UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

ANÁLISIS COMPARATIVO ENTRE LORAWAN Y LTE EN UN ESCENARIO DE RED PARA LA INTERNET DE LAS COSAS USANDO EL SOFTWARE DE SIMULACIÓN NS3

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN INGENIERÍA DE REDES DE COMUNICACIONES

LENIN PATRICIO VARGAS REINOSO

PROFESOR GUÍA:
CLAUDIO ESTÉVEZ MONTERO

MIEMBROS DE LA COMISIÓN:
JORGE SANDOVAL ARENAS
SEBASTÍAN RÍOS PÉREZ

SANTIAGO DE CHILE
2018

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN INGENIERÍA DE REDES DE COMUNICACIONES
POR: LENIN PATRICIO VARGAS REINOSO
FECHA: 2018
PROF. GUÍA: CLAUDIO ESTÉVEZ MONTERO

ANÁLISIS COMPARATIVO ENTRE LORAWAN Y LTE EN UN ESCENARIO DE RED PARA LA INTERNET DE LAS COSAS USANDO EL SOFTWARE DE SIMULACIÓN NS3

LoRaWAN y LTE definen la coyuntura predominante como tecnologías de distribución para la Internet de las Cosas (IoT, Internet of Things). LoRaWAN define una red de largo alcance y baja potencia específica para servicios IoT, mientras que LTE tiene como objetivo satisfacer los requerimientos de la 3GPP (3rd Generation Partnership Project) para redes de cuarta generación (4G) y que a partir del Release 12 nace el concepto de LTE-M ya que empieza a considerar la comunicación tipo maquina (MTC, Machine Type Communication), el Release se enfoca en definir una nueva categoría de terminal manteniendo la misma red LTE pero considerando una configuración específica para dicho terminal. Es por eso que una comparación entre estas puede ayudar a entender las posibilidades que ofrecen.

En este trabajo se desarrollan dos escenarios simulados en NS3 (Network Simulator 3) escritos en C++ usando módulos de LoRaWAN y LTE ya existentes dentro de la comunidad NS3, para que sean escenarios comparables la configuración del sistema es la misma para las dos tecnologías, excepto para la frecuencia (915 MHz para LoRaWAN y 781 MHz para LTE) y ancho de banda (250 kHz para LoRaWAN y 1.4 MHz para LTE) donde se procura configurar lo más cercanos posibles. Los parámetros considerados para evaluar las tecnologías son: throughput y packet lost, el área de cobertura será la variable para las distintas simulaciones.

La topología de red de los escenarios es la misma para los dos casos y está comprendida por un gateway al cual se conectan hasta 100 sensores. La ubicación de los sensores puede ser en una área circular o en una área cuadrada, la ubicación circular es definida mediante un algoritmo que busca la ubicación óptima dentro de una área circular maximizando la distancia entre sensores; este algoritmo es escrito en Matlab.

Los resultados obtenidos muestran que estadísticamente el desempeño de LoRaWAN es mejor que LTE en los escenarios propuestos: para una distribución óptima de sensores es 1302,7 bps mayor en throughput y 64,85 % menor en packet lost y para la ubicación en grilla cuadrada es 1292,24 bps mayor en throughput y 64,58 % menor en packet lost. El desempeño de la ubicación en el área circular es similar al desempeño de la ubicación en el área cuadrada (diferencias de 17,92 bps y 7,46 bps en throughput y 0,64 % y 0,3 % en packet lost).

Agradecimientos

A todas las personas que directa o indirectamente han sido parte de mi vida este par de añitos, y a la SENESCYT por el auspicio.

Tabla de Contenido

Introducción **1**
Motivación . . . 1
Objetivo General . . . 1
Objetivos Específicos . . . 2
Hipótesis . . . 2
Metodología . . . 2

1. Marco Teórico **4**
1.1. Tecnologías IoT . . . 4
1.1.1. LoRaWAN . . . 4
Bit Rate, Simbol Rate y Chirp Rate . . . 6
Canales de Frecuencia . . . 6
Especificaciones LoRaWAN [6] . . . 7
Formato de Mensajes . . . 8
1.1.2. LTE . . . 9
Arquitectura y Stack de Protocolos LTE . . . 10
Capa física LTE . . . 13
LTE-M . . . 17
1.2. Software . . . 18
1.2.1. NS3 . . . 18
LoRaWAN NS3 Module . . . 18
LENA . . . 19
1.2.2. Matlab . . . 20
1.2.3. R . . . 20
1.3. Herramientas Estadísticas . . . 20
Test de Normalidad Shapiro-Wilk . . . 20
Test de Outliers Grubbs . . . 21
Test de Suma de Rangos de Wilcoxon . . . 21
Análisis de Regresión . . . 22

2. Simulación **23**
2.1. Topología . . . 23
2.2. LoRaWAN . . . 24
2.3. LTE . . . 25
2.4. Algoritmo de Optimización . . . 26
2.5. Configuración y Resultados . . . 28

3. Resultados y Análisis de Datos **29**
3.1. Gráficas de Datos 29
3.2. Análisis Estadístico de Datos 32
3.2.1. Modelos Throughput y Packet Lost 34

Conclusión **38**

Bibliografía **39**

Anexos **41**
Código NS3 LoRaWAN 41
Código NS3 LTE 48
Código Matlab 53
Código R 55

Índice de Tablas

1.1. Canales LoRaWAN 7
1.2. Releases LTE [14]. 10
1.3. Descripción Nodos LTE [8]. 11
1.4. Interfaces LTE [8]. 11

2.1. Condiciones de Configuración. 28

3.1. Media, Desviación Estándar y P-value, Óptimo. 33
3.2. Media, Desviación Estándar y P-value, Grilla. 33
3.3. Media, Desviación Estándar y P-value, Óptimo. 33
3.4. Media, Desviación Estándar y P-value, Grilla. 34
3.5. Estadística Descriptiva. 34
3.6. Modelos Throughput. 35
3.7. Modelos Packet Lost. 35
3.8. Comparación Parámetros. 36

Índice de Ilustraciones

1.1. Técnicas Spread Spectrum. 5
1.2. Relación del SF con la duración del chirp. 5
1.3. Solución LoRaWAN. 7
1.4. Topología LoRaWAN [11]. 8
1.5. Estructura del Mensaje [11]. 9
1.6. Red LTE [8]. 10
1.7. Stack de Protocolos Plano de Usuario [8].. 12
1.8. Stack de Protocolos Plano de Control [8]. 12
1.9. Prefijo Cíclico [19]. 14
1.10. Transmisión de Datos OFDMA. 15
1.11. Matriz Tiempo-Frecuencia OFDMA y SC-FDMA. 15
1.12. Trama, Subtrama, Slot, RB y RE en LTE. 16
1.13. Mapeo de Canales Up-link [9]. 17
1.14. Mapeo de Canales Down-link [9]. 17
1.15. Arquitectura LTE LENA [20]. 19

2.1. Flujo de Desarrollo de Simulación. 23
2.2. Topología IoT a Simular. 24
2.3. Topología IoT a Simular. 24
2.4. Estructura Código LoRaWAN. 25
2.5. Estructura Código LTE. 26
2.6. Ubicación Óptima. 27

3.1. Área de Cobertura. 30
3.2. Throughput LoRaWAN. 30
3.3. Throughput LTE. 30
3.4. Punto de Cambio de Tecnología. 31
3.5. Punto de Cambio de Tecnología. 31
3.6. Packet Lost LoRaWAN. 32
3.7. Packet Lost LTE. 32
3.8. Comparación 1. 36
3.9. Comparación 2. 36
3.10. Comparación 3. 37
3.11. Comparación 4. 37

Acrónimos

IoT	Internet of Things.
MTC	Machine-Type Communication.
LTE	Long Term Evolution.
LoRaWAN	Long Range Wide Area Network.
M2M	Machine to Machine.
3GPP	3rd Generation Partnership Project.
GCC	GNU Compiler Collection.
LPWAN	Low Power Wide Area Network.
FHSS	Frequency Hopping Spread Spectrum.
DSSS	Direct Sequence Spread Spectrum.
CSS	Chirp Spread Spectrum.
SF	Spreading Factor.
CRC	Cyclic Redundancy Check.
ISM	Industrial, Scientific and Medical.
3GPP	3rd Generation Partnership Project.
OFDMA	Orthogonal Frequency Division Multiple Access.
SC-FDMA	Single Carrier Frequency Division Multiple Access.
CMAS	Commercial Mobile Alert System.
MTC	Machine Type Communications.
NB-IoT	NarrowBand IoT.
EPS	Evolved Packet System.
E-UTRAN	Evolved Universal Mobile Telecommunications System Terrestrial Radio Access Network.
EPC	Evolved Packet Core.
PDN	Packet Data Network.
eNB	Evolved NodeB.
MME	Mobility Management Entity.
RRM	Radio Resource Management.
EMM	EPS Mobility Management.
ESM	EPS Session Management.
NAS	Non-Access Stratum.
S-GW	Serving Gateway.
P-GW	Packet Data Network Gateway.
PCRF	Policy and Charging Control Function.
SDF	Service Data Flow.
PCC	Policy and Charging Control.

OCS	Online Charging System.
CDR	Charging Data Records.
QoS	Quality of Service.
E-RAB	E-UTRAN Radio Access Bearer.
PDCP	Packet Data Convergence Protocol.
RLC	Radio Link Control.
MAC	Medium Access Control.
OFDMA	Orthogonal Frequency Division Multiple Access.
SC-FDMA	Single Carrier Frequency Division Multiple Access.
MIMO	Multiple Input Multiple Output.
PAPR	Peak to Average Power Ratio.
IFFT	Inverse Fast Fourier Transform.
BCH	Broadcast Channel.
PCH	Paging Channel.
DL-SCH	Downlink Shared Channel.
MCH	Multicast Channel.
UL-SCH	Uplink Shared Channel.
RACH	Random Access Channel.
MBMS	Multimedia Broadcast Multicast Services.
DRX	Discontinuous Reception.
NS3	Network Simulator 3.
LENA	LTE-EPC Network SimulAtor.
RMSE	Root Mean Square Deviation.
AIC	Akaike Information Criteria.

Introducción

Motivación

Internet de las cosas permite tener prácticamente cualquier dispositivo conectado a la red y así satisfacer nuevas necesidades. Un escenario IoT involucra el concepto de comunicación maquina a máquina (M2M), incluye cosas acopladas a sensores y módulos de comunicación conectados a Internet que envían datos a un servidor remoto donde se hace un procesamiento de datos para tomar alguna acción en particular sin la intervención del ser humano.

IoT requiere una red de distribución con características diferentes a las redes tradicionales como LTE o WiFi, estas son: largo alcance, bajas tasas de transmisión y ahorro de energía por parte del terminal. Este problema ha permitido el desarrollo de redes LPWAN (WAN para baja potencia) tales como: LTE-M, LoRaWAN, Sigfox.

Para el análisis se escoge LoRaWAN y LTE-M que son dos de las tecnologías más populares, por un lado LoRaWAN es nueva en el mercado y de fácil acceso y LTE-M define una nueva categoría de terminal usado sobre la red LTE con configuración específica y con un alto porcentaje de cobertura mundial. El desempeño de cada una puede intuirse dadas las características técnicas de cada tecnología, por ejemplo LTE ofrece mayor capacidad y LoRaWAN mayor cobertura, por lo que la decisión de escoger una u otra tecnología para desarrollar un proyecto depende de la aplicación.

De esto deriva la necesidad de tener referencias objetivas que permitan evaluar las prestaciones de estas dos tecnologías en un escenario específico y bajo condiciones específicas. Por otro lado en un escenario IoT los terminales tienen altas probabilidades de estar fijos, bajo estas consideraciones es útil saber si la ubicación física de los terminales tiene o no una influencia considerable en el desempeño del sistema.

Objetivo General

Con lo anteriormente descrito se definen dos objetivos generales:

- Comparar el desempeño de LoRaWAN vs LTE.
- Comparar el desempeño en dos diferentes formas de ubicación de sensores: una óptima

circular versus otra en grilla cuadrada.

Objetivos Específicos

Para cumplir con los objetivos generales se debe cumplir con:

- Desarrollar un escenario LoRaWAN en NS3.
- Desarrollar un escenario LTE en NS3.
- Desarrollar un algoritmo de optimización de ubicación en Matlab.
- Analizar los resultados de las simulaciones.

Hipótesis

Se definen dos hipótesis a comprobar:

- El desempeño de LoRaWAN es mejor que el desempeño de LTE bajo las mismas condiciones
- El desempeño de una distribución óptima de los terminales en una área circular es similar a la distribución en una grilla cuadrada.

Metodología

Se parte con la preparación del software necesario a utilizar; para el desarrollo del algoritmo de ubicación de sensores se usa Matlab y para el análisis estadístico se usa R, estos se instalan sobre Windows. Para el caso de NS3 este funciona mejor sobre Linux, entonces se usa una máquina virtual en VirtualBox sobre la que se instala Ubuntu 16.04, los requisitos mínimos para ejecutar simulaciones básicas son: un compilador GCC (GNU Compiler Collection) y un intérprete de Python, con estos prerrequisitos se instala y prueba NS3.

NS3 es software libre y hay una comunidad enorme desarrollando módulos. Para el presente trabajo se usa un módulo LoRaWAN y uno LTE (Constituido por objetos y clases que simulan el comportamiento de la tecnología), con estas herramientas disponibles se construye el código en C++ que define el escenario de red propuesto para cada tecnología. El algoritmo de optimización de ubicación escrito en Matlab tiene como salidas los puntos de ubicación optima referenciados a un círculo de área igual a uno y estos valores son puestos en el código de NS3.

Para las dos tecnologías se realizan las mismas simulaciones. La variable considerada es el área de cobertura calculada en función del radio del círculo, en cada área se hacen las simulaciones y se obtiene valores de throughput y packet lost.

Con los datos obtenidos se generan gráficas que permitan ver el contraste ente tecnologías y el contraste entre las formas de ubicación de los sensores. Se usan algunas herramientas estadísticas para ajustar los datos a un modelo y a estos aplicarle un test de hipótesis que permita comparar las muestras.

Capítulo 1

Marco Teórico

Este capítulo describe las características generales de las tecnologías LoRaWAN y LTE consideradas. Las características generales del software utilizado (NS3, Matlab y R) y una descripción de los módulos disponibles para estas tecnologías así como también una descripción de las herramientas estadísticas usadas.

1.1. Tecnologías IoT

Dentro del escenario de redes inalámbricas que pueden dar apertura o soporte a las IoT se tiene: tecnologías de largo alcance como las redes tradicionales WiFi y celulares (2G, 3G y 4G), así como también, redes ideadas para aplicaciones IoT (LoRaWAN, Sigfox); por otro lado están las redes de corto alcance como: ZigBee, 6LowPAN; y otras que habilitan la posibilidad de comunicarse entre terminales como Bluetooth [3]. El presente documento se concentra en el análisis comparativo entre dos tecnologías de largo alcance pensadas para aplicaciones de las IoT: LoRaWAN y LTE.

1.1.1. LoRaWAN

Entorno a LoRaWAN está definida la LoRa Alliance, que es un organismo que pretende consolidar esta tecnología como la referente para las IoT. Existen al momento más de 100 miembros que cooperan en este proyecto, dentro de los cuales están: Cisco, Semtech, Microchips, IBM, SKTelecom, etc.

Usa modulación LoRa misma que está patentada por SEMTECH y hace referencia a una tecnología de capa física que permite coexistir con arquitecturas de red existentes en capas superiores. Mantiene un ancho de banda constante y utiliza factores de dispersión (SF) ortogonales para controlar la capacidad, es robusto frente a la degradación del canal y tiene requerimientos de baja potencia de transmisión, lo que le hace atractivo para aplicaciones en las IoT [17].

Existen varias técnicas para ensanchar el espectro, las más usadas son: Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS) y Chirp Spread Spectrum (CSS). FHSS usa saltos en frecuencia cuyo orden depende de un código de ensanchamiento, aquí el pedazo de espectro tiene un ancho de banda igual al de la señal original (Bluetooth usa esta técnica); mientras que DSSS es modulado directamente sobre el código de ensanchamiento, aquí el ancho de banda modulado generalmente es mayor al ancho de banda de la señal original (GPS usa esta técnica) [2].

CSS nació para aplicaciones de radar en los años 40s, esta técnica no usa un código de ensanchamiento, sino que hace un barrido a lo largo de todo el espectro disponible mediante un comportamiento de rampa lineal al cual se le conoce como chirp, característica que permite la fabricación de equipos no tan costosos [2]. La Figura 1.1 muestra el comportamiento de las técnicas descritas.

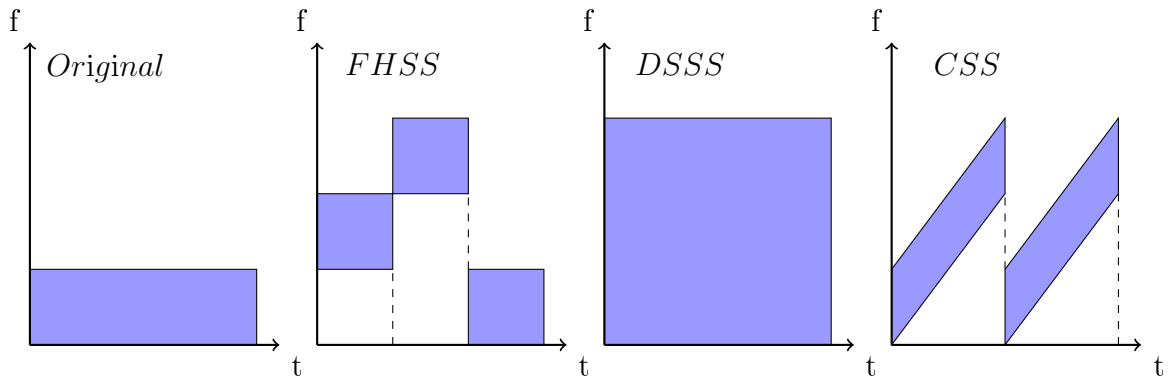

Figura 1.1: Técnicas Spread Spectrum.

El chirp puede ser up-chirp si hay un incremento de la frecuencia y es down-chirp si por el contrario hay un decremento de la frecuencia. El SF está relacionado con la duración del chirp como se puede ver en la Figura 1.2.

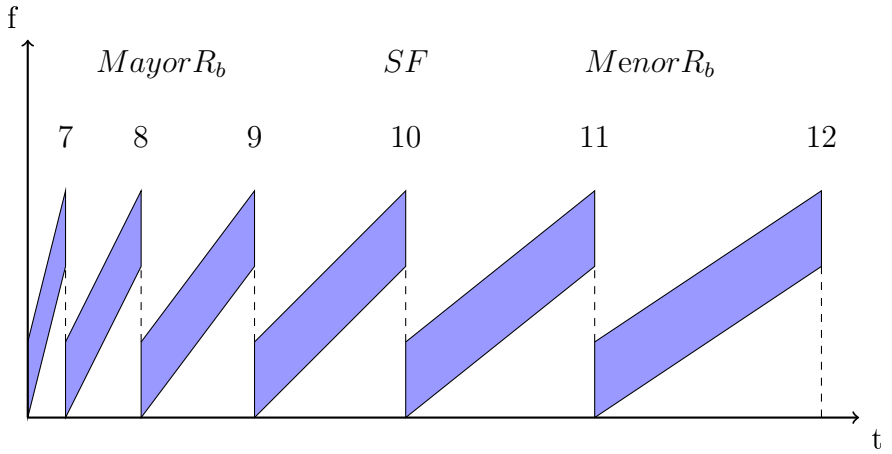

Figura 1.2: Relación del SF con la duración del chirp.

Bit Rate, Simbol Rate y Chirp Rate

El Bit Rate(R_b), Simbol Rate (R_S)and Chirp Rate (R_C) se definen en las ecuaciones (1.1), (1.2) y (1.2) respectivamente.

$$R_b = SF \cdot \frac{1}{\left(\frac{2^{SF}}{BW}\right)} \quad \left[\frac{b}{s}\right] \tag{1.1}$$

$$R_S = \frac{1}{T_S} = \frac{BW}{2^{SF}} \quad \left[\frac{S}{s}\right] \tag{1.2}$$

$$R_C = R_S \cdot 2^{SF} \quad \left[\frac{chips}{s}\right] \tag{1.3}$$

$SF \rightarrow$ Spreading Factor, puede tomar valores: 7, 8, 9, 10, 11, 12

$BW \rightarrow$ Bandwidth [Hz]

$T_S \rightarrow$ Simbol Time [s]

LoRa dispone de 4 esquemas de corrección de errores: $\frac{4}{5}$, $\frac{4}{6}$, $\frac{4}{7}$, $\frac{4}{8}$, mismos que pueden seleccionarse con el parámetro Code Rate CR (1,2,3 ó 4) según la relación: $Esq = \frac{1}{4+CR}$; con esta consideración el R_b estaría definido según la Ecuación (1.4). El SF es la relación logarítmica entre la tasa de chirp y la tase de símbolo.

$$R_b = SF \cdot \frac{\frac{1}{1+CR}}{\left(\frac{2^{SF}}{BW}\right)} \quad \left[\frac{b}{s}\right] \tag{1.4}$$

Canales de Frecuencia

LoRaWAN describe la arquitectura de red y el protocolo de la capa MAC. Las consideraciones de diseño buscan optimizar el consumo de potencia, definiendo los dispositivos en tres clases, además se definen las frecuencias y el ancho de banda del canal [5], esto puede visualizarse en la Figura 1.3.

La banda para uso no comercial de frecuencias ISM (Industrial, Scientific and Medical) es la usada por LoRa, el ancho de banda es en la mayoria de casos 125 kHz, pero también se podría usar 250 kHz y 500 kHz (esto para la banda de 915 y 868 MHz), las asignaciones de canales específicos se ve con detalle en las especificaciones de parámetros regionales de LoRaWAN [7]. La Tabla 1.1 resume la asignación de frecuencias para distintos lugares, las consideraciones especificas se las puede ver con detalle en [7].

