Figura 5-7.: Comparación de resultados de simulación para throughput y goodput en canales PLC y Wifi con protocolo de enrutamiento DSDV

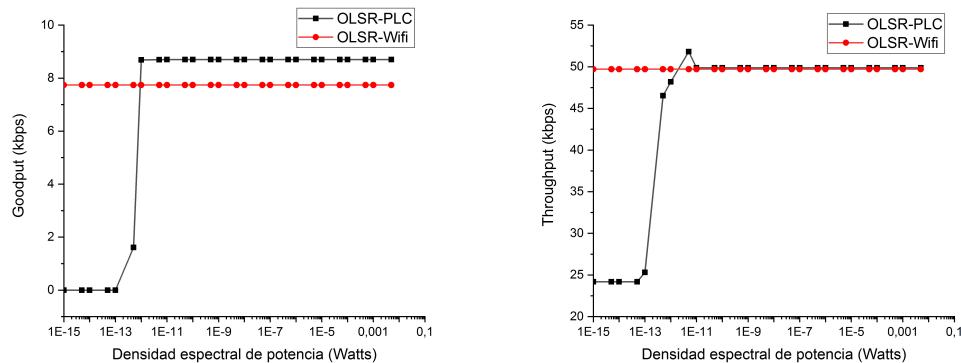


Figura 5-8.: Comparación de resultados de simulación para throughput y goodput en canales PLC y Wifi con protocolo de enrutamiento OLSR

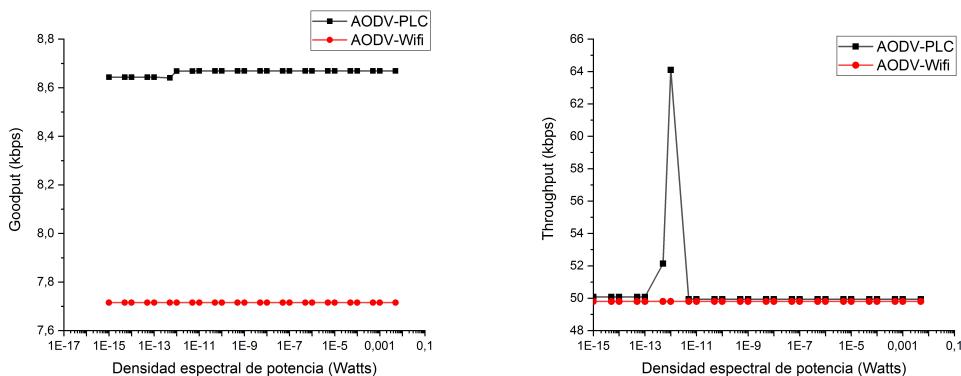


Figura 5-9.: Comparación de resultados de simulación para throughput y goodput en canales PLC y Wifi con protocolo de enrutamiento AODV

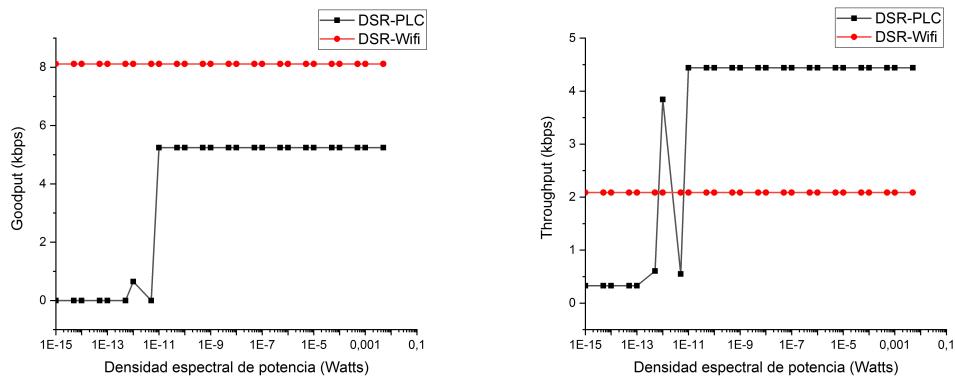


Figura 5-10.: Comparación de resultados de simulación para throughput y goodput en canales PLC y WiFi con protocolo de enrutamiento DSR

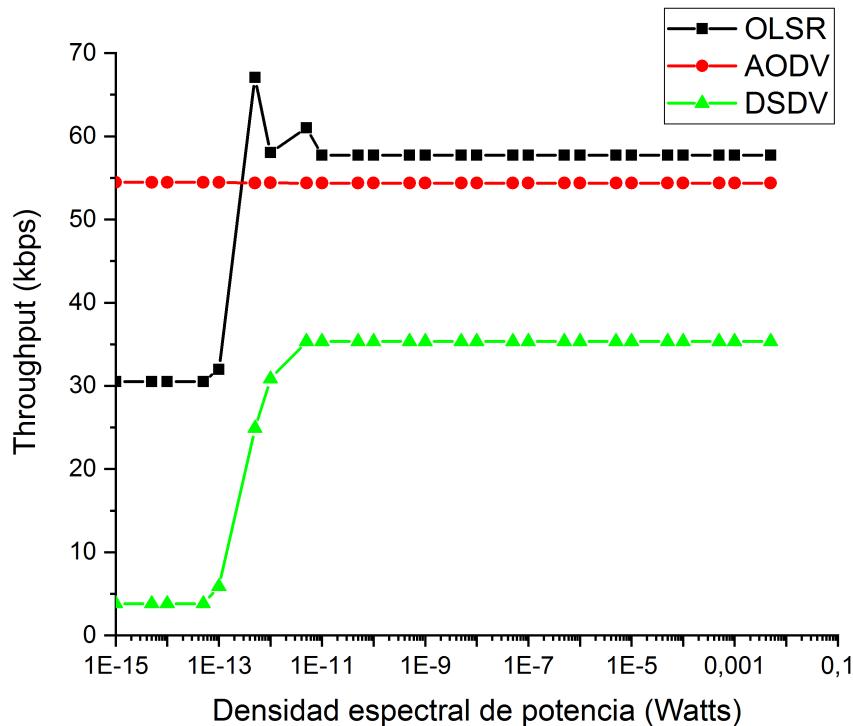


Figura 5-11.: Comparación de resultados de simulación para throughput en canales mixtos PLC - WiFi con protocolos de enrutamiento DSDV, OLSR y AODV en primer escenario

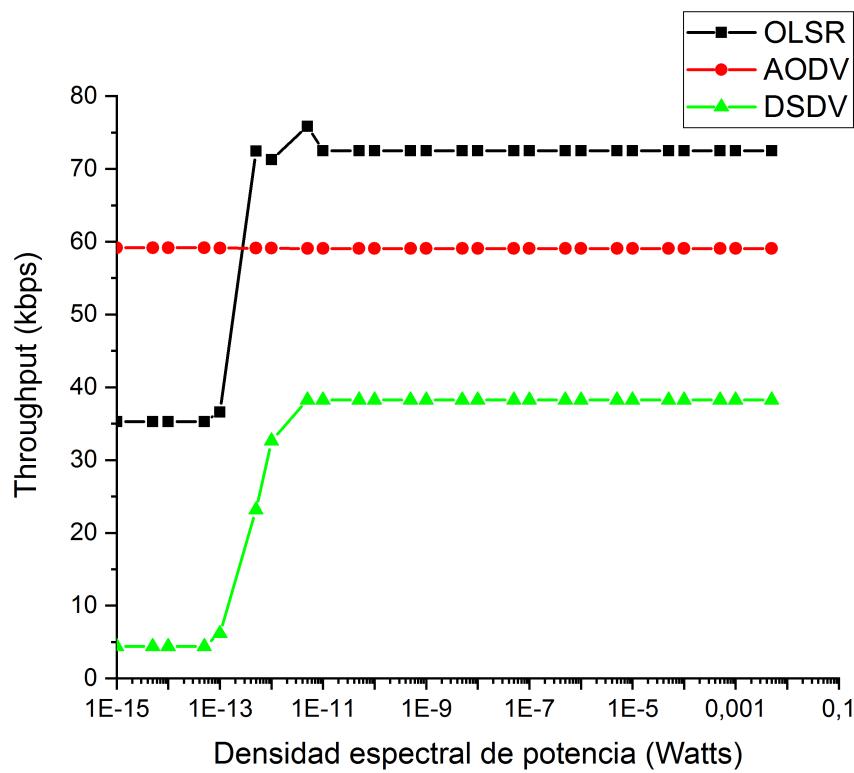


Figura 5-12.: Comparación de resultados de simulación para throughput en canales mixtos PLC - Wifi con protocolos de enrutamiento DSDV, OLSR y AODV en segundo escenario

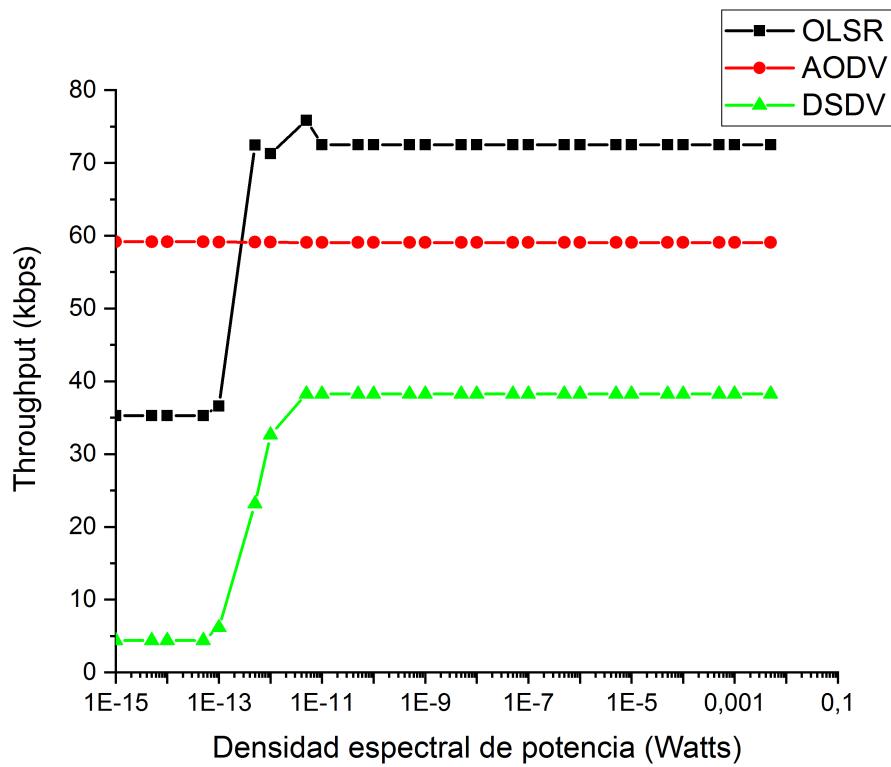


Figura 5-13.: Comparación de resultados de simulación para throughput en canales mixtos PLC - WiFi con protocolos de enruteamiento DSDV, OLSR y AODV en tercer escenario

6. Conclusiones y recomendaciones

6.1. Conclusiones

A partir del escenario de simulación planteado en el presente trabajo, en el que se construye un backbone de comunicación usando un canal netamente PLC, puede concluirse que este tipo de canales en espacios internos a una vivienda poseen una buena perspectiva para ser usados como medio de extensión o interconexión de redes mesh en aplicaciones de redes comunitarias, sin embargo, es necesario ampliar las perspectivas de simulación, aumentando la cantidad y complejidad de los parámetros de entrada del modelo.

