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

ANÁLISIS DE COGNITIVE RADIO EN REDES MÓVILES

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

PABLO GEOVANNY PALACIOS JÁTIVA

PROFESOR GUÍA:
ALBERTO CASTRO ROJAS

MIEMBROS DE LA COMISIÓN:
CÉSAR AZURDIA MEZA
JUAN PÉREZ RETAMALES

SANTIAGO DE CHILE
2017

RESUMEN DE LA TESIS PARA OPTAR
AL TÍTULO DE MAGÍSTER EN INGENIERÍA DE REDES DE COMUNICACIONES
POR: PABLO GEOVANNY PALACIOS JÁTIVA
FECHA: 2017
PROF. GUÍA: SR. ALBERTO CASTRO ROJAS

ANÁLISIS DE COGNITIVE RADIO EN REDES MÓVILES

El análisis del desempeño de Cognitive Radio (CR) genera un gran interés en las comunidades académicas e industriales para satisfacer las crecientes necesidades de los recursos de espectro y la comunicación de datos de alta velocidad.

La principal ventaja que ofrece CR es la eficiencia en el uso del espectro electromagnético, ya que permite una gestión óptima del espectro a través de un proceso de cuatro etapas (ciclo cognitivo): detección del espectro, decisión del espectro, compartición del espectro y movilidad. Dentro de todo el ciclo cognitivo, las etapas mas críticas son las de detección y decisión, siendo estas áreas de enorme importancia para todo el proceso cognitivo y presentan dificultades al tener mayores exigencias de hardware para la detección adecuada de los Primary Users (PU) y potenciales problemas de seguridad en la red. Bajo estos parámetros, la detección correcta y efectiva de los PU se convierte en una prioridad. Por lo tanto, se han propuesto varios métodos, incluyendo algoritmos de detección, esquemas de detección de energía, detección basada en el valor propio, detección de descomposición de valores singulares SVD, entre otros.

En este trabajo se analiza el método de detección de PU SVD, por medio del desarrollo de un simulador CR aplicado a las tecnologías móviles WiFi y LTE, estructurado con cada una de las cuatro etapas de un sistema CR, para lo cual en cada una de estas etapas se usan métodos y algoritmos evaluados en trabajos previos de manera individual. La implementación del simulador CR adapta de manera conjunta a los distintos métodos, los cuales son: Singular Value Decomposition (SVD), Teoría del juego de coaliciones y handoff por RSSI. Se usa el software Network Simulator 3 (NS-3) como entorno de desarrollo de una red cognitiva heterogénea basado en el simulador creado, permitiendo evaluar el rendimiento del método de detección Singular Value Decomposition (SVD) aplicado a una red móvil CR, esto permite validar el método en una red móvil funcional.

El rendimiento del método de detección SVD se analiza en términos de probabilidad de detección (P_{d}) versus Signal to Noise Ratio (SNR) mediante simulaciones numéricas y se compara con el método tradicional de Energy Detection y el algoritmo de detección SVD Teórico, utilizando el estimador estadístico Maximum Likelihood Estimation (MLE).

Utilizando el estimador estadístico MLE, los resultados obtenidos fueron los siguientes: MLE (SVD propuesta) = 0,587388, MLE (SVD) = 0,524820, MLE (Energy Detection) = 0,495913, con una desviación estándar de SD (SVD propuesto) = 0.2817681, SD (SVD teórico) = 0.3383261, SD (Energy detection teórico) = 0.4970421. En consecuencia, el sistema evaluado supera los métodos teóricos propuestos en términos de P_{d}.

Agradecimientos

"Si quieres que tu vida sea diferente, tienes que hacer algo diferente".

Nada de esto hubiera sido posible sin la ayuda de Dios, mis padres Silvia Játiva y Ángel Palacios, mi hermana Krhistin Palacios y mi familia, que fueron mi fortaleza en momentos de flaqueza; a pesar de la distancia siempre estuvieron, están y estarán para apoyarme, este logro es para Ustedes. A mis compañeros de Maestría, con los cuales formamos un grupo muy unido, apoyándonos ante cualquier dificultad y cultivando una gran amistad.

A mi país, cuyo representante, el Gobierno con la Senescyt, me otorgaron la beca de estudios, confiaron en mí, y no los he defraudado.

A mis profesores, en especial a mi profesor guía Alberto Castro, que con su conocimiento y experiencia, supo encaminarme en el desarrollo de este trabajo.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

1. Antecedentes

La tecnología conocida como Cognitive Radio (CR), ha evolucionado con el transcurso del tiempo, desde llamarse en una primera instancia Software Radio (SR), la cual era una radio reconfigurable, capaz de realizar distintas funciones en tiempos diferentes, únicamente realizándole cambios en su configuración mediante software. Luego paso a ser Software-Defined Radio (SDR), que no era otra cosa que una versión de SR implementable y mejor estructurado. Una de las primeras pruebas de SDR fue el proyecto militar estadounidense *SpeakEasy* que implementó diez tipos de tecnologías inalambricas en un rango de 2 Mhz hasta los 200 Mhz, en un solo equipo reprogramable. Todo esto se llevo acabo hasta llegar a 1999, Donde Joseph Mitola III definió a CR como un sistema SDR inteligente, capaz de detectar necesidades de comunicación de los usuarios, poder conocer su entorno y gracias a este conocimiento llegar a satisfacer las necesidades de dichos usuarios [6].

En la actualidad no existen redes CR desplegadas comercialmente, a pesar de que las ideas del sistema de compartir diferentes tecnologías sin producir interferencia ya ha sido demostrada. Se han realizado pruebas en la banda de los 5 GHz de 802.11a para ser compartida con los radares de gran sensibilidad de navegación aeronáutica, también se han realizado pruebas e incluso estandarizado la tecnología para tener acceso dinámico al espectro de TV en la banda entre 400-800 Mhz lo que se conoce como Cognitive Radio Wireless Regional Area Network (WRAN) y su estándar IEEE-802.22 [7].

En los últimos diez años ha existido un significativo interés en la investigación de CR en cada una de sus etapas, tanto en la academia como en la industria, presentando muchas propuestas, métodos y algoritmos para su implementación en un sistema CR, pero hasta el momento una arquitectura con métodos estandarizados no ha podido ser definida [7].

2. Hipótesis

"El desempeño del método de detección Singular Value Decomposition (SVD) implementado en una red móvil CR en términos de Probabilidad de detección (P_{d}) vs SNR es mas eficiente comparado con los métodos de detección SVD teórico y Energy Detection teórico."

3. Objetivos

3.1. Objetivo general

Comprobar la eficiencia en el desempeño del método de detección Singular Value Decomposition (SVD) implementado en una red móvil CR en términos de Probabilidad de detección (P_{d}) vs SNR, comparado con los métodos de detección SVD teórico y Energy Detection teórico.

3.2. Objetivos específicos

3.2.1. Analizar el mecanismo de CR y sus cuatro principios fundamentales tanto en el ámbito académico-investigativo como en estándares en la industria.

3.2.2. Usar los algoritmos de Teoría del juego aplicables a CR

3.2.3. Realizar un desarrollo para su uso en el software NS-3, con los algoritmos a utilizar en detección, decisión, compartición y movilidad de CR, aplicados a las redes móviles WiFi y LTE.

3.2.4. Realizar escenarios de simulación aplicables a la red estructurada en NS-3.

3.2.5. Obtener datos de los escenarios experimentales realizados

3.2.6. Analizar los datos obtenidos mediante métodos estadísticos válidos.

Para poder cumplir con el objetivo general y con los objetivos específicos, se implementó en un desarrollo para el software NS-3, las etapas de detección, decisión, compartición y movilidad de una red CR, aplicadas a las tecnologías móviles WiFi y LTE, mediante el uso de algoritmos y métodos que involucren en una de las etapas del sistema la Teoría del Juego y el método de detección SVD, para luego poder analizarlo mediante simulaciones.

4. Metodología

La etapas del desarrollo del trabajo son: investigación, desarrollo, simulación y análisis. Investigación de papers académicos y estándares en la industria que involucren esquemas de detección de espectro, decisión de uso de espectro mediante algoritmos de teoría del juego, compartición del espectro con usuarios licenciados y movilidad para el uso de frecuencias. Desarrollo de escenarios de simulación en NS-3 para obtener datos

experimentales y su posterior análisis mediante modelos estadísticos.

Capítulo 1

Marco Teórico

En este capítulo nos vamos a referir a los conceptos básicos de probabilidad de detección (P_d) y SNR, luego a los mecanismos y algoritmos existentes para cada una de las etapas de Cognitive Radio, seguido de una reseña de los estándares y desarrollos existentes en la industria y la academia asociados a CR, además de incluir una pequeña sección de la herramienta estadística para el análisis de datos usada.
En sentido general, el acceso dinámico al espectro requiere de cuatro funcionalidades estrechamente vinculadas al ciclo cognitivo: 1) Identificar las oportunidades de acceso al espectro (spectrum detection). 2) Seleccionar las bandas de frecuencia a utilizar (spectrum decision). 3) Coordinar el acceso al espectro con otros usuarios secundarios (spectrum sharing). 4) Desocupar los canales utilizados cuando son requeridos por los PU (spectrum mobility/handoff).

1.1. Probabilidad de detección (P_d)

En las redes CR existen dos tipos de usuarios: licenciados o Primary Users (PU) y usuarios sin licencia o Secundary Users (SU). Los SU están detectando y midiendo periódicamente las bandas del espectro para verificar los canales que se encuentren libres [8].
La detección del espectro tiene dos objetivos: reducir las interferencias perjudiciales entre la comunicación de los PU mientras están usando el canal y al mismo tiempo mejorar el uso efectivo del espectro mientras no está siendo usado por los PU [9]. El rendimiento de la detección del espectro se mide por probabilidad de detección de ocupación (P_d).
(P_d) es la probabilidad de que un canal sea declarado como ocupado cuando está ocupado en realidad, es decir cuando un PU esta usando el canal.
También se encuentra la Probabilidad de falsa alarma (P_f) que es la probabilidad de que se anuncie un canal como ocupado cuando realmente no lo está [10] [11].

1.2. Signal to Noise Ratio (SNR)

La razón Señal a Ruido SNR es una medida que indica cuánto se ha corrompido una señal $x(t)$ en presencia del ruido $n(t)$, Donde $x(t)$ es la señal recibida por el SU y $n(t)$ es Additive White Gaussian Noise (AWGN). Esta relación nos permite evaluar la información de una señal, en particular la presencia o ausencia de eventos que pueden confundirse con otros equipos o ruido [12].
El SNR se define (en escalas logarítmicas) como el cociente entre la potencia de la señal y la potencia del ruido AWGN[13]:

$$SNR_{\mathrm{db}} = 20log_{10}[\frac{x_{rms}}{n_{rms}}]. \tag{1.1}$$

El rendimiento de un método de detección se mide por la capacidad para lograr una cierta (P_{d}) teniendo muchas veces como parámetro de entrada una (P_f) específica para un valor de SNR dado.

1.3. Spectrum Detection

Los métodos de identificación de las oportunidades de acceso al espectro pueden clasificarse como pasivos o activos. Para nuestro análisis nos centraremos en los parámetros de identificación activa ya que es donde se encuentra el tipo de detección que se implementó en el proyecto.
Se parte con la suposición de que los usuarios con capacidad CR, también llamados Secundary Users (SU) de la red, no tienen ninguna interacción en tiempo real con los transmisores y receptores primarios de la red, también llamados Primary Users (PU) [1]. Para la detección del transmisor y para poder distinguir entre bandas usadas y sin usar, los SU detectan la señal de un transmisor principal sólo a través de las observaciones locales, que deben tener la capacidad de determinar si la señal del PU, está presente [1].
Se plantea una prueba de hipótesis básica para la detección del transmisor [1]:

$$x(t) = \begin{cases} n(t) & H_0 \\ h(t) * s(t) + n(t) & H_1 \end{cases} \tag{1.2}$$

Donde $x(t)$ es la señal recibida por el SU, $s(t)$ es la señal transmitida por el PU, $n(t)$ es el Additive White Gaussian Noise (AWGN) y $h(t)$ es la ganancia del canal. La hipótesis nula (H_0) establece que no existe señal de usuario licenciado en cierta banda. La hipótesis alternativa (H_1) establece que existe alguna señal de un usuario primario [1].
Este esquema se lo puede observar en la Figura1.1.

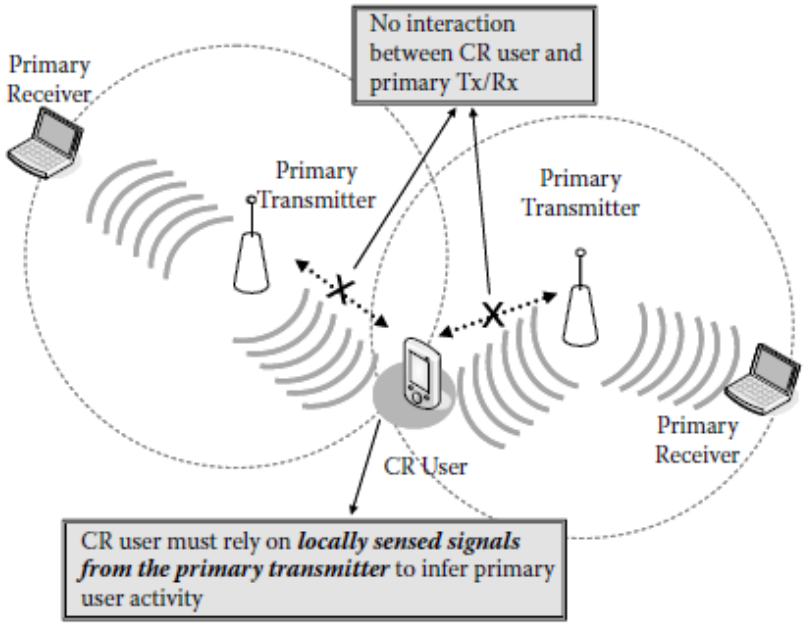


Figura 1.1: Detección de PU [1]

Los métodos que generalmente se usan para detectar un transmisor primario son:

- Matched filter detection o Coherent detection
- Energy detection
- Feature detection

1.3.1. Match Filter Detection

Características [1]

- Óptimo cuando la información del PU es conocida por el SU.
- Como conocimiento previo acerca de las características de la señal del PU se puede optar por tipo y orden, forma de pulso, velocidad de datos, o propiedades estadísticas de dicha señal.

Desventajas [1]

- La desventaja de este método es que además de que se requiere de antemano un conocimiento de las características de la señal del PU, se requiere la sincronización entre el

transmisor primario y el SU. Si esta información no es correcta, o si la sincronización es mala, los resultados de este método de detección serán pobres.

1.3.2. Energy Detection

Características [1]

- Si el SU no tiene suficiente información sobre la señal transmitida por el SU, este método es mas óptimo.
- El método consiste en que los SU detectan la presencia o ausencia de los PU a través de la energía de la señal primaria recibida.
- El método requiere dos veces más número de muestras para determinar una probabilidad de error, por lo tanto, si un SU necesita detectar señales débiles, por ejemplo con un SNR entre 10 dB a -40 dB, este filtro necesita de más tiempo de detección en comparación con otros métodos.
- La característica principal de este método es su facilidad de implementación.

Desventajas [1]

- Sólo puede determinar la presencia de la señal, pero no su tipo, por lo tanto, en ciertas ocasiones genera falsas detecciones activadas por señales no deseadas.
- Sólo depende del SNR de la señal recibida, por lo que puede verse afectado el rendimiento por las variaciones de potencia del ruido.

1.3.3. Feature Detection

Características [1]

- Este método consiste en explotar la periodicidad inherente a la señal transmitida por el PU, mediante el análisis de una función de correlación espectral, o también llamado función de auto correlación cíclica.
- Funciona muy bien en regiones con bajo SNR y con incertidumbre en el valor de la potencia del ruido.

Desventajas [1]

- Este método es críticamente dependiente de que el SU conozca de antemano el período de la señal PU modulada.
- Es computacionalmente más complejo y requiere mayor número de observaciones.

1.4. Spectrum Decision

Debido a la falta de interacción en tiempo real entre los Primary Users (PU) y los Secundary Users (SU), los métodos de detección por si solos pueden basarse en señales que en muchas ocasiones pueden llegar a ser muy débiles, lo cual generaría interferencia en el Primary Users (PU) como se muestra en la Figura1.2, o incluso fallas en la detección por parte de los SU [1].

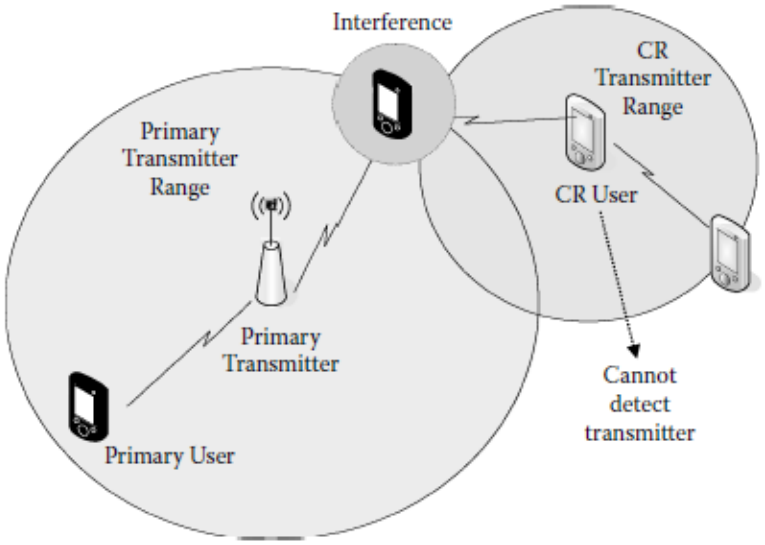


Figura 1.2: Incertidumbre del receptor SU [1]

1.4.1. Detección cooperativa

Para solucionar los problemas de interferencia o desvanecimiento se requiere detectar también la información de otros usuarios, esto es lo que se conoce como detección cooperativa. Este método es teóricamente más preciso, ya que la incertidumbre de lo que detecta un solo usuario CR puede minimizarse a través de la colaboración de otros con él, además, el desvanecimiento por trayectos múltiples y efectos de sombra puede ser mitigado de modo que la probabilidad de detección se mejora en gran medida [3].

1.4.2. Teoría de Juegos aplicada a Spectrum Decision

La teoría de juegos es una herramienta matemática que analiza múltiples interacciones estratégicas entre los participantes de un determinado entorno para la toma de decisiones. En las redes CR, los usuarios de la red, toman las decisiones sobre el uso del espectro y parámetros de operación basados en la detección de espectro dinámico y acciones adoptadas

por otros usuarios. Además, los usuarios que compiten por los recursos del espectro muchas veces no son incentivados para cooperar entre sí y pueden comportarse de forma egoísta. Por lo tanto, es recomendable el uso de la teoría de juegos para estudiar los comportamientos inteligentes e interacciones de los usuarios en una red de este tipo [2].
En la Figura1.3 se muestra algunos enfoques de teoría de juegos y sus aplicaciones.

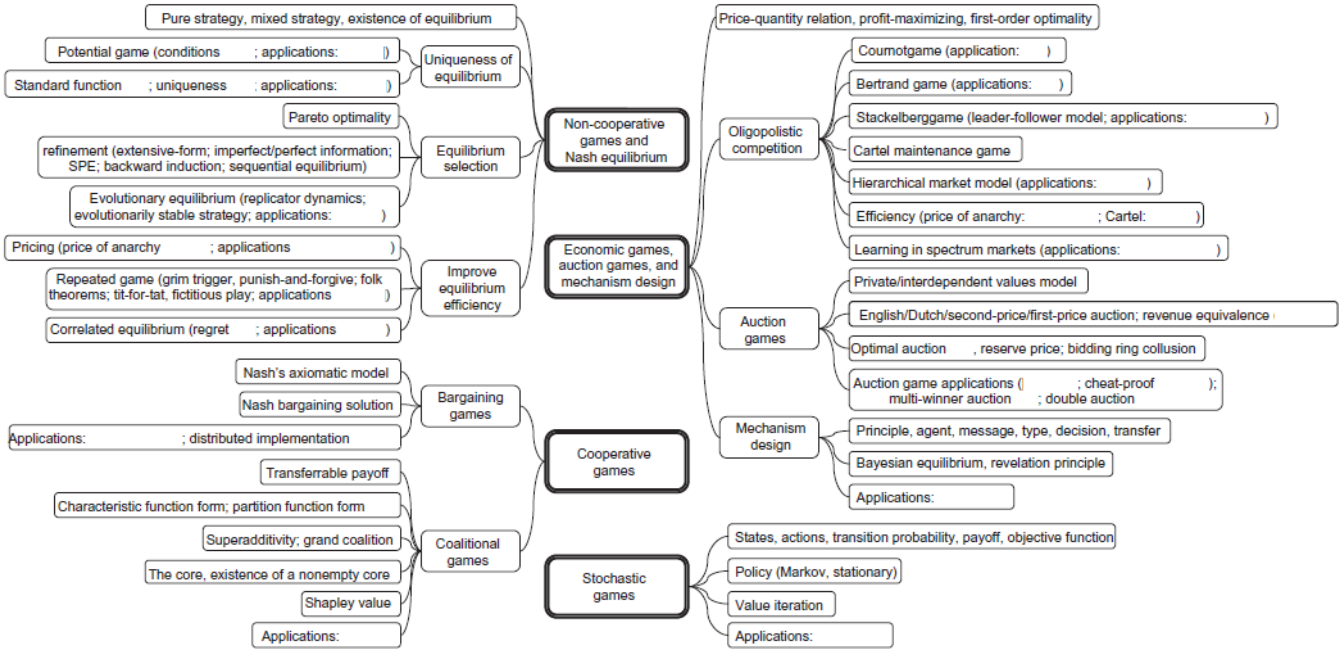


Figura 1.3: Categorías de teoría de juegos [2]

1.4.3. Juego de Coaliciones

El rendimiento de detección del espectro puede mejorar por la cooperación entre los SU que pueden formar coaliciones (grupos) y compartir resultados individuales como se muestra en la Figura1.4. Con más información de la coalición, las decisiones en la detección del espectro se pueden hacer más precisas [14].
Un centro de fusión ó cabeza de coalición desempeña un papel importante en el rendimiento de este método. Una herramienta típica para un juego de coaliciones, es la aplicación de la regla OR, similar a la función lógica del mismo nombre, ya que si un grupo contiene pocos SU, la probabilidad de que se indique que un usuario primario accede al canal, será baja [3].

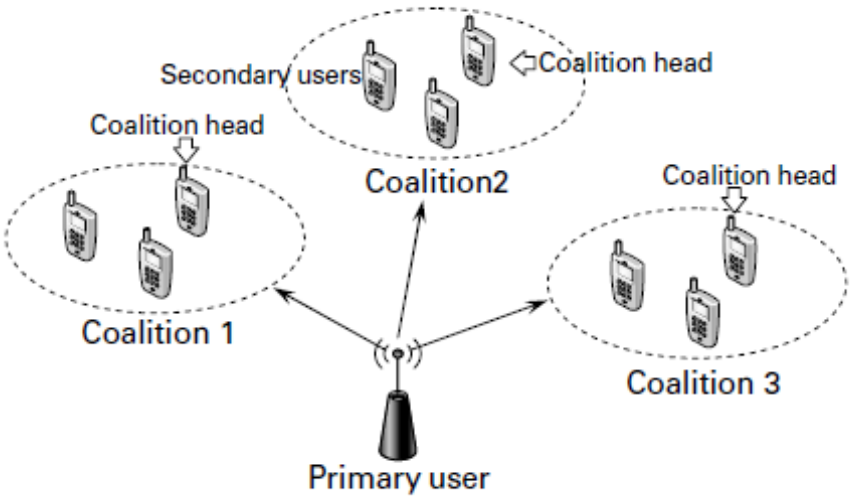


Figura 1.4: Formación de coaliciones [3]

1.5. Spectrum Sharing and Mobility

La movilidad de espectro o Handoff espectral es el proceso en donde un SU cambia su frecuencia de operación, cuando las condiciones de un canal se degradan o cuando un PU aparece ya que este es el usuario autorizado para el uso del espectro. Durante esté proceso es inevitable que se pierda la comunicación por un lapso de tiempo, por lo que se requiere un modelo de handoff que permita tener la mínima degradación de la comunicación [15].

Los modelos de handoff se pueden clasificar de acuerdo a tres criterios diferentes:

1. **De acuerdo con el momento en el que se selecciona el canal y se realiza el cambio**: estos modelos se resumen en la Tabla 1.1.