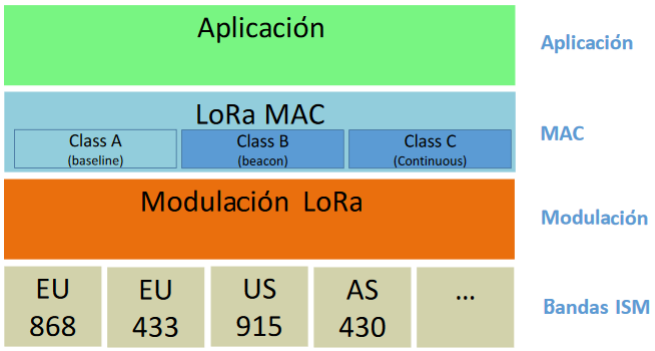

Figura 1.3: Solución LoRaWAN.

Código Región	Banda IMS [MHz]	Ejemplo de Canal [MHz]
EU 863-870	863-870	868,1
US 902-928	902-928	902,3
China 779-787	779-787	779,5
EU 433	433	433,175
Australia 915-928	915-928	915,2
CN 470-510	470-510	470,3
AS 923	923	923,2
South Korea 920-923	920-923	920,9
India 865-867	865-867	865,0625

Tabla 1.1: Canales LoRaWAN

Especificaciones [6]

Define un esquema de red con una topología en estrella donde un gateway concentra los mensajes de los sensores conectados, y este tráfico se transmite a un servidor remoto mediante una conexión IP estándar (ver Figura 1.4), la comunicación es bidireccional aunque la comunicaciones del sensor al gateway (up-link) es predominante. La tasa de datos (R_b) teórica puede variar entre 0.3 a 50 kbps.

Se define también tres tipos de terminales diferenciados como Clase A, B y C,

- Clase A.- Define al dispositivo básico, tiene función bidireccional y se caracteriza por tener dos ventana muy cortas para la comunicación del gateway al sensor (down-link), es el que menos consumo de energía tiene.
- Clase B.- Esta tiene la funcionalidad del de Clase A sumado a la posibilidad de agendar ventanas de comunicación adicionales para el down-link, el consumo de energía es mayor.
- Clase C.- Esta clase tiene ventanas de comunicación down-link mucho mayores, casi continuas, y es el que más energía consume.

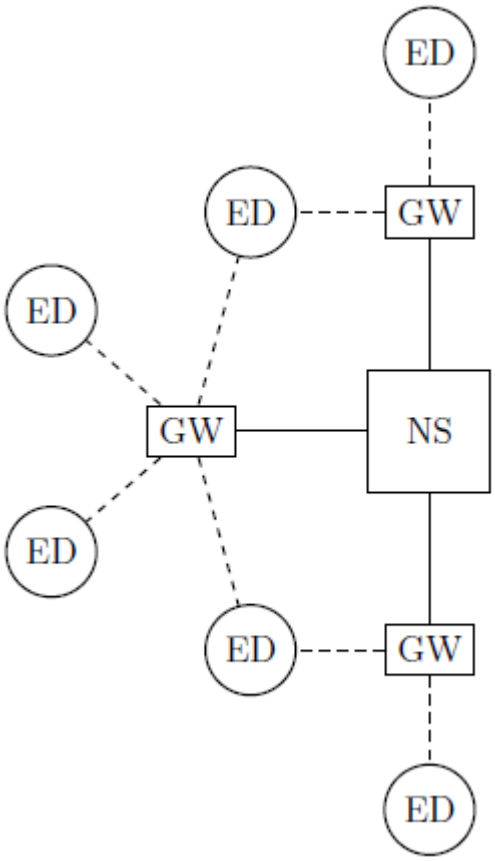

Figura 1.4: Topología LoRaWAN [11].

Formato de Mensajes

Hay dos tipos de mensajes de capa física, uno para el up-link y otro para el down-link, la diferencia entre estos es que el up-link contiene un campo opcional para chequeo de redundancia cíclica (CRC). Como se puede ver en la Figura 1.5, la estructura del campo preámbulo puede cambiar dependiendo de la región, el *Prembulo* inicia con una secuencia constante de up-chirps que permiten encontrar el inicio de la trama, el campo *Cabecera Phy* contiene la dirección, el detalle de campos es limitado debido a la naturaleza propietaria de LoRa.

Respecto a la capa MAC, el campo *Cabecera MAC* contiene información acerca de la versión del estándar, del dispositivo que esta siendo usado y acerca del tipo de mensaje (join, data o proprietary), el campo *Payload FRAME* contiene datos de la capa de aplicación, el campo *Puerto FRAME* indica a que aplicación corresponde, el campo *Direccion Dispositivo* tiene 4 bytes, el campo *Control FRAME* es un byte para ACK y el campo *Opciones FRAME* ayuda a soportar la opción de velocidad de datos adaptativa (ADR).

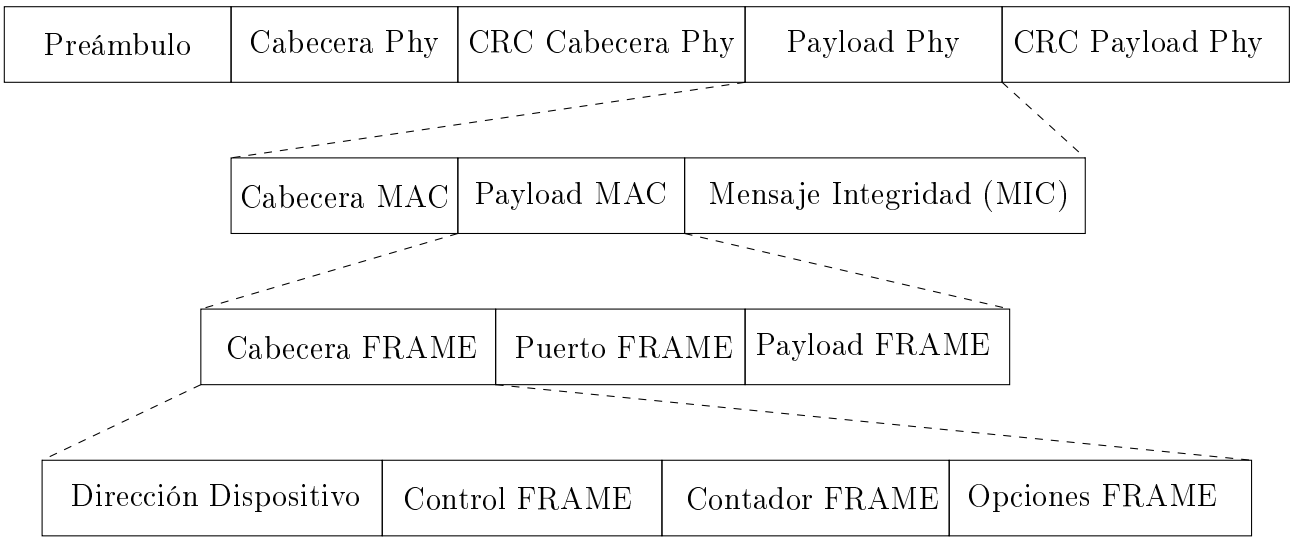

Figura 1.5: Estructura del Mensaje [11].

1.1.2. LTE

Es un estándar desarrollado por la 3GPP que define una red de acceso y un core para servicios móviles cuyos consideraciones más importantes son: aumento de la tasa de datos y eficiencia espectral, disminución de latencia. El core de LTE está basado en IP y el acceso usa OFDMA para el down-link y SC-FDMA para el up-link [1]. Una de las opciones que ofrece es LTE-M para las IoT, la cual no requiere modificación en la red LTE, y se concentran en disminuir el consumo de potencia del terminal definiendo nuevas categorías del mismo.

El desarrollo de LTE parte en el 2008 con el Release 8 y hoy se tiene el Release 13 con algunas consideraciones para IoT, la Tabla 1.2 resume las características de todos los Releases LTE. Los Releases siguientes (14, 15 y 16) van a definir las redes de quinta generación (5G) y tendrán mejoras para aplicaciones IoT.

Release	**Características**
8	Primer estándar LTE. Tasas de 300 Mbps (down-link) y 75 Mbps (up-link). Ancho de banda flexible. Latencia baja (5 ms). Acceso OFDMA (down-link) y SC-FDMA (up-link). Red IP. Multiples antenas (MIMO).
9	Mejora Release 8. Incluye sistema de alerta móvil (CMAS). Introduce femtoceldas. Redes autoconfigurables.
10	LTE Advanced (4G real). Mejora las tasas a 1 Gbps (down-link) y 500 Mbps (up-link).

Release	Características
11	Mejora LTE Advanced. Primera consideración para comunicación MTC para IoT.
12	Mejora adicional LTE Advanced. Define UE (Cat 0) para operaciones MTC.
13	LTE para aplicaciones IoT. UE para 1.4 MHz (LTE-M y UE Cat. M1) y para 200 kHz (NB-IoT y UE Cat. NB1).

Tabla 1.2: Releases LTE [14].

Arquitectura y Stack de Protocolos LTE

La red LTE se la conoce como EPS (Evolved Packet System) la cual esta dividida en: E-UTRAN (acceso de radio), EPC (core) y el UE (terminal). El E-UTRAN está constituido por la radio base llamada eNB, y el EPC tiene los siguientes nodos: S-GW, P-GW, MME, HSS, PCRF, SPR, OCS y OFCS, estos nodos son generalmente lógicos. La red puede conectarse a una PDN (red de paquetes IP) que puede ser interna o externa a la red LTE, ver Figura 1.6.

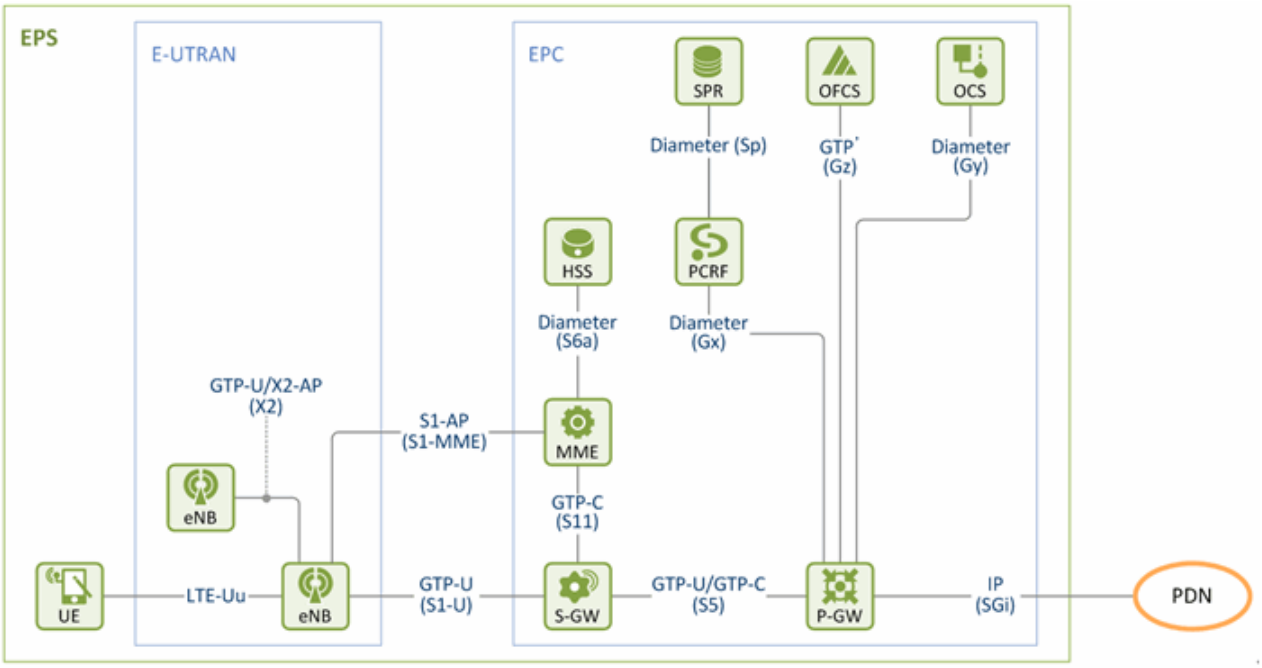

Figura 1.6: Red LTE [8].

Las funciones generales de cada nodo del EPS se describe en la Tabla 1.3, por otro lado la Tabla 1.4 resume las interfaces, protocolos que usan y si estos corresponden al plano de control o al plano de usuario. Las interfaces entre nodos del sistema definen un stack de protocolos específico diferenciados entre los que corresponden al plano de usuario y los que corresponden al plano de control, algunos protocolos se usan en los dos casos, ver las Figuras 1.7 y 1.8 respectivamente.

Nodo	Descripción
UE	Terminal de usuario.
eNB	Encargado de la gestión de recursos de radio (RRM), es decir funciones como: asignación dinámica de recursos, control de movilidad, interferencia entre celdas, etc.
S-GW	Es el punto de conexión al E-UTRAN, es el punto de anclaje para cambio de eNB. Es un nodo de datos de usuario.
P-GW	Es el punto de conexión a la PDN, es el punto de anclaje para cambio de red. Es un nodo de datos de usuario.
MME	Principal nodo de control del E-UTRAN, se comunica con el nodo HSS para obtener información del usuario, también proporciona a los usuarios gestión de movilidad EPS (EMM) y gestión de sesión EPS (ESM) controlado por NAS (es una capa del plano de control y un protocolo).
HSS	Provee autenticación y es la base de datos con información de todos los usuarios.
PCRF	Toma decisiones de los SDFs (que son los flujos de datos de servicio), también define y aplica el PCC (que es un proceso de políticas definidas por el operador para los usuarios de acuerdo al servicio contratado).
SPR	Provee de información de los usuarios al PCRF para la definición de reglas PCC.
OCS	Control del crédito en tiempo real.
OFCS	Genera registros de datos de cobro (CDR) para la facturación.

Tabla 1.3: Descripción Nodos LTE [8].

Interfaz	Protocolo	Plano de Control	Plano de Usuario
LTE-Uu	E-UTRA	SI	SI
X2	X2-AP	SI	NO
X2	GTP-U	NO	SI
S1-U	GTP-U	NO	SI
S1-MME	S1-AP	SI	NO
S11	GTP-C	SI	NO
S5	GTP-C	SI	NO
S5	GTP-U	NO	SI
S6a	Diameter	SI	NO
Sp	Diameter	SI	NO
Gx	Diameter	SI	NO
Gy	Diameter	SI	NO
Gz	GT'	SI	NO
SGi	IP	SI	SI

Tabla 1.4: Interfaces LTE [8].

La Figura 1.7 muestra los protocolos del plano de usuario, las interfaces relacionadas a estos son: LTE-Uu, S1-U, S5 y X2. El protocolo asociado a la interfaz LTE-Uu es E-UTRA (termino que hace referencia a los protocolos PDCP, RLC y MAC) y el protocolo asociado a

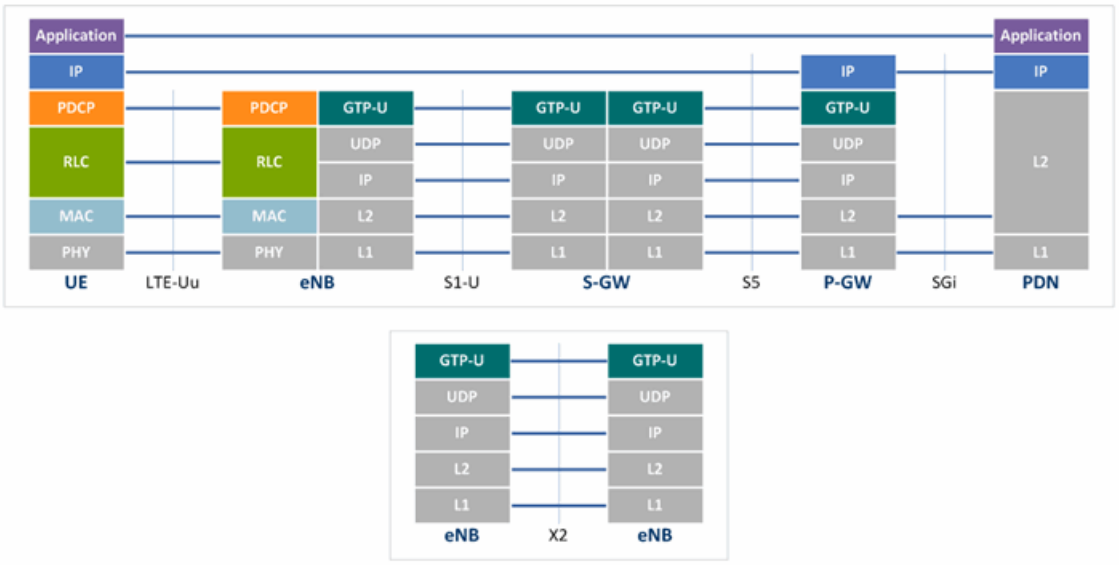

Figura 1.7: Stack de Protocolos Plano de Usuario [8]..

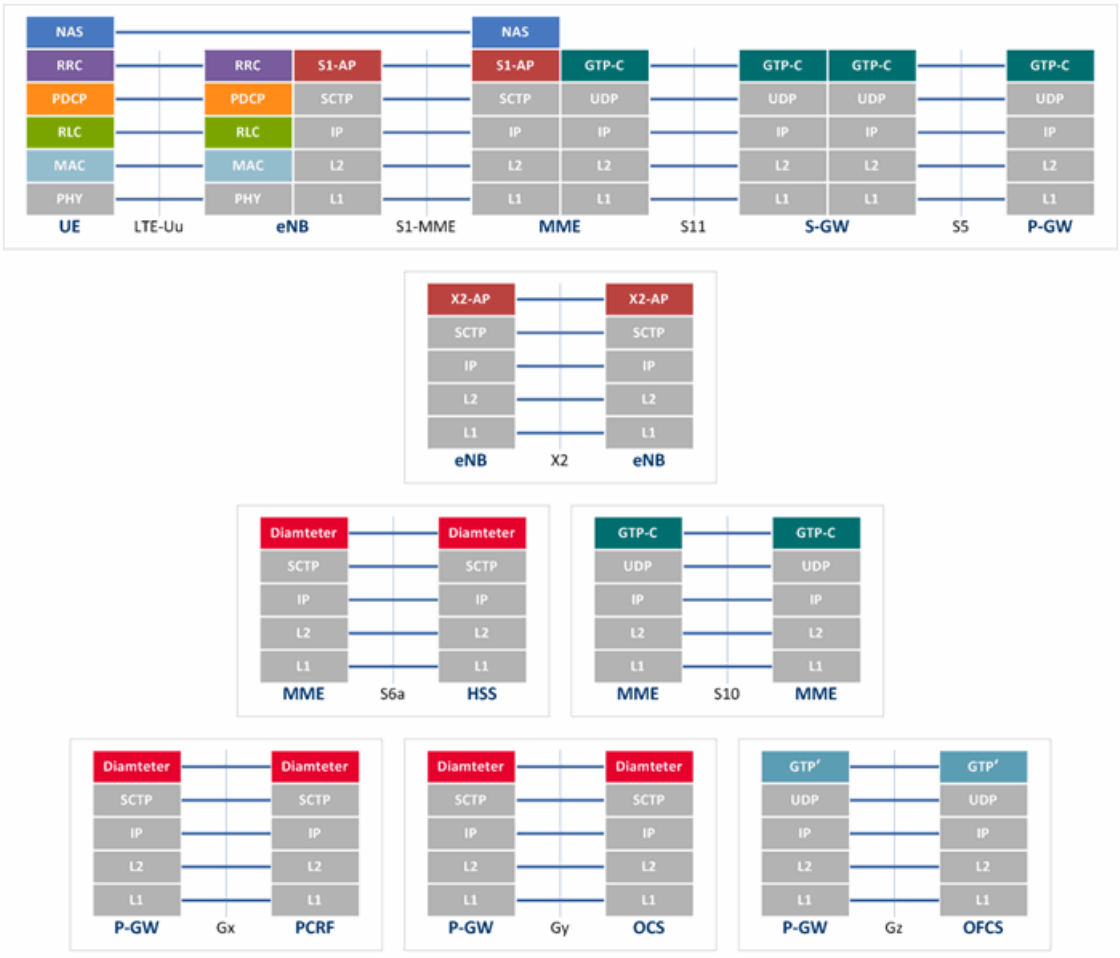

Figura 1.8: Stack de Protocolos Plano de Control [8].

las interfaces S1-U, S5 y X2 es GTP-U, protocolos cuya función se describe a continuación:

- PDCP.- Ayuda al transporte eficiente de paquetes IP a través del enlace de radio, comprime el header, se encarga del cifrado e integridad de los datos y el reordenamiento y reenvío de paquetes en el handover.
- RLC.- En el lado de transmisión se encarga de la construcción, segmentación y concatenación del PDU y en el lado de recepción lo contrario, realiza el reordenamiento y retransmisión de paquetes, puede operar de tres modos diferentes: transparente, reconocido y no reconocido.
- MAC.- Es el límite con la capa física, está conectado a la capa RLC mediante canales lógicos y a la capa física mediante canales de transporte, se encarga de multiplexar y demultiplexar entre canales lógicos y de transporte.
- GTP-U.- Usado para reenviar paquetes IP estableciendo un tunel GTP.

