



Graduado en Tecnologías y Servicios de Telecomunicación

Universidad a distancia de Madrid

Departamento de Departamento de Ingeniería de
Telecomunicaciones

PROYECTO FIN DE GRADO

Estudio de prestaciones de un entorno de IoT mediante herramientas de simulación de software libre

Autor:Miguel Angel Barrero Díaz

Director:Isaac Seoane Pujol

MADRID, JUNIO DE 2019

RESUMEN

Las redes IoT están suponiendo un cambio de paradigma en la manera de entender el mundo, la ingente cantidad de datos capturados por los pequeños dispositivos que conforman sus redes y que habitualmente se tratan en la nube está impulsando el desarrollo de nuevas soluciones aplicables a multitud de campos, desde los cuidados de la salud, a la agricultura, pasando por la logística y un largo etcétera, siendo por tanto tecnologías aplicables a todos los sectores productivos.

El hecho de que las redes que conforman el Internet de la Cosas se componen principalmente de dispositivos inalámbricos hace que el medio de transmisión sea el aire, es decir, generalmente se emplean comunicaciones por radio, lo que supone un gran reto a la hora de saber como se comportará el diseño previo en condiciones reales de funcionamiento debido a los entornos heterogéneos donde se implementan, la saturación del espectro y a que en general se utilizan para la comunicación bandas no licenciadas donde todas las tecnologías compiten por los mismos recursos, con lo que deben ser muy eficientes en su empleo y utilizar herramientas que permitan combatir la interferencias provocadas por las diferentes redes y tecnologías que hacen uso de estas bandas.

Para solucionar en la medida de lo posible esta problemática se utilizan simuladores con los cuáles, aplicando los parámetros adecuados, es posible conocer la respuesta de una red en condiciones próximas a la realidad, permitiendo a los investigadores adoptar soluciones en la fase de diseño ahorrando costes.

El presente trabajo pretende hacer un recorrido por las tecnologías existentes, los protocolos y los tipos de redes empleados en despliegues IoT, explicar el funcionamiento de uno de los simuladores de red más usados en investigación como es el caso de NS-3 e implementar una solución concreta probando así las prestaciones del simulador en un caso de estudio real.

ABSTRACT

Internet of Things (IoT) networks are causing a paradigm shift in the way we understand the world. There is a huge amount of data captured by the small devices that make up their networks and that are usually processed in the cloud. This, in turn, is driving the development of new solutions with applications in many different fields, from healthcare to agriculture, through logistics and many more. Therefore, these technologies are applicable to all productive sectors.

The networks that make up the IoT are composed mainly of wireless devices, meaning that the transmission medium is air. Radio communications are generally used. This represents a great challenge when trying to know how the preliminary design will behave in real operating conditions due to the heterogeneous environments where they are implemented, the spectrum saturation, and the fact that, generally, unlicensed bands are used for communication. As all technologies compete for the same resources, they must be very efficient in their use, adopting tools to combat the interference caused by the different networks and technologies that make use of these bands.

In order to minimise this issue, simulators are used. Applying the appropriate parameters, it is possible to know how a network would respond in conditions which are close to reality. This, in turn, allows researchers to adopt solutions in the design phase, saving costs.

This paper aims to survey the existing technologies, protocols and types of networks used in IoT deployments, explain the operation of one of the most used network simulators in research such as NS-3 and implement a specific solution, thus testing the simulator's performance in a real case study.

Índice

1.INTRODUCCIÓN Y OBJETIVOS	1
1.1.¿QUÉ ES LA INTERNET DE LAS COSAS?.....	1
1.2. MOTIVACIÓN.....	2
1.3.OBJETIVOS.....	3
2.ESTADO DEL ARTE EN EL ÁMBITO DE LA IOT.....	4
2.1. CAMPOS DE APLICACIÓN ACTUALES.....	4
2.1.1.AGRICULTURA.....	5
2.1.2.COMERCIO MINORISTA.....	5
2.1.3.CUIDADOS DE SALUD.....	6
2.1.4.INDUSTRIA MANUFACTURERA.....	7
2.1.5.CIUDADES INTELIGENTES.....	8
2.1.6 .EDIFICIOS INTELIGENTES.....	8
2.1.7.ENERGÍA.....	9
2.1.8.TRANSPORTE.....	10
2.1.9.PREVENCIÓN DE DESASTRES Y MONITOREO AMBIENTAL....	11
2.2.REDES Y TECNOLOGÍAS.....	12
2.2.1.WBAN.....	12
2.2.1.1.IEEE 802.15.6-2012.....	13
2.2.2 WPAN.....	14
2.2.2.1.RFID (Radio Frequency Identification).....	14
2.2.2.2. NFC.....	15
2.2.2.3.BLUETOOTH.....	16

2.2.2.4.ZIGBEE.....	18
2.2.2.5.LoWPAN.....	20
2.2.2.6.THREAD.....	22
2.2.2.7.Z-WAVE.....	23
2.2.2.8.RuBee.....	26
2.2.2.9.UWB.....	26
2.2.3. WLAN.....	26
2.2.3.1.WIFI.....	26
2.2.3.2.WAVE (Wireless Access in Vehicular Environments).....	29
2.2.3.3.HiperLAN2.....	30
2.2.4 WMAN.....	30
2.2.4.1.WIMAX.....	31
2.2.4.2.LMDS.....	32
2.2.4.3.WiBro.....	33
2.2.5.WRAN.....	33
2.2.5.1.IEEE 802.22.....	33
2.2.6 WWAN.....	34
2.2.6.1.SigFox.....	34
2.2.6.2.TECNOLOGÍAS MÓVILES.....	35
2.2.6.3.INGENU RPMA (Random Phase Multiple Access).....	38
2.2.6.4.LoRa y LoRaWan.....	38
2.2.6.5.WEIGHTLESS.....	40
2.2.6.6.TELENSA.....	40

2.2.6.7.WAVIoT.....	42
3.SIMULADOR NS3.....	43
3.1 INTRODUCCIÓN.....	43
3.2 FUNCIONAMIENTO DE NS-3.....	45
3.3 PREPARACIÓN DEL ENTORNO DE TRABAJO.....	55
3.3.1.INSTALACIÓN DE REQUISITOS.....	56
3.3.2.INSTALACIÓN DE NS-3.....	57
4.DESPLIEGUE DE UNA RED.....	61
4.1.INTRODUCCIÓN.....	61
4.2.ELECCIÓN DE LA TECNOLOGÍA.....	61
4.3.ARQUITECTURA DE LA RED.....	63
4.3.1.TEPOLOGÍA.....	63
4.3.2.PROTOCOLOS.....	64
5.SIMULACIÓN DEL DISEÑO EN NS-3 Y OBTENCIÓN DE RESULTADO.....	69
5.1.INTRODUCCIÓN.....	69
5.2.EL PROGRAMA.....	70
5.3.PRUEBA DE SIMULACIÓN.....	85
6.CONCLUSIONES.....	91
7.BIBLIOGRAFÍA.....	93
ANEXOS. TABLA DE REFERENCIA DE TECNOLOGÍAS.....	104
ÍNDICE DE ILUSTRACIONES	
ÍNDICE DE TABLAS	

Índice de ilustraciones

Ilustración 1: Modelo de red IoUT [16].....	12
Ilustración 2: Aplicaciones de redes WBAN [17].....	13
Ilustración 3: Pila de protocolos de Bluetooth versión 5 [21].....	16
Ilustración 4: Características de la capa física Bluetooth 5 [22].....	17
Ilustración 5: Topologías definidas en el estándar IEEE 802.15.4 [23].....	18
Ilustración 6: Pila de protocolos 6LoWPAN y modelo OSI [25].....	19
Ilustración 7: Modelo de interconexión de red 6LoWPAN[25].....	20
Ilustración 8: Pila de protocolos THREAD [29].....	21
Ilustración 9: Pila de protocolos de tecnología Z-wave [30].....	23
Ilustración 10: Pila de protocolos NFC [35].....	25
Ilustración 11: Pila de protocolos WAVE [43].....	29
Ilustración 12: Características de los diferentes estándares 802.16 [51].....	31
Ilustración 13: Arquitectura de una red WiMAX [51].....	32
Ilustración 14: Releases actualmente en desarrollo [56].....	37
Ilustración 15: Compartativa de redes 4G y 5G [57].....	37
Ilustración 16: Pila de protolos LoRaWan [59].....	39
Ilustración 17: Tecnologías LPWAN [60].....	42
Ilustración 18: Compendio pila de protocolos [61].....	42
Ilustración 19: Ejemplo de plantilla.....	47
Ilustración 20: Directivas Include en un código NS-3 típico.....	47
Ilustración 21: Ejemplo de definición de clases y funciones en programa ns-3...50	

Ilustración 22: Declaración de clase MyObject que hereda de la clase Object.....	51
Ilustración 23: Función que realiza las tareas de trace sink.....	51
Ilustración 24: Conexión entre trace source y trace sink.....	52
Ilustración 25: Salida de ejecución del programa.....	52
Ilustración 26: Declaración de función para realizar callback.....	53
Ilustración 27: Empleo de Config::Connect para conectar con fuente de rastreo..	53
Ilustración 28: Ejecución de programa con rastreo de movilidad.....	54
Ilustración 29: Comandos admitidos por el programa waf.....	54
Ilustración 30: Ejecución de programa con opción PrintHelp.....	55
Ilustración 31: Ejecución de comando which.....	56
Ilustración 32: Versiones de paquetes instalados.....	52
Ilustración 33: Contenido de carpeta ns-3-allinone.....	57
Ilustración 34: Utilidad de opción --build-profile.....	58
Ilustración 35: Utilidad de las opciones --enable-test y --enable-examples.....	58
Ilustración 36: Configuración de repositorios NetBeans.....	59
Ilustración 37:Dispositivo 6LoWPAN con antena integrada de reducidas dimensiones [64].....	62
Ilustración 38: 6LoWPAN gateway [65].....	63
Ilustración 39: Estructura de la trama 802.15.4[19].....	65
Ilustración 40: Fragmentación 6LoWPAN[66].....	65
Ilustración 41: Cabecera IPv6 [67].....	66
Ilustración 42: Representación del escenario propuesto para el despliegue.....	68

Ilustración 43: Directivas include.....	70
Ilustración 44: Implementación funciones de tracing.....	71
Ilustración 45: Variables de línea de comandos.....	72
Ilustración 46: Opciones de ejecución.....	72
Ilustración 47: Condicional verbose.....	73
Ilustración 48: Condicional para no asignar 0 nodos y configuración de semilla	73
Ilustración 49: Configuración de nodos LrWPAN.....	74
Ilustración 50: Configuración de planta.....	74
Ilustración 51: Definición de opción de movilidad --disc.....	75
Ilustración 52: Definición de la opción de movilidad --box.....	76
Ilustración 53: Definición de la opción de movilidad --interiores.....	76
Ilustración 54: Instalación de nodos en el edificio.....	77
Ilustración 55: Función para modificar atributos de canal.....	77
Ilustración 56: Modificación de variables canal y page.....	77
Ilustración 57: Método SetMyPhyOptoin.....	78
Ilustración 58: Definición de canal de propagación en interiores.....	79
Ilustración 59: Configuración de exponente de pérdidas.....	79
Ilustración 60: Añadiendo protocolos a la pila.....	80
Ilustración 61: Instalación de aplicaciones HTTP.....	81
Ilustración 62: Instalación de aplicaciones UDP.....	82
Ilustración 63: Tracing, Callblack y generación de animación.....	83

Ilustración 64: Estados tarjeta lrwpan [69].....	84
Ilustración 65: Inicio y fin de simulación.....	85
Ilustración 66: Captura tfg.pcap vista en wireshark.....	86
Ilustración 67: Salida del comando cat.....	86
Ilustración 68: Segunda captura vista en wireshark mostrando leve mejoría en tránsito de paquetes.....	87
Ilustración 69: Medidas de potencia en la nueva simulación.....	89
Ilustración 70: Paquetes UDP/seg.....	89
Ilustración 71: Visualización de la simulación en entorno NetAnim.....	90

Índice de tablas

Tabla 1: Bandas de trabajo RFID.....	15
Tabla 2: Características de los últimos estándares 802.11.....	28
Tabla 3: Releases para redes móviles.....	36
Tabla 4: Requisitos de instalación ns-3.....	56

INTRODUCCIÓN Y OBJETIVOS

1.1. ¿QUÉ ES LA INTERNET DE LAS COSAS?

Cuando hablamos de *Internet of things* o Internet de las cosas, puede surgir la pregunta: ¿qué es IoT?. Es habitual encontrarse con planteamientos que consideran la IoT como la tecnología del futuro, o referirse a ella como una tecnología disruptiva que marcará un punto de inflexión. No obstante, es poco habitual encontrar definiciones que vayan directamente al fondo de la cuestión, y que hagan referencia directa a la tecnología que la sustenta. Encontrar una definición capaz de resumir en un único párrafo lo que implica la IoT no es una tarea fácil, puesto que en no pocas ocasiones, tal definición depende de la solución propuesta por la entidad que plantea el despliegue. En [1] podemos encontrar una definición que desde mi punto de vista es de las más apropiadas, y que reza para sistemas de baja complejidad:

“Una IoT es una red que conecta “cosas” identificables de manera única a Internet. Las “cosas” disponen de detección/actuación y capacidades potenciales de programación. A través de la explotación de la identificación y detección únicas, información sobre la “cosa” puede recopilarse y el estado de la “cosa” cambiarse en cualquier lugar, en cualquier momento, con cualquier fin”

En cambio, si hablamos de entornos grandes, una definición idónea podría ser [1]:

INTRODUCCIÓN Y OBJETIVOS

“Internet of things prevé una red auto configurable,adaptativa y compleja que interconecta “cosas” a Internet mediante el uso de protocolos de comunicación estandarizados. Las cosas interconectadas tienen representación física o virtual en el mundo digital, capacidades de detección/actuación, una capacidad de programación característica y son identificables de manera unívoca. La representación contiene información que incluye la identidad de las cosas,su estado,localización o cualquier otra actividad cuya información sea social o privada. Las cosas ofrecen servicios con o sin intervención humana, por medio de la explotación de la identificación única, la captura de datos y las capacidades de actuación y comunicación. El servicio es explotado a través del uso de interfaces inteligentes y está disponible en cualquier sitio,en cualquier momento y para cualquier cosa tomando la seguridad en consideración”

Tomando en consideración las anteriores, podríamos definir la IoT como “una red de objetos capaces de conectarse a Internet, cuya información facilitada puede ser tratada y, en base a la misma y de manera opcional, se les pueda enviar órdenes para que lleven a cabo ciertas acciones.”

1.2 MOTIVACIÓN

Se ha comprobado que resulta complicado encontrar información acerca de las tecnologías empleadas en el despliegue de soluciones IoT y más complicado aún encontrarla en castellano reunida en una sola obra. Lo heterogéneo de este tipo de infraestructuras hace que habitualmente la documentación esté enfocada a entornos concretos que no nos aportan una visión general. Por otro lado, el disponer de una herramienta de simulación que nos permita explorar de primera mano este tipo de implementaciones puede ser de gran interés,sobre todo si se trata de una solución basada en software libre con la ventaja que esto aporta en cuanto a documentación disponible y desarrollos abiertos.

1.3 OBJETIVOS

Mediante la realización del presente trabajo se tratará de hacer una recopilación de todas aquellas tecnologías radio puestas a disposición de aquel que quiera desarrollar redes basadas en IoT desde un punto de vista amplio, haciendo un recorrido previo por algunas soluciones modernas desarrolladas mediante estas técnicas para poner en contexto el resto del trabajo. Asimismo, se pretende elaborar una pequeña guía introductoria para uno de los simuladores de redes más empleados en la actualidad en el ámbito de la investigación, como NS-3, llevando a término nuestro propio caso de uso real mediante simulación y así apreciar el potencial de la herramienta para ello, se abordará un estudio enfocado en el ámbito de los cuidados de la salud y comprobar así la respuesta del sistema utilizando el simulador presentado.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IOT

En el presente capítulo exploraremos en primer lugar los campos de aplicación actuales de las redes y los dispositivos IoT, haciendo un repaso a algunos proyectos que están en marcha hoy día, para a continuación profundizar en las tecnologías que nos permiten desarrollar estas soluciones en función del alcance. La elección de una u otra dependerá del lugar del despliegue, pues no es lo mismo una red de interiores de corto alcance que una red en un entorno rural con la que se deben cubrir miles de hectáreas de terreno,. Por lo tanto, no me he limitado a analizar exclusivamente aquellas tecnologías que aplican directamente a la implementación de redes IoT, si no que he querido también analizar brevemente algunas técnicas de comunicación radio que podrían permitir actuar como enlaces entre varias redes basadas en este tipo de tecnologías, como Wimax, LMDS o las soportadas por el estándar IEEE 802.22 , puesto que en escenarios complejos podría ser necesario recurrir a ellas. Se incluye además un anexo con una tabla de consulta que permitiría encontrar la tecnología de implementación IoT que mejor se adapte a las diferentes circunstancias.

2.1.CAMPOS DE APLICACIÓN ACTUALES

Las aplicaciones de la Internet de las cosas son casi infinitas, se están desarrollando

proyectos en multitud de campos diferentes entre los que podemos citar los siguientes:

2.1.1.AGRICULTURA

Al contrario de lo que se podría suponer, la agricultura es una de las industrias más abiertas a la innovación. Ha sido históricamente uno de los primeros sectores en implementar la automatización para mejorar la productividad, y en el caso que nos ocupa no es diferente.

Existen multitud de proyectos en el ámbito de la agricultura. Alguno como *FarmBeats* [2], desarrollado por Microsoft, tiene como objetivo llevar las soluciones IoT a zonas remotas, ayudando a los agricultores a mejorar la producción o a controlar las reservas de agua, en ocasiones escasas. Y todo ello en lugares sin conexión a internet y/o sin suministro eléctrico. Por ello, plantean utilizar el espectro libre de las bandas utilizadas para radiodifusión de televisión (*Networking Over White Spaces*), con el objetivo de construir redes de sensores de largo alcance, además de imágenes obtenidas por drones con baterías de larga duración. Todos estos datos pueden ser manejados por el agricultor en tiempo real, ofreciéndole la posibilidad de monitorizar la explotación e incorporarlos a servicios en la nube, proveyendo de este modo soluciones de *Machine Learning* que pueden resultar de utilidad para la industria agrícola.

La captura de imágenes realizadas por medio de drones y los datos que proveen las redes de sensores para explotarlas en beneficio del agricultor constituyen, como vemos, una de las principales tendencias en este sector.

2.1.2.COMERCIO MINORISTA

El comercio minorista es uno de esos entornos donde la IoT tendrá mayor presencia en un futuro no lejano. Están surgiendo multitud de soluciones enfocadas a este sector. Podemos citar, entre otras[3]:

- El cobro automático de productos. La tienda Amazon Go ya lo utiliza.
- Los descuentos personalizados automáticos cuando este se aproxima al producto.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

- La tecnología *iBeacon de Apple Inc.* basada en bluetooth de baja energía (BLE) que permite notificaciones utilizando beacons (balizas) bluetooth y puede ser utilizado para enviar ofertas a los viandantes al acercarse a una tienda, informar de precios de restaurantes y disponibilidad para hacer reservas entre muchas otras cosas.
- Las estanterías inteligentes, cuya función es la de ayudar a los encargados de reponer los productos en las mismas. Estas se equipan de lectores de etiquetas RFID con lo que la propia estantería puede informar de la necesidad de reponer algún producto o informar si estos están mal colocados.
- La optimización de la cadena de suministros. Utilizando la combinación de dispositivos GPS, etiquetas RFID y otros sensores, es posible conocer las variables ambientales, la ubicación o el tiempo de tránsito de los productos que están en el transporte. Esto permite a las empresas de logística mejorar sus servicios y al cliente saber que se están cumpliendo las condiciones pactadas.

2.1.3.CUIDADOS DE SALUD

El ámbito de los cuidados de la salud quizás sea uno de los entornos donde estas tecnologías puedan tener un impacto más positivo. Se está convirtiendo en un gran negocio para las empresas del sector, puesto que la implantación de sistemas de este tipo permite ahorrar costes y mejorar los servicios que prestan.

Sensores conectados a pequeños dispositivos -*Real-time Health Systems* (RTHS)- permiten evitar el uso de equipamiento médico pesado. Pueden ser utilizados para fines médicos u otros. Dispositivos como los inhaladores o los blister inteligentes entran pisando fuerte en el mercado.

Biosensores conectados a Internet , capaces de medir parámetros como la tensión, el nivel de glucosa en sangre, el pulso etc., permiten conocer el estado del paciente fuera de hospital dando la posibilidad al facultativo de adaptar el tratamiento sin necesidad de que este vaya a la consulta del médico y permitiendo que cuando esto ocurra este disponga de una mejor información. La tecnología de los biosensores abre la puerta a que

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

en un futuro puedan existir portales en línea que permitan realizar consultas médicas de manera telemática [4].

Por otro lado existen sistemas de monitorización y seguimiento de ancianos o personas con enfermedades cerebrales degenerativas. El seguimiento y control de constantes de recién nacidos entra dentro de las aplicaciones actualmente en desarrollo. Las tecnologías IoT enfocadas al cuidado de los niños, reciben el nombre de Internet de los niños pequeños (*Internet of Toddlers*).

2.1.4.INDUSTRIA MANUFACTURERA

En la actualidad se han puesto en marcha proyectos muy interesantes en la industria, que se encuentran encuadrados en lo que se conoce como IIoT (*Industrial Internet of Things*). Entre ellos podemos citar soluciones implementadas para la ayuda en la realización de los mantenimientos industriales. Para ello, la maquinaria se puede equipar con sensores capaces de capturar parámetros fuera de rango y notificar al servicio de mantenimiento la incidencia, e incluso medir cambios que serían imperceptibles para el ojo humano y ayudar de este modo en las tareas de mantenimiento predictivo. Estos sensores combinados con el *Machine Learning* y el *Big Data* están ayudando en la actualidad a mejorar los procesos industriales.

Haciendo referencia a ejemplos concretos de empresas que están desarrollando soluciones de este tipo, tenemos a la compañía DAQRI [5] que vende soluciones basadas en el empleo de la realidad aumentada implementada en cascos inteligentes, y que permite ver sobrepuertos parámetros sobre los diferentes sensores o medidores, facilitando de este modo el trabajo de ingenieros/as.

Todas estas tecnologías - la IoT, la realidad aumentada e incluso la realidad virtual - se están integrando de manera conjunta en sistemas denominados *Computerized Maintenance Management System* (CMMS), donde los datos de producción y mantenimiento son capturados en tiempo real y facilitados a los responsables de producción para tomar decisiones de manera inmediata.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

2.1.5.CIUDADES INTELIGENTES

Quizás sea el escenario de desarrollo de IoT que tendrá un mayor impacto en nuestras vidas en los próximos años, de hecho, en algunas ciudades este extremo ya empieza a ser una realidad en gran medida debido al gran impulso que ha dado a este sector la Unión Europea por medio, entre otras medidas, del Dictamen del Comité Económico y Social Europeo sobre «Las ciudades inteligentes como motor de una nueva política industrial europea» [6] y otras iniciativas soportadas por esta institución como *The European innovation partnership on smart cities and communities* (EIP-SCC) [7], demostrando una apuesta decidida por fomentar la implantación de este tipo de tecnologías dentro de la Unión Europea.

Podemos imaginar el potencial de una enorme red de sensores facilitando datos de tráfico, iluminación, sistemas pluviales, abastecimiento de agua etc. Todo esto usado en conjunto permite realizar una gestión mucho más eficiente del entorno urbano.

Han proliferado las compañías que trabajan en el campo de las Smart Cities, entre ellas podemos citar a *Wellness Telecom* [8], una empresa sevillana que ofrece soluciones para diferentes ámbitos como *WeLight* (un sistema de telegestión del alumbrado público), Quamtra, un sistema que permite gestionar los residuos urbanos, o *WeGo&Park*, para la detección inteligente de plazas públicas de aparcamiento. Asimismo, proveen un sistema de gestión centralizado de todas estas herramientas denominado *Smart City Brain*.

La empresa Ikusi Velatia se dedica a la ingeniería y al desarrollo tecnológico y está implantando en la actualidad soluciones de este tipo, por ejemplo, en San Sebastián, por medio de su aplicación *Spider Urban Management Platform* [9]. Esta aplicación permitirá a los gestores monitorizar parámetros como el uso del transporte público, controlar el equilibrio energético en los edificios públicos, la gestión del agua, la seguridad ciudadana y el tráfico -incluso el peatonal- desde un punto centralizado.

2.1.6.EDIFICIOS INTELIGENTES

Los edificios inteligentes son muchas veces considerados dentro del campo de

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

trabajo de las *smart cities* [5] pero dado el elevado número de proyectos que se desarrollan específicamente para los edificios resulta interesante adentrarse un poco más.

La inclusión de soluciones IoT permite mejorar las características de la edificación -incluso con aplicaciones durante el proceso constructivo- mejorando la habitabilidad, la gestión y el mantenimiento. Los datos recolectados en la aplicación de manera centralizada por los dispositivos, son utilizados para mejorar aspectos como la seguridad, la optimización del consumo energético o la experiencia de los habitantes, y así lograr maximizar los beneficios.

La compañía IONICS ha desarrollado un software que utiliza la funcionalidad *Azure Digital Twins* de Microsoft para crear un doble digital del edificio. De este modo, se pueden procesar todos los datos recolectados por sensores instalados en el edificio y controlar parámetros de calefacción, aire acondicionado e iluminación, entre otros. Todos estos datos son almacenados, al tiempo que los encargados de la gestión de la infraestructura pueden utilizarlos en la toma de decisiones. Por otro lado, el sistema tiene implementada una inteligencia artificial, entrenada para predecir fallos en dispositivos o incluso conocer cuándo un dispositivo no está siendo eficiente. De este modo, se consigue mejorar el rendimiento del mismo [10].

Otras empresas como IBM disponen de productos con funcionalidades parecidas a las ofrecidas por IONICS. En concreto, IBM Watson IoT provee soluciones específicas para edificios, usando un doble digital y/u ofreciendo a sus ocupantes prestaciones basadas en AI para alertas de seguridad personalizadas o notificaciones metereológicas, entre otras funciones[11].

2.1.7.ENERGÍA

IoT aplicado al sector energético permite a las compañías tener un mayor control sobre las operaciones. Así, en ámbitos como el del transporte de energía, resulta mucho más rentable optimizar el uso que se hace de las redes de suministro que construir otras nuevas. Tales avances permiten desarrollar proyectos como V2G (*Vehicle to Grid*) [12],

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

en los cuales los vehículos eléctricos conectados a la red no sólo consumen energía, sino que incluso pueden ceder la que tengan almacenada en sus baterías, contribuyendo de este modo a afrontar posibles picos de consumo.

Otro concepto diferente es el aplicado por la compañía Rockwell Automation que dispone de un novedoso sistema que monitoriza mediante sensores conectados a la nube de Microsoft Azure,a la que suministra datos, los mismos son tratados y puestos a disposición de los equipos de ingenieros que de ese modo pueden controlar toda la cadena de suministro de combustibles desde su extracción,pasando por transformación y distribución. Otra de las soluciones ofertadas por esta compañía es la posibilidad de conectar los surtidores de gas natural para la automoción que algunas empresas suministradoras empiezan a implementar en sus gasolineras a la nube y de este modo facilitar datos de consumos,existencias o incidencias detectadas en los surtidores y así responder en tiempo real a las necesidades [13].