Teniendo en cuenta que protocolos de enrutamiento como el DSDV y AODV, basados en métricas de vector distancia, presentan desempeños que se mantienen casi constantes en todo el rango de potencias de transmisión sobre el canal PLC, se percibe la posibilidad de implementación de pruebas tanto en simulación como en emulación y experimentación en escenarios reales con protocolos de enrutamiento que varíen de acuerdo con el medio de transmisión.

La comparación del canal netamente PLC con el WiFi en modo de operación Ad-Hoc en condiciones topológicas idénticas, planteado en el primer escenario de simulación, permite concluir que, bajo las condiciones asignadas a este experimento, el canal de comunicación inalámbrico podría ser reemplazado por el cableado eléctrico de las viviendas en casos donde la red WiFi no sea una opción viable, sin embargo, se hace necesaria la realización de pruebas en campo que permitan corroborar los resultados de simulación.

A partir de los datos encontrados en los escenarios de simulación del presente trabajo, donde se comparan los canales PLC y WiFi en modo de operación Ad-Hoc, puede concluirse que el cambio de interfaz no representa una degradación tangible de la calidad del servicio para audio y video del enlace de comunicación, sin embargo el aumento de potencia planteado sobre las simulación puede representar problemas en implementaciones reales por compatibilidad electromagnética, dado que el canal se utiliza esencialmente para la distribución de energía.

Tomando como base los resultados de simulación de las pruebas sobre enlaces mixtos PLC-WiFi con los parámetros aquí planteados, se concluye que el protocolo de enrutamiento que

posee un mejor desempeño, sin tener en cuenta las posibles limitaciones de potencia sobre el canal, es el OLSR, por lo que se abre la posibilidad de iniciar experimentaciones con el protocolo B.A.T.M.A.N. (Better Approach to Mobile Ad-hoc Networking) que ha sido desarrollado para reemplazarlo.

6.2. Recomendaciones

Teniendo en cuenta las limitaciones propias a las pruebas realizadas a través de la simulación, en cuanto a la reducción en la complejidad de las condiciones que se presentan en escenarios reales, el presente trabajo presenta limitaciones en sus resultados. Es por esto por lo que es indispensable ahondar en los escenarios de simulación, haciendo que éstos sean cada vez más completos en la cantidad de entradas y en la complejidad de los modelos.

Esto, sin embargo, se encuentra limitado por las capacidades computacionales y la complejidad del modelo matemático usado, por lo que eventualmente será necesario realizar pruebas experimentales en escenarios reales, razón por la cual el trabajo futuro también debe enfocarse en la consecución de equipos de transmisión sobre canales PLC que permitan iniciar experimentaciones y verificaciones de los modelos planteados.

De igual forma se hace necesaria la revisión de los modelos y desarrollos en simulación, en especial el módulo usado para NS3 que permite la simulación de enlaces PLC, dado que la no inclusión de éste en el software oficial hace que las versiones actuales del programa y de los sistemas operativos Linux no lo soporten, dificultando en gran medida los avances en simulación y limitando la posibilidad de complejizar los escenarios experimentales. Por tanto, es esencial el desarrollo de un módulo especializado en canales PLC y de interfaz mixta para el software de simulación de eventos discretos NS3 que se encuentre respaldado por el software para todas sus versiones, lo que permitiría plantear escenarios de simulación mucho más amplios.

Un parámetro esencial en los canales PLC y que no fue considerado en el presente trabajo debido al aumento de la complejidad, del tiempo de simulación y de los requerimientos computacionales que implicaría en el mismo, es el ruido sobre el canal, por esta razón es necesario que escenarios futuros evalúan su influencia sobre la calidad del servicio en redes descentralizadas.

Las pruebas realizadas en este trabajo consideran un escenario dentro del hogar, sin embargo, es de interés para la implementación de redes la perspectiva de escenarios que contemplen la transmisión de información entre más de una vivienda, por lo que se requiere un trabajo futuro con consideraciones de topologías más amplias de redes eléctricas, tales como los brea-

kers y cableados externos al hogar, así como la consideración de redes de media y alta tensión.

Por último, es importante generar hardware y dispositivos especializados en transmisión a través de canales PLC que permitan la implementación de redes Ad-Hoc con interfaces mixtas que hagan posible la implementación práctica de este tipo de redes y, en consecuencia, comprobar los escenarios de simulación propuestos en el presente trabajo investigativo.

A. Anexo: Código de NS3 para la comparación de protocolos de enrutamiento en canal PLC

```
1 #include <fstream>
2 #include <iostream>
3 #include "ns3/core-module.h"
4 #include "ns3/network-module.h"
5 #include "ns3/internet-module.h"
6 #include "ns3/mobility-module.h"
7 #include "ns3/wifi-module.h"
8 #include "ns3/aodv-module.h"
9 #include "ns3/olsr-module.h"
10 #include "ns3/dsdu-module.h"
11 #include "ns3/dsr-module.h"
12 #include "ns3/applications-module.h"
13 #include "ns3/flow-monitor-helper.h"
14
15 #include <sstream>
16 #include <time.h>
17
18 #include <ns3/core-module.h>
19 #include <ns3/nstime.h>
20 #include <ns3/simulator.h>
21 #include <ns3/output-stream-wrapper.h>
22 #include "ns3/plc.h"
23 #include "ns3/internet-module.h"
24 #include "ns3/applications-module.h"
25
26 using namespace ns3;
27 using namespace dsr;
28
29 NS_LOG_COMPONENT_DEFINE ("PLC-routing-compare");
30
31
32
33 class PLCRoutingExperiment
34 {
35 public:
```

```

36 PLCRoutingExperiment ();
37 void Run (int nSinks, double txp, std::string CSVfileName);
38 std::string CommandSetup (int argc, char **argv);
39
40 private:
41     Ptr<Socket> SetupPacketReceive (Ipv4Address addr, Ptr<Node> node);
42     void ReceivePacket (Ptr<Socket> socket);
43     void CheckThroughput ();
44
45     uint32_t port;
46     uint32_t bytesTotal;
47     uint32_t packetsReceived;
48
49     std::string m_CSVfileName;
50     int m_nSink_r;
51     int m_nSink_e;
52     std::string m_protocolName;
53     double m_txp;
54     uint32_t m_protocol;
55     std::string m_packetSize;
56 };
57
58 PLCRoutingExperiment::PLCRoutingExperiment ()
59     : port (9),
60      bytesTotal (0),
61      packetsReceived (0),
62      m_CSVfileName ("PLC-routing.output.csv"),
63      m_nSink_r(6),
64      m_nSink_e(0),
65      m_txp(0.005),
66      m_protocol (4), // 1=OLSR;2=AODV;3=DSDV;4=DSR
67      m_packetSize ("64")
68
69 {
70 }
71
72 static inline std::string
73 PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet)
74 {
75     SocketAddressTag tag;
76     bool found;
77     found = packet->PeekPacketTag (tag);
78     std::ostringstream oss;
79
80     oss << Simulator::Now ().GetSeconds () << " " << socket->GetNode ()->GetId ();
81
82     if (found)
83     {

```

```

84     InetSocketAddress addr = InetSocketAddress::ConvertFrom (tag.GetAddress ());
85     oss << " received one packet from " << addr.GetIpv4 ();
86 }
87 else
88 {
89     oss << " received one packet!";
90 }
91 return oss.str ();
92 }

93
94 void
95 PLCRoutingExperiment::ReceivePacket (Ptr<Socket> socket)
96 {
97     Ptr<Packet> packet;
98     while ((packet = socket->Recv ()))
99     {
100         bytesTotal += packet->GetSize ();
101         packetsReceived += 1;
102         NS_LOG_UNCOND (PrintReceivedPacket (socket, packet));
103     }
104 }

105
106
107
108 void
109 PLCRoutingExperiment::CheckThroughput ()
110 {
111     double kbs = (bytesTotal * 8.0) / 1000;
112     bytesTotal = 0;
113
114     std::ofstream out (m_CSVfileName.c_str (), std::ios::app);
115
116     out << (Simulator::Now ()).GetSeconds () << ","
117             << kbs << ","
118             << packetsReceived << ","
119             << m_nSink_r << ","
120             << m_protocolName << ","
121             << m_txp << ""
122             << std::endl;
123
124     out.close ();
125     packetsReceived = 0;
126     Simulator::Schedule (Seconds (1.0), &PLCRoutingExperiment::CheckThroughput, this);
127 }

128
129 Ptr<Socket>
130 PLCRoutingExperiment::SetupPacketReceive (Ipv4Address addr, Ptr<Node> node)
131 {

```

A Anexo: Código de NS3 para la comparación de protocolos de enrutamiento en canal
64 PLC

```
132 TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
133 Ptr<Socket> sink = Socket::CreateSocket (node, tid);
134 InetSocketAddress local = InetSocketAddress (addr, port);
135 sink->Bind (local);
136 sink->SetRecvCallback (MakeCallback (&PLCRoutingExperiment::ReceivePacket, this));
137
138     return sink;
139 }
140
141 std::string
142 PLCRoutingExperiment::CommandSetup (int argc, char **argv)
143 {
144     CommandLine cmd;
145     cmd.AddValue ("CSVfileName", "The name of the CSV output file name", m_CSVfileName);
146     cmd.AddValue ("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR", m_protocol);
147     cmd.AddValue ("packetSize", "Packet Size", m_packetSize);
148     cmd.AddValue ("txp", "Transmit power spectral density", m_txp);
149     cmd.AddValue ("nSinkr", "Sink Receptor", m_nSink_r);
150     cmd.AddValue ("nSinke", "Sink Emitter", m_nSink_e);
151     cmd.Parse (argc, argv);
152     return m_CSVfileName;
153 }
154
155
156
157
158 int main (int argc, char *argv[])
159 {
160     PLCRoutingExperiment experiment;
161     std::string CSVfileName = experiment.CommandSetup (argc, argv);
162     std::ofstream out (CSVfileName.c_str ());
163     out << "#" << "SimulationSecond," <<
164     "ReceiveRate," <<
165     "PacketsReceived," <<
166     "NumberOfSinks," <<
167     "RoutingProtocol," <<
168     "TransmissionPower" <<
169     std::endl;
170     out.close ();
171
172     int nSinks = 2;
173     double txp = 1e-8;
174     experiment.Run (nSinks, txp, CSVfileName);}
175
176
177
178 void
179 PLCRoutingExperiment::Run (int nSinks, double txp, std::string CSVfileName)
```

```

180 {
181     Packet::EnablePrinting ();
182     m_CSVfileName = CSVfileName;
183     std::string rate ("512kbps");
184     double TotalTime = 200.0;
185
186
187     Config::SetDefault ("ns3::OnOffApplication::PacketSize",StringValue
188     ↵ (m_packetSize));
189     Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue (rate));
190
191     PLC_SpectrumModelHelper smHelper;
192     Ptr<const SpectrumModel> sm;
193     sm = smHelper.GetSpectrumModel(0, 10e6, 100);
194
195     // Define transmit power spectral density
196     Ptr<SpectrumValue> txPsd = Create<SpectrumValue> (sm);
197     (*txPsd) = m_txp; // -50dBm/Hz
198
199     Ptr<PLC_Cable> cable = CreateObject<PLC_AL3x95XLPE_Cable> (sm);
200     Ptr<PLC_ConstImpedance> shuntImp2 = Create<PLC_ConstImpedance> (sm,
201     ↵ PLC_Value(50, 0));
202     Ptr<PLC_ConstImpedance> shuntImp3 = Create<PLC_ConstImpedance> (sm,
203     ↵ PLC_Value(50, 0));
204     Ptr<PLC_ConstImpedance> shuntImp4 = Create<PLC_ConstImpedance> (sm,
205     ↵ PLC_Value(50, 0));
206     Ptr<PLC_ConstImpedance> shuntImp5 = Create<PLC_ConstImpedance> (sm,
207     ↵ PLC_Value(50, 0));
208     Ptr<PLC_ConstImpedance> shuntImp21 = Create<PLC_ConstImpedance> (sm,
209     ↵ PLC_Value(50, 0));
210
211     Ptr<PLC_Node> n1 = CreateObject<PLC_Node> ();
212     Ptr<PLC_Node> n2 = CreateObject<PLC_Node> ();
213     Ptr<PLC_Node> n3 = CreateObject<PLC_Node> ();
214     Ptr<PLC_Node> n4 = CreateObject<PLC_Node> ();
215     Ptr<PLC_Node> n5 = CreateObject<PLC_Node> ();
216     Ptr<PLC_Node> n20 = CreateObject<PLC_Node> ();
217     Ptr<PLC_Node> n21 = CreateObject<PLC_Node> ();
218
219     n1->SetPosition(0,0,0);
220     n2->SetPosition(3,0,0);
221     n3->SetPosition(6,0,0);
222     n4->SetPosition(9,0,0);
223     n5->SetPosition(14,0,0);
224     n20->SetPosition(16,-4,0);
225     n21->SetPosition(14,-4,0);
226
227     n1->SetName("Node1");