Tabla 1.1: Comparación de modelos de Handoff

Estrategia	**No Handoff**	**Reactivo puro**	**Proactivo puro**	**Híbrido**
Idea principal	Parar y esperar	Monitoreo reactivo y acción reactiva	Monitoreo proactivo y acción proactiva	Monitoreo proactivo y accion reactiva
Ventajas	Muy baja interferencia al PU	Precisión en la selección del canal	Muy rápida respuesta y selección del canal inteligente	Respuesta rápida
Desventajas	Muy alta interferencia al SU	Respuesta lenta	Datos de la disponibilidad del canal no actualizados y costo computacional alto	Datos de la disponibilidad del canal no actualizados
Latencia	Alta (impredecible)	Media	Muy baja	Baja
Dependencia	Actividad del PU	Monitoreo del espectro	Canal de backup y precisión del modelo de tráfico del PU	Canal de backup

2. **De acuerdo con el tipo de tecnología a la cual se hace el cambio de canal**: Handoff horizontal, Handoff vertical y handoff diagonal [16].

3. **De acuerdo a la movilidad en su propia tecnología**: Handoff hard y Handoff soft [16].

1.5.1. Criterios de información para realizar Handoff

Estos criterios proporcionan la información necesaria para ayudar a los algoritmos de toma de decisiones a la asignación de espectro a los SU en los sistemas CR. Estos criterios pueden cambiar de acuerdo con los objetivos a optimizar y priorizar en cada esquema de Handoff espectral [17].
Entre los principales criterios de información se encuentran calidad de servicio, calidad del enlace, disponibilidad del canal, tiempo estimado de disponibilidad del canal, patrón de tráfico del SU y PU, geo-localización y Received Signal Strength Indication (RSSI) [17].

1.6. Estándar de Cognitive Radio en la Industria: IEEE 802.22 Cognitive Radio Wireless Regional Area Network (WRAN)

También llamado el primer estándar de CR o *TV White Space*. El desarrollo del estándar IEEE 802.22 WRAN se da por el uso de técnicas de CR para permitir el uso compartido del espectro geográficamente no utilizado en las bandas de televisión, y garantizar la no interferencia en la operación de televisión digital, analógica y dispositivos de baja potencia con licencia como los micrófonos inalámbricos [4].

1.6.1. Arquitectura básica

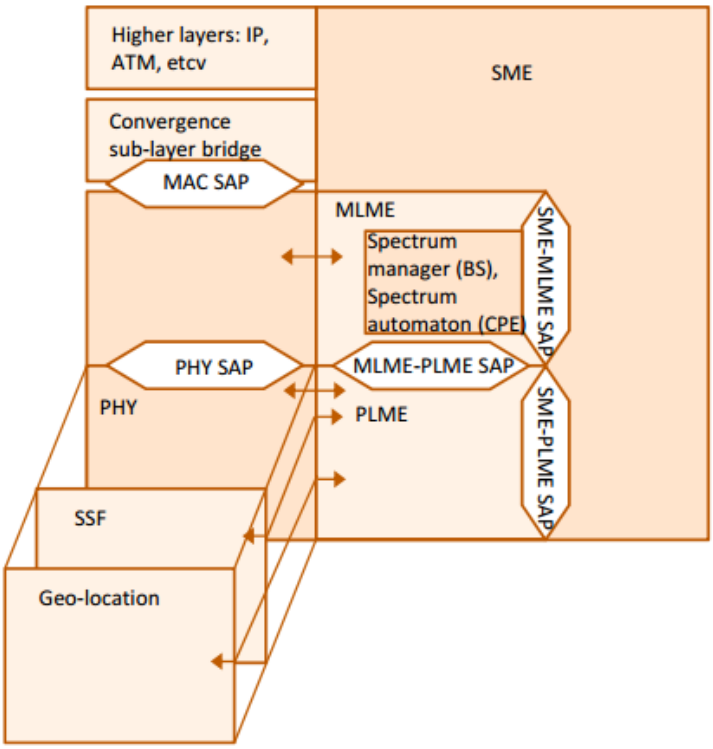


Figura 1.5: Arquitectura básica de WRAN [4]

Esta arquitectura mostrada en la Figura1.5, referencia el direccionamiento del sistema entre Physical Layer (PHY) y la capa Media Access Control (MAC), presentes en la interfaz Server Manager Entity (SME), a través de la gestión de entidades de la capa PHY y MAC denominada MAC Layer Management Entity (MLME). La comunicación con las capas superiores se da a través del estándar 802.1d que es una subcapa compatible de convergencia. En la capa PHY hay tres funciones principales: la transmisión de los datos, la función de

Spectrum Sensing Function (SSF), y la función de geo-localización. La interfaz PHY con la interfaz MAC se comunican a través del Service Access Point (SAP) [4].

1.6.2. Funciones cognitivas en WRAN

1. **Administración del espectro**: El administrador de espectro es la función cognitiva que va a utilizar las entradas a partir del SSF de la geo-localización y de la base de datos para decidir sobre el canal a utilizar por la celda WRAN [4].

2. **Geo-localización y base de datos**: La norma IEEE 802.22 requiere que todos dispositivos de la red que se instalarán tengan una ubicación fija tanto en la estación base como su Customer Premises Equipment (CPE) asociado. La ubicación de la estación base debe ser conocida en un radio de 15 m, mientras que la ubicación del CPE debe ser conocida dentro de un radio de 100 m [4].

3. **Detección del espectro**:Los requisitos de detección se resumen en función de cuatro parámetros: la sensibilidad de detección del detector, el tiempo de detección del canal, la probabilidad de detección, y probabilidad de falsa alarma. Para TV digital la sensibilidad es de -116 dBm, para TV analógica, la sensibilidad es -94 dBm, mientras que para micrófonos inalámbricos, la sensibilidad es -107 dbm [4]. El tiempo de detección de canal para todos tipos de señales es de 2 s. La probabilidad de detección es 0,9, mientras que la probabilidad de falsa alarma es 0,1 para todos los tipos de señales [4].

1.7. Desarrollo académico de CR: Módulo CR para NS-3

Existen varios trabajos académicos en función de desarrollo de simuladores para CR, entre los que se pueden mencionar Cognitive Radio Cognitive Network Simulator (CRCN), que es un simulador basado en Network Simulator 2 (NS-2), en el cual los usuarios pueden usar modelos de radio para simulaciones de redes CR y además se ha incorporado diferentes topologías y generadores de tráfico, que permiten a los usuarios crear diferentes escenarios de simulación [18].
También ha sido creado un módulo CR para NS-3 el cual tiene las siguientes características [19]:

- Proporciona capacidades CR como la detección de espectro, la detección de PU, handoff de canales y toma de decisiones en las diferentes capas de red.

- Incorpora la posibilidad de consultar una base de datos para obtener resultados de la actividad de los PU.

- Permite simular redes CR con redes legacy no cognitivas en un mismo entorno.

Las funciones típicas de un nodo CR en el simulador, cumplen las cuatro etapas básicas

de un sistema CR, como se observa en la Figura 1.6 [5]

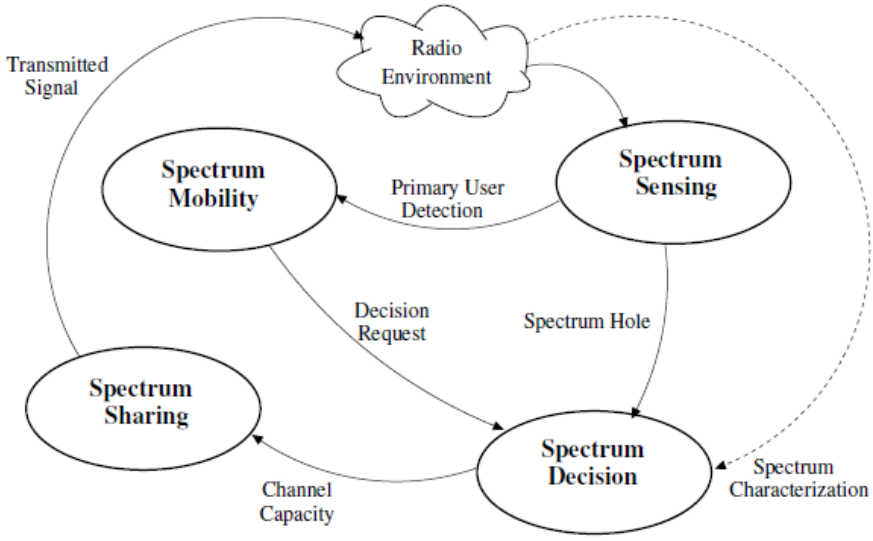


Figura 1.6: Ciclo Cognitivo de un nodo CR [5]

Dentro del administrador del espectro (Spectrum manager) en la Figura 1.7 existen submódulos que se asignan al ciclo cognitivo mostrado en la Figura1.6 entre ellos están Spectrum Sensing/Database Query, Spectrum Decision, Spectrum Mobility y Spectrum Sharing [5]

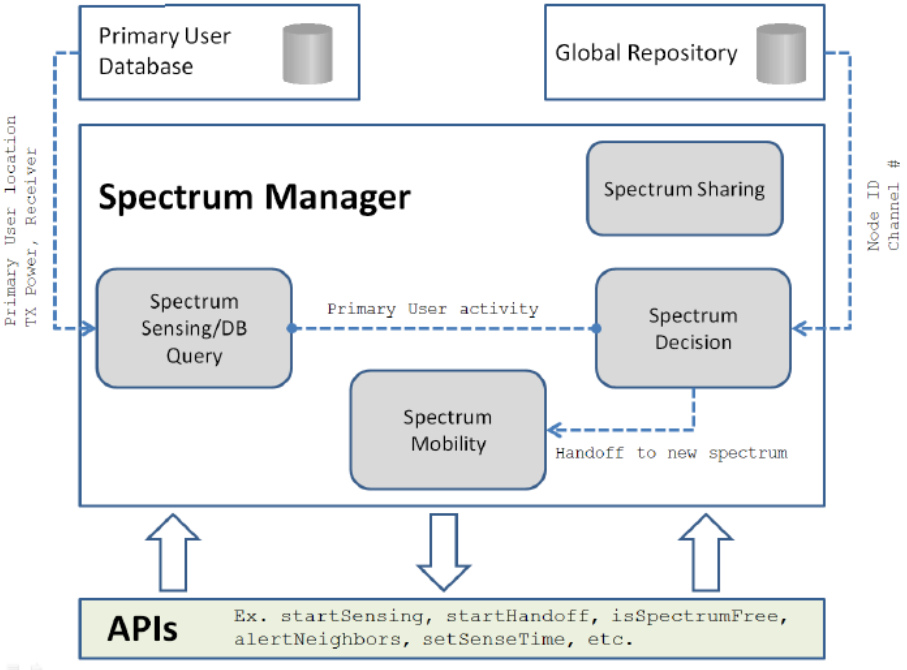


Figura 1.7: Submódulos del Simulador [5]

1.8. Maximum Likelihood Estimation (MLE) para análisis de datos

Para poder realizar un análisis estadístico de los datos recopilados de las simulaciones, y al ser estos datos modelados como funciones de probabilidad acumulada CDF, se necesita una herramienta que permita dicho análisis entre modelos, para verificar eficiencia en términos de los parámetros involucrados.
La función de verosimilitud $L(\theta)$ es la función de densidad de los datos observados, visto como una función de un parámetro desconocido θ que representa la probabilidad que tenemos de obtener una muestra de los datos observados. La estimación de máxima verosimilitud (Maximum Likelihood Estimation (MLE)) es una técnica para encontrar la función más probable que explica los datos observados, en otras palabras, es el valor del parámetro que hace que los datos obtenidos sean más probables de haber sido observados.
De acuerdo a [20] existe valores mas probables de θ por ende tienen una probabilidad relativamente alta. El valor mas probable que será el máximo valor de $L(\theta)$ es el estimador de máxima verosimitud.
Por definición el $(MLE)\hat{\theta}_{ML}$ de un parámetro θ se obtiene maximizando la función de verosimilitud:

$$\hat{\theta}_{ML} = arg\ max\ L(\theta). \tag{1.3}$$

Este estimador viene asociado con un error que es posible determinar dado que todos los errores estándar en MLE son asintóticos y si existe un buen estimador de una muestra pequeña, se verá como máxima verosimilitud [21].
La métrica que se usa para corroborar la variabilidad de los datos, es la desviación estándar del estimador. Si se está interesado en la precisión de los datos o en comparar y probar las diferencias entre algunos parámetros, entonces el error estándar del estimador es la métrica [22].

Capítulo 2

Metodología

Para lograr los objetivos planteados, la metodología usada fue de investigación, desarrollo, simulación y análisis que se explicarán en este capítulo y posteriores. Mediante la investigación de textos científicos (papers), se pudo escoger métodos basados en algoritmos matemáticos y de teoría de juegos para cada etapa del sistema CR. En el presente capítulo se realiza una descripción a detalle de los métodos y algoritmos usados en este trabajo.
El flujo de la metodología investigada (Figura 2.1), es el siguiente y sigue al flujo de un sistema CR:

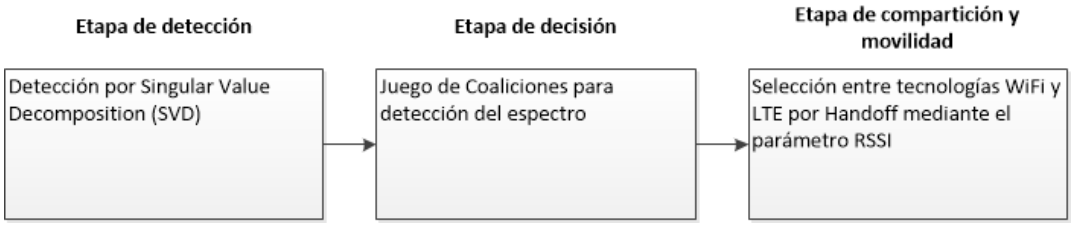


Figura 2.1: Diagrama de flujo para la metodología usada

2.1. Etapa de detección: Detección por Singular Value Decomposition (SVD) para redes CR

De acuerdo a lo revisado en el Capítulo 1, existen tres principales tipos de detección de PU:

- Matched filter detection o Coherent detection [23]
- Energy detection [24] [23]
- Feature detection [23]

De estos métodos, el de Energy detection es el más usado por su facilidad de implementación, siendo el más básico y de mayor uso para su comparación con otros métodos mas sofisticados [24].
A diferencia de los otros métodos, el Energy detection no necesita ninguna información de la señal que se va a detectar y es más robusto a la dispersión del canal desconocido [24]. A pesar de esta ventaja, el Energy Detection es vulnerable a la incertidumbre del ruido porque el método se basa en saber con precisión la Potencia del ruido. En la práctica, es muy difícil obtener esta potencia de manera precisa [24].
Para superar esta desventaja, para este trabajo se plantea y adapta el método Singular Value Decomposition (SVD) a una red CR móvil [25]. Este método está basado en los valores propios o *Eigenvalores*, además de que el umbral de decisión que determina la presencia de la señal PU se deriva de Random Matrix Theory (RMT) [26]. Esto se realiza para determinar la hipótesis de la detección de la señal.
El método SVD es muy similar al método de descomposición por valores propios. Sin embargo, SVD es más general ya que puede aplicarse a cualquier tipo de matriz sin necesidad de ser una matriz cuadrada [25].

Ventajas [25]

- Es más robusto al error numérico
- La estructura geométrica de una matriz la toma como un aspecto importante de muchos cálculos matriciales
- Cuantifica el cambio resultante entre la geometría subyacente de esos espacios vectoriales.

2.1.1. Modelo matemático del algoritmo de detección SVD implementado

1. Seleccionar los valores de N, L y k, tales que $k < L < N - k$, donde N es el número de muestras que debe tomar el receptor CR, L se define como el 'Factor de suavizado' es decir el número de valores consecutivos que la matriz de covarianza de la señal recibida puede tomar, y k es el número de valores singulares. Siguiendo la notación dada en [25], se consideró $k = 2$ y $L = 16$. Entonces la matriz de covarianzas viene dada por la siguiente expresión:

$$R(N) = \frac{1}{N} \sum_{L=16}^{16-1+N} \hat{x}(n) x^{\dagger}(n), \tag{2.1}$$

   Donde $R(N)$ es la matriz de covarianza de la señal discreta $\hat{x}(n)$ recibida por el receptor CR y $x^{\dagger}(n)$ es el *Hermitiano* de $\hat{x}(n)$.

2. Factorizar la matriz de covarianzas, es decir aplicar el método SVD para obtener los valores singulares, obteniendo la siguiente expresión:

$$R = U\Sigma V^t, \tag{2.2}$$

donde $R = (mxn)$ es la matriz de covarianza, $U = (mxn)$ es la matriz de vectores singulares de columnas de R, $\Sigma = (mxn)$ es la matriz de valores singulares, y $V^t(mxn)$ es la matriz de vectores singulares de filas R.

3. Obtener los valores propios máximos y mínimos de la matriz de covarianza R(N_s), siendo estos λ_{max} and λ_{min} respectivamente.

4. Calcular el valor umbral 'threshold' para comparar con los valores singulares usando la siguiente expresión:

$$\gamma = \frac{(\sqrt{N}+\sqrt{L})^2}{(\sqrt{N}-\sqrt{L})^2} * (1 + (\frac{(\sqrt{N}+\sqrt{L})^{-\frac{2}{3}}}{N * L^{\frac{1}{6}}}) * F_1^{-1}(1 - P_{fa})), \tag{2.3}$$

donde P_{fa} es la probabilidad de falsa alarma que se requiere que sea igual o menor que 0.1 (P_{fa} <= 0,1) [25]. La expresión $F_1^-1(1 - P_{fa})$ es la función de $Tracy - Widom$, que es la distribución de probabilidad normalizada para los valores singulares, como se muestra en la Tabla 2.1 [27]:

Tabla 2.1: Función Tracy-Widom

t	-3.9	-3.18	-2.78	-1.91	-1.27	-0.59	0.45	0.98
$F_1(t)$	0.01	0.05	0.10	0.30	0.50	0.70	0.90	0.95

5. Comparar la relación de valores singulares máximos y mínimos de la matriz de covarianzas con el umbral, por lo tanto si $\frac{\lambda_{max}}{\lambda_{min}} < \gamma$, la señal PU está presente, de otro modo la señal no está presente.

6. Obtener el bit de detección de señal individual de PU.

El enfoque anterior se resume en Algoritmo I.

Algorithm 1 Algoritmo de detección SVD aplicado

Require: k, L, N y $k < L < N - k$.
Ensure: *bit* de d*eteccion* de PU.

while comunicación este en progreso **do**
  Inicializar $k = 2$ y $L = 16$
  Obtener $CovMat = CrearMatrizCovarianza(L)$
  Factorizar(CovMat)
  Obtaner $Max = Maximo(CovMat)$ y $Min = Minimo(CovMat)$
  Obtaner $Threshol\text{d} = Calcularumbral()$
  if $(Max/Min < Threshol\text{d})$ **then**
    return *bit* de d*eteccion* de PU=1
  else
    return *bit* de d*eteccion* de PU=0
  end if
end while

2.2. Etapa de decisión: Juego de Coaliciones para detección del espectro en CR

En esta etapa de decisión es donde se implementó la teoría de juegos basada en coaliciones (juego cooperativo) [28] [29].
De todas los algoritmos y métodos analizados previamente, se decidió usar un tipo de detección de espectro colaborativo entre los SU debido a que gracias a estudios previos [30], se muestra una significativa mejora del rendimiento, aunque se debe tomar en cuenta que existe un $tradeoff$ entre las ganancias en términos de probabilidad de detección del PU y los costes en términos de probabilidad de falsa alarma. Es decir aumentará la probabilidad de detección de un PU, pero también aumentará la probabilidad de falsa alarma.

2.2.1. Modelo matemático del algoritmo de Juego de Coaliciones implementado

1. **FASE 1: Detección local [28] [29]**

   - Cada SU individual obtendrá su bit de detección de señal de PU.

2. **FASE 2: Formación de coaliciones adaptativas [28] [29]**

   - Durante la formación de la coalición adaptativa se asume que cualquier SU decide de manera autónoma formar o romper una coalición para maximizar su utilidad en términos de la probabilidad de detección, mientras contabiliza un costo de falsa alarma, siendo modelado un juego de coalición (N, v) donde N es el conjunto de jugadores (SU) y v es la función de utilidad o valor de una coalición.

- Dentro de una coalición, el SU que tenga la mayor probabilidad de detección se escogerá como cabeza de coalición.

- Las coaliciones y los SU deben seguir las siguientes propiedades:

  – En el juego de coaliciones propuesto, la utilidad de una coalición S es igual a la utilidad de cada SU en la coalición.

  – Las probabilidades de falsa alarma y de no detección de un PU por parte de cualquier SU que pertenece a la coalición, están dadas por las probabilidades de la coalición.

- La comparación se realiza basada en la siguiente expresión que representa la función de utilidad de cada SU:

$$v\left(S\right) = Q_{\mathrm{d},S} - C(Q_{f,S}) = \ (1 - Q_{m,S}) - C(Q_{f,S}), \tag{2.4}$$

Donde $C(Q_{f,S})$ es la función de costo de probabilidad de falsa alarma dentro de la coalición S.
$Q_{f,S}$ es la probabilidad de falsa alarma de la coalición S, teniendo como cabeza de coalición k, y esta dada por la siguiente expresión:

$$Q_{f,S} = 1 - \prod_{\mathrm{i}\in S} \left[\left(1 - P_f\right)\left(1 - P_{\mathrm{e,i},k}\right) + P_f * P_{\mathrm{e,i},k}\right], \tag{2.5}$$

Donde P_f es la probabilidad de falsa alarma de un SU y $P_{\mathrm{e,i},k}$ es la probabilidad de reportar un error entre un SU de la coalición y su cabeza de coalición k.
$Q_{m,S}$ es la probabilidad de no detección de un PU por parte de la coalición, y esta dada por la siguiente expresión:

$$Q_{m,S} = \prod_{\mathrm{i}\in S} \left[\left(P_{m,\mathrm{i}}\right)\left(1 - P_{\mathrm{e,i},k}\right) + \left(1 - P_{m,\mathrm{i}}\right)P_{\mathrm{e,i},k}\right], \tag{2.6}$$

Donde $P_{m,\mathrm{i}}$ es la probabilidad de no detección de un PU por parte de un SU.
La función costo de probabilidad de falsa alarma $C(Q_{f,S})$, dentro de la coalición S, está dada por la siguiente expresión:

$$C(Q_{f,S}) = \begin{cases} -\alpha^2.log(1 - \left(\left(\frac{Q_{f,S}}{\alpha}\right)^2\right) & \text{si } Q_{f,S} < \alpha \\ +\infty & \text{si } Q_{f,S} \geq \alpha \end{cases} \tag{2.7}$$

- Se forman las coaliciones basadas en las siguiente reglas de unión y separación de SU:

– **Unión:** la coalición decide unirse de acuerdo al siguiente teorema:

Se decide unir cualquier set de coaliciones si la función de utilidad de la unión es mejor comparada con cada coalición individual, además si el set de coaliciones abarca todos los usuarios de la partición, y por Pareto, es preferible, dada su función de utilidad, comparada con las particiones no coalicionadas.

– **Separación:** la coalición decide separarse de acuerdo al siguiente teorema:

Se decide separar un set de coaliciones si la función de utilidad de cada coalición del set por individual es mejor que la unión de las coaliciones.

- Se realiza esta metodología hasta encontrar el óptimo conjunto de coaliciones en los elementos de la red, mientras la red se encuentre en funcionamiento.

- **Restricciones de coalición:** La coalición mínima a formar será de un SU por colación (es decir cuando no forma parte de ninguna coalición). El número máximo de elementos de una coalición esta dada por la siguiente restricción:

$$M_{max} = \frac{log\,(1-\alpha)}{log\,(1-P_f)} \tag{2.8}$$

3. **FASE 3: Detección de la coalición [28] [29]**

- Cada SU reporta su bit de detección a cada cabeza de la coalición.

- La cabeza de cada coalición toma una decisión final sobre la presencia o no de un PU usando una regla OR.

- Los SU en una coalición cumplirán con la decisión final tomada por la cabeza de la coalición.

2.3. Etapa de compartición y movilidad: Selección de tecnología por Handoff mediante el parámetro RSSI

Debido a que la tecnología LTE [31] es compatible con las especificaciones 3rd Generation Partnership Project (3GPP) LTE [32] y la tecnología WiFi [33] es compatible con las especificaciones IEEE 802.11 [34], y dado que el trabajo se basó exclusivamente en métodos y análisis de la capa PHY CR, se investigó y analizó un parámetro físico característico común en ambas tecnologías, para poder considerarlo como parámetro de decisión para la movilidad de CR entre tecnologías [35].

Este parámetro físico es el Received Signal Strength Indication (RSSI), dado que provee información acerca del nivel de potencia que está siendo recibido por la antena, el cual

decrementa cuando el usuario se aleja del actual punto de acceso a la red. Este criterio permite determinar el momento en el que se hace necesario realizar un cambio de canal [36].