2.1.8.TRANSPORTE

Si hubo un escenario donde históricamente se han utilizado dispositivos conectados para controlar diferentes parámetros este ha sido el transporte. El tráfico aéreo y el marítimo , además del ferrocarril, llevan desde hace muchos años siendo seguidos en tiempo real. El control del tráfico urbano e interurbano tampoco es ninguna novedad con sensores conectados a las ETD (Estación de toma de datos), capaces de detectar el paso de vehículos. Este último dispositivo se encarga de procesar la información,contando y calculando la velocidad de paso de los mismos y suministrando a los centros de control los datos obtenidos para conocer el estado de saturación de las vías en tiempo real. Los CCTV (Circuitos cerrados de TV) permiten visualizar por medio de imágenes en tiempo real el estado del tráfico. Los modernos reguladores de tráfico conectados a las salas de control facilitan a los operadores el estado de ocupación de la vía mediante sensores pudiendo actuar sobre los tiempos para cada fase o la duración del ciclo de la intersección entre otros parámetros, incluso de manera automática, por medio de sistemas adaptativos.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

La llegada de los vehículos autónomos estará estrechamente ligado a la aplicación de redes IoT capaces de comunicar unos vehículos con otros y con los centros de gestión. Permitirán buscar rutas óptimas, minimizando los atascos y consiguiendo optimizar el consumo energético. Mientras no llegue la revolución de los vehículos autónomos se han lanzado otras soluciones basadas en IoT para mejorar la situación del tráfico, como el servicio Car2Go[14]. Este es un sistema de carsharing donde se alquila un vehículo con una aplicación en el teléfono móvil que nos indica los vehículos que están próximos a nuestra ubicación. Con la propia app se reserva y se abre el vehículo. Una vez a finalizado el uso se puede estacionar en cualquier punto dentro de la zona de operación sin cargo adicional. Los vehículos utilizados son mayoritariamente eléctricos. Toda esta infraestructura es posible gracias a que los vehículos están conectados a Internet y disponen de dispositivos de radiolocalización por satélite de modo que parámetros como su posición y velocidad son conocidos y se controlan de manera centralizada.

Por otro lado se están desarrollando e implantando nuevas tecnologías para su aplicación en el ámbito del transporte, como por ejemplo las vías de tren inteligentes de la *Deutsche Bahn* [15], equipadas con sensores que permiten detectar fallos en su fase temprana, ayudando en la realización de los mantenimientos predictivos y así mitigar la ocurrencia de averías que paralicen el servicio.

2.1.9.PREVENCIÓN DE DESASTRES Y MONITOREO AMBIENTAL

Se está extendiendo el empleo de tecnologías basadas en IoT para la predicción de desastres naturales. A modo de ejemplo podemos citar las redes de sensores en el mar Caribe que facilitan datos en tiempo real para, por ejemplo, conocer el comportamiento de los huracanes, el seguimiento de la actividad volcánica en tiempo real o incluso los sensores ubicados en los árboles que se utilizan para medir la probabilidad de que se produzca un incendio forestal.

Las novedosas redes IoUT (*Internet of underwater Things*) [16] pueden ser usadas para sistemas de aviso de tsunamis, terremotos e inundaciones. Por otro lado, este tipo de

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

redes son muy empleadas también para monitoreo ambiental, midiendo la calidad del agua, la presión y la temperatura entre otras variables

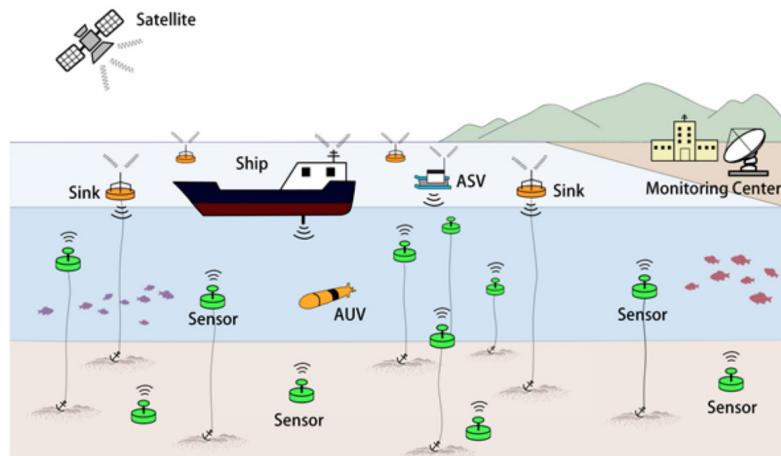


Ilustración 1: Modelo de red IoUT

2.2.REDES Y TECNOLOGÍAS

Las redes IoT están formadas por dispositivos que se conectan a Internet o entre ellos formando redes de comunicación en entornos muy diferentes y con necesidades también diferentes. Para implementar todas estas redes son necesarios los estándares que definen de que modo se lleva a cabo esta comunicación y como se estructura la pila de protocolos. A continuación expondremos los estándares y las tecnologías utilizados típicamente a la hora de construir infraestructuras IoT y los separaremos en función del alcance, desde aquellas de menor alcance hasta las redes de área extensa, centrándonos en aquellas donde el medio de transmisión es el aire, las redes inalámbricas, pues son las más ampliamente utilizadas para el despliegue de redes IoT.

2.2.1.WBAN

La redes WBAN(*Wireless Body Area Network*) son redes de muy corta distancia que se utilizan para conectar dispositivos de muy baja potencia y por tanto con alcance muy limitado en un ser humano o en un animal. Normalmente los dispositivos empleados son sensores implantados dentro del cuerpo o en su superficie que miden parámetros

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

fisiológicos como el ritmo cardíaco o la temperatura corporal y que se emplean para implementar las denominadas redes BSN (*Body Sensor Network*) o también pequeños dispositivos multimedia como auriculares, relojes inteligentes o incluso ropa “inteligente”.

El estándar definido por la IEEE para el acceso al medio en estas redes es el siguiente:

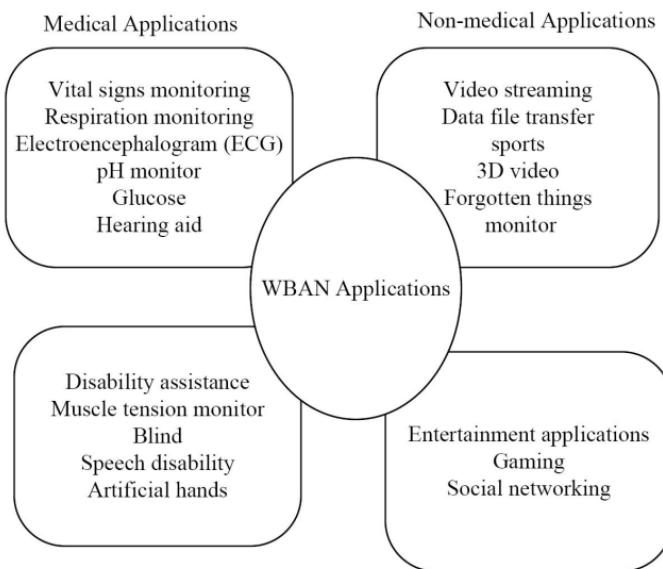


Ilustración 2: Aplicaciones de redes WBAN [17]

2.2.1.1. IEEE 802.15.6-2012

El TG6 (*task group 6*) encuadrado dentro del grupo de trabajo 802.15 de la IEEE se encarga de elaborar normas internacionales relativas a las redes WBAN, el trabajo desarrollado por este grupo ha llevado al lanzamiento del estándar IEEE 802.15.6-2012 [18]. En el mismo se definen las capas MAC (*Medium Access Control*) y PHY para dispositivos de baja potencia, con tres modelos para la capa física, concretamente:

- NB (*Narrow Band*)
- UWB (*Ultra Wide Band*)
- HBC (*Human Body Communications*)

Por otra parte la capa MAC puede operar en tres modos :

- Modo baliza con límites de supertrama de período de baliza.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

- Modo sin baliza con límites de supertrama.
- Modo sin baliza sin límites de supertrama.

2.2.2.WPAN

Las redes WPAN (*Wireless Personal Area Networks*) son aquellas cuyo alcance está limitado a unos pocos metros, aunque realmente estos rangos de alcance han ido aumentando. Las tecnologías empleadas para construir este tipo de redes se enumeran a continuación.

2.2.2.1.RFID (*Radio Frequency Identification*)

La tecnología RFID, basada en la norma ISO 14443, se utiliza principalmente para la identificación de objetos y es ampliamente empleado desde la década de los 60 en la implementación de sistemas EAS (*Electronic Article Surveillance*), que es como se denominan los sistemas antirrobo en las tiendas. La tecnología se caracteriza por el empleo de las llamadas etiquetas (*tags* en inglés), que contienen cierta información almacenada, y que pueden ser activas, pasivas o semipasivas/semiactivas en función de si se alimentan mediante baterías o mediante la energía recibida desde el lector. Por otro lado, estos dispositivos pueden ser de solo lectura si la etiqueta es programable una sola vez, o lectura-escritura si puede ser programada añadiendo nueva información a lo largo de su vida útil.

La contraparte de las etiquetas es el conjunto formado por el lector que incluye un modulo RF y un decodificador. Además, se debe contar con una antena. Por medio de estos dos dispositivos se puede enviar una señal que induce las etiquetas que se encuentren en su radio de acción, recibiendo de este modo la información que proviene de las mismas[44].

Las aplicaciones típicas son las siguientes:

- Sistemas antirrobo
- Identificación de objetos

- Verificación de autenticidad
- Control de accesos
- Trazabilidad de procesos
- Almacenamiento de información relativa a personas u objetos.

Las bandas de frecuencia de funcionamiento para esta tecnología son las siguientes:

BANDA	ALCANCE
LF (125 y 134 KHz)	<0,5 metros
HF (13,56 Mhz)	De 1 a 3 metros
UHF (433 y 860-890Mhz)	De 3 a 10 metros
2,45-5,8 GHz	Más de 10 metros

Tabla 1: Bandas de trabajo RFID[44]

2.2.2.2.NFC

NFC (*Near Field Communications*) es una tecnología de comunicación inalámbrica de muy corto alcance, que funciona en la banda de 13,56 Mhz y está pensada para el intercambio de datos entre dispositivos. El protocolo se desarrolla sobre la norma ISO 14443 y por el estandar NFCIP-1 [33], elaborado por el NFC forum I(ISO/IEC 18092:2004 [34]).

La comunicación establecida mediante esta tecnología es similar a RFID, con la salvedad de que el alcance es mucho menor. Ambos dispositivos pueden funcionar como iniciador y monitor de la comunicación, rol que ejercerá el extremo que solicite el inicio de la comunicación. Se pueden establecer dos modos de funcionamiento definidos en el estandar:

- **Modo Activo:**

Los dos dispositivos generan un campo electromagnético de modo que se puede producir envío de datos en los dos sentidos:

- **Modo pasivo:**

Sólo uno de los dispositivos genera el campo, que aprovecha el pasivo para alimentar su tag y enviar información. La velocidad de transmisión es de 424 Kbps para un alcance

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

máximo de unos 20 cm y el tiempo de configuración es muy rápido, menor a los 100 ms. Estas características lo convierten en un método ideal para el pago por medio de dispositivos móviles, como un teléfono o aplicaciones como control de accesos.

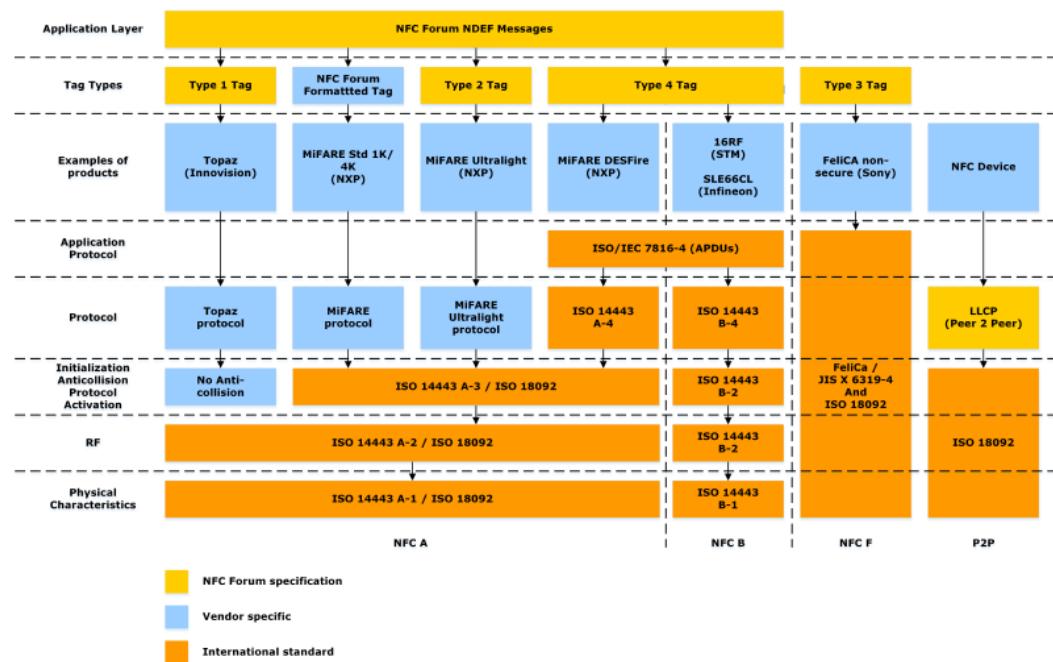


Ilustración 3: Pila de protocolos NFC [35]

2.2.2.3.BLUETOOTH

La tecnología *bluetooth* se desarrolla sobre el estándar IEEE 802.15.1 donde se especifica el comportamiento de las capas MAC y PHY en redes WPAN. La asociación “*Bluetooth Special Interest Group (SIG), Inc.*”, de carácter privado e integrada por empresas del sector, se encarga de desarrollar la tecnología.

Hasta la fecha la versión más reciente del estándar es la 5.1, que tiene como principal novedad respecto a versiones anteriores el permitir conocer, de manera aproximada, la ubicación de los dispositivos con los que existe conexión [20].

Desde la primera versión la tecnología se ha ido desarrollando y ha mejorado notablemente sus prestaciones. La versión 1.1 dispone de una tasa de transmisión limitada a los 721 kbps, con una potencia de transmisión de 100 mW como máximo, aunque

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

habitualmente no se superan los 2,5 mW y las redes admiten 10 piconets compuestas por un máximo de 10 dispositivos por piconet. Además, como técnica de modulación, se emplea FHSS con 1600 saltos por segundo, todo esto funcionando en la banda ISM.

Por su parte la versión 5 mejora notablemente las prestaciones de la versión 4 añadiendo dos nuevas variantes de la capa PHY, para tener finalmente tres tipos de capa, en concreto LE 1M, LE 2M y LE Codificada. La capa LE 1M, ya usada en la versión 4, utiliza una modulación GFSK con una tasa de símbolo 1 Msps. La capa LE 2M dobla la capacidad de la anterior permitiendo tasas de 2 Msps. Se utiliza también modulación GFSK con diferente desviación de frecuencia para los dos símbolos, que en este caso será de 370 KHz frente a los 185 KHz utilizados anteriormente, que permiten reducir la interferencia intersímbolo.

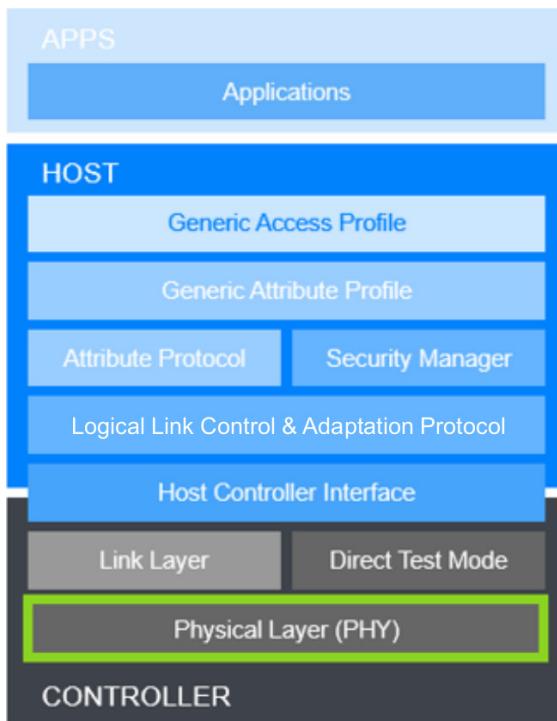


Ilustración 4: Pila de protocolos de Bluetooth versión 5 [21]

El alcance de los dispositivos desarrollados conforme a la versión 4 en condiciones de uso real (con espectro saturado y entornos con obstáculos) llega a los 350 metros e incluso a los 500 metros con equipamiento especial [22]. Con la capa física LE Codificada se puede cuadriplicar el alcance. Esto se consigue al implementar métodos de corrección

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

de errores FEC, empleando códigos convolucionales, con dos esquemas de codificación, mejorando así las prestaciones de la versión 4 que sólo incluye detección de errores. Estas mejoras lo convierten en una tecnología muy interesante para su empleo en redes IoT e incluso hacen que, hoy día, la tecnologías Bluetooth en sus versiones más modernas superen ampliamente el alcance típico de una red WPAN.

	LE 1M	LE Coded S=2	LE Coded S=8	LE 2M
Symbol Rate	1 Ms/s	1 Ms/s	1 Ms/s	2 Ms/s
Data Rate	1 Mbit/s	500 Kbit/s	125 Kbit/s	2 Mbit/s
Error Detection	CRC	CRC	CRC	CRC
Error Correction	NONE	FEC	FEC	NONE
Range Multiplier (approx.)	1	2	4	0.8
Bluetooth 5 Requirement	Mandatory	Optional	Optional	Optional

Ilustración 5: Características de la capa física Bluetooth 5 [22]

2.2.2.4. ZIGBEE

La especificación *ZigBee* se basa en el estándar IEEE 802.15.4 (LR-WPAN), cuya última versión data del año 2015 y que define el funcionamiento de las capas MAC y PHY para dispositivos de baja tasa de transmisión, baja potencia, radiofrecuencia de corto alcance y baja complejidad [19]. Sobre las características técnicas definidas en el estándar se construye la pila de protocolos.

Las frecuencias de funcionamiento a nivel mundial se ubican en la banda ISM con un total de 16 canales entre los 2,405 y los 2,480 Ghz, con velocidades de hasta 250 Kbps, que se obtienen gracias a la utilización de técnicas de espectro ensanchado,concretamente DSSS y modulación offset-QPSK. En la banda 902-928 Mhz se dispone de 10 canales

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

con tasas de 40 Kbps, sólo para el continente americano, y otro canal en los 868 Mhz para Europa con capacidad de transmitir datos a 20 Kbps. El método de control de acceso al medio es CSMA/CA y el alcance aproximado está entre los 30 y los 100 metros.

Otra característica heredada del estándar IEEE 802.15.4 son las topologías de red. Se pueden construir redes de tipo estrella, árbol y malla (redes P2P). La red de malla puede estar formada por hasta 65536 nodos agrupados en subredes de 255 nodos, con los siguientes tipos de dispositivo:

- **Dispositivo final:**

Este tipo de nodo se corresponde con aquellos dispositivos que sólo se pueden comunicar con sus nodos asociados y no pueden redirigir mensajes con otros destinos. Estos dispositivos pueden permanecer en estado dormido con el consiguiente ahorro de energía.

- **Router:**

El router es un nodo que ofrece funciones de enrutamiento al resto de dispositivos de la red y puede intercambiar mensajes con otros router. Al contrario que el dispositivo terminal debe permanecer en estado activo.

- **Coordinador:**

Este dispositivo es el que se encarga de formar la red, después de haber formado la red se comporta como un router en una red de tipo malla.

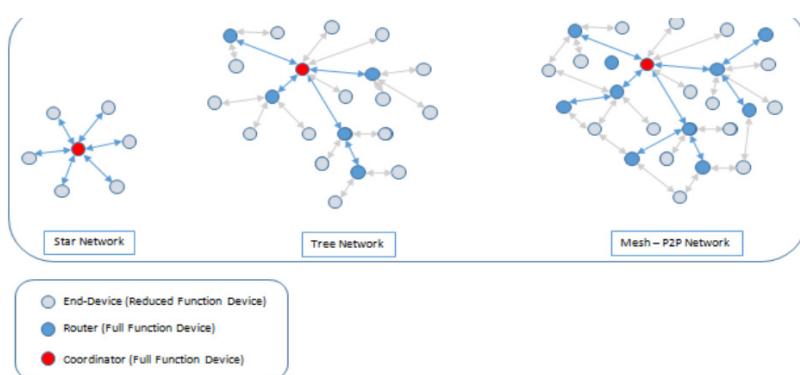


Ilustración 6: Topologías definidas en el estándar IEEE 802.15.4 [23]

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

El estándar IEEE 502.15.4 permite establecer tres niveles de seguridad:

- Sin seguridad
- Lista de acceso
- Seguridad con clave simétrica basada en un algoritmo de cifrado AES-128

2.2.2.5.LoWPAN

Esta tecnología surge del uso combinado del protocolo de la capa de red IPv6 junto con el protocolo 6LoWPAN que actúa como capa de convergencia entre las capas IPv6 y las capas MAC/PHY definidas en el estándar IEEE 802.15.4 y cuyo funcionamiento se recoge en el RFC 6282[24]. El hecho de utilizar IPv6 en la capa de red permite que esta tecnología soporte redes de tipo TCP/IP, y por otro lado, el hecho de usar direccionamiento IPv6 nos da acceso a un elevado número de direcciones ip para signar a los dispositivos.

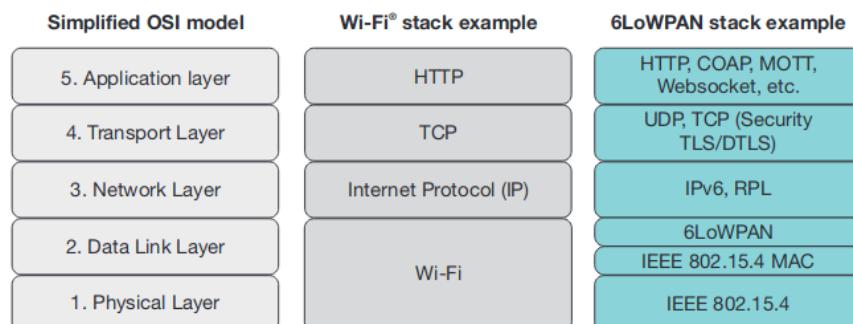


Ilustración 7: Pila de protocolos 6LoWPAN y modelo OSI [25]

El hecho de soportar de manera nativa redes IP le otorga una gran ventaja sobre otras tecnologías como ZigBee, Z-wave o Bluetooth que requieren pasarelas complejas para acceder a redes basadas en enrutamiento IP.

En las redes 6LoWPAN existen tres tipos de dispositivos:

- **Router Edge.** Su función es la de conectar las redes 6LoWPAN a otras redes como Internet. Como hemos indicado, el hecho de que las redes basadas en IP permitan que el router frontera pueda abstraerse de las capas

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

superiores - lo que hace que su carga de trabajo sea mucho menor - los convierte en dispositivos mas sencillos de utilizar.

- **Router.** En estas redes algunos dispositivos pueden comportarse como enruteadores, lo que les permite enrutar los datos con destino a otros nodos de la red.
- **Host.** Este tipo de dispositivos no tiene capacidad de enrutado. Son elementos terminales de la red y pueden permanecer en estado durmiente, despertándose periódicamente para comprobar si hay datos para él en el router al que está asociado.

El protocolo de enrutamiento utilizado generalmente es RPL, que viene definido en el RFC 6550 [26]. Se trata de un protocolo diseñado para redes de tipo LLN (*Low-Power and Lossy Networks*) y su funcionamiento básico consiste en crear una topología tipo árbol (*DAG-Directed Acyclic Graph-*), asignando un rango a cada nodo el cuál aumenta a medida que nos alejamos del nodo raíz. Los paquetes ICMPv6 que utiliza el protocolo son de tres tipos: DIS para solicitar información DODAG (Destination Oriented DAG), DIO para compartir información del DAG actuando como respuesta a una solicitud de tipo DIS y DAO que se envía periódicamente para actualizar la información de los nodos de rango superior.

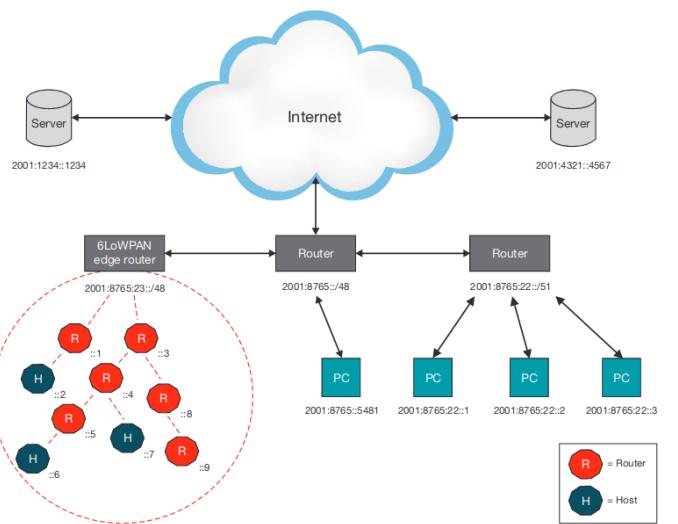


Ilustración 8: Modelo de interconexión de red 6LoWPAN[25]

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

La capa de adaptación 6LoWPAN se encarga de realizar la funciones de compresión de cabecera para segmentos de tipo UDP y funciones de fragmentación y reensamblado, debido a que la trama IEEE 802.15.4 tiene una longitud máxima de 127 bytes y el MTU máximo de IPv6 es de 1280 bytes, además de auto configuración sin estado - artificio mediante el cuál los dispositivos dentro de la red generan automáticamente su propia dirección IPv6-. Para evitar que dos dispositivos tomen la misma dirección se incluye un mecanismo llamado DAD (*duplicate address detection*).

Como ya se ha comentado, en la capa de red el protocolo utilizado es IPv6 y en la capa de transporte se puede utilizar UDP o TCP. Generalmente es recomendable utilizar un protocolo no orientado a la conexión como UDP, debido a la menor carga de datos en las cabeceras, lo que lo hace más recomendable para dispositivos de bajo consumo [25].

La seguridad en las redes IoT es muy importante. A mayores de la seguridad del cifrado AES-128 de las capas inferiores y del de la propia capa de red por medio de IPsec, la tecnología 6LoWPAN soporta seguridad en la capa de transporte por medio del protocolo TLS/DTLS, aunque conviene indicar que para que la seguridad a nivel de capa de transporte pueda ser implementada los dispositivos finales deben de incluir hardware específico de cifrado [25].

En la capa de aplicación se suelen usar protocolos como HTTP, que se corresponden con el RFC 2616 [26], CoAP definido en el RFC 7252[27] o MQTT, cuya operativa ha sido recogida en la norma ISO/IEC 20922:2016 [28]. Todos estos protocolos son típicos en el despliegue de redes IoT.

2.2.2.6. THREAD

Thread es una tecnología desarrollada por el “*Thread Group*” y específicamente pensada para redes IoT. Se trata de un protocolo para redes de tipo malla basada en IP y elaborado a partir de protocolos y estándares probados, como 6LoWPAN o UDP. El estándar utilizado para el acceso radio es el ya comentado IEEE 802.15.4, ideal para dispositivos IoT. También se incluyen funciones de seguridad obligatorias y permite

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

conectar un elevado número de dispositivos, bien dispositivo a dispositivo, o bien dispositivo a la nube. Las redes Thread son capaces de recuperarse de manera automática del fallo de un nodo, configurando nuevamente las conexiones de la red con lo que se evita el punto único de fallo.

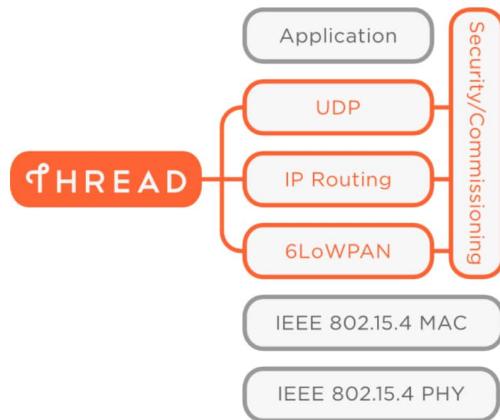


Ilustración 9: Pila de protocolos THREAD [29]

2.2.2.7.Z-WAVE

Z-wave es una tecnología de comunicación inalámbrica desarrollada por la compañía Zen-Sys. Está enfocada en el despliegue de redes para automatización en el entorno del hogar y equipos de poca potencia y con baja tasa de transmisión. Admite un total de 232 nodos [31], distribuidos en redes con topología en malla, los cuales pueden ser de dos tipos: controladores y esclavos. La distancia máxima entre dos dispositivos directamente conectados es de aproximadamente 30 metros y estos pueden actuar como retransmisores, aumentando el alcance de la red y permitiendo superar obstáculos. De todos modos, en las bandas de trabajo la señal es capaz de atravesar paredes. En España y otros países adscritos a la CEPT (*European Conference of Postal and Telecommunications Administrations*) las frecuencias en las que funciona la tecnología son 868.40, 868.42 y 869.85 Mhz.