```

```

222         n2->SetName("Junction2");
223         n3->SetName("Junction3");
224         n4->SetName("Junction4");
225         n5->SetName("Junction5");
226         n20->SetName("Node20");
227         n21->SetName("Node21");
228
229
230     PLC_NodeList nodes;
231     nodes.push_back(n1);
232     nodes.push_back(n2);
233     nodes.push_back(n3);
234     nodes.push_back(n4);
235     nodes.push_back(n5);
236     nodes.push_back(n20);
237     nodes.push_back(n21);
238
239     CreateObject<PLC_Line> (cable, n1, n2);
240     CreateObject<PLC_Line> (cable, n2, n3);
241     CreateObject<PLC_Line> (cable, n3, n4);
242     CreateObject<PLC_Line> (cable, n4, n5);
243     CreateObject<PLC_Line> (cable, n5, n21);
244     CreateObject<PLC_Line> (cable, n21, n20);
245
246     PLC_ChannelHelper channelHelper(sm);
247     channelHelper.Install(nodes);
248     Ptr<PLC_Channel> channel = channelHelper.GetChannel();
249
250
251
252     Ptr<PLC_Outlet> outlet1 = CreateObject<PLC_Outlet> (n2, shuntImp2);
253     Ptr<PLC_Outlet> outlet2 = CreateObject<PLC_Outlet> (n3, shuntImp3);
254     Ptr<PLC_Outlet> outlet4 = CreateObject<PLC_Outlet> (n4, shuntImp4);
255     Ptr<PLC_Outlet> outlet5 = CreateObject<PLC_Outlet> (n5, shuntImp5);
256     Ptr<PLC_Outlet> outlet21 = CreateObject<PLC_Outlet> (n21, shuntImp21);
257
258     PLC_NetDeviceHelper deviceHelper(sm, txPsd, nodes);
259     deviceHelper.DefinePhyType(TypeId::LookupByName
260         → ("ns3::PLC_InformationRatePhy"));
261     deviceHelper.DefineMacType(TypeId::LookupByName ("ns3::PLC_ArqMac"));
262     deviceHelper.SetHeaderModulationAndCodingScheme(ModulationAndCodingScheme(BPSK_1_4,0));
263     deviceHelper.SetPayloadModulationAndCodingScheme(ModulationAndCodingScheme(BPSK_1_2,0));
264     deviceHelper.Setup();
265
266     channel->InitTransmissionChannels();
267     channel->CalcTransmissionChannels();
268
        NodeContainer nodes1;

```

```
269     nodes1=deviceHelper.GetNS3Nodes();
270
271
272     NetDeviceContainer d;
273     d = deviceHelper.GetNetDevices();
274
275     AodvHelper aodv;
276     OlsrHelper olsr;
277     DsdvHelper dsdv;
278     DsrHelper dsr;
279     DsrMainHelper dsrMain;
280     Ipv4ListRoutingHelper list;
281     InternetStackHelper internet;
282
283
284
285     switch (m_protocol)
286     {
287         case 1:
288             list.Add (olsr, 100);
289             m_protocolName = "OLSR";
290             break;
291         case 2:
292             list.Add (aodv, 100);
293             m_protocolName = "AODV";
294             break;
295         case 3:
296             list.Add (dsdv, 100);
297             m_protocolName = "DSDV";
298             break;
299         case 4:
300             m_protocolName = "DSR";
301             break;
302         default:
303             NS_FATAL_ERROR ("No such protocol:" << m_protocol);
304     }
305
306     if (m_protocol < 4)
307     {
308         internet.SetRoutingHelper (list);
309         internet.Install (nodes1);
310     }
311     else if (m_protocol == 4)
312     {
313         internet.Install (nodes1);
314         dsrMain.Install (dsr, nodes1);
315     }
316
```

```
317     NS_LOG_INFO ("assigning ip address");
318
319     Ipv4AddressHelper addressAdhoc;
320     addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
321     Ipv4InterfaceContainer adhocInterfaces;
322     adhocInterfaces = addressAdhoc.Assign (d);
323
324     OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
325     onoff1.SetAttribute ("OffTime",StringValue
326         → ("ns3::LogNormalRandomVariable[Mu=0.4026,Sigma=0.0352]"));
327     onoff1.SetAttribute ("OnTime",StringValue
328         → ("ns3::WeibullRandomVariable[Shape=10.2063,Scale=57480.9]"));
329     onoff1.SetAttribute ("MaxBytes",UintegerValue(10989173));
330     Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress (m_nSink_r), nodes1.Get
331         → (m_nSink_r));
332
333     AddressValue remoteAddress (InetSocketAddress (adhocInterfaces.GetAddress (m_nSink_r),
334         → port));
335     onoff1.SetAttribute ("Remote", remoteAddress);
336
337
338
339     AsciiTraceHelper ascii;
340     internet.EnableAsciiIpv4All (ascii.CreateFileStream ("PLCinternet.tr"));
341     Ptr<FlowMonitor> flowmon;
342     FlowMonitorHelper flowmonHelper;
343     flowmon = flowmonHelper.InstallAll ();
344
345     NS_LOG_INFO ("Run Simulation.");
346
347     CheckThroughput ();
348
349     Simulator::Stop (Seconds (TotalTime));
350     Simulator::Run ();
351     flowmon->SerializeToXmlFile ("plc_lan_wifi.xml", true, true);
352     Simulator::Destroy ();
353
354 }
```

B. Anexo: Código de NS3 para la comparación de protocolos de enrutamiento en canal inalámbrico

```
1 #include <iostream>
2 #include <fstream>
3 #include "ns3/core-module.h"
4 #include "ns3/network-module.h"
5 #include "ns3/internet-module.h"
6 #include "ns3/mobility-module.h"
7 #include "ns3/wifi-module.h"
8 #include "ns3/aodv-module.h"
9 #include "ns3/olsr-module.h"
10 #include "ns3/dsuv-module.h"
11 #include "ns3/dsr-module.h"
12 #include "ns3/applications-module.h"
13 #include "ns3/flow-monitor-helper.h"
14 using namespace ns3;
15 using namespace dsr;
16
17 NS_LOG_COMPONENT_DEFINE ("manet-routing-compare");
18
19 class RoutingExperiment
20 {
21 public:
22     RoutingExperiment ();
23     void Run (int nSinks, double txp, std::string CSVfileName);
24     std::string CommandSetup (int argc, char **argv);
25
26 private:
27     Ptr<Socket> SetupPacketReceive (Ipv4Address addr, Ptr<Node> node);
28     void ReceivePacket (Ptr<Socket> socket);
29     void CheckThroughput ();
30
31     uint32_t port;
32     uint32_t bytesTotal;
33     uint32_t packetsReceived;
34
35     std::string m_CSVfileName;
```

```
36 int m_nSink_r;
37 int m_nSink_e;
38 std::string m_protocolName;
39 double m_txp;
40 bool m_traceMobility;
41 uint32_t m_protocol;
42 };
43
44 RoutingExperiment::RoutingExperiment ()
45 : port (9),
46   bytesTotal (0),
47   packetsReceived (0),
48   m_CSVfileName ("manet-routing.output.csv"),
49   m_nSink_r(6),
50   m_nSink_e(0),
51   m_txp(1.00E-3),
52   m_traceMobility (false),
53   m_protocol (4) // 1=OLSR;2=AODV;3=DSDV;4=DSR
54 {
55 }
56
57 static inline std::string
58 PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet)
59 {
60   SocketAddressTag tag;
61   bool found;
62   found = packet->PeekPacketTag (tag);
63   std::ostringstream oss;
64
65   oss << Simulator::Now ().GetSeconds () << " " << socket->GetNode ()->GetId ();
66
67   if (found)
68   {
69     InetSocketAddress addr = InetSocketAddress::ConvertFrom (tag.GetAddress ());
70     oss << " received one packet from " << addr.GetIpv4 ();
71   }
72   else
73   {
74     oss << " received one packet!";
75   }
76   return oss.str ();
77 }
78
79 void
80 RoutingExperiment::ReceivePacket (Ptr<Socket> socket)
81 {
82   Ptr<Packet> packet;
83   while ((packet = socket->Recv ()))
```

```

84     {
85         bytesTotal += packet->GetSize ();
86         packetsReceived += 1;
87         NS_LOG_UNCOND (PrintReceivedPacket (socket, packet));
88     }
89 }
90
91 void
92 RoutingExperiment::CheckThroughput ()
93 {
94     double kbs = (bytesTotal * 8.0) / 1000;
95     bytesTotal = 0;
96
97     std::ofstream out (m_CSVfileName.c_str (), std::ios::app);
98
99     out << (Simulator::Now ()).GetSeconds () << ","
100        << kbs << ","
101        << packetsReceived << ","
102        << m_nSink_r << ","
103        << m_protocolName << ","
104        << m_txp << ""
105        << std::endl;
106
107     out.close ();
108     packetsReceived = 0;
109     Simulator::Schedule (Seconds (1.0), &RoutingExperiment::CheckThroughput, this);
110 }
111
112 Ptr<Socket>
113 RoutingExperiment::SetupPacketReceive (Ipv4Address addr, Ptr<Node> node)
114 {
115     TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
116     Ptr<Socket> sink = Socket::CreateSocket (node, tid);
117     InetSocketAddress local = InetSocketAddress (addr, port);
118     sink->Bind (local);
119     sink->SetRecvCallback (MakeCallback (&RoutingExperiment::ReceivePacket, this));
120
121     return sink;
122 }
123
124 std::string
125 RoutingExperiment::CommandSetup (int argc, char **argv)
126 {
127     CommandLine cmd;
128     cmd.AddValue ("CSVfileName", "The name of the CSV output file name", m_CSVfileName);
129     cmd.AddValue ("traceMobility", "Enable mobility tracing", m_traceMobility);
130     cmd.AddValue ("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR", m_protocol);
131     cmd.Parse (argc, argv);