El esquema seguido para realizar el Handoff se muestra en la Figura 2.2 :

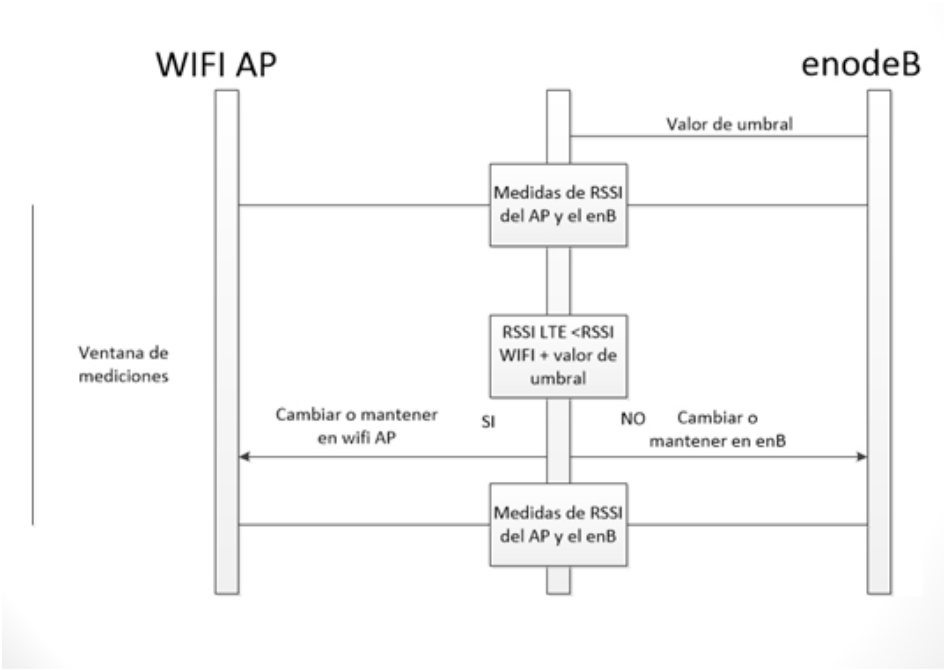


Figura 2.2: Esquema de Handoff entre tecnologías

El valor de umbral de RSSI sirve para aumentar/disminuir el área de cobertura efectiva del Access Point (AP) WiFi dependiendo de la capacidad de la red [37].Una limitación de este método es que el valor del umbral debe adaptarse en función del conocimiento de toda la red y de la distribución de los usuarios.
Para efectos experimentales y para poder comparar cuando un SU debe cambiar de tecnología, dependiendo de su movilidad, se usó un valor umbral, de acuerdo a lo investigado está dado en 1.76 dB [35].
La medición del RSSI más el valor de umbral de acuerdo a la potencia recibida sea de WiFi o LTE, será comparada mediante histéresis para disminuir el número de conmutaciones de frontera de celda [38].
Por lo tanto los valores que usa un SU que se encuentra transmitiendo en una banda de la tecnología WiFi para cambiarse a la tecnología LTE y viceversa son los siguientes (Tabla 2.2:

Tabla 2.2: Medidas de RSSI para cambio de tecnología

Cambio entre tecnologías	**Valores**
$WiFi\ a\ LTE$	RSSI + umbral = 37 dB
$LTE\ a\ WiFi$	RSSI + umbral = 34 dB

El diagrama de flujo de interacción entre los métodos y algoritmos implementados se

muestra en la Figura 2.3

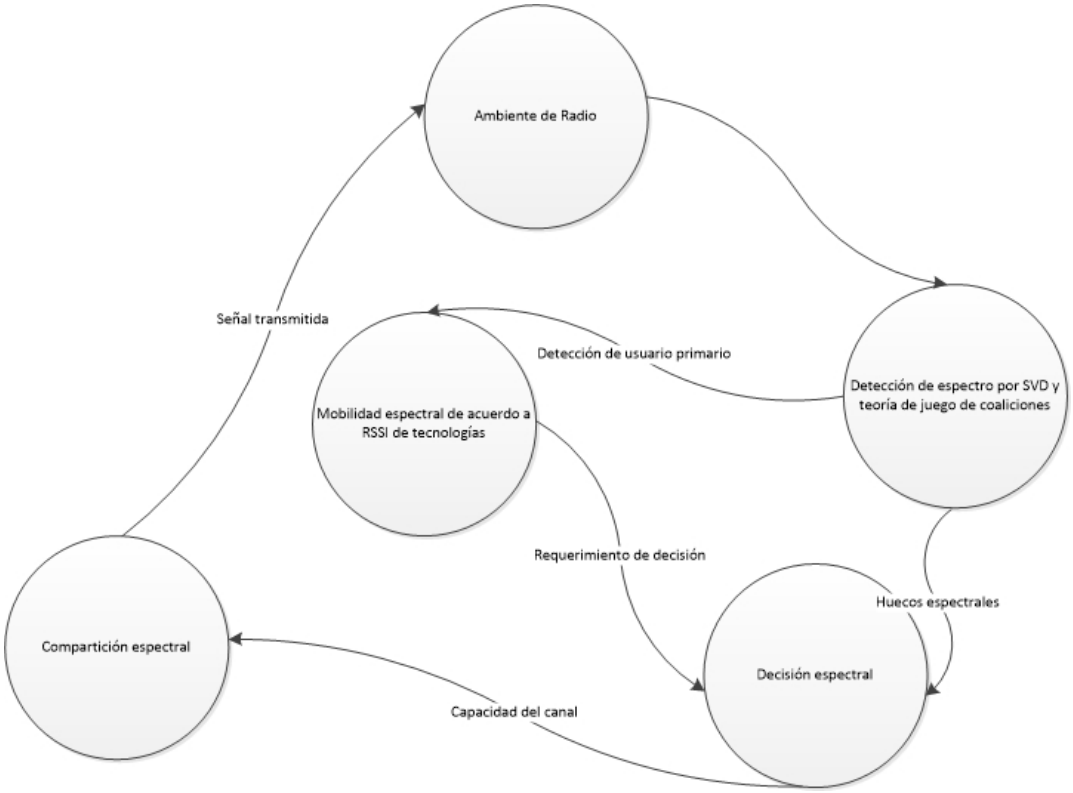


Figura 2.3: Diagrama de interacción de etapas CR

Capítulo 3

Desarrollo e Implementación

La implementación de los algoritmos y métodos escogidos en el Capítulo 2, se la realizó mediante un desarrollo para Network Simulator 3 (NS-3) [39], y su resultado, fue un simulador CR para las redes móviles LTE y WiFi.

Para la elaboración de este simulador, y para seguir las mejores prácticas de desarrollo de software [40], se realizó un documento técnico en el cual constan todos los parámetros a tomar en cuanta para la creación del simulador, y su apreciación se encuentra en el Anexo B. El desarrollo se encuentra estructurado en 5 componentes principales:

- Módulo *Cognitive*
- Módulo LTE
- Módulo $WiFi$
- Programa principal para generación de simulación $LTE - WIFI.CC$
- Script para inicialización de parámetros $Lte - Wifi.sh$

3.1. Módulo *Cognitive*

Contiene los desarrollos $svd.cc$ y $svd.h$, que son los que permiten realizar la etapa de detección del ciclo CR debido a que en estos desarrollos está implementado el método de detección SVD especificado en el Capítulo 2 y en el Anexo B. La Tabla 3.1 lista todos los archivos del módulo cognitivo.

Tabla 3.1: Archivos en Módulo *Cognitive*

Archivos
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/cognitive.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/cognitive.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/cognitive-packet-tags.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/cognitive-packet-tags.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/common-cognitive-header.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/pu-model.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/pu-model.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/repository.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/repository.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-data.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-data.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-decision.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-decision.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-sensing.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-sensing.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/svd.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/svd.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/wscript

3.2. Módulo *LTE*

Contiene todos los parámetros principales de la tecnología LTE, adaptada a las funciones cognitivas del desarrollo. Entre los archivos principales se encuentran *coalition.cc* y *coalition.h* que son los desarrollos que contienen el algoritmo de teoría de juego de coaliciones encargado de la decisión de uso del espectro, especificado en el Capítulo 2 y en el Anexo B. La Tabla 3.2 lista todos los archivos del módulo LTE.

Tabla 3.2: Archivos en Módulo *LTE*

Archivos
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/a2-a4-rsrq-handover-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/a2-a4-rsrq-handover-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/a3-rsrp-handover-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/a3-rsrp-handover-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/coalition.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/coalition.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-rrc-sap.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-rrc-sap.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-sap.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-sap.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-soft-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-soft-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-hard-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-hard-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-soft-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-soft-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-strict-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-strict-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-handover-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-handover-algorithm.h

3.3. Módulo *WiFi*

Contiene todos los parámetros principales de la tecnología WiFi, adaptada a las funciones cognitivas del desarrollo, listados en la Tabla 3.3.

Tabla 3.3: Archivos en Módulo $Wi Fi$

Archivos
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/aarfcd-wifi-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/aarfcd-wifi-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ap-wifi-mac.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ap-wifi-mac.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/arf-wifi-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/arf-wifi-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ctrl-headers.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ctrl-headers.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/dcf-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/dcf-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/interference-helper.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/interference-helper.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/snr-tag.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/snr-tag.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ssid.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ssid.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-channel.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-channel.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-mac.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-mac.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-net-device.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-net-device.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-phy.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-phy.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-remote-station-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-remote-station-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/yans-wifi-channel.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/yans-wifi-channel.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/yans-wifi-phy.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/yans-wifi-phy.h

3.4. Programa principal para generación de simulación $LTE - WIFI.CC$

Código principal donde se encuentran estructurados todos los parámetros técnicos básicos de la simulación detallados a continuación:
En esta sección del código se inicializa todos los parámetros básicos del escenario de simulación, los cuales pueden ser modificados directamente en este archivo (LTE-WIFI.CC), o en el archivo $Lte - Wifi.sh$. El código del archivo LTE-WIFI.CC es el siguiente:

```
/*
 * Main Function
 */
int main(int argc, char *argv[]) {
  uint16_t numberOfDualNodes = 10;// Number of dual nodes
  uint16_t numberOfEUL = 5; // Number of Primary User LTE
  uint16_t numberOfCUL = 5; // Number of User CR LTE
  uint16_t numberOfEUW = 5; // Number of Primary User WIFI
  uint16_t numberOfCUW = 5; // Number of User CR WIFI

  // The number of dual nodes will be 10
  uint16_t numberOfeNBNodes = 1;  // Number of eNB node for LTE
  uint16_t numberOfapNodes = 1; // Number of AP node for WIFI
  double simTime = 5;   // Simulating Time
  double distance = 200.0;    // Distance between eNB and AP
  double interPacketInterval = 100; // Interval time for sending
     packet
  int nodeSpeed = 100;      // Speed of Nodes in m/s
  int nodePause = 0;      // Pause time in sec
  uint8_t bandwidth = 25;     // BandWidth of Network
  uint8_t upBW = 25;      // Uplink BandWidth
  uint8_t downBW = 25;      // Downlink BandWidth
  bool generateSpectrumTrace = false;
  double eNBX, eNBY, eNBZ;    // position of eNB node for LTE
  double ApX, ApY, ApZ;     // position of AP node for WIFI
  eNBX = eNBY = eNBZ = 100.0;
  ApX = ApY = ApZ = 100.0;

  double tx = 0.37;   // transmisstion data power
  double rx = 0.6;    // received data power
  double noise = 1e-15;   // Noise is the -120dBm
  string trafficName, traffic;

  uint32_t wifiFrequency = 2.4;
  uint32_t lteFrequency = 100;
  double ApRange = 250.0;
  double EnbRange = 500.0;
  time_t startTd, endTd;

```

```
39   int numberOfSample = 16000;
40
41   char dtbuf[256] = { 0 };
42   FILE* timeFp = fopen("simulating-time.txt", "a+");
43   time(&startTd);
44         strftime(dtbuf, sizeof dtbuf, "%A %b %d %H:%M:%S %Y",
           localtime(&startTd));
45   fprintf(timeFp, "StartTime: %s\n", dtbuf);
46
47   SeedManager::SetSeed((unsigned int)time(NULL));
```

En esta sección del código se da la opción de configuración de estos parámetros en un script para llamado inmediato del programa, que será el $Lte - Wifi.sh$.

```
1  // Command line arguments
2    CommandLine cmd;
3    cmd.AddValue("ENB", "Number of ENB", numberOfeNBNodes);
4    cmd.AddValue("UserCrWifi", "Number of UserCrWifi", numberOfCUW);
5    cmd.AddValue("UserCrLte", "Number of UserCrLte", numberOfCUL);
6    cmd.AddValue("DualUserCr", "Number of DualUserCr",
       numberOfDualNodes);
7    cmd.AddValue("PrimaryUserWifi", "Number of PrimaryUserWifi",
       numberOfEUW);
8    cmd.AddValue("PrimaryUserLte", "Number of PrimaryUserLte",
       numberOfEUL);
9
10   cmd.AddValue("simTime", "Total duration of the simulation [s])",
       simTime);
11   cmd.AddValue("NumberOfSample", "Number of the sample for SVD
       Matrix)", numberOfSample);
12   cmd.AddValue("trafficName", "Traffic Name of Network (TCP or
       UDP)", traffic);
13   cmd.AddValue("eNBX", "X position of eNB node", eNBX);
14   cmd.AddValue("eNBY", "Y position of eNB node", eNBY);
15   cmd.AddValue("eNBZ", "Z position of eNB node", eNBZ);
16   cmd.AddValue("LteUpBandWidth", "Upload bandwidth in LTE", upBW);
17   cmd.AddValue("LteDownBandWidth", "Download bandwidth in LTE",
       downBW);
18
19   cmd.AddValue("ApX", "X position of AP node", ApX);
20   cmd.AddValue("ApY", "Y position of AP node", ApY);
21   cmd.AddValue("ApZ", "Z position of AP node", ApZ);
22   cmd.AddValue("WifiBandWidth", "BandWidth in WIFI", bandwidth);
23
24   cmd.AddValue("TX", "Transmission data power", tx);
25   cmd.AddValue("RX", "Received data power", rx);
26   cmd.AddValue("Noise", "Noise of Network", noise);
```

```
27
28    cmd.AddValue("WifiFrequency", "Wifi Frequency (GHz)",
          wifiFrequency);
29    cmd.AddValue("LteFrequency", "Lte Frequency (MHz)", lteFrequency);
30    cmd.AddValue("ApRange", "Ap cover area (m)", ApRange);
31    cmd.AddValue("EnbRange", "Enb cover area (m)", EnbRange);
32
33    cmd.Parse(argc, argv);
34    trafficName = "udp";
```

En las siguientes secciones del código principal se inicializan y configuran capa por capa los parámetros técnicos básicos de los nodos y usuarios WiFi y LTE. Todo esto se encuentra en el Anexo C

3.5. Script para inicialización de parámetros $Lte-Wifi.sh$

Este script permite modificar los parámetros técnicos básicos de simulación según lo mencionado en el Anexo A. Además para iniciar la simulación en el terminal se debe ejecutar este script, como se indica en el Anexo B. En esta sección del código se da la opción de configuración de estos parámetros en un script para llamado inmediato del programa, que será el $Lte - Wifi.sh$.

```
1  #!/bin/bash
2  #-----------------------------------------------------#
3  #      Run the LTE-WIFI project using this Script
4  # NOTE:
5  # Please set all parameters correctly!
6  #-----------------------------------------------------#
7  rm -f *.txt
8  ./waf --run "LTE-WIFI --ENB=3 --UserCrLte=5 --UserCrWifi=5
       --DualUserCr=10 --PrimaryUserLte=5 --PrimaryUserWifi=5
       --simTime=1800 --NumberOfSample=16000 --trafficName=tcp
       --LteFrequency=729 --WifiFrequency=2.4 --LteUpBandWidth=20.0
       --LteDownBandWidth=20.0 --WifiBandWidth=20.0 --TX=0.037 --RX=0.06
       --Noise=-15 --ApX=100.0 --ApY=200.0 --ApZ=0.0 --eNBX=100.0
       --eNBY=100.0 --eNBZ=0.0 --ApRange=200 --EnbRange=350"
```

La Tabla 3.4 muestra los parámetros técnicos que pueden ser cambiados con sus respectivos rangos permitidos por el desarrollo y por NS-3, de acuerdo al Anexo A:

Tabla 3.4: Tabla con umbrales de parámetros modificables en el desarrollo

Parámetros	Mínimo	Máximo
Frecuencia LTE	729 Mhz	2170 Mhz
Frecuencia WiFi	2400 Mhz	5875 Mhz
Celdas eNB	1	3
Ancho de Banda LTE	10 Mhz	60 Mhz
Ancho de Banda WiFi	20 Mhz	150 Mhz
Ruido	0 dBm	-250 dBm
Potencia de transmisión	0.0025 mW	1.27 mW
Potencia de recepción	0.0025 mW	1.27 mW
SU LTE	0	100
SU WiFi	0	100
SU duales	0	100
PU LTE	0	100
PU WiFi	0	100
Muestras para detección	1000	30000
Rango de Cobertura AP	200 m	500 m
Rango de Cobertura eNB	350 m	5000 m
Tiempo de simulación	0	120 $horas$

3.6. Salidas del simulador CR

3.6.1. Archivos de salida de la tecnología LTE

Todos los datos sobre la tecnología LTE que resultan de la simulación se encuentran en la siguiente tabla (Tabla 3.5):

Tabla 3.5: Archivos de salida de la tecnología LTE

Archivos
/home/ns-allinone-3.23/ns-3.23/DlPdcpStats.txt
/home/ns-allinone-3.23/ns-3.23/DlRlcStats.txt
/home/ns-allinone-3.23/ns-3.23/DlRsrpSinrStats.txt
/home/ns-allinone-3.23/ns-3.23/DlRxPhyStats.txt
/home/ns-allinone-3.23/ns-3.23/DlPdcpStats.txt
/home/ns-allinone-3.23/ns-3.23/DlPdcpStats.txt
/home/ns-allinone-3.23/ns-3.23/DlTxPhyStats.txt
/home/ns-allinone-3.23/ns-3.23/UlPdcpStats.txt
/home/ns-allinone-3.23/ns-3.23/UlRlcStats.txt
/home/ns-allinone-3.23/ns-3.23/UlRsrpSinrStats.txt
/home/ns-allinone-3.23/ns-3.23/UlRxPhyStats.txt
/home/ns-allinone-3.23/ns-3.23/UlTxPhyStats.txt

3.6.2. Archivos de promedios de datos ip para la tecnología WiFi

Los datos promedios de Jitter, Packet loss, Delay y Througput de la tecnología WiFi se encuentran en el archivo */home/ns-allinone-3.23/ns-3.23/wifi-network-perf.txt*
Throughput (kbps): 30.376
DelayTime (sec): 0.019
JitterTime (sec): 0.003
PacketLoss: 52.357

3.6.3. Archivo con datos de la etapa de detección con el método SVD

Los datos resultantes de la etapa de detección mediante el algoritmo SVD, según el tipo de SU se encuentran en el archivo */home/ns-allinone-3.23/ns-3.23/svd-result.txt*, que tiene la estructura mostrada en la siguiente tabla (Tabla 3.6):

Tabla 3.6: Estructura del archivo de salida de la etapa de detección

Tiempo	**Tipo de usuario**	**Detección (0 no detecto, 1 detecto)**	**Usuario primario**

3.6.4. Archivo con datos de la etapa de decisión con la teoría de juego de coaliciones

Los datos resultantes de la etapa de detección mediante el algoritmo de la teoría de juego de coaliciones se encuentra en el archivo */home/ns-allinone-3.23/ns-3.23/coalition-result.txt*, que tiene la estructura mostrada en la siguiente tabla (Tabla 3.7):

Tabla 3.7: Estructura del archivo de salida de la etapa de detección

Tiempo	**Coalición**	**Cabeza de coalición**	**Decisión de detección de la coalición**	**Tecnología**

3.6.5. Archivo con datos de las etapas de compartición y movilidad con handoff

Los datos resultantes de las etapas de compartición y movilidad mediante el handoff entre tecnología LTE y WiFi o viceversa se encuentran en el archivo */home/ns-allinone-3.23/ns-3.23/handoff-result.txt*, que tiene la estructura mostrada en la siguiente tabla (Tabla 3.8):

Tabla 3.8: Estructura del archivo de salida de la etapa de compartición y movilidad

Tiempo	**Usuario que realiza el handoff**	**Coalición a la que pertenece**	**Desde, valor de RSSI**	**Hacia, valor de RSSI**	**Canal anterior**	**Canal actual**

3.6.6. Interacción de los módulos desarrollados en NS-3

Los módulos desarrollados en NS-3 interactúan entre si en un entorno cognitivo, como se muestran en la Figura 3.1

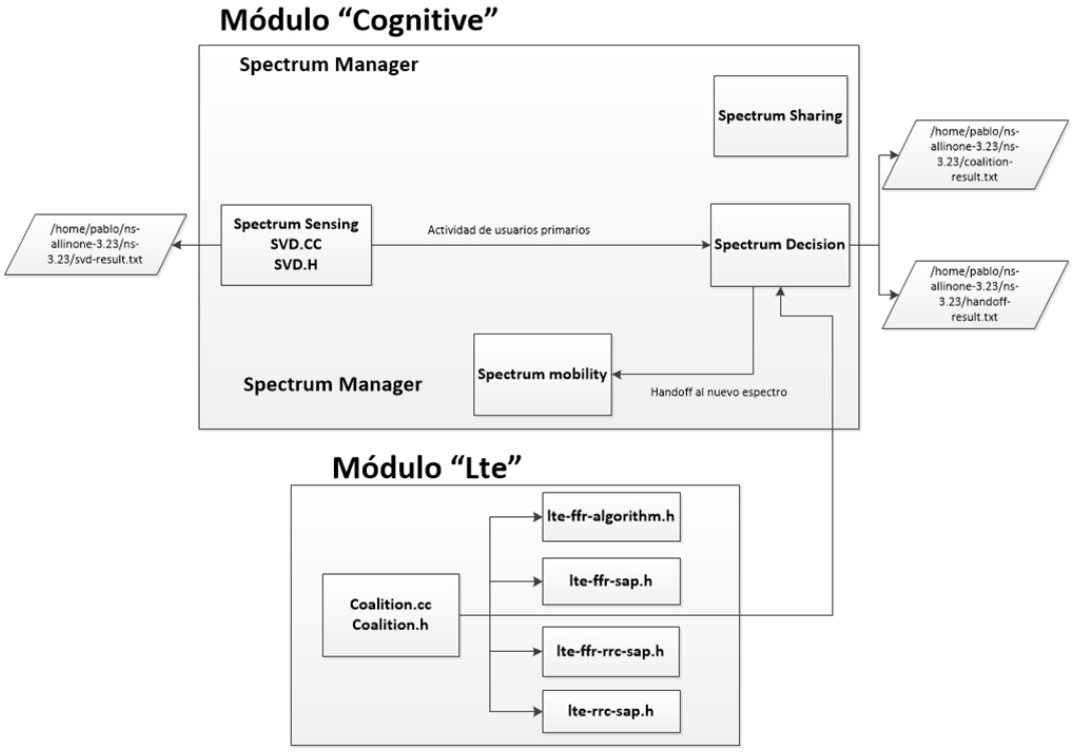


Figura 3.1: Diagrama de interacción los módulos desarrollados

Capítulo 4

Análisis de Datos y Resultados

En este capítulo se realiza inicialmente el diseño de los experimentos para la obtención de datos que permitirán realizar un análisis matemático y estadístico de los mismos y su posterior resultado, gracias al simulador CR desarrollado en el Capítulo 3. Debido a la gran extensión de tipos de experimentación que ofrece el simulador, se tuvo que acotar el análisis a una de las etapas de una red CR la etapa de detección, para poder cumplir con el Objetivo General planteado.

Para esto se sigue el siguiente flujo de trabajo (Figura 4.1), que será descrito en este Capítulo:

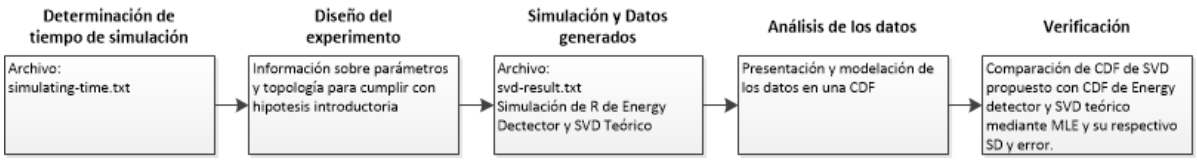


Figura 4.1: Diagrama de flujo para Análisis de datos

4.1. Determinación de tiempo de simulación

Antes de realizar los experimentos, debemos tomar en cuenta un factor importante para el simulador, que el tiempo de simulación no es el mismo que el tiempo real. Para este fin, se realizaron varias simulaciones con distintos tiempos de simulación, manteniendo los parámetros técnicos básicos sin modificar los cuales se indican en la Tabla 4.1, para poder observar el comportamiento del tiempo real [41].