Algunas aplicaciones típicas son:

- **Control del hogar.** Permite controlar de manera remota los dispositivos

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

del hogar conectados.

- **Ahorro de energía.** Se utiliza en aplicaciones que permiten el ahorro de energía, como la regulación automática de la intensidad lumínica de bombillas en base a lecturas facilitadas por sensores [31].
- **Sistemas de seguridad en el hogar:** Permite construir soluciones de seguridad, como la apertura de una cerradura de forma remota y asociar acciones complementarias, como encender las luces.
- **Entretención:** Se puede utilizar para implementar soluciones relacionadas con el entretenimiento, como activar un proyector de vídeo y al hacerlo bajar la pantalla motorizada de manera automática.

En cuanto a los protocolos empleados en el acceso al medio el estándar de referencia es el ITU-T G.9959, que indica el comportamiento de las capas PHY y MAC. En el caso de la capa física se definen dos tipos de modulación: FSK, para unas tasas de transmisión de 9,6 con codificación Manchester y 40 Kbps con codificación NRZ, y GFSK con BT=0,6 para 100 Kbps y NRZ. La capa MAC establece un control de acceso al medio de tipo CSMA/CA [32].

La capa de transporte es responsable de la retransmisión, el reconocimiento y la autenticación de origen de paquetes, además de despertar los nodos. Las tramas pueden ser de cuatro tipos:

- **Trama de difusión simple:**

Este tipo de tramas se envían a un nodo específico. El nodo debe contestar con un ACK, de modo que se sabrá si la trama ha sido o no recibida. En el supuesto de que no llegue la confirmación, se realiza una retransmisión de trama.

- **Trama de tipo ACK:**

Son las tramas de confirmación y no tienen carga útil.

- **Trama de tipo multicast:**

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

Son tramas que se difunden a más de un nodo de los 232 dispositivos dentro de la red y no existe la opción de trama de confirmación.

- **Tramas de tipo broadcast:**

Son recibidas por todos los dispositivos de la red y como el caso de las tramas multicast tampoco existe confirmación.

La capa de red se encarga del enrutamiento de la información y tanto los dispositivos de tipo controlador como de tipo esclavo ejercen esta función. En esta capa también se produce la comprobación de la topología de la red y sirve para el mantenimiento de la tabla de rutas en el controlador. Los datagramas pueden ser de tipo enrutado individual, que incluye información del repetidor, o bien de acuse de recibo, que no incluye carga útil y que sólo incluye un ACK en respuesta al datagrama de enrutado individual.

En la capa de aplicación se incluyen los comandos que se van a ejecutar en la red z-wave. La APDU en esta tecnología incluye: la clase de comando de aplicación, el comando de aplicación y el parámetro de comando.

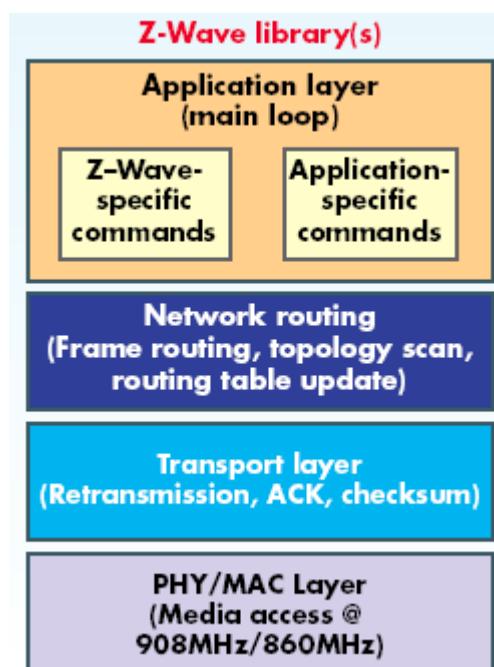


Ilustración 10: Pila de protocolos de tecnología Z-wave [30]

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

2.2.2.8.RuBee

Es una tecnología definida en el estándar IEEE 1902.1 [36], que ofrece una alternativa a las redes basadas en el estándar IEEE 802.15.4 como ZigBee o 6LoWPAN. Opera en las frecuencias de bandas de longitudes de onda hectométricas y kilométricas, concretamente frecuencias menores de 450 KHz. Estos dispositivos tienen un consumo muy bajo, del orden de micro vatios, con alcances que van desde el medio metro hasta los 30 metros. La tasa de transmisión es modesta, con velocidades máximas de 9,6 kbps, pero suficiente para construir redes de sensores y otras aplicaciones ligeras.

2.2.2.9.UWB

UWB (*Ultra Wide Band*) se basa en el estándar 802.15.3 [37]. Como su nombre indica es una tecnología de banda ancha pensada para redes de corto alcance -alrededor de 10 metros- y alta tasa de transmisión -en torno a 200 Mbps-, lo que lo convierte en un estándar ideal para redes multimedia en el entorno del hogar.

2.2.3.WLAN

Las redes WLAN (*Wireless Local Area Network*) son redes de alcance mayor a las redes WPAN. Se consideran redes de este tipo aquellas con un alcance de unos 100 metros, aunque realmente estos límites son un poco difusos, pues como hemos visto en apartados anteriores con tecnologías de tipo WPAN se pueden alcanzar distancias bastante grandes y próximas a los valores de las WLAN.

A continuación enumeramos algunas de las tecnologías destacadas para el despliegue de este tipo de redes.

2.2.3.1.WIFI

WIFI es la tecnología por excelencia para construir redes WLAN. Los estándares desarrollados por el grupo de trabajo 802.11 son los que definen el comportamiento de la capas de enlace (MAC) y física (PHY) en la construcción de redes de este tipo. Los

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

estándares principales son IEEE 802.11 a/b/g/n (WiFi) /ac (WiFi 5) /ad(WiGig)/ax (WiFi 6) en orden de más antiguo a más reciente. El 802.11ax denominado comercialmente por parte de la Wi-Fi Alliance como WiFi 6 será publicado a lo largo de este año 2019 junto con el estándar 802.11ay (WiGig segunda parte). Existen otros estándares que definen otras características de las redes Wi-Fi, como 802.11c que indica la información que será necesaria para interconectar redes de este tipo o 802.11e que define los mecanismos de calidad de servicio.

Los dispositivos que se pueden encontrar en estas redes son:

STA (*Stations*). Son los dispositivos con capacidad de utilizar las características de los estándares 802.11. Estos dispositivos pueden estar fijos en un emplazamiento o ser móviles, además pueden comunicarse entre ellos utilizando los protocolos diseñados a tal efecto.

AP (*Access Point*). Este dispositivo está diseñado para centralizar en un sólo punto las comunicaciones dentro de la red e incluso proveer capacidad para conectarse con otras redes, bien sea cableadas o inalámbricas.

Las arquitecturas de las redes WiFi pueden tener dos configuraciones diferentes:

Punto a punto, *red ad hoc*. En este tipo de red las STA se comunican unas con otras sin necesitar la presencia de un punto de acceso que gestione las conexiones, de modo que un dispositivo que esté dentro del alcance de cualquier otro de la red podría acceder a la misma.

Red de infraestructura. En este caso, la gestión de la red está centralizada en un AP que hace las funciones de nodo central y que permite establecer conexiones con otras redes externas si es necesario. Si la red que conforma está conectada a otras redes externas a través de una red troncal -denominada DS (*Distribution System*)- recibe la denominación de BSS (Basic Service Set) y engloba a todos aquellos equipos conectados dentro de su rango de alcance. De otro modo, si se trata de una red aislada, recibe el nombre de IBSS (*Isolated BSS*). Un modo de aumentar el alcance de estas redes es utilizando un DS de

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

tipo inalámbrico por medio de un radio enlace, pasando a denominarse entonces EBSS (*Extended BSS*).

En este trabajo no vamos a entrar en un análisis exhaustivo de los estándares para estas tecnologías, pero si indicar en la siguiente tabla las características técnicas de los últimos estándares de la serie 802.11.

	802.11n [38]	802.11ac [39]	802.11ad [40]	802.11ax [41]
Tasa máxima(Mbps)	600	6930	8085	11000*
Técnica de transmisión	MIMO/ OFDM	MU-MIMO (downlink)/ OFDM	SC(Single Carrier) y OFDM	MU-MIMO (uplink y downlink)/ OFDMA/OFDMA
Control de acceso al medio	DCF	CCA (Clear channel Assesment)	CBAP (contention based access period)	TWT (Target Wake Time)
	PCF		SP (service period)	Trigger-based Random Access
			PCP/AP Poll	
MCS para tasa máxima	64-QAM	256-QAM	64-QAM	1024-QAM
	GI 400 ns codificación 5/6	GI 400 ns codificación 5/6	Codificación LDPC 7/8	GI 800 ns u codificación LDPC 5/6
	40 Mhz	160 Mhz	1760 Mhz	160 Mhz
	4 flujos	8 flujos		
Banda (Ghz)	2,4	5 (puede funcionar en 2,4)	60	2,4
	5			5
Ancho de canal (Mhz)	20 o 40	20,40 o 80	1760(SC) o 1830,47(OFDM)	20,40,80,80+80 o 160
N.º de flujos	1,2,3 o 4	1,2,4,6 o 8		1,2,4,6,8

Tabla 2: Características de los últimos estándares 802.11

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

Otro estándar WiFi interesante y que está enfocado a los dispositivos IoT es WiFi HaLow -IEEE 802.11ah- que ha sido diseñado para operar en la banda de 900 Mhz lo que permite mayores alcances utilizando menor potencia de transmisión reduciendo así el consumo.

2.2.3.2. WAVE (*Wireless Access in Vehicular Environments*)

Esta tecnología se construye partiendo de las capas física y de acceso al medio definidas en el estándar 802.11p [42], sobre las cuáles se elabora el resto de la pila de protocolos que se encuentran en el conjunto de estándares IEEE 1609.

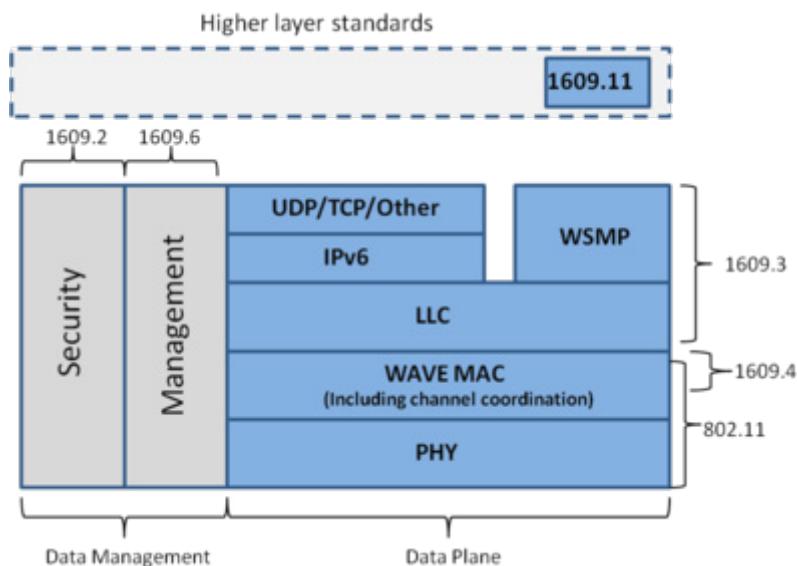


Ilustración 11: Pila de protocolos WAVE [43]

En el estándar 802.11p se intenta dar soporte a comunicaciones móviles para vehículos en las bandas de 5,9 Ghz y 6,2 Ghz .Se mejoran con respecto a otros estándares la serie 802.11 la en seguridad y se añaden mecanismos de handover más complejos [44].

En IEEE 1609.2[45] se definen las características de seguridad de la tecnología. En IEEE 1609.3[46] se indica el comportamiento de las capas de red y transporte. El estándar IEEE 1609.4[47] se encarga de extender las características básicas del comportamiento definido para la capa MAC en el estándar IEEE 802.11p.

Por otro lado, en las capas superiores, el estándar IEEE 1906.11[48] define el

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

Protocolo de intercambio de datos de pago electrónico por el aire para sistemas ITS (Intelligent Transportation Systems).

El protocolo WSMP (*Wave Short Message Protocol*), que forma parte del estándar IEEE 1609.3 se utiliza para tareas relacionadas con los parámetros físicos de la transmisión, que pueden ser controlados directamente desde las aplicaciones de las capas superiores.

Esta tecnología tiene aplicaciones muy interesantes en el ámbito del transporte, por citar algunos ejemplos:

- Sistemas de aviso para vehículos de emergencia
- Sistemas de aviso de frenada
- Peajes y cobro automáticos

2.2.3.3.HiperLAN2

HiperLAN2 (*High Performnace radio LAN*) [49] es una tecnología WLAN impulsada por el ETSI (*European Telecommunications Standards Institute*), pensada para funcionar en la banda de 5 Ghz con tasas de hasta 54 Mbps. La capa física es muy similar a la definida en el estándar 802.11a, admitiendo modulaciones BPSK,QPSK,16-QAM y 64-QAM. La capa MAC utiliza una solución basada en TDMA dinámico, en lugar de CSMA/CA -que se emplea en 802.11a-, y cuenta asimismo con mecanismos de QoS .

Esta tecnología ha sido ampliamente superada por los nuevos estándares WiFi, por lo que ha caído en desuso.

2.2.4.WMAN

Las redes WMAN (*Wireless metropolitan anrea network*) son aquellas cuyo alcance abarca distancias mayores que las wlan, del orden de varios kilómetros, pudiendo dar cobertura a ciudades enteras. Dentro de estas redes la tecnología más destacada es WiMAX, cuyas características en cuanto a alcance la hacen interesante para usarla como puerta de acceso a Internet para las redes IoT en entornos aislados.

2.2.4.1.WIMAX

La tecnología WiMAX se construye sobre los estándares desarrollados por el grupo de trabajo IEEE 802.16, en la versión más reciente es el 802.16-2017 [50]. Como es habitual, el IEEE se encarga de desarrollar los protocolos de las capas física y de enlace. El Wimax Forum es una entidad sin ánimo de lucro creada por actores de la industria WiMAX y que tiene como finalidad garantizar que la tecnología pueda interoperar entre los diferentes fabricantes.

A grandes rasgos, las principales características técnicas que presenta el estándar se ilustran en la siguiente tabla:

	IEEE 802.16	IEEE 802.16a/ Rev'd	IEEE 802.16e	IEEE 802.16j- 2009
Completed	December 2001	May 2004	Mid-2005	May 2009
Frequency	10 - 66 GHz	2 – 11 GHz	2 – 6 GHz	10 – 66 GHz Below 11 GHz
Application	Backhaul	Wireless DSL and Backhaul	Mobile Internet	Wireless MAN
Channel Conditions	Line of Sight Only	Non-Line of Sight	Non-Line of Sight	LOS NLOS
Bit Rate	32 – 134 Mbps at 28 MHz Channelization	Up to 75 Mbps at 20 MHz Channelization	Up to 15 Mbps at 5 MHz Channelization	Up to 120 Mbps at 25 MHz or 28 MHz Channelization
Modulation	QPSK, 16QAM and 64QAM	OFDM 256, OFDMA 2048 QPSK, 16QAM, 64QAM	Same as 802.16d, Scalable OFDMA	BPSK, QPSK, 16-QAM, 64-QAM, OFDM, OFDMA
Channel Bandwidths	20,25 and 28 MHz	Selectable Channel Bandwidths Between 1.5 and 20 MHz	Same as 802.16d	25 MHz or 28 MHz are typical

Ilustración 12: Características de los diferentes estándares 802.16 [51]

Las redes WiMAX surgieron como complemento a las redes de banda ancha para dar cobertura a zonas de difícil acceso. Al contrario de lo que ocurría con otras redes como WiFi (antes de las modificaciones a la capa MAC introducidas en el estándar 802.11e.), ofrece la ventaja de disponer de capacidades de QoS.

Las estaciones base WiMAX pueden dar cobertura a zonas muy amplias, de hasta 8000 km², y los dispositivos receptores pueden ser estaciones fijas o móviles instaladas en dispositivos portátiles, como ordenadores.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

El hecho de poder dar servicio a zonas muy amplias lo convierte en una solución de bajo coste y que puede suponer una alternativa a otro tipo de redes de datos.

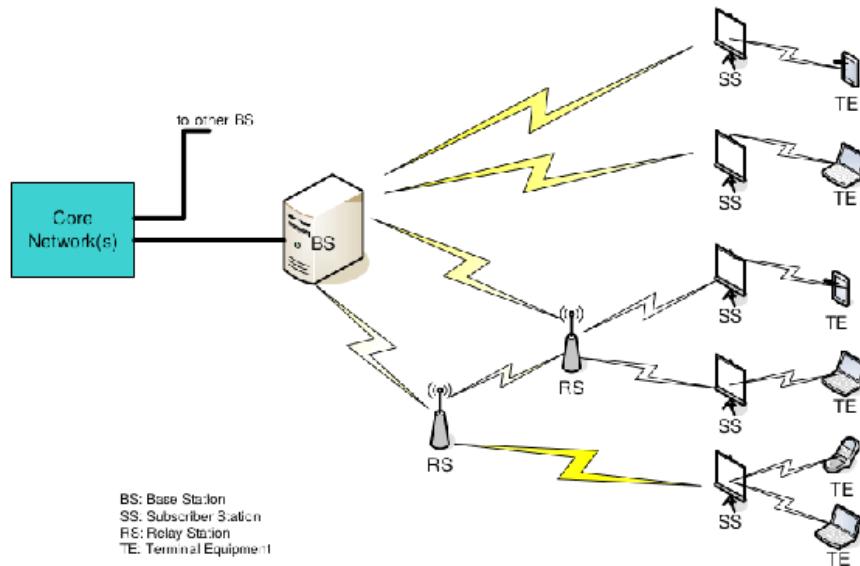


Ilustración 13: Arquitectura de una red WiMAX [51]

2.2.4.2.LMDS

LMDS (*Local Multipoint Distribution System*) es otra tecnología inalámbrica punto-multipunto de largo alcance. Al igual que WIMAX podría resultar interesante bajo ciertas condiciones para el uso en IoT y se construye sobre el estándar IEEE 802.16.1 [52]. En un principio fue pensada como una alternativa para servicios de transmisión de televisión digital. La tecnología puede operar en frecuencias entre los 2 y los 40 Ghz y dependiendo de la atribución de la frecuencia en cada país, la banda de trabajo oscila entre los 26 y 28 GHz, lo que hace que sea necesario LOS (*Line of sight*) y que las pérdidas debido a las inclemencias atmosféricas, como la lluvia, sea grande.

En frecuencias menores alrededor de los 3 Ghz, existe otra tecnología similar a LMDS: la MMDS (*Microwave Multipoint Distribution System*).

En el enlace se emplea un esquema de acceso múltiple, el TDMA, mientras que admite duplexión en el tiempo y en frecuencia. Es decir, TDD y FDD. Además, el estándar provee técnicas de modulación adaptativa, de modo que en aquellos casos que el enlace

presente mejores condiciones físicas - como puede ser la cercanía a la estación base - se podrían utilizar esquemas de modulación de mayor orden y codificación mas eficiente que permitan mayores tasas de transmisión.

2.2.4.3. WiBro

Esta tecnología es el equivalente asiático de la tecnología WIMAX vista en un apartado anterior, se basa en el estándar IEEE 802.16e [53] y ha sido desarrollada por la compañía Samsung. La tecnología de acceso múltiple es OFDMA y el duplexing es TDD, además se dispone de varios esquemas de modulación seleccionables mediante un sistema de modulación adaptativa.

2.2.5.WRAN

Las redes WRAN son redes de área regional con coberturas bastante grandes, del orden de las decenas de kilómetros. El estándar IEEE 802.22 [54] define el funcionamiento de las capas físicas y de enlace para estas redes, que podrían ser de interés para dotar de conexión con el exterior a redes IoT aisladas o zonas sin acceso con otro tipo de soluciones.

2.2.5.1.IEEE 802.22

En este estándar se define el funcionamiento de las capas PHY y MAC para redes de tipo WRAN. Las frecuencias de transmisión se sitúan en un rango que va desde los 54 Mhz a los 862 Mhz. Son utilizadas para la transmisión de señales de radiodifusión terrestre, aunque en virtud del segundo dividendo digital, las frecuencias desde 694 a los 790 Mhz pasarán a ser de uso exclusivo de los servicios de telefonía móvil. Por tanto, dado que esta tecnología está pensada para operar en bandas en uso, se propone el empleo de radio cognitiva. Por medio ésta se aprovechan los espacios libres dejados por las otras transmisiones (*white spaces*), no pudiendo interferir de ningún modo con los otros servicios. Esta función es gestionada por la capa MAC.

Para la capa física el estándar propone un método de acceso múltiple OFDMA con

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

ajuste dinámico de la modulación, la codificación y el ancho de banda. De este modo se permiten alcances de hasta 100 Km con tasas máximas de 22 Mbps para un canal de 6 Mhz de ancho de banda, siendo el mínimo de 1,5 Mbps/usuario para el enlace de bajada y 384 kbps para el de subida en el borde de la célula [44].

Los dispositivos empleados en este tipo de redes son la BS (*Base station*) y los CPE (*Consumer premises equipment*). Éstos deben implementar dos antenas independientes, una de ellas omnidireccional - para permitir la realización de medidas del canal en tiempo real - y la otra directiva - para comunicarse con la estación base -.

Las principales ventajas de esta tecnología son por un lado la ausencia de la necesidad de licencia, la posibilidad de funcionar sin línea de vista entre la estación base y el equipo terminal, además de su idoneidad para dar servicio a zonas remotas.

2.2.6.WWAN

Las redes WWAN son las redes inalámbricas de área extensa, entendiendo como tal aquellas que tienen un alcance capaz de cubrir áreas extensas. Las tecnologías que se pueden englobar en este apartado son las tecnologías de telefonía móvil celular o las comunicaciones por satélite, entre otras.

2.2.6.1.SigFox

SigFox es una tecnología muy interesante en el despliegue de redes IoT de tipo celular que utiliza la banda de 868 Mhz , de uso común en nuestro país. Mediante esta tecnología se pueden construir grandes redes de sensores, aportando como novedad que gran parte de las tareas de gestión de la red se realizan en la nube, aliviando de este modo la carga de trabajo de los dispositivos. De este modo se permite el ahorro de energía.

Por otro lado, el alcance puede llegar a ser grande. Señalar que en condiciones favorables de línea de vista se ha llegado a enviar un mensaje a 1000 kilómetros de distancia, si bien lo habitual es que estos valores se sitúen entre los 30-50 kilómetros en entornos rurales y entre los 3 y los 10 en entornos urbanos.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

Las tasas de transmisión son muy modestas para el canal ascendente: 100 bps con modulación DBPSK. Mientras que para el descendente será de 600 bps con modulación GFSK para anchos de banda de 1,5 KHz y 100 Hz (*UNB -Ultra Narrow Band-*) respectivamente. Valores que, a pesar de todo, son suficientes para el cometido de los dispositivos, puesto que están pensados para transmitir alrededor de 140 mensajes de 12 bytes por día [55]. Añadir que como método de acceso al medio se emplea un algoritmo basado en ALOHA.

Los dispositivos típicos que emplean esta tecnología son sensores, actuadores y pequeños dispositivos IoT para comunicación M2M(Machine to machine). Dadas las características de los dispositivos que conforman estas redes es una tecnología ideal debido a su bajo coste si la comparamos con las redes de datos móviles que, para este tipo de usos ofrece unas prestaciones excesivas y que significan una mayor inversión.

2.2.6.2.TECNOLOGÍAS MÓVILES

Dentro de este apartado se incluyen todas aquellas redes de tipo celular características de las redes de datos móviles. Las características de estas tecnologías son definidas por el 3GPP, que tiene como miembros organizativos a diferentes institutos de estandarización internacionales. En el caso de Europa esta se encuentra representada por el ETSI. También colaboran organizaciones como el UMTS Forum o el CDMA Development Group, en representación de los mercados.

A lo largo de los años se han ido lanzando diferentes versiones de la tecnología. Cada una de estas releases ha supuesto un avance en las prestaciones ofrecidas con respecto a la anteriores. Un estudio profundo de todas ellas supera con creces el objetivo del presente trabajo de modo que, ofrecemos un listado de las diferentes releases y las tecnologías implementadas en cada una de ellas.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

VERSIÓN	AÑO DE LANZAMIENTO	CONTENIDO
Phase 1	1992	GSM (FDMA/TDMA-FDD)
Phase 2	1995	GSM,códec audio EFR
Release 96	1997	GSM, 14,4 Kbps
Release 97	1998	GSM,GPRS
Release 98	1999	GSM, códec audio AMR,EDGE, GPRS para PCS1900
Release 99	2000	UMTS (WCDMA)
Release 4	2001	Release 2000
Release 5	2002	IMS,HSDPA
Release 6	2004	HSUPA,MBMS,integración con WLAN y mejoras en IMS
Release 7	2007	QoS mejorada,VoIP,HSPA+,EDGE Evolution,protocolo SIM de alta velocidad para permitir funcionalidades NFC
Release 8	2008	LTE(OFDMA,MIMO,FDE),All-IP,Dual-Cell HSDPA,UMTS HNB
Release 9	2009	Interoperatividad UMTS/LTE con WiMAX,Dual-Cell HSUPA,LTE HeNB
Release 10	2011	LTE Advanced que cumple los requisitos de IMT Advanced 4G
Release 11	2012	Redes heterogéneas HetNet,nueva capa de servicio e interconexión de servicios por IP
Release 12	2015	Small Cells mejoradas,agregación de portadoras,beamforming,MIMO masivo,comunicación D2D
Release 13	2016	LTE Advanced Pro, posicionamiento en interiores,LTE en banda sin licencia
Release 14	2017	Eficiencia energética,Servicios de localización (LCS),Mission Critical Data sobre LTE, Mission Critical Video sobre LTE, FMSS,MBSP,IoT masivo,CBS
Release 15	2018	5G-NR (New Radio)

Tabla 3: Releases para redes móviles

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

El trabajo de elaboración de nuevas versiones de la tecnología no descansa. En la actualidad el 3GPP ya se encuentra trabajando en la Release 16 que incide en el desarrollo de las incipientes redes móviles 5G, que prometen tener un gran impacto en el mundo de la IoT.

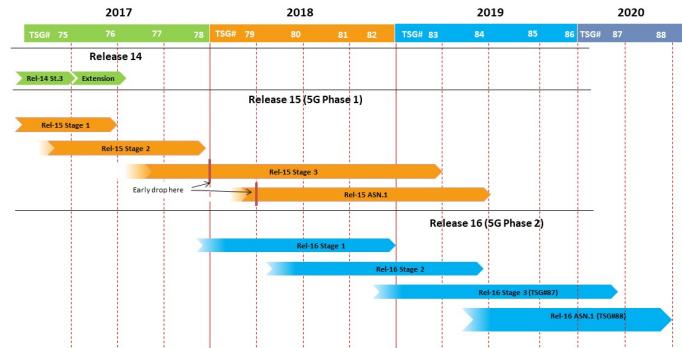


Ilustración 14: Releases actualmente en desarrollo [56]

Las nuevas redes 5G suponen un salto de calidad con respecto a las anteriores, prometiendo tasas de transmisión teóricas mucho mayores, menor latencia y capacidad para acopiar en las celdas en un número mucho mayor de dispositivos.