La Figura 1.8 muestra los protocolos del plano de control, las interfaces relacionadas a estos son: LTE-Uu, X2, S1-MME, S11, S5, S10, S6a, Gx, Gy y Gz. El protocolo asociado a la interfaz LTE-Uu es E-UTRA (termino que hace referencia a los protocolos PDCP, RLC, MAC, NAS y RCC), PDCP, RLC y MAC tienen la misma función que el plano de usuario; la descripción de las funciones de los protocolos y la relación con las interfaces se detalla a continuación:

- NAS.- Administra la movilidad y los radio bearers (QoS).
- RRC.- Gestión eficiente de los recursos de radio mediante las funciones: transmisión de información del sistema, configuración, reconfiguración y liberación de la conexión RRC, así como también la modificación, configuración y liberación de los radio bearers.
- X2AP.- Relacionada con la interfaz X2; este protocolo permite la movilidad de los terminales mediante el reenvío de datos del usuario, intercambio de información del estado de los recursos, carga de tráfico.
- S1AP.- Relacionada con la interfaz S1-MME; permite el servicio de señalización entre el E-UTRAN y el EPC, también el transporte de señalización NAS.
- GTP-C.- Relacionada con las interfaces S11/S5/S10; permite el intercambio de información de control para la administración de túneles GTP.
- Diameter.- Permite el intercambio de información de: usuarios entre el HSS y MME en la interfaz S6a, de políticas PCC desde el PCRF al PG-W en la interfaz Gx y de información de control de crédito entre el P-GW y el OCS en la interfaz Gy.
- GTP'.- Permite la transferencia de CDRs del P-GW al OFCS en la interfaz Gz.

Capa física LTE

Para satisfacer los requerimientos de LTE se usa en la capa física dos tecnologías: Multiple Antenna Technology (MIMO) y Multicarrier Technology (OFDMA y SC-FDMA), OFDMA se usa para el down-link (comunicación del eNB al UE) y SC-FDMA se usa para el up-link (comunicación del UE al eNB), la razón por la que se usa SC-FDMA para el up-link es debido a que este tiene un menor consumo de energía: el parámetro PAPR (promedio de potencia de

radio peak), es mayor en OFDMA que en SC-FDMA, lo que traduce en un requerimiento de un amplificador de potencia más grande para el eNB y un ahorro de energía para el terminal. OFDMA y SC-FDMA son técnicas de acceso múltiple en frecuencia, es decir que el ancho de banda se divide en sub-portadoras y se asigna a cada usuario una o varias sub-portadoras del total disponible, esta técnica tiene la ventaja de ser robusta frente al multitrayecto y a la interferencia, mientras que la desventaja está relacionada con las exigencias de sincronismo.

Las sub-portadoras OFDM son ortogonales entres sí, lo que permite tener más sub-portadoras en un mismo ancho de banda. El multitrayecto que toma la señal genera interferencia entre sub-portadoras, misma que se la controla dejando un intervalo de guarda en el dominio del tiempo donde se coloca una copia del final de la señal, evitando cambios abruptos en el dominio del tiempo, y evitando componentes de alta frecuencia que pueden alterar la ortogonalidad, a esto se lo conoce como prefijo cíclico (CP), ver Figura 1.9

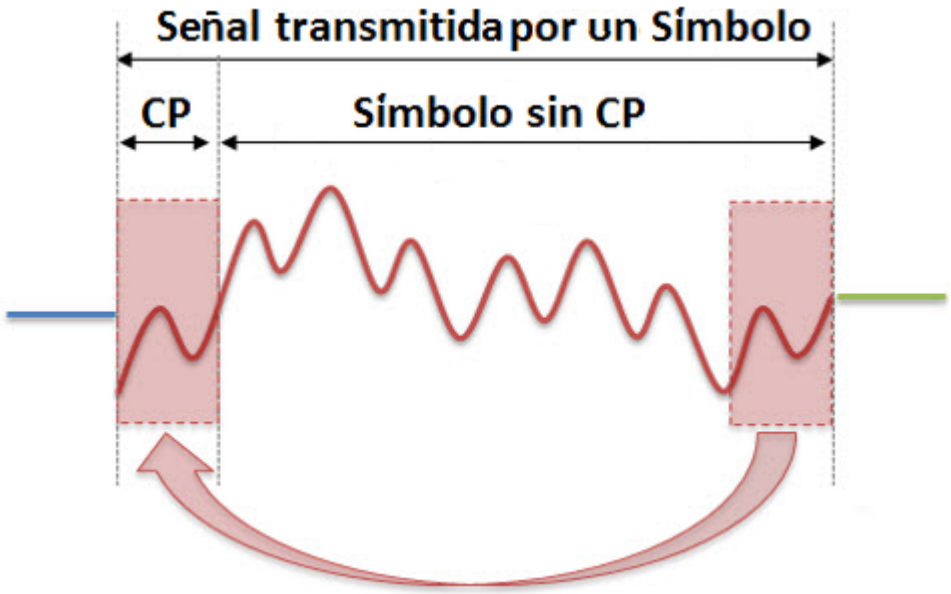

Figura 1.9: Prefijo Cíclico [19].

Cuando se quiere transmitir datos, el número de bits por símbolo está definido por el esquema de modulación (QPSK, 16QAM, 64QAM con 2, 4 o 6 b/S respectivamente,) cada símbolo es multiplicado por la sub-pordadora sobre la cual se va a enviar y después se usa IFFT (transformada rapida de Fourier inversa) para asociar todas las sub-portadoras en una sola señal misma que sera modulada para enviarse a la antena, la IFFT evita la complejidad de implementar tantos osciladores como subportadoras existen. La Figura 1.10 ilustra lo aquí descrito (existen bloques omitidos, como el bloque que convierte el flujo de datos de serie a paralelo o el bloque que añade el prefijo cíclico), [19].

Para el caso de SC-FDMA los datos a transmitir se esparcen en todas las sub-portadoras que ocupan el ancho de banda destinado al usuario. La matriz frecuencia tiempo para OFDMA y SC-FDMA se puede ver en la Figura 1.11 [19].

La trama LTE tiene un tiempo de duración de 10 ms, y la sub-trama 1 ms, lo que significa que en una trama existen 10 sub-tramas, también la sub-trama tiene 2 slots de 0,5 ms, lo que

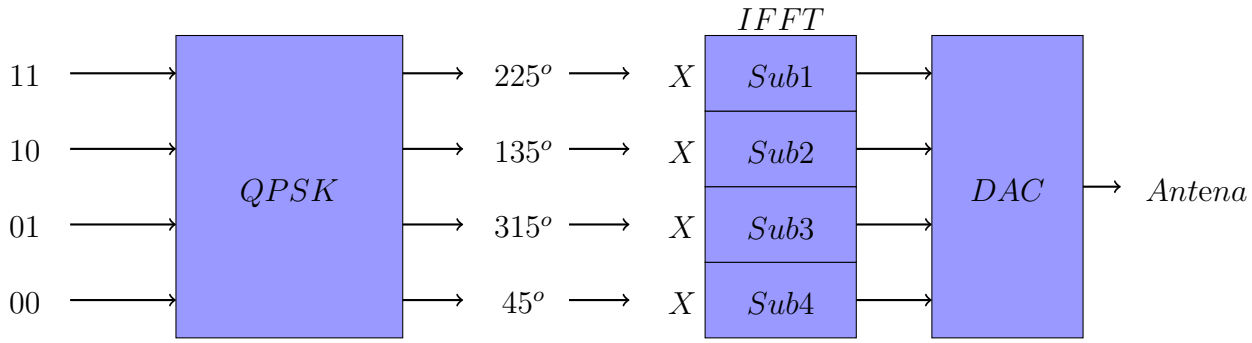

Figura 1.10: Transmisión de Datos OFDMA.

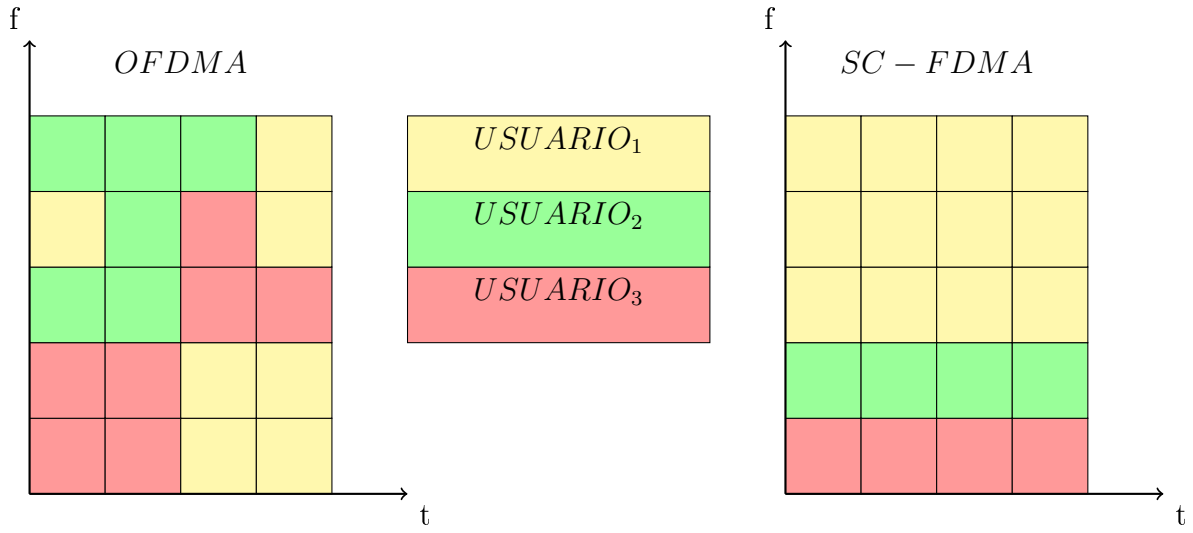

Figura 1.11: Matriz Tiempo-Frecuencia OFDMA y SC-FDMA.

significa que una trama tiene 20 slots, cada slot tiene 7 símbolos OFDM, el tiempo de símbolo depende del ancho de banda de la sub-portadora y de tiempo asignado para el CP, esto es: $Tsim = \frac{1}{15\ kHz} + 4{,}675\ \mu s$. Se define como RB (Resource Block) a un elemento constituido por 12 sub-portadoras y un slot, esto significa, que un RB tiene un ancho de banda de 180 kHz y una duración de 0,5 ms, se define cono RE (Resource Element) a un elemento constituido por 1 sub-portadora y un símbolo OFDM, esto significa, que un RE tiene un ancho de banda de 15 kHz y una duración de 71,34 μs aproximadamente, ver Figura 1.12.

Como se mencionó anteriormente (stack de protocolos) la capa MAC está junto a la capa física y usa canales lógicos y de transporte; a estos canales hay que sumarle los canales físicos que son recursos tiempo-frecuencia. La relación de canales es distinta para el up-link (Figura 1.13) y para el down-link (Figura 1.14).

Cada canal provee características específicas a las capas adyacentes, los canales lógicos proporsionan abstracciones de canales superiores, algunos canales físicos son recursos tiempo-frecuencia relacionados a los canales de transporte. Estos canales de transporte son de especial importancia ya que es donde se maneja el acceso aleatoreo y el scheduling (gestión de recursos) [9]. estos canales son:

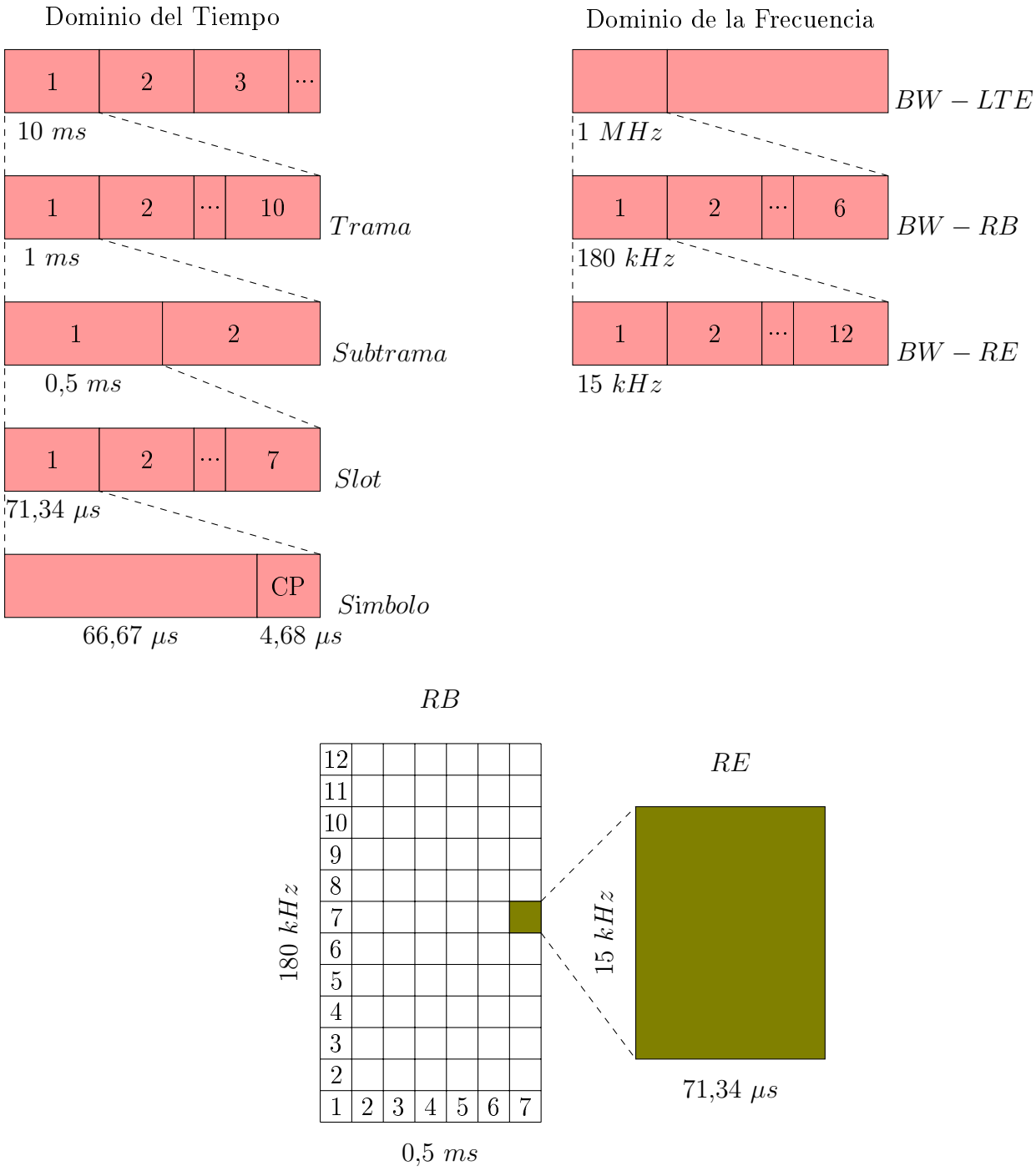

Figura 1.12: Trama, Subtrama, Slot, RB y RE en LTE.

- BCH.- Usado para transmisión de información del sistema (configuración de la red).
- PCH.- usado para ayudar a buscar información sobre usuarios.
- DL-SCH.- Admite técnicas de adaptación de velocidad
- MCH.- Usando para servicios de multimedia (MBMS).
- UL-SCH.- Transmite datos del up-link.
- RACH.- Se usa para acceso aleatorio.

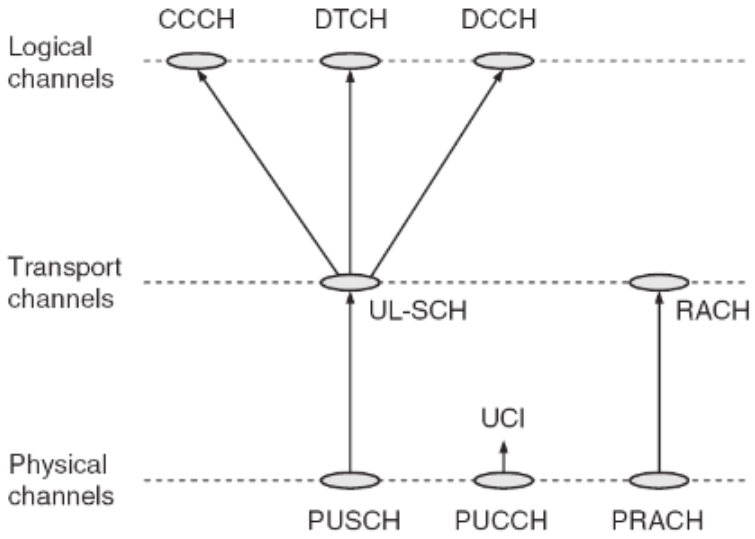

Figura 1.13: Mapeo de Canales Up-link [9].

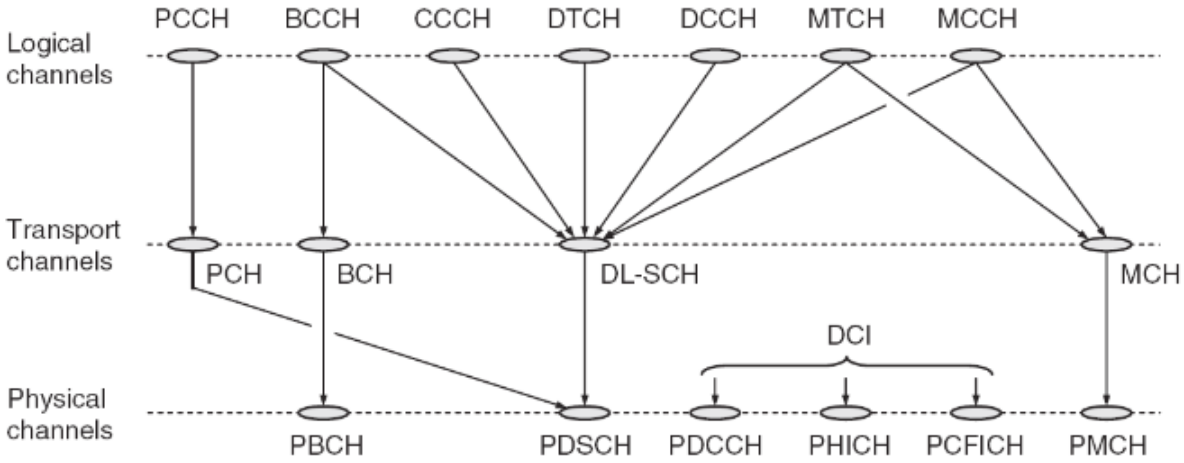

Figura 1.14: Mapeo de Canales Down-link [9].

LTE-M

Algunas características que los Releases 12 y 13 consideran para la comunicacion MTC (LTE-M) [1] se muestran a continuación:

- LTE-M trabaja dentro de una red LTE normal.
- Define nuevas categorías de terminales.
- Trabaja con un ancho de banda de 1.4 MHz o 6 RB (Resource Blocks).
- Agrega características de recepción discontinua (DRX) que permite ahorro de energía al terminal.
- Reducción en la complejidad del sistema de antenas.
- Potencia de transmisión máxima del terminal de 20 dBm.

- Esquemas de modulación QPSK y 16-QAM.
- Puede tener comunicación Half o Full Duplex.
- La referencia en tasa de datos es de: 300 kbps (down-link) y 375 kbps (up-link)

1.2. Software

Las herramientas de software que se usa para el desarrollo de este trabajo son: NS3, Matlab y R, NS3 simula las redes LoRaWAN y LTE, con Matlab se define un algoritmo de ubicación óptima de sensores y con R se hace el análisis estadístico.

1.2.1. NS3

NS3 es un simulador de redes de eventos discretos, tiene fines académicos y de investigación, es un software libre con licencia GNU GPLv2 license. NS3 tiene la característica de disponer de modelos de simulación suficientemente realistas, para tener datos con aproximaciones muy confiables, abarca varias tecnologías como WiFi, WiMAX, LTE, etc. [13].

NS3 es organizado por librerías lo que le hace modular, puede ser compatible con librerías de otros software como Maltab, o incorporarse con software para tener interfaz gráfica. Los modelos en NS3 son hechos en C++ o Python, y el desarrollo de de escenarios se hace en linea de comandos con estos lenguajes. Nace sobre una plataforma Linux aunque ya existen opciones para otros sistemas operativos como Windows; NS3 también usa Mercurial que es una solución para administrar los cambios de sistemas de software complejos, así como también Waf que sirve para compilar los códigos en programas utilizables mediante Python [13].

Para usar NS3, en el caso del presente trabajo se usa una maquina virtual con Linux, sobre el que se prepara, descarga y compila todas las librerías y recursos que requiere.

LoRaWAN NS3 Module

Es un módulo NS3 usado para realizar simulaciones de una red LoRaWAN. La versión inicial de este módulo fue desarrollada como parte de una tesis de la Universidad de Padova [11] y tiene disponible toda la documentación en GITHUB con el nombre de LoRaWAN ns-3 module.

El módulo tiene desarrollado la capa física (tomando como referencia el manual de diseño de SEMTECH [18] para los modelos SX1272/3/6/7/8), la capa MAC (según algunos parámetros definidos para LoRaWAN), el canal de transmisión, una clase para el envió periódico de paquetes. No tiene desarrollado la conexión a un servidor remoto.

LENA

Es un simulador de red de LTE/EPC hecho en NS3 que permite probar y diseñar algoritmos y soluciones LTE, fue diseñado por el CTTC, que es el Centro Tecnológico de Telecomunicaciones de Cataluña [4]. La arquitectura propuesta por LENA se muestra en la Figura 1.15.