```

```
132     return m_CSVfileName;
133 }
134
135 int
136 main (int argc, char *argv[])
137 {
138     RoutingExperiment experiment;
139     std::string CSVfileName = experiment.CommandSetup (argc, argv);
140
141     std::ofstream out (CSVfileName.c_str ());
142     out << "SimulationSecond," <<
143         "ReceiveRate," <<
144         "PacketsReceived," <<
145         "NumberOfSinks," <<
146         "RoutingProtocol," <<
147         "TransmissionPower" <<
148     std::endl;
149     out.close ();
150
151     int nSinks = 10;
152     double txp = 1e-50;
153
154     experiment.Run (nSinks, txp, CSVfileName);
155 }
156
157 void
158 RoutingExperiment::Run (int nSinks, double txp, std::string CSVfileName)
159 {
160     Packet::EnablePrinting ();
161     m_CSVfileName = CSVfileName;
162
163     int nWifis = 7;
164
165     double TotalTime = 200.0;
166     std::string rate ("512bps");
167     std::string phyMode ("DsssRate11Mbps");
168     std::string tr_name ("manet-routing-compare");
169     m_protocolName = "protocol";
170
171     Config::SetDefault ("ns3::OnOffApplication::PacketSize", StringValue ("64"));
172     Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue (rate));
173
174     Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode", StringValue
175                         → (phyMode));
176
177     NodeContainer adhocNodes;
178     adhocNodes.Create (nWifis);
```

```

179 WifiHelper wifi;
180 wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
181
182 YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
183 YansWifiChannelHelper wifiChannel;
184 wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
185 wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
186 wifiPhy.SetChannel (wifiChannel.Create ());
187
188 WifiMacHelper wifiMac;
189 wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
190     "DataMode",StringValue (phyMode),
191     "ControlMode",StringValue (phyMode));
192
193 wifiPhy.Set ("TxPowerStart",DoubleValue (m_txp));
194 wifiPhy.Set ("TxPowerEnd", DoubleValue (m_txp));
195
196 wifiMac.SetType ("ns3::AdhocWifiMac");
197 NetDeviceContainer adhocDevices = wifi.Install (wifiPhy, wifiMac, adhocNodes);
198
199
200 MobilityHelper mobility;
201 Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
202 positionAlloc->Add (Vector (0.0, 0.0, 0.0));
203 positionAlloc->Add (Vector (3.0, 0.0, 0.0));
204 positionAlloc->Add (Vector (6.0, 0.0, 0.0));
205 positionAlloc->Add (Vector (9.0, 0.0, 0.0));
206 positionAlloc->Add (Vector (14.0, 0.0, 0.0));
207 positionAlloc->Add (Vector (16.0, -4.0, 0.0));
208 positionAlloc->Add (Vector (14.0, -4.0, 0.0));
209 mobility.SetPositionAllocator (positionAlloc);
210 mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
211 mobility.Install (adhocNodes);
212
213 AodvHelper aodv;
214 OlsrHelper olsr;
215 DsdvHelper dsdv;
216 DsrHelper dsr;
217 DsrMainHelper dsrMain;
218 Ipv4ListRoutingHelper list;
219 InternetStackHelper internet;
220
221 switch (m_protocol)
222 {
223     case 1:
224         list.Add (olsr, 100);
225         m_protocolName = "OLSR";
226         break;

```

```
227     case 2:
228         list.Add (aodv, 100);
229         m_protocolName = "AODV";
230         break;
231     case 3:
232         list.Add (dsdv, 100);
233         m_protocolName = "DSDV";
234         break;
235     case 4:
236         m_protocolName = "DSR";
237         break;
238     default:
239         NS_FATAL_ERROR ("No such protocol:" << m_protocol);
240     }
241
242     if (m_protocol < 4)
243     {
244         internet.SetRoutingHelper (list);
245         internet.Install (adhocNodes);
246     }
247     else if (m_protocol == 4)
248     {
249         internet.Install (adhocNodes);
250         dsrMain.Install (dsr, adhocNodes);
251     }
252
253     NS_LOG_INFO ("assigning ip address");
254
255     Ipv4AddressHelper addressAdhoc;
256     addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
257     Ipv4InterfaceContainer adhocInterfaces;
258     adhocInterfaces = addressAdhoc.Assign (adhocDevices);
259
260     OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
261     onoff1.SetAttribute("OffTime",StringValue
262     ↳ ("ns3::LogNormalRandomVariable[Mu=0.4026,Sigma=0.0352]"));
263     onoff1.SetAttribute("OnTime",StringValue
264     ↳ ("ns3::WeibullRandomVariable[Shape=10.2063,Scale=57480.9]"));
265     onoff1.SetAttribute("MaxBytes",UintegerValue(10989173));
266
267     Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress (m_nSink_r),
268     ↳ adhocNodes.Get (m_nSink_r));
269
270     AddressValue remoteAddress (InetSocketAddress (adhocInterfaces.GetAddress (m_nSink_r),
271     ↳ port));
272     onoff1.SetAttribute ("Remote", remoteAddress);
273
274     Ptr<UniformRandomVariable> var = CreateObject<UniformRandomVariable> ();
```

```
271 ApplicationContainer temp = onoff1.Install (adhocNodes.Get (m_nSink_e));
272 temp.Start (Seconds (var->GetValue (100.0,101.0)));
273 temp.Stop (Seconds (TotalTime));
274
275 AsciiTraceHelper ascii;
276 internet.EnableAsciiIpv4All (ascii.CreateFileStream ("manet_internet.tr"));
277
278 Ptr<FlowMonitor> flowmon;
279 FlowMonitorHelper flowmonHelper;
280 flowmon = flowmonHelper.InstallAll ();
281
282
283 NS_LOG_INFO ("Run Simulation.");
284
285 CheckThroughput ();
286
287 Simulator::Stop (Seconds (TotalTime));
288 Simulator::Run ();
289
290 Simulator::Destroy ();
291 }
```

C. Anexo: Código de NS3 para la implementación de primer escenario de simulación mixto PLC-WiFi

```
1 #include <iostream>
2 #include <string>
3 #include "ns3/core-module.h"
4 #include "ns3/network-module.h"
5 #include "ns3/applications-module.h"
6 #include "ns3/mobility-module.h"
7 #include "ns3/config-store-module.h"
8 #include "ns3/wifi-module.h"
9 #include "ns3/csma-module.h"
10 #include "ns3/olsr-helper.h"
11 #include "ns3/internet-module.h"
12 #include "ns3/netanim-module.h"
13 #include "ns3/plc.h"
14 #include "ns3/aodv-module.h"
15 #include "ns3/olsr-module.h"
16 #include "ns3/dsuv-module.h"
17 #include "ns3/dsr-module.h"

18
19 using namespace ns3;
20 NS_LOG_COMPONENT_DEFINE ("MixedWireless");
21 static void
22 CourseChangeCallback (std::string path, Ptr<const MobilityModel> model)
23 {
24     Vector position = model->GetPosition ();
25     std::cout << "CourseChange " << path << " x=" << position.x << ", y=" << position.y <<
26     " , z=" << position.z << std::endl;
27 }
28
29 int
30 main (int argc, char *argv[])
31 {
32     uint32_t backboneNodes = 2;
33     uint32_t infraNodes = 2;
34     uint32_t lanNodes = 2;
```

```

35     uint32_t stopTime = 104;
36     bool useCourseChangeCallback = false;
37
38     Config::SetDefault ("ns3::OnOffApplication::PacketSize", StringValue ("64"));
39     Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("512b/s"));
40
41
42     CommandLine cmd;
43     cmd.AddValue ("backboneNodes", "number of backbone nodes", backboneNodes);
44     cmd.AddValue ("infraNodes", "number of leaf nodes", infraNodes);
45     cmd.AddValue ("lanNodes", "number of LAN nodes", lanNodes);
46     cmd.AddValue ("stopTime", "simulation stop time (seconds)", stopTime);
47     cmd.AddValue ("useCourseChangeCallback", "whether to enable course change tracing",
48                   ↵ useCourseChangeCallback);
49
50     cmd.Parse (argc, argv);
51
52     if (stopTime < 10)
53     {
54         std::cout << "Use a simulation stop time >= 10 seconds" << std::endl;
55         exit (1);
56     }
57     PLC_SpectrumModelHelper smHelper;
58     Ptr<const SpectrumModel> sm;
59     sm = smHelper.GetSpectrumModel(0, 10e6, 100);
60
61         Ptr<SpectrumValue> txPsd = Create<SpectrumValue> (sm);
62         (*txPsd) = 0.005; // -50dBm/Hz
63         Ptr<PLC_Cable> cable = CreateObject<PLC_AL3x95XLPE_Cable> (sm);
64         Ptr<PLC_ConstImpedance> shuntImp2 = Create<PLC_ConstImpedance> (sm,
65                           ↵ PLC_Value(50, 0));
66         Ptr<PLC_ConstImpedance> shuntImp3 = Create<PLC_ConstImpedance> (sm,
67                           ↵ PLC_Value(50, 0));
68         Ptr<PLC_ConstImpedance> shuntImp4 = Create<PLC_ConstImpedance> (sm,
69                           ↵ PLC_Value(50, 0));
70         Ptr<PLC_ConstImpedance> shuntImp5 = Create<PLC_ConstImpedance> (sm,
71                           ↵ PLC_Value(50, 0));
72         Ptr<PLC_ConstImpedance> shuntImp21 = Create<PLC_ConstImpedance> (sm,
73                           ↵ PLC_Value(50, 0));
74
75         Ptr<PLC_Node> n1 = CreateObject<PLC_Node> ();
76         Ptr<PLC_Node> n2 = CreateObject<PLC_Node> ();
77         Ptr<PLC_Node> n3 = CreateObject<PLC_Node> ();
78         Ptr<PLC_Node> n4 = CreateObject<PLC_Node> ();
79         Ptr<PLC_Node> n5 = CreateObject<PLC_Node> ();
80         Ptr<PLC_Node> n20 = CreateObject<PLC_Node> ();
81         Ptr<PLC_Node> n21 = CreateObject<PLC_Node> ();