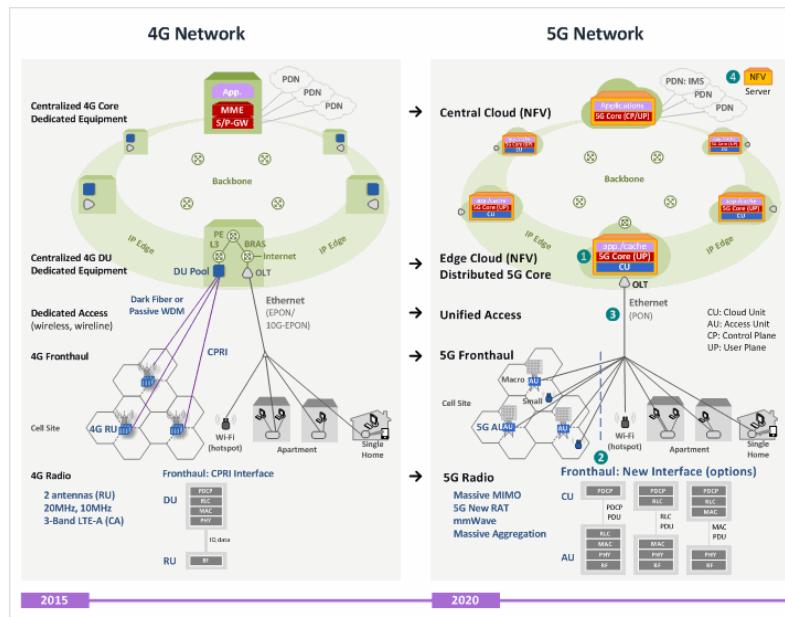


Ilustración 15: Compratativa de redes 4G y 5G [57]

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

Las releases 13, 14 y 15 incluyen soluciones específicas para IoT. Las denominadas *narrow band cellular IoT*, las tecnologías NB-IoT (*Narrowband IoT*), EC-GSM-IoT y MTC están definidas en estas versiones.

2.2.6.3.Ingenu RPMA (Random Phase Multiple Access)

RPMA es una solución tecnológica para redes de tipo LPWAN desarrollada por la compañía Ingenu. Se emplea una técnica de espectro ensanchado, concretamente DSSS en la banda ISM de 2,4 Ghz. La tecnología hace uso de esta banda debido a que la regulación favorable a la misma permite emitir a mayor potencia sin necesidad de licencia y se logran alcances de hasta 16 kilómetros -normalmente ronda los 4-. La tasa máxima es de 8 kKbps y los dispositivos de la red se conectan en una topología de red de tipo estrella [58].

La técnica RPMA es capaz de adaptar la tasa de transmisión de manera adaptativa. Para ello los dispositivos terminales de la red adecúan el factor de ensanchamiento empleado en función del nivel de señal recibido desde la estación base. La estación base es informada mediante un mensaje del nivel de señal recibido en cada dispositivo pudiendo adaptar de este modo las tasas de transmisión en el canal descendente. Para garantizar la correcta recepción de mensajes se emplea un algoritmo de Viterbi, para la descodificación de códigos convolucionales, que permite recuperar la información incluso en situaciones donde la pérdida de bits es elevada. La información está protegida pues se envía cifrada por el medio aéreo por medio de un algoritmo AES-128.

2.2.6.4.LoRa y LoRaWan

LoRa es una tecnología de redes de baja potencia y área extensa desarrollada por Semtech, que define la capa física. Se admiten redes con topologías de tipo estrella y de tipo malla. Se emplea una técnica de espectro ensanchado (*CSS -Chirp Spread Spectrum-*) operando en las bandas ISM por debajo de 1 Ghz - concretamente 433 Mhz y 868 Mhz- e incluye corrección de errores FEC (*Forward Error Correction*).

Las capas superiores de la tecnología no son propietarias. De todos los desarrollos

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

para estas capas el más destacado es LoRaWan que está gestionado por la LoRa Alliance, entidad encargada de facilitar la NetID necesaria para identificar la red. Esto será útil si queremos que nuestra red funcione en colaboración con otras redes, por ejemplo, para dar servicio a los dispositivos que se encuentre en roaming. De todos modos, las identificaciones 0x000000 o 0x000001 se pueden usar libremente si nuestra red es independiente.

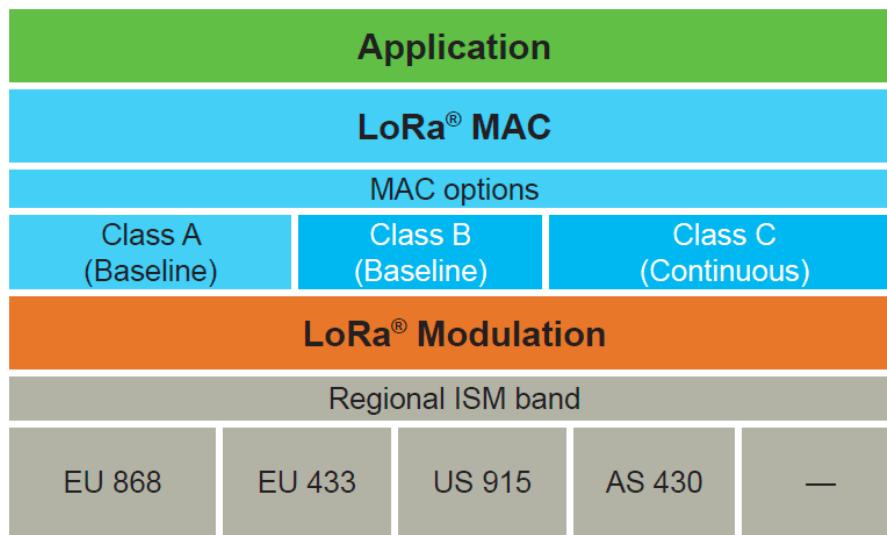


Ilustración 16: Pila de protocolos LoRaWan [59]

LoRaWan permite comunicación bidireccional con una carga útil que va desde los 19 a los 250 bytes con una cabecera de 12 bytes[58]. El alcance depende del balance del canal, aunque en condiciones de laboratorio se ha llegado a alcances de 5 kilómetros en zonas urbanas y 30 kilómetros en condiciones de LOS. En entornos rurales se ha medido un alcance de 8 kilómetros entregando el 100% de los paquetes.

El ancho de banda de los dispositivos LoRa puede ser configurado desde los 7,8 KHz a los 500 KHz, pasando por 125 KHz y 250 KHz y los factores de ensanchamiento son ortogonales, permitiendo así alojar en la misma banda diferentes señales. En función del factor de ensanchamiento y del ancho de banda y en base a un mecanismo ADR (Adaptive Data Rate), que adapta la tasa a cada dispositivo es posible optimizar la capacidad de la red y la duración de la batería de los dispositivos.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

La red soporta, usando ADR, hasta 1600 nodos que emplean el método de acceso al medio ALOHA simple. Los nodos pueden ser: de clase A (dispositivo básico), de clase B (al que se le puede realizar ping) y de clase C (similar a los de clase A con la salvedad de que está constantemente en modo recepción cuando no está transmitiendo).

2.2.6.5. Weightless

Weightless es un grupo de tecnologías definidas y administradas por el Weightless-SIG. Se han definido tres estándares:

- Weightless-N. Este estándar está enfocado a dispositivos de muy bajo coste.
- Weightless-W. Utiliza los espacios en blanco en las bandas de radiodifusión
- Weightless-P. El más actual de los tres. Es un estándar de banda estrecha que funciona en las bandas ISM por debajo de 1 Ghz. Se basa en dividir el espectro en canales de 12,5 Khz usando modulaciones GMSK y offset-QPSK para una tasa de transmisión adaptativa que va desde los 200 bps a los 100 kbps. Para garantizar la correcta recepción de los mensajes se utiliza FEC y ARQ (*Automatic Retransmission Request*). Implementa cifrado AES-128/256.

El consumo de los dispositivos es muy bajo. La potencia máxima de transmisión de 17 dBm para un alcance de 2 kilómetros y el consumo en reposo de apenas 100uW, permite que estos sean alimentados con una pequeña pila de botón. Además el fabricante promete capacidad para realizar actualizaciones de firmware “*over-the-air*”.

2.2.6.6. Telensa

Esta tecnología es una solución de banda estrecha para redes IoT funcionando en las bandas ISM menores de 1 Ghz. Al contrario que otras tecnologías LPWAN admite comunicación bidireccional lo que permite, no sólo monitorizar, sino también enviar órdenes a los dispositivos.

Una estación base telensa permite albergar mas de 5000 nodos con una cobertura de 2 kilómetros en entornos urbanos y de 4 en entornos rurales. Los nodos pueden funcionar

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

incluso sin conexión en función de la programación. Se estima una vida útil de hasta 20 años. En la actualidad está siendo utilizado ampliamente en el despliegue de redes IoT en redes del ámbito de las *smart city*.

2.2.6.7..WAVIoT

WAVIoT es un proveedor de servicios IaaS texano que ofrece una solución denominada NB-Fi (Narrowband fidelity). Funciona, como en los casos anteriores, en las bandas ISM inferiores y que divide los 500 KHz disponibles en 5000 canales de 100 Hz, transmitiendo por tanto las señales uplink y downlink en canales de 50 Hz a una tasa mínima de 50 baudios. La sensibilidad de los receptores de los equipos de pasarela es de -154 dBm y son capaces de dar servicio a 1 millón de nodos[58], el balance de enlace es de 176 dBm. El alcance puede llegar a los 16 kilómetros en zona urbana y a los 50 en zona rural con unas latencias de 30 segundos en el canal ascendente y 60 en el descendente. La información viaja cifrada mediante un algoritmo cifrado de bloque- XTEA 256-. El consumo de energía es mínimo. Durante el envío de una ráfaga el consumo es de tan sólo 50 mA y en estado silente de unos pocos uW lo que permite una vida útil de unos 20 años [58].

NB-IoT es un estándar de pila completa, con lo que todas las capas están definidas en el mismo. La información que captura la pasarela es almacenada en servicios en la nube y puede ser accedida desde plataformas diseñadas a tal efecto mediante una API.

En la siguiente imagen podemos ver las tecnologías LPWAN (Low Power Wide Area Network) más utilizadas y organizadas en función de si se trata de tecnologías abiertas o propietarias.

ESTADO DEL ARTE EN EL ÁMBITO DE LA IoT

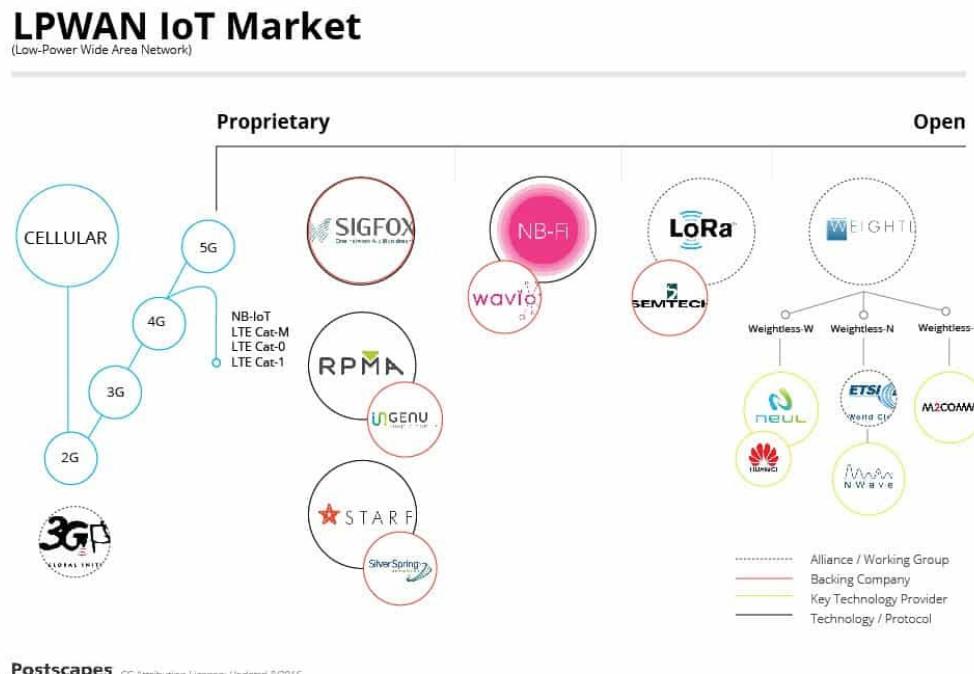


Ilustración 17: Tecnologías LPWAN [60]

Y en la siguiente, a modo de resumen, se muestran los protocolos y los estándares empleados en la mayoría de implementaciones inalámbricas. Son los utilizados generalmente para el desarrollo de soluciones IoT.

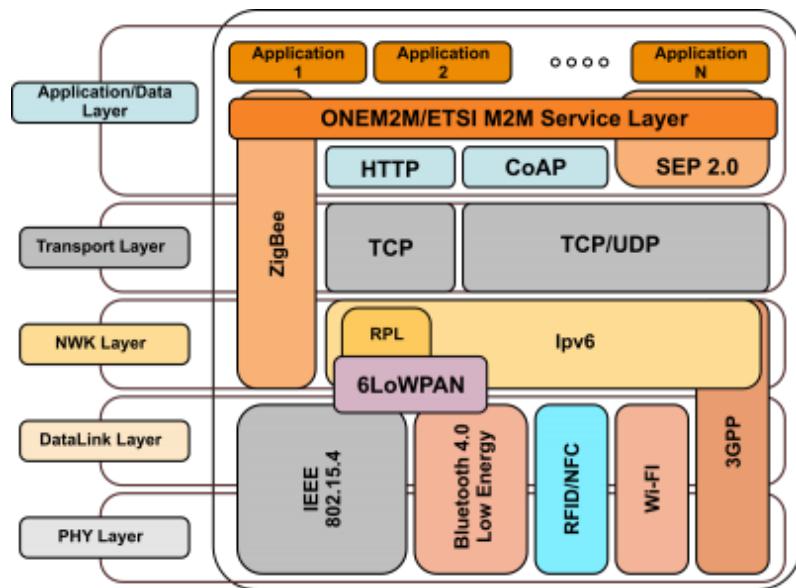


Ilustración 18: Compendio pila de protocolos [61]

**3****SIMULADOR NS3**

3.1.INTRODUCCIÓN

Existen multitud de simuladores de red. Cada uno dirigido a resolver diferentes tipos de simulación en función de las necesidades del usuario. Estas necesidades vendrán marcadas por le tipo de sistema que se pretende probar. Estos se pueden diferenciar atendiendo a diferentes características:

- **Sistemas estáticos y sistemas dinámicos.** En función de si la variable de estado cambia o no a largo del tiempo.
- **Sistemas deterministas y sistemas estocásticos.** Cuando el sistema a simular no atiende a cambios dependientes de variables probabilísticas se dice que es determinista. Por el contrario, cuando las variables de estudio se definen mediante procesos estocásticos se dice que son sistemas estocásticos.
- **Sistemas continuos y sistemas discretos.** En este caso la diferencia la establece el tipo de variable temporal sobre la que se construye el sistema,bien sea esta continua o discreta.

En general, los simuladores de red se pueden separar en dos grandes grupos:

- **Simuladores de eventos discretos.** En este caso el simulador se basa en la

SIMULADOR NS3

modificación de ciertas variables con la exigencia de que estas no cambian su valor entre dos instantes de tiempo concretos, es decir, a partir de un estado inicial se van creando nuevos estados o eventos tomando una variable temporal discreta.

- **Simuladores de tiempo continuo.** Los simuladores de tiempo continuo se basan en una variable temporal continua, es decir, las variables a simular cambian de manera continua con el tiempo. Estos sistemas emplean modelos matemáticos complejos para llevar a cabo la simulación.

NS-3 es un simulador de eventos discretos de software libre, que se ofrece bajo la versión 2 de la GNU (*General Public License*). Ha sido desarrollado para proveer una plataforma versátil de simulación de redes con fines de investigación y educativos. Con NS-3 se pueden simular redes que difícilmente se podrían reproducir en entornos reales. Proveen un entorno de trabajo que permite emular el funcionamiento de los protocolos de red de manera modular y con gran nivel de detalle, permitiendo incluso simular eventos de tipo físico, como el consumo de energía de los dispositivos, diferentes modelos de propagación y un largo etcétera. Estas características diferencian a NS-3 de otros simuladores:

- NS-3 no ofrece un interfaz gráfico como otros simuladores. Deberemos trabajar con la línea de comandos y con los lenguajes de programación C++ y Python y los entornos de desarrollo asociados. Esto que a priori parece una desventaja permite que se puedan utilizar diferentes herramientas de análisis y visualización externas al propio framework lo que ofrece mucha libertad.
- NS-3 ha sido diseñado para ser utilizado en sistemas *Linux* y *macOS*, aunque se puede utilizar en entornos Windows usando Cygwin o *Windows Subsystem for Linux*, además de contar con una versión para *Visual Studio*.
- NS-3 no está mantenido por ninguna compañía. Su desarrollo lo lleva a cabo la

comunidad NS-3.

La documentación está disponible de manera abierta en la página web del proyecto donde también es posible descargar las publicaciones más recientes del producto.

3.2.FUNCIONAMIENTO DE NS-3

Como hemos indicado la simulación se realiza en tiempo discreto, de modo que la misma va saltando de evento en evento. Todo el proceso se realiza por medio de funciones C++ que al ejecutarse lanzan los eventos para que estos se lleven a cabo en instantes concretos. El programador de la simulación ordena la ejecución de los mismos. Para iniciar la ejecución de una simulación se invoca la función *Simulation::Run()* y a partir de ese momento se van lanzando los diferentes eventos enlazados por las diferentes funciones. La ejecución finaliza en función de un tiempo que se haya programado mediante la función *Simulator::Stop(Time const &delay)* o cuando el último evento haya terminado.

Los elementos que forman parte de una simulación con NS-3 son los siguientes:

- **Nodo:**

El nodo hace las funciones de contenedor para una pila de protocolos completa y un dispositivo de red. Existen módulos que nos permiten añadir diferentes protocolos, desde la capa física a la capa de aplicación. Además, existe la posibilidad de utilizar contenedores de nodos, lo que será muy práctico para construir redes donde tengamos nodos que se comporten del mismo modo.

- **Canal:**

En NS-3 podemos añadir canales a nuestra simulación de modo que, esta se comporte de la manera mas fiel posible a como lo haría en el mundo real. El canal será el medio por el cuál la información viaje entre dos nodos. Existen multitud de canales diferentes para añadir a nuestra simulación.

- **Dispositivo de Red:**

SIMULADOR NS3

Cuando tratamos de acceder a un canal en el mundo real, necesitamos un dispositivo que realice esta función. Del mismo modo, en NS-3 debemos añadir una dispositivo de red que permita conectar nuestro nodo con otros. Este dispositivo se añadiría al nodo que hemos indicado antes.

- **Aplicaciones:**

En lo más alto de la pila de protocolos se encuentran las aplicaciones. En NS-3 disponemos de algunos módulos ya programados que nos permiten añadir aplicaciones a nuestra pila dentro de cada nodo. Además, por medio de la clase *Application* podemos crear nuestras propias aplicaciones y poder así simular su comportamiento en la red simulada.

- **Topology Helpers:**

Se trata de clases que nos ofrece el *framework NS-3* para ayudarnos a construir nuestra topología de red sin tener que preocuparnos de cada una de las partes de manera individual. Existen muchos helper para diferentes funciones, por ejemplo, la clase *CsmaHelper* nos ayuda a crear dispositivos de red de tipo *csma* e incluye métodos para poder configurar atributos de canal, instalarlo en un nodo particular o en un contenedor de nodos.

Las simulaciones, como cualquier programa escrito en C++, constan de las siguientes partes:

- **Plantilla:**

La primera línea del programa será una línea de modo *emacs* que indica a *emacs* la convención utilizada para el código fuente. Ha sido adoptado este estilo de código de modo que las contribuciones al proyecto deben seguirlo.

Dado que el código está escrito bajo licencia *GNU GPL* al principio del fichero también se incluyen unas líneas indicando este extremo, en ocasiones podría encontrarse un mensaje de copyrigth de alguna de las instituciones involucradas en el proyecto y el

nombre y el correo electrónico del autor del programa.

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 */

```

Ilustración 19: Ejemplo de plantilla

- **Directivas Include**

A continuación se añaden las *sentencias include* que son directivas utilizadas por el preprocesador para indicarle al mismo que archivos deben ser incluidos en el código, de modo que cuando se encuentra esta sentencia se sustituye por el código del archivo indicado. Aquí se añadirán aquellos archivos de cabecera que sean necesarios para nuestra implementación:

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
```

Ilustración 20: Directivas Include en un código NS-3 típico

- **Ns3 Namespace**

El *namespace* para los proyectos NS-3 es ns3, de modo que se debe añadir la línea de código *using namespace ns3* al principio de los programas. Al hacer esto evitaremos utilizar la expresión ns3:: antes de llamar a una clase del *framework ns3*.

- **Logging**

La siguiente línea que encontramos en los programas NS-3 es del estilo *NS_*

SIMULADOR NS3

`LOG_COMPONENT_DEFINE ("ThirdScript")`. Se trata de una macro definida en el archivo `log.h` que pertenece al módulo core y que es necesario incluir al principio de todos los programas que vayan a usar la macro `NS_LOG`. Lo que hace es definir un nuevo componente log que puede ser activado o desactivado usando las funciones `ns3::LogComponentEnable` y `ns3::LogComponentDisable`, respectivamente, o con la variable de entorno `NS_LOG`. Esto sirve para que durante la ejecución del programa se nos vaya mostrando información.

Hay siete niveles de mensajes de log por medio de los cuales se va incrementando el nivel de verbosidad:

- **LOG_ERROR.** Van asociados a la macro `NS_LOG_ERROR` y muestra mensajes de error.
- **LOG_WARN.** La macro es `NS_LOG_WARN` y muestra mensajes de alerta.
- **LOG_DEBUG.** Asociado a la macro `NS_LOG_DEBUG`. Se utiliza para obtener mensajes de depuración y su uso no es habitual.
- **LOG_INFO.** Tiene como macro `NS_LOG_INFO`. Se usa para ir mostrando información durante la ejecución del programa.
- **LOG_FUNCTION.** Que tiene asociada la macro `NS_LOG_FUNCTION` para funciones miembro y `NS_LOG_FUNCTION_NOARGS` para las funciones estáticas. Se usa para describir las funciones que se van ejecutando.
- **LOG_LOGIC.** Se emplea para describir el flujo lógico dentro de una función. La macro asociada es `NS_LOG_LOGIC`.
- **LOG_ALL.** En este caso no hay macro asociada y se emplea para sacar un log de todos los parámetros citados con anterioridad.

Existe otra funcionalidad denominada `LOG_LEVEL_TYPE` (donde en lugar de type iría el log que deseemos de los anteriores) que activa todos los log desde este hacia abajo, por ejemplo. Empleando `LOG_LEVEL_INFO` estamos activando también los mensajes

provenientes de las macros NS_LOG_DEBUG, NS_LOG_WARN y NS_LOG_ERROR.

Para activar el sistema de logging podemos utilizar la función *ns3::LogComponentEnable(char const* name, enum LogLevel)* como en el siguiente código (siguiendo con el ejemplo donde habíamos modificado la macro *NS_LOG_COMPONENT_DEFINE ("ThirdScript");*):

```
LogComponentEnable ("ThirdScript", LOG_LEVEL_INFO);
```

donde estamos activando los mensajes provenientes de las macros NS_LOG_INFO, NS_LOG_DEBUG, NS_LOG_WARN y NS_LOG_ERROR para el programa “*Thirdscript*”, si por ejemplo en el código añadimos una línea como la siguiente:

```
NS_LOG_INFO ("Creando topología de red");
```

A la hora de ejecutar en programa veremos en pantalla el mensaje “*Creando topología de red*”.

Otra opción es utilizar variables de entorno del shell, si por ejemplo, quisiésemos aumentar el nivel de logging podríamos introducir por consola el siguiente comando antes de ejecutar el programa: *\$export NS_LOG ThirdScript=level_all* con lo cual habríamos activado todos los niveles para el programa tfg.

Existe otra macro denominada NS_LOG_UNCOND que está asociada a mensajes incondicionales, es decir, se mostrará siempre independientemente del nivel de logging configurado.

- **Definición de clases y funciones:**

A continuación se escriben las nuevas clases y funciones que necesitamos para nuestro programa, además de escribir el contenido de la función main.

Dado que el objetivo último de diseñar un programa en el entorno ns-3 es la simulación, será muy importante obtener información de la misma. De nada serviría si no podemos obtener datos para ello. Además de sacar valores de variables a través de la salida estándar *std::cout*, se pone a nuestra disposición un sistema de rastreo que nos

SIMULADOR NS3

ofrece información de multitud de parámetros.

El sistema de rastreo se basa en las *tracing source* y los *tracing sink*. Las *tracing sources* son entidades que señalan eventos durante una simulación. Las *trace sink* son las entidades que consumen la información facilitada por las *trace sources*. De este modo, a la hora de diseñar el programa podemos añadir estas fuentes de información en aquellos puntos que consideremos importantes y sacar datos de interés.

Las entidades *trace sink* emplean Callbacks para acceder a la información almacenada. Cuando una *trace sink* necesita información de una *trace source* se añade como una Callback a una lista de Callbacks de la *trace source* de modo que, cuando la fuente dispone de alguna información invoca las Callbacks almacenadas.

```
using namespace ns3;

class MyObject : public Object
{
public:
    /**
     * Register this type.
     * \return The TypeId.
     */
    static TypeId GetTypeId (void)
    {
        static TypeId tid = TypeId ("MyObject")
            .SetParent<Object> ()
            .SetGroupName ("Tutorial")
            .AddConstructor<MyObject> ()
            .AddTraceSource ("MyInteger",
                "An integer value to trace.",
                MakeTraceSourceAccessor (&MyObject::m_myInt),
                "ns3::TracedValueCallback::Int32");
        ;
        return tid;
    }

    MyObject () {}
    TracedValue<int32_t> m_myInt;
};

void
IntTrace (int32_t oldValue, int32_t newValue)
{
    std::cout << "Traced " << oldValue << " to " << newValue << std::endl;
}

int
main (int argc, char *argv[])
{
    Ptr<MyObject> myObject = CreateObject<MyObject> ();
    myObject->TraceConnectWithoutContext ("MyInteger", MakeCallback (&IntTrace));
    myObject->m_myInt = 1234;
}
```

Ilustración 21: Ejemplo de definición de clases y funciones en programa ns-3

Las *trace source* son instancias de la clase Object dentro de la simulación, de modo que así debemos de tratarlas. En el código de ejemplo que presento a continuación vemos la declaración de una clase *MyObject* que hereda de la clase Object y al que se la añade

una *trace source*.

```
class MyObject : public Object
{
public:
    /**
     * Register this type.
     * \return The TypeId.
     */
    static TypeId GetTypeId (void)
    {
        static TypeId tid = TypeId ("MyObject")
            .SetParent<Object> ()
            .SetGroupName ("Tutorial")
            .AddConstructor<MyObject> ()
            .AddTraceSource ("MyInteger",
                "An integer value to trace.",
                MakeTraceSourceAccessor (&MyObject::m_myInt),
                "ns3::TracedValueCallback::Int32")
        ;
        return tid;
    }

    MyObject () {}
    TracedValue<int32_t> m_myInt;
};
```

Ilustración 22: Declaración de clase MyObject que hereda de la clase Object

Por medio de la clase *template ns3::TracedValue< T >* se declara una variable llamada *m_myInt* de tipo *typedef void(* ns3::TracedValueCallback::Int32) (int32_t oldValue, int32_t newValue)* que toma dos valores *int32_t*, de modo que si queremos crear una función que haga las funciones de *trace sink* esta tendrá que tener la misma firma.

```
void
IntTrace (int32_t oldValue, int32_t newValue)
{
    std::cout << "Traced " << oldValue << " to " << newValue << std::endl;
}
```

Ilustración 23: Función que realiza las tareas de trace sink

Vemos que la función que se muestra tiene la misma firma que el tipo definido para *Int32*. Toma dos valores *int32_t* que serán el valor antiguo y el valor nuevo y simplemente imprime por pantalla ambos valores.

Para conectar la *trace source* y la *trace sink* será necesario crear un puente entre

SIMULADOR NS3

ambos, esto se consigue con el método `bool ns3::ObjectBase::TraceConnectWithoutContext (std::string name, const CallbackBase & cb)` del modo que vemos a continuación:

```
int
main (int argc, char *argv[])
{
    Ptr<MyObject> myObject = CreateObject<MyObject> ();
    myObject->TraceConnectWithoutContext ("MyInteger", MakeCallback (&IntTrace));
    myObject->m_myInt = 1234;
}
```

Ilustración 24: Conexiónb entre trace source y trace sink

A la función se le pasan como argumentos la cadena que identifica a la *trace source*, en este caso MyInteger, y se crea un *callback* mediante el método `MakeCallback` que toma a su vez como argumento la dirección de la función `IntTrace` cuya definición vimos antes. A partir de este momento existe una conexión entre ambas entidades, con lo que un cambio en la variable `m_myInt` forzará la ejecución de la función `IntTrace` y sacará por pantalla los valores antiguo y nuevo. Esto es lo que se fuerza al modificar el valor que contiene la variable `m_myInt` en la última línea del programa.

```
miguel@miguel-N24-25JU:~/ns-3-allinone/ns-3-dev$ ./waf --run scratch/fourth
Waf: Entering directory `/home/miguel/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/home/miguel/ns-3-allinone/ns-3-dev/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (8.022s)
Traced 0 to 1234
```

Ilustración 25: Salida de ejecución del programa.