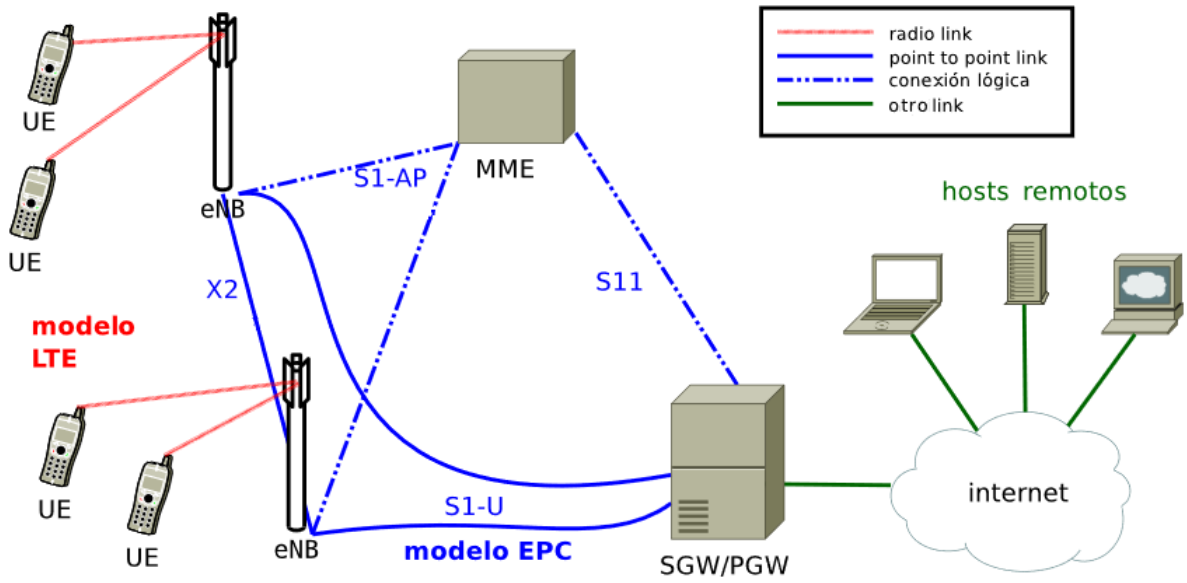

Figura 1.15: Arquitectura LTE LENA [20].

El diseño considera dos componentes principales: el primero es LTE radio que modela las capas RRC, PDCP, RLC, MAC y PHY dentro del terminal (UE) y la radio base (eNB), y el segundo es el EPC que incluye algunos nodos (SGW, PGW y MME, y parte del eNB) e interfaces del core LTE. No todas las funcionalidades del estándar LTE están implementadas, por ejemplo, el modelo EPC tiene desarrollado solo el plano de datos mientras que las funciones del plano de control son resueltas con helpers (objetos del lenguaje de programación).

Respecto a la propuesta de diseño de LENA de LTE, entre otras consideraciones se toma el RB como el más bajo nivel de granularidad, con este nivel se puede asignar recursos correctamente y esto permite que se pueda implementar redes grandes (decenas de eNBs y centenas de UEs), ya que una granularidad a nivel de símbolo requiere una mayor complejidad computacional; respecto a la propuesta de diseño de LENA de EPC, entre otras consideraciones busca dar conectividad IP end-to-end sobre LTE, está disponible solo para IPv4. SGW y PGW están implementados en el mismo nodo [20].

LENA usa muchos modelos para una aproximación mas realista, por ejemplo, los modelos para el canal de propagación consideran: modelo de edificios, de desvanecimiento, de antenas; para una descripción detallada de los modelos que abarca LENA en su diseño se puede visitar [20] en la parte de Design Documentation.

1.2.2. Matlab

Matlab hace referencia a Matrix Laboratory, es una herramienta matemática que dispone de lenguaje de programación propio (lenguaje m) y que se usa para manipulación de matrices, presentación de funciones, análisis de datos, implementación de algoritmos, etc. Es un software propietario y está disponible para distintas plataformas: Windows, Linux, etc. [10]

1.2.3. R

Es un software libre con licencia GNU desarrollado para análisis estadístico y graficación, dispone de un lenguaje propio, desarrollado inicialmente por Bell Laboratories, disponible para distintas plataformas: Windows, Linux, etc. Dentro de las herramientas estadísticas se tiene: modelos lineales y no lineales, pruebas de estadísca clásica, análisis de series de tiempo, calsificación, clustering, etc. [21].

1.3. Herramientas Estadísticas

Para analizar los datos las herramientas estadísticas utilizadas son: test de normalidad, test para outliers, test de medianas, análisis de regresión.

Test de Normalidad Shapiro-Wilk

Este test es usado para muestras pequeñas con un tamaño máximo de 50, para llevar acabo el test se ordenan las observaciones de la muestra y se restan la primera con la última, después la segunda con la penúltima, y así sucesivamente con todas las observaciones, esto define el estadístico W según la Ecuación 1.5.

$$W = \frac{1}{nS^2}\left(\sum_{\mathrm{i}=1}^{h} a_{\mathrm{i}n}(X_{(n-\mathrm{i}+1)} - X_{(\mathrm{i})})\right)^2 \tag{1.5}$$

De la ecuación se tiene que:

$a_{\mathrm{i}n}$ y W están tabulados.

S^2 es la varianza de la muestra.

$h = \frac{n}{2}$ si la muestra es par.

$h = \frac{n-1}{2}$ si la muestra es impar.

El contraste de hipótesis plantea:

Hipótesis nula (H0): La muestra viene de una población normal.

Hipótesis alternativa (H1): La muestra no proviene de una población normal.

Con el valor de W se puede conocer la probabilidad crítica la cual permite evaluar si se rechaza o no la hipótesis nula, esto significa que si $Pc < 0{,}05$ se rechaza la H0 con 95 % de significancia, [16].

Si $Pc < 0{,}05$, entonces NO proviene de una normal.

Si $Pc > 0{,}05$, entonces SI proviene de una normal.

Test de Outliers Grubbs

Utilizado para encontrar valores atípicos de una muestra univariante, para llevar acabo el test se ordenan las observaciones, se escoge un valor sospechoso y se calcula el valor T con la media y desviación estándar de la muestra según la Ecuación 1.6

$$T = \frac{|\bar{X} - X_n|}{Xn} \tag{1.6}$$

Posteriormente este valor T es comparado con unos valor tabulados según nivel de significancia y se concluye si el valor X_n es o no un Outlier, [15].

Test de Suma de Rangos de Wilcoxon

Utilizado para analizar si la distribución de una variable es igual, mayor o menor que otra en dos poblaciones basados en sus muestras, este es un test no paramétrico, el tamaño de las muestras no necesariamente tienen que ser iguales. El proceso parte con ordenar las observaciones para luego asignarles un rango, la observación de valor mas pequeño es de rango 1, el siguiente es de rango 2 y así sucesivamente, cuando hay varias observaciones con en mismo valor se saca el promedio de los rangos correspondientes.

Si los rangos de las observaciones se mezclan aleatoriamente entre las dos muestras significa que son poblaciones parecidas, mientras que si la suma de los rangos de una población es diferente a la suma de los rangos de la otra población significa que las poblaciones son distintas.

El estadístico para este test es la suma de los rangos de cada muestra, Ecuación 1.7.

$$T = \sum_{j=1}^{n} Rango_{(j)} \tag{1.7}$$

Para cuando el tamaño de la muestra es mayor a 15 se debe hacer una aproximación normal según la Ecuación 1.8.

$$z = \frac{T - m_T}{\sigma_T} \tag{1.8}$$

Donde:

$m_T = \frac{1}{2}n1(n1 + n2 + 1)$

$\sigma_T = \frac{n1 \cdot n2(n1+n2+1)}{12}$

Con el valor de z hay que usar tablas para distribución normal y ver el valor de significancia.

Análisis de Regresión

Es un proceso estadístico utilizado para estimar la relación entre variables, concrétamente entre una variable dependiente y una o varias variables independientes. Pueden existir varios tipos de regresiones [12], por ejemplo:

- Regresión Lineal.- Cuando el ajuste de los datos se hace a una función lineal: $y = bo + b1x$.
- Regresión no Lineal.- Cuando el ajuste de los datos se hace a una función no lineal: $y = bo + b1x + b2x^2$.
- Regresión Spline.- Esta técnica divide los datos en segmentos y a estos se ajustan funciones polinomiales, los puntos que segmentan los datos se llaman nudos (knots).
- Modelos aditivos generalizados (GAM).- Estos son como la regresión spline pero los knots son asignados automáticamente.

Capítulo 2

Simulación

Este capítulo describe la topología definida a simular, la estructura de los códigos de simulación LoRaWAN y LTE, junto con las respectivas consideraciones y configuraciones para la extracción de datos de throughput y packt lost. Se define el algoritmo de optimización para la ubicación de los sensores en el espacio físico. Ver Figura 2.1.

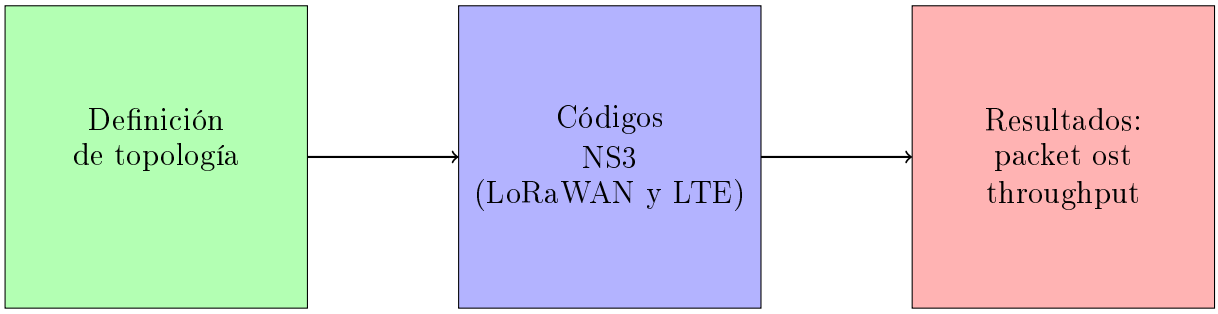

Figura 2.1: Flujo de Desarrollo de Simulación.

2.1. Topología

El escenario de simulación IoT esta conformada por un solo gateway a una altura de 15 metros y un número variable de sensores (a 0 metros de altura) que puede ir desde 4 hasta 100 según la relación $n = s^2$ donde s puede tomar los valores $(2, 3, 4, 5, 6, 7, 8, 9, 10)$. El comportamiento del sensor está configurado para enviar paquetes de 50 bytes cada 19 segundos y el resultado es analizado como tráfico agregado en el gateway mediante throughput y packet lost durante un tiempo de simulación de 40 segundos. ver Figura 2.2.

Se plantea dos figuras de ubicación de sensores en el área de cobertura: una ubicación en grilla cuadrada y una ubicación circular óptima, el área de ubicación es la misma para los dos casos, la ubicación del gateway es en el centro del área que forman los sensores. Otra variable que se considera es el radio del área circular óptima la cual puede variar de 100 a 5000 metros, este área circular es el mismo que el área de la grilla cuadrada. ver Figura 2.3.

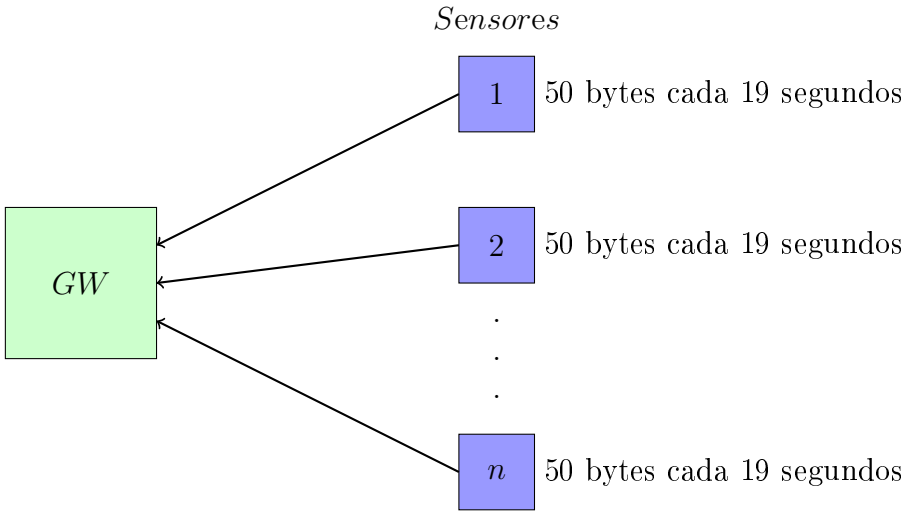

Figura 2.2: Topología IoT a Simular.

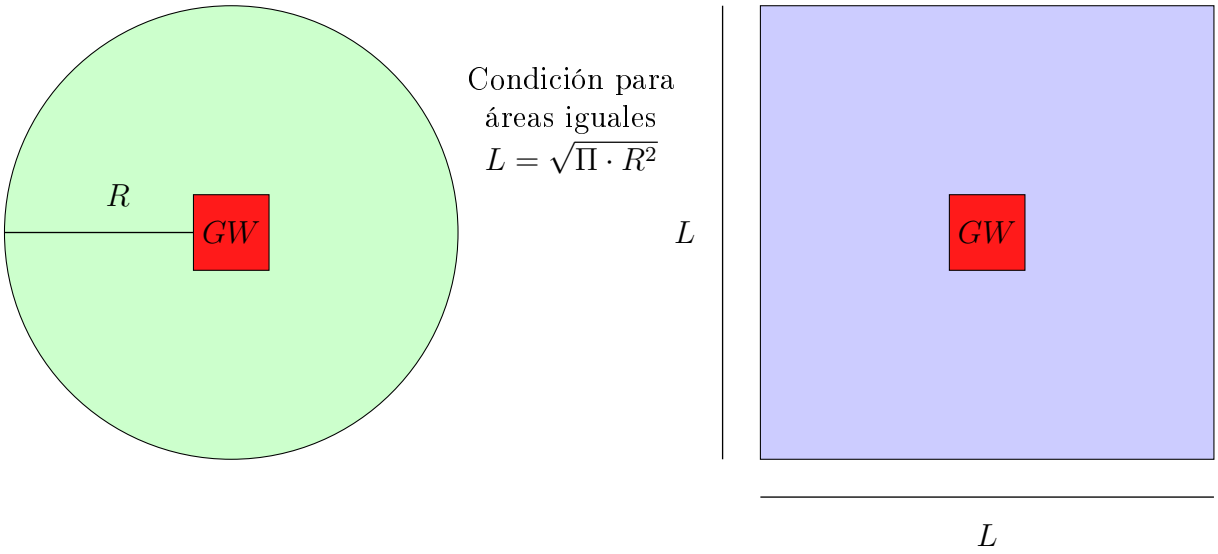

Figura 2.3: Topología IoT a Simular.

2.2. LoRaWAN

El código de simulación está basado en el módulo LoRaWAN definido en [11]. El archivo de referencia usado es complete-lorawan-network-example.cc, disponible como ejemplo en el módulo LoRaWAN NS3.

El código está escrito en lenguaje de programación C++ y su estructura está definida básicamente por: instrucciones donde se asocian las librerías y clases LoRaWAN utilizadas, declaraciones de variables y la función principal con todas las instrucciones del programa. La Figura 2.4 muestra gráficamente como se estructura la función principal.

El número de nodos es una variable definida al inicio del código, un nodo funciona como un elemento agnóstico sobre el que se le instala una personalidad, para este caso la personalidad

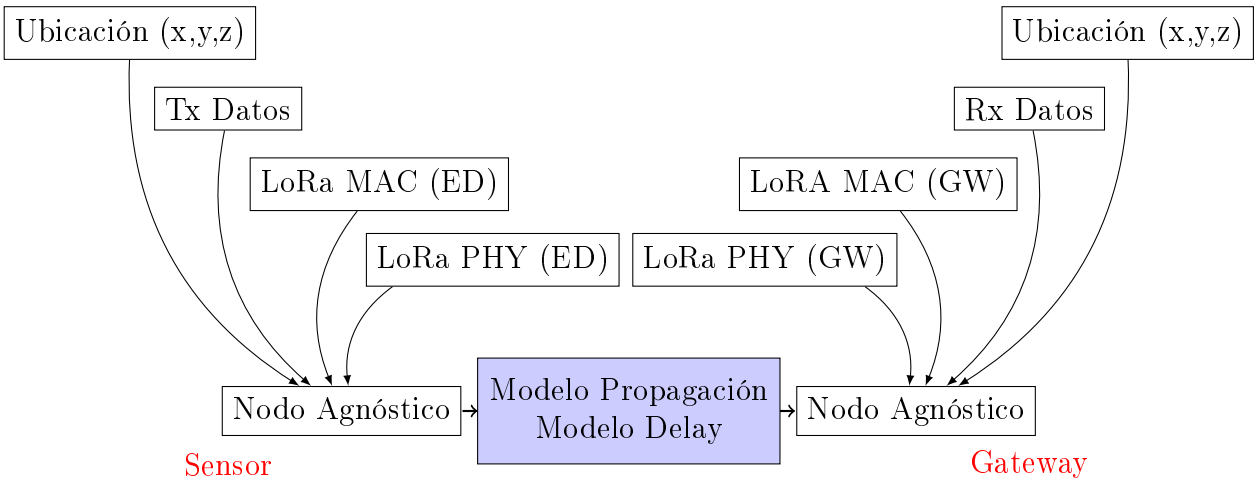

Figura 2.4: Estructura Código LoRaWAN.

de un sensor (ED) y la de un gateway (GW). Sobre cada nodo se instala el stack de protocolos (LoRa PHY y LoRa MAC), además de habilitar el envío de datos para el caso de los sensores y la recepción para el gateway, el envió de datos usa una clase que envía 50 bytes cada 19 segundos.

La ubicación de los sensores se define con vectores de ubicación (x,y,z) para cada nodo, esta ubicación define dos códigos distintos: en grilla cuadrada y en círculo óptimo, para la grilla cuadrada se definen dos bucles *for* para definir filas y columnas equidistantes, mientras que para el círculo óptimo los vectores de ubicación se importan de Matlab, la ubicación del gateway está en el centro y a una altura de 15 metros. Se define en el código también un modelo de propagación LogDistance mismo que simula un ambiente urbano, y un modelo de delay, con estas consideraciones se simula el canal de transmisión. Al final se define los datos de simulación obtenidos de la capa MAC, estos son: packet lost y throughput.

2.3. LTE

El código de simulación está basado en el proyecto LENA definido en [4]. El archivo de referencia usado es lena-simple-epc.cc, disponible como ejemplo. El código está escrito en lenguaje de programación C++ y su estructura está definida básicamente por: instrucciones donde se asocian las librerías y clases LENA utilizadas, declaraciones de variables y la función principal con todas las instrucciones del programa. La Figura 2.5 muestras gráficamente como se estructura la función principal.

El número de nodos es una variable definida al inicio del código, un nodo funciona como un elemento agnóstico sobre el que se le instala un personalidad, para este caso la personalidad de un sensor (ED), la de un gateway (GW) y la de un servidor remoto IP, sobre el nodo sensor se instala el stack de protocolos LTE e IP, sobre el nodo gateway se instala el stack de protocolos LTE y sobre el nodo servidor remoto se instala el stack de protocolos IP, además se habilita sockets UDP sobre el sensor y el servidor remoto para el envío de datos, este usa

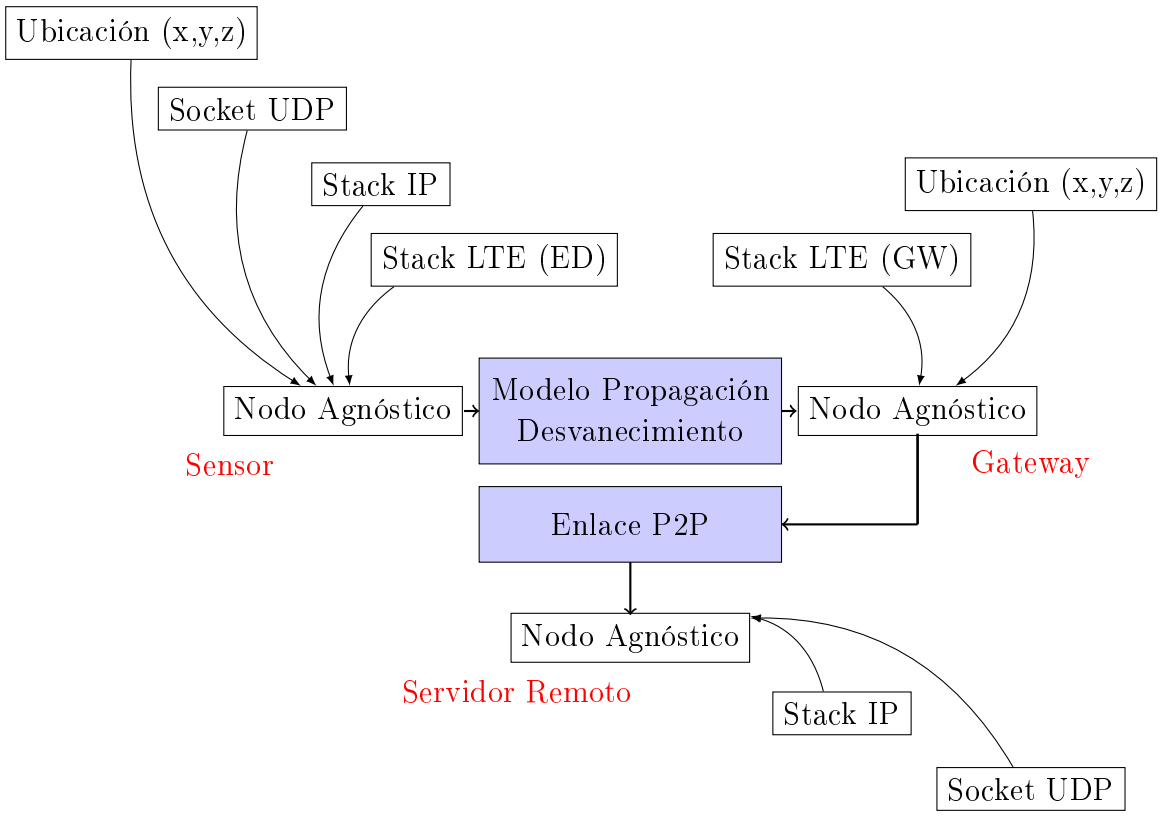

Figura 2.5: Estructura Código LTE.

un bucle que envía 50 bytes cada 19 segundos. La ubicación de los sensores se lo define con vectores de ubicación (x,y,z) para cada nodo, esta ubicación define dos códigos distintos: en grilla cuadrada y en círculo óptimo, para la grilla cuadrada se definen dos bucles *for* para definir filas y columnas equidistantes, mientras que para el círculo óptimo los vectores de ubicación se importan de Matlab, la ubicación del gateway está en el centro y a una altura de 15 metros.