```

```
77
78
79     n1->SetPosition(0,0,0);
80     n2->SetPosition(3,0,0);
81     n3->SetPosition(6,0,0);
82     n4->SetPosition(9,0,0);
83     n5->SetPosition(14,0,0);
84     n20->SetPosition(16,-4,0);
85     n21->SetPosition(14,-4,0);
86
87
88     n1->SetName("Node1");
89     n2->SetName("Junction2");
90     n3->SetName("Junction3");
91     n4->SetName("Junction4");
92     n5->SetName("Junction5");
93     n20->SetName("Node20");
94     n21->SetName("Node21");
95
96
97     PLC_NodeList nodes;
98     nodes.push_back(n1);
99     nodes.push_back(n2);
100    nodes.push_back(n3);
101    nodes.push_back(n4);
102    nodes.push_back(n5);
103    nodes.push_back(n20);
104    nodes.push_back(n21);
105
106    CreateObject<PLC_Line> (cable, n1, n2);
107    CreateObject<PLC_Line> (cable, n2, n3);
108    CreateObject<PLC_Line> (cable, n3, n4);
109    CreateObject<PLC_Line> (cable, n4, n5);
110    CreateObject<PLC_Line> (cable, n5, n21);
111    CreateObject<PLC_Line> (cable, n21, n20);
112
113
114     PLC_ChannelHelper channelHelper(sm);
115     channelHelper.Install(nodes);
116     Ptr<PLC_Channel> channel = channelHelper.GetChannel();
117
118
119     Ptr<PLC_Outlet> outlet1 = CreateObject<PLC_Outlet> (n2, shuntImp2);
120     Ptr<PLC_Outlet> outlet2 = CreateObject<PLC_Outlet> (n3, shuntImp3);
121     Ptr<PLC_Outlet> outlet4 = CreateObject<PLC_Outlet> (n4, shuntImp4);
122     Ptr<PLC_Outlet> outlet5 = CreateObject<PLC_Outlet> (n5, shuntImp5);
123     Ptr<PLC_Outlet> outlet21 = CreateObject<PLC_Outlet> (n21, shuntImp21);
124
```

```

125     PLC_NetDeviceHelper deviceHelper(sm, txPsd, nodes);
126     deviceHelper.DefinePhyType(TypeId::LookupByName ("ns3::PLC_InformationRatePhy"));
127     deviceHelper.DefineMacType(TypeId::LookupByName ("ns3::PLC_ArqMac"));
128     deviceHelper.SetHeaderModulationAndCodingScheme(ModulationAndCodingScheme(BPSK_1_4,0));
129     deviceHelper.SetPayloadModulationAndCodingScheme(ModulationAndCodingScheme(BPSK_1_2,0));
130     deviceHelper.Setup();
131     channel->InitTransmissionChannels();
132     channel->CalcTransmissionChannels();
133     NodeContainer PLCBackbone;
134     PLCBackbone=deviceHelper.GetNS3Nodes();
135
136
137     NetDeviceContainer backboneDevices;
138     backboneDevices = deviceHelper.GetNetDevices();
139
140     NS_LOG_INFO ("Enabling OLSR routing on all backbone nodes");
141     InternetStackHelper internet;
142     DsdvHelper dsdv;
143     internet.SetRoutingHelper (dsdv);
144     internet.Install (PLCBackbone);
145     Ipv4AddressHelper ipAddrs;
146     ipAddrs.SetBase ("192.168.0.0", "255.255.255.0");
147     ipAddrs.Assign (backboneDevices);
148     MobilityHelper mobility;
149     mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
150     mobility.Install(PLCBackbone);
151     ipAddrs.SetBase ("172.16.0.0", "255.255.255.0");
152
153     YansWifiPhyHelper wifiPhy1 = YansWifiPhyHelper::Default ();
154
155     for (uint32_t i = 0; i < 1; ++i)
156     {
157         NS_LOG_INFO ("Configuring wireless network 1 for backbone node ");
158         NodeContainer stas1;
159         stas1.Create (1);
160         NodeContainer infra1 (PLCBackbone.Get (0), stas1);
161
162         WifiHelper wifiInfra1;
163         WifiMacHelper macInfra1;
164
165         YansWifiChannelHelper wifiChannel1 = YansWifiChannelHelper::Default ();
166         wifiPhy1.SetChannel (wifiChannel1.Create ());
167         std::string ssidString1 ("wifi-infra1");
168         std::stringstream ss1;
169         ss1 << i;
170         ssidString1 += ss1.str ();
171         Ssid ssid1 = Ssid (ssidString1);
172         wifiInfra1.SetRemoteStationManager ("ns3::ArfWifiManager");

```

```

173     macInfra1.SetType ("ns3::StaWifiMac",
174                               "Ssid", SsidValue (ssid1),
175                               "ActiveProbing", BooleanValue (false));
176     NetDeviceContainer staDevices1 = wifiInfra1.Install (wifiPhy1, macInfra1,
177                               ↳ stas1);
178     macInfra1.SetType ("ns3::ApWifiMac",
179                               "Ssid", SsidValue (ssid1),
180                               "BeaconGeneration", BooleanValue (true),
181                               "BeaconInterval", TimeValue(Seconds(2.5)));
182     NetDeviceContainer apDevices1 = wifiInfra1.Install (wifiPhy1, macInfra1,
183                               ↳ PLCBackbone.Get (0));
184     NetDeviceContainer infraDevices1 (apDevices1, staDevices1);
185     internet.Install (stas1);
186     ipAddrs.Assign (infraDevices1);
187     ipAddrs.NewNetwork ();
188
189     Ptr<ListPositionAllocator> subnetAlloc =
190         CreateObject<ListPositionAllocator> ();
191     for (uint32_t j = 0; j < infra1.GetN (); ++j)
192     {
193         subnetAlloc->Add (Vector (0.0, j, 0.0));
194         mobility.PushReferenceMobilityModel (PLCBackbone.Get (0));
195         mobility.SetPositionAllocator (subnetAlloc);
196         mobility.SetMobilityModel ("ns3::RandomDirection2dMobilityModel",
197                               "Bounds", RectangleValue (Rectangle (-2, 2, -2, 2)),
198                               "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=3]"),
199                               "Pause", StringValue ("ns3::ConstantRandomVariable[Constant=0.4]"));
200         mobility.Install (stas1);
201     }
202     ipAddrs.SetBase ("10.0.0.0", "255.255.255.0");
203
204     for (uint32_t i = 0; i < 0; ++i)
205     {
206         NS_LOG_INFO ("Configuring wireless network 2 for backbone node ");
207         NodeContainer stas2;
208         stas2.Create (1);
209         NodeContainer infra2 (PLCBackbone.Get (6), stas2);
210         WifiHelper wifiInfra2;
211         WifiMacHelper macInfra2;
212
213         YansWifiChannelHelper wifiChannel2 = YansWifiChannelHelper::Default ();
214         wifiPhy1.SetChannel (wifiChannel2.Create ());
215         std::string ssidString2 ("wifi-infra2");
216         std::stringstream ss2;
217         ss2 << i;
218

```

```

219         ssidString2 += ss2.str ();
220         Ssid ssid2 = Ssid (ssidString2);
221         wifiInfra2.SetRemoteStationManager ("ns3::ArfWifiManager");
222         macInfra2.SetType ("ns3::StaWifiMac",
223 "Ssid", SsidValue (ssid2),
224 "ActiveProbing", BooleanValue (false));
225         NetDeviceContainer staDevices2 = wifiInfra2.Install (wifiPhy1, macInfra2,
226             ↳ stas2);
227         macInfra2.SetType ("ns3::ApWifiMac",
228 "Ssid", SsidValue (ssid2),
229 "BeaconGeneration", BooleanValue (true),
230 "BeaconInterval", TimeValue (Seconds(2.5)));
231         NetDeviceContainer apDevices2 = wifiInfra2.Install (wifiPhy1, macInfra2,
232             ↳ PLCBackbone.Get (6));
233         NetDeviceContainer infraDevices2 (apDevices2, staDevices2);
234
235         internet.Install (stas2);
236         ipAddrs.Assign (infraDevices2);
237         ipAddrs.NewNetwork ();
238
239         Ptr<ListPositionAllocator> subnetAlloc =
240             CreateObject<ListPositionAllocator> ();
241         for (uint32_t j = 0; j < infra2.GetN (); ++j)
242         {
243             subnetAlloc->Add (Vector (0.0, j, 0.0));
244             mobility.PushReferenceMobilityModel (PLCBackbone.Get (0));
245             mobility.SetPositionAllocator (subnetAlloc);
246             mobility.SetMobilityModel ("ns3::RandomDirection2dMobilityModel",
247 "Bounds", RectangleValue (Rectangle (-10, 10, -10, 10)),
248 "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=3]"),
249 "Pause", StringValue ("ns3::ConstantRandomVariable[Constant=0.4]"));
250             mobility.Install (stas2);
251         }
252
253
254
255
256
257 NS_LOG_INFO ("Create Applications.");
258 uint16_t port = 9; // Discard port (RFC 863)
259
260 NS_ASSERT (lanNodes > 1 && infraNodes > 1);
261 Ptr<Node> appSource = NodeList::GetNode (7);
262 Ptr<Node> appSink = NodeList::GetNode (6);
263
264 OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address ());

```

```
265 onoff1.SetAttribute("OffTime",StringValue
266   ↳ ("ns3::LogNormalRandomVariable[Mu=0.4026,Sigma=0.0352]"));
267 onoff1.SetAttribute("OnTime",StringValue
268   ↳ ("ns3::WeibullRandomVariable[Shape=10.2063,Scale=57480.9]"));
269 onoff1.SetAttribute("MaxBytes",UintegerValue(10989173));
270
271
272 ApplicationContainer apps = onoff1.Install (appSource);
273 apps.Start (Seconds (3));
274 apps.Stop (Seconds (stopTime - 1));
275
276 PacketSinkHelper sink ("ns3::UdpSocketFactory",
277   InetSocketAddress (Ipv4Address::GetAny (), port));
278 apps = sink.Install (appSink);
279 apps.Start (Seconds (3));
280
281 NS_LOG_INFO ("Configure Tracing.");
282
283 AsciiTraceHelper ascii;
284 Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("PLC-LAN-WiFi.tr");
285 wifiPhy1.EnableAsciiAll (ascii.CreateFileStream ("wifi.tr"));
286 internet.EnableAsciiIpv4All (ascii.CreateFileStream ("internet.tr"));
287 wifiPhy1.EnablePcapAll ("wifiPhyCap");
288 internet.EnablePcapIpv4All("InternetCap");
289
290 if (useCourseChangeCallback == true)
291 {
292     Config::Connect ("/ NodeList/*/$ns3::MobilityModel/CourseChange", MakeCallback
293       ↳ (&CourseChangeCallback));
294 }
295
296 AnimationInterface anim ("PLC-LAN-WiFi.xml");
297 NS_LOG_INFO ("Run Simulation.");
298 Simulator::Stop (Seconds (stopTime));
299 Simulator::Run ();
300 Simulator::Destroy ();
301 }
```

D. Anexo: Código de NS3 para la implementación de segundo escenario de simulación mixto PLC-WiFi

```
1 #include <iostream>
2 #include <string>
3 #include "ns3/core-module.h"
4 #include "ns3/network-module.h"
5 #include "ns3/applications-module.h"
6 #include "ns3/mobility-module.h"
7 #include "ns3/config-store-module.h"
8 #include "ns3/wifi-module.h"
9 #include "ns3/csma-module.h"
10 #include "ns3/olsr-helper.h"
11 #include "ns3/internet-module.h"
12 #include "ns3/netanim-module.h"
13 #include "ns3/plc.h"
14 #include "ns3/aodv-module.h"
15 #include "ns3/olsr-module.h"
16 #include "ns3/dsuv-module.h"
17 #include "ns3/dsr-module.h"
18
19 using namespace ns3;
20
21 NS_LOG_COMPONENT_DEFINE ("MixedWireless");
22
23 static void
24 CourseChangeCallback (std::string path, Ptr<const MobilityModel> model)
25 {
26     Vector position = model->GetPosition ();
27     std::cout << "CourseChange " << path << " x=" << position.x << ", y=" << position.y <<
28     " z=" << position.z << std::endl;
29 }
30
31 int
32 main (int argc, char *argv[])
33 {
34     uint32_t backboneNodes = 2;
35     uint32_t infraNodes = 2;
```

```

35  uint32_t lanNodes = 2;
36  uint32_t stopTime = 104;
37  bool useCourseChangeCallback = false;
38
39  Config::SetDefault ("ns3::OnOffApplication::PacketSize", StringValue ("64"));
40  Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("512b/s"));
41
42  CommandLine cmd;
43  cmd.AddValue ("backboneNodes", "number of backbone nodes", backboneNodes);
44  cmd.AddValue ("infraNodes", "number of leaf nodes", infraNodes);
45  cmd.AddValue ("lanNodes", "number of LAN nodes", lanNodes);
46  cmd.AddValue ("stopTime", "simulation stop time (seconds)", stopTime);
47  cmd.AddValue ("useCourseChangeCallback", "whether to enable course change tracing",
48    → useCourseChangeCallback);
49
50  cmd.Parse (argc, argv);
51
52  if (stopTime < 10)
53  {
54      std::cout << "Use a simulation stop time >= 10 seconds" << std::endl;
55      exit (1);
56  }
57
58  PLC_SpectrumModelHelper smHelper;
59  Ptr<const SpectrumModel> sm;
60  sm = smHelper.GetSpectrumModel(0, 10e6, 100);
61
62  Ptr<SpectrumValue> txPsd = Create<SpectrumValue> (sm);
63  (*txPsd) = 0.005; // -50dBm/Hz
64
65  // Ptr<PLC_Cable> cable = CreateObject<PLC_NAYY150SE_Cable> (sm);
66  Ptr<PLC_Cable> cable = CreateObject<PLC_AL3x95XLPE_Cable> (sm);
67
68  Ptr<PLC_ConstImpedance> shuntImp2 = Create<PLC_ConstImpedance> (sm,
69    → PLC_Value(50, 0));
70  Ptr<PLC_ConstImpedance> shuntImp3 = Create<PLC_ConstImpedance> (sm,
71    → PLC_Value(50, 0));
72  Ptr<PLC_ConstImpedance> shuntImp4 = Create<PLC_ConstImpedance> (sm,
73    → PLC_Value(50, 0));
74  Ptr<PLC_ConstImpedance> shuntImp5 = Create<PLC_ConstImpedance> (sm,
75    → PLC_Value(50, 0));
76  Ptr<PLC_ConstImpedance> shuntImp21 = Create<PLC_ConstImpedance> (sm,
77    → PLC_Value(50, 0));
78
79  Ptr<PLC_Node> n1 = CreateObject<PLC_Node> ();
80  Ptr<PLC_Node> n2 = CreateObject<PLC_Node> ();
81  Ptr<PLC_Node> n3 = CreateObject<PLC_Node> ();
82  Ptr<PLC_Node> n4 = CreateObject<PLC_Node> ();