El método que acabamos de ilustrar no es el único para acceder a los datos facilitados por las *trace source*. Es posible hacerlo también invocando la función `void ns3::Config::Connect (std::string path, const CallbackBase & cb);`; de hecho este es el método más habitual de hacerlo. En este caso se accede a fuentes de información ya definidas (se puede acceder a un listado completo de trace sources en [63]), a las que se llega a través de un path de configuración.

En la siguiente imagen vemos la declaración de una función que hará las tareas de colector de información.

```

static void
CourseChange (std::string foo, Ptr<const MobilityModel> mobility)
{
    Vector pos = mobility->GetPosition ();
    Vector vel = mobility->GetVelocity ();
    std::cout << Simulator::Now () << ", model=" << mobility << ", POS: x=" << pos.x << ", y=" << pos.y
           << ", z=" << pos.z << "; VEL: " << vel.x << ", y=" << vel.y
           << ", z=" << vel.z << std::endl;
}

```

Ilustración 26: Declaración de función para realizar callback.

Esta función se utiliza para realizar un callback a la fuente de rastreo *CourseChange* de la clase *ns3::MobilityModel*:

```

int main (int argc, char *argv[])
{
    Config::SetDefault ("ns3::RandomWalk2dMobilityModel::Mode", StringValue ("Time"));
    Config::SetDefault ("ns3::RandomWalk2dMobilityModel::Time", StringValue ("2s"));
    Config::SetDefault ("ns3::RandomWalk2dMobilityModel::Speed", StringValue ("ns3::ConstantRandomVariable[Constant=1.0]"));
    Config::SetDefault ("ns3::RandomWalk2dMobilityModel::Bounds", StringValue ("0|200|0|200"));

    CommandLine cmd;
    cmd.Parse (argc, argv);

    NodeContainer c;
    c.Create (4);

    MobilityHelper mobility;
    mobility.SetPositionAllocator ("ns3::RandomDlscPositionAllocator",
                                  "X", StringValue ("100.0"),
                                  "Y", StringValue ("100.0"),
                                  "Rho", StringValue ("ns3::UniformRandomVariable[Min=0|Max=30]"));
    mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                              "Mode", StringValue ("Time"),
                              "Time", StringValue ("2s"),
                              "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=1.0]"),
                              "Bounds", StringValue ("0|200|0|200"));

    mobility.InstallAll ();
    Config::Connect ("/$NodeList/*/$ns3::MobilityModel/CourseChange",
                    MakeCallback (&CourseChange));

    Simulator::Stop (Seconds (100.0));
    Simulator::Run ();
    Simulator::Destroy ();
    return 0;
}

```

Ilustración 27: Ejemplo de Config::Connect para conectar con fuente de rastreo.

Vemos que la ruta de configuración, que se pasa como string para el primer argumento del método *Connect*, se define como, */NodeList/*/\$ns3::MobilityModel/CourseChange* donde indicamos a que lugar tiene que ir a realizar el callback dentro de la lista de nodos. En este caso, al incluir el asterisco, le estamos diciendo que vaya a todos los nodos. En el último segmento vemos el atributo *CourseChange* que en este caso es un *trace source*. El segundo argumento es el *callback* realizado por la función *CourseChange* que se ha mostrado antes.

La ejecución del programa nos arrojará la siguiente salida:

SIMULADOR NS3

```
Miguel@miguel-N24-25JU:~/ns-3-allinone/ns-3-dev$ ./waf --run "scratch/main-random-walk"
Waf: Entering directory `/home/miguel/ns-3-allinone/ns-3-dev/build'
[1987/2043] Compiling scratch/main-random-walk.cc
[2009/2043] Linking build/scratch/main-random-walk
Waf: Leaving directory `/home/miguel/ns-3-allinone/ns-3-dev/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (5.365s)
+0.0ns, model=0x55f972e37290, POS: x=99.5906, y=98.3402, z=0; VEL:-0.83308, y=-0.553153, z=0
+0.0ns, model=0x55f972e56ed0, POS: x=92.1842, y=98.6223, z=0; VEL:0.241381, y=-0.97043, z=0
+0.0ns, model=0x55f972e0d640, POS: x=100.093, y=103.149, z=0; VEL:0.675295, y=-0.737548, z=0
+0.0ns, model=0x55f972e2aa30, POS: x=99.8588, y=92.4455, z=0; VEL:-0.897968, y=0.440662, z=0
+2000000000.0ns, model=0x55f972e37290, POS: x=97.9244, y=97.2339, z=0; VEL:-0.0196281, y=0.999807, z=0
+2000000000.0ns, model=0x55f972e56ed0, POS: x=92.667, y=96.6814, z=0; VEL:0.569708, y=0.821847, z=0
+2000000000.0ns, model=0x55f972e0d640, POS: x=101.444, y=101.074, z=0; VEL:0.973606, y=-0.227979, z=0
+2000000000.0ns, model=0x55f972e2aa30, POS: x=98.0629, y=93.3256, z=0; VEL:0.522381, y=0.852712, z=0
+4000000000.0ns, model=0x55f972e37290, POS: x=97.8852, y=99.2335, z=0; VEL:0.242547, y=0.97014, z=0
+4000000000.0ns, model=0x55f972e56ed0, POS: x=93.8064, y=98.3251, z=0; VEL:0.599436, y=0.800423, z=0
+4000000000.0ns, model=0x55f972e0d640, POS: x=103.391, y=101.218, z=0; VEL:0.432898, y=-0.901443, z=0
+4000000000.0ns, model=0x55f972e2aa30, POS: x=99.1077, y=95.031, z=0; VEL:0.686941, y=0.726713, z=0
```

Ilustración 28: Ejecución de programa con rastreo de movilidad

Vemos que se va mostrando la posición y la velocidad de los cuatro nodos definidos en diferentes instantes hasta que se complete el tiempo de ejecución.

- **La herramienta Waf**

A la hora de ejecutar un programa de simulación, como se ha estado haciendo hasta ahora en los ejemplos, se dispone de una herramienta basada en *python* denominada Waf. Esta herramienta ha sido diseñada para configurar, compilar e instalar aplicaciones. De hecho, este es el programa utilizado para configurar el entorno de trabajo ns-3. Lo veremos más adelante cuando se muestre como preparar el entorno de trabajo.

Para obtener ayuda basta con ejecutar *./waf --help* dentro de la carpeta donde tenemos el ejecutable y se nos mostrará por pantalla un extenso manual de ayuda, donde figuran los comandos y las opciones admitidas.

```
miguel@miguel-N24-25JU:~/ns-3-allinone/ns-3-dev$ ./waf --help
waf [commands] [options]

Main commands (example: ./waf build -j4)
  build   : executes the build
  check   : run the equivalent of the old ns-3 unit tests using test.py
  clean   : cleans the project
  configure: configures the project
  dist    : makes a tarball for redistributing the sources
  distcheck: checks if the project compiles (tarball from 'dist')
  docs   : build all the documentation: doxygen, manual, tutorial, models
  doxygen: do a full build, generate the introspected doxygen and then the doxygen
  install : installs the targets on the system
  list   : lists the targets to execute
  shell   : run a shell with an environment suitably modified to run locally built programs
  sphinx : build the Sphinx documentation: manual, tutorial, models
  step   : executes tasks in a step-by-step fashion, for debugging
  uninstall: removes the targets installed
```

Ilustración 29: Comandos admitidos por el programa waf.

De las múltiples opciones disponibles la que se usará de manera más asidua será la opción *waf--run “[ruta al programa a ejecutar] [opciones del programa a ejecutar]”* que es la utilizada para ejecutar un programa. Si queremos conocer las opciones de ejecución de un programa concreto será de interés utilizar la opción *--PrintHelp* como se ve en la siguiente captura.

```
miguel@miguel-N24-25JU:~/ns-3-allinone/ns-3-dev$ ./waf --run "scratch/second" --PrintHelp
Waf: Entering directory `/home/miguel/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/home/miguel/ns-3-allinone/ns-3-dev/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (3.977s)
second [Program Options] [General Arguments]

Program Options:
  --nCsma:      Number of "extra" CSMA nodes/devices [3]
  --verbose:    Tell echo applications to log if true [true]

General Arguments:
  --PrintGlobals:        Print the list of globals.
  --PrintGroups:         Print the list of groups.
  --PrintGroup=[group]:  Print all TypeIds of group.
  --PrintTypeIds:        Print all TypeIds.
  --PrintAttributes=[typeid]: Print all attributes of typeid.
  --PrintHelp:           Print this help message.
```

Ilustración 30: Ejecución de programa con opción PrintHelp

En el caso mostrado el programa admite la introducción de dos opciones de programa, en concreto, *nCsma* y *verbose*. Además se pueden introducir los argumentos generales.

3.3.PREPARACIÓN DEL ENTORNO DE TRABAJO

Para poder empezar a generar nuestras simulaciones debemos preparar el entorno de trabajo ns-3, para ello descargaremos el software del repositorio oficial y comprobamos los requisitos que debemos satisfacer para poder llevar a cabo la instalación de la plataforma. El proceso de instalación que se muestra a continuación se ha realizado en un entorno virtualizado en *VirtualBox* empleando un sistema operativo *Linux Ubuntu 18.04.2 LTS* con *kernel release 4.18.0-15-generic* instalado en una plataforma con arquitectura x86-64 y procesador Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz, a la que se le han asignado 2,4 GiB de memoria RAM y asignación dinámica de espacio en disco.

SIMULADOR NS3

3.3.1. INSTALACIÓN DE REQUISITOS

Los siguientes paquetes de software deben estar disponibles para comenzar con ns-3

REQUISITO	PAQUETE
Compilador C++	clang++ o g++(versión 4.9 o superior)
Python	python2(versión 2.7.10 mínimo) o python3 (versión 3.4 mínimo)
Git	Cualquier versión reciente
tar	Cualquier versión reciente
unzip	Cualquier versión reciente

Tabla 4: Requisitos de instalación ns-3

Además de los programas anteriores se necesitarán las herramientas de desarrollo *qt5* para utilizar *NetAnim*.

En primer lugar instalamos los compiladores c++ y el lenguaje de programación con el comando `$sudo apt-get install gcc g++ python`. Una vez finalizado el proceso de instalación comprobamos que los paquetes se hayan instalado correctamente- para ello utilizamos el comando `which`-.

```
miguel@miguel-VirtualBox:~$ which g++ gcc python python2 python3
/usr/bin/g++
/usr/bin/gcc
/usr/bin/python
/usr/bin/python2
/usr/bin/python3
miguel@miguel-VirtualBox:~$
```

Ilustración 31: Ejecución de comando which

Comprobamos también que las versiones son superiores a las requeridas:

```
miguel@miguel-VirtualBox:~$ python3 --version && python2 --version && g++ --version
Python 3.6.7
Python 2.7.15rc1
g++ (Ubuntu 7.4.0-1ubuntu1~18.04) 7.4.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Ilustración 32: Versiones de paquetes instalados.

A continuación instalamos el programa Git que nos servirá para obtener el código desde los repositorios de GitHub con el comando `$sudo apt-get install python-setuptools git mercurial`.

Las herramientas `tar` y `bunzip2` ya vienen instaladas por defecto en la distribución, de modo que no será necesario instalarlas.

Para finalizar instalamos las herramientas `qt5` con el comando `$sudo apt-get install qt5-default mercurial`.

Por otro lado, además de los requisitos propios de ns-3 trabajamos con un IDE. Para ayudar en las tareas de programación de las simulaciones he elegido *NetBeans* que a pesar de estar pensado para Java admite extensiones para otros lenguajes entre los que está C++.

Para instalar el IDE usamos el comando `$sudo apt-get install netbeans`.

3.3.2.INSTALACIÓN DE NS-3

Una vez hemos preparado el entorno de trabajo -para dar soporte a la instalación- procedemos a instalar ns-3 descargando el código desde el repositorio empleando git. Existen otros métodos, pero a mi juicio este es el más cómodo.

Para ello ejecutamos en consola los siguientes comandos:

```
$cd
```

```
$git clone https://gitlab.com/nsnam/ns3-allinone.git
```

Una vez hemos clonado los datos desde el repositorio vemos que se ha copiado dentro de nuestra carpeta home una carpeta con el nombre *ns-3-allinone*. Si accedemos a la misma vemos el siguiente contenido:

```
miguel@miguel-VirtualBox:~$ cd ns-3-allinone/
miguel@miguel-VirtualBox:~/ns-3-allinone$ ls
build.py constants.py dist.py download.py README util.py
miguel@miguel-VirtualBox:~/ns-3-allinone$ █
```

Ilustración 33: Contenido de carpeta ns-3-allinone

SIMULADOR NS3

Como se puede ver se incluyen varios ejecutables. El siguiente paso será ejecutar el archivo `download.py` que se encargará de descargar la última versión de ns-3 de manera automática. Para ello ejecutamos el comando con `$python download.py`.

Una vez haya finalizado el proceso ya tenemos disponible el entorno de desarrollo. Ahora tan sólo falta el proceso de *build* para ello podemos usar el archivo `build.py`, la herramienta `bake` o `waf`. Hemos optado por utilizar `waf`.

Con anterioridad hemos hecho una referencia a la herramienta `waf` y como se había indicado se dispone de una serie de comandos. En este caso usaremos `clean` y `configure` para llevar a cabo las tareas necesarias para dejar nuestro entorno listo para trabajar.

```
./waf clean
```

```
./waf configure --build-profile=optimized --enable-examples --enable-tests
```

Con el comando `clean` limpiamos el proyecto y con el comando `configure` lo configuramos en función de las opciones introducidas. En este caso, como vemos en la página de ayuda del programa `waf` con `--build-profile` especificamos un perfil de montaje a fin de controlar las *flags* por defecto, en caso de que las banderas `CCFLAGS`/`CXXFLAGS` no estén configuradas.

```
-d BUILD_PROFILE, --build-profile=BUILD_PROFILE
Specify the build profile. Build profiles control the
default compilation flags used for C/C++ programs, if
CCFLAGS/CXXFLAGS are not set in the environment.
[Allowed Values: 'debug', 'release', 'optimized']
```

Ilustración 34: Utilidad de opción `--build-profile`

Mediante `--enable-tests` y `--enable-examples` vemos que se realiza la construcción de los tests y los examples.

```
--enable-tests      Build the ns-3 tests.
--disable-tests    Do not build the ns-3 tests.
--enable-examples  Build the ns-3 examples.
--disable-examples Do not build the ns-3 examples.
```

Ilustración 35: Utilidad de las opciones `--enable-test` y `--enable-examples`

En cuanto se haya ejecutado el comando se generará una carpeta `build` dentro

del proyecto que contendrá los archivos producto de su ejecución. Usando el comando *clean* podremos eliminar las configuraciones anteriores y crear otras en función de las necesidades.

Una vez hemos finalizado con el proceso anterior será interesante utilizar el archivo *test.py* para comprobar que todo funciona correctamente. Dentro de la carpeta *ns3-dev* ejecutamos *./tesy.py*. Este programa efectúa la compilación de todo el código a fin de encontrar algún fallo durante el proceso de construcción, por lo que su ejecución llevará un tiempo.

Este test emplea *waf*. Al finalizar se nos mostrará por pantalla un sumario de las tareas realizadas indicando si todos los test realizados han sido correctos.

Para completar el proceso nos queda configurar el *IDE Apache NetBeans 10.0* que hemos instalado antes para trabajar con el proyecto ns-3. Para ello, en primer lugar, debemos instalar el plugin que nos permita utilizar C++ en el IDE; abrimos la ventana Tools>Plugins y seleccionamos la pestaña Settings. Una vez dentro, debemos activar las casillas de los repositorios y dentro de la pestaña Available Pluggins se pincha sobre Check for Newest y una vez tenemos disponible la lista buscamos el plugin C/C++ y lo instalamos.

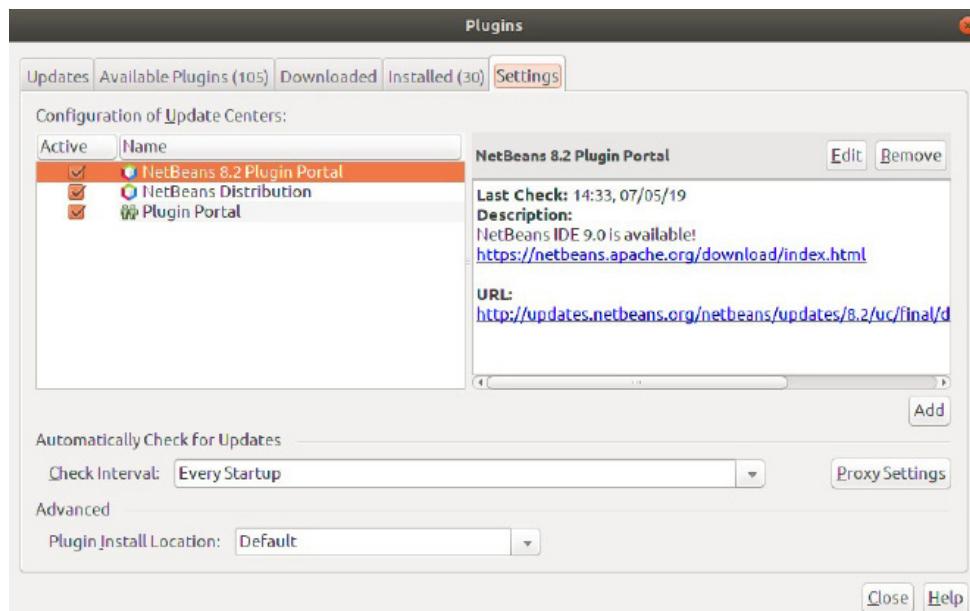


Ilustración 36: Configuración de repositorios NetBeans.

SIMULADOR NS3

El siguiente paso será crear un nuevo proyecto C++ dentro del IDE, para hacer esto abrimos una consola y nos situamos dentro de la capeta ns-3-dev, donde está el ejecutable waf y iniciamos una shell de waf con el comando `./waf shell` y desde esta shell arrancamos Netbeans con `$sh /usr/bin/netbeans`. Una vez se ha abierto el IDE, vamos a File>New Project y una vez dentro seleccionamos la opción *C/C++ Project with Existing Sources* y dejando marcada la casilla *Automatic* seleccionamos la carpeta ns-3-dev pulsando a continuación en finalizar. Ahora tenemos a nuestra disposición las herramientas de Netbeans que nos facilitarán el trabajo a la hora de realizar las simulaciones como las funciones de *Code Assistance* y *debugging*.

DESPLIEGUE DE UNA RED

4.1. INTRODUCCIÓN

Para probar las capacidades del simulador necesitamos un entorno de despliegue sobre el que realizar pruebas de concepto. En nuestro caso, la idea a desarrollar es una infraestructura IoT donde los dispositivos se utilizan para monitorizar las constantes vitales de los internos de una residencia de ancianos. Dadas las limitaciones físicas y en ocasiones cognitivas de los internos es interesante conocer en tiempo real cierta información que permita una atención rápida ante cualquier incidencia. Estos dispositivos podrían estar conectados al mostrador de enfermería. Toda la información recogida en las residencias (si se trata de una cadena) se podría tratar en la nube con fines estadísticos. Por ejemplo, se podría considerar medir los efectos sobre las constantes vitales de los internos ante de cambios en los menús y/o controlar el aumento de la actividad física en función de parámetros ambientales como la humedad, la temperatura ambiente ...

Para realizar la simulación debemos tener en cuenta el entorno donde se realiza el despliegue. La comunicación en zonas de interior, el movimiento de los internos, la necesidad de equipos de tamaño y peso reducidos marcará la selección de la tecnología a emplear, el tipo de canal y los parámetros de movilidad elegidos para la simulación.

4.2. ELECCIÓN DE LA TECNOLOGÍA

DESPLIEGUE DE UNA RED

Como se ha indicado anteriormente, los dispositivos empleados deben ser de pequeño tamaño y preferiblemente de muy bajo consumo. Esto último permitirá utilizar baterías de pequeño tamaño con lo que se consigue una reducción de peso del dispositivo, además de minorizar las necesidades de mantenimiento.

La tecnología elegida es 6LoWPAN por diversos motivos:

- Reducido tamaño de los dispositivos y bajo coste.
- Funcionamiento en bandas ISM inferiores a 1 GHz, con lo que mejora el comportamiento en interiores, a estas frecuencias la señal puede traspasar las paredes sin excesivas pérdidas.
- Integración en redes IP debido a la capa de convergencia 6LoWPAN.
- Empleo de protocolo de red IPv6, lo que pone a nuestra disposición un gran número de direcciones para asignar a los dispositivos.
- Bajo consumo con alcances entre 30 y 100 metros.
- Tasas de transmisión bajas, máximas del orden de 20 Kbps en la banda europea de 868MHz, pero suficientes para nuestra aplicación.

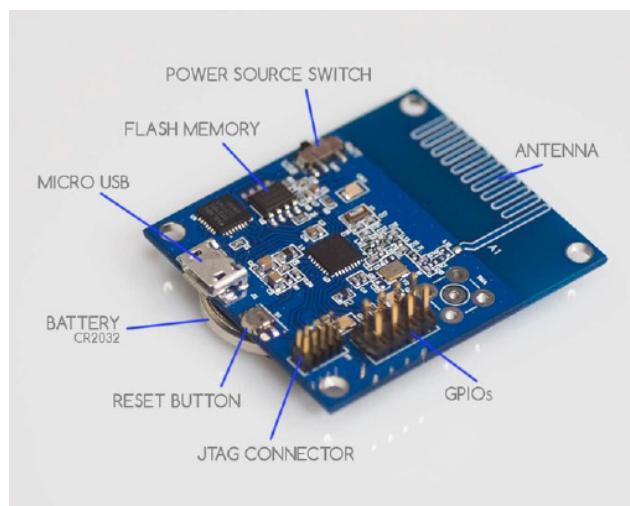


Ilustración 37: Dispositivo 6LoWPAN con antena integrada de reducidas dimensiones [64]

Los datos recogidos por los sensores conectados al dispositivo, como los empleados

para medir el ritmo cardíaco, la temperatura corporal, acelerómetros para medir el número de pasos etc... generarán los datos que se envían por la red para su posterior tratamiento.

4.3.ARQUITECTURA DE LA RED

4.3.1.TOPOLOGÍA

Dadas las condiciones del entorno la solución idónea es una red con topología en estrella para la red de acceso. Los dispositivos en cada planta estarán conectados a un nodo central en el mostrador de enfermería que hará las funciones de gateway y coordinador PAN, por tanto transmitirá los datos recibidos al equipo de sobremesa por ethernet. De existir varias plantas, los nodos centrales de cada planta se conectarán por medio de otra tecnología ethernet a un servidor ubicado en el CPD que enviará los datos a la nube si así se decide.



Ilustración 38: 6LoWPAN gatway [65]

Por lo tanto tendremos un segmento de red-la red de acceso- que funcionará con tecnología 6LoWPAN basada en el estándar IEEE 802.15.4. La red de distribución será Ethernet y el núcleo de la red será dependiente del ISP.

DESPLIEGUE DE UNA RED

4.3.2. PROTOCOLOS

Puesto que centraremos nuestro estudio en la red de acceso dedicaremos este apartado a comentar los protocolos empleados en la tecnología 6LoWPAN y aquellos que hemos elegido para las capas de transporte y aplicación. Haremos un breve repaso a los usados en cada capa:

- **Capa física:**

Las funciones de la capa física serán: la activación y desactivación del transceptor radio, la detección de energía, la indicación de calidad del enlace, la selección de canal, CCA (*clear channel assessment*) y la transmisión y recepción de paquetes. En la capa física se utiliza el protocolo PHY definido en el estándar IEEE 802.14.5. El estándar está diseñado para funcionar en frecuencias libres. En el caso de Europa la banda será la que se encuentra entre 868.0 y 868.6 MHz. La modulación empleada es BPSK con un código de ensanchamiento de 15 chip.

- **Subcapa MAC:**

La capa MAC se encarga de dos tareas principales: el servicio de datos MAC y el servicio de gestión MAC. Este último conecta la capa de gestión MLME (*MAC Sublayer management Entity*) al SAP (*Service Access Point-MLME-SAP-*). Por su parte la capa de datos se encarga de enviar las MPDU a través del servicio de datos de la capa física.

Las funcionalidades que aporta esta capa -además de proveer el enganche con los mecanismos de seguridad- son: la gestión de portadoras, el acceso al canal, la gestión GTS (*guaranteed time slot*), la validación de tramas, la entrega de tramas de confirmación, la asociación y separación.

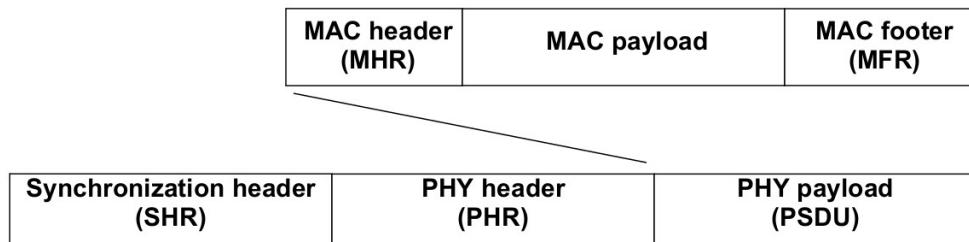


Figure 5-9—Schematic view of the PPDU

Ilustración 39: Estructura de la trama 802.15.4 [19]

- **Capa 6LoWPAN:**

El IETF define este protocolo cuya función es la de adaptar los datagramas IPv6 para pasarlos a las capas inferiores definidas en el estándar 802.15.4, se trata por lo tanto de una capa de convergencia.

La presencia de esta capa es necesaria puesto que la MTU de en IPv6 es de 1280 octetos mientras que en 802.15.4 es de 127 octetos en la capa física, que eliminando la cabecera se queda en 102 octetos, por tanto, la función de esta capa será la fragmentación y el re ensamblado de los datagramas IP.

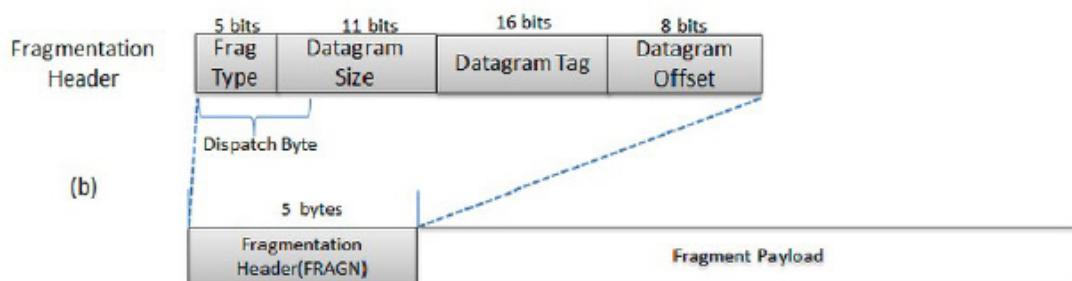
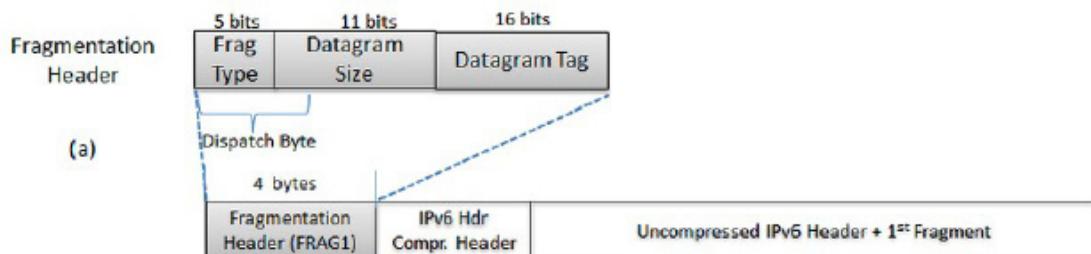


Ilustración 40: Fragmentación 6LoWPAN [66]

DESPLIEGUE DE UNA RED

- **Capa IPv6:**

En la tecnología 6LoWPAN se utiliza direccionamiento IPv6, al contrario que IPv4 donde se dispone de 4 octetos, en este protocolo se dispone de 16 octetos. La cabecera IPv6 tiene un tamaño total de 40 bytes.

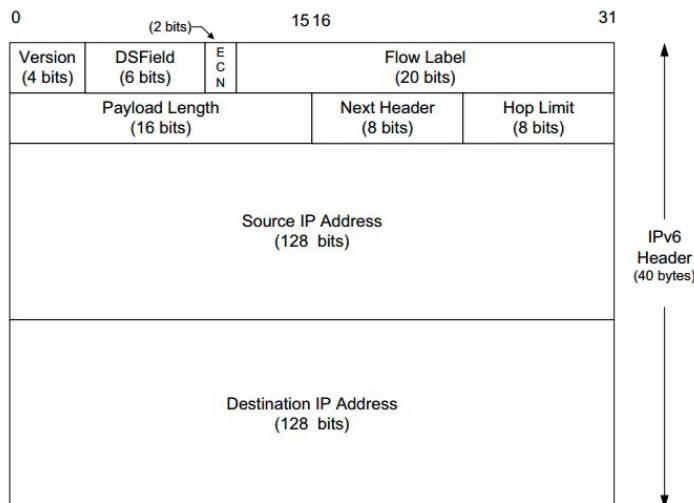


Ilustración 41: Cabecera IPv6 [67]

Además de los campos que vemos en el gráfico anterior se pueden añadir cabeceras de extensión. El tipo de la misma vendrá indicado en el campo *next header*. Estos nuevos campos pueden emplearse para las cabeceras AH o ESP, parámetros de fragmentación etc. El campo *next header* de estas nuevas capas indicará el tipo de protocolo de la capa superior.

- **Capa UDP:**

UDP es un protocolo de transporte no orientado a conexión, por tanto, tiene una cabecera de pequeño tamaño - en concreto, 32 bits-. Optamos por utilizar UDP debido a que el tamaño reducido de las cabeceras implicará un ahorro de energía al enviar una menor cantidad de datos por el canal. Además, dado que se tratan datos referidos a parámetros de interés médico es interesante que el flujo de datos sea en tiempo real y el comportamiento de UDP es mejor en estas circunstancias. Por otro lado, es preferible asumir la pérdida de algún paquete que demorarse en retransmisiones.

La cabecera en UDP es muy sencilla. Incluye los campos de la dirección origen,

la dirección destino, un campo para especificar la longitud y otro campo que incluye el *checksum* para identificar datagramas UDP erróneos, cuyo uso será obligatorio al emplear IPv6.

- **Capa CoAP:**

El protocolo CoAP está definido en RFC 7252 del IETF [68] y está pensado para dispositivos con limitaciones y para tasas de transmisión bajas con cargas muy pequeñas, como es el caso que nos ocupa. Se puede considerar CoAP como una modificación del protocolo *RESTfull* muy eficiente. Por otro lado, al hacer las peticiones de un modo parecido a como lo hace *RESTfull* es sencillo implementar un *proxy* que traduzca las peticiones a un protocolo como http.

La pila definitiva de protocolos queda como vemos en la siguiente tabla:

CoAP
UDP
IPv6
6LoWPAN
802.15.4 MAC
802.15.4 PHY

El equipo coordinador tendrá dos pilas: una como la anterior para la interfaz con la red lrwpan y otra ethernet para conectar con la red del edificio. Esta queda como se ve a continuación.

CoAP
UDP
IPv4
802.3 MAC (ethernet)
802.3 PHY (ethernet)

DESPLIEGUE DE UNA RED

Un diseño más visual de la red puede ser el mostrado en la ilustración siguiente, donde se ha recreado un escenario con dos plantas con sus respectivas redes conectados a un CPD y desde aquí a un servicio en la nube a través de Internet.

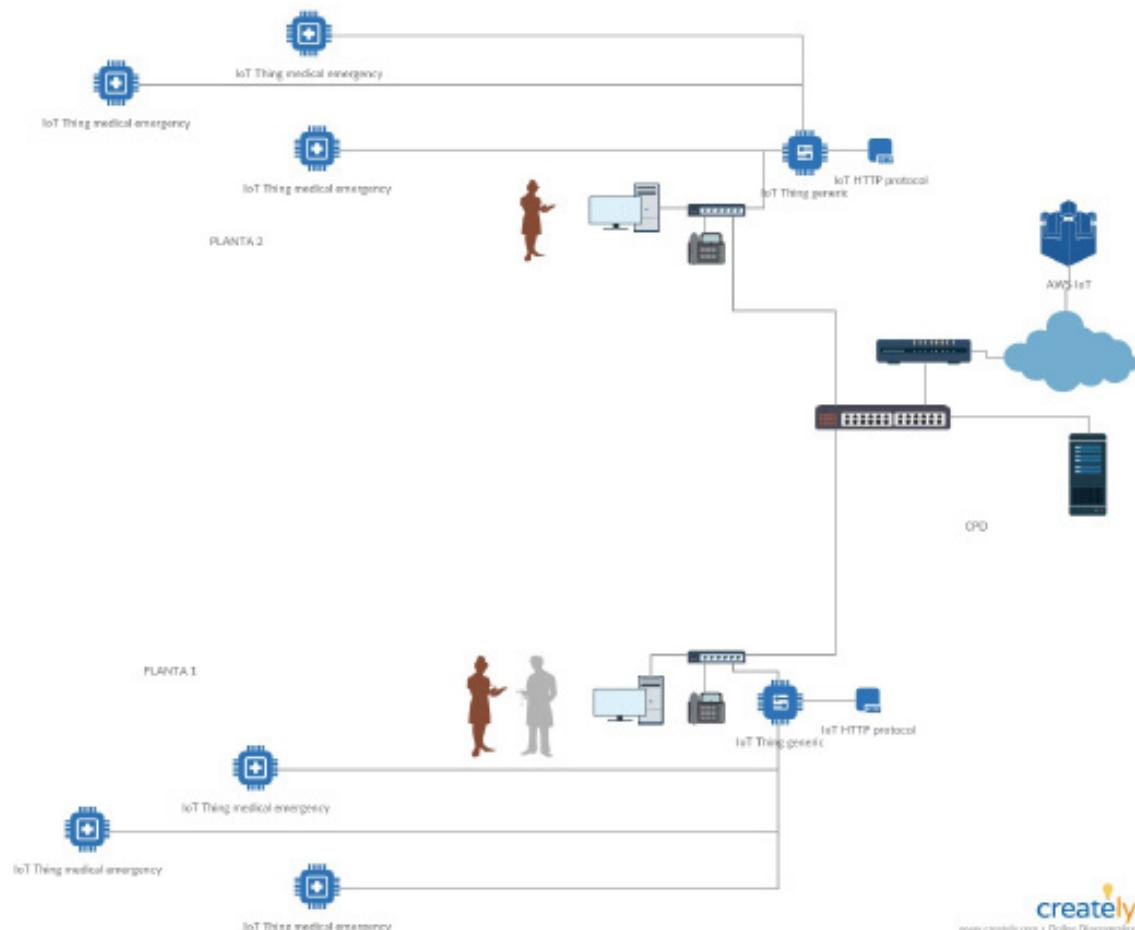


Ilustración 42: Representación del escenario propuesto para el despliegue

SIMULACIÓN DEL DISEÑO EN NS-3 Y OBTENCIÓN DE RESULTADOS

5.1.INTRODUCCIÓN

A lo largo de los siguientes apartados se desarrollará una simulación en NS-3 para la conexión entre los dispositivos 6LoWPAN y su nodo gateway. Se tendrá en cuenta que la comunicación de estos dispositivos se realiza en interior, por lo que se modelará un edificio y se añadirá movilidad a los nodos para efectuar la simulación. Además, se implementará en todos los nodos la pila de protocolos y se podrá obtener información relevante de la simulación, como un fichero pcap que captura los paquetes en el nodo gateway y otra información que se sacará por pantalla como: la caída de potencia entre el origen y el destino, el estado de la tarjeta inalámbrica o los paquetes perdidos durante la transmisión. Se realizarán comprobaciones para diferentes modelos de propagación y se observarán los resultados.

Para llevar a cabo la simulación se usará ns-3 y los módulos asociados que nos permitan desarrollar la pila de protocolos mostrada en el apartado anterior. Se implementarán las funciones necesarias para la realización de los Callback y obtención de información.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

5.2.EL PROGRAMA

A continuación iremos desarrollando las diferentes partes del programa de simulación y dando una pequeña pincelada sobre las diferentes partes que lo conforman.

- **Directivas include**

En la siguiente captura vemos las directivas include utilizadas para añadir los archivos de encabezado necesarios para la implementación de la simulación. Además vemos que se usará el espacio de nombres ns3 y definimos el *logging* para el programa tfg, que así se ha llamado a la simulación.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/lr-wpan-module.h"
#include "ns3/sixlowpan-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/propagation-module.h"
#include "ns3/buildings-module.h"
#include "ns3/netanim-module.h"
#include "ns3/spectrum-module.h"
#include "ns3/stats-module.h"
#include "ns3/gnuplot.h"
#include "ns3/gnuplot-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("tfg");
```