Tabla 4.1: Valores fijos para determinación de tiempo de simulación

Parámetros	**Valor**
Frecuencia LTE	729 Mhz
Frecuencia WiFi	2400 Mhz
Celdas eNB	3
Ancho de Banda LTE	20 Mhz
Ancho de Banda WiFi	20 Mhz
Ruido	-15 dBm
Potencia de transmisión	0.037 mW
Potencia de recepción	0.06 mW
SU LTE	5
SU WiFi	5
SU duales	10
PU LTE	5
PU WiFi	5
Muestras para detección	16000
Rango de Cobertura AP	200 m
Rango de Cobertura eNB	350 m
Tipo de tráfico	TCP
Tiempo de simulación	Variable

Los valores de potencias, anchos de banda, ruido y frecuencias fueron elegidos para hacer la simulación del modelo más real porque estos son valores típicos de despliegue comercial. Teniendo estos parámetros configurados, se obtuvieron los siguientes resultados en cuanto a tiempos reales (Tabla 4.2):

Tabla 4.2: Tiempos de simulación vs Tiempos reales

Tiempo de simulación (seg)	**Tiempo real (min)**
5	2.25
10	5.98
15	10.40
30	16.78
60	34.80
90	53.88
180	130.82
300	225.92
450	382.77
600	429.09
900	732.23
1800	1639.28

Observamos el comportamiento lineal y creciente del tiempo real vs tiempo de simulación:

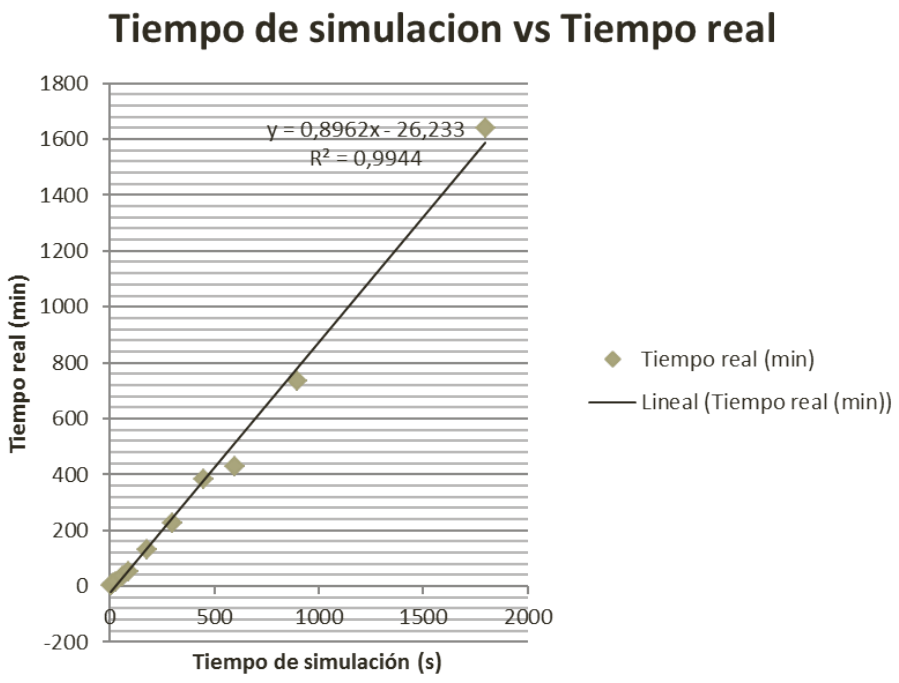


Figura 4.2: Tiempo de simulación vs Tiempo real

Para efectos de la experimentación y para hacer las simulaciones mas reales posibles, se determinó que un tiempo de 1200 seg (20 minutos), será el tiempo a usar para las pruebas,

debido a que es el tiempo promedio estimado de un usuario que se encuentra en uso de una red ya sea esta WiFi O LTE [42].

4.2. Diseño del experimento

Para cumplir con la hipótesis propuesta y conociendo que se encuentra implementado el algoritmo de detección SVD, se diseñó una simulación en una red mixta CR basada en LTE y WiFi. El experimento esta basado en una topología de red mostrada en la Figura 4.3, cabe resaltar que esta topología presentada sólo se muestra como un ejemplo ya que la posición y el movimiento de los UE es dinámico y aleatorio durante todo el tiempo de simulación.
El experimento consta de las siguientes características:

1. Modelos de propagación y movilidad utilizados y que son proporcionados por NS-3 y permiten el movimiento dinámico y aleatorio de los UE [39].

2. Los nodos incluidos en la topología propuesta son PU WiFi y PU LTE sin capacidad cognitiva (UE primarios).

3. Se utilizan dos tipos de SU; Los que tienen capacidad cognitiva sólo para una tecnología específica (PU WiFi y PU LTE), y capacidad cognitiva para ambas tecnologías (SU dual).

4. La frecuencia central de LTE se ha fijado a 729MHz, mientras que la frecuencia central de WiFi se ha fijado a 2400MHz.

5. El ancho de banda utilizado para ambas tecnologías es igual a 20MHz. Tanto los valores de frecuencia como el de ancho de banda fueron elegidos para hacer la simulación del modelo más real porque estos son valores típicos de despliegue comercial.

6. El número de PU WiFi, PU LTE, PU WiFi y PU LTE se ha establecido en 5, respectivamente, mientras que el número de SU duales se ha establecido en 10.

7. El intervalo de cobertura del punto AP se ha establecido en 200m, mientras que el rango de cobertura del eNB se ha establecido igual a 350m. Esto se hizo para generar interferencia entre las tecnologías evaluadas en la simulación.

8. La variable a tomar en consideración será SNR.

9. Todos los parámetros utilizados en la simulación se muestran en la Tabla 4.3, y pueden ser variados en trabajos futuros.

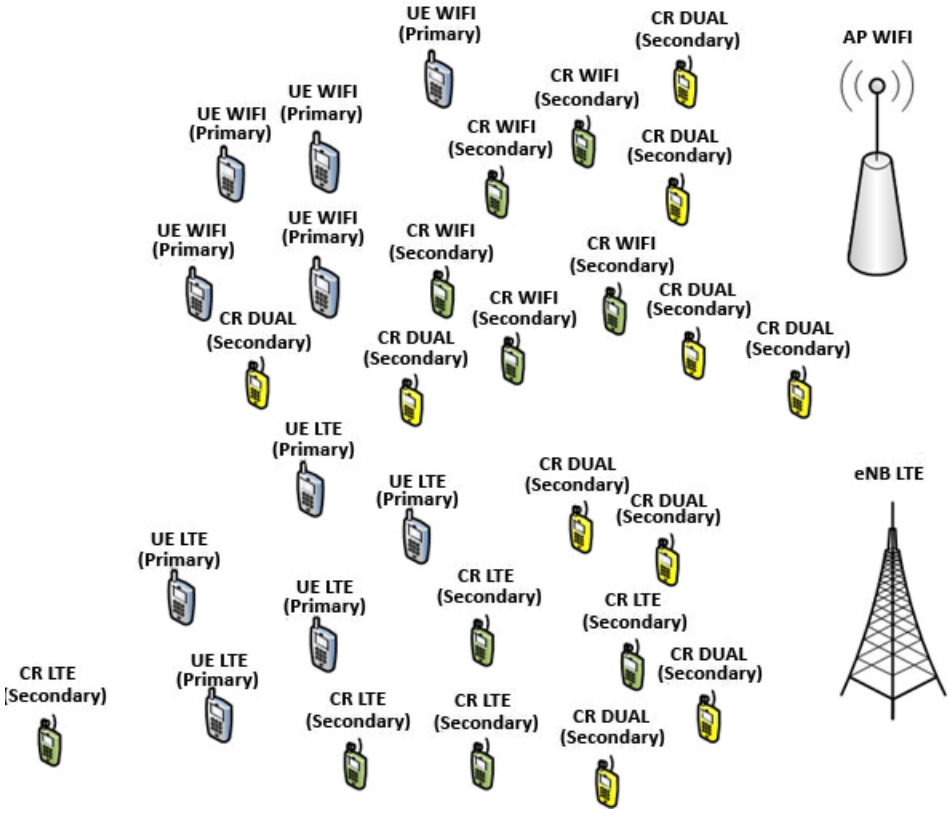


Figura 4.3: Topología de red propuesta para simulación

Tabla 4.3: Parámetros del sistema para simulación

Parámetros	Valor
Frecuencia LTE	729 Mhz
Frecuencia WiFi	2400 Mhz
Celdas eNB	3
Ancho de Banda LTE	20 Mhz
Ancho de Banda WiFi	20 Mhz
Ruido	Variable
Potencia de transmisión	0.037 mW
Potencia de recepción	0.06 mW
SU LTE	5
SU WiFi	5
SU duales	10
PU LTE	5
PU WiFi	5
Muestras para detección	16000
Rango de Cobertura AP	200 m
Rango de Cobertura eNB	350 m
Tipo de tráfico	TCP
Tiempo de simulación	1200 s
Modelo de Movilidad	Random Waypoint
Modelo de Propagación	Range Propagation Loss

4.3. Simulación y Datos generados

Para conocer el número de simulaciones que se deben realizar, tomando en cuenta factores como número de muestras en cada simulación, recursos computacionales y confiabilidad de los datos generados por las simulaciones, se generaron simulaciones con las siguientes características:

1. Las variables de consideración en las simulaciones son la Probabilidad de detección (P_{d}) en función del SNR, escogidas para poder ser comparadas con otros métodos de detección que están en función de estos parámetros.

2. La variación del SNR esta dada en pasos de 5 dB, entre -25 dB y 25 dB para una mejor visualización de la escala logarítmica.

3. El número de simulaciones se lo definió usando el método de Monte Carlo, con 21 iteraciones para cada valor de SNR, esto es para tener estimaciones confiables de las

distribuciones de los estadísticos de interés de los datos generados [43] [44].

4. Se usaron los parámetros de la Tabla 4.3, teniendo como resultados de la simulación, el archivo de datos generados */home/ns-allinone-3.23/ns-3.23/svd-result.txt* que se usará para el análisis posterior.

5. El parámetro Probabilidad de detección (P_{d}) de cada simulación se obtuvo dividiendo todas las muestras de cada simulación donde la detección fue 1 (detecto), para el número total de muestras.

6. A su vez el parámetro Probabilidad de detección (P_{d}) para cada valor de SNR especificado se obtuvo dividiendo la suma de todas las Probabilidades de detección (P_{d}) de cada simulación para el número de simulaciones realizadas (21), obteniendo la Tabla 4.4:

Tabla 4.4: Probabilidad de detección P_{d} vs SNR para método SVD propuesto

P_{d}	**SNR**
0.130483	-25 dB
0.209077	-20 dB
0.311604	-15 dB
0.418972	-10 dB
0.539313	-5 dB
0.652746	0 dB
0.761855	5 dB
0.815718	10 dB
0.851838	15 dB
0.856569	20 dB
0.913090	25 dB

7. Para poder realizar la comparación de eficiencia en términos de Probabilidad de detección del método propuesto, basada en los datos obtenidos en la simulación, se tuvo que realizar además simulaciones en el software estadístico R [45], de los métodos de detección SVD teórico usando la señal Raised Cosine [46] y el método Energy detection [47], con la configuración de parámetros semejante a las simulaciones en NS-3 para que resulten comparables, obteniendo la siguiente tabla (Tabla 4.5):

Tabla 4.5: Probabilidad de detección P_{d} vs SNR para método SVD Teórico y método Energy Detection

SVD	**Teórico**	**Energy**	**Detection**
P_{d}	**SNR**	P_{d}	**SNR**
0.108472	-25 dB	0.000012	-25 dB
0.141566	-20 dB	0.000034	-20 dB
0.146318	-15 dB	0.000125	-15 dB
0.208328	-10 dB	0.000689	-10 dB
0.429086	-5 dB	0.008365	-5 dB
0.540720	0 dB	0.467326	0 dB
0.663368	5 dB	0.983251	5 dB
0.731325	10 dB	0.996968	10 dB
0.889684	15 dB	0.998972	15 dB
0.918901	20 dB	0.999535	20 dB
0.995247	25 dB	0.999752	25 dB

4.4. Análisis de los datos

Una vez realizada las simulaciones y obtenido los datos, se procede a realizar el análisis de los mismos, mediante el siguiente procedimiento:

1. Para el análisis de los datos y construcción del modelo del método propuesto se usa el parámetro Probabilidad de detección (P_{d}) para cada valor de SNR especificado.

2. La probabilidad de detección (P_{d}) vs SNR se presenta como una Cumulative Distribution Function (CDF) [48] tanto para el método SVD propuesto como para los métodos de comparación SVD teórico y Energy detector, como se muestra en las Figuras 4.4, 4.5 y 4.6.

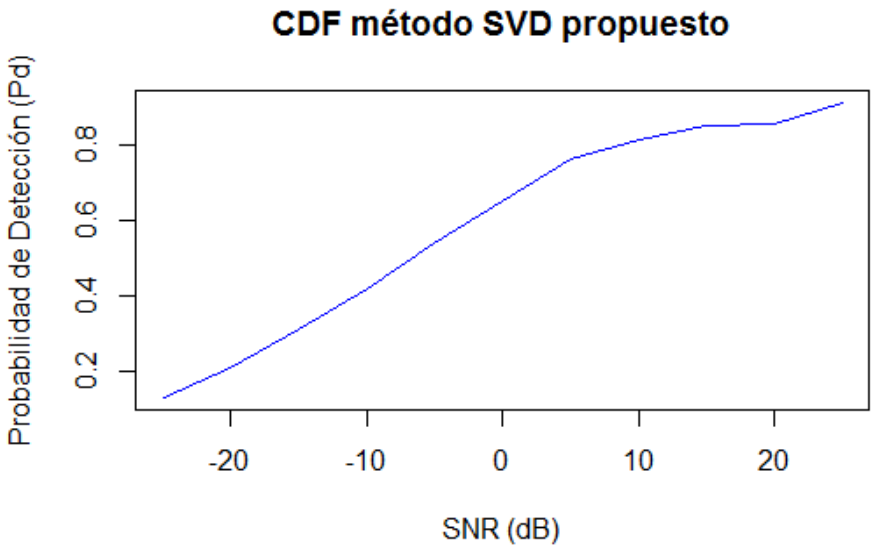


Figura 4.4: CDF para el método SVD propuesto

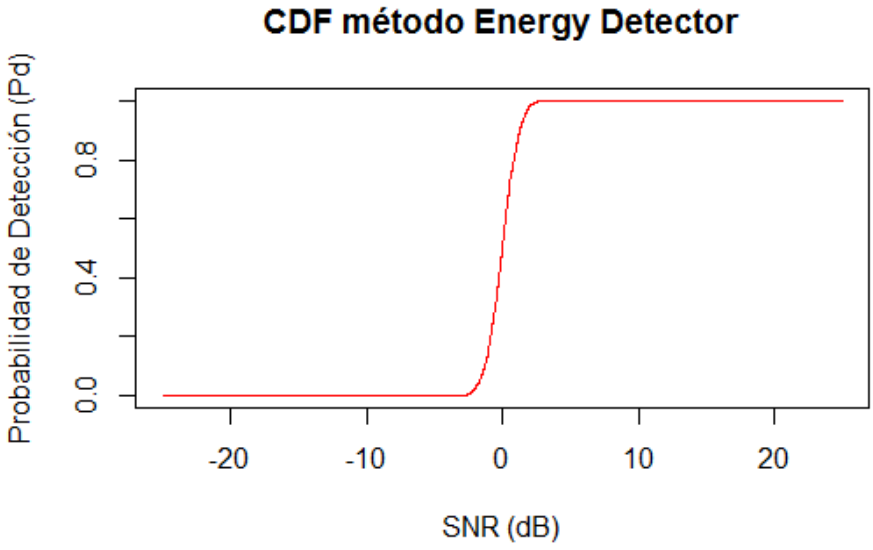


Figura 4.5: CDF para el método Energy Detector

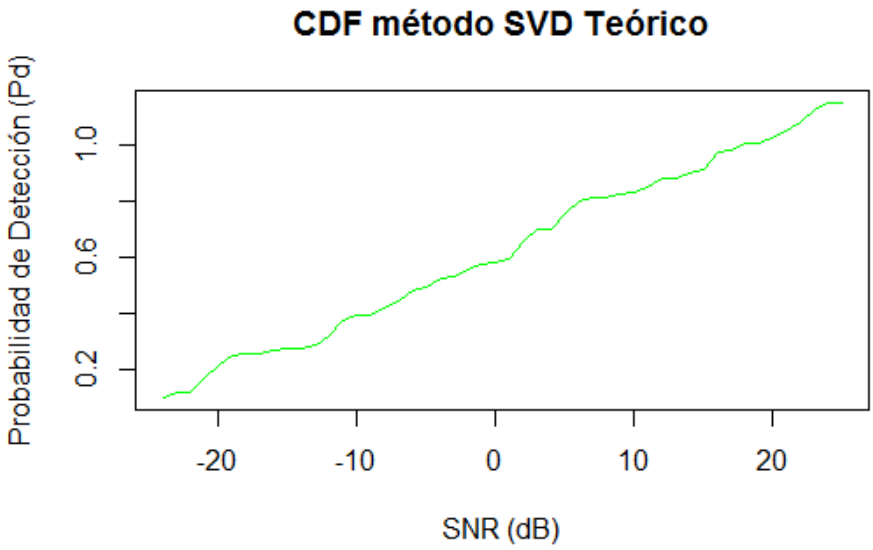


Figura 4.6: CDF para el método SVD teórico

3. Para el análisis visual de los modelos se optó por colocarlos en un mismo gráfico, como se observa en la Figura 4.7

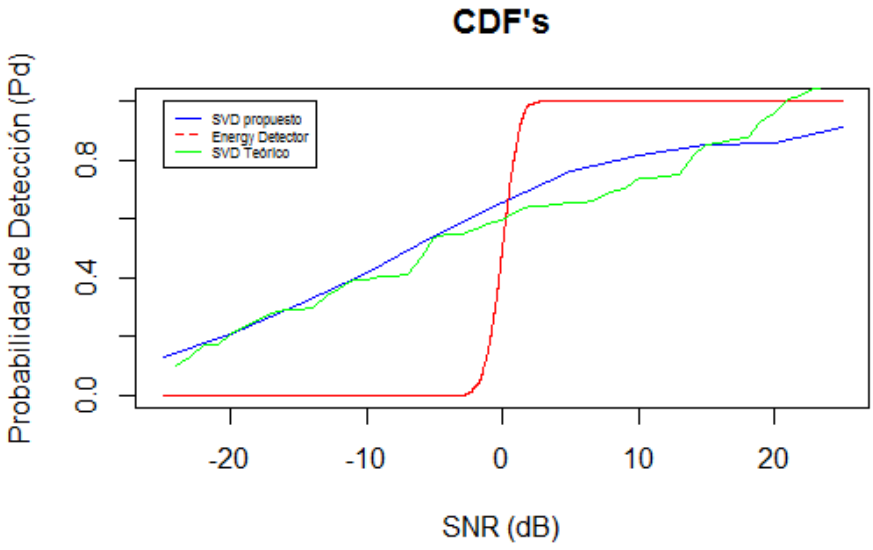


Figura 4.7: CDF de los tres métodos

4.5. Verificación

Para verificar la eficiencia del método de detección SVD propuesto, y una vez obtenido los datos y modelos de la Tabla 4.4 y Figura 4.4, es necesario usar un método estadístico confiable que permita validar dichos resultados al ser comparados con el método SVD teórico y el de Energy Detection, para lo cual se utilizó el Maximum Likelihood Estimation (MLE) derivada en [20], aplicado en un código desarrollado en el software R [45].
El MLE estima los parámetros de un determinado modelo probabilístico, en este caso los modelos de los métodos de detección obtenidos experimentalmente, permitiendo realizar una comparación estadística.
Cabe resaltar que se incluye el Error Estándar y la Desviación Estándar de los parámetros MLE debido a que un modelo de datos está incompleto si no se indica el error asociado a los valores del mismo y no se podría estimar el rango de validez del modelo presentado, por tal motivo la importancia de su inclusión [22]. Los resultados obtenidos se muestran en la Tabla 4.6:

Tabla 4.6: MLE de los métodos de detección

	MLE	**Std Error**	**SD**
SVD propuesto	0.587388	0.08495629	0.2817681
SVD Teórico	0.524820	0.1020092	0.3383261
Energy Detector	0.495913	0.1498638	0.4970421

En consecuencia, el sistema evaluado supera los métodos teóricos con los cuales es comparado.

4.6. Pruebas para generalización de modelo obtenido

Antes de realizar los experimentos para obtener los datos que nos permitirán cumplir el objetivo de verificación de eficiencia del método SVD, fue necesario realizar pruebas para observar el comportamiento de los datos variando un parámetro importante: el número de usuarios.
Presentando una pequeña hipótesis: *"Sí al variar el número de PU y SU de las simulaciones, con los demás parámetros fijos, excepto el Ruido, los datos obtenidos deben mantener su tendencia y no tener demasiadas diferencias para poder generalizar el modelo"*
Estas pruebas presentaron las siguientes características:

1. Los parámetros usados para las pruebas se especifican en la Tabla 4.7.

Tabla 4.7: Parámetros del sistema para pruebas previas

Parámetros	Valor
Frecuencia LTE	729 Mhz
Frecuencia WiFi	2400 Mhz
Celdas eNB	3
Ancho de Banda LTE	20 Mhz
Ancho de Banda WiFi	20 Mhz
Ruido	Variable
Potencia de transmisión	0.037 mW
Potencia de recepción	0.06 mW
SU LTE	Variable
SU WiFi	Variable
SU duales	Variable
PU LTE	Variable
PU WiFi	Variable
Muestras para detección	16000
Rango de Cobertura AP	200 m
Rango de Cobertura eNB	350 m
Tipo de tráfico	TCP
Tiempo de simulación	1200 s
Modelo de Movilidad	Random Waypoint
Modelo de Propagación	Range Propagation Loss

2. Por los recursos computacionales necesarios para este tipo de simulaciones y al ser este una limitante en la pruebas, se varió el número de SU y PU en pasos de 5 en 5 en el rango de 5 usuarios a 30 usuarios para cada tipo de usuario.

3. Las variables de consideración en las simulaciones son la Probabilidad de detección (P_{d}) en función del SNR, que también se usarán en los experimentos del siguiente capítulo.

4. La variación del SNR esta dada en pasos de 5 dB, entre -25 dB y 25 dB la misma que se usará también en el siguiente capítulo.

5. El número de simulaciones se lo definió en 21 para cada valor de SNR con determinado número de usuarios.

6. El parámetro Probabilidad de detección (P_{d}) para cada valor de SNR con determinado número de usuarios, se obtuvo dividiendo la suma de todas las Probabilidades de detección (P_{d}) de cada simulación para el número de simulaciones realizadas (21), obteniendo las siguientes tablas (Tabla 4.8, 4.9 y 4.10) :

Tabla 4.8: Probabilidad de detección P_{d} vs SNR para 15 SU, 10 PU y Probabilidad de detección P_{d} vs SNR para 30 SU, 20 PU

15 SU	**10 PU**	**30 SU**	**20 PU**
P_{d}	**SNR**	P_{d}	**SNR**
0.130543	-25 dB	0.130034	-25 dB
0.209341	-20 dB	0.204569	-20 dB
0.312341	-15 dB	0.310012	-15 dB
0.419452	-10 dB	0.415674	-10 dB
0.539654	-5 dB	0.536759	-5 dB
0.652865	0 dB	0.651954	0 dB
0.761345	5 dB	0.760034	5 dB
0.813429	10 dB	0.811563	10 dB
0.852345	15 dB	0.850214	15 dB
0.857658	20 dB	0.853462	20 dB
0.914567	25 dB	0.911043	25 dB

Tabla 4.9: Probabilidad de detección P_{d} vs SNR para 45 SU, 30 PU y Probabilidad de detección P_{d} vs SNR para 60 SU, 40 PU

45 SU	**30 PU**	**60 SU**	**40 PU**
P_{d}	**SNR**	P_{d}	**SNR**
0.130021	-25 dB	0.129994	-25 dB
0.201245	-20 dB	0.200010	-20 dB
0.310009	-15 dB	0.309934	-15 dB
0.412349	-10 dB	0.409954	-10 dB
0.531268	-5 dB	0.534593	-5 dB
0.650011	0 dB	0.634569	0 dB
0.760008	5 dB	0.759965	5 dB
0.810101	10 dB	0.805643	10 dB
0.850021	15 dB	0.849956	15 dB
0.851576	20 dB	0.850043	20 dB
0.910023	25 dB	0.910045	25 dB

Tabla 4.10: Probabilidad de detección P_{d} vs SNR para 75 SU, 50 PU y Probabilidad de detección P_{d} vs SNR para 90 SU, 60 PU

75 SU	**50 PU**	**90 SU**	**60 PU**
P_{d}	**SNR**	P_{d}	**SNR**
0.129867	-25 dB	0.125687	-25 dB
0.205467	-20 dB	0.194356	-20 dB
0.310345	-15 dB	0.295648	-15 dB
0.415674	-10 dB	0.402143	-10 dB
0.536794	-5 dB	0.521395	-5 dB
0.650423	0 dB	0.632349	0 dB
0.760003	5 dB	0.713455	5 dB
0.814563	10 dB	0.795832	10 dB
0.850032	15 dB	0.845326	15 dB
0.852345	20 dB	0.849658	20 dB
0.911131	25 dB	0.902559	25 dB

7. Como se puede observar en las tablas de los datos obtenidos en las pruebas, la tendencia se mantiene, comprobando que la tasa de detección está determinado por el umbral calculado en el algoritmo SVD, por lo tanto, cambiando el número de usuarios no afecta a dicho algoritmo, es decir se puede generalizar el modelo.