```

```
77     Ptr<PLC_Node> n5 = CreateObject<PLC_Node> ();
78     Ptr<PLC_Node> n20 = CreateObject<PLC_Node> ();
79     Ptr<PLC_Node> n21 = CreateObject<PLC_Node> ();
80
81
82     n1->SetPosition(0,0,0);
83     n2->SetPosition(3,0,0);
84     n3->SetPosition(6,0,0);
85     n4->SetPosition(9,0,0);
86     n5->SetPosition(14,0,0);
87     n20->SetPosition(16,-4,0);
88     n21->SetPosition(14,-4,0);
89
90
91     n1->SetName("Node1");
92     n2->SetName("Junction2");
93     n3->SetName("Junction3");
94     n4->SetName("Junction4");
95     n5->SetName("Junction5");
96     n20->SetName("Node20");
97     n21->SetName("Node21");
98
99
100    PLC_NodeList nodes;
101    nodes.push_back(n1);
102    nodes.push_back(n2);
103    nodes.push_back(n3);
104    nodes.push_back(n4);
105    nodes.push_back(n5);
106    nodes.push_back(n20);
107    nodes.push_back(n21);
108
109    CreateObject<PLC_Line> (cable, n1, n2);
110    CreateObject<PLC_Line> (cable, n2, n3);
111    CreateObject<PLC_Line> (cable, n3, n4);
112    CreateObject<PLC_Line> (cable, n4, n5);
113    CreateObject<PLC_Line> (cable, n5, n21);
114    CreateObject<PLC_Line> (cable, n21, n20);
115
116
117    PLC_ChannelHelper channelHelper(sm);
118    channelHelper.Install(nodes);
119    Ptr<PLC_Channel> channel = channelHelper.GetChannel();
120
121
122    Ptr<PLC_Outlet> outlet1 = CreateObject<PLC_Outlet> (n2, shuntImp2);
123    Ptr<PLC_Outlet> outlet2 = CreateObject<PLC_Outlet> (n3, shuntImp3);
124    Ptr<PLC_Outlet> outlet4 = CreateObject<PLC_Outlet> (n4, shuntImp4);
```

```
125     Ptr<PLC_Outlet> outlet5 = CreateObject<PLC_Outlet> (n5, shuntImp5);
126     Ptr<PLC_Outlet> outlet21 = CreateObject<PLC_Outlet> (n21, shuntImp21);
127
128
129
130
131     PLC_NetDeviceHelper deviceHelper(sm, txPsd, nodes);
132     deviceHelper.DefinePhyType(TypeId::LookupByName ("ns3::PLC_InformationRatePhy"));
133     deviceHelper.DefineMacType(TypeId::LookupByName ("ns3::PLC_ArqMac"));
134     deviceHelper.SetHeaderModulationAndCodingScheme(ModulationAndCodingScheme(BPSK_1_4,0));
135     deviceHelper.SetPayloadModulationAndCodingScheme(ModulationAndCodingScheme(BPSK_1_2,0));
136     deviceHelper.Setup();
137
138
139
140     channel->InitTransmissionChannels();
141     channel->CalcTransmissionChannels();
142
143     NodeContainer PLCBackbone;
144     PLCBackbone=deviceHelper.GetNS3Nodes();
145
146
147     NetDeviceContainer backboneDevices;
148     backboneDevices = deviceHelper.GetNetDevices();
149
150     NS_LOG_INFO ("Enabling OLSR routing on all backbone nodes");
151
152     InternetStackHelper internet;
153     DsdvHelper dsdv;
154     internet.SetRoutingHelper (dsdv); // has effect on the next Install ()
155     internet.Install (PLCBackbone);
156     Ipv4AddressHelper ipAddrs;
157     ipAddrs.SetBase ("192.168.0.0", "255.255.255.0");
158     ipAddrs.Assign (backboneDevices);
159     MobilityHelper mobility;
160     mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
161     mobility.Install(PLCBackbone);
162     ipAddrs.SetBase ("172.16.0.0", "255.255.255.0");
163
164     YansWifiPhyHelper wifiPhy1 = YansWifiPhyHelper::Default ();
165
166     for (uint32_t i = 0; i < 1; ++i)
167     {
168         NS_LOG_INFO ("Configuring wireless network 1 for backbone node ");
169         NodeContainer stas1;
170         stas1.Create (1);
171         NodeContainer infra1 (PLCBackbone.Get (0), stas1);
172         WifiHelper wifiInfra1;
```

```

173     WifiMacHelper macInfra1;
174
175     YansWifiChannelHelper wifiChannel1 = YansWifiChannelHelper::Default ();
176     wifiPhy1.SetChannel (wifiChannel1.Create ());
177     std::string ssidString1 ("wifi-infra1");
178     std::stringstream ss1;
179     ss1 << i;
180     ssidString1 += ss1.str ();
181     Ssid ssid1 = Ssid (ssidString1);
182     wifiInfra1.SetRemoteStationManager ("ns3::ArfWifiManager");
183     macInfra1.SetType ("ns3::StaWifiMac",
184                         "Ssid", SsidValue (ssid1),
185                         "ActiveProbing", BooleanValue (false));
186     NetDeviceContainer staDevices1 = wifiInfra1.Install (wifiPhy1, macInfra1,
187               ↳ stas1);
188     macInfra1.SetType ("ns3::ApWifiMac",
189                         "Ssid", SsidValue (ssid1),
190                         "BeaconGeneration", BooleanValue (true),
191                         "BeaconInterval", TimeValue (Seconds(2.5)));
192     NetDeviceContainer apDevices1 = wifiInfra1.Install (wifiPhy1, macInfra1,
193               ↳ PLCBackbone.Get (0));
194     NetDeviceContainer infraDevices1 (apDevices1, staDevices1);
195     internet.Install (stas1);
196     ipAddrs.Assign (infraDevices1);
197     ipAddrs.NewNetwork ();
198
199     Ptr<ListPositionAllocator> subnetAlloc =
200         CreateObject<ListPositionAllocator> ();
201     for (uint32_t j = 0; j < infra1.GetN (); ++j)
202     {
203         subnetAlloc->Add (Vector (0.0, j, 0.0));
204         mobility.PushReferenceMobilityModel (PLCBackbone.Get (0));
205         mobility.SetPositionAllocator (subnetAlloc);
206         mobility.SetMobilityModel ("ns3::RandomDirection2dMobilityModel",
207             "Bounds", RectangleValue (Rectangle (-2, 2, -2, 2)),
208             "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=3]"),
209             "Pause", StringValue ("ns3::ConstantRandomVariable[Constant=0.4]"));
210         mobility.Install (stas1);
211     }
212
213     ipAddrs.SetBase ("10.0.0.0", "255.255.255.0");
214
215     for (uint32_t i = 0; i < 1; ++i)
216     {
217         NS_LOG_INFO ("Configuring wireless network 2 for backbone node ");

```

```

219     NodeContainer stas2;
220     stas2.Create (1);
221     NodeContainer infra2 (PLCBackbone.Get (6), stas2);
222     WifiHelper wifiInfra2;
223     WifiMacHelper macInfra2;
224
225     YansWifiChannelHelper wifiChannel2 = YansWifiChannelHelper::Default ();
226     wifiPhy1.SetChannel (wifiChannel2.Create ());
227     // Create unique ssids for these networks
228     std::string ssidString2 ("wifi-infra2");
229     std::stringstream ss2;
230     ss2 << i;
231     ssidString2 += ss2.str ();
232     Ssid ssid2 = Ssid (ssidString2);
233     wifiInfra2.SetRemoteStationManager ("ns3::ArfWifiManager");
234     // setup stas
235     macInfra2.SetType ("ns3::StaWifiMac",
236 "Ssid", SsidValue (ssid2),
237 "ActiveProbing", BooleanValue (false));
238     NetDeviceContainer staDevices2 = wifiInfra2.Install (wifiPhy1, macInfra2,
239             ↪ stas2);
240     macInfra2.SetType ("ns3::ApWifiMac",
241 "Ssid", SsidValue (ssid2),
242 "BeaconGeneration", BooleanValue (true),
243 "BeaconInterval", TimeValue (Seconds(2.5)));
244     NetDeviceContainer apDevices2 = wifiInfra2.Install (wifiPhy1, macInfra2,
245             ↪ PLCBackbone.Get (6));
246     NetDeviceContainer infraDevices2 (apDevices2, staDevices2);
247
248     internet.Install (stas2);
249     ipAddrs.Assign (infraDevices2);
250     ipAddrs.NewNetwork ();
251
252     Ptr<ListPositionAllocator> subnetAlloc =
253         CreateObject<ListPositionAllocator> ();
254     for (uint32_t j = 0; j < infra2.GetN (); ++j)
255     {
256         subnetAlloc->Add (Vector (0.0, j, 0.0));
257         mobility.PushReferenceMobilityModel (PLCBackbone.Get (0));
258         mobility.SetPositionAllocator (subnetAlloc);
259         mobility.SetMobilityModel ("ns3::RandomDirection2dMobilityModel",
260             "Bounds", RectangleValue (Rectangle (-2, 2, -2, 2)),
261             "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=3]"),
262             "Pause", StringValue ("ns3::ConstantRandomVariable[Constant=0.4]"));
263         mobility.Install (stas2);
264     }