Ilustración 43: Directivas include

- **Definición de funciones para trazado de información**

A continuación se muestra la definición e implementación de algunas funciones que se usarán para obtener información de la simulación.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

```
static void
CourseChange (std::string contexto, Ptr<const MobilityModel> mobilidad)
{
    Vector pos = mobilidad->GetPosition ();
    Vector vel = mobilidad->GetVelocity ();
    std::cout << Simulator::Now () << ", modelo=" << mobilidad << ", POSICION: x=" << pos.x << ", y=" << pos.y
    << ", z=" << pos.z << "; VELOCIDAD: " << vel.x << ", y=" << vel.y
    << ", z=" << vel.z << std::endl;
}
static void
PathLoss (std::string contexto ,Ptr< const SpectrumPhy > txPhy, Ptr< const SpectrumPhy > rxPhy, double lossDb)
{
    NS_LOG_UNCOND (Simulator::Now()<<"-----" <<"La pérdida en dB es:"<< lossDb);
}

static void
EstadoTarjeta (std::string contexto ,ns3::Time time ,LrWpanPhyEnumeration oldValue, LrWpanPhyEnumeration newValue)
{
    NS_LOG_UNCOND ("Estado antiguo:" << LrWpanHelper::LrWpanPhyEnumerationPrinter (oldValue) <<
        " Estado nuevo: " << LrWpanHelper::LrWpanPhyEnumerationPrinter (newValue));
}

double contador=0;
static void
PaquetesPerdidos (std::string contexto ,Ptr< const Packet > packet)
{
    contador++;
    std::cout<<Simulator::Now()<<"Paquetes caídos en transmisión:"<<contador<<std::endl;
}
```

Ilustración 44: Implementación funciones de tracing

Con *CourseChange* obtenemos datos de movilidad de los nodos durante la ejecución. Con *PathLoss* tendremos acceso a información sobre la caída de de potencia durante la transmisión entre diferentes nodos. *EstadoTarjeta* nos devolverá el estado de la tarjeta inalámbrica cuando se produzca un cambio de estado en la misma. Y finalmente, *PaquetesPerdidos* implementa un contador que nos irá sacando por pantalla el número de paquetes perdidos en el canal durante la transmisión.

- **Variables y línea de comandos**

En las siguientes líneas se definen una serie de variables que nos permitirán elegir de que modo se comprobará el programa en función de los valores introducidos a la hora de ejecutar el mismo.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

```

int main(int argc, char** argv) {
    CommandLine cmd;
    bool verbose = true;
    uint32_t nNodos = 5;
    bool tcp = false;
    bool disc = false;
    bool box = false;
    bool interiores = false;
    bool verbose_phy = false;
    bool verbose_mob = false;
    uint8_t semilla = 1;
    double experd = 3;

    cmd.AddValue("nNodos", "Número de nodos", nNodos);
    cmd.AddValue("verbose", "Activar verbose", verbose);
    cmd.AddValue("verbose_phy", "Activar verbose capa física", verbose_phy);
    cmd.AddValue("verbose_mob", "Activar verbose movilidad", verbose_mob);
    cmd.AddValue("tcp", "Uso de TCP si true, flase por defecto", tcp);
    cmd.AddValue("disc", "Position allocator disc", disc);
    cmd.AddValue("box", "Position allocator box", box);
    cmd.AddValue("interiores", "Con true se genera un modelo de propagación en interiores", interiores);
    cmd.AddValue("semilla", "Si se introduce nuevo valor de semilla cambia la ejecución, no usar 0;", semilla);
    cmd.AddValue("experd", "Introducir el valor del expoente de perdidas para single spectrum channel, por defecto 3", experd);

    cmd.Parse(argc, argv);
}

```

Ilustración 45: Variables de línea de comandos.

Todas las variables se han declarado con un valor por defecto. A continuación se muestra el resultado de ejecutar el programa con la opción --PrintHelp.

```

miguel@miguel-N24-25JU:~/ns-3-allinone/ns-3-dev$ ./waf --run "tfg --PrintHelp"
Waf: Entering directory '/home/miguel/ns-3-allinone/ns-3-dev/build'
[1995/2050] Compiling scratch/tfg.cc
[2013/2050] Linking build/scratch/tfg
Waf: Leaving directory '/home/miguel/ns-3-allinone/ns-3-dev/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (7.480s)
tfg [Program Options] [General Arguments]

Program Options:
  --nNodos:      Número de nodos [5]
  --verbose:      Activar verbose [true]
  --verbose_phy: Activar verbose capa física [false]
  --verbose_mob: Activar verbose movilidad [false]
  --tcp:          Uso de TCP si true, flase por defecto [false]
  --disc:         Position allocator disc [false]
  --box:          Position allocator box [false]
  --interiores:  Con true se genera un modelo de propagación en interiores [false]
  --semilla:      Si se introduce nuevo valor de semilla cambia la ejecución [1]
  --experd:       Introducir el valor del expoente de perdidas para single spectrum channel [3]

General Arguments:
  --PrintGlobals:      Print the list of globals.
  --PrintGroups:       Print the list of groups.
  --PrintGroup=[group]: Print all TypeIds of group.
  --PrintTypeIds:     Print all TypeIds.
  --PrintAttributes=[typeid]: Print all attributes of typeid.
  --PrintHelp:         Print this help message.

```

Ilustración 46: Opciones de ejecución.

- **Condicional Verbose**

Si se pone la opción *verbose* a *true* obtenemos información del servidor Udp y de la capa física *lrwpan*. Además de activar el logging del programa, este parámetro se

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

pone por defecto *en true* pues se muestra un mensaje cuando finaliza la ejecución y será de interés para depurar posibles errores de ejecución. En ocasiones, por cuestiones de claridad, puede resultar de interés desactivarlo o incluso poner como comentario alguno de los logs con //.

```
if (verbose){  
    LogComponentEnable ("tfg", LOG_LEVEL_INFO);  
    LogComponentEnable ("UdpServer",LOG_LEVEL_INFO);  
    LogComponentEnable ("LrWpanPhy",LOG_LEVEL_INFO);  
}
```

Ilustración 47: Condicional verbose.

En el siguiente recorte se muestra el final de la ejecución del programa con los logs activados:

```
TraceDelay: RX 1012 bytes from 2001::ff:fe00:5 Sequence Number: 57 UId: 3277 TXtime: +590680000000.0ns RXtime: +59164995293.0ns Delay: +164995293.0ns  
turn on PHY_TX_ON  
0x560aad1f0800 receiving packet with power: -87.3741dBm  
0x560aad1f1c00 receiving packet with power: -88.7949dBm  
0x560aad1f1d00 receiving packet with power: -88.9589dBm  
0x560aad1f1e00 receiving packet with power: -92.4029dBm  
0x560aad1f3200 receiving packet with power: -95.2119dBm  
Packet successfully transmitted  
Fin de ejecución
```

- **Condicional para número de nodos y configuración se semilla**

Mediante el condicional indicamos que en caso de que por teclado se introduzcan el valor nNodos como 0 se le asignen 4 nodos.

```
nNodos = nNodos == 0 ? 1:nNodos; //Si nnodos es igual a 0 nnodos=1 else nnodos=4 (valor por defecto)  
RngSeedManager::SetSeed(semilla);
```

Ilustración 48: Condicional para no asignar 0 nodos y configuración de semilla.

Por otro lado mediante la ejecución de la función *SetSeed* configuramos la semilla en función del valor introducido por teclado. Si este valor no se modifica, cada vez que realicemos una simulación con los mismos parámetros los valores de salida de la misma no cambiarán. Por tanto, modificando el valor de la semilla añadimos aleatoriedad a la ejecución.

- **Creación de contenedores de nodos.**

El siguiente paso es crear los contenedores en los que ubicaremos todos los dispositivos, protocolos etc. Creamos dos contenedores, en uno de ellos albergaremos el

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

dispositivo *gateway* y en el otro los dispositivos *lrwpan* que están asociados al mismo punto de acceso. Finalmente ambos contenedores se concatenan en uno sólo permitiéndonos así implementar elementos comunes a ambos contenedores.

```
/*-----Creamos nodos LrWPAN-----*/
NodeContainer nodoGateway;
NodeContainer nodosLrwpanTer;
nodoGateway.Create(1);
nodosLrwpanTer.Create(nNodos);
NodeContainer nodosLrwpan(nodoGateway,nodosLrwpanTer);
```

Ilustración 49: Configuración de nodos LrWPAN

- **Configuración de planta del edificio**

Para simular la comunicación en interiores - como es el caso que nos ocupan debemos crear una infraestructura donde ubicar los nodos. De esto se encargan las siguientes líneas de código donde generamos la planta de un edificio que situamos en el origen de coordenadas, con 15x40 metros de área y 4 metros de altura que constará de 2 habitaciones a lo largo del eje x y 6 en el eje y. Se trata de una infraestructura residencial y con muros externos de cemento con ventanas.

```
/*-----Creamos edificio e instalamos los nodos en el-----*/
double x_min = 0.0;
double x_max = 15.0;
double y_min = 0.0;
double y_max = 40.0;
double z_min = 0.0;
double z_max = 4.0;
Ptr<Building> edificio = CreateObject <Building> ();
edificio->SetBoundaries (Box (x_min, x_max, y_min, y_max, z_min, z_max));
edificio->SetBuildingType (Building::Residential);
edificio->SetExtWallsType (Building::ConcreteWithWindows);
edificio->SetNFloors (1);
edificio->SetNRoomsX (2);
edificio->SetNRoomsY (6);
```

Ilustración 50: Configuración de planta

- **Generar patrones de movilidad para los nodos**

El siguiente objetivo es generar diferentes patrones de movilidad para hacer pruebas.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

En estos patrones se define donde se colocan los nodos al iniciar la simulación y se decide también de que modo se mueven por el escenario. Se generarán tres escenarios diferentes que se pueden seleccionar al inicio de la simulación.

El primero de ellos distribuye los nodos mediante una clase denominada *RandomDiscPositionAllocator* que inicialmente colocará los nodos conformando un disco cuyo centro situaremos en el punto que decidamos y los va ubicando según una variable aleatoria a la que le daremos el valor máximo. También permite añadir otra variable aleatoria que distribuye los ángulos, pero hemos decidido prescindir de su uso.

Por otro lado se define también la movilidad de los nodos mediante la función *RandomWalk2dMobilityModel*, donde configuramos atributos como la variable aleatoria que se usará para determinar la velocidad del movimiento. En el caso que nos ocuparemos una función que genera un valor constante de manera aleatoria, la clase es *ConstantRandomVariable*. Otros parámetros que se definen son los límites por los que se pueden mover los nodos o el tiempo de refresco de la posición- 500 ms,en este caso- Indicar que el nodo *gateway* se configura de modo que este permanentemente en la posición 0,0 (el origen de coordenadas). Y así será en todos los patrones.

```
if(disc){  
    MobilityHelper movilidad_gateway;  
    movilidad_gateway.Install(nodoGateway);  
  
    MobilityHelper movilidad_nodos;  
    movilidad_nodos.SetPositionAllocator ("ns3::RandomDiscPositionAllocator","X", StringValue ("10.0"), "Y", StringValue ("30.0"),  
                                         "Rho", StringValue ("ns3::UniformRandomVariable[Min=0|Max=10]"));  
    movilidad_nodos.SetMobilityModel ("ns3::RandomWalk2dMobilityModel","Mode", StringValue ("Time"), "Time", StringValue ("500ms"),  
                                     "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=2.0]"),  
                                     "Bounds", StringValue ("0|15|0|40"));  
  
    movilidad_nodos.Install (nodosLwpnTer);  
}  
//
```

Ilustración 51: Definición de opción de movilidad --interiores

Otra opción es la de usar la distribución *RandomBoxPositionAllocator*. En este caso, los nodos se colocan de manera aleatoria en forma de caja, donde los valores de las coordenadas hemos decidido generarlos usando una función de distribución uniforme para ambas coordenadas. Asignando los valores con cuidado para los máximos y así no exceder los límites del patrón de movilidad que provocaría un error al ejecutar el código.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

El patrón de movilidad sigue siendo *RandomWalk2dMobilityModel* pues es el ideal para el tipo de simulación (peatones caminando), pero en esta ocasión hemos decidido modificar la distribución de la velocidad, que vendrá definido por una distribución normal -de media 1, varianza 4 y límite máximo 10 -.

```
else if(box){
    MobilityHelper movilidad_gateway;
    movilidad_gateway.Install(nodoGateway);

    MobilityHelper movilidad_nodos;
    movilidad_nodos.SetPositionAllocator ("ns3::RandomBoxPositionAllocator","X",StringValue (" ns3::UniformRandomVariable[Min=0.0|Max=25]"),
                                         "Y",StringValue (" ns3::UniformRandomVariable[Min=0.0|Max=75.0]"));
    movilidad_nodos.SetMobilityModel ("ns3::RandomWalk2dMobilityModel","Mode", StringValue ("Time"), "Time", StringValue ("500ms"),
                                     "Speed", StringValue ("ns3::NormalRandomVariable[Mean=1|Variance=4|Bound=10]"),
                                     "Bounds", StringValue ("0|15|0|40"));

    movilidad_nodos.Install (nodosLrwpanTer);
}
```

Ilustración 52: Definición de la opción de movilidad --box

Finalmente, simularemos aquel caso más próximo al escenario real. En este caso la ubicación de los nodos y el patrón de movilidad irán en consonancia con el edificio en el que realizamos la simulación. Esto se consigue empleando una clase que toma en consideración el edificio que se ha generado anteriormente e irá ubicando los nodos en posiciones que sean consistentes con ese escenario. Para ello se ha usado *RandomBuildingPositionAllocator* que elige un escenario de modo aleatorio de la lista de edificios y ubica los nodos en el mismo también de manera aleatoria. En este caso sólo se ha definido un edificio. Se tomará este de la lista.

El patrón de movilidad no difiere demasiado de los casos anteriores.

```
else{
    MobilityHelper movilidad_gateway;
    movilidad_gateway.Install(nodoGateway);

    MobilityHelper movilidad_nodos;
    movilidad_nodos.SetPositionAllocator ("ns3::RandomBuildingPositionAllocator");
    movilidad_nodos.SetMobilityModel ("ns3::RandomWalk2dMobilityModel","Mode", StringValue ("Time"), "Time", StringValue ("500ms"),
                                     "Speed", StringValue ("ns3::ConstantRandomVariable[Constant=2.0]"),
                                     "Bounds", StringValue ("0|15|0|40"));

    movilidad_nodos.Install (nodosLrwpanTer);
}
```

Ilustración 53: Definición de la opción de movilidad --interiores.

- **Instalar nodos con movilidad en el edificio**

Una vez se ha dotado a los nodos de movilidad estos se instalan en el edificio y se hace que el patrón de movilidad sea consistente con el edificio. Para hacernos una idea,

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

```
/*-----Instalamos nodos con movilidad en edificio-----*/
BuildingsHelper::Install (nodosLrwpan);
BuildingsHelper::MakeMobilityModelConsistent();
```

Ilustración 54: Instalación de nodos en el edificio.

- **Definición de características del dispositivo físico, modulación y creación de dispositivos.**

A continuación configuramos las características en cuanto a modulación. Esta selección se efectúa teniendo dos valores en consideración: el canal y la página. Ambos valores se obtienen mirando la declaración del método privado *SetMyPhyOption* de la clase *LrWpanPhy* y modificando los valores con una función que declaramos como se ve en la siguiente captura:

```
void PasarValorCanal(LrWpanPhyPibAttributes atributos,int canal, int page)
{
    atributos.phyCurrentChannel=canal;
    atributos.phyCurrentPage=page;
}
```

Ilustración 55: Función para modificar atributos.

Como vemos pasamos como atributos un objeto de clase *LrWpanPhyPibAttributes* y dos enteros con el número del canal y de la página. En este caso se ha seleccionado el canal 0 (869 MHz) y el valor de page 0, como se puede ver en el siguiente recorte.

Además de la configuración de estos parámetros se crea una variable con el *helper lrwpan* que se usará para configurar posteriormente los canales de propagación.

```
/*-----Creamos dispositivos LrWpan-----*/
//Configuramos para canal 0 IEEE_802_15_4_868MHZ_BPSK (0 y 0) IEEE_802_15_4_868MHZASK (0 y 1) IEEE_802_15_4_868MHZ_QQPSK (1 y 2)
LrWpanPhyPibAttributes atributos;
uint8_t canal =0;
uint32_t page =2;
PasarValorCanal(atributos,canal,page);

LrWpanHelper lrWpanHelper;
```

Ilustración 56: Modificación de variables canal y page

Si nos fijamos en la declaración del método privado *SetMyPhyOption*, esta combinación de valores empleará la opción de modulación IEEE_802_15_4_868Mhz_

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

OQPSK.

```
LrWpanPhy::SetMyPhyOption (void)
{
    NS_LOG_FUNCTION (this);

    m_phyOption = IEEE_802_15_4_INVALID_PHY_OPTION;

    if (m_phyPIBAttributes.phyCurrentPage == 0)
    {
        if (m_phyPIBAttributes.phyCurrentChannel == 8)
        {
            // 868 MHz BPSK
            m_phyOption = IEEE_802_15_4_868MHZ_BPSK;
        }
        else if (m_phyPIBAttributes.phyCurrentChannel <= 18)
        {
            // 915 MHz BPSK
            m_phyOption = IEEE_802_15_4_915MHZ_BPSK;
        }
        else if (m_phyPIBAttributes.phyCurrentChannel <= 26)
        {
            // 2.4 GHz MHz O-QPSK
            m_phyOption = IEEE_802_15_4_2_4GHZ_OQPSK;
        }
    }
    else if (m_phyPIBAttributes.phyCurrentPage == 1)
    {
        if (m_phyPIBAttributes.phyCurrentChannel == 8)
        {
            // 868 MHz ASK
            m_phyOption = IEEE_802_15_4_868MHZ_ASK;
        }
        else if (m_phyPIBAttributes.phyCurrentChannel <= 18)
        {
            // 915 MHz ASK
            m_phyOption = IEEE_802_15_4_915MHZ_ASK;
        }
    }
    else if (m_phyPIBAttributes.phyCurrentPage == 2)
    {
        if (m_phyPIBAttributes.phyCurrentChannel == 8)
        {
            // 868 MHz O-QPSK
            m_phyOption = IEEE_802_15_4_868MHZ_OQPSK;
        }
        else if (m_phyPIBAttributes.phyCurrentChannel <= 18)
        {
            // 915 MHz O-QPSK
            m_phyOption = IEEE_802_15_4_915MHZ_OQPSK;
        }
    }
}
```

Ilustración 57: Método SetMyPhyOptoin

- Creación de canal de propagación en interiores y canal alternativo

El siguiente paso es crear el canal de propagación en interiores. Este modelo tendrá en cuenta el edificio, las paredes, el tipo de muro exterior y otros parámetros que definimos en su momento. La clase empleada para ello es *HybridBuildingsPropagationLossModel*, donde se manipulan parámetros como la frecuencia, la pérdida en los muros interiores, los valores de la desviación estándar para el desvanecimiento por sombra interior y exterior, además de otros parámetros que no se han modificado.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

```
/*-----Con esta opción añadimos canal de propagación en interiores-----*/
if(intiores){
SpectrumChannelHelper channelHelper = SpectrumChannelHelper::Default ();
channelHelper.AddPropagationLoss("ns3::HybridBuildingsPropagationLossModel","ShadowSigmaOutdoor",DoubleValue(7),"ShadowSigmaIndoor",DoubleValue(3),
"ShadowSigmaExtWalls",DoubleValue(7),"InternalWallLoss",DoubleValue(2.5),"Frequency",DoubleValue(869e6));
//channelHelper.AddSpectrumPropagationLoss("ns3::FriisSpectrumPropagationLossModel");
channelHelper.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
lwpnHelper.SetChannel(channelHelper.Create());
}
```

Ilustración 58: Definición de canal de propagación en interiores.