Se define en el código también un modelo de propagación LogDistance mismo que simula un ambiente urbano, y un modelo de desvanecimiento, con estas consideraciones se simula el canal inalámbrico de transmisión entre los sensores y el gateway, para el canal de transmisión entre el gateway y el servidor remoto se simula un enlace punto a punto de 1 Gbps. Como los sensores y el servidor remoto son IP, estos tiene asignado un enrutamiento dentro de la subred 7.0.0.0/8. Al final se define los datos de simulación obtenidos de la capa PDCP, estos son: packet lost y throughput.

2.4. Algoritmo de Optimización

El algoritmo es desarrollado en Maltab, tiene como objetivo una distribución óptima de puntos en un área normalizada circular de radio 1. El número de puntos es variable y está relacionada a las ubicaciones en el plano (x,y) que tendrán los sensores en la simulación, la Figura 2.6 muestra el resultado del algoritmo para 16 puntos.

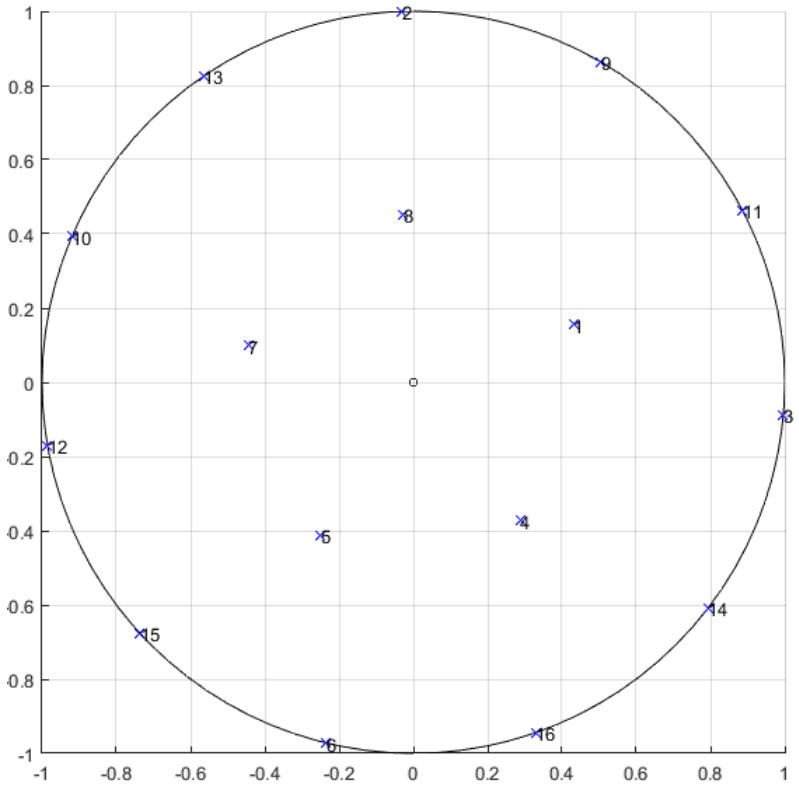

Figura 2.6: Ubicación Óptima.

Los puntos (x,y) tienen una ubicación aleatoria dentro del área circular, estos van moviéndose hasta optimizar la distancia acumulada entre todos los puntos, la distancia entre puntos esta definida por la Ecuación 2.1

$$DISTANCIA = \frac{1}{(X^2 + Y^2)^{-}p} \tag{2.1}$$

La distancia acumulada es el resultado de sumar todos los X y Y donde: X es la distancia entre dos puntos en el eje de las abscisas y Y es la distancia entre dos puntos en el eje las ordenadas. Para el caso particular el valor de $p = 5$, con esto la Ecuación 2.1 tiene el siguiente comportamiento:

Si $X^2 + Y^2$ es grande el valor de $DISTANCIA$ definida en el Ecuación 2.1 tiene un peso bajo, mientras que, si $X^2 + Y^2$ es pequeño el peso de $DISTANCIA$ es grande. Con esta consideración lo que se busca es minimizar el valor de la distancia acumulada, el valor mas pequeño define la mayor distancia posible entre puntos dentro del área circular, mediante un bucle esta optimización tiene un tiempo de aprendizaje.

2.5. Configuración y Resultados

Aquí se definen las condiciones técnicas sobre las cuales los escenarios LoRaWAN y LTE van a operar la simulación, por un lado se tiene las condiciones particulares (para cada tecnología) y condiciones generales (para ambas tecnologías), estos parámetros se describen en la Tabla 2.1.

Generales		**Particulares**			
Potencia Sensor	20 dBm	**LoRaWAN**		**LTE**	
Potencia GW	20 dBm	WB	250 kHz	WB	1.4 MHz
No. GW	1	Frec. UL	915 MHz	No. RB	6
No. Sensor	4-100			Frec. UL	781 MHz
T. Simulación	40 s			Scheduler	Round Robin
Tamaño Paquete	50 B				
Periodo Envío	19 s				
Mod. Propagación	LogDistance				
Área Cobertura	0,03-78,54 Km^2				
Altura Antena	15 m				
Altura Sensor	0 m				
Mod. Antena	Isotrópica				
Modo de Tranmsión	SISO				

Tabla 2.1: Condiciones de Configuración.

Los datos obtenidos de las simulaciones, como se sabe, son: throughput y packet lost, para la obtención de estos parámetros lo que se necesita contabilizar es el número de paquetes correctamente recibidos en el gateway, para el caso de LoRaWAN la simulación muestra en consola los paquetes correctamente recibidos y para el caso de LTE, LENA genera un archivo $UlPdcpStats.txt$ donde se muestran los paquetes correctamente recibidos. Como el tiempo de simulación es de 40 s y el periodo de envío de datos es de 19 s, cada sensor puede enviar dos veces, de esto se tiene que el número de paquetes enviados es igual a $ENV = 2 \cdot No.Sensores$, con en número de paquetes enviados y recibidos se tiene el valor de packet lost en porcentaje mediante la Ecuación 2.2.

$$packetlost = \frac{Pack.Enviados - Pack.Recibidos}{Pack.Enviados} \cdot 100[\,\%] \tag{2.2}$$

El throughput por otro lado se calcula mediante la Ecuación 2.3.

$$throughput = \frac{Pack.Recibidos \cdot 8 \cdot 50}{40}\,[bps] \tag{2.3}$$

El número de simulaciones depende de las combinaciones posibles de las variables: número de sensores y área de cobertura.

Capítulo 3

Resultados y Análisis de Datos

Este capítulo muestra el análisis de los datos obtenidos en las simulaciones. Se gráfican los datos para mirar su comportamiento y también se usa herramientas estadísticas para poder aplicar test de hipótesis y poder comparar dos muestras.

3.1. Gráficas de Datos

Esta primera parte pretende dar una idea de como los datos se ven, y mediante el análisis gráfico ver que nos pueden sugerir los datos. Se considera que el número de sensores son: 4, 9, 16, 25, 36, 49, 64, 81 y 100 y áreas definidas por la distancia del radio de una circunferencia que va de 100 a 5000 metros cada 500 metros, esto define 11 distintas áreas de cobertura, haciendo una combinación de estas variables se tiene 99 simulaciones. LTE tiene menor área de cobertura que LoRaWAN y para este último el área es mayor para la ubicación óptima de los sensores, ver Figura 3.1.

Se observa que LoRaWAN tiene su máximo throughput para una mayor área de cobertura comparado con LTE. Para LoRaWAN la ubicación óptima de sensores tiene mejor desempeño que la grilla, y para LTE es indiferente, para LTE la ubicación óptima tiene menos pérdidas que la grilla, y para LoRaWAN es indiferente, ver Figuras 3.2 y 3.3.

Existe un punto de cambio de tecnología (de LTE a LoRaWAN) recomendado si se quiere extender el área de cobertura, este es menor a 1000 metros, el área donde se define este punto es ligeramente mayor cuando mayor es el número de sensores en servicio. Ver Figura 3.4. Dentro del área de cobertura de las dos tecnologías se tiene que la influencia debido al número de sensores empieza a notarse para sensores cuyo número supera los 42 aproximadamente, ver Figura 3.5. La pérdida de paquetes hasta el 20 %, suponiendo este valor como un umbral de tolerancia, tiene una cobertura mayor para LoRaWAN si se compara tecnologías y para la distribución óptima de sensores si se compara ubicación, ver Figura 3.6 y 3.7.

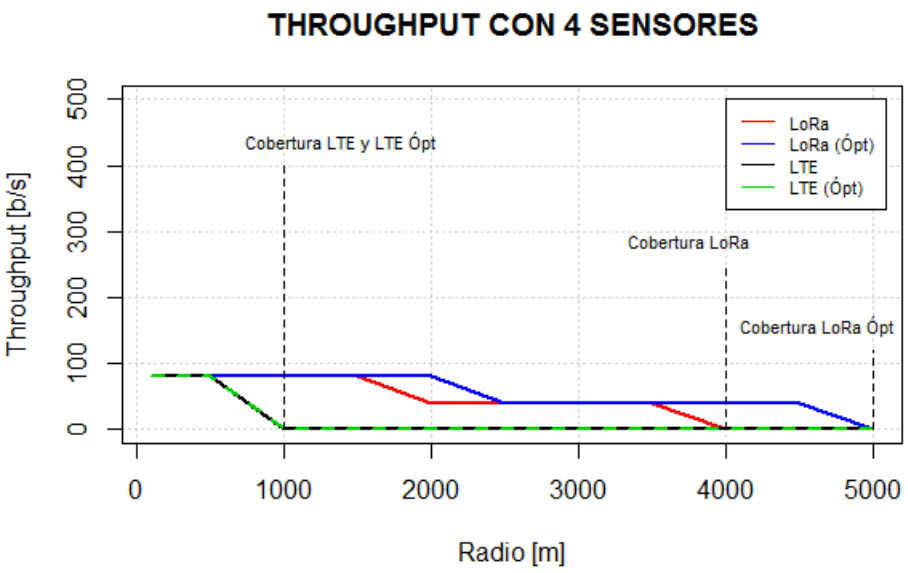

Figura 3.1: Área de Cobertura.

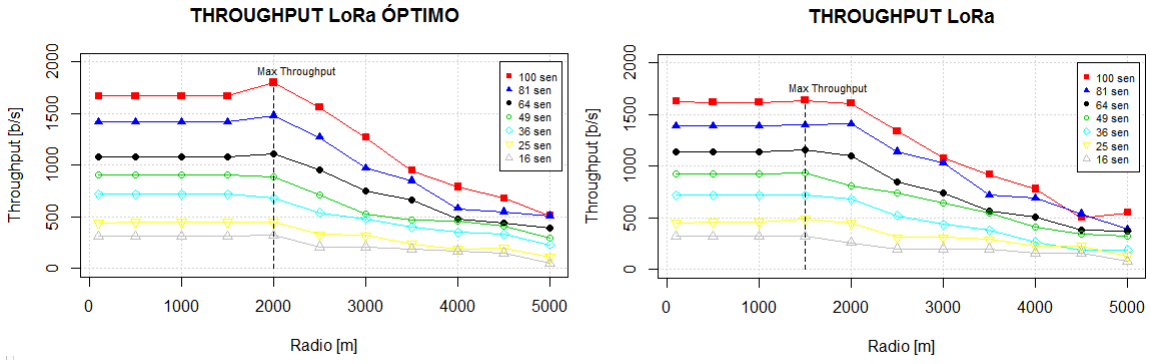

Figura 3.2: Throughput LoRaWAN.

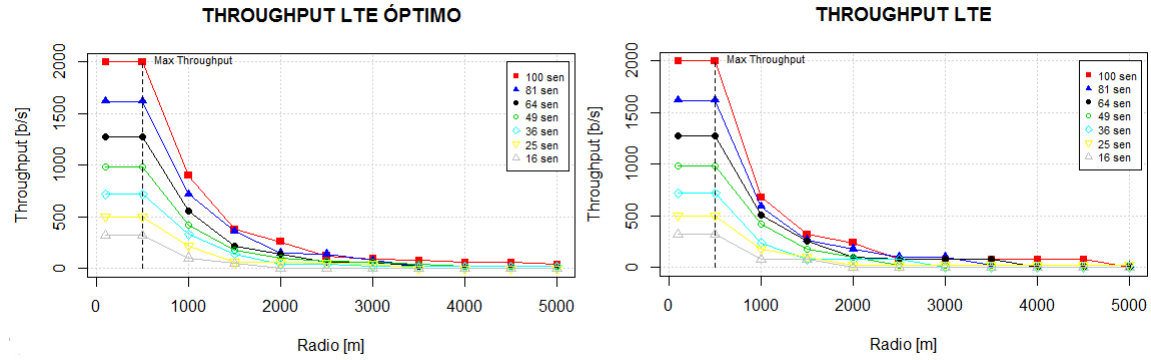

Figura 3.3: Throughput LTE.

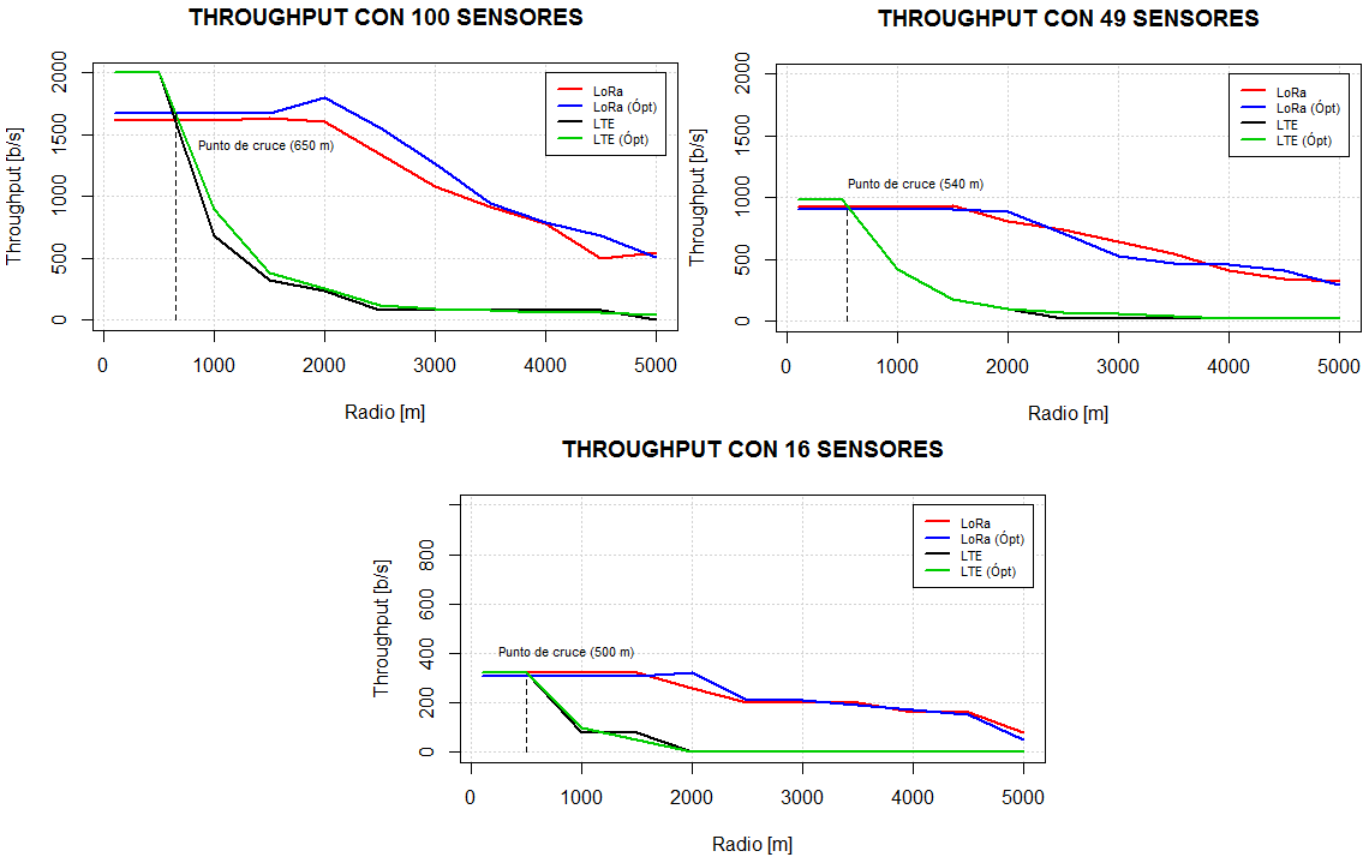

Figura 3.4: Punto de Cambio de Tecnología.

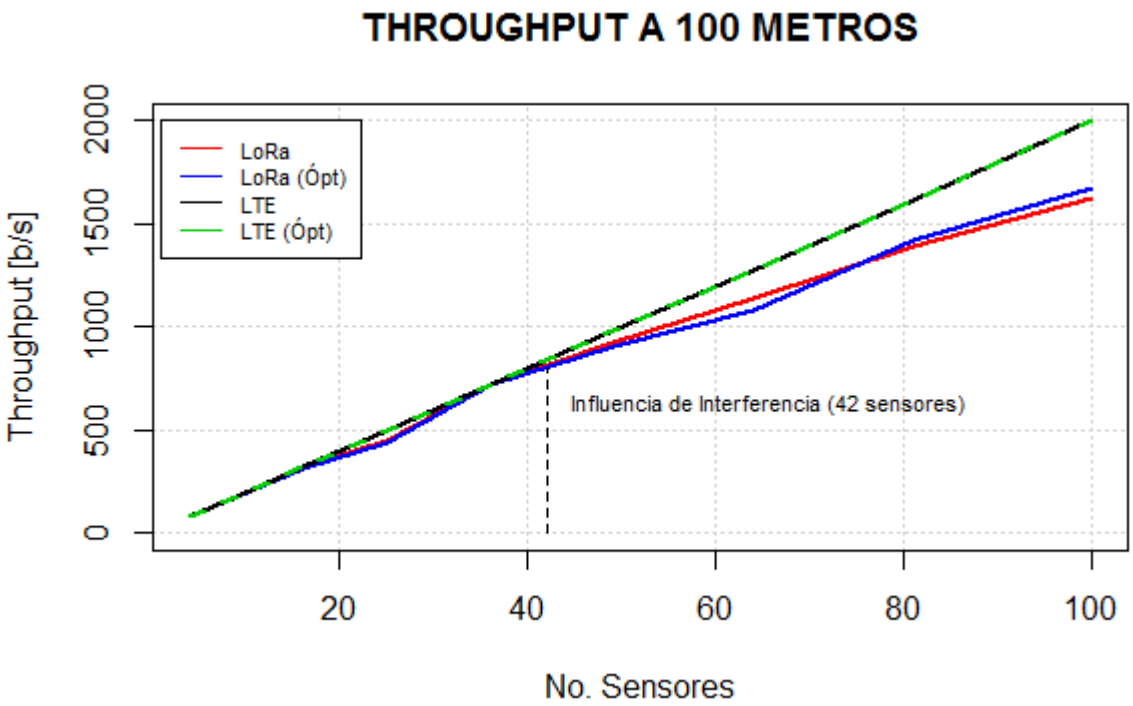

Figura 3.5: Punto de Cambio de Tecnología.

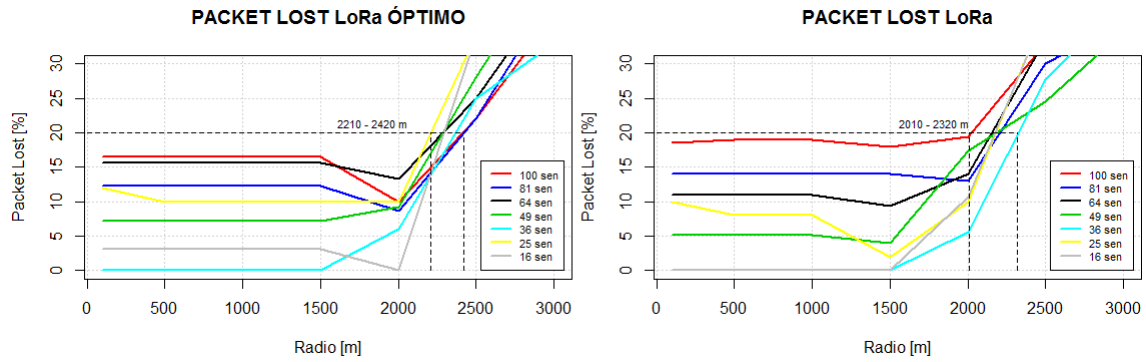

Figura 3.6: Packet Lost LoRaWAN.

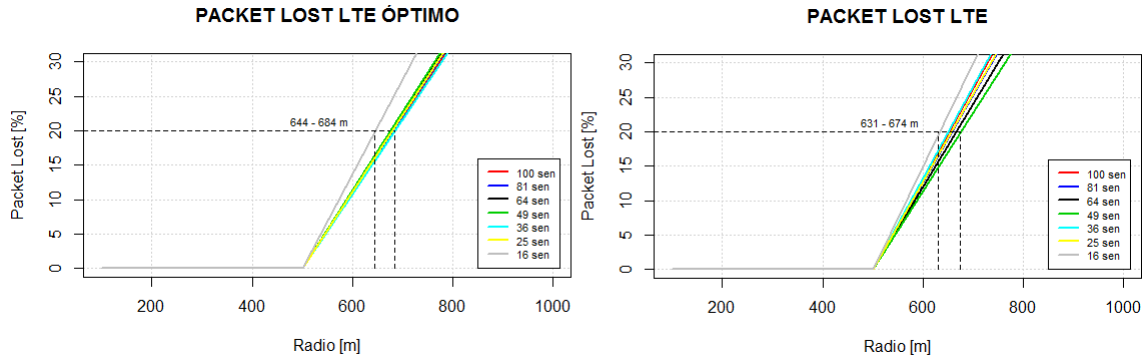

Figura 3.7: Packet Lost LTE.