```

```

265
266
267
268     NS_LOG_INFO ("Create Applications.");
269     uint16_t port = 9;    // Discard port (RFC 863)
270
271     NS_ASSERT (lanNodes > 1 && infraNodes > 1);
272     Ptr<Node> appSource = NodeList::GetNode (7);
273     Ptr<Node> appSink = NodeList::GetNode (8);
274
275     OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address ());
276     onoff1.SetAttribute("OffTime", StringValue
277         ↳ ("ns3::LogNormalRandomVariable[Mu=0.4026,Sigma=0.0352]"));
278     onoff1.SetAttribute("OnTime", StringValue
279         ↳ ("ns3::WeibullRandomVariable[Shape=10.2063,Scale=57480.9]"));
280     onoff1.SetAttribute("MaxBytes", UintegerValue(10989173));
281
282     ApplicationContainer apps = onoff1.Install (appSource);
283     apps.Start (Seconds (3));
284     apps.Stop (Seconds (stopTime - 1));
285
286     PacketSinkHelper sink ("ns3::UdpSocketFactory",
287         InetSocketAddress (Ipv4Address::GetAny (), port));
288     apps = sink.Install (appSink);
289     apps.Start (Seconds (3));
290
291     NS_LOG_INFO ("Configure Tracing.");
292
293     AsciiTraceHelper ascii;
294     Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("PLC-LAN-WiFi.tr");
295     wifiPhy1.EnableAsciiAll (ascii.CreateFileStream ("wifi.tr"));
296     internet.EnableAsciiIpv4All (ascii.CreateFileStream ("internet.tr"));
297
298     wifiPhy1.EnablePcapAll ("wifiPhyCap");
299     internet.EnablePcapIpv4All ("InternetCap");
300
301     if (useCourseChangeCallback == true)
302     {
303         Config::Connect ("/ NodeList /*/$ ns3::MobilityModel/CourseChange", MakeCallback
304             ↳ (&CourseChangeCallback));
305     }
306
307     AnimationInterface anim ("PLC-LAN-WiFi.xml");
308
309     NS_LOG_INFO ("Run Simulation.");
310     Simulator::Stop (Seconds (stopTime));
311     Simulator::Run ();

```

D Anexo: Código de NS3 para la implementación de segundo escenario de simulación
90 mixto PLC-WiFi

```
310     Simulator::Destroy ();  
311 }
```

E. Anexo: Código de NS3 para la implementación de tercer escenario de simulación mixto PLC-WiFi

```
1 #include <iostream>
2 #include <string>
3 #include "ns3/core-module.h"
4 #include "ns3/network-module.h"
5 #include "ns3/applications-module.h"
6 #include "ns3/mobility-module.h"
7 #include "ns3/config-store-module.h"
8 #include "ns3/wifi-module.h"
9 #include "ns3/csma-module.h"
10 #include "ns3/olsr-helper.h"
11 #include "ns3/internet-module.h"
12 #include "ns3/netanim-module.h"
13 #include "ns3/plc.h"
14 #include "ns3/aodv-module.h"
15 #include "ns3/olsr-module.h"
16 #include "ns3/dsuv-module.h"
17 #include "ns3/dsr-module.h"
18
19 using namespace ns3;
20
21 NS_LOG_COMPONENT_DEFINE ("MixedWireless");
22
23 static void
24 CourseChangeCallback (std::string path, Ptr<const MobilityModel> model)
25 {
26     Vector position = model->GetPosition ();
27     std::cout << "CourseChange " << path << " x=" << position.x << ", y=" << position.y <<
28     " z=" << position.z << std::endl;
29 }
30
31 int
32 main (int argc, char *argv[])
33 {
34     uint32_t backboneNodes = 2;
35     uint32_t infraNodes = 2;
```

```
35 uint32_t lanNodes = 2;
36 uint32_t stopTime = 104;
37 bool useCourseChangeCallback = false;
38
39 Config::SetDefault ("ns3::OnOffApplication::PacketSize", StringValue ("64"));
40 Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("512b/s"));
41
42 CommandLine cmd;
43 cmd.AddValue ("backboneNodes", "number of backbone nodes", backboneNodes);
44 cmd.AddValue ("infraNodes", "number of leaf nodes", infraNodes);
45 cmd.AddValue ("lanNodes", "number of LAN nodes", lanNodes);
46 cmd.AddValue ("stopTime", "simulation stop time (seconds)", stopTime);
47 cmd.AddValue ("useCourseChangeCallback", "whether to enable course change tracing",
   ↳ useCourseChangeCallback);
48
49 cmd.Parse (argc, argv);
50
51 if (stopTime < 10)
52 {
53     std::cout << "Use a simulation stop time >= 10 seconds" << std::endl;
54     exit (1);
55 }
56     PLC_SpectrumModelHelper smHelper;
57     Ptr<const SpectrumModel> sm;
58     sm = smHelper.GetSpectrumModel(0, 10e6, 100);
59
60     Ptr<SpectrumValue> txPsd = Create<SpectrumValue> (sm);
61     (*txPsd) = 0.00005;
62
63     Ptr<PLC_Cable> cable = CreateObject<PLC_AL3x95XLPE_Cable> (sm);
64
65     Ptr<PLC_ConstImpedance> shuntImp2 = Create<PLC_ConstImpedance> (sm,
   ↳ PLC_Value(50, 0));
66     Ptr<PLC_ConstImpedance> shuntImp3 = Create<PLC_ConstImpedance> (sm,
   ↳ PLC_Value(50, 0));
67     Ptr<PLC_ConstImpedance> shuntImp4 = Create<PLC_ConstImpedance> (sm,
   ↳ PLC_Value(50, 0));
68     Ptr<PLC_ConstImpedance> shuntImp5 = Create<PLC_ConstImpedance> (sm,
   ↳ PLC_Value(50, 0));
69     Ptr<PLC_ConstImpedance> shuntImp21 = Create<PLC_ConstImpedance> (sm,
   ↳ PLC_Value(50, 0));
70
71     Ptr<PLC_Node> n1 = CreateObject<PLC_Node> ();
72     Ptr<PLC_Node> n2 = CreateObject<PLC_Node> ();
73     Ptr<PLC_Node> n3 = CreateObject<PLC_Node> ();
74     Ptr<PLC_Node> n4 = CreateObject<PLC_Node> ();
75     Ptr<PLC_Node> n5 = CreateObject<PLC_Node> ();
76     Ptr<PLC_Node> n20 = CreateObject<PLC_Node> ();
```

```
77     Ptr<PLC_Node> n21 = CreateObject<PLC_Node> ();
78
79
80     n1->SetPosition(0,0,0);
81     n2->SetPosition(3,0,0);
82     n3->SetPosition(6,0,0);
83     n4->SetPosition(9,0,0);
84     n5->SetPosition(14,0,0);
85     n20->SetPosition(16,-4,0);
86     n21->SetPosition(14,-4,0);
87
88
89     n1->SetName("Node1");
90     n2->SetName("Junction2");
91     n3->SetName("Junction3");
92     n4->SetName("Junction4");
93     n5->SetName("Junction5");
94     n20->SetName("Node20");
95     n21->SetName("Node21");
96
97
98     PLC_NodeList nodes;
99     nodes.push_back(n1);
100    nodes.push_back(n2);
101    nodes.push_back(n3);
102    nodes.push_back(n4);
103    nodes.push_back(n5);
104    nodes.push_back(n20);
105    nodes.push_back(n21);
106
107    CreateObject<PLC_Line> (cable, n1, n2);
108    CreateObject<PLC_Line> (cable, n2, n3);
109    CreateObject<PLC_Line> (cable, n3, n4);
110    CreateObject<PLC_Line> (cable, n4, n5);
111    CreateObject<PLC_Line> (cable, n5, n21);
112    CreateObject<PLC_Line> (cable, n21, n20);
113
114
115     PLC_ChannelHelper channelHelper(sm);
116     channelHelper.Install(nodes);
117     Ptr<PLC_Channel> channel = channelHelper.GetChannel();
118
119
120     Ptr<PLC_Outlet> outlet1 = CreateObject<PLC_Outlet> (n2, shuntImp2);
121     Ptr<PLC_Outlet> outlet2 = CreateObject<PLC_Outlet> (n3, shuntImp3);
122     Ptr<PLC_Outlet> outlet4 = CreateObject<PLC_Outlet> (n4, shuntImp4);
123     Ptr<PLC_Outlet> outlet5 = CreateObject<PLC_Outlet> (n5, shuntImp5);
124     Ptr<PLC_Outlet> outlet21 = CreateObject<PLC_Outlet> (n21, shuntImp21);
```

```
125
126
127
128     PLC_NetDeviceHelper deviceHelper(sm, txPsd, nodes);
129     deviceHelper.DefinePhyType(TypeId::LookupByName ("ns3::PLC_InformationRatePhy"));
130     deviceHelper.DefineMacType(TypeId::LookupByName ("ns3::PLC_ArqMac"));
131     deviceHelper.SetHeaderModulationAndCodingScheme(ModulationAndCodingScheme(BPSK_1_4,0));
132     deviceHelper.SetPayloadModulationAndCodingScheme(ModulationAndCodingScheme(BPSK_1_2,0));
133     deviceHelper.Setup();
134
135
136
137     channel->InitTransmissionChannels();
138     channel->CalcTransmissionChannels();
139
140     NodeContainer PLCBackbone;
141     PLCBackbone=deviceHelper.GetNS3Nodes();
142
143
144     NetDeviceContainer backboneDevices;
145     backboneDevices = deviceHelper.GetNetDevices();
146
147 NS_LOG_INFO ("Enabling OLSR routing on all backbone nodes");
148
149 InternetStackHelper internet;
150   DsdvHelper dsdv;
151   internet.SetRoutingHelper (dsdv);
152   internet.Install (PLCBackbone);
153
154 Ipv4AddressHelper ipAddrs;
155   ipAddrs.SetBase ("192.168.0.0", "255.255.255.0");
156   ipAddrs.Assign (backboneDevices);
157 MobilityHelper mobility;
158   mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
159   mobility.Install(PLCBackbone);
160
161 ipAddrs.SetBase ("172.16.0.0", "255.255.255.0");
162
163 YansWifiPhyHelper wifiPhy1 = YansWifiPhyHelper::Default ();
164
165 for (uint32_t i = 0; i < 1; ++i)
166 {
167   NS_LOG_INFO ("Configuring wireless network 1 for backbone node ");
168   NodeContainer stas1;
169   stas1.Create (1);
170   NodeContainer infra1 (PLCBackbone.Get (0), stas1);
171   WifiHelper wifiInfra1;
172   WifiMacHelper macInfra1;
```

```

173
174     YansWifiChannelHelper wifiChannel1 = YansWifiChannelHelper::Default ();
175     wifiPhy1.SetChannel (wifiChannel1.Create ());
176     std::string ssidString1 ("wifi-infra1");
177     std::stringstream ss1;
178     ss1 << i;
179     ssidString1 += ss1.str ();
180     Ssid ssid1 = Ssid (ssidString1);
181     wifiInfra1.SetRemoteStationManager ("ns3::ArfWifiManager");
182     macInfra1.SetType ("ns3::StaWifiMac",
183                         "Ssid", SsidValue (ssid1),
184                         "ActiveProbing", BooleanValue (false));
185     NetDeviceContainer staDevices1 = wifiInfra1.Install (wifiPhy1, macInfra1,
186     ↵   stas1);
186     macInfra1.SetType ("ns3::ApWifiMac",
187     "Ssid", SsidValue (ssid1),
188     "BeaconGeneration", BooleanValue (true),
189     "BeaconInterval", TimeValue (Seconds(2.5)));
190     NetDeviceContainer apDevices1 = wifiInfra1.Install (wifiPhy1, macInfra1,
191     ↵   PLCBackbone.Get (0));
191     NetDeviceContainer infraDevices1 (apDevices1, staDevices1);
192
193     internet.Install (stas1);
194     ipAddrs.Assign (infraDevices1);
195     ipAddrs.NewNetwork ();
196
197
198     Ptr<ListPositionAllocator> subnetAlloc =
199     CreateObject<ListPositionAllocator> ();
200     for (uint32_t j = 0; j < infra1.GetN (); ++j)
201     {
202         subnetAlloc->Add (Vector (0.0, j, 0.0));
203     }
204     mobility.PushReferenceMobilityModel (PLCBackbone.Get (0));
205     mobility.SetPositionAllocator (subnetAlloc);
206     mobility.SetMobilityModel ("ns3::RandomDirection2dMobilityModel",
207     "Bounds", RectangleValue (Rectangle (-10, 10, -10, 10)),
208     "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=3]"),
209     "Pause", StringValue ("ns3::ConstantRandomVariable[Constant=0.4]"));
210     mobility.Install (stas1);
211 }
212
213     ipAddrs.SetBase ("10.0.0.0", "255.255.255.0");
214
215
216     for (uint32_t i = 0; i < 1; ++i)
217     {
218         NS_LOG_INFO ("Configuring wireless network 2 for backbone node ");