Capítulo 5

Conclusiones

En el presente capítulo se realizan las respectivas conclusiones del trabajo por secciones para un mejor análisis. Se analiza el cumplimiento de los objetivos del trabajo. Además se hace un análisis general y concluyente del desarrollo implementado y los datos obtenidos para los modelos estructurados. Se presentan recomendaciones y trabajos futuros, ya que la herramienta creada es un potente simulador que puede ser base de muchas otras investigaciones.

5.1. Objetivos

El objetivo principal del trabajo consiste en comprobar la eficiencia en el desempeño del método de detección Singular Value Decomposition (SVD) implementado en una red móvil CR en términos de Probabilidad de detección (Pd) vs SNR, comparado con los métodos de detección SVD teórico y Energy Detection teórico.

Se puede concluir en esta sección que el objetivo principal del trabajo es cumplido en su totalidad, desde el primer paso que es la implementación del método SVD en una red móvil LTE y WiFi basada en un desarrollo que consta de un simulador construido en NS-3 para la preparación del escenario de pruebas, en el cual incluye como entradas del simulador las métricas necesarias para mediciones y comparaciones como el SNR y como salidas de datos la Probabilidad de detección (Pd). Como segundo paso se implementó también en desarrollos basados en el software estadístico R, los métodos SVD teórico y Energy Detection teórico para poder completar el objetivo, luego con el análisis estadístico, se verificó la eficiencia de estos tres métodos.

Los objetivos específicos están relacionados con la metodología usada, por lo que cumplir el objetivo principal involucra satisfacer los objetivos específicos.

5.2. Metodología

La metodología descrita en el punto 4 de la Introducción, fue llevada de acuerdo a la planificación estructurada para este trabajo, la cual nos permitió el cumplimiento del objetivo principal. Como primer paso la investigación de los métodos y algoritmos necesarios para completar un sistema CR que cumpla con todas las etapas de su ciclo, en segundo lugar el desarrollo de un simulador donde los métodos y algoritmos escogidos en la etapa de investigación sean implementados en el software, en tercer lugar fueron desarrollados los escenarios de simulación adecuados para la verificación del método SVD propuesto y su respectiva comparación con los otros métodos de detección y en cuarto lugar el análisis de los datos obtenidos en la simulación mediante el método estadístico MLE, que permite concluir si el método investigado y adaptado al trabajo es más eficiente que los escogidos para su contraste.

5.3. Desarrollo e implementación

El desarrollo y la implementación esta involucrado directamente con parte de los objetivos específicos, en particular con el diseño del simulador CR de redes móviles, herramienta fundamental para la creación de los experimentos que permitieron obtener los datos necesarios. Las cuatro etapas de un sistema CR, detección, decisión, compartición y movilidad, fueron implementadas en los módulos *cognitive*, *wifi* y *lte* modificados, adaptados y compatibles con el software NS-3.
Dada la completa implementación del sistema CR en las tecnologías WiFi y LTE,este simulador se convierte en una potente herramienta no sólo para cumplir los objetivos del trabajo en cuanto a la etapa de detección, sino para pruebas en cualquier otra etapa del sistema, investigaciones posteriores en cuanto a rendimientos de la red, ya que nos permite variar los parámetros de entrada del simulador en rangos muy usados en las redes comerciales.

5.4. Análisis de datos y Resultados

De acuerdo al diseño de los experimentos realizados, los datos obtenidos y su análisis se puede concluir en las siguientes afirmaciones:

1. Para verificar la eficiencia de los métodos se utilizó los datos de las simulaciones con los parámetros adecuados, obteniendo modelos estadísticos CDF de los tres métodos de detección. Se utilizó el estimador estadístico MLE y sus respectivos errores asociados para estimar los parámetros de los modelos probabilísticos, y realizar las comparaciones entre sí de dichos modelos. Utilizando este estimador, los resultados obtenidos son los siguientes: MLE(SVD propuesto) = 0,587388, Std Error (SVD propuesto) = 0.08495629, SD (SVD propuesto) = 0.2817681, MLE(SVD teórico) = 0,524820, Std Error (SVD teórico) = 0.1020092, SD (SVD teórico) = 0.3383261 y MLE (Energy detection teórico) = 0,495913, Std Error (Energy detection teórico) = 0.1498638, SD (Energy detection teó-

rico) = 0.4970421. En consecuencia, el sistema propuesto evaluado supera los métodos teóricos propuestos.

2. Se observó el comportamiento de los datos variando el número de usuarios PU y SU, factores importante y claves para medir en una red. De acuerdo a estas pruebas realizadas exclusivamente para poder generalizar el modelo y los resultados obtenidos, se pudo verificar que la tendencia del modelo probabilístico CDF del método de detección SVD propuesto se mantiene, concluyendo que la tasa de detección de PU por parte de los SU, está determinado por el umbral calculado y ajustado en el algoritmo SVD, por lo tanto, al variar el número de usuarios el modelo no es afectado drásticamente, es decir podemos generalizar el modelo.

5.5. Recomendaciones

Entre las recomendaciones principales para la comprensión y uso del trabajo y simulador se encuentran las siguientes:

1. Se recomienda leer los anexos, en especial el Anexo C antes de realizar alguna o prueba o simulación en el desarrollo creado.

2. Colocar todas las entradas de parámetros técnicos de manera correcta, entre los límites establecidos, para que no se produzcan problemas en las simulaciones y resultados.

3. Para experimentos que requieren de gran cantidad de parámetros de entrada, como mayor número de usuarios, incremento de número de muestras, es recomendable tener gran capacidad de procesamiento y equipos de computo a disposición, para acortar tiempos en la presentación de resultados.

5.6. Trabajos complementarios y futuros

Este trabajo fue base para la elaboración de un manuscrito científico (paper) de título *SVD Detection Analysis in Cognitive Mobile Radio Networks*, que será expuesto en Italia, Milan en el ICUFN 2017 Ninth International Conference on Ubiquitous and Future Networks 2017 (link del evento http://icufn.org/), del 4 al 7 de julio del 2017.
El desempeño del método SVD descrito en este trabajo puede mejorarse aumentando el número de muestras configuradas en los SU, teniendo esto en cuenta para futuros trabajos.
El simulador CR desarrollado en el presente trabajo, cumple con los principios básicos indispensables en un sistema CR, entregándonos resultados en cada una de las etapas CR, por lo tanto se pueden realizar análisis futuros y comparaciones con otros métodos en las etapas de decisión, compartición y movilidad, incluso nos permite proponer nuevos métodos en cada etapa CR para trabajos investigativos y doctorales.

Bibliografía

[1] Y. Xiao and F. Hu, *Cognitive Radio Networks*. Auerbach Publications, 2008.

[2] B. Wang, Y. Wu, and K. R. Liu, "Game theory for cognitive radio networks: An overview," *Computer networks*, vol. 54, no. 14, pp. 2537–2561, 2010.

[3] Z. Han, *Game Theory in Wireless and Communication Networks*. Cambridge University Press, 2011.

[4] I. M. Kalil, "Cognitive radio the ieee 802.22 standard," *Integrated Communication Systems*, vol. 15, 2011.

[5] A. K. Al-Ali, *Database-assisted end-to-end theoretical and simulation framework for cognitive radio networks*. PhD thesis, NORTHEASTERN UNIVERSITY, 2014.

[6] J. H. Aguilar Rentería and A. Navarro Cadavid, "Radio cognitiva–estado del arte," *Sistemas y Telemática*, vol. 9, no. 16, pp. 31–53, 2011.

[7] D. B. Cabric, *Cognitive Radios: Systems Design Perspective*. PhD thesis, University of California, Berkeley, 2007.

[8] E. Biglieri, A. J. Goldsmith, L. J. Greenstein, H. V. Poor, and N. B. Mandayam, *Principles of cognitive radio*. Cambridge University Press, 2013.

[9] Z. Feng, Q. Zhang, and P. Zhang, *Cognitive Wireless Networks*. Springer International Publishing, 2015.

[10] D. A. López, E. R. Trujillo, and O. E. Gualdron, "Elementos fundamentales que componen la radio cognitiva y asignación de bandas espectrales," *Información tecnológica*, vol. 26, no. 1, pp. 23–40, 2015.

[11] S. Bhattacharjee, P. Das, S. Mandal, and B. Sardar, "Optimization of probability of false alarm and probability of detection in cognitive radio networks using ga," in *2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)*, pp. 53–57, July 2015.

[12] M. Welvaert and Y. Rosseel, "On the definition of signal-to-noise ratio and contrast-to-noise ratio for fmri data," *PloS one*, vol. 8, no. 11, p. e77089, 2013.

[13] J. Cheng and G. Shi, "Symbolic computation of snr for variational analysis of sigma-delta modulator," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pp. 443–448, IEEE, 2014.

[14] B. Cao, Q. Zhang, and J. W. Mark, *Cooperative Cognitive Radio Networking*. Springer-Verlag GmbH, 2016.

[15] R. B. López and S. Montejo Sánchez, "La radio cognitiva y su impacto en el uso eficiente del espectro de radio," *Ingeniería Electrónica, Automática y Comunicaciones*, vol. 36, no. 1, pp. 42–55, 2015.

[16] C. Hemández, L. Pedraza, I. Páez, and E. Rodríguez-Colina, "Análisis de la movilidad espectral en redes de radio cognitiva," *Información tecnológica*, vol. 26, no. 6, pp. 169–186, 2015.

[17] J. Rodriguez, *Fundamentals of 5G Mobile Networks*. John Wiley & Sons Inc, 2015.

[18] N. S. Foundation, "Cognitive radio cognitive network simulator." `http://faculty.uml.edu/Tricia_Chigan/Research/CRCN_Simulator.htm`. Visitado: 2016-05-15.

[19] A. Al-Ali and K. Chowdhury, "Simulating dynamic spectrum access using ns-3 for wireless networks in smart environments," in *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking Workshops (SECON Workshops)*, pp. 28–33, June 2014.

[20] L. Held and D. Sabanés Bové, *Applied statistical inference*, vol. 10. Springer, 2014.

[21] J. DeMeritt, "Maximum likelihood estimation (mle)." `http://jdemeritt.weebly.com/uploads/2/2/7/7/22771764/mle_introduction1.pdf`. Visitado el: 2017-01-20.

[22] B. Efron and R. Tibshirani, "Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy," *Statistical science*, pp. 54–75, 1986.

[23] N. Muchandi and R. Khanai, "Cognitive radio spectrum sensing: A survey," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 3233–3237, Mar. 2016.

[24] R. Abdelrassoul, E. Fathy, and M. S. Zaghloul, "Comparative study of spectrum sensing for cognitive radio system using energy detection over different channels," in *2016 World Symposium on Computer Applications & Research (WSCAR)*, pp. 32–35, Mar. 2016.

[25] M. F. Fahim and M. S. Raeen, "Svd detection for cognitive radio network based on average of maximum-minimum of the icdf," *International Journal of Advanced Computer Research*, vol. 2, Sep. 2012.

[26] S. Xu, K. S. Kwak, and R. R. Rao, "Svd based wideband spectrum sensing and carrier aggregation for lte-advanced networks," in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, pp. 1190–1194,

Sep. 2014.

[27] Y. Zeng and Y. C. Liang, "Maximum-minimum eigenvalue detection for cognitive radio," in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–5, Sep. 2007.

[28] W. Saad, Z. Han, M. Debbah, A. Hjorungnes, and T. Basar, "Coalitional games for distributed collaborative spectrum sensing in cognitive radio networks," in *IEEE INFOCOM 2009*, pp. 2114–2122, April 2009.

[29] T. Wang, L. Song, Z. Han, and W. Saad, "Overlapping coalitional games for collaborative sensing in cognitive radio networks," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 4118–4123, Apr. 2013.

[30] Q. Ni, R. Zhu, Z. Wu, Y. Sun, L. Zhou, and B. Zhou, "Spectrum allocation based on game theory in cognitive radio networks.," *JNW*, vol. 8, no. 3, pp. 712–722, 2013.

[31] P. Masek, M. Slabicki, J. Hosek, and K. Grochla, "Transmission power optimization in live 3gpp lte-a indoor deployment," in *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 164–170, Oct 2016.

[32] 3rd Generation Partnership Project, "Long term evolution overview.." `http://www.3gpp.org/technologies/keywords-acronyms/98-lte`. Visitado: 2016-09-12.

[33] T. Lei, X. Wen, Z. Lu, W. Jing, B. Zhang, and G. Cao, "Handoff management scheme based on frame loss rate and rssi prediction for ieee 802.11 networks," in *2016 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 555–559, Sept 2016.

[34] I. S. Association *et al.*, "Ieee 802.11™-2012," *standards. ieee. org/about/get/802/802.11. html*. Visitado: 2016-09-13.

[35] M. Gerasimenko, N. Himayat, S. p. Yeh, S. Talwar, S. Andreev, and Y. Koucheryavy, "Characterizing performance of load-aware network selection in multi-radio (wifi/lte) heterogeneous networks," in *2013 IEEE Globecom Workshops (GC Wkshps)*, pp. 397–402, Dec. 2013.

[36] M. Nahas, M. Mjalled, Z. Zohbi, Z. Merhi, and M. Ghantous, "Enhancing lte - wifi interoperability using context aware criteria for handover decision," in *2013 25th International Conference on Microelectronics (ICM)*, pp. 1–4, Dec 2013.

[37] R. Bassoli, H. Marques, J. Rodriguez, S. Vahid, and R. Tafazolli, "Network coding for vertical handoffs between lte and ieee 802.11n: An energy perspective," in *European Wireless 2013; 19th European Wireless Conference*, pp. 1–5, April 2013.

[38] A. L. Yusof, N. Ya'acob, and M. T. Ali, "Hysteresis margin for handover in long term evolution (lte) network," in *2013 International Conference on Computing, Management*

and Telecommunications (ComManTel), pp. 426–430, Jan 2013.

[39] *ns-3 Model Library, Release ns-3.23*, Aug. 2015.

[40] A. Spillner, T. Linz, and H. Schaefer, *Software testing foundations: a study guide for the certified tester exam.* Rocky Nook, Inc., 2014.

[41] S. Smith, D. R. Jefferson, P. D. Barnes, Jr., and S. Nikolaev, "Improving per processor memory use of ns-3 to enable large scale simulations," in *Proceedings of the 2015 Workshop on Ns-3*, WNS3 '15, (New York, NY, USA), pp. 60–66, ACM, 2015.

[42] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan, "Wifi, lte, or both?: Measuring multi-homed wireless internet performance," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, (New York, NY, USA), pp. 181–194, ACM, 2014.

[43] I. C. Ramírez, C. J. Barrera, and J. C. Correa, "Efecto del tamaño de muestra y el número de réplicas bootstrap," *Ingeniería y Competitividad*, vol. 15, no. 1, pp. 93–101, 2013.

[44] U. M. Alfonso and M. V. Carla, *Modelado y simulación de eventos discretos.* Editorial UNED, 2013.

[45] R. C. Team, "R: A language and environment for statistical computing.," *Vienna: R Foundation for Statistical Computing.*, 2014.

[46] S. S. Ali, C. Liu, and M. Jin, "Minimum eigenvalue detection for spectrum sensing in cognitive radio," *International Journal of Electrical and Computer Engineering*, vol. 4, pp. 623–630, Aug. 2014.

[47] R. Abdelrassoul, E. Fathy, and M. S. Zaghloul, "Comparative study of spectrum sensing for cognitive radio system using energy detection over different channels," in *2016 World Symposium on Computer Applications & Research (WSCAR)*, pp. 32–35, Mar. 2016.

[48] G. Zhao, W. Zhang, and S. Li, "Advanced sensing techniques for cognitive radio," *SpringerBriefs in electrical and computer engineering (*, 2016.

Acrónimos

3GPP 3rd Generation Partnership Project.

AODV Ad hoc On-Demand Distance Vector.

AP Access Point.

ARP Address Resolution Protocol.

AWGN Additive White Gaussian Noise.

BER Bit error Rate.

CDF Cumulative Distribution Function.

CPE Customer Premises Equipment.

CR Cognitive Radio.

CRCN Cognitive Radio Cognitive Network Simulator.

DCF Distributed Coordination Function.

eNB E-UTRAN Node B.

FCC Federal Communications Commission.

LTE Long Term Evolution.

MAC Media Access Control.

MLE Maximum Likelihood Estimation.

MLME MAC Layer Management Entity.

NS-2 Network Simulator 2.

NS-3 Network Simulator 3.

PHY Physical Layer.

PU Primary Users.

QoS Quality of Service.

RMT Random Matrix Theory.

RSS Received Signal Strength.

RSSI Received Signal Strength Indication.

SAP Service Access Point.

SDR Software-Defined Radio.

SINR Signal to interference plus Noise Ratio.

SIR Signal to Interference Ratio.

SME Server Manager Entity.

SNR Signal to Noise Ratio.

SR Software Radio.

SSF Spectrum Sensing Function.

SU Secundary Users.

SVD Singular Value Decomposition.

UE User Equipment.

UHF Ultra High Frecuency.

VHF Very High Frecuency.

WiFi Wireless Fidelity.

WRAN Cognitive Radio Wireless Regional Area Network.

Anexos

A. Paper enviado y aceptado por el ICUFN 2017

1. **Título**: SVD Detection Analysis in Cognitive Mobile Radio Networks.
2. **Conferencia**: ICUFN The Ninth International Conference on Ubiquitous and Future Networks 2017
3. **Fecha y Lugar de la Conferencia**: Julio 4-Julio 7 del 2017. Milan, Italia.
4. **Fecha de submmit**: 10 de febrero del 2017.
5. **Fecha de aceptación**: 22 de abril del 2017.
6. **Página web**: http://icufn.org/

SVD Detection Analysis in Cognitive Mobile Radio Networks

Pablo Palacios, Alberto Castro, Cesar Azurdia-Meza, Claudio Estevez
Department of Electrical Engineering
Universidad de Chile
Santiago, Chile
$pablo.palacios@ug.uchile.cl$, $alberto.castro@uchile.cl$,
$\{cazurdia,\ cestevez\}@ing.uchile.cl$

***Abstract*—In this work, the performance of the spectrum detection method known as singular value decomposition (SVD) is evaluated. The performance of the SVD detection method was analyzed in terms of probability of detection (P_d) versus signal-to-noise ratio (SNR) in a mixed LTE and WiFi cognitive radio network. The results were compared via numerical simulations with the theoretical SVD detection method and the theoretical energy detection method. The maximum likelihood estimation (MLE) statistical estimator was used to verify the efficiency of the evaluated techniques. The evaluated system outperformed the other methods in terms of P_d.**

***Keywords*—Cognitive mobile radio networks, probability of detection (P_d), singular value decomposition (SVD), spectrum sensing.**

I. Introduction

Cognitive radio has attracted plenty of studies in recent years because it helps to solve the under-utilization of frequency spectrum [1], [2]. In a cognitive radio system, users without spectrum license, called secondary users (SU), are allowed to use frequency bands not being used by primary users (PU), thus improving the spectrum efficiency [3].

Cognitive radio (CR) has drawn significant attention from academic and industrial communities to meet the ever-growing needs of spectrum resources and high data rate communication [4]. The main advantage offered by CR is efficiency in the use of the electromagnetic spectrum, since it allows optimum spectrum management through a four step process (cognitive cycle): spectrum detection, spectrum decision, and spectrum sharing and mobility [5]. Other characteristics that CR presents are the following: reconfiguration of operation parameters (frequency, power, among others) for the use of different access technologies, and the capacity of supporting larger bandwidths in heterogeneous networks [6]. Within the entire cognitive cycle, there are more challenges in the stages of detection and decision, being these areas of enormous importance for the whole process, and presenting the following difficulties: high hardware requirements difficulties in the detection of PU, and security issues on the network [7]. Under these parameters, the correct and effective detection of PU becomes a priority. Therefore, a number of methods have been proposed, including detection algorithms [8], energy detection schemes [9], Eigen-value based detection [10], singular value decomposition (SVD) detection [11], among others.

In detection methods based on maximum and minimum eigenvalues [12], the decision threshold that determines the presence of the signal is derived from the theory of the random matrix (RMT) to determine the signal detection hypothesis. SVD detection is very similar to the eigenvalues decomposition method. However, SVD detection is more general since it can be applied to any type of matrix because it doesn't have to be a square matrix. Another efficient algorithm is the one proposed in [13], which provides an average of the maximum-minimum inverse cumulative distribution function (ICDF). Using a raised cosine (RC) test signal, and without having to know a priori information regarding the signal, the performance of the SVD signal detector is more efficient compared to other detection methods [13].

The main goal of this manuscript is to evaluate the performance of the SVD detection method applied to a cognitive mobile radio network, specifically in a mixed LTE and WiFi cognitive network via Network Simulator 3 (NS-3.23) modules [14]. This will validate the SVD detection method in a functional mobile network. The performance of the SVD detection method is analyzed in terms of probability of detection (P_d) versus signal-to-noise ratio (SNR) using numerical simulations and compared with the traditional theoretical energy detection method [12] and the SVD detection (RC signal) algorithm [13].

II. Mathematical Model of the Implemented SVD Detection Algorithm

In this paper we evaluate spectrum sensing using the SVD detection technique in a cognitive mobile radio network, specifically implemented for LTE and WiFi, which are state-of-the-art technologies and basic techniques in heterogeneous networks [15]. First, select the value of N, L and k, such that $k < L < N - k$, where N is the number of samples to be taken by the receiver, L is defined as the 'smoothing factor' i.e. number of consecutive values covariance matrix of the received signal can take, and k is the number of singular values. Following the notation given in [13], consider $k = 2$ and $L = 16$. Then the matrix of covariances is given by the following expression

$$R(N) = \frac{1}{N} \sum_{L=16}^{16-1+N} \hat{x}(n) x^{\dagger}(n), \tag{1}$$

where $R(N)$ is the covariance matrix of the discrete signal $\hat{x}(n)$ received by the CR, and $x^{\dagger}(n)$ is the Hermitian of $\hat{x}(n)$. Then, we factorize the covariance matrix, i.e. apply the SVD

TABLE I. NORMALIZED DISTRIBUTION PROBABILITY

t	-3.9	-3.18	-2.78	-1.91	-1.27	-0.59	0.45	0.98
$F_1(t)$	0.01	0.05	0.10	0.30	0.50	0.70	0.90	0.95

Algorithm 1 Proposed SVD Detection Algorithm

Require: k, L, N and $k < L < N - k$.
Ensure: $PU\ detection\ bit$.

```
while communication is progress do
    Setup k = 2 and L = 16
    Obtain CovMat = CreateMatrixCovariance(L)
    Factorize(CovMat)
    Obtain Max = Maximum(CovMat) and Min = Minimum(CovMat)
    Obtain Threshold = CalculateThreshold()
    if (Max/Min < Threshold) then
        return PU detection bit=1
    else
        return PU detection bit=0
    end if
end while
```

method to obtain the singular values, obtaining the following expression [13]

$$R = U\Sigma V^t, \tag{2}$$

where $R(mxn)$ is the covariance matrix, $U(mxn)$ is the matrix of singular vectors of columns R, $\Sigma(mxn)$ is the matrix of singular values, and $V^t(mxn)$ is the matrix of singular vectors of rows R. Then, the maximum and minimum eigenvalues of the covariance matrix R(N_s) are obtained, λmax and λmin, respectively. Next, we calculate the threshold value and compare it with the eigenvalues using the following expression [13]

$$\gamma = \frac{(\sqrt{N}+\sqrt{L})^2}{(\sqrt{N}-\sqrt{L})^2} * (1 + (\frac{(\sqrt{N}+\sqrt{L})^{-\frac{2}{3}}}{N * L^{\frac{1}{6}}}) * F_1^{-1}(1 - P_{fa})), \tag{3}$$

where P_{fa} is the probability of false alarm that is required to be equal or less than 0.1 ($P_{fa} <= 0.1$). The expression $F_1^-1(1 - P_{fa})$ is the Tracy-Widom function, which is the normalized probability distribution for the eigenvalues described in Table I [16]. We compare the relation between the maximum and minimum eigenvalues of the covariance matrix with the threshold; therefore, if $\frac{\lambda_{max}}{\lambda_{min}} < \gamma$, the signal is present, otherwise the signal is absent. Finally we obtain the signal detection bit of the PU. The algorithm for the above approach is shown in Algorithm 1.