3.2. Análisis Estadístico de Datos

Para esta segunda parte se escoge un escenario en particular y se hace un análisis estadístico de los datos para definir modelos que nos sirvan para aplicar un test de hipótesis. Se considera el escenario definido con 100 sensores y áreas definidas por la distancia del radio de una circunferencia que va de 100 a 5000 metros cada 100 metros.

Para LoRaWAN cada simulación se repite 20 veces, lo que significa 2000 simulaciones; y para LTE solo una simulación (porque los resultados no cambian en distintas simulaciones), se toma un valor de throughput y packet lost en cada distancia lo que significa 100 simulaciones. Para el caso de LoRaWAN se toma los valores de throughput y packet lost en cada distancia, de las 20 simulaciones se busca un valor estadístico representativo, se puede obtener la media como una aproximación del valor si los datos tiene una distribución normal.

La cantidad de simulaciones (20) es muy poco para aplicar un test de normalidad por lo que primero se hace un sampleo de los mismos datos con reemplazo hasta completar 30 muestras, a estas se aplica un test de normalidad (Shapiro–Wilk), para las muestras que no superan el test de normalidad se aplica un test de Outlies (Grubbs), si existen Outlier estos se elimina, se samplea nuevamente y se aplica el test de normalidad nuevamente. Se asume la media como valor único de las 20 simulaciones, esto se hace para todas las distancias.

Los datos de media, desviación estándar y p-value de los test de normalidad se exponen

en las Tablas 3.1, 3.2, 3.3, 3.4, para optimizar la distribución del documento solo se muestran algunos datos. La Tabla 3.5 muestra la estadística descriptiva de los datos obtenidos.

Test de Normalidad (Throughput Óptimo)							
Num	Media	SD	P-value	Num	Media	SD	P-value
1	1624,33	57,58	0,105	26	1364,00	58,23	0,437
2	1653,00	48,51	0,112	27	1237,00	42,68	0,091
3	1643,00	73,07	0,292	28	1242,67	33,62	0,109
4	1629,33	93,77	0,114	29	1251,33	37,21	0,063
5	1647,00	60,58	0,075	30	1217,33	50,51	0,053
6	1655,67	57,28	0,196	31	1227,33	60,51	0,092
7	1683,67	50,41	0,130	32	1094,67	49,60	0,097
8	1652,00	52,22	0,462	33	1033,33	47,59	0,074
9	1637,00	91,54	0,065	34	989,67	67,70	0,220

Tabla 3.1: Media, Desviación Estándar y P-value, Óptimo.

Test de Normalidad (Throughput Grilla)							
Num	Media	SD	P-value	Num	Media	SD	P-value
1	1633,33	65,99	0,077	26	1336,67	44,13	0,218
2	1659,33	62,91	0,100	27	1320,67	46,08	0,068
3	1677,33	78,34	0,060	28	1312,67	65,60	0,156
4	1645,67	43,13	0,056	29	1217,00	47,28	0,859
5	1651,33	73,75	0,380	30	1204,67	63,18	0,068
6	1658,33	86,94	0,052	31	1054,33	40,57	0,372
7	1650,67	43,15	0,033	32	1031,33	77,05	0,130
8	1637,00	91,36	0,059	33	1037,33	63,30	0,337
9	1644,33	58,59	0,055	34	1014,33	53,09	0,145

Tabla 3.2: Media, Desviación Estándar y P-value, Grilla.

Test de Normalidad (Packet Lost Óptimo)							
Num	Media	SD	P-value	Num	Media	SD	P-value
1	17,92	2,87	0,195	26	32,03	2,77	0,377
2	16,78	2,68	0,308	27	38,00	2,06	0,094
3	18,43	3,81	0,272	28	37,90	1,95	0,066
4	18,42	4,23	0,203	29	37,88	1,75	0,139
5	16,22	2,05	0,082	30	39,40	2,30	0,277
6	17,12	2,84	0,223	31	39,47	2,96	0,061
7	16,15	2,29	0,208	32	45,32	2,73	0,148
8	16,65	2,51	0,428	33	48,33	2,44	0,077
9	16,80	3,92	0,092	34	50,53	3,50	0,173

Tabla 3.3: Media, Desviación Estándar y P-value, Óptimo.

Test de Normalidad (Packet Lost Grilla)							
Num	Media	SD	P-value	Num	Media	SD	P-value
1	18,33	2,81	0,095	26	32,75	2,06	0,571
2	16,73	2,98	0,089	27	33,90	2,19	0,071
3	16,65	3,57	0,073	28	34,10	2,87	0,440
4	17,70	2,11	0,100	29	39,43	2,71	0,570
5	16,37	3,66	0,290	30	40,47	2,93	0,140
6	17,20	4,28	0,120	31	47,40	2,25	0,603
7	18,55	2,30	0,066	32	47,62	3,54	0,150
8	18,67	4,80	0,075	33	48,15	3,29	0,214
9	18,42	2,92	0,056	34	48,48	2,64	0,356

Tabla 3.4: Media, Desviación Estándar y P-value, Grilla.

Parámetro	Media	SD	Mediana	Q1	Q3	No. Muestras
Throughput LoRa	1249,0	434,4	1367,5	820,8	1650,5	50
Throughput LoRa Óptimo	1260,4	427,9	1430,7	832,6	1641,5	50
Throughput LTE	460,8	657,7	80,0	80,0	452,5	50
Throughput LTE Óptimo	469,6	658,5	110,0	80,0	520,0	50
Packet Lost LoRa	37,6	21,7	31,6	17,2	58,7	50
Packet Lost LoRa Óptimo	36,9	21,5	28,7	17,3	57,9	50
Packet Lost LTE	77,0	32,9	96,0	77,4	96,0	50
Packet Lost LTE Óptimo	76,5	32,9	94,5	74,0	96,0	50

Tabla 3.5: Estadística Descriptiva.

3.2.1. Modelos Throughput y Packet Lost

En este punto se definen modelos de regresión para definir throughput y packet lost, para obtener nuevos datos y hacer un análisis comparativo no paramétrico. Los modelos usados en el orden expuesto son:

- Regresión lineal.
- Regresión no lineal de grado 4.
- Regresión no lineal de grado 3.
- Regresión no lineal dada una función conocida.
- Regresión Spline.
- Regresión GAM.

Se toman el mejor modelo considerando el error cuadrático medio (RMSE), normalidad de residuos (NOR RES), media de residuos (MED RES) y el criterio de información de Akaike (AIC). las Tablas 3.6 y 3.7 muestran estos datos.

Una vez definidos los mejores modelos, se generan nuevos datos, después de aplica un test de normalidad (Shapiro–Wilk) se observa que los datos no son normales y se aplica un test de medianas no paramétrico, la comparación se hace entre:

Throughput							
LoRaWAN							
Grilla				**Óptimo**			
RMSE	NOR RES	MED RES	AIC	RMSE	NOR RES	MED RES	AIC
140,93	0,28	-7,26	520,26	151,74	0,04	-10,97	524,76
43,85	0,65	-2,07	431,92	61,41	0,02	-5,65	459,84
54,37	0,21	-4,35	446,03	68,89	0,01	-7,67	464,99
57,34	0,14	-1,36	452,96	67,66	0,01	-4,97	466,43
38,47	0,84	-1,01	426,89	50,05	0,36	-4,17	446,89
27,98	0,16	-1,76	400,50	38,33	0,41	-2,48	427,66
LTE							
RMSE	NOR RES	MED RES	AIC	RMSE	NOR RES	MED RES	AIC
410,78	3,02 e-3	17,61	603,41	399,11	2,71 e-3	11,40	602,13
138,42	1,70 e-4	4,99	521,82	132,80	3,49 e-3	-2,20	522,23
139,48	9,34 e-6	6,12	520,33	133,22	1,46 e-5	-0,50	521,37
365,33	6,44 e-9	-58,67	592,55	363,52	4,04 e-9	-66,71	594,69
87,26	1,63 e-2	0,21	491,70	82,14	8,97 e-4	-7,78	478,82
52,34	2,30 e-4	2,95	453,66	53,26	8,86 e-7	-6,00	424,80

Tabla 3.6: Modelos Throughput.

Packet Lost							
LoRaWAN							
Grilla				**Óptimo**			
RMSE	NOR RES	MED RES	AIC	RMSE	NOR RES	MED RES	AIC
7,07	0,28	0,43	280,61	7,44	0,05	0,53	283,59
2,22	0,68	0,16	191,65	2,99	0,01	0,28	218,59
2,77	0,22	0,27	205,66	3,33	0,01	0,37	223,34
2,90	0,08	0,12	213,51	3,31	0,00	0,24	224,97
1,94	0,65	0,11	186,86	2,45	0,34	0,20	205,74
1,38	0,17	0,15	158,47	1,85	0,45	0,12	185,21
LTE							
RMSE	NOR RES	MED RES	AIC	RMSE	NOR RES	MED RES	AIC
20,54	3,02 e-3	-0,88	363,75	19,96	2,71 e-3	-0,57	362,47
6,92	1,70 e-4	-0,25	282,16	6,64	3,49 e-3	0,11	282,58
6,97	9,34 e-6	-0,31	280,67	6,66	1,46 e-5	0,03	281,72
16,28	1,24 e-2	-1,91	344,67	15,72	8,95 e-3	-1,63	343,82
4,36	1,63 e-2	-0,01	252,04	4,11	8,97 e-4	0,39	239,16
2,62	2,30 e-4	-0,15	214,01	2,66	8,86 e-7	0,30	185,14

Tabla 3.7: Modelos Packet Lost.

- LoRaWAN óptimo vs LoRaWAN grilla.
- LTE óptimo vs LTE grilla.
- LTE óptimo vs LoRaWAN óptimo.
- LTE grilla vs LoRaWAN grilla.

Los resultados de esto se muestra en la Tabla 3.8 y en las Figuras 3.8, 3.9, 3.10 y 3.11.

Throughput							
Par A	Par B	NOR A	NOR B	P-V	ME A	ME B	RESUL
LoRa Op	LoRa	8,51 e-6	7,50 e-6	0,94	1408,54	1390,62	IGUALES
LTE Op	LTE	1,19 e-9	7,64 e-10	0,93	105,84	98,38	IGUALES
LTE Op	LoRa Op	1,19 e-9	8,51 e-6	1	105,84	1408,54	LORA >
LTE	LoRa	7,64 e-10	7,50 e-6	1	98,38	1390,62	LORA >
Packet Lost							
Par A	Par B	NOR A	NOR B	P-V	ME A	ME B	RESUL
LoRa Op	LoRa	8,26 e-3	8,17 e-3	0,84	29,86	30,50	IGUALES
LTE Op	LTE	1,19 e-9	7,63 e-10	0,93	94,71	95,01	IGUALES
LTE Op	LoRa Op	1,19 e-9	8,26 e-3	4,96 e-9	94,71	29,86	LORA >
LTE	LoRa	7,63 e-10	8,17 e-3	1,70 e-9	95,08	30,50	LORA >

Tabla 3.8: Comparación Parámetros.

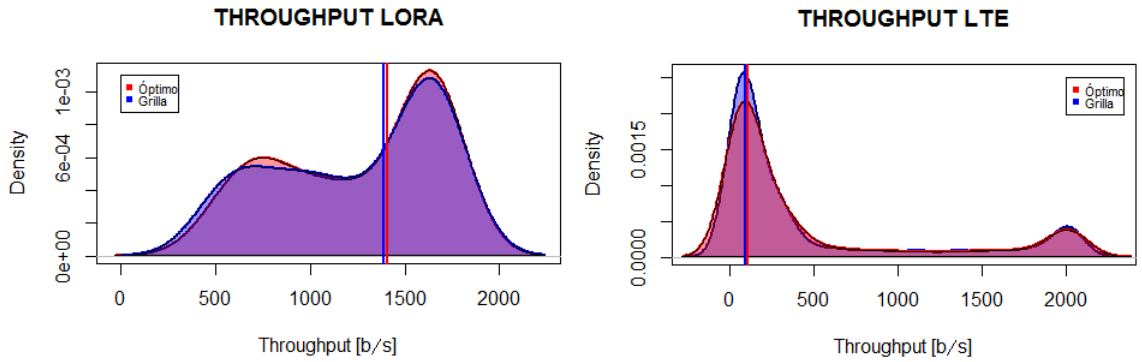

Figura 3.8: Comparación 1.

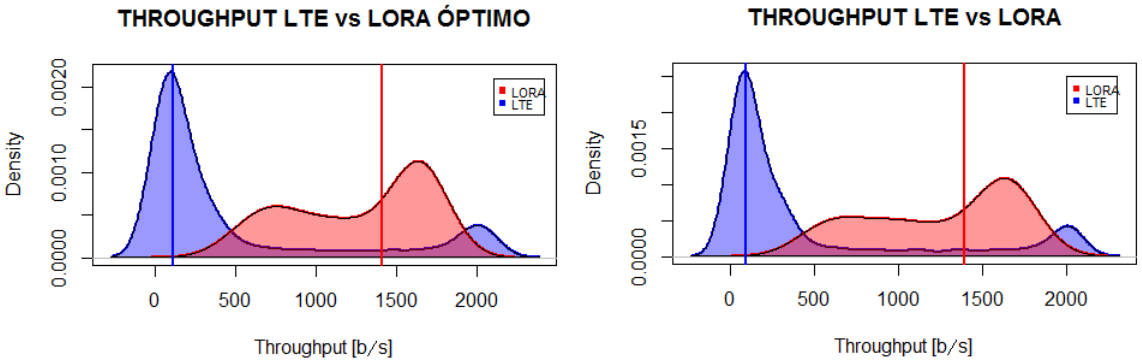

Figura 3.9: Comparación 2.

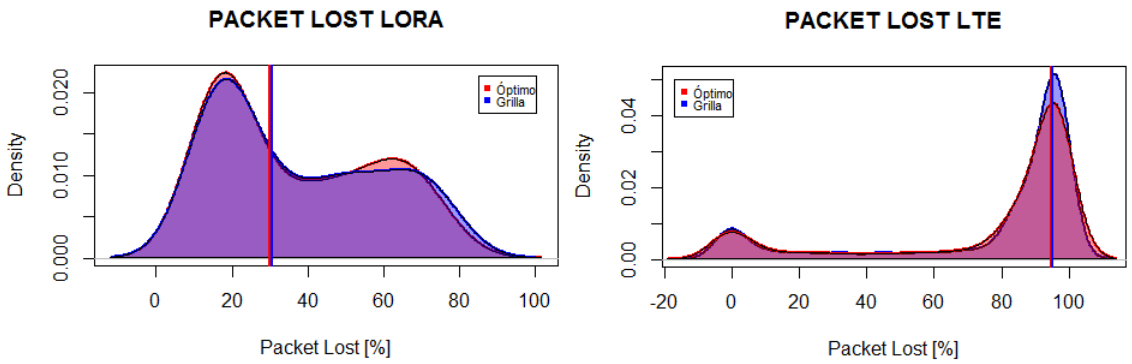

Figura 3.10: Comparación 3.

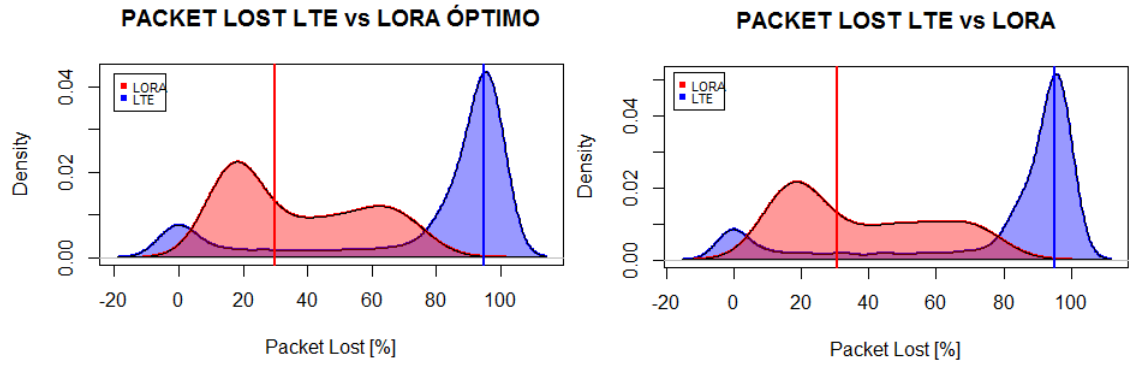

Figura 3.11: Comparación 4.

Conclusión

Haciendo un análisis de las gráficas se puede ver que:

- El comportamiento de LoRaWAN muestra menor throughput a distancias cortas y mayor throughput a distancias largas comparado con LTE.
- La influencia negativa de la cantidad de sensores en el desempeño es mayor en LoRaWAN que en LTE.
- El umbral de tolerancia de pérdida de paquetes (suponiendo 20 %) para LoRaWAN esta definido a una distancia mayor que para LTE.
- LTE no presenta pérdida de paquetes a distancias cortas comparado con LoRaWAN.
- La diferencia entre la distribución óptima de los sensores y la distribución en grilla cuadrada existe pero es mínima a favor de la ubicación óptima.

Por otro lado las conclusiones obtenidas del test de hipótesis sobre los modelos muestran que:

- El desempeño de la ubicación óptima respecto a la ubicación en grilla para LoRaWAN es igual. Diferencias de 17,92 bps en throughput y 0,64 % en packet lost.
- El desempeño de la ubicación óptima respecto a la ubicación en grilla para LTE es igual. Diferencias de 7,46 bps en throughput y 0,3 % en packet lost.
- El desempeño de LoRaWAN es mayor que el desempeño de LTE en función de la distancia. 1302,7 bps mayor en throughput y 64,85 % menor en packet lost para la ubicación óptima y 1292,24 bps mayor en throughput y 64,58 % menor en packet lost para la ubicación en grilla cuadrada.

Si bien se hace un análisis comparativo de un escenario específico, el uso de una u otra tecnología va a depender de la aplicación, por lo que se puede decir que:

- Si la aplicación requiere baja cobertura y el packet lost es crítico entonces conviene usar LTE, mientras que LoRaWAN trabaja mucho mejor en aplicaciones que demandan grandes coberturas y tolerancia al packet lost, este es el caso de monitoreo de cultivos.
- Si se requiere aplicaciones con calidad de servicio (QoS) la mejor opción es LTE ya que la red maneja este concepto mientras que LoRaWAN no maneja QoS solo tiene distinción en clases de terminales y modo de configuración.

Bibliografía

[1] Capacity study for LTE release 13 eMTC with coverage enhanched RACH, Khan, S.H., 2016. https://pure.tue.nl/ws/files/46944860/855330-1.pdf. Accessed: 25 de julio de 2018.

[2] Kristoffer Olsson y Sveinn Finnsson, Exploring LoRa and LoRaWAN, Sweden 2017. http://publications.lib.chalmers.se/records/fulltext/252610/252610.pdf. Accessed: 19 de julio de 2018.

[3] Las redes más usadas en el IoT, Bruno Cendón. http://www.bcendon.com/las-redes-mas-usadas-en-el-iot/. Accessed: 19 de julio de 2018.

[4] LENA Overview. http://networks.cttc.es/mobile-networks/software-tools/lena/. Accessed: 3 de agosto de 2018.

[5] LoRa Scalability: A Simulation Model Based on Interference Measurements, Jetmir Haxhibeqiri *, Floris Van den Abeele, Ingrid Moerman and Jeroen Hoebeke, 27 March 2017. http://www.mdpi.com/1424-8220/17/6/1193/htm. Accessed: 21 de julio de 2018.

[6] LoRaWAN™ 1.0.3 Specification, 2018 LoRa Alliance, March 20, 2018, Final release. https://www.lora-alliance.org/sites/default/files/2018-06/lorawan1.0.3_final_0.pdf. Accessed: 21 de julio de 2018.

[7] LoRaWAN™ 1.1 Regional Parameters, 2017 LoRa Alliance, October 11, 2017, Final release. https://www.lora-alliance.org/sites/default/files/2018-05/lorawan-regional-parameters-v1.1ra.pdf. Accessed: 21 de julio de 2018.

[8] LTE Network Architecture: Basic, By Netmanias (tech@netmanias.com). https://www.netmanias.com/en/post/techdocs/5904/lte-network-architecture/lte-network-architecture-basic. Accessed: 26 de julio de 2018.

[9] Massive Access for Machine-to-Machine Communication in Cellular Networks, Massive Machine-to-Machine Communication in Long Term Evolution and Long Range Wide Area Network, Aalborg University. https://projekter.aau.dk/projekter/files/239500086/master.pdf. Accessed: 1 de agosto de 2018.

[10] MATLAB ® The Language of Technical Computing, Getting Started with MATLAB ®, Version 7. https://www.mn.uio.no/astro/english/services/it/

help/mathematics/matlab/getstart.pdf. Accessed: 3 de agosto de 2018.

[11] Network Level Performances of a LoRa System, Magrin Davide, Università degli Studi di Padova. http://tesi.cab.unipd.it/53740/1/dissertation.pdf. Accessed: 21 de julio de 2018.

[12] Nonlinear Regression Essentials in R: Polynomial and Spline Regression Models. http://www.sthda.com/english/articles/40-regression-analysis/162-nonlinear-regression-essentials-in-r-polynomial-and-spline-regression-models/#polynomial-regression. Accessed: 4 de septiembre de 2018.