```

```
219     NodeContainer stas2;
220     stas2.Create (1);
221     NodeContainer infra2 (PLCBackbone.Get (6), stas2);
222     WifiHelper wifiInfra2;
223     WifiMacHelper macInfra2;
224
225     YansWifiChannelHelper wifiChannel2 = YansWifiChannelHelper::Default ();
226     wifiPhy1.SetChannel (wifiChannel2.Create ());
227     std::string ssidString2 ("wifi-infra2");
228     std::stringstream ss2;
229     ss2 << i;
230     ssidString2 += ss2.str ();
231     Ssid ssid2 = Ssid (ssidString2);
232     wifiInfra2.SetRemoteStationManager ("ns3::ArfWifiManager");
233     macInfra2.SetType ("ns3::StaWifiMac",
234 "Ssid", SsidValue (ssid2),
235 "ActiveProbing", BooleanValue (false));
236     NetDeviceContainer staDevices2 = wifiInfra2.Install (wifiPhy1, macInfra2,
237             ↳ stas2);
238     macInfra2.SetType ("ns3::ApWifiMac",
239 "Ssid", SsidValue (ssid2),
240 "BeaconGeneration", BooleanValue (true),
241 "BeaconInterval", TimeValue(Seconds(2.5)));
242     NetDeviceContainer apDevices2 = wifiInfra2.Install (wifiPhy1, macInfra2,
243             ↳ PLCBackbone.Get (6));
244     NetDeviceContainer infraDevices2 (apDevices2, staDevices2);
245
246     internet.Install (stas2);
247     ipAddrs.Assign (infraDevices2);
248     ipAddrs.NewNetwork ();
249
250     Ptr<ListPositionAllocator> subnetAlloc =
251         CreateObject<ListPositionAllocator> ();
252     for (uint32_t j = 0; j < infra2.GetN (); ++j)
253     {
254         subnetAlloc->Add (Vector (0.0, j, 0.0));
255     }
256     mobility.PushReferenceMobilityModel (PLCBackbone.Get (0));
257     mobility.SetPositionAllocator (subnetAlloc);
258     mobility.SetMobilityModel ("ns3::RandomDirection2dMobilityModel",
259 "Bounds", RectangleValue (Rectangle (-10, 10, -10, 10)),
260 "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=3]"),
261 "Pause", StringValue ("ns3::ConstantRandomVariable[Constant=0.4]"));
262     mobility.Install (stas2);
263 }
264 }
```

```

265
266
267 NS_LOG_INFO ("Create Applications.");
268 uint16_t port = 9;
269 NS_ASSERT (lanNodes > 1 && infraNodes > 1);
270 Ptr<Node> appSource = NodeList::GetNode (7);
271 Ptr<Node> appSink = NodeList::GetNode (8);
272
273 OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
274 onoff1.SetAttribute("OffTime",StringValue
275   ↳ ("ns3::LogNormalRandomVariable[Mu=0.4026,Sigma=0.0352]"));
276 onoff1.SetAttribute("OnTime",StringValue
277   ↳ ("ns3::WeibullRandomVariable[Shape=10.2063,Scale=57480.9]"));
278 onoff1.SetAttribute("MaxBytes",UintegerValue(10989173));
279
280
281 ApplicationContainer apps = onoff1.Install (appSource);
282 apps.Start (Seconds (3));
283 apps.Stop (Seconds (stopTime - 1));
284
285 PacketSinkHelper sink ("ns3::UdpSocketFactory",
286   InetSocketAddress (Ipv4Address::GetAny (), port));
287 apps = sink.Install (appSink);
288 apps.Start (Seconds (3));
289
290 NS_LOG_INFO ("Configure Tracing.");
291 AsciiTraceHelper ascii;
292 Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("PLC-LAN-WiFi.tr");
293 wifiPhy1.EnableAsciiAll (ascii.CreateFileStream ("wifi.tr"));
294 internet.EnableAsciiIpv4All (ascii.CreateFileStream ("internet.tr"));
295 wifiPhy1.EnablePcapAll ("wifiPhyCap");
296 internet.EnablePcapIpv4All("InternetCap");
297
298 if (useCourseChangeCallback == true)
299 {
300   Config::Connect ("/ NodeList/*/$ns3::MobilityModel/CourseChange", MakeCallback
301     ↳ (&CourseChangeCallback));
302 }
303
304 AnimationInterface anim ("PLC-LAN-WiFi.xml");
305
306 NS_LOG_INFO ("Run Simulation.");
307 Simulator::Stop (Seconds (stopTime));
308 Simulator::Run ();
309 Simulator::Destroy ();
310 }
```

Bibliografía

[AlterMundi,] AlterMundi. La subida histórica de las redes comunitarias — Blog de Alter-Mundi.

[Anatory et al., 2004] Anatory, J., Mvungi, N., and Kissaka, M. (2004). Trends in telecommunication services provision: powerline network can provide alternative for access in developing countries. *In Proceedings of the 7th AFRICON Conference in Africa. ISSN 0-7083-8605.*, pages pp. 601–606.

[ANDRÉS PARRA LEÓN, ELKIN M. PIEDRAHITA, 2011] ANDRÉS PARRA LEÓN, ELKIN M. PIEDRAHITA, O. S. (2011). Aplicaciones del modelo On / Of f al tráfico agregado en las redes de comunicaciones. pages 129–147.

[Antoine Varet, 2014] Antoine Varet, N. L. (2014). How to generate realistic network traffic? *IEEE COMPSAC 2014, 38th Annual International Computers, Software.*

[Barbeau and Kranakis, 2007] Barbeau, M. and Kranakis, E. (2007). *Principles of Ad Hoc Networking*. John Wiley & Sons, Ltd, Chichester, UK.

[Basagni et al., 2004] Basagni, S., Conti, M., Giordano, S., and Stojmenovic, I. (2004). *Mobile Ad Hoc Networking*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

[Butler, 2013] Butler, J. (2013). Wireless networking in the developing world [a practical guide to planning and building low-cost telecommunications infrastructure.

[Campo-muñoz et al., 2017] Campo-muñoz, W. Y., Astaiza-hoyos, E., and Muñoz-sanabria, L. F. (2017). Traffic modelling of the video-on-demand service through NS-3 • Modelado de tráfico del servicio de video bajo demanda mediante NS-3. 84(202):55–64.

[Dib et al., 2018] Dib, L. D. M., Fernandes, V., Filomeno, M. D. L., and Ribeiro, M. V. (2018). Hybrid PLC/Wireless Communication for Smart Grids and Internet of Things Applications. *IEEE Internet of Things Journal*, 5(2).

[Fariba Aalamifar,] Fariba Aalamifar, Alexander Schlogl, D. H. L. L. NS3PLCsoftware.

[Fariba Aalamifar, 2013] Fariba Aalamifar, Alexander Schlogl, D. H. L. L. (2013). Modelling power line communication using network simulator-3. *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 2969–2974.

- [Ferreira et al., 2010] Ferreira, H. C., Lampe, L., Newbury, J., and Swart, T. G. (2010). *Power Line Communications: Theory and Applications for Narrowband and Broadband Communications over Power Lines*.
- [Goralski, 2002] Goralski, W. J. (2002). *Juniper and Cisco routing : policy and protocols for multivendor IP networks*. John Wiley, New York, NY.
- [Group et al., 2006] Group, F., Middleton, B., Golden, P., Dedieu, H., Keyes, J., Blackley, J. A., and Gold, T. (2006). *Fundamental of DSL Technology*.
- [Guifi.net,] Guifi.net. guifi.net - Red de telecomunicaciones abierta, libre y neutral.
- [Guifi.net, 2017] Guifi.net (2017). ¿Qué es guifi.net? — guifi.net.
- [He et al., 2004] He, H., Cheng, S., Zhang, Y., and Nguimbis, J. (2004). Analysis of Reflection of Signal Transmitted in Low-Voltage Powerline with Complex Wavelet. *IEEE Transactions on Power Delivery*, 19(1):86–91.
- [Hekmat, 2006] Hekmat, R. (2006). *Ad-hoc Networks: Fundamental Properties and Network Topologies*.
- [Ilyas, 2003] Ilyas, M. (2003). *Wireless Networks the Handbook of Wireless*.
- [J. Anatory and N. Theethayi, 2008] J. Anatory and N. Theethayi (2008). *Broadband Power-line Communication Systems*.
- [Johnson et al., 2007] Johnson, D., Matthee, K., Sokoya, D., and Mboweni, L. (2007). Building a Rural Wireless Mesh Network: A do-it-yourself guide to planning and building a Freifunk based mesh network. *Meraka Institutue*, (October).
- [Laneman et al., 2004] Laneman, J. N., Tse, D. N., and Wornell, G. W. (2004). Cooperative diversity in wireless networks: Efficient protocols and outage behavior. *IEEE Transactions on Information Theory*, 50(12):3062–3080.
- [Leung, 2008] Leung, K. (2008). *Wireless mesh networks: architectures and protocols*. New York London.
- [Lutz Lampe, 2016] Lutz Lampe, A. M. T. a. G. S. (2016). *Power Line Communications Principles , Standards and Applications From Multimedia to Smart Grids*.
- [Meng et al., 2004] Meng, H., Chen, S., Guan, Y. L., Law, C. L., So, P. L., Gunawan, E., and Lie, T. T. (2004). Modeling of transfer characteristics for the broadband power line communication channel. *IEEE Transactions on Power Delivery*, 19(3):1057–1064.

- [Mengi et al., 2016] Mengi, A., Kortebi, A., Lucht, H., Brzozowski, M., Koch, M., Bouchet, O., and Maye, O. (2016). IEEE 1905.1 hybrid home networking standard and its implementation with PLC, Wi-Fi and Ethernet technologies. *2016 International Symposium on Power Line Communications and its Applications, ISPLC 2016*, pages 162–166.
- [Nosratinia et al., 2004] Nosratinia, A., Hunter, T. E., and Hedayat, A. (2004). Cooperative communication in wireless networks. *IEEE Communications Magazine*, 42(10):74–80.
- [Orfanidis, 2008] Orfanidis, S. J. (2008). Waves and Antennas Electromagnetic. *Media*, 2:525–570.
- [Sarkar et al., 2016] Sarkar, S. K., Basavaraju, T., and Puttamadappa, C. (2016). *Ad Hoc Mobile Wireless Networks*, volume 31. CRC Press.
- [Zarate-Ceballos, 2021] Zarate-Ceballos, Henry; Parra-Amaris, J. J.-J. H. R.-R. D. A.-R. O. O.-T. J. (2021). *WIRELESS NETWORK SIMULATION: A Guide using Ad Hoc Networks and the ns-3 Simulator*. APRESS, S.l.
- [Zhang et al., 2007] Zhang, Y., Luo, J., Hu, H., and Zhang, Y. (2007). *Wireless Mesh Networking: Architectures, Protocols and Standards*.
- [Zimmermann and Dostert, 2002] Zimmermann, M. and Dostert, K. (2002). A multipath model for the powerline channel. *IEEE Transactions on Communications*, 50(4):553–559.