III. PERFORMANCE EVALUATION

We implemented the SVD detection algorithm and simulated it in a mixed cognitive radio network based on LTE and WiFi in a module developed in Network Simulator 3 (NS-3.23). This module contains the 4 main steps of a cognitive cycle, which are spectrum detection using the SVD detection method [13], spectrum decision based on a game of coalitions [17], and spectrum Sharing and mobility based on the power parameter RSSI, parameter used to carry out the hand-off between WiFi and LTE [18]. It should be stressed that NS3 has WiFi and LTE modules included, but these technologies were adapted and modified to give it the cognitive ability.

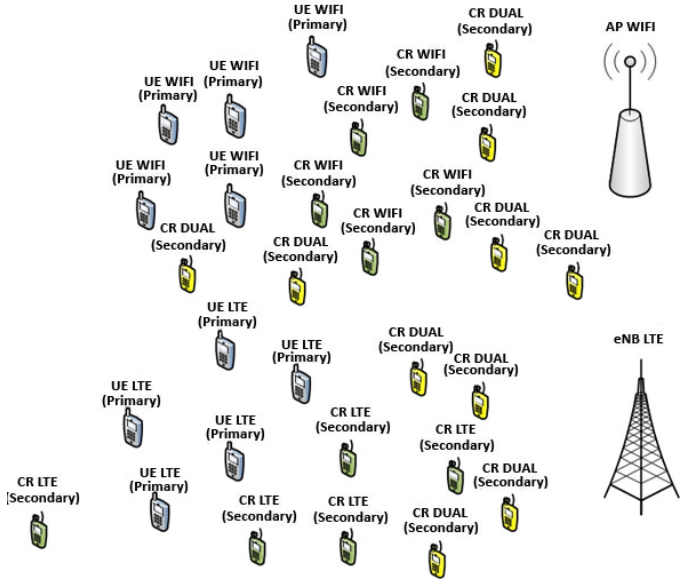


Fig. 1. Proposed Network Topology for Simulation

There are several nodes in the proposed network topology, as depicted in Fig. 1. The topology presented in Fig. 1 is shown only as an example since the position and movement of the user equipments (UE) is dynamic and random during the entire simulation time, due to the propagation and mobility models used and that are provided by the NS-3 simulator [14]. Some nodes included in the proposed topology are WiFi PU and LTE PU without cognitive ability (primary UE). Two types of SU are used; the ones with cognitive ability only for a specific technology (WiFi CR UE or LTE CR UE), and cognitive ability for both technologies (dual CR UE).

The central LTE frequency has been set to 729MHz, whereas the WiFi central frequency has been set to 2400MHz. The bandwidth used for both technologies is equal to 20MHz. These values were chosen to make the simulation of the model more real because these are typical values of commercial deployment. The number of CR LTE UE, CR WiFi UE, LTE UE, and primary WiFi UE have been set to 5, respectively, while the dual CR UE have been set equal to 10. The range of coverage of the access point (AP) has been set to 200m, while the range of coverage the evolved node B (eNB) has been set equal to 350m. This was done to generate interference between the evaluated technologies in the simulation. All of the parameters used in the simulation are shown in Table II, and may be varied in future work.

The probability of detection (P_d) vs SNR is presented as a cumulative distribution (CDF) function for various methods, as shown in Fig. 2. This is done by varying the SNR in steps of 5 dB, between -25 dB and 25 dB for a better visualization of the logarithmic scale. The proposed SVD method curve was obtained through the implementation and simulation of the algorithm in NS-3, whereas the curves of the SVD detection method (RC signal) [13] and the traditional energy detector method [12] were obtained via theoretical simulations using MATLAB.

To verify the efficiency of the methods, and by using the data and models obtained from Fig. 2, the statistical

TABLE II. SYSTEM SIMULATION PARAMETERS

Parameter	Value
LTE Frequency	729MHz
WiFi Frequency	2400MHz
LTE Bandwidth	20MHz
WiFi Bandwidht	20MHz
CR LTE UE	5
CR WiFi UE	5
Dual CR UE	10
Primary LTE UE	5
Primary WiFi UE	5
AP Range of Coverage	200m
eNB Range of Coverage	350m
Time of Simulation	1200s
Mobility model	Random Waypoint
Propagation model	Range Propagation Loss

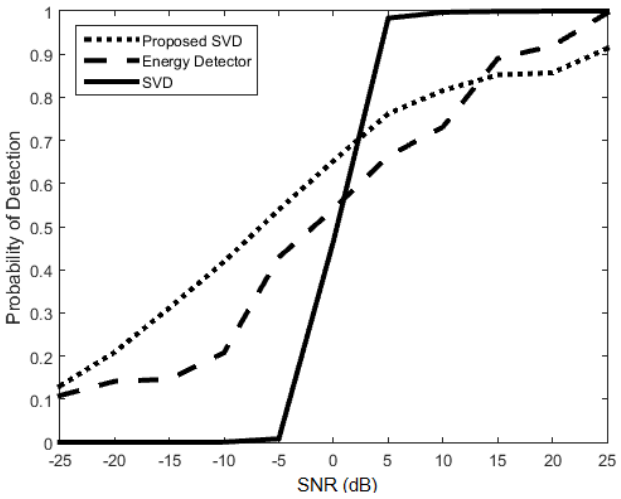


Fig. 2. Probability of Detection vs SNR

estimator Maximum Likelihood Estimate (MLE) derived in [19] was used. The MLE estimates the parameters of a certain probabilistic model, and it is also used to compare different models. Using this statistical estimator, the results obtained are as follows: MLE (Proposed SVD) = 0.587388, MLE (SVD) = 0.524820, and MLE (Energy Detector) = 0.495913. Consequently, the evaluated system outperforms the theoretical methods proposed in [12] and [13].

IV. CONCLUSION

In this work, we highlight an approach based on the SVD detection method applied to cognitive radio, specifically in WiFi and LTE wireless networks. The system was evaluated in terms of the probability of detection P_d, and compared with other traditional detection methods. The maximum likelihood estimation (MLE) statistical estimator was used to verify the efficiency of the evaluated methods. Overall, the evaluated system outperformed the other methods in terms of P_d. The performance of the SVD method reported in this paper can be improved by increasing the number of samples in the CR receptor, which might me considered as future work.

ACKNOWLEDGMENT

The authors acknowledge the partial financial support of SENESCYT "Convocatoria abierta 2014-primera fase, Acta CIBAE-023-2014", as well as the projects FONDECYT No. 11160517 and project ERANET-LAC ELAC2015/T10-0761.

REFERENCES

[1] D. M. Alias and R. G. K, "Cognitive radio networks: A survey," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Mar. 2016, pp. 1981–1986.

[2] S. Bhattarai, J. M. J. Park, B. Gao, K. Bian, and W. Lehr, "An overview of dynamic spectrum sharing: Ongoing initiatives, challenges, and a roadmap for future research," *IEEE Transactions on Cognitive Communications and Networking*, vol. 2, no. 2, pp. 110–128, June 2016.

[3] Y. Ma, Y. Gao, Y. C. Liang, and S. Cui, "Reliable and efficient sub-nyquist wideband spectrum sensing in cooperative cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 10, pp. 2750–2762, Oct. 2016.

[4] Y. Xu and M.-S. Lim, "The selective transmission with majority even/odd subcarriers for the high data throughput in ofdma based cognitive radio systems," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2016, pp. 628–631.

[5] E. Biglieri, A. J. Goldsmith, L. J. Greenstein, H. V. Poor, and N. B. Mandayam, *Principles of cognitive radio*. Cambridge University Press, 2013.

[6] C. Hemández, L. Pedraza, I. Páez, and E. Rodríguez-Colina, "Análisis de la movilidad espectral en redes de radio cognitiva," *Información tecnológica*, vol. 26, no. 6, pp. 169–186, Dec. 2015.

[7] M. Z. Alom, T. K. Godder, and M. N. Morshed, "A survey of spectrum sensing techniques in cognitive radio network," in *2015 International Conference on Advances in Electrical Engineering (ICAEE)*, Dec. 2015, pp. 161–164.

[8] N. Muchandi and R. Khanai, "Cognitive radio spectrum sensing: A survey," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Mar. 2016, pp. 3233–3237.

[9] R. Abdelrassoul, E. Fathy, and M. S. Zaghloul, "Comparative study of spectrum sensing for cognitive radio system using energy detection over different channels," in *2016 World Symposium on Computer Applications & Research (WSCAR)*, Mar. 2016, pp. 32–35.

[10] S. M. Jacob and N. S., "Spectrum sensing technique in cognitive radio based on sample covariance matrix," in *2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, Dec. 2015, pp. 139–144.

[11] S. Xu, K. S. Kwak, and R. R. Rao, "Svd based wideband spectrum sensing and carrier aggregation for lte-advanced networks," in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, Sep. 2014, pp. 1190–1194.

[12] S. S. Ali, C. Liu, and M. Jin, "Minimum eigenvalue detection for spectrum sensing in cognitive radio," *International Journal of Electrical and Computer Engineering*, vol. 4, no. 4, pp. 623–630, Aug. 2014.

[13] M. F. Fahim and M. S. Raeen, "Svd detection for cognitive radio network based on average of maximum-minimum of the icdf," *International Journal of Advanced Computer Research*, vol. 2, no. 3, Sep. 2012.

[14] *ns-3 Model Library, Release ns-3.23*, Aug. 2015. [Online]. Available: https://www.nsnam.org/docs/release/3.23/models/ns-3-model-library.pdf

[15] A. Galanopoulos, F. Foukalas, and T. A. Tsiftsis, "Efficient coexistence of LTE with WiFi in the licensed and unlicensed spectrum aggregation," *IEEE Transactions on Cognitive Communications and Networking*, vol. 2, no. 2, pp. 129–140, Jun. 2016.

[16] Y. Zeng and Y. C. Liang, "Maximum-minimum eigenvalue detection for cognitive radio," in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, Sep. 2007, pp. 1–5.

[17] T. Wang, L. Song, Z. Han, and W. Saad, "Overlapping coalitional games for collaborative sensing in cognitive radio networks," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 4118–4123.

[18] M. Gerasimenko, N. Himayat, S. p. Yeh, S. Talwar, S. Andreev, and Y. Koucheryavy, "Characterizing performance of load-aware network selection in multi-radio (wifi/lte) heterogeneous networks," in *2013 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2013, pp. 397–402.

[19] L. Held and D. Sabanés Bové, *Applied statistical inference*. Springer, 2014, vol. 10.

B. Documento técnico para desarrollo de software

Technical specifications to develop in the project

Step 1

For the creation of nodes, these have to have the following characteristic technicians based in the capacities of the software NS3:

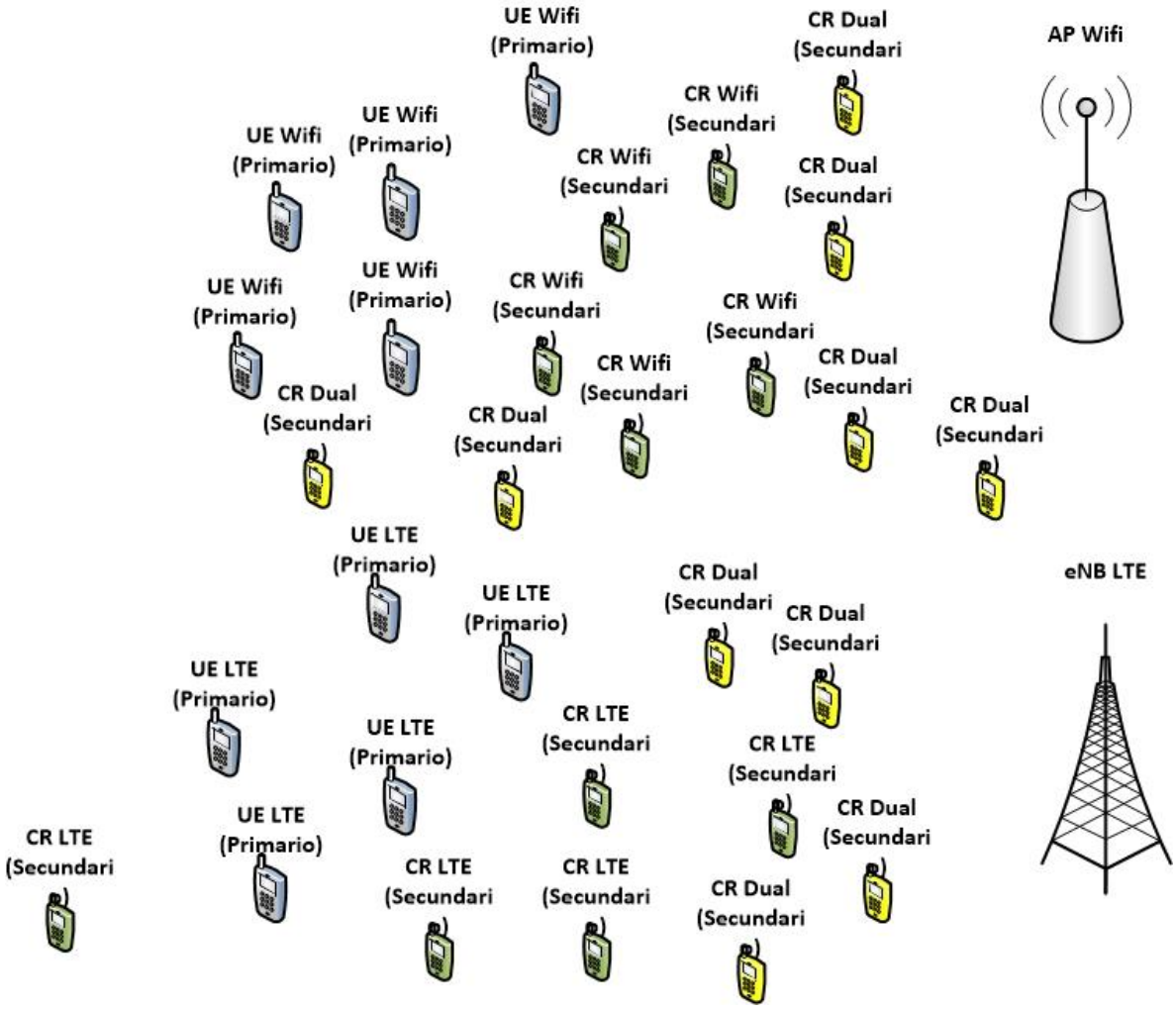
Node	Type	Quantity	Mobility	Channels	Frequency	Power/Sensitivity
Wifi	AP	1	Fixed	11	2.4 Ghz	(11,30 dbm)
Wifi	EU	5	Random	11	2.4 Ghz	/(Max -70 dbm)
LTE	eNB	1	Fixed	41	2.6 Ghz	(11,30 dbm)
LTE	EU	5	Random	41	2.6 Ghz	(11,30 dbm)/max -120 dbm)
DUAL CR	Dual (capacity of reconfiguration wifi and lte)	10	Random	11+41	Adaptative (2.4-2.6 Ghz)	(11,30 dbm)/max -120 dbm)
WIFI-CR	Only capacity CR wifi	5	Random	11	2.4 Ghz	/(Max -70 dbm)
LTE-CR	Only capacity CR LTE	5	Random	41	2.6 Ghz	(11,30 dbm)/max -120 dbm)

Considerations for functions of simulation and Restrictions:

- The nodes LTE have to allow change his carrier frecuency (by defect 2.6 Ghz and 700 Mhz) in the function that uses to simulate.
- For the location of the eNB and the AP to use in the simulation has to program a function that extracts the coordinates of each node from an archive of text. The first row of the archive will correspond to a cell with his coordinates (x, y, z) and bands of operation for the DL and UL, The second row of the archive will correspond to the AP with his coordinates (x, y, z) and band of operation. Each one of these values have to be separated by tabulations.
- The initial location of the EU has to be random, the in rank of coverage of the eNB and the AP.
- The EU Wifi and EU LTE will have his interface by defect (primary users), the EU WIFI-CR only have to have the interface CR with capacity of recognition WIFI, the EU LTE-CR only will have the interface CR with capacity of recognition LTE, The DUAL EU-CR will have the dual capacity CR, so that when they lose connectivity wifi, are able of reconfiguration for the connection LTE.

- The players (UEs) have to compete by the spectrum, that is to say, The EU WIFI will be the main users of the network Wi-Fi, The EU LTE will be the main users of the network LTE, THE EU CR will be the secondary users that they have to compete for the channels.
- The EU con dual capacity, when they lose a connection wifi, and if they find in the area of coverage LTE, has to negotiate the resources with the eNB, using his cognitive skill (detection, decision, sharing and mobility). Of equal way when it loses the connection LTE and find in the area of coverage wifi, have to negotiate with the resources AP.
- The EU Wifi or LTE and to his time the EU CR with only capacity Wifi or LTE, if they lose the connection, will be out of the game, until they return to the area of coverage of his respective technologies.
- Once initiated the simulation the terminal will move inside the area established by the the AP and the eNB using the model of mobility "Random Walk".
- The traffic between transmitters and receptors to simulate has to be TCP with option to that in the function of simulation, can it to him change to UDP.
- The simulator has to have an archive of basic configurations that will have the following parameters: Power of transmission and reception of all the nodes, figure of noise for all the nodes, bandwidths, frequencies, that the simulator takes by defect in each execution if it does not redefine the specific value for the variables through the console with the function of simulation or inside the script.
- The diagram of the implementation will have to be the following:

Figure 1: Network Topology for Simulation



Step 2

For the creation of the module CR, will have to fulfil with the basic principles of the cognitive radio (detection, decision, sharing and mobility).

Detection:

For the detection of the spectrum has to use the method of "SVD (Singular Value Decomposition) Detection", which describes to continuation:

1) Select the number of columns of the matrix of covariances L in the interface receptor CR of the secondary users, such that k < L < N-k, where N is the number of samples to take by part of the receptor and k is the number of singular values, for this procedure consider k=2 and L=16
2) Proceed to obtain the matrix of covariances

$$R(N_s) = \frac{1}{N_s} \sum_{n=16}^{16-1+N_s} x(n)x^t(n)$$

Where x(n) is the SNR of the channel that the receptor is detecting in this instant and divided for the number of samples.

3) Factorizar The matrix of covariances
4) Obtain the maximum and the minimum eigenvalues of the matrix of covariance $R(N_s)$ which will be λ_{max} and λ_{min}.
5) Calculate the value threshold to compare with the eigenvalues with the following formulates:

$$\gamma = \frac{\left(\sqrt{N_s} + \sqrt{16}\right)}{\left(\sqrt{2\,N_s} - \sqrt{L16}\right)} 2 * (1 + \frac{\left(\sqrt{N_s} + \sqrt{16}\right)^{\overline{2}}}{(3\,N_s * 16)^{\overline{1}} 6 * (F_1^{-1}(1 - P_{fa}))})$$

Where P_{fa} is the probability of false alarm that requires that it was $\leq$ to 0.1.

The function $(F_1^{-1}(1 - P_{fa}))$ is the function Tracy-Widom that it is distribution of probability normalized for the eigenvalues whose table distribution is the following:

t	-3.90	-3.18	-2.78	-1.91	-1.27	-0.59	0.45	0.98	2.02
$F_1(t)$	0.01	0.05	0.10	0.30	0.50	0.70	0.90	0.95	0.99

6) Compare the relation of eigenvalues maximum and minima of the matrix of covariance with the threshold, therefore If $\frac{\lambda_{max}}{\lambda_{min}} > \gamma$, the signal is present, of another way the signal is not present.
7) Obtain the bit of detection of signal of individual primary user.

Step 3

Decision, sharing and mobility

In the stage of decision and sharing is where will implement the theory of the game based in coalitions (cooperative game) in three phases:

PHASE 1: local Detection

- Each user CR individual will obtain his bit of detection of signal of primary user.

PHASE 2: Training of adaptative coalitions

- During the training of the adaptative coalition assumes that any secondary user of random way can begin the process of union.
- Form coalitions based in the algorithm of merge and separation indicated to continuation:

Merge: the coalition decides to join following the following steps:

1) It decides join any set of coalitions if the function of utility of the merge is better compared con each coalition by individual, besides if the set of coalitions covers all the users of the partition, and by Pareto, is preferable, given his function of utility, compared with the partitions no united.
2) The comparison makes based in the following function of utility of each user CR:

$$v(S) = Q_{d,S} - C(Q_{f,S}) = (1 - Q_{m,S}) - C(Q_{f,S})$$

Where $C(Q_{f,S})$ is the function cost of probability of false alarm inside the coalition S.

$$Q_{f,S} = 1 - \prod_{i \in S}\left[\left(1 - P_f\right)\left(1 - P_{y,i,k}\right) + P_f P_{y,i,k}\right]$$

$Q_{f,S}$ *is the probability of false alarm of the coalition having like head of coalition* k

P_f *Is the probability of false alarm of an user CR*

$P_{y,i,k}$ *is thepr obabilidad to report an error between an user CR of the coalition And his head of coalition* k

$$Q_{m,S} = \prod_{i \in S}\left[\left(P_{m,i}\right)\left(1 - P_{y,i,k}\right) + (1 - P_{m,i})P_{y,i,k}\right]$$

$Q_{m,S}$ *is the probability to lose the detection of a PU by part of the coalition*

$P_{m,i}$ *is the probability to lose the deteccion of a PU by part of an usuario CR*

3) The coalitions and the users CR have to follow the following properties:

- In the game of detection of collaboration proposed, the utility of a coalition S is equal to the utility of each user CR in the coalition.
- The probabilities of false alarm and to lose the detection of a PU of any user CR that belongs to the coalition are given by said probabilities but of the coalition

Separation: it decides separate a set of coalitions if the function of utility of each coalition of the set by individual is better that the union of the coalitions

- It makes this procedure until finding the optimum group of coalitions in the elements of the network
- **Restrictions of coalition:** The minimum coalition to form will be of an user CR by collation (that is to say there are not coalitions in reality), The maximum coalition will be of all the users CR (that is to say an alone coalition of all the users CR)

PHASE 3 Detection of the coalition

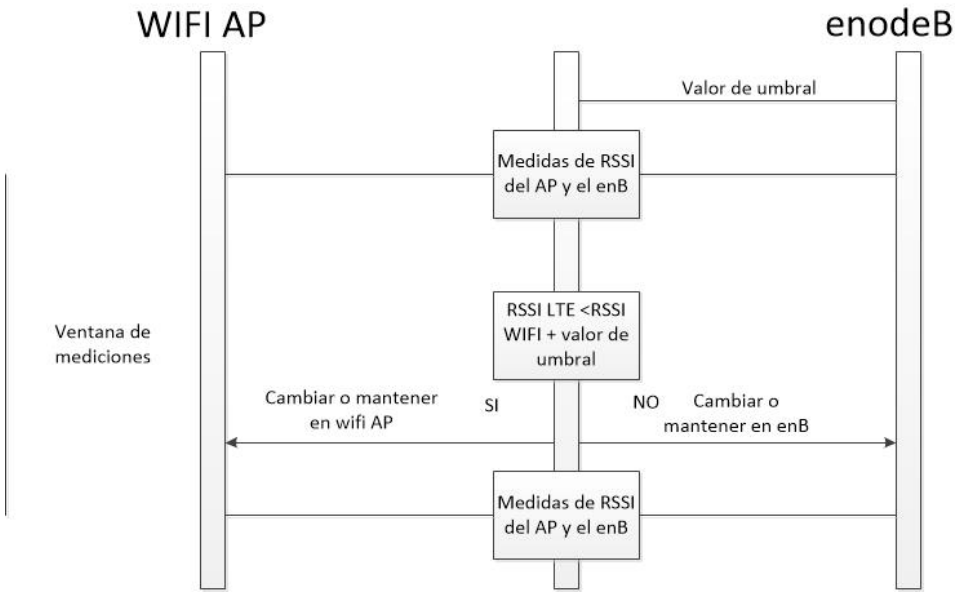
- Each user CR reports his bit of detection to each head of the coalition
- The head of each coalition takes a final decision on the presence or no of a primary user using a rule OR

- Its Users CR in a coalition will fulfil with the final decision taken by the head of the coalition.