[13] ns-3 Tutorial, Release ns-3.28, ns-3 Tutorial, March 21, 2018. https://www.nsnam.org/docs/release/3.28/tutorial/ns-3-tutorial.pdf. Accessed: 1 de agosto de 2018.

[14] Overview of LTE 3GPP releases, All about Wired and Wireless Technology. http://www.simpletechpost.com/2015/02/overview-of-lte-3gpp-releases.html. Accessed: 26 de julio de 2018.

[15] Prueba de Grubbs. http://www.bdigital.unal.edu.co/2033/1/71644758.20101.pdf. Accessed: 31 de agosto de 2018.

[16] Prueba de Shapiro-Wilks. http://riotorto.users.sourceforge.net/R/noparam_shapiro/. Accessed: 31 de agosto de 2018.

[17] SEMTECH, AN1200.22-LoRa™ Modulation Basics, Revision 2, May 2015. https://www.semtech.com/uploads/documents/an1200.22.pdf. Accessed: 19 de julio de 2018.

[18] SX1272/3/6/7/8: LoRa Modem, Designer's Guide AN1200.13, July 2013. https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf. Accessed: 3 de agosto de 2018.

[19] Técnicas de Acceso Múltiple. OFDMA y SC-FDMA . http://bibing.us.es/proyectos/abreproy/12081/fichero/OFDMA+y+SC-FDMA+en+la+Interfaz+Radio+de+LTE%252F4.+T%C3%A9cnicas+de+acceso+m%C3%BAltiple.+OFDMA+y+SC-FDMA.pdf. Accessed: 31 de julio de 2018.

[20] The LENA ns-3 LTE Module Documentation, Release v8, January 21, 2014. http://networks.cttc.es/wp-content/uploads/sites/2/2014/01/lena-lte-module-doc.pdf. Accessed: 3 de agosto de 2018.

[21] What is R? https://www.r-project.org/about.html. Accessed: 3 de agosto de 2018.

Anexos

Código NS3 LoRaWAN

```
// Basado en el desarrollo de Davide Magrin, Università degli Studi di Padova
// Red LoRa para aplicaciones IoT.

#include "ns3/end-device-lora-phy.h"
#include "ns3/gateway-lora-phy.h"
#include "ns3/end-device-lora-mac.h"
#include "ns3/gateway-lora-mac.h"
#include "ns3/simulator.h"
#include "ns3/log.h"
#include "ns3/pointer.h"
#include "ns3/constant-position-mobility-model.h"
#include "ns3/lora-helper.h"
#include "ns3/node-container.h"
#include "ns3/mobility-helper.h"
#include "ns3/position-allocator.h"
#include "ns3/double.h"
#include "ns3/random-variable-stream.h"
#include "ns3/periodic-sender-helper.h"
#include "ns3/command-line.h"
#include <algorithm>
#include <ctime>


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ComplexLorawanNetworkExample");

  double gri_x = 1;
  double gri_y = 1;
  double ues_gri_x = 10;
  double ues_gri_y = 10;
  uint16_t nGateways = gri_x * gri_y;
  uint16_t nDevices = ues_gri_x * ues_gri_y * nGateways;
  double distanceUEs = 4431.134627/(ues_gri_x-1);
  double altura_antena = 15.0;
  double simulationTime = 40;
  int appPeriodSeconds = 20;
  std::vector<int> sfQuantity (6);
```

```
double escala = 5000;
int noMoreReceivers = 0;
int interfered = 0;
int received = 0;
int underSensitivity = 0;
int cont = 0;

// Output control
bool printEDs = true;
bool buildingsEnabled = false;

enum PacketOutcome {
  RECEIVED,
  INTERFERED,
  NO_MORE_RECEIVERS,
  UNDER_SENSITIVITY,
  UNSET
};

struct PacketStatus {
  Ptr<Packet const> packet;
  uint32_t senderId;
  int outcomeNumber;
  std::vector<enum PacketOutcome> outcomes;
};

std::map<Ptr<Packet const>, PacketStatus> packetTracker;

void
CheckReceptionByAllGWsComplete (std::map<Ptr<Packet const>, PacketStatus>::
iterator it)
{

  if ((*it).second.outcomeNumber == nGateways)
    {

      PacketStatus status = (*it).second;
      for (int j = 0; j < nGateways; j++)
        {
          switch (status.outcomes.at (j))
            {
            case RECEIVED:
              {
                received += 1;
                break;
              }
            case NO_MORE_RECEIVERS:
              {
                noMoreReceivers += 1;
                break;
              }
            case INTERFERED:
              {
                interfered += 1;
                break;
              }
            case UNSET:
              {
                break;
```

```
            }
          }
        }

      packetTracker.erase (it);
    }
}

void
TransmissionCallback (Ptr<Packet const> packet, uint32_t systemId)
{
  NS_LOG_INFO ("Transmitted a packet from device " << systemId);
  PacketStatus status;
  status.packet = packet;
  status.senderId = systemId;
  status.outcomeNumber = 0;
  status.outcomes = std::vector<enum PacketOutcome> (nGateways, UNSET);
  packetTracker.insert (std::pair<Ptr<Packet const>, PacketStatus> (packet,
  status));
  cont=cont+1;
}

void
PacketReceptionCallback (Ptr<Packet const> packet, uint32_t systemId)
{
  NS_LOG_INFO ("A packet was successfully received at gateway " << systemId);

  std::map<Ptr<Packet const>, PacketStatus>::iterator it = packetTracker.find
  (packet);
  (*it).second.outcomes.at (systemId - nDevices) = RECEIVED;
  (*it).second.outcomeNumber += 1;

  CheckReceptionByAllGWsComplete (it);
}

void
InterferenceCallback (Ptr<Packet const> packet, uint32_t systemId)
{
  NS_LOG_INFO ("A packet was lost because of interference at gateway " <<
  systemId);

  std::map<Ptr<Packet const>, PacketStatus>::iterator it = packetTracker.find
  (packet);
  (*it).second.outcomes.at (systemId - nDevices) = INTERFERED;
  oldtime = std::time (0);
  Simulator::Schedule (Minutes (30), &PrintSimulationTime);
}

void
PrintEndDevices (NodeContainer endDevices, NodeContainer gateways, std::string
filename)
{
  const char * c = filename.c_str ();
  std::ofstream spreadingFactorFile;
  spreadingFactorFile.open (c);
  for (NodeContainer::Iterator j = endDevices.Begin (); j != endDevices.End ();
  ++j)
```

```
    {
      Ptr<Node> object = *j;
      Ptr<MobilityModel> position = object->GetObject<MobilityModel> ();
      NS_ASSERT (position != 0);
      Ptr<NetDevice> netDevice = object->GetDevice (0);
      Ptr<LoraNetDevice> loraNetDevice = netDevice->GetObject<LoraNetDevice> ();
      NS_ASSERT (loraNetDevice != 0);
      Ptr<EndDeviceLoraMac> mac = loraNetDevice->GetMac ()->
      GetObject<EndDeviceLoraMac> ();
      int sf = int(mac->GetDataRate ());
      Vector pos = position->GetPosition ();
      spreadingFactorFile << pos.x << " " << pos.y << " " << sf << std::endl;
    }

  for (NodeContainer::Iterator j = gateways.Begin (); j != gateways.End (); ++j)
    {
      Ptr<Node> object = *j;
      Ptr<MobilityModel> position = object->GetObject<MobilityModel> ();
      Vector pos = position->GetPosition ();
      spreadingFactorFile << pos.x << " " << pos.y << " GW" << std::endl;
    }
  spreadingFactorFile.close ();
}

int main (int argc, char *argv[])
{

  CommandLine cmd;
  cmd.AddValue ("nDevices", "Number of end devices to include in the simulation",
  nDevices);
  cmd.AddValue ("simulationTime", "The time for which to simulate", simulationTime);
  cmd.AddValue ("appPeriod", "The period in seconds to be used by periodically
  transmitting applications",
  appPeriodSeconds);
  cmd.AddValue ("printEDs", "Whether or not to print a file containing the ED's
  positions", printEDs);

  cmd.Parse (argc, argv);

  Time appPeriod = Seconds (appPeriodSeconds);

  //Configuracion de canal
  Ptr<LogDistancePropagationLossModel> loss =
  CreateObject<LogDistancePropagationLossModel> ();
  Ptr<PropagationDelayModel> delay =
  CreateObject<ConstantSpeedPropagationDelayModel> ();
  Ptr<LoraChannel> channel = CreateObject<LoraChannel> (loss, delay);

  /************************
  *  Create the helpers  *
  ************************/

  // Create the LoraPhyHelper
  LoraPhyHelper phyHelper = LoraPhyHelper ();
  phyHelper.SetChannel (channel);

  // Create the LoraMacHelper
  LoraMacHelper macHelper = LoraMacHelper ();
```

```
  // Create the LoraHelper
  LoraHelper helper = LoraHelper ();

  /************************
  *  Create End Devices  *
  ************************/

  // Create a set of nodes
  NodeContainer endDevices;
  endDevices.Create (nDevices);

  // Create the gateway nodes (allocate them uniformely on the disc)
  NodeContainer gateways;
  gateways.Create (nGateways);

//Crea objeto Posición que definirá una matriz de posición física de los nodos
Ptr<ListPositionAllocator> positionAllocUE = CreateObject<ListPositionAllocator> ();
Ptr<ListPositionAllocator> positionAllocENB = CreateObject<ListPositionAllocator> ();

//Define la ubicación de UEs y EnB, en el área circular

positionAllocUE->Add(Vector(escala*0.49854,escala*0.86687,0));
positionAllocUE->Add(Vector(escala*0.29997,escala*-0.95395,0));
positionAllocUE->Add(Vector(escala*-0.092776,escala*0.25636,0));
positionAllocUE->Add(Vector(escala*-0.93765,escala*-0.34757,0));
positionAllocUE->Add(Vector(escala*0.36459,escala*0.54813,0));
positionAllocUE->Add(Vector(escala*-0.28775,escala*-0.33712,0));
positionAllocENB->Add(Vector(0,0,altura_antena));

//Arma las grillas de posición de UEs para cada EnB y la grilla de posición
  //uint16_t c_x = 0;
  //uint16_t c_y = 0;
  //for (uint16_t j = 0; j < gri_y; j++)
  //{
    //c_y = c_y + 1;
    //for (uint16_t i = 0; i < gri_x; i++)
    //{
        //c_x = c_x + 1;
        //for (uint16_t j = 0; j < ues_gri_y; j++)
            //{
                //for (uint16_t i = 0; i < ues_gri_x; i++)
                //{
                //positionAllocUE->Add (Vector(distanceUEs * (i + ((c_x - 1) *
                //ues_gri_x)),
                //distanceUEs * (j + ((c_y - 1) * ues_gri_y)),
                //0));
                //}
            //}
    //positionAllocENB->Add (Vector((distanceUEs * i * ues_gri_x) + distanceUEs *
    //((ues_gri_x - 1) / 2),
    //(distanceUEs * j * ues_gri_y) + distanceUEs * ((ues_gri_y - 1) / 2),
    //altura_antena));
    //}
  //}
```

```
MobilityHelper mobility;
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.SetPositionAllocator(positionAllocENB);
mobility.Install (gateways);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.SetPositionAllocator(positionAllocUE);
mobility.Install (endDevices);

// Create the LoraNetDevices of the end devices
phyHelper.SetDeviceType (LoraPhyHelper::ED);
macHelper.SetDeviceType (LoraMacHelper::ED);
helper.Install (phyHelper, macHelper, endDevices);

// Now end devices are connected to the channel

// Connect trace sources
for (NodeContainer::Iterator j = endDevices.Begin ();
     j != endDevices.End (); ++j)
  {
    Ptr<Node> node = *j;
    Ptr<LoraNetDevice> loraNetDevice = node->GetDevice (0)->GetObject<LoraNetDevice> ();
    Ptr<LoraPhy> phy = loraNetDevice->GetPhy ();
    phy->TraceConnectWithoutContext ("StartSending",
                                     MakeCallback (&TransmissionCallback));
  }

// Create a netdevice for each gateway
phyHelper.SetDeviceType (LoraPhyHelper::GW);
macHelper.SetDeviceType (LoraMacHelper::GW);
helper.Install (phyHelper, macHelper, gateways);

// Install reception paths on gateways
for (NodeContainer::Iterator j = gateways.Begin (); j != gateways.End (); j++)
  {
    Ptr<Node> object = *j;
    // Get the device
    Ptr<NetDevice> netDevice = object->GetDevice (0);
    Ptr<LoraNetDevice> loraNetDevice = netDevice->GetObject<LoraNetDevice> ();
    NS_ASSERT (loraNetDevice != 0);
    Ptr<GatewayLoraPhy> gwPhy = loraNetDevice->GetPhy ()->GetObject<GatewayLoraPhy> ();
    Ptr<MobilityModel> gwMob = (*j)->GetObject<MobilityModel> ();
    Vector position = gwMob->GetPosition ();
    position.z = 15;
    gwMob->SetPosition (position);

    // Global callbacks (every gateway)
    gwPhy->TraceConnectWithoutContext ("ReceivedPacket",
                                       MakeCallback (&PacketReceptionCallback));
    gwPhy->TraceConnectWithoutContext ("LostPacketBecauseInterference",
                                       MakeCallback (&InterferenceCallback));
    gwPhy->TraceConnectWithoutContext ("LostPacketBecauseNoMoreReceivers",
                                       MakeCallback (&NoMoreReceiversCallback));
    gwPhy->TraceConnectWithoutContext ("LostPacketBecauseUnderSensitivity",
                                       MakeCallback (&UnderSensitivityCallback));
  }
macHelper.SetSpreadingFactorsUp (endDevices, gateways, channel);

NS_LOG_DEBUG ("Completed configuration");
```

```
  Time appStopTime = Seconds (simulationTime);
  PeriodicSenderHelper appHelper = PeriodicSenderHelper ();
  appHelper.SetPeriod (Seconds (appPeriodSeconds));
  ApplicationContainer appContainer = appHelper.Install (endDevices);
  appContainer.Start (Seconds (0));
  appContainer.Stop (appStopTime);

  if (printEDs)
    {
      PrintEndDevices (endDevices, gateways,
                       "endDevices.dat");
    }

  Simulator::Stop (appStopTime + Hours (2));
  Simulator::Run ();
  Simulator::Destroy ();

  double receivedProb = double(received)/cont;
  double packetlost = cont - double(received);
  double interferedProb = double(interfered)/nDevices;
  double noMoreReceiversProb = double(noMoreReceivers)/nDevices;
  double underSensitivityProb = double(underSensitivity)/nDevices;

  double receivedProbGivenAboveSensitivity = double(received)/(nDevices -
  underSensitivity);
  double interferedProbGivenAboveSensitivity = double(interfered)/(nDevices -
  underSensitivity);
  double noMoreReceiversProbGivenAboveSensitivity = double(noMoreReceivers)/
  (nDevices - underSensitivity);
<< receivedProb << " " << interferedProb
<< " " << noMoreReceiversProb << " " << underSensitivityProb <<  " " <<
receivedProbGivenAboveSensitivity << " " <<
interferedProbGivenAboveSensitivity <<" "<< noMoreReceiversProbGivenAboveSensitivity
<< std::endl;
NS_LOG_UNCOND ("EL AREA DE COBERTURA ES: "<< (distanceUEs * (ues_gri_x - 1)*
distanceUEs * (ues_gri_y - 1))/1000000 << " Km²");
std::cout <<"\nRESULTADOS\nNo. sensores: " << nDevices
          <<"\nsend packets: " << cont
          <<"\nPacket lost: " << packetlost
          <<"\nreceivedProb: " << receivedProb
          <<"\nreceibed packets: " << received
          <<"\n\nThrougputh[bps]: " << (double(received)*50*8)/
          double(simulationTime)
      <<"\npktlost%: " << double(((double(nDevices)*2)-double(received))/
      (double(nDevices)*2))*100
      <<"\n\ninterferedProb: " << interferedProb
          <<"\nnoMoreReceiversProb: " << noMoreReceiversProb
          <<"\nunderSensitivityPrcd ob: " << underSensitivityProb
          <<"\nreceivedProbGivenAboveSensitivity: " <<
          receivedProbGivenAboveSensitivity
          <<"\ninterferedProbGivenAboveSensitivity: " <<
          interferedProbGivenAboveSensitivity
          <<"\nnoMoreReceiversProbGivenAboveSensitivity: " <<
          noMoreReceiversProbGivenAboveSensitivity
            << std::endl;

  return 0;
}
```