Por otro lado, en caso de no seleccionar la opción de interiores, el otro canal que se usará es el que se instala por defecto en los dispositivos LrWPAN. En este caso se trata de un canal con un modelo pérdidas de tipo *LogDistancePropagationLossModel*.

```
else{
/*-----Configuramos canal single spectrum-----*/
Config::Set("/NodeList/*/$DeviceList/*/$ns3::LrWpanNetDevice/Channel/PropagationLossModel/$ns3::LogDistancePropagationLossModel/Exponent",DoubleValue(experd));
}
/*-----Añadimos protocolo 6lowpan a la pila-----*/
NetDeviceContainer lwpnDevices = lwpnHelper.Install(nodosLwpn);
```

Ilustración 59: Configuración de exponente de pérdidas.

Hemos dejado una opción de configuración para este canal, donde se puede definir el exponente de pérdidas. La ecuación que define el comportamiento de este modelo es la siguiente:

$$L=L_0+10n\log_{10}(d/d_0)$$
 El valor del exponente de pérdidas es n

- Añadir capa de convergencia 6LoWPAN, asociar los nodos a la misma PAN e instalar pila TCP/IP**

Hasta ahora nos hemos centrado en las capas física y de enlace, ambas definidas en el estándar IEEE 802.15.4 2006 que es en el que se basa el modelo implementado en NS-3. Para que la comunicación se pueda realizar tenemos que añadir el resto de protocolos para completar la pila. Se añaden por lo tanto desde abajo hacia arriba las diferentes capas como definimos en el apartado correspondiente a protocolos.

En primer lugar se instalan los dispositivos LrWPAN creados con el helper a los nodos quedando definido un contenedor con los dispositivos de red a los que se añade el protocolo de la capa de convergencia 6LoWPAN. A continuación se asocian todos

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

los nodos a la misma PAN y se instala el stack TCP/IP que implementará las capas de transporte y de red. Para la capa de transporte se generan tanto una capa TCP como una capa UDP.

```
/*-----Añadimos protocolo 6lowpan a la pila-----*/
NetDeviceContainer lrWpanDevices = lrWpanHelper.Install(nodosLrwpan);

SixLowPanHelper sixLowPanHelper;
NetDeviceContainer sixLowPanDevices = sixLowPanHelper.Install (lrWpanDevices);

/*-----Asociar a misma PAN,añadir stack internet y capa 6lowpan a los nodos lrwpan-----*/
lrWpanHelper.AssociateToPan (lrWpanDevices, 0); //Asociamos nodos a la misma PAN

InternetStackHelper internetStackHelper;
internetStackHelper.Install (nodosLrwpan);
```

Ilustración 60: Añadiendo protocolos a la pila.

- **Asignación de IP's IPv6 a los dispositivos LrWPAN**

Siguiendo con el trabajo de configuración se deben asignar direcciones IP a los diferentes equipos. Para ello se usa el helper *Ipv6AddressHelper* y se asigna una máscara de subred de 64 bits y las direcciones comenzarán a asignarse desde la dirección 2001::.

Las direcciones generadas se asignan a los dispositivos y se almacenan en un contenedor de interfaces de red. El resultado de la asignación se imprime por la salida estándar usando un bucle.

```
/*-----Añadimos direcciones ipv6 a los dispositivos lrwpan-----*/
Ipv6AddressHelper ipv6AddressHelper;
ipv6AddressHelper.SetBase (Ipv6Address ("2001::"), Ipv6Prefix (64));
Ipv6InterfaceContainer sixLowPanInterfaces = ipv6AddressHelper.Assign (sixLowPanDevices);

for (uint32_t i = 0; i < sixLowPanInterfaces.GetN(); ++i){
    std::cout<<"Direcciones de dispositivos lrwpan:"<<std::endl;
    std::cout<< sixLowPanInterfaces.GetAddress(i,1)<<std::endl;
}
```

- **Definición de la capa de aplicación**

Llegados a este punto sólo nos resta instalar nuestras aplicaciones en los nodos.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

Hemos decidido tener la opción de simular dos tipos de tráfico, uno basado en transporte TCP y otro basado en transporte UDP, siendo este último el que habíamos elegido para nuestro diseño, pero con fines de estudio se incorporan ambas opciones.

Para el caso del tráfico TCP la capa de aplicación se definirá por medio de unos helper que generan una aplicación que simula tráfico HTTP. La clase empleada es *ThreeGppHttpServerHelper* para el lado servidor y *ThreeGppHttpClientHelper* para el lado cliente.

Una vez se han creado se instalan en sus respectivos contenedores de aplicación y se indica en qué momento de la simulación se arranca la aplicación y en qué momento se detiene la misma. El tráfico generado es similar al que se genera al utilizar el navegador y realizar peticiones de páginas web y se ha de indicar a los nodos cliente la dirección IP del nodo servidor.

```
/*-----Arrancamos aplicaciones-----*/
if(tcp){ //Intercambio TCP
    /*-----Añadimos servidor http en nodo PAN Coordinador-----*/
    ThreeGppHttpServerHelper serverhttpHelper (sixLowPanInterfaces.GetAddress(0,1));
    ApplicationContainer serverHttpApp = serverhttpHelper.Install(nodosLrwpan.Get(0));
    /*-----Añadimos clientes http en los nodos End-Device-----*/
    ThreeGppHttpClientHelper clienthttpHelper (sixLowPanInterfaces.GetAddress(0,1));
    ApplicationContainer clientHttpApp = clienthttpHelper.Install(nodosLrwpanTer);
    serverHttpApp.Start(Seconds (1.));
    serverHttpApp.Stop(Minutes (1.));

    clientHttpApp.Start(Seconds (1.));
    clientHttpApp.Stop(Minutes (1.));
}
```

Ilustración 61: Instalación de aplicaciones HTTP.

Para la otra opción se utiliza una aplicación que simula tráfico UDP. La ausencia de un modulo específico que simule la capa de aplicación CoAP nos ha llevado a tener que utilizar esta otra solución alternativa. Se utilizan como en el caso anterior dos helper, uno para el servidor y otro para el cliente, *UdpServerHelper* y *UdpClientHelper*

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

respectivamente.

Como en el caso anterior se instalan las aplicaciones en sus contenedores respectivos para a continuación iniciarlas. En este caso se ha activado el atributo *ChecksumEnabled* para que la simulación calcule las sumas de comprobación de los datagramas UDP en busca de errores.

```
else{ //Intercambio UDP
    GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));
    UdpServerHelper udpServerHelper(5683);
    UdpClientHelper udpClientHelper(sixLowPanInterfaces.GetAddress(0,1),5683);

    ApplicationContainer serverUdp =udpServerHelper.Install(nodosLrwpan.Get(0));
    ApplicationContainer clientUdp =udpClientHelper.Install(nodosLrwpanTer);

    serverUdp.Start(Seconds (1.));
    serverUdp.Stop(Seconds (60.));

    clientUdp.Start(Seconds (2.));
    clientUdp.Stop(Seconds (60.));
}
```

Ilustración 62: Instalación de aplicaciones UDP.

En el lado servidor se configura el puerto y en el lado cliente se facilita el *socket* remoto sobre el que efectuar la conexión. En este caso la dirección del servidor y el puerto 5683, que es el que utiliza la aplicación CoAP.

- **Habilitar tracing, obtención de información mediante callbacks y generar fichero de animación**

Con la ejecución del método *EnablePcapIpv6* creamos un archivo *pcap* que se genera indicando mediante un puntero el nodo sobre el que queremos hacer la lectura. Aunque se podría facilitar el contenedor completo preferimos sacar sólo la lectura en el servidor, pues es el punto que consideramos más importante de la red y evitamos sobrecargar de archivos que aportan a priori poca información práctica. Este archivo se puede leer con aplicaciones como *TCPDump*, *Wireshark* y otras.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

```
/*
 * Habilitamos tracing
 */
inetnetStackHelper.EnablePcapIPv6("tfg", nodoGateway.Get(0) -> GetId(), 1, true);
double tasa = lrWpanDevices.Get(0) -> GetObject<LrWpanNetDevice>() -> GetPhy() -> GetDataOrSymbolRate(true);
std::cout << "La tasa de transmisión es: " << tasa << " bps" << std::endl;

/*
 * -----Simulación-----
 */
Config::Connect ("/Nodelist/1/DeviceList/0/$ns3::LrWpanNetDevice/Mac/MacTxDrop", MakeCallback (&PaquetesPerdidos));
if(verbose_mob){
Config::Connect ("/Nodelist/1/$ns3::MobilityModel/CourseChange", MakeCallback (&CourseChange));
}
if(verbose_phy){
Config::Connect ("/Nodelist/0/DeviceList/0/$ns3::LrWpanNetDevice/Channel/$ns3::SingleModelSpectrumChannel/PathLoss", MakeCallback (&PathLoss));
Config::Connect ("/Nodelist/0/DeviceList/0/$ns3::LrWpanNetDevice/Phy/MacTxDrop", MakeCallback (&EstadoTarjeta));
}
/*
 * -----Creamos animación-----
 */
AnimationInterface animacion ("animación_tfg.xml");
```

Ilustración 63: Tracing, Callback y generación de animación.

Creamos cuatro conexiones que realizan un *callback* sobre las funciones que hemos definido al principio del programa y que actúan sobre los tracesources (existe una lista completa de tracesources definidos en la documentación del framework) definidos a tal efecto en las clases que corresponden. Como vemos en las rutas, el primero de ellos nos ofrece el número de paquetes perdidos en la transmisión, el segundo lo utilizamos para conocer las coordenadas de la ubicación del nodo 1 -aunque si en lugar de este valor ponemos un asterisco se mostrará la posición de todos los nodos presentes en la simulación- el nodo 0 permanece siempre en una posición fija (concretamente en las coordenadas (0,0)), con lo que no tiene demasiado sentido seguir su recorrido. Otro de los *callbacks* nos devuelve el valor de la pérdida en dB en el medio de transmisión. El último de ellos nos indica el estado de la tarjeta de transmisión en cada uno de los estados de la siguiente tabla.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

Enumerator
IEEE_802_15_4_PHY_BUSY
IEEE_802_15_4_PHY_BUSY_RX
IEEE_802_15_4_PHY_BUSY_TX
IEEE_802_15_4_PHY_FORCE_TRX_OFF
IEEE_802_15_4_PHY_IDLE
IEEE_802_15_4_PHY_INVALID_PARAMETER
IEEE_802_15_4_PHY_RX_ON
IEEE_802_15_4_PHY_SUCCESS
IEEE_802_15_4_PHY_TRX_OFF
IEEE_802_15_4_PHY_TX_ON
IEEE_802_15_4_PHY_UNSUPPORTED_ATTRIBUTE
IEEE_802_15_4_PHY_READ_ONLY
IEEE_802_15_4_PHY_UNSPECIFIED

Ilustración 64: Estados de tarjeta Irwpan [69]

Finalmente se genera un fichero mediante la clase *AnimationInterface* y a cuyo constructor le pasamos una cadena de texto con el nombre del archivo. Este archivo generado tiene extensión xml y se genera durante la simulación. Se puede abrir con el software NetAnim, que se descarga junto con el *framework* y nos permite visualizar nuestra simulación de manera gráfica.

- **Ejecutar simulación**

El último paso es indicar al compilador que inicie la simulación. Para ello se utilizan las siguientes líneas de código:

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

```
    Simulator::Stop(Minutes(1.0));
    Simulator::Run ();
    Simulator::Destroy ();

    NS_LOG_INFO ("Fin de ejecución");

    return 0;
}
```

Ilustración 65: Inicio y fin de simulación.

Invocando el método Stop() indicamos la duración de la simulación. Mediante Run() la iniciamos y mediante Destroy() la destruimos una vez haya finalizado el tiempo estipulado. La última línea es un log que saca por pantalla la frase "Fin de ejecución" que nos indica que se ha llegado al final de la misma.

5.3.PRUEBA DE SIMULACIÓN

Para finalizar con este trabajo se realizará una simulación sobre le escenario propuesto y se observará el comportamiento de la red. En este caso, para dicha prueba los parámetros elegidos son el número de nodos total -que será de 15-, supondremos un total de 15 internos y sus respectivos dispositivos y las dimensiones de la planta del edificio serán las que hemos indicado en su momento -15x40 metros- y el modelo de propagación será en interiores.

Para ejecutar la simulación en este contexto lanzamos el siguiente comando dentro de la carpeta de ns3:

```
./waf --run "tfg --interiores=true -nNodos=15" > tfg_prueba1.txt 2>&1
```

Al finalizar la simulación se obtiene un archivo con extensión *pcap* denominado tfg.pcap, un archivo denominado animación_tfg.xml y otro archivo al que volcamos la salida de la ejecución del comando en texto plano denominado tfg_prueba1.txt.

Al observar los datos obtenidos vemos varias cosas interesantes. La primera de ellas es que si abrimos el fichero tfg.pcap sólo vemos un paquete en la captura.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

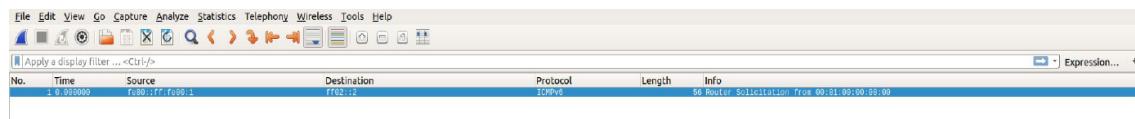


Ilustración 66: Captura tfg.pcap vista en wireshark.

Comprobemos entonces el contenido del fichero tfg_prueba1.txt. Para ello ejecutamos el siguiente comando filtrando los resultados que nos interesan. En este caso la potencia de recepción.

```
$cat tfg_prueba1.txt | grep "receiving packet with power" |more
```

Obteniendo la siguiente salida:

```
miguel@miguel-N24-25JU: ~/ns-3-allinone/ns-3-dev 80x24
0x55bda531a830 receiving packet with power: -127.798dBm
0x55bda53192c0 receiving packet with power: -162.813dBm
0x55bda531be40 receiving packet with power: -149.871dBm
0x55bda531a830 receiving packet with power: -106.62dBm
0x55bda531be40 receiving packet with power: -147.149dBm
0x55bda531a830 receiving packet with power: -127.801dBm
0x55bda53192c0 receiving packet with power: -162.82dBm
0x55bda531be40 receiving packet with power: -149.873dBm
0x55bda531a830 receiving packet with power: -106.638dBm
0x55bda531be40 receiving packet with power: -147.141dBm
0x55bda53192c0 receiving packet with power: -107.105dBm
0x55bda5318000 receiving packet with power: -126.796dBm
0x55bda531be40 receiving packet wlth power: -124.717dBm
0x55bda53192c0 receiving packet with power: -107.12dBm
0x55bda5318000 receiving packet wlth power: -126.798dBm
0x55bda531be40 receiving packet with power: -124.717dBm
0x55bda531a830 receiving packet wlth power: -106.692dBm
0x55bda531be40 receiving packet with power: -145.5dBm
0x55bda5318000 receiving packet wlth power: -147.392dBm
0x55bda531be40 receiving packet with power: -150.49dBm
0x55bda53192c0 receiving packet wlth power: -145.689dBm
0x55bda531a830 receiving packet with power: -142.245dBm
0x55bda5318000 receiving packet wlth power: -145.075dBm
--Más--
```

Ilustración 67: Salida del comando cat.

Vemos los valores de potencia obtenidos para los paquetes enviados y estos valores son bastante bajos. Si acudimos a la documentación de NS-3 [72] y visitamos los apartados relativos a LrWPAN vemos que los equipos están configurados para una sensibilidad de -106.58 dB, por lo que los valores en recepción no alcanzan el valor máximo de sensibilidad.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

Acabamos de ver la importancia de la simulación de la red en nuestro propio caso de uso. Se ha podido comprobar que de haber implementado la solución directamente sin haber realizado una simulación previa nos habría llevado al fracaso.

El hecho de que la señal viaje en interiores hace que tenga que superar obstáculos, como las paredes. Esto hace que la potencia de la señal caiga rápidamente. Una posible solución sería aumentar la potencia de transmisión, pero esto presenta varios inconvenientes. El primero de ellos es que las baterías durarán menos tiempo y si ponemos baterías de mayor capacidad aumentaremos el peso, algo poco deseable en este tipo de aplicación. Por otro lado, la potencia está configurada a 0 dBm que se corresponden con 1 mW. Se ha comprobado la respuesta cambiando la potencia de transmisión a 25mw, es decir, 13.9 dBm, que es el máximo admitido por la nota UN-39 para la banda de uso (869MHz) y aún así no ha sido suficiente.

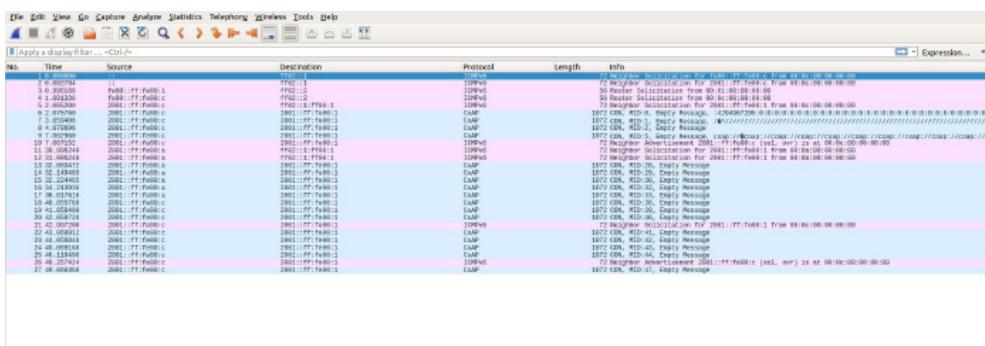


Ilustración 68: Segunda captura vista en wireshark mostrando leve mejoría en tránsito de paquetes.

Aunque la situación mejora levemente pues algún nodo consigue establecer comunicación está muy lejos de ser la condición deseada.

Otra solución y considerada más ventajosa sería estudiar la posibilidad de añadir en los dispositivos una mejora en la sensibilidad en recepción junto con la implementación en los nodos de un protocolo de enrutado que permita utilizar los elementos de la red como dispositivos ad-hoc y de ese modo alcanzar el nodo que funciona como puerta de enlace, es decir, pasar de una topología en estrella a una topología en malla. Protocolos de este tipo son: *Routing Protocol for Low-Power and Lossy Networks (RPL)*, *Ad Hoc On-*

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

Demand Distance Vector (AODV), *Destination-Sequenced Distance-Vector* (DSDV), *Optimized Link State Routing* (OLSR) y *Dynamic Source Routing* (DSR). En concreto AODV se encuentra actualmente en desarrollo para su implementación en el *framework* NS-3.

Lamentablemente a día de hoy no se dispone en NS-3 de ningún protocolo de este estilo para estos dispositivos que funcione con direccionamiento IPv6, cabría la posibilidad de simular la red con capa de red IPv4, pero no existe convergencia entre la capa de red y las capas LrWPAN, es decir, se debe utilizar siempre la capa de convergencia 6LoWPAN que sólo sirve con IPv6.

Para finalizar, comprobamos las prestaciones de la red en un entorno distinto al estudiado antes. Dispondremos una simulación carente de obstáculos con una de las otras opciones para comprobar que efectivamente en las circunstancias adecuadas la red funciona.

Para ello ejecutamos el comando siguiente:

```
./waf --run "tfg --box=true -nNodos=15" > tfg_prueba2.txt 2>&1
```

Simulamos la red con el patrón box y se elimina la opción de interiores con lo que se elimina el efecto de los muros interiores en la simulación. En este caso se usa un modelo de pérdidas *LogDistancePropagationLossModel*.

Si ahora realizamos la comprobación anterior al abrir el fichero de texto generado vemos que las medidas de la potencia obtenidas favorecen la posibilidad de establecer conexiones.

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

```
miguel@miguel-N24-25JU: ~/ns-3-allinone/ns-3-dev
miguel@miguel-N24-25JU: ~/ns-3-allinone/ns-3-dev 80x24
0x55a25e725af0 receiving packet with power: -73.3667dBm
0x55a25e722c10 receiving packet with power: -76.0849dBm
0x55a25e66fad0 receiving packet with power: -79.0554dBm
0x55a25e673f40 receiving packet with power: -79.264dBm
0x55a25e7214a0 receiving packet with power: -79.8981dBm
0x55a25e6750b0 receiving packet with power: -80.0389dBm
0x55a25e724380 receiving packet with power: -80.3863dBm
0x55a25e672800 receiving packet with power: -81.1039dBm
0x55a25e66d140 receiving packet with power: -81.4105dBm
0x55a25e6710e0 receiving packet with power: -82.3637dBm
0x55a25e71b780 receiving packet with power: -83.3814dBm
0x55a25e66be80 receiving packet with power: -84.5807dBm
0x55a25e66d140 receiving packet with power: -85.7293dBm
0x55a25e672800 receiving packet with power: -87.2031dBm
0x55a25e724380 receiving packet with power: -87.1757dBm
0x55a25e722c10 receiving packet with power: -87.425dBm
0x55a25e7214a0 receiving packet with power: -87.6117dBm
0x55a25e725af0 receiving packet with power: -87.6174dBm
0x55a25e71ce50 receiving packet with power: -87.8661dBm
0x55a25e71e5c0 receiving packet with power: -88.6005dBm
0x55a25e71fd30 receiving packet with power: -88.3871dBm
0x55a25e673f40 receiving packet with power: -89.2723dBm
0x55a25e66e500 receiving packet with power: -90.1348dBm
--Más--
```

Ilustración 69: Medidas de potencia en la nueva simulación.

Si abrimos el fichero pcap con wireshark y visualizamos las gráficas que nos ofrece la herramienta estadística observamos el tránsito fluido de paquetes dentro de la red.

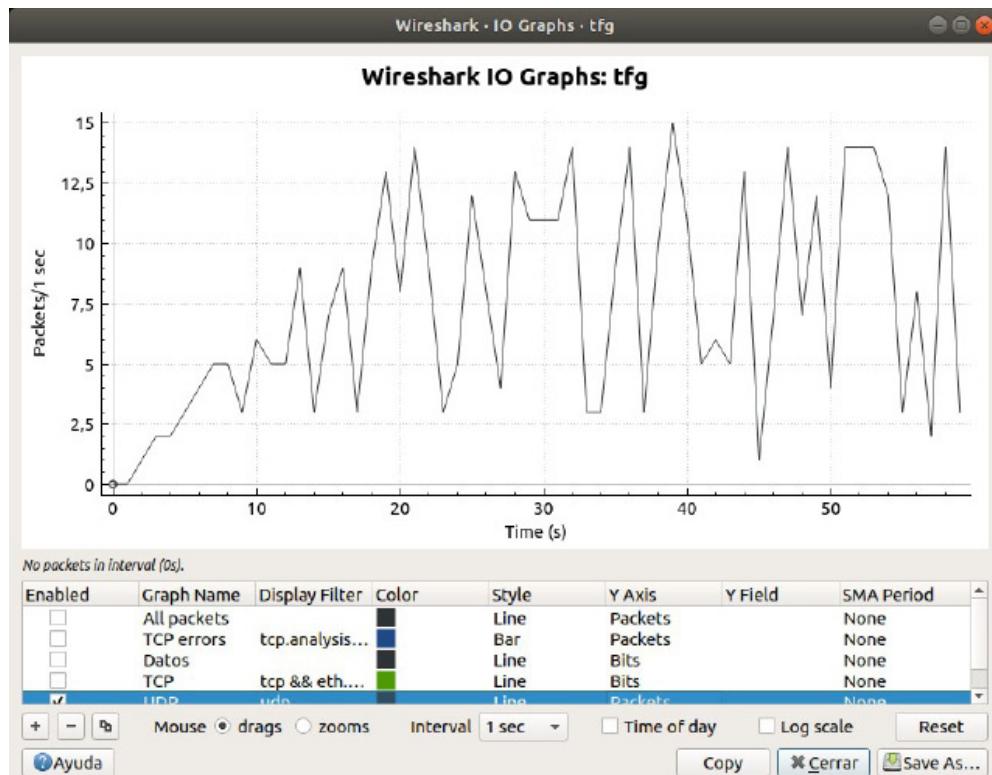


Ilustración 70: Paquetes UDP/seg

SIMULACIÓN DEL DISEÑO EN NS - 3 Y OBTENCIÓN DE RESULTADOS

Para finalizar es siempre interesante poder ver gráficamente la red en funcionamiento. Para ello ejecutamos el siguiente comando (situados dentro de la carpeta ns-3-allinone) para abrir la herramienta NetAnim

```
$cd netanim/ && ./NetAnim
```

Una vez hemos abierto NetAnim cargamos el archivo animación_tfg.xml que generamos y vemos correr la simulación con los nodos en movimiento.

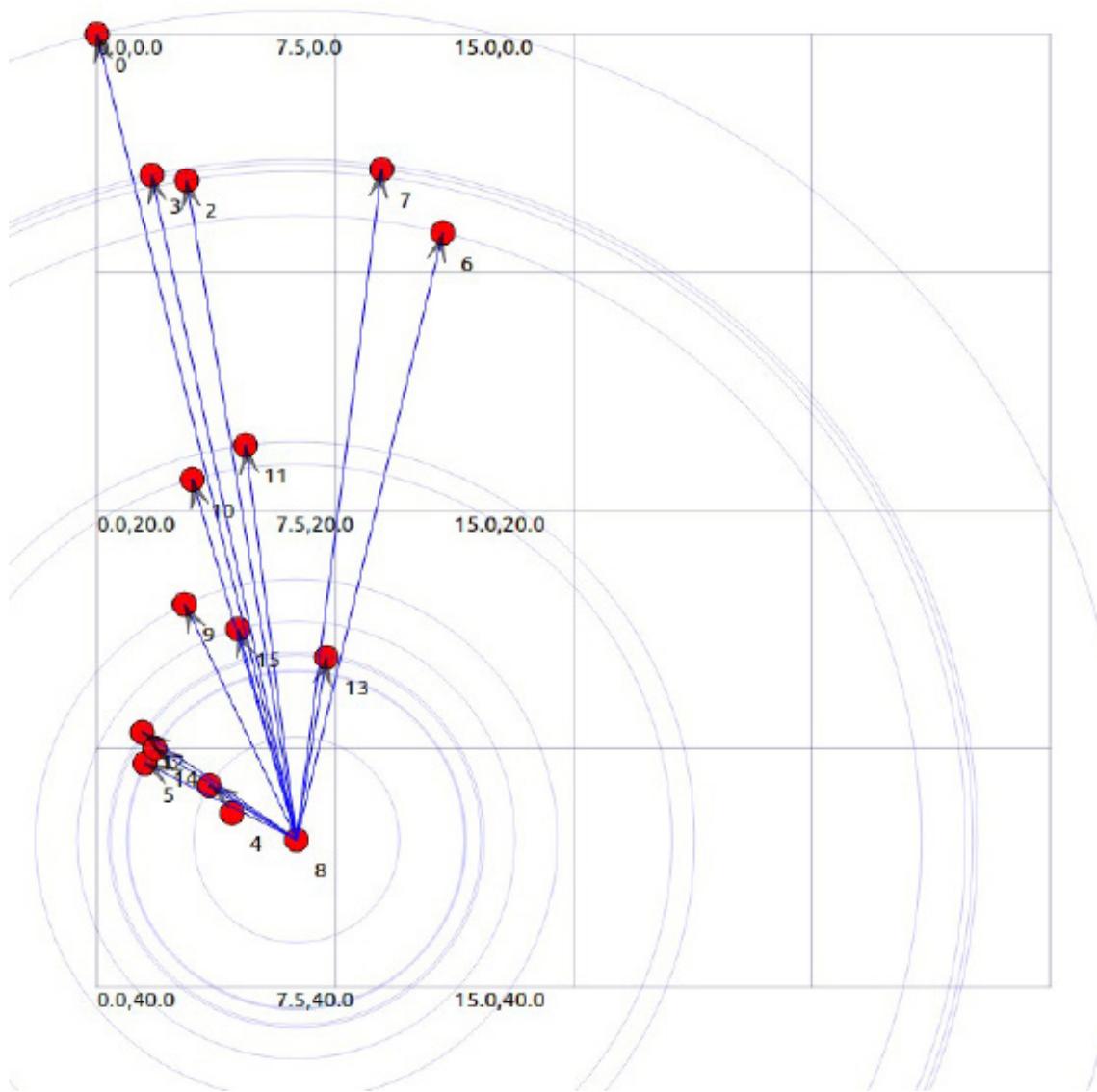


Ilustración 71: Visualización de la simulación en entorno NetAnim.