All these phases have to repeat along the operation of the network

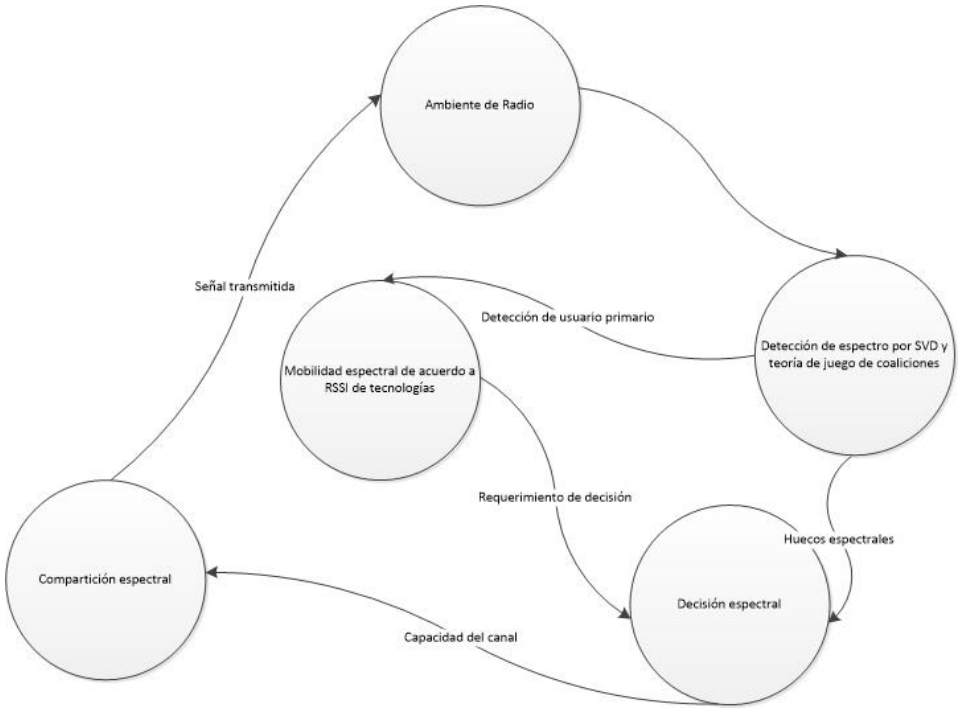
In the phase of mobility will take in account the following diagram to know when make the change of technology.

Figure 2: Diagram for change of technology



To compare when an user CR has to change of technology, depending on his mobility, will use the measurement of the RSSI of agreement to the signal wifi and lte, and comparing both RSSI but a value threshold that according to experimental studies is given in 1.76 db.

Diagram simplified of development



Functions to consider for the development:

1) **Main function to initiate the simulation:**

This function has to allow changes mentioned in the step 1 like number of primary and secondary users, carrier frequency of the technology, parameters of location of the nodes, choose the type of traffic, and read the archive of basic configurations.

2) **Svd(snr Channel, probability of false alarms user CR)**

This function will commission to make the detection of the spectrum by part of the user CR by individual, and will return us a value of 0 if the user CR does not detect a primary user, and 1 if the user CR detects a primary user

3) **Coalition (quantity of users Cr, bit of detection of CR, probability of false alarms user CR, probability to lose the detection of an user CR, type of technology)**

This function will commission to make the coalitions of the users CR of agreement to the parameters specified in the algorithm of coalitions and will return the already structured coalitions and the final decision taken by the coalition.

4) **Makehandoff(Channel, rssi ap, rssi eNB)**

This function will commission to make the handoff of technologies depending on the parameters of entrance and the comparison between rssi, having like result, the change of technology in the channel indicated, or the change of channel in the same technology, if this is used by a primary user.

5) Spectrum Sensing/Database Query

This bloque commissions to check if it exists a primary user in a determinate channel in a determinate term of time. Is infers the activity of the primary users from a static database of primary users that loads before that it begin the simulation.

This submodule can use for: simulate the detection with a probability gives of error of detection or consult a database of approved by the FCC to determine the activity of the primary users.

The database of the primary users that loads in the simulator is an archivor of text that defines the number of primary users, his current busy channel, the power of transmission to determine his rank of coverage, and a list with the number of times that lights or turns off.

6) Spectrum Decision:

In this block, carry out diverse political. In the first place, a politics to determine if a hand -off has to make on the results of the detection/consults of the results. Second, a politics that determines to which channel of has to give the hand-off.

7) Spectrum Mobility:

This submodule initiates the protocol of hand-off in the current node. It is tied directly to the physical layer.

8) Spectrum Sharing:

This submodule use the support incorporated for the detection of bearer of the layer MAC of the protocol 802.11 in NS3 to ensure that the available spectrum shares in a free environment of collisions for the nodes CR that decided to transmit by the same channel.

- All functions must be tested with test data for proper operation.
- If functions need other input parameters can be added.
- You can create intermediate functions or other functions as may be necessary.

Summary of main functions and parameters:

Functions	Parameters	Restrictions	Results expected
Main function	Quantity of users primary and secondary Frequency Location of nodes Type of traffic Archive of basic configurations	• Primary users; minímum 2 (1 wifi and 1 lte), maximum 10 (5 wifi and 5 lte). • Secondary users: minimum 3 (1 CR wifi, 1 CR LTE 1 CR dual), maximum 20(5 CR wifi, 5 CR LTE, 5 CR dual)	Main parameters of the network (delay, jitter, throughput, packet loss) in data and graphic.
SVD	• SNR Of the channel measured by the user CR • Probability of the false alarm of the user CR	SNR Maximum and minimum of the channels for his operation	If it detects a primary user 1 If it does not detect a primary user 0
Coalicion	• Quantity of users Cr • Bit of detection of CR • probability of false alarms user CR • Probability to lose the detection of an user CR • Type of technology and user CR	Maximum and minimum quantities of users CR	Head of coalition Coalitions structured Decision of the coalition
Empezarhandoff	• Channel • RSSI AP • RSSI eNB	Values of RSSI in the AP and eNB maximum and minima for correct operation	Make the handoff Not making the handoff

C. Manual de uso práctico del simulador

Reporte de instalación y uso de desarrollo para Tesis

Fecha de entrega: 28 de octubre 2016

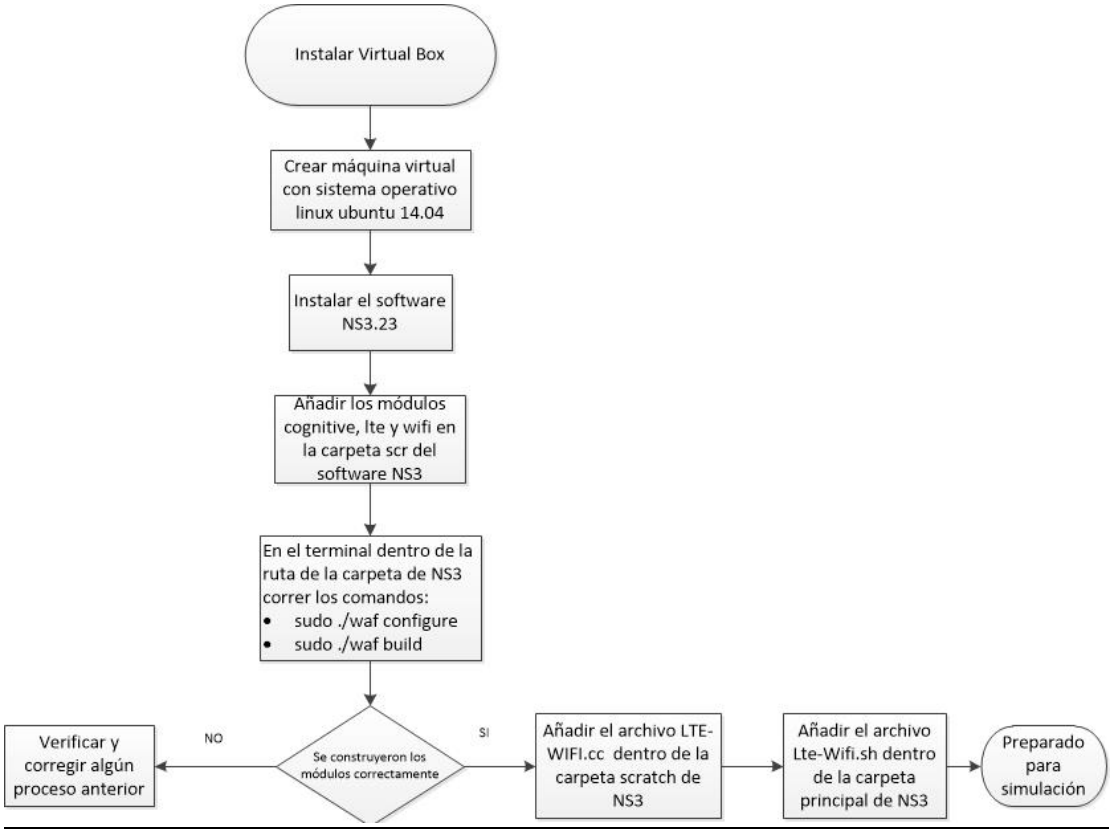
Pablo Palacios Játiva

1. **Desarrollo:**
   - Módulo cognitive desarrollado en ns3
   - Módulo lte desarrollado en ns3
   - Módulo wifi desarrollado en ns3
   - Programa principal para generación de simulación LTE-WIFI.CC
   - Script para inicialización de parámetros Lte-Wifi.sh

2. **Requerimientos de software para instalación de desarrollo**
   - NS3.23
   - Ubuntu 14.04

3. **Configuración de los módulos desarrollados**

Diagrama de flujo para la instalación del desarrollo



Pasos para la instalación del desarrollo

- Instalar el software NS3.23 en la máquina virtual creada en Virtual Box
- Añadir los módulos cognitive, lte y wifi desarrollados dentro de la carpeta src (home/pablo/ns-allinone-3.23/ns-3.23/src) que contiene módulos ya preestablecidos en ns3, si existen ya los módulos wifi y lte en esa carpeta, reemplazarlos.

Módulos
/home/pablo/ns-allinone-3.23/ns-3.23/src/antenna
/home/pablo/ns-allinone-3.23/ns-3.23/src/aodv
/home/pablo/ns-allinone-3.23/ns-3.23/src/applications
/home/pablo/ns-allinone-3.23/ns-3.23/src/bridge
/home/pablo/ns-allinone-3.23/ns-3.23/src/brite
/home/pablo/ns-allinone-3.23/ns-3.23/src/buildings
/home/pablo/ns-allinone-3.23/ns-3.23/src/click
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive
/home/pablo/ns-allinone-3.23/ns-3.23/src/config-store
/home/pablo/ns-allinone-3.23/ns-3.23/src/core
/home/pablo/ns-allinone-3.23/ns-3.23/src/csma
/home/pablo/ns-allinone-3.23/ns-3.23/src/csma-layout
/home/pablo/ns-allinone-3.23/ns-3.23/src/dsdv
/home/pablo/ns-allinone-3.23/ns-3.23/src/dsr
/home/pablo/ns-allinone-3.23/ns-3.23/src/energy
/home/pablo/ns-allinone-3.23/ns-3.23/src/fd-net-device
/home/pablo/ns-allinone-3.23/ns-3.23/src/flow-monitor
/home/pablo/ns-allinone-3.23/ns-3.23/src/internet
/home/pablo/ns-allinone-3.23/ns-3.23/src/lr-wpan
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte
/home/pablo/ns-allinone-3.23/ns-3.23/src/mesh
/home/pablo/ns-allinone-3.23/ns-3.23/src/mobility
/home/pablo/ns-allinone-3.23/ns-3.23/src/mpi
/home/pablo/ns-allinone-3.23/ns-3.23/src/netanim
/home/pablo/ns-allinone-3.23/ns-3.23/src/network
/home/pablo/ns-allinone-3.23/ns-3.23/src/nix-vector-routing
/home/pablo/ns-allinone-3.23/ns-3.23/src/olsr
/home/pablo/ns-allinone-3.23/ns-3.23/src/openflow
/home/pablo/ns-allinone-3.23/ns-3.23/src/point-to-point
/home/pablo/ns-allinone-3.23/ns-3.23/src/point-to-point-layout
/home/pablo/ns-allinone-3.23/ns-3.23/src/propagation
/home/pablo/ns-allinone-3.23/ns-3.23/src/sixlowpan
/home/pablo/ns-allinone-3.23/ns-3.23/src/spectrum

/home/pablo/ns-allinone-3.23/ns-3.23/src/stats
/home/pablo/ns-allinone-3.23/ns-3.23/src/tap-bridge
/home/pablo/ns-allinone-3.23/ns-3.23/src/test
/home/pablo/ns-allinone-3.23/ns-3.23/src/topology-read
/home/pablo/ns-allinone-3.23/ns-3.23/src/uan
/home/pablo/ns-allinone-3.23/ns-3.23/src/virtual-net-device
/home/pablo/ns-allinone-3.23/ns-3.23/src/visualizer
/home/pablo/ns-allinone-3.23/ns-3.23/src/wave
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi
/home/pablo/ns-allinone-3.23/ns-3.23/src/wimax
/home/pablo/ns-allinone-3.23/ns-3.23/src/create-module.py
/home/pablo/ns-allinone-3.23/ns-3.23/src/wscript

- Abrir el terminal e ir a la ruta ns-allinone-3.23/ns-3.23
- Correr el comando sudo ./waf configure
- Deberá aparecer la siguiente pantalla (las opciones no habilitadas no afectan el desarrollo):

```
---- Summary of optional NS-3 features:
Build profile             : debug
Build directory           :
Python Bindings           : enabled
Python API Scanning Support   : not enabled (Missing 'pygccxml' Python module)
BRITE Integration         : not enabled (BRITE not enabled (see option --with-
brite))
NS-3 Click Integration    : not enabled (nsclick not enabled (see option --with-
nsclick))
GtkConfigStore            : enabled
XmlIo                     : enabled
Threading Primitives      : enabled
Real Time Simulator       : enabled
File descriptor NetDevice     : enabled
Tap FdNetDevice           : enabled
Emulation FdNetDevice     : enabled
PlanetLab FdNetDevice     : not enabled (PlanetLab operating system not
detected (see option --force-planetlab))
Network Simulation Cradle     : not enabled (NSC not found (see option --with-
nsc))
MPI Support               : not enabled (option --enable-mpi not selected)
NS-3 OpenFlow Integration     : not enabled (Required boost libraries not found)
SQlite stats data output      : enabled
```

```
Tap Bridge                 : enabled
PyViz visualizer           : enabled
Use sudo to set suid bit   : not enabled (option --enable-sudo not selected)
Build tests                : not enabled (defaults to disabled)
Build examples             : not enabled (defaults to disabled)
GNU Scientific Library (GSL) : enabled
'configure' finished successfully (12.283s)
```

- Correr el comando sudo ./waf build
- Deberá aparecer la siguiente pantalla (solo existe un warning que no afecta al desarrollo):

```
Waf: Entering directory `/home/pablo/ns-allinone-3.23/ns-3.23/build'
wscript:66: Warning: (in applications) Requested to build modular python
bindings, but apidefs dir not found => skipped the bindings.
 # these variables are mandatory ('/' are converted automatically)
wscript:310: Warning: (in lte) Requested to build modular python bindings, but
apidefs dir not found => skipped the bindings.
 env.append_value('LINKFLAGS', '-lgcov')
wscript:172: Warning: (in wifi) Requested to build modular python bindings, but
apidefs dir not found => skipped the bindings.
 help=('DEPRECATED (run ./waf shell)'),
Waf: Leaving directory `/home/pablo/ns-allinone-3.23/ns-3.23/build'
'build' finished successfully (1m47.299s)

Modules built:
antenna                aodv                   applications
bridge                 buildings              cognitive (no Python)
config-store           core                   csma
csma-layout            dsdv                   dsr
energy                 fd-net-device          flow-monitor
internet               lr-wpan                lte
mesh                   mobility               mpi
netanim (no Python)    network                nix-vector-routing
olsr                   point-to-point         point-to-point-layout
propagation            sixlowpan              spectrum
stats                  tap-bridge             test (no Python)
topology-read          uan                    virtual-net-device
visualizer             wave                   wifi
wimax
```

Modules not built (see ns-3 tutorial for explanation):		
brite	click	openflow

- Verificar que se haya contruido con los modulos cognitive, lte y wifi.
- Añadir el archivo LTE-WIFI.cc desarrollado dentro de la carpeta scratch (home/pablo/ns-allinone-3.23/ns-3.23/scratch)
- Añadir el archivo Lte-Wifi.sh desarrollado dentro de la carpeta principal de NS3 (home/pablo/ns-allinone-3.23/ns-3.23)

4. Inicio de la simulación

Diagrama de flujo de uso del desarrollo

*Archivos de resultados:
Todos los archivos de resultados de la simulación se encuentran en las siguientes rutas:

- /home/pablo/ns-allinone-3.23/ns-3.23/DlPdcpStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/DlRlcStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/DlRsrpSinrStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/DlRxPhyStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/DlTxPhyStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/UlPdcpStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/UlRlcStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/UlRsrpSinrStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/UlRxPhyStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/UlTxPhyStats.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/wifi-network_perf.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/handover-result.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/svd-result.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/coalition-result.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/graph-information.txt
- /home/pablo/ns-allinone-3.23/ns-3.23/Lte-Wifi.flomon
- Archivos con extensión .pcap

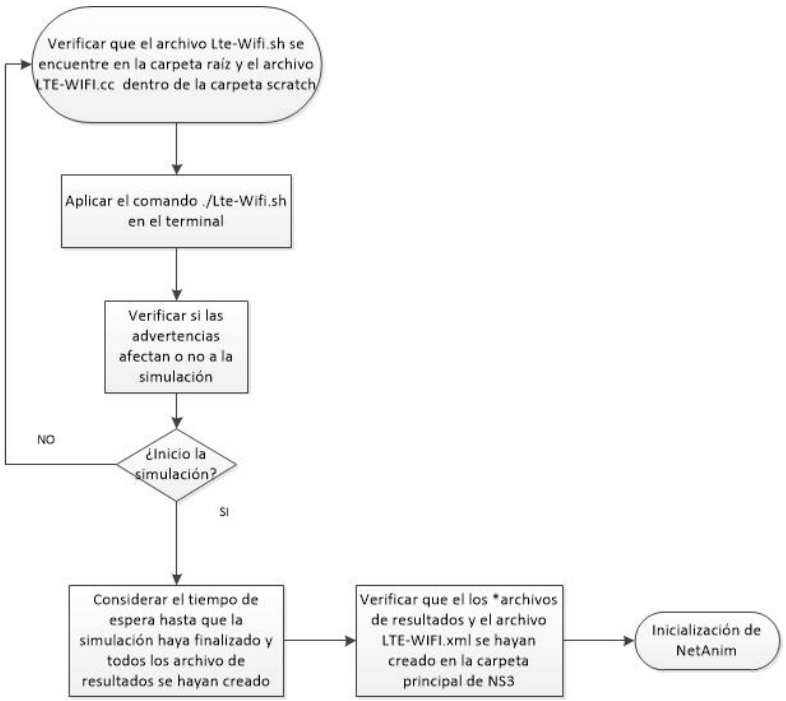
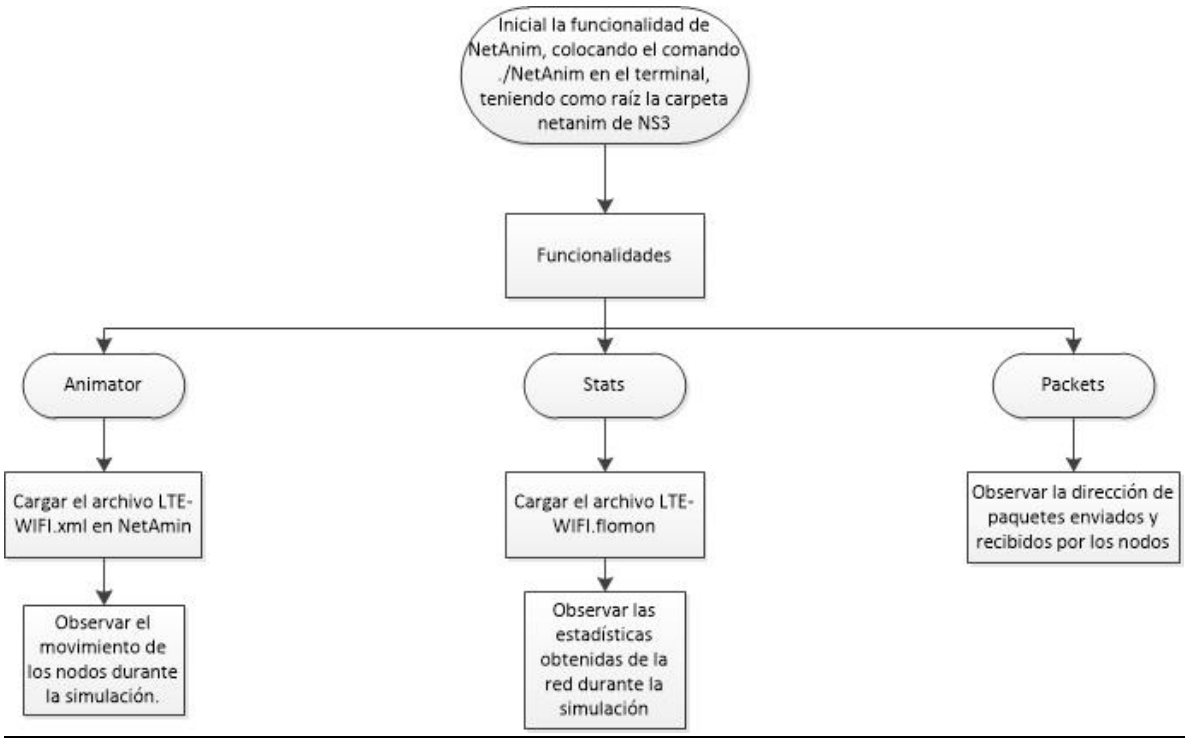


Diagrama de flujo de uso de NetAnim



Pasos para uso del desarrollo

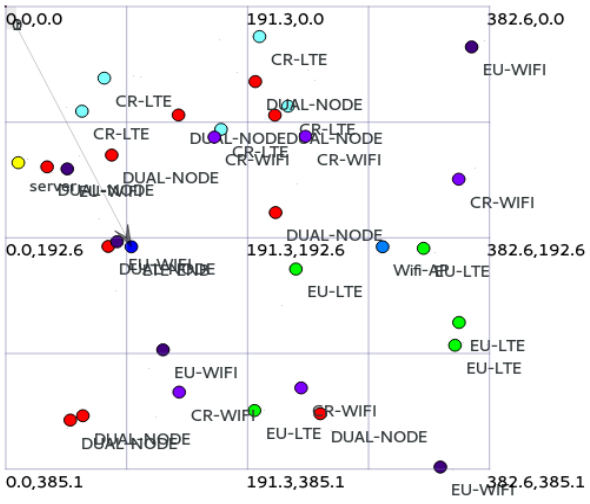
- Correr el comando ./Lte-Wifi.sh en el terminal
- Deberá aparecer la siguiente pantalla:

```
Waf: Entering directory `/home/pablo/ns-allinone-3.23/ns-3.23/build'
wscript:66: Warning: (in applications) Requested to build modular python
bindings, but apidefs dir not found => skipped the bindings.
  # these variables are mandatory ('/' are converted automatically)
wscript:310: Warning: (in lte) Requested to build modular python bindings, but
apidefs dir not found => skipped the bindings.
  env.append_value('LINKFLAGS', '-lgcov')
wscript:172: Warning: (in wifi) Requested to build modular python bindings, but
apidefs dir not found => skipped the bindings.
  help=('DEPRECATED (run ./waf shell)'),
Waf: Leaving directory `/home/pablo/ns-allinone-3.23/ns-3.23/build'
'build' finished successfully (4.781s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use
SetConstantPosition if it is stationary
```

```
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use
SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:23 Does not have a mobility model. Use
SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use
SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use
SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:23 Does not have a mobility model. Use
SetConstantPosition if it is stationary
Max Packets per trace file exceeded
```

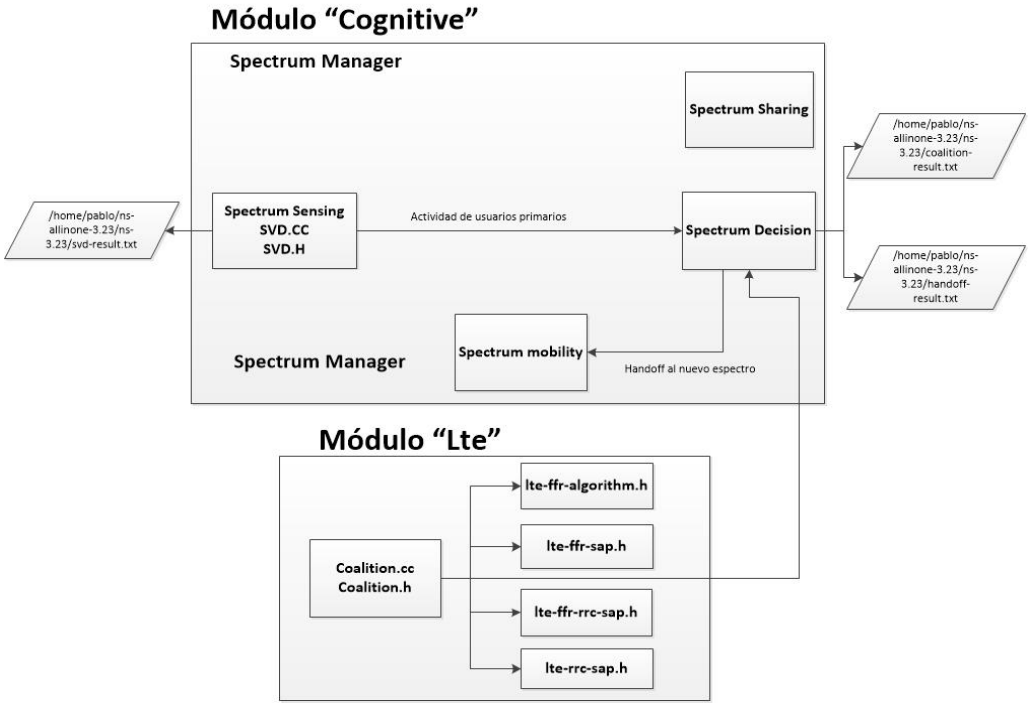
- Considerar que las advertencias mostradas corresponden a configuraciones de elementos no afectan a la simulación, además el aviso “Max packets per trace file exceded”, sólo es un indicador para el NetAnim.
- Verificar que se haya creado el archivo LTE-WIFI.xml y todos los archivos correspondientes a resultados y datos que se especificarán más adelante en la carpeta principal de NS3 (home/pablo/ns-allinone-3.23/ns-3.23).
- Abrir NetAnim.
- Cargar el archivo LTE-WIFI.xml en NetAmin
- Iniciar la simulación y recopilar toda la información que puede otorgar la herramienta NetAmin.