Código NS3 LTE

```
// Basado en el script de Jaume Nin <jaume.nin@cttc.cat>
// Red LTE para aplicaciones IoT.

#include "ns3/lte-helper.h"
#include "ns3/epc-helper.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/lte-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-helper.h"
#include "ns3/config-store.h"
#include "ns3/flow-monitor-module.h"
#include <ns3/flow-monitor-helper.h>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("IoTLTE");

//Activa flow monitor que permite ver tripleta
  void ThroughputMonitor (FlowMonitorHelper *fmhelper, Ptr<FlowMonitor> flowMon);

//Permite mostrar en consola la asociación de cada UE a su respetivo eNB
  void NotifyConnectionEstablishedEnb (std::string context,
                                       uint64_t imsi,
                                       uint16_t cellid,
                                       uint16_t rnti)
  {
    std::cout << Simulator::Now ().GetSeconds () << " " << context
              << " eNB CellId " << cellid
              << ": successful connection of UE with IMSI " << imsi
              << " RNTI " << rnti
              << std::endl;
  }

  int
  main (int argc, char *argv[])
  {

//definiendo variables
  double gri_x = 1;     //2
  double gri_y = 1;  //1
  double ues_gri_x = 10;  //16
  double ues_gri_y = 10;  //16
  uint16_t numberOfEnBs = gri_x * gri_y; //10
  uint16_t numberOfUEs = ues_gri_x * ues_gri_y * numberOfEnBs + 1; //560
  double simTime = 40;  // 40 segundos
```

```
  double distanceUEs = 8862.269255/(ues_gri_x-1); // 100 metros
  double interPacketInterval = 19000; //20000 milisegundos
  double m_packetSize = 20;  //80
  double altura_antena = 15.0;
  double escala = 4900;

//Configuración de parámetros LTE
  NS_LOG_UNCOND ("Configurando parámetros LTE...");
  Config::SetDefault ("ns3::LteEnbRrc::SrsPeriodicity", UintegerValue (320));
  Config::SetDefault ("ns3::LteSpectrumPhy::DataErrorModelEnabled", BooleanValue
  (false));
  Config::SetDefault ("ns3::LteSpectrumPhy::CtrlErrorModelEnabled", BooleanValue
  (false));
  Config::SetDefault ("ns3::LteEnbPhy::TxPower", DoubleValue (20));     //40 en dBm
  Config::SetDefault ("ns3::LteEnbPhy::NoiseFigure", DoubleValue (5));
  Config::SetDefault ("ns3::LteEnbPhy::MacToChannelDelay", UintegerValue (2));
  Config::SetDefault ("ns3::LteEnbPhy::UeSinrSamplePeriod", UintegerValue (1));
  Config::SetDefault ("ns3::LteEnbPhy::InterferenceSamplePeriod", UintegerValue (1));
  Config::SetDefault ("ns3::LteUePhy::TxPower", DoubleValue (20));
  Config::SetDefault ("ns3::LteUePhy::NoiseFigure", DoubleValue (9));
  Config::SetDefault ("ns3::LteEnbRrc::DefaultTransmissionMode", UintegerValue (0));
  Config::SetDefault ("ns3::LteUePhy::TxMode1Gain", DoubleValue (0.0));
  Config::SetDefault ("ns3::LteEnbNetDevice::UlBandwidth", UintegerValue (6));
  Config::SetDefault ("ns3::LteEnbNetDevice::DlBandwidth", UintegerValue (6));
  Config::SetDefault ("ns3::LteEnbNetDevice::DlEarfcn", UintegerValue (5220));
  Config::SetDefault ("ns3::LteEnbNetDevice::UlEarfcn", UintegerValue (23220));
  Config::SetDefault ("ns3::LteHelper::PathlossModel", StringValue
  ("ns3::LogDistancePropagationLossModel"));
  Config::SetDefault ("ns3::LteHelper::FadingModel", StringValue ());
  Config::SetDefault ("ns3::LteHelper::UsePdschForCqiGeneration", BooleanValue
  (true));
  CommandLine cmd;
  cmd.Parse(argc, argv);
//Crea EPC definido por un nodo SGW/PGW
  NS_LOG_UNCOND ("Creando EPC...");
  Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
  Ptr<PointToPointEpcHelper>  epcHelper = CreateObject<PointToPointEpcHelper> ();
  lteHelper->SetEpcHelper (epcHelper);
//Configuracion del Scheduler LTE MAC
  lteHelper->SetSchedulerType ("ns3::RrFfMacScheduler");
//Crea objeto gateway para salir a internet
  Ptr<Node> pgw = epcHelper->GetPgwNode ();
  NS_LOG_UNCOND ("Creando host remoto...");
  NodeContainer remoteHostContainer;
  remoteHostContainer.Create (1);
  Ptr<Node> remoteHost = remoteHostContainer.Get (0);
  InternetStackHelper internet;
  internet.Install (remoteHostContainer);

// Create the Internet
  NS_LOG_UNCOND ("Creando acceso a Internet..");
  PointToPointHelper p2ph;
  p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate ("1Gb/s"))); //100G
  p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
  p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.010)));
```

```
  NetDeviceContainer internetDevices = p2ph.Install (pgw, remoteHost);
  Ipv4AddressHelper ipv4h;
  ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
  Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (internetDevices);
  Ipv4Address remoteHostAddr = internetIpIfaces.GetAddress (1);
//Ruteo de internet a lte red
  Ipv4StaticRoutingHelper ipv4RoutingHelper;
  Ptr<Ipv4StaticRouting> remoteHostStaticRouting = ipv4RoutingHelper.GetStaticRouting
  (remoteHost->GetObject<Ipv4> ());
  remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("7.0.0.0"),
  Ipv4Mask ("255.0.0.0"), 1);
//Contenedores para crear nodos Ues y Enbs
  NS_LOG_UNCOND ("Creando y ubicando sensores y ENBs");
  NodeContainer ueNodes;
  NodeContainer enbNodes;
  enbNodes.Create(numberOfEnBs);
  ueNodes.Create(numberOfUEs);
//Crea objeto Posición que definirá una matriz de posición física de los nodos
Ptr<ListPositionAllocator> positionAllocUE = CreateObject<ListPositionAllocator> ();
Ptr<ListPositionAllocator> positionAllocENB = CreateObject<ListPositionAllocator> ();
//Define la ubicación de UEs y EnB, en el área circular
positionAllocUE->Add(Vector(escala*0.49854,escala*0.86687,0));
positionAllocUE->Add(Vector(escala*0.29997,escala*-0.95395,0));
positionAllocUE->Add(Vector(escala*-0.092776,escala*0.25636,0));
positionAllocUE->Add(Vector(escala*-0.93765,escala*-0.34757,0));
positionAllocUE->Add(Vector(escala*0.36459,escala*0.54813,0));
positionAllocENB->Add(Vector(0,0,altura_antena));

//Arma las grillas de posición de UEs para cada EnB y la grilla de posición
  //uint16_t c_x = 0;
  //uint16_t c_y = 0;
  //for (uint16_t j = 0; j < gri_y; j++)
  //{
    //c_y = c_y + 1;
    //for (uint16_t i = 0; i < gri_x; i++)
    //{
        //c_x = c_x + 1;
        //for (uint16_t j = 0; j < ues_gri_y; j++)
            //{
                //for (uint16_t i = 0; i < ues_gri_x; i++)
                //{
                //positionAllocUE->Add (Vector(distanceUEs * (i + ((c_x - 1) *
                //ues_gri_x)),
                //distanceUEs * (j + ((c_y - 1) * ues_gri_y)),
            //0));
                //}
            //}
    //positionAllocENB->Add (Vector((distanceUEs * i * ues_gri_x) + distanceUEs *
    //((ues_gri_x - 1) / 2),
    //(distanceUEs * j * ues_gri_y) + distanceUEs * ((ues_gri_y - 1) / 2),
    //altura_antena));
    //}
  //}
//Asocia las matrices de posición física de los UEs y EnBs a los nodos
  MobilityHelper mobility;
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.SetPositionAllocator(positionAllocENB);
  mobility.Install (enbNodes);
```

```
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.SetPositionAllocator(positionAllocUE);
  mobility.Install (ueNodes);

// Instala dispositivos LTE a los nodos
  NetDeviceContainer enbLteDevs = lteHelper->InstallEnbDevice (enbNodes);
  NetDeviceContainer ueLteDevs = lteHelper->InstallUeDevice (ueNodes);

// Instala el stack IP en los UEs
  internet.Install (ueNodes);
  Ipv4InterfaceContainer ueIpIface;
  ueIpIface = epcHelper->AssignUeIpv4Address (NetDeviceContainer (ueLteDevs));

// Asigna una IP a cada UE y define su Gateway
  NS_LOG_UNCOND ("Asignando IPs a los sensores...");
  for (uint32_t u = 0; u < ueNodes.GetN (); ++u)  //
    {
      Ptr<Node> ueNode = ueNodes.Get (u);
      Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting
      (ueNode->GetObject<Ipv4> ());
      ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (),
      1);
    }

// Asocia los UEs a su respectivo EnB
  NS_LOG_UNCOND ("Asociando los sensores a sus respectivos ENBs...");
  uint16_t aux = ues_gri_x * ues_gri_y;
  uint16_t cont = 0;
  for (uint16_t i = 0; i < numberOfEnBs; i++)
    {
    for(uint16_t j = 0; j < aux; j++)
            {
             cont = cont + 1;
             lteHelper->Attach (ueLteDevs.Get(cont), enbLteDevs.Get(i));
            }
    }

// Define el socket UDP y lo configura para envío de paquetes al RemoteHost
   NS_LOG_UNCOND ("Configurando comportamiento de los sensores...");
  uint16_t ulPort = 5000;
  ApplicationContainer clientApps;
  ApplicationContainer serverApps;
  for (uint32_t u = 0; u < ueNodes.GetN (); ++u)
    {
      ++ulPort;
      PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory",
      InetSocketAddress
      (Ipv4Address::GetAny (), ulPort));
      serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
      UdpClientHelper ulClient (remoteHostAddr, ulPort);
      ulClient.SetAttribute ("Interval", TimeValue (MilliSeconds
      (interPacketInterval)));
      ulClient.SetAttribute ("MaxPackets", UintegerValue(1000000));
      ulClient.SetAttribute ("PacketSize", UintegerValue (m_packetSize));
      clientApps.Add (ulClient.Install (ueNodes.Get(u)));
    }
  serverApps.Start (Seconds (0.01));
  clientApps.Start (Seconds (0.01));
  lteHelper->EnableTraces ();
```

```
//Sirve para mostrar en consola la asociación de cada UE a su respetivo eNB
  Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/ConnectionEstablished",
  MakeCallback (&NotifyConnectionEstablishedEnb));

 NS_LOG_UNCOND ("...");
 NS_LOG_UNCOND ("...");
 NS_LOG_UNCOND ("EL AREA DE COBERTURA ES: "<< distanceUEs * (ues_gri_x - 1)*
 distanceUEs * (ues_gri_y - 1) << " m²");

// Permite habilitar Flow Monitor
  NS_LOG_UNCOND ("Habilitando Flowmonitor...");
  NodeContainer monitoredNodes;
  monitoredNodes.Add (ueNodes);
  monitoredNodes.Add (remoteHost);
  Ptr<FlowMonitor> flowMonitor;
  FlowMonitorHelper flowHelper;
  flowHelper.SetMonitorAttribute("MaxPerHopDelay", TimeValue(Seconds(1.0)));
  flowMonitor = flowHelper.Install(monitoredNodes);

  Simulator::Stop(Seconds(simTime));

  Simulator::Run();

  flowMonitor->SerializeToXmlFile("tripleta.xml", false, true);

  Simulator::Destroy();
  return 0;

}
```

Código Matlab

```
clc
close all
clear

%% Iniciatilize

nodes = 16;
dgw_min = 0.01;
LL=6500;
radius = rand(1,nodes);
angle = rand(1,nodes)*2*pi;
xpos = radius.*cos(angle);
ypos = radius.*sin(angle);

hold off
h = zeros(1,nodes);
t = zeros(1,nodes);

hold on
rectangle('Position',[-1 -1 2 2],'Curvature',[1,1]);
rectangle('Position',[-dgw_min -dgw_min dgw_min*2 dgw_min*2],'Curvature',[1,1]);
daspect([1,1,1])
grid on

for n = 1:nodes
    h(n) = plot(xpos(n), ypos(n), 'xb');
    t(n) = text(xpos(n), ypos(n), num2str(n));
end

pause(0.01)

search_radius = 0.1;
max_tries = ceil(256/nodes);
max_tries = 10;
tries = 0;

for n=1:nodes
    [xpos(n), ypos(n)] = boundcc(xpos(n), ypos(n),dgw_min);
end

while (1)

    save_xpos = xpos;
    save_ypos = ypos;
   dist = metricc(xpos, ypos, 5);
    change = zeros(1,nodes);
    for n = 1:nodes
        xpos(n) = xpos(n) + (rand(1)-rand(1))*search_radius;
        ypos(n) = ypos(n) + (rand(1)-rand(1))*search_radius;
        [xpos(n), ypos(n)] = boundc(xpos(n), ypos(n),dgw_min);
```

```
        new_dist = metricc(xpos, ypos, 5);
        if new_dist > dist  %Ignora ultimo cambio
            xpos(n) = save_xpos(n);
            ypos(n) = save_ypos(n);
            else
            dist = new_dist;
            change(n) = 1;
        end
        set(h(n), 'XData', xpos(n), 'YData', ypos(n))
        set(t(n), 'Position', [xpos(n), ypos(n)])
        pause(0.1/nodes)
    end

    if sum(change) == 0
        tries = tries + 1;
        if tries >= max_tries
            break
        end
    else
        tries = 0;
    end
end

hold off

for n = 1:nodes
    eval('disp([''positionAllocUE->Add(Vector('', num2str(xpos(n)), '','',
    num2str(ypos(n)), '',0));''])')
end
eval('disp([''positionAllocENB->Add(Vector('', ''0'', '','', ''0'', '',
altura_antena));''])')

clearvars -EXCEPT xpos ypos
save C:\Users\USER\Documents\MATLAB\Tesis\cordena\xpos4.mat
```

Código R

```
##Ejemplos Gráficos

Datos = read.table("lora_lte.txt", header = TRUE, dec = ",")
Datos1 = read.table("lora_lte1.txt", header = TRUE, dec = ",")

plot(Datos$Radio.m..4, Datos$THR_LORA_CU.bps..4, type = "l",lwd=2,
     col=2,
     ylim =c(0,500),
     main = "THROUGHPUT CON 4 SENSORES",
     xlab = "Radio [m]",
     ylab = "Throughput [b/s]" ,
     panel.first=grid())
lines(Datos$Radio.m..4, Datos$THR_LORA_CI.bps..4, col=4,lwd=2)
lines(Datos$Radio.m..4, Datos$THR_LTE_CU.bps..4, col=9,lwd=2)
lines(Datos$Radio.m..4, Datos$THR_LTE_CI.bps..4, col=11,lwd=2,lty=2)
legend(4000,500,
       c("LoRa","LoRa (Ópt)","LTE","LTE (Ópt)"),
       col=c(2,4,9,11),
       lwd=c(1,1,1,1),
       cex=0.7)
lines(1000,400,type = "h",lty=2,lwd=1)
lines(4000,250,type = "h",lty=2,lwd=1)
lines(5000,120,type = "h",lty=2,lwd=1)
text(1000,430,"Cobertura LTE y LTE Ópt", adj=c(0.2,0.2),cex=0.7)
text(3500,280,"Cobertura LoRa", adj=c(0.2,0.2),cex=0.7)
text(4300,150,"Cobertura LoRa Ópt", adj=c(0.2,0.2),cex=0.7)


plot(Datos$Radio.m..4, Datos$THR_LORA_CI.bps..100, type = "o",
     col=2,
     pch=15 ,
     ylim =c(0,2000),
     main = "THROUGHPUT LoRa ÓPTIMO",
     xlab = "Radio [m]",
     ylab = "Throughput [b/s]",
     panel.first=grid())
lines(Datos$Radio.m..4, Datos$THR_LORA_CI.bps..81, col=4,
      pch=17 ,type = "o")
lines(Datos$Radio.m..4, Datos$THR_LORA_CI.bps..64, col=9,
      pch=19 ,type = "o")
lines(Datos$Radio.m..4, Datos$THR_LORA_CI.bps..49, col=11,
      pch=21 ,type = "o")
lines(Datos$Radio.m..4, Datos$THR_LORA_CI.bps..36, col=13,
      pch=23 ,type = "o")
lines(Datos$Radio.m..4, Datos$THR_LORA_CI.bps..25, col=15,
      pch=25 ,type = "o")
lines(Datos$Radio.m..4, Datos$THR_LORA_CI.bps..16, col=16,
      pch=24 ,type = "o")
legend(4450,2000,
```

```
        c("100 sen","81 sen","64 sen","49 sen","36 sen","25 sen","16 sen"),
        col=c(2,4,9,11,13,15,16),
        pch = c(15,17,19,21,23,25,24),
        cex=0.7)
        lines(2000,max(Datos$THR_LORA_CI.bps..100),type = "h",lty=2)
text(2000,max(Datos$THR_LORA_CI.bps..100)+100,"Max Throughput", adj=c(0.2,0.2),
cex=0.7)


##Análisis de Datos

library(dplyr)
library(mgcv)
library(splines)
require(quantmod) #
require(reshape) #

require(nortest)
require(outliers)
library(nls2)

#DatA = read.table("lora_lte_sim.txt", header = TRUE, dec = ",")
DatA1 = read.table("Tesis/lora_dis.txt", header = TRUE, dec = ",")
DatA2 = read.table("Tesis/lora_dis_opt.txt", header = TRUE, dec = ",")
DatA3 = read.table("Tesis/lte_dis.txt", header = TRUE, dec = ",")

DatA11 = read.table("Tesis/lora_dis_pl.txt", header = TRUE, dec = ",")
DatA22 = read.table("Tesis/lora_dis_opt_pl.txt", header = TRUE, dec = ",")
DatA33 = read.table("Tesis/lte_dis_pl.txt", header = TRUE, dec = ",")

#-----------------FUNCIONES-----------------
muestreo10<-function(P)       ##Función muestreo para completar datos
{
  a=10
  aux <- sample(P,a,replace=T)
  sal <- c(P,aux)
  #sal <- data.frame(sal)
  sal5 <- cbind(sal)
  return(sal5)
}
 muestreo11<-function(P)       ##Función muestreo para completar datos
{
   a=11
   aux <- sample(P,a,replace=T)
   sal <- c(P,aux)
   #sal <- data.frame(sal)
   sal5 <- cbind(sal)
   return(sal5)
}
```

```r
muestreo12<-function(P)        ##Función muestreo para completar datos
{
  a=12
  aux <- sample(P,a,replace=T)
  sal <- c(P,aux)
  sal5 <- cbind(sal)
  return(sal5)
}

normtest <- function (Example)  ##Función test de normalidad
{
  t1 <- shapiro.test(Example)
  t2 <- ad.test(Example)
  t3 <- cvm.test(Example)
  t4 <- lillie.test(Example)
  t5 <- pearson.test(Example)
  t6 <- sf.test(Example)
  normt <- c(0,0,0,0,0,0)
  normt[1] <- t1$p.value
  normt[2] <- t2$p.value
  normt[3] <- t3$p.value
  normt[4] <- t4$p.value
  normt[5] <- t5$p.value
  normt[6] <- t6$p.value
  return(normt)
}

#Consideraciones específicas

dato <- DatA1$m5000

#Scrip general
#plot(density(dato))
normtest(dato)
datosample <- muestreo10(dato)
length(datosample)
normtest(datosample)

grubbs.test(datosample, type = 10, opposite = FALSE, two.sided = FALSE)
grubbs.test(datosample, type = 10, opposite = TRUE, two.sided = FALSE)
grubbs.test(datosample, type = 11, opposite = FALSE, two.sided = TRUE)
grubbs.test(datosample, type = 11, opposite = TRUE, two.sided = TRUE)
grubbs.test(datosample, type = 20, opposite = FALSE, two.sided = FALSE)
grubbs.test(datosample, type = 20, opposite = TRUE, two.sided = FALSE)
```

```r
datooutlier<-dato[-c(7,3)]
datooutlier
datooutlier <- muestreo12(datooutlier)
length(datooutlier)
normtest(datooutlier)
plot(density(datooutlier))
plot(density(dato))
final<-datooutlier
final<-datosample
throug[length(throug)+1] <- mean(final)
throug
throug_sd[length(throug_sd)+1] <- sd(final)
```

```
throug_sd
r<-normtest(final)
throug_pvalu[length(throug_pvalu)+1] <- r[1]

#write.table(throug_opt, "/througfinal_opt.txt", sep="\t")
#write.table(throug_sd_opt, "/throug_sd_final_opt.txt", sep="\t")
#write.table(throug_pvalu_opt, "/throug_pvalue_final_opt.txt", sep="\t")

#write.table(throug, "/througfinal.txt", sep="\t")
#write.table(throug_sd, "/throug_sd_final.txt", sep="\t")
#write.table(throug_pvalu, "/throug_pvalue_final.txt", sep="\t")

plot(throug, type = "l")
lines(throug_opt, type = "l", col=2)
plot(density(throug))
lines(density(throug_opt), col=2)

tr = read.table("Tesis/througfinal.txt", header = TRUE)
tr_opt = read.table("Tesis/througfinal_opt.txt", header = TRUE)
pl = read.table("Tesis/pkltfinal.txt", header = TRUE)
pl_opt = read.table("Tesis/pkltfinal_opt.txt", header = TRUE)
tr_lte = read.table("Tesis/lte_dis.txt", header = TRUE, dec = ",")
pl_lte = read.table("Tesis/lte_dis_pl.txt", header = TRUE, dec = ",")

genericdata = pl_lte

dist <- c(1:50)
#data <- genericdata$x     ##para lora
data <- genericdata$pck_ci   ##para lte
plot(data)

#80% modelo 20% validación
set.seed(125)
data_ran<-sample(data,50,replace=F)
set.seed(125)
dist_ran<-sample(dist,50,replace=F)
data_80 <- data_ran[1:40]
data_20 <- data_ran[41:50]
dist_80 <- dist_ran[1:40]
dist_20 <- dist_ran[41:50]
data_80 <- cbind(dist_80, data_80)
data_20 <- cbind(dist_20, data_20)
data_80 <- data.frame(data_80)
data_20 <- data.frame(data_20)

data_80 <- arrange(data_80, dist_80)
data_20 <- arrange(data_20, dist_20)
plot(data_20)
plot(data_80)
plot(data)

#Modelos univariables
model1 <- lm(data_80 ~ dist_80, data = data_80)
model2 <- lm(data_80 ~ poly(dist_80, 4), data = data_80)
model3 <- lm(data_80 ~ poly(dist_80, 3), data = data_80)
```

```
#model4 <- nls(data_80 ~ ((atan(-(dist_80-a)/b))*c)+d,    ##para lora
 #             data = data_80,
  #            start = list(a = 36, b = 5, c = 100, d = 1166),
   #           trace = TRUE)

model4 <- nls(data_80 ~ a*dist_80^b,
                        data = data_80,
                        start = list(a = 4, b=2),
                        trace = TRUE)
knots <- quantile(data_80$dist_80, p = c(0.25, 0.5, 0.75))
model5 <- lm (data_80 ~ bs(dist_80, knots = knots), data = data_80)
model6 <- gam(data_80 ~ s(dist_80), data = data_80)

#Evaluación de modelos

model <- model6

plot(fitted(model)~data_80$dist_80)

summary(model)
plot(residuals(model))
plot(density(residuals(model)))
#sd(residuals(model))
RMSE <- sqrt(mean((residuals(model))^2))
RMSE
Test_norm <- shapiro.test(residuals(model))
Test_norm$p.value
Media_err <- mean(residuals(model))
Media_err

new <- data.frame(dist_80=data_20$dist_20)
pre<-predict(model, newdata = new)
plot(data_20)
lines(pre~data_20$dist_20, type="l")

new <- data.frame(dist_80=dist)
pre<-predict(model, newdata = new)
plot(data)
lines(pre~dist, type="l")

residuos <- data-pre
plot(residuos)
plot(density(residuos))
RMSE <- sqrt(mean(residuos^2))
RMSE
Test_norm <- shapiro.test(residuos)
Test_norm$p.value
Media_err <- mean(residuos)
Media_err
AIC_model <- AIC(model)
AIC_model

#Guardar datos

#RMSE_lora_p_opt<-NULL
RMSE_lora_p_opt[length(RMSE_lora_p_opt)+1] <- RMSE
RMSE_lora_p_opt
```

```
#NORM_lora_p_opt<-NULL
NORM_lora_p_opt[length(NORM_lora_p_opt)+1] <- Test_norm$p.value
NORM_lora_p_opt
#MEDE_lora_p_opt<-NULL     ##cague lora_p_opt
MEDE_lora_p_opt[length(MEDE_lora_p_opt)+1] <- Media_err
MEDE_lora_p_opt
#AIC_lte_p_opt<-NULL     ##cague lora_p_opt
AIC_lte_p_opt[length(AIC_lte_p_opt)+1] <- AIC_model
AIC_lte_p_opt
MOD_lte_p_opt <- model6
AJU_lte_p_opt <- pre
TRO_lte_p_opt <- genericdata$pck_ci    ##para lora
#TRO_lte_p_opt <- genericdata    ##para lte
#GRAFICOS Ajuste vs Datos
#troughput_lora
plot(TRO_lora_t~dist, type = "l", col = 2, lwd=2,
     main = "THROUGHPUT LORA",
     xlab = "Radio*100 [m]",
     ylab = "Throughput [bps]")
lines(AJU_lora_t~dist, type = "l", col = 4, lwd=2)
legend(45,1700,
       c("Datos","Ajuste"),
       col=c(2,4),
       pch = c(15,15),
       cex=0.7)
##Ejemplo Test de hipotesis
#TRO
#LORA ÓPTIMO vs LORA
shapiro.test(AJU_lora_t_opt)
shapiro.test(AJU_lora_t)
wilcox.test(AJU_lora_t_opt , AJU_lora_t, alternative = "two.sided")
median(AJU_lora_t_opt)
median(AJU_lora_t)
##EJEMPLO GRAFICO DE MODELOS
plot(density(AJU_lora_t_opt), type = "l", col = 2, lwd=2,#lora op vs lora
     main = "THROUGHPUT LORA",
     xlab = "Throughput [bps]")
cord.x <- density(AJU_lora_t_opt)$x
cord.y <- density(AJU_lora_t_opt)$y
polygon(cord.x, cord.y, col = rgb(1, 0, 0, alpha = 0.4))
lines(density(AJU_lora_t), type = "l", col = 4, lwd=2)
cord.x <- density(AJU_lora_t)$x
cord.y <- density(AJU_lora_t)$y
polygon(cord.x, cord.y, col = rgb(0, 0, 1, alpha = 0.4))
abline(v=median(AJU_lora_t_opt), col = 2, lwd=2)
abline(v=median(AJU_lora_t), col = 4, lwd=2,)
legend(0,0.0011,
       c("Óptimo","Grilla"),
       col=c(2,4),
       pch = c(15,15),
       cex=0.7)
```