6

CONCLUSIONES

A lo largo del presente trabajo se ha hecho un recorrido por todas las tecnologías que se emplean para el despliegue de redes IoT. Sin duda, el potencial de las soluciones basadas en las tecnologías de este tipo son enormes. Se ha podido comprobar que a día de hoy se están poniendo en marcha multitud de proyectos ambiciosos que harán posible el desarrollo tecnológico alrededor de sectores en los que, de manera tradicional, las telecomunicaciones no tenían una gran presencia. Por su parte, diferentes organismos están desarrollando nuevos estándares en base a novedosas técnicas enfocadas a conseguir cada vez mayores alcances y fiabilidad en la transmisión de datos sobre el escaso espectro disponible.

Todas estas técnicas permiten que se puedan conectar infinidad de dispositivos entre si y a Internet, generando un enorme volumen de datos de todo tipo que, tratados adecuadamente - como por ejemplo, los métodos de análisis basados en *Big Data* - , abren un nuevo mundo de posibilidades de desarrollo.

En todo este conglomerado la simulación de redes ocupará un lugar muy importante en la evolución tecnológica que viene, permitiendo a los investigadores desarrollar sus productos con una visión previa muy sólida de lo que cabe esperar de los nuevos diseños, funcionando en entornos próximos a la realidad.

Las herramientas de simulación de software libre, como es el caso de NS-3, crecen

CONCLUSIONES

al amparo de los desarrollos realizados por la comunidad y tienen detrás una ingente cantidad de desarrolladores aportando nuevas ideas y mejoras a la plataforma.

Cierto es, que el manejo de NS-3 resulta al principio un poco abrumador debido sobretodo al gran nivel de detalle que permite alcanzar y a la enorme cantidad de módulos de que dispone. El hecho de tener que programar directamente las simulaciones por no disponer de un *GUI*, tampoco pone las cosas demasiado fáciles. Sin embargo, poco tiempo después se puede apreciar el enorme potencial que ofrece para el campo de la investigación. A pesar de ello, como ha ocurrido en este caso, una limitación importante ha sido el no disponer de algún modelo de enrutado *ad - hoc* IPv6 para la simulación, lo que ha impedido desarrollar en su totalidad la idea planteada, limitación que se podría salvar programando módulos propios. No obstante, esta es una circunstancia que sobrepasaba ampliamente los objetivos perseguidos en este trabajo.

Resultaría de gran interés para trabajos futuros - más centrados en la parte de simulación - enfocarse en la implementación de algún módulo que permitiese realizar el enrutado con direccionamiento IPv6 y adaptar los módulos LrWPAN para poder funcionar con el protocolo de red IPv4 y poner en marcha la capa de aplicación CoAP.

BIBLIOGRAFÍA

- [1] IEEE Internet of Things. (2015). Towards a definition of the Internet of Things (IoT) (Revision 1 – Published 27 MAY 2015). Recuperado de https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf
- [2] Microsoft. (2018, 28 febrero). FarmBeats - Microsoft Garage. Recuperado 14 febrero, 2019, de <https://www.microsoft.com/en-us/garage/wall-of-fame/farmbeats/>
- [3] Iot for all. (2018, 16 octubre). How Connected Devices are Changing the Manufacturing Industry. Recuperado 28 febrero, 2019, de <https://www.iotforall.com/iiot-devices-change-manufacturing-industry/>
- [4] Writer, G. (2018, 24 abril). Top 5 IoT Solutions for Healthcare Providers. Recuperado 28 febrero, 2019, de <https://www.iotforall.com/top-digital-health-solutions/>
- [5] DAQRI - Professional Grade AR. (s.f.). Recuperado 5 marzo, 2019, de <https://daqri.com/>
- [6] RONDINELLI, D. (2015). Dictamen del Comité Económico y Social Europeo sobre «Las ciudades inteligentes como motor de una nueva política industrial europea». Recuperado de <https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A52015IE0586>
- [7] European Commission. (s.f.). Smart cities. Recuperado 5 marzo, 2019, de

BIBLIOGRAFÍA

https://ec.europa.eu/info/eu-regional-and-urban-development/topics/cities-and-urban-development/city-initiatives/smart-cities_en

[8] Wellness Telecom. (s.f.). Soluciones para smart cities: hardware, software y a medida. Recuperado 5 marzo, 2019, de <http://www.wtelecom.es/productos-y-servicios/smart-cities-solutions/>

[9] Ikusi Velatia. (s.f.). Spider. Recuperado 5 marzo, 2019, de <https://www.urbansolutions.es/es/es>

[10]EDX.(2019).Getting Started with the Internet of Things(IoT)[Material de curso]. Recuperado de <https://courses.edx.org/courses/course-v1:Microsoft+DEV296x+1T2019/course/>

[11] IBM Watson IoT. (s.f.). IoT for Buildings, Smart Buildings. Recuperado 6 marzo, 2019, de <https://www.ibm.com/internet-of-things/explore-iot/buildings>

[12]Reilly, D. (2018, 24 septiembre). IoT Energy Applications: From Smart Vehicles to Smart Meters. Recuperado 6 marzo, 2019, de <https://www.iotforall.com/iot-energy-applications/>

[13] Microsoft. (2014, marzo). Fueling the oil and gas industry with IoT. Recuperado 8 marzo, 2019, de <https://customers.microsoft.com/en-hk/story/724546-fueling-the-oil-and-gas-industry-with-iot-1>

[14] car2go Iberia S.L.. (s.f.). car2go carsharing España. Recuperado 8 marzo, 2019, de <https://www.car2go.com/ES/es/>

[15] Railway Innovation.com. (s.f.). How is Deutsche Bahn using smart sensor technology to avoid infrastructure failure? Recuperado 8 marzo, 2019, de <https://www.konux.com/wp-content/uploads/KONUX-case-study-deutsche-bahn-using-smart-sensor-technology-to-avoid-infrastructure-failure.pdf>

[16]Kao, C.-C.; Lin, Y.-S.; Wu, G.-D.; Huang,, C. J. A. (2017). A Comprehensive Study on the Internet of Underwater Things: Applications, Challenges, and Channel

BIBLIOGRAFÍA

Models. Recuperado 13 marzo, 2019, de <https://www.konux.com/wp-content/uploads/KONUX-case-study-deutsche-bahn-using-smart-sensor-technology-to-avoid-infrastructure-failure.pdf>

[17] Kyung Sup, K. (2010, diciembre). An overview of IEEE 802.15.6 standard [Ilustración]. Recuperado 13marzo,2019,dehttps://www.researchgate.net/publication/224215458_An_overview_of_IEEE_802156_standard

[18] IEEE. (2012, 29 febrero). IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks. IEEE Std, 15(6), 1–271. Recuperado de <https://ieeexplore.ieee.org/document/6161600/references#references>

[19] IEEE - SA. (2015). IEEE 802.15.4-2015 - IEEE Standard for Low-Rate Wireless Networks. Recuperado 29 febrero, 2019, de https://standards.ieee.org/standard/802_15_4-2015.html

[20] Martin, W. (2019, 28 enero). Bluetooth Core Specification v5.1 Feature Overview | Bluetooth Technology Website. Recuperado 28 febrero, 2019, de <https://www.bluetooth.com/bluetooth-resources/bluetooth-core-specification-v5-1-feature-overview/>

[21] Martin, W. (2017, 13 febrero). Exploring Bluetooth 5 – Going the Distance [Ilustración]. Recuperado 28 febrero, 2019, de <https://www.bluetooth.com/blog/exploring-bluetooth-5-going-the-distance/>

[22] Martin, W. (2017, 13 febrero). Exploring Bluetooth 5 – Going the Distance | Bluetooth Technology Website [Ilustración]. Recuperado 28 febrero, 2019, de <https://www.bluetooth.com/blog/exploring-bluetooth-5-going-the-distance/>

[23] Ince, A & Elma, Onur & S. SELAMOGULLARI, Ugur & Vural,, B. (2014). Data Reliability and Latency Test for ZigBee-based Smart Home Energy Management Systems. Documento presentado en 7th International Ege Energy Symposium & Exhibition, At Husak, Turkey. Recuperado de https://www.researchgate.net/publication/263220840_Data_Reliability_and_Latency_Test_for_ZigBee-based_Smart_

BIBLIOGRAFÍA

Home_Energy_Management_Systems

- [24] Thubert, P. (2011, septiembre). Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. Recuperado 2 marzo, 2019, de <https://tools.ietf.org/pdf/rfc6282.pdf>
- [25] Olsson, J. (2014). 6LoWPAN demystified. Recuperado 5 marzo, 2019, de <http://www.ti.com/lit/wp/swry013/swry013.pdf>
- [26] Thubert, P., Winter, T., Brandt, A., W Hui, J. (2012). RPL: IPv6 Routing Protocol for Low power and Lossy Networks Article · March 2012 with 763 Reads. Recuperado 5 marzo, 2019, de <https://tools.ietf.org/html/rfc6550>
- [27] Gettys,J.,Mogul,J.C.,Frystyk,H.,Masinter,L.,Leach,P.,Berners-Lee,T. (1999). 403 Forbidden. Recuperado 5 marzo, 2019, de <https://tools.ietf.org/html/rfc2616+403>
- [28] Shelby,Z.,Hartke,K.,Bormann, C. (2014). The Constrained Application Protocol (CoAP). Recuperado 13 marzo, 2019, de <https://tools.ietf.org/html/rfc7252>
- [29] ISO/, I. E. C. (2016). Information technology — Message Queuing Telemetry Transport (MQTT) v3.1.1. Recuperado de <https://www.iso.org/standard/69466.html++>
- [30] Manz, B. (s.f.). IoT Protocols: A Growing Stack [Ilustración]. Recuperado 23 marzo, 2019, de <https://eu.mouser.com/applications/IoT-protocols-growing-stack/>
- [31] ADDIS NETWORK S.L. (s.f.-b). A FONDO: Z-WAVE, SIN CABLES. Recuperado 23 marzo, 2019, de <https://www.domodesk.com/162-a-fondo-z-wave-sin-cables.html+https://www.domodesk.com/>
- [30] Pei, Zhongmin & Deng, Zhidong & Yang, Bo & Cheng, X. I. A. O. L. I. A. N. G. (2008). Application-oriented wireless sensor network communication protocols and hardware platforms: A survey - Scientific Figure on ResearchGate [Ilustración]. Recuperado 23 marzo, 2019, de https://www.researchgate.net/figure/Z-Wave-protocol-stack_fig2_224327774

- [32] IoT Point. (s.f.). Z-Wave Tutorial [Publicación en un blog]. Recuperado 23 marzo, 2019, de <https://iotpoint.wordpress.com/z-wave-tutorial/>
- [33] ECMA-340. (2013). Near Field Communication - Interface and Protocol (NFCIP-1). Recuperado 28 marzo, 2019, de <https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-340.pdf>
- [34] ISO/IEC. (2004). Information technology -- Telecommunications and information exchange between systems -- Near Field Communication -- Interface and Protocol (NFCIP-1). Recuperado 28 marzo, 2019, de <https://www.iso.org/standard/38578.html>
- [35] Hubers, E. (2015). NFC Protocol Stack [Ilustración]. Recuperado 7 abril, 2019, de https://commons.wikimedia.org/wiki/File:NFC_Protocol_Stack.png
- [36] RuBee Working Group. (2009). IEEE Standard for Long Wavelength Wireless Network Protocol. Recuperado 7 abril, 2019, de https://standards.ieee.org/standard/1902_1-2009.html
- [37] IEEE STANDARDS ASSOCIATION. (2016). IEEE Standard for High Data Rate Wireless Multi-Media Networks. Recuperado 8 abril, 2019, de https://standards.ieee.org/standard/802_15_3-2016.html
- [38] IEEE STANDARDS ASSOCIATION. (2009). IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. Recuperado 8 abril, 2019, de https://standards.ieee.org/standard/802_11n-2009.html
- [39] IEEE STANDARDS ASSOCIATION. (2013). IEEE Standard for Information technology--Telecommunications and information exchange between systems—Local and metropolitan area networks--Specific requirements--Part 11: Wireless LAN

BIBLIOGRAFÍA

Medium Access Control (MAC) and Physical Layer (PHY) Specifications--Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz.. Recuperado 28 abril, 2019, de https://standards.ieee.org/standard/802_11n-2009.html+

[40] Rohde & Schwarz. (2017). WLAN at 60 GHz A Technology Introduction. Recuperado 2 mayo, 2019, de https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_application/application_notes/1ma220/1MA220_3e_WLAN_11ad_WP.pdf+

[41] Rohde & Schwarz. (2016). IEEE 802.11ax Technology Introduction. Recuperado 2 mayo, 2019, de https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/1MA222_1e_IEEE80211ax.pdf+

[42] IEEE STANDARDS ASSOCIATION. (2010). IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. Recuperado 2 mayo, 2019, de https://standards.ieee.org/standard/802_11p-2010.html

[43] Colaboradores de Wikipedia. (s.f.). Wireless Access in Vehicular Environments+ - Wikipedia, la enciclopedia libre. Recuperado 3 mayo, 2019, de https://es.wikipedia.org/wiki/Wireless_Access_in_Vehicular_Environments+

[44] HIDOBRO MOYA, J.M., LUQUE ORDÓÑEZ,, J. (2013). COMUNICACIONES POR RADIO. TECNOLOGÍAS, REDES Y SERVICIOS DE RADIOPCOMUNICACIONES. EL ESPECTRO ELECTROMAGNÉTICO. Madrid, España: RA-MA EDITORIAL.

[45] IEEE STANDARDS ASSOCIATION. (2013). IEEE Standard for Wireless Access in Vehicular Environments ? Security Services for Applications and Management Messages. Recuperado 13 mayo, 2019, de https://standards.ieee.org/standard/1609_2-2013.html

- [46] IEEE STANDARDS ASSOCIATION. (2010). IEEE Standard for Wireless Access in Vehicular Environments (WAVE)--Networking Services Corrigendum 1: Miscellaneous Corrections. Recuperado 13 mayo, 2019, de https://standards.ieee.org/standard/1609_3-2010-Cor1-2012.html
- [47] IEEE STANDARDS ASSOCIATION. (2010). IEEE Standard for Wireless Access in Vehicular Environments (WAVE)--Networking Services Corrigendum 1: Miscellaneous Corrections. Recuperado 13 mayo, 2019, de https://standards.ieee.org/standard/1609_4-2010.html
- [48] IEEE STANDARDS ASSOCIATION. (2010). IEEE Standard for Wireless Access in Vehicular Environments (WAVE)-- Over-the-Air Electronic Payment Data Exchange Protocol for Intelligent Transportation Systems (ITS). Recuperado 14 mayo, 2019, de https://standards.ieee.org/standard/1609_11-2010.html
- [49] European Telecommunications Standards Institute. (2000). Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; System Overview. Recuperado de https://www.etsi.org/deliver/etsi_tr/101600_101699/101683/01.01.01_60/tr_101683v010101p.pdf
- [50] IEEE Standard Association. (2018). IEEE Standard for Air Interface for Broadband Wireless Access Systems. Recuperado de <https://ieeexplore.ieee.org/document/8303870/references#references>
- [51] Mojtaba,S., Mohamed, O. (2013). IEEE 802.16: WiMAX overview, WiMAX architecture. Recuperado 4 mayo, 2019, de https://www.researchgate.net/publication/271302282_IEEE_80216_WiMAX_overview_WiMAX_architecture
- [52] IEEE. (2012, 7 septiembre). IEEE Standard for WirelessMAN-Advanced Air Interface for Broadband Wireless Access Systems. IEEE, -(-), 1–1090. Recuperado de <https://ieeexplore.ieee.org/document/6297413>

BIBLIOGRAFÍA

- [53] IEEE. (2006, 28 febrero). IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems - Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands. IEEE, -(-), 1–822. Recuperado de <https://ieeexplore.ieee.org/document/1603394>
- [54] IEEE. (2015, 30 octubre). IEEE Standard for Information Technology--Telecommunications and information exchange between systems - Wireless Regional Area Networks (WRAN)--Specific requirements - Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Policies and Procedures for Operation in the TV Bands - Amendment 2: Enhancement for Broadband Services and Monitoring Applications. IEEE, -(-), 1–299. Recuperado de <https://ieeexplore.ieee.org/document/7336461>
- [55] Electronics notes. (s.f.). SigFox for M2M & IoT. Recuperado 15 mayo, 2019, de <https://www.electronics-notes.com/articles/connectivity/sigfox/what-is-sigfox-basics-m2m-iot.php+>
- [56] 3GPP. (s.f.). Releases [Ilustración]. Recuperado 15 mayo, 2019, de <https://www.3gpp.org/specifications/releases+>
- [57] Dr. Harrison J. Son and Dr. Michelle, M. (2015, 3 noviembre). KT's 5G network architecture [Ilustración]. Recuperado 15 mayo, 2019, de <https://www.netmanias.com/en/post/blog/8256/5g-c-ran-fronthaul-kt-korea/5g-network-as-envisioned-by-kt-analysis-of-kt-s-5g-network-architecture>
- [58] Finnegan, J., Brown, S. (2018, 12 febrero). A Comparative Survey of LPWA Networking. Recuperado 15 mayo, 2019, de <https://arxiv.org/pdf/1802.04222.pdf>
- [59] Park, Seung-Kyu & Hwang, Kwang-il & Kim, Hyo-Seong & Shim (2017, mayo). Challenges and Experiment with LoRaWAN [Ilustración]. Recuperado 16 mayo, 2019, de https://www.researchgate.net/figure/LoRaWAN-Protocol-Stack_fig2_317173603

- [60] Postscapes. (2019, 23 marzo). LPWAN IoT Market [Ilustración]. Recuperado 17 mayo, 2019, de <https://www.postscapes.com/long-range-wireless-iot-protocol-lora/>
- [61] Network of Excellence of our Robotic HomeLab Kit Community.. (2017, 28 noviembre). en:iot-open:communications_and_communicating_sut:iot_protocols.png [Robotic & Microcontroller Educational Knowledgepage - Network of Excellence] [Ilustración]. Recuperado 18 mayo, 2019, de https://home.roboticlab.eu/_detail/en/iot-open/communications_and_communicating_sut/iot_protocols.png?id=en%3Aiot-open%3Acommunications_and_communicating_sut%3Amost_widely_used_wireless_communication_protocols
- [62] Nsnam. (s.f.). ns-3 is a discrete-event network simulator. Recuperado 18 mayo, 2019, de <https://www.nsnam.org/>
- [63] NS-3. (s.f.). The list of all trace sources.. Recuperado 18 mayo, 2019, de https://www.nsnam.org/docs/release/3.19/doxygen/group__trace_source_list.html
- [64] Verolino, F. F. (s.f.). 6LoWPAN[Ilustración]. Recuperado 19 mayo, 2019, de <https://www.kickstarter.com/projects/smartmachine/6lowpan-devices-designed-for-network-and-iot-appli>
- [65] RScomponents. (s.f.). WEP-6LoWPAN-IoT-GW [Ilustración]. Recuperado 19 mayo, 2019, de <https://uk.rs-online.com/web/p/networking-modules/8952463/>
- [66] Ishaq, I. (2013). 6LoWPAN Fragmentation Header [Ilustración]. Recuperado 19 mayo, 2019, de https://www.researchgate.net/figure/6LoWPAN-Fragmentation-Header-a-6LowPAN-encapsulation-header-stack-for-the-first_fig6_236334231
- [67] Shichao, A. (s.f.). Encabezado de IPv6 [Ilustración]. Recuperado 19 mayo, 2019, de <https://notes.shichao.io/tcpv1/ch5/>
- [68] Shelby,Z., Hartke,K.,Bormann, C. (2014, junio). The Constrained Application Protocol (CoAP). Recuperado 19 mayo, 2019, de <https://tools.ietf.org/html/rfc7252>

BIBLIOGRAFÍA

- [69] NS3. (2006). PHY Emumerations Table 18 in section 6.2.3. [Ilustración]. Recuperado 20 mayo, 2019, de https://www.nsnam.org/doxygen/group__lr-wpan.html
- [70] NS-3. (2019, 8 enero). ns-3 Tutorial. Recuperado 24 mayo, 2019, de <https://www.nsnam.org/docs/release/3.29/tutorial/ns-3-tutorial.pdf>
- [71] NS-3. (s.f.). ns-3 Manual. Recuperado 24 mayo, 2019, de <https://www.nsnam.org/docs/manual/ns-3-manual.pdf>
- [72] NS-3. (s.f.). Model Library. Recuperado 25 mayo, 2019, de <https://www.nsnam.org/docs/models/ns-3-model-library.pdf>
- EDX. (2019). Course | IOT3x | edX [Curso]. Recuperado de <https://courses.edx.org/courses/course-v1:CurtinX+IOT3x+3T2018/course/>
- EDX.org. (2019). Introduction to the Internet of Things (IoT). Recuperado de <https://courses.edx.org/courses/course-v1:CurtinX+IOT1x+1T2019/course/>
- IPv4 and IPv6 Headers [Ilustración]. (s.f.). Recuperado 15 mayo, 2019, de <https://notes.shichao.io/tcpv1/ch5/>
- W. Hui,J., E. Culler, D. (s.f.). 6LoWPAN. Recuperado 20 mayo, 2019, de <https://www.iiitd.edu.in/%7Eamarjeet/EmSys2013/6LoWPAN.pdf>

ANEXOS

ANEXOS

TABLA DE REFERENCIA DE TECNOLOGÍAS

TECNOLOGÍA	ESTÁNDAR PARA ACCESO AL MEDIO	ALCANCE TÍPICO	FRECUENCIA	POSIBLES APLICACIONES
WBAN				
N/A	IEEE 802.15.6	<i>Alrededor del cuerpo, muy corta distancia</i>	HBC: 5-50 Mhz NB: 863-870Mhz y 2,4-2,45 Ghz UWB: 3,1-10,6 Ghz	Wearables. Sensores para uso médico.
WPAN				
RFID	ISO 14443	<i>Desde unos pocos centímetros a varios metros en función de la banda de trabajo.</i>	125 y 134 KHz 13,56 Mhz 433 Mhz 860-960 Mhz 2,45-5,8 Ghz	Sistemas antirrobo. Identificación de personas. Control de accesos. Autenticidad de productos.
NFC	ISO 14443	<i>Unos pocos centímetros</i>	13,56 Mhz	Sistemas de pago con tarjeta o teléfono. Identificación de personas.

BLUETOOTH	<i>IEEE 802.15.1</i>	<i>Desde unos pocos metros hasta 350 metros para las últimas versiones del estándar.</i>	<i>2,4-2,48 Ghz</i>	<i>Wearables. Comunicaciones con periféricos (teclados, ratones). Control remoto. Publicidad beacon (BLE).</i>
ZIGBEE	<i>IEEE 802.15.4</i>	<i>Del orden de unos pocos metros llegando a 100 metros en condiciones favorables.</i>	<i>433,92 Mhz 868 Mhz 2,405-2,480 Ghz</i>	<i>Domótica, hogar conectado. Redes de sensores.</i>
6LoWPAN	<i>IEEE 802.15.4</i>	<i>Del orden de unos pocos metros llegando a 100 metros en condiciones favorables.</i>	<i>433,92 Mhz 868 Mhz 2,405-2,480 Ghz</i>	<i>Principalmente redes de sensores con conectividad directa a redes Ip.</i>
THREAD	<i>IEEE 802.15.4</i>	<i>Del orden de unos pocos metros llegando a 100 metros en condiciones favorables.</i>	<i>433,92 Mhz 868 Mhz 2,405-2,480 Ghz</i>	<i>Aplicaciones similares a las de 6lowpan con seguridad mejorada.</i>
Z-WAVE	<i>ITU-T G.9959</i>	<i>Alrededor de los 30 metros.</i>	<i>868.40, 868.42 y 869.85 Mhz</i>	<i>Domótica. Sistemas deseguridad. Entretenimiento.</i>

ANEXOS

RuBee	<i>IEEE 1902.1</i>	<i>Entre 0,5 y 30 metros.</i>	<450 KHz.	<i>Redes de sensores. Aplicaciones con requisitos bajos en cuanto a tasa de transmisión, como accionamiento remoto de interruptores, actuadores, etc.</i>
UWB	<i>IEEE 802.15.3</i>	<i>Alrededor de 10 metros.</i>	<i>Ultra Wide Band</i>	<i>Aplicaciones multimedia (el uso en infraestructuras IoT es hoy día residual pero podría tener aplicaciones interesantes debido a la elevada tasa de transmisión)</i>
WLAN				
WIFI	<i>IEEE 802.11x</i>	<i>Dependiendo del estándar y del tipo de conexión los alcances varían desde alguno metraza incluso varios kilómetros con antenas directivas (direccionales).</i>	2,4 Ghz 5 Ghz 60 Ghz	<i>Redes de área local. Conexión de dispositivos multimedia. Redes de dispositivos con necesidades de tasas de transmisión elevadas.</i>

WWAN				
SigFox	ETSI 300 220-1 ETSI 300 220-2	<i>Alcance típico entre 30 y 50 kilómetros en entornos rurales.</i> <i>Entre 3 y 10 kilómetros en entornos urbanos.</i>	868Mhz	<i>Grandes redes de sensores gestionadas en la nube.</i> <i>Control de actuadores a gran distancia.</i> <i>Comunicaciones M2M.</i>
Tecnologías móviles	Estándares 3GPP: NB-IoT EC-GSM-IoT MTC	<i>Variará en función de la tecnología móvil empleada y del entorno y será del orden de varios kilómetros.</i>	Bandas de telefonía móvil con licencia.	<i>Redes de sensores.</i> <i>Redes masivas formadas por dispositivos de bajas tasa de transmisión y necesidades de baja latencia.</i>
Ingenu RPMA	Estándar propietario RPMA (Random Phase Multiple Access)	Alrededor de 4 kilómetros pudiendo alcanzar los 16.	2,4 Ghz	<i>Principalmente redes M2M(red pública “the Machine Network” desplegada por Ingenu)</i>
LoRaWan	Estándar propietario Capa física SEMTECH AN1200.22. Capa de enlace LoRa	5 kilómetros en zonas urbanas 30 en zonas rurales.	433 Mhz 868 Mhz	Redes de sensores y actuadores.

ANEXOS

Weightless	<i>Especificaciones Weightlees de acceso libre.</i>	<i>Alrededor de los 2 kilómetros en entornos urbanos.</i>	<i>169,433, 470, 780,868, 915,923 MHz</i>	<i>Todo tipo de redes de sensores y pequeños dispositivos de gran alcance con capacidades de actualizaciones over-the-air y QoS.</i>
Telensa	<i>Tecnologías UNB</i>	<i>Aproximadamente 2 kilómetros en entorno urbano y 4 kilómetros en rural.</i>	<i>Bandas ISM sub 1 Ghz.</i>	<i>Grandes redes de dispositivos IoT (5000 nodos) operadores desde una estación base. Redes con aplicación en Smart City como análisis de tráfico, control de alumbrado etc.</i>
Wave-IoT	<i>NB-Fi</i>	<i>Alcance entre 16 y 50 kilómetros en función del entorno.</i>	<i>Bandas ISM sub 1 Ghz</i>	<i>Redes con aplicaciones en IoT para dispositivos de muy bajo consumo admitiendo hasta un millón de nodos con un solo Gateway cuya información se trata en la nube.</i>
TECNOLOGÍAS ACCESORIAS				
WMAN	WIMAX	LMDS	WIBRO	
WRAN		IEEE 802.22		