Movimiento de los nodos en NetAmin



5. Detalle del desarrollo:

Diagrama de flujo de algoritmos y módulos



Módulo cognitive:

- Contiene funciones del módulo "cognitivo" que ha sido desarrollado para ns3, estas funciones no son usadas en el desarrollo, no obstante, se sigue el mismo principio de ellas, la principal función añadida es el archivo "svd.cc" y "svd.h", que es el algoritmo de detección de uso de espectro especificado en el documento técnico.

Archivos
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/cognitive.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/cognitive.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/cognitive-packet-tags.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/cognitive-packet-tags.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/common-cognitive-header.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/pu-model.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/pu-model.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/repository.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/repository.h

/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-data.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-data.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-decision.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-decision.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-sensing.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/spectrum-sensing.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/svd.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/svd.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/cognitive/model/wscript

Módulo lte:

- Contiene todos los parámetros principales de la tecnología lte que se usará, adaptada a las funciones cognitivas del desarrollo
- Se encuentra los archivos “coalition.cc” y “coalition.h” algoritmo de teoría del juego encargado de la decisión de uso del espectro, especificado en el documento técnico.

Archivos
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/a2-a4-rsrq-handover-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/a2-a4-rsrq-handover-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/a3-rsrp-handover-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/a3-rsrp-handover-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/coalition.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/coalition.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-rrc-sap.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-rrc-sap.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-sap.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-sap.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-soft-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-ffr-soft-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-hard-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-hard-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-soft-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-soft-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-strict-algorithm.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-fr-strict-algorithm.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-handover-algorithm.cc

/home/pablo/ns-allinone-3.23/ns-3.23/src/lte/model/lte-handover-algorithm.h

Módulo wifi:

- Contiene todos los parámetros principales de la tecnología wifi que se usará, adaptada a las funciones cognitivas del desarrollo

Archivos
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/aarfcd-wifi-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/aarfcd-wifi-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ap-wifi-mac.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ap-wifi-mac.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/arf-wifi-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/arf-wifi-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ctrl-headers.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ctrl-headers.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/dcf-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/dcf-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/interference-helper.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/interference-helper.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/snr-tag.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/snr-tag.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ssid.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/ssid.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-channel.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-channel.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-mac.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-mac.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-net-device.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-net-device.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-phy.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-phy.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-remote-station-manager.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/wifi-remote-station-manager.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/yans-wifi-channel.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/yans-wifi-channel.h
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/yans-wifi-phy.cc
/home/pablo/ns-allinone-3.23/ns-3.23/src/wifi/model/yans-wifi-phy.h

Programa principal para generación de simulación LTE-WIFI.CC

Código principal donde se encuentran todos los parámetros básicos de la simulación que se detalla a continuación:

- En esta parte del código se inicializa todos los parámetros básicos del escenario de simulación, los cuales pueden ser modificados directamente en este archivo, o en el archivo “Lte-Wifi.sh”

```
/*
* Main Function
*/
int main (int argc , char * argv []) {
uint16 _t numberOfDualNodes = 10;// Number of dual nodes
uint16 _t numberOfEUL = 5; // Number of Primary User LTE
uint16 _t numberOfCUL = 5; // Number of User CR LTE
uint16 _t numberOfEUW = 5; // Number of Primary User WIFI
uint16 _t numberOfCUW = 5; // Number of User CR WIFI

// The number of dual nodes will be 10
uint16 _t numberOfeNBNodes = 1; // Number of eNB node for LTE
uint16 _t numberOfapNodes = 1; // Number of AP node for WIFI
double simTime = 5; // Simulating Time
double distance = 200.0; // Distance between eNB and AP
double interPacketInterval = 100; // Interval time for sending
packet
int nodeSpeed = 100; // Speed of Nodes in m/s
int nodePause = 0; // Pause time in sec
uint8 _t bandwidth = 25; // BandWidth of Network
uint8 _t upBW = 25; // Uplink BandWidth
uint8 _t downBW = 25; // Downlink BandWidth
bool generateSpectrumTrace = false ;
double eNBX , eNBY , eNBZ ; // position of eNB node for LTE
double ApX , ApY , ApZ; // position of AP node for WIFI
eNBX = eNBY = eNBZ = 100.0;
ApX = ApY = ApZ = 100.0;

double tx = 0.37; // transmisstion data power
double rx = 0.6; // received data power
double noise = 1e -15; // Noise is the -120 dBm
string trafficName , traffic ;
```

- En esta parte del código se da la opción de configuración de estos parámetros en un script para llamado inmediato del programa, que será el “Lte-Wifi.sh”.

```
// Command line arguments
CommandLine cmd ;
cmd. AddValue (" ENB", " Number of ENB", numberOfeNBNodes );
cmd. AddValue (" UserCrWifi ", " Number of UserCrWifi ", numberOfCUW );
cmd. AddValue (" UserCrLte ", " Number of UserCrLte ", numberOfCUL );
cmd. AddValue (" DualUserCr ", " Number of DualUserCr ",
numberOfDualNodes );
cmd. AddValue (" PrimaryUserWifi ", " Number of PrimaryUserWifi ",
numberOfEUW );
cmd. AddValue (" PrimaryUserLte ", " Number of PrimaryUserLte ",
numberOfEUL );

cmd. AddValue (" simTime ", " Total duration of the simulation [s])",
simTime );
cmd. AddValue (" NumberOfSample ", " Number of the sample for SVD
Matrix )", numberOfSample );
cmd. AddValue (" trafficName ", " Traffic Name of Network (TCP or
UDP)", traffic );
cmd. AddValue (" eNBX ", "X position of eNB node ", eNBX );
cmd. AddValue (" eNBY ", "Y position of eNB node ", eNBY );
cmd. AddValue (" eNBZ ", "Z position of eNB node ", eNBZ );
cmd. AddValue (" LteUpBandWidth ", " Upload bandwidth in LTE", upBW );
cmd. AddValue (" LteDownBandWidth ", " Download bandwidth in LTE ",
downBW );

cmd. AddValue (" ApX", "X position of AP node ", ApX );
cmd. AddValue (" ApY", "Y position of AP node ", ApY );
cmd. AddValue (" ApZ", "Z position of AP node ", ApZ );
cmd. AddValue (" WifiBandWidth ", " BandWidth in WIFI ", bandwidth );
```

- En los siguientes tramos del código principal se inicializan y configuran capa por capa los parámetros de los nodos y usuarios wifi y lte.

Script para inicialización de parámetros Lte-Wifi.sh

- Este script permite modificar los parámetros básicos de simulación según lo mencionado en el documento técnico.
- Para iniciar la simulación en el terminal se debe ejecutar este script

```
#!/bin/ bash
# ----------------------------------------------------#
# Run the LTE - WIFI project using this Script
# NOTE :
# Please set all parameters correctly !
# ----------------------------------------------------#
rm -f *.txt
./ waf --run "LTE - WIFI --ENB =3 -- UserCrLte =3 -- UserCrWifi =3
-- DualUserCr =15 -- PrimaryUserLte =8 -- PrimaryUserWifi =8
```

```
-- simTime =10 -- NumberOfSample =16000 -- trafficName =tcp
-- LteFrequency =729 -- WifiFrequency =2.4 -- LteUpBandWidth =20.0
-- LteDownBandWidth =20.0 -- WifiBandWidth =20.0 --TX =0.037 --RX =0.06
--Noise = -15 --ApX =200.0 --ApY =400.0 --ApZ =0.0 --eNBX =200.0
--eNBY =100.0 --eNBZ =0.0 -- ApRange =300 -- EnbRange =1000 "
```

La siguiente tabla muestra los parámetros que pueden ser cambiados y los rangos permitidos por el desarrollo y el software ns3:

Parámetros	Mínimo	Máximo
Frecuencia LTE	729Mhz	2170Mhz
Frecuencia WIFI	2400Mhz	5875Mhz
Celdas en Enb	1	3
Ancho de Banda LTE	10Mhz	60Mhz
Ancho de Banda WIFI	20Mhz	150Mhz
Ruido dbm	0dbm	-250dbm
Potencia de transmisión	0.025mW	1.27mW
Potencia de recepción	0.025mW	1.27mW
Usuarios CR LTE	0	100
Usuarios CR WIFI	0	100
Usuarios duales CR	0	100
Usuarios primarios LTE	0	100
Usuarios primarios WIFI	0	100
Muestras para detección	1000	30000
Rango de cobertura AP	200m	500m
Rango de cobertura eNB	350m	5000m
Tiempo de simulación	0	120 horas

6. Archivos de resultados

- Todos los datos sobre lte que resultan de la simulación se encuentran en los siguientes archivos:
  /home/pablo/ns-allinone-3.23/ns-3.23/DlPdcpStats.txt
  /home/pablo/ns-allinone-3.23/ns-3.23/DlRlcStats.txt
  /home/pablo/ns-allinone-3.23/ns-3.23/DlRsrpSinrStats.txt
  /home/pablo/ns-allinone-3.23/ns-3.23/DlRxPhyStats.txt
  /home/pablo/ns-allinone-3.23/ns-3.23/DlTxPhyStats.txt
  /home/pablo/ns-allinone-3.23/ns-3.23/UlPdcpStats.txt

/home/pablo/ns-allinone-3.23/ns-3.23/UlRlcStats.txt
/home/pablo/ns-allinone-3.23/ns-3.23/UlRsrpSinrStats.txt
/home/pablo/ns-allinone-3.23/ns-3.23/UlRxPhyStats.txt
/home/pablo/ns-allinone-3.23/ns-3.23/UlTxPhyStats.txt

- Los datos promedio de la red se encuentran en el archivo /home/pablo/ns-allinone-3.23/ns-3.23/wifi-network_perf.txt

Throughput (kbps): 30.376
DelayTime (sec): 0.019
JitterTime (sec): 0.003
PacketLoss: 52.357

- Los datos de handoff entre tecnología lte y wifi o viceversa se encuentran en el archivo /home/pablo/ns-allinone-3.23/ns-3.23/handoff-result.txt
- El handoff se realiza de acuerdo al RSSI, especificado en el documento técnico, la salida es un .txt al finalizar la simulación, que tiene la siguiente estructura:

Tiempo	Usuario que realiza el Handoff	Coalición a la que pertenece	Desde, valor de RSSI	Hacia, valor de RSSI

Handoff-result.txt

Time	User that makes the Handoff	Coalition to which belongs	From,RSSI	To,RSSI	Previous Channel	Current Channel
0.220000	CR-Dual-18	CR-Dual-9,CR-Dual-1,CR-Dual-16,CR-Dual-17,CR-Dual-13,CR-Dual-18,CR-Dual-15,CR-Dual-8,CR-Dual-4,CR-Dual-8,	Lte,33.746 ->	Wifi,36.123	lte	wifi
1.105.523	CR-Dual-4	CR-Dual-4,CR-Dual-11,CR-Dual-12,	Lte,29.050 ->	Wifi,34.868	lte	wifi

1.994.839	CR-Dual-16	CR-Dual-6,CR-Dual-7,CR-Dual-11,CR-Dual-13,CR-Dual-16,CR-Dual-5,CR-Dual-16,CR-Dual-17,	Wifi,26.136 ->	Lte,36.783	wifi	lte
2.921.562	CR-Dual-4	CR-Dual-0,CR-Dual-18,CR-Dual-4,CR-Dual-3,CR-Dual-15,CR-Dual-6,	Lte,28.820 ->	Wifi,36.807	lte	wifi
4.743.278	CR-Dual-16	CR-Dual-17,CR-Dual-10,CR-Dual-3,CR-Dual-16,CR-Dual-7,CR-Dual-16,CR-Dual-15,CR-Dual-11,	Lte,28.129 ->	Wifi,36.091	lte	wifi
5.211.961	CR-Dual-15	CR-Dual-15,CR-Dual-12,CR-Dual-7,	Lte,26.882 ->	Wifi,36.068	lte	wifi
7.628.540	CR-Dual-16	CR-Dual-14,CR-Dual-6,CR-Dual-19,CR-Dual-16,CR-Dual-12,CR-Dual-2,CR-Dual-19,CR-Dual-17,	Wifi,33.708 ->	Lte,36.471	wifi	lte
8.872.944	CR-Dual-3	CR-Dual-8,CR-Dual-15,CR-Dual-18,CR-Dual-3,CR-Dual-19,CR-Dual-11,CR-Dual-10,	Lte,27.262 ->	Wifi,34.256	lte	wifi
9.249.632	CR-Dual-17	CR-Dual-14,CR-Dual-10,CR-Dual-19,CR-Dual-5,CR-Dual-14,CR-Dual-17,CR-Dual-19,	Wifi,27.107 ->	Lte,35.143	wifi	lte
9.888.745	CR-Dual-1	CR-Dual-15,CR-Dual-1,CR-Dual-7,CR-Dual-7,	Wifi,27.408 ->	Lte,35.113	wifi	lte
9.9934.245	CR-Dual-7	CR-Dual-7,CR-Dual-1,CR-Dual-15,CR-Dual-13,CR-Dual-3,CR-Dual-17,CR-Dual-18,	Wifi,28.674 ->	Lte,36.420	wifi	lte

- Los datos de el algoritmo SVD según el tipo de usuario CR se encuentran en el archivo /home/pablo/ns-allinone-3.23/ns-3.23/svd-result.txt
- El algoritmo SVD detecta la presencia o no de un usuario primario, especificado en el documento técnico, la salida es un .txt al finalizar la simulación, que tiene la siguiente estructura:

Tiempo	Tipo de usuario	Detección (0 no detecto, 1 detecto)	Usuario Primario

Svd-result.txt

Time	Type of User	Detection (0 not detect, 1 detect)	Primary User
0.220000	CR-Lte-0	1	EU-Lte-5
1.105.523	CR-Dual-8	1	EU-Lte-8
1.994.839	CR-Wifi-0	1	EU-Wifi-5
2.488.930	CR-Wifi-0	1	EU-Wifi-5
2.921.562	CR-Dual-15	1	EU-Lte-9
3.266.540	CR-Lte-2	1	EU-Lte-3
3.538.929	CR-Lte-2	0	EU-Lte-4
3.958.482	CR-Lte-2	1	EU-Lte-8
4.743.278	CR-Dual-6	1	EU-Wifi-1
4.964.039	CR-Dual-17	1	EU-Wifi-3
5.211.961	CR-Dual-4	1	EU-Lte-1
5.536.223	CR-Dual-16	1	EU-Lte-8
6.354.093	CR-Lte-3	1	EU-Lte-7
7.085.685	CR-Lte-4	1	EU-Lte-6
7.628.540	CR-Dual-0	1	EU-Wifi-1
7.889.481	CR-Dual-4	1	EU-Wifi-7
8.159.196	CR-Lte-3	0	EU-Lte-7
8.872.944	CR-Wifi-0	0	EU-Wifi-4
9.249.632	CR-Dual-3	0	EU-Wifi-3
9.888.745	CR-Wifi-3	1	EU-Wifi-9
10.661.858	CR-Dual-14	1	EU-Wifi-2
11.550.767	CR-Lte-3	1	EU-Lte-2
12.237.540	CR-Dual-11	0	EU-Lte-5
12.619.910	CR-Dual-12	1	EU-Lte-4
13.548.115	CR-Lte-0	1	EU-Lte-1
13.762.521	CR-Wifi-2	0	EU-Wifi-8
14.207.478	CR-Wifi-4	1	EU-Wifi-3
Probability of Detect: 0.777778 Probability of false Detect: 0.222222			

- Los datos de el algoritmo de coalición se encuentran en el archivo /home/pablo/ns-allinone-3.23/ns-3.23/coalition-result.txt
- El algoritmo de coalición realiza las uniones o no de usuarios CR, dependiendo el tipo, especificado en el documento técnico, la salida es un .txt al finalizar la simulación, que tiene la siguiente estructura:

Tiempo	Coalicion	Cabeza de coalicion	Decisión de detección de la coalición	Tecnología

Coalition-result.txt

Time	Coalicion	Head of Coalicion	Detection of the coalition	Technology
0.272243	CR-Dual-9,CR-Dual-1,CR-Dual-16,CR-Dual-17,CR-Dual-13,CR-Dual-18,CR-Dual-15,CR-Dual-8,CR-Dual-4,CR-Dual-8,	CR-Dual-15	0	lte
1.497.116	CR-Dual-4,CR-Dual-11,CR-Dual-12,	CR-Dual-11	1	wifi
2.783.173	CR-Dual-6,CR-Dual-7,CR-Dual-11,CR-Dual-13,CR-Dual-16,CR-Dual-5,CR-Dual-16,CR-Dual-17,	CR-Dual-7	0	wifi
3.477.865	CR-Wifi-4,CR-Wifi-3,CR-Wifi-0,CR-Wifi-2,CR-Wifi-3,CR-Wifi-1,CR-Wifi-4,CR-Wifi-3,CR-Wifi-3,CR-Wifi-1,	CR-Wifi-1	1	wifi
3.058.317	CR-Dual-0,CR-Dual-18,CR-Dual-4,CR-Dual-3,CR-Dual-15,CR-Dual-6,	CR-Dual-15	0	wifi
3.769.874	CR-Wifi-0,CR-Wifi-2,CR-Wifi-4,CR-Wifi-3,	CR-Wifi-3	0	wifi
4.253.031	CR-Wifi-2,CR-Wifi-0,CR-Wifi-0,CR-Wifi-2,CR-Wifi-4,CR-Wifi-3,CR-Wifi-1,	CR-Wifi-3	1	wifi
4.524.573	CR-Wifi-4,CR-Wifi-3,CR-Wifi-4,CR-Wifi-0,CR-Wifi-4,CR-Wifi-4,CR-Wifi-3,CR-Wifi-2,CR-Wifi-4,CR-Wifi-0,	CR-Wifi-4	1	wifi
5.137.489	CR-Dual-17,CR-Dual-10,CR-Dual-3,CR-Dual-16,CR-Dual-7,CR-Dual-16,CR-Dual-15,CR-Dual-11,	CR-Dual-16	0	wifi
5.011.231	CR-Lte-0,CR-Lte-2,CR-Lte-1,CR-Lte-1,CR-Lte-3,	CR-Lte-0	1	lte
6.050.778	CR-Dual-15,CR-Dual-12,CR-Dual-7,	CR-Dual-12	0	wifi
6.463.371	CR-Wifi-4,CR-Wifi-3,CR-Wifi-4,CR-Wifi-2,CR-Wifi-4,CR-Wifi-4,CR-Wifi-4,CR-Wifi-2,	CR-Wifi-4	1	wifi

- Los datos de rendimiento de la red simulada en general se encuentran en el archivo /home/pablo/ns-allinone-3.23/ns-3.23/graph-information.txt

Graph-information.txt

No	ThroughPut(Kbps)	DelayTime(sec)	JitterTime(sec)	PacketLoss(%)
1	852.549	0.010001	0	0
2	852.549	0.0100018	0	0

3	852.549	0.0100027	0	0
4	852.549	0.0100036	0	0
5	852.549	0.0100045	0	0
6	852.549	0.0100053	0	0
7	852.549	0.0100062	0	0
8	852.549	0.0100071	0	0
9	852.548	0.010008	0	0
10	852.548	0.0100088	0	0
11	852.548	0.0100097	0	0
12	852.549	0.0100019	0	0
13	852.549	0.0100028	0	0
14	852.549	0.0100036	0	0

- Además existen archivos de salida para ser usados en el flow monitor de extensión .flomon y para ser usados por otros analizadores de extensión .pcap que se encuentran en /home/pablo/ns-allinone-3.23/ns-3.23

7. Ejemplos de uso de simulaciones

- Como primer ejemplo de uso, vamos a observar el movimiento de los terminales de NetAnim con parámetros especificados en el código Lte-wifi.sh

Primer ejemplo:

- Abrir el archivo Lte-wifi.sh, y configurarlo con los parámetros de la red deseados de acuerdo a la siguiente tabla de parámetros de entrada:

Parámetros	Configuración	Mínimo	Máximo
Frecuencia LTE	729Mhz	729Mhz	2170Mhz
Frecuencia WIFI	2400Mhz	2400Mhz	5875Mhz
Celdas en Enb	3	1	3
Ancho de Banda LTE	20Mhz	10Mhz	60Mhz
Ancho de Banda WIFI	20Mhz	20Mhz	150Mhz
Ruido dbm	-15dbm	0dbm	-250dbm
Potencia de transmisión	0.037mW	0.025mW	1.27mW
Potencia de recepción	0.06mW	0.025mW	1.27mW

Usuarios CR LTE	5	0	100
Usuarios CR WIFI	5	0	100
Usuarios duales CR	10	0	100
Usuarios primarios LTE	5	0	100
Usuarios primarios WIFI	5	0	100
Muestras para detección	16000	1000	30000
Rango de cobertura AP	200m	200m	500m
Rango de cobertura eNB	350m	350m	5000m
Tiempo de simulación	5 seg	0	120 horas

```
#!/bin/bash

#----------------------------------------------------#

#     Run the LTE-WIFI project using this Script

# NOTE:

#          Please set all parameters correctly!

#----------------------------------------------------#

rm -f *.txt

./waf --run "LTE-WIFI --ENB=3 --UserCrLte=5 --UserCrWifi=5 --DualUserCr=10 --
PrimaryUserLte=5 --PrimaryUserWifi=5 --simTime=5 --NumberOfSample=16000 --
trafficName=tcp --LteFrequency=729 --WifiFrequency=2.4 --LteUpBandWidth=20.0 --
LteDownBandWidth=20.0 --WifiBandWidth=20.0 --TX=0.037 --RX=0.06 --Noise=-15 --
ApX=200.0 --ApY=400.0 --ApZ=0.0 --eNBX=200.0 --eNBY=100.0 --eNBZ=0.0 --ApRange=200 --
EnbRange=350
```

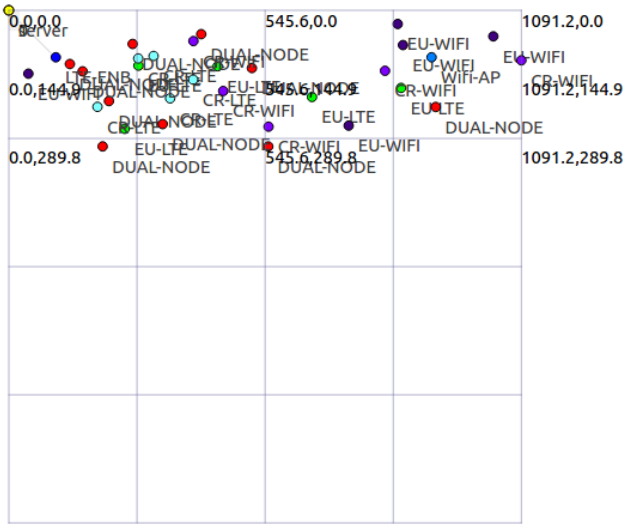
- Guardar los cambios
- Abrir el terminal y ejecutar el comando ./Lte-Wifi.sh

```
pablo@pablo-VirtualBox:~/ns-allinone-3.23/ns-3.23$ ./Lte-Wifi.sh
Waf: Entering directory `/home/pablo/ns-allinone-3.23/ns-3.23/build'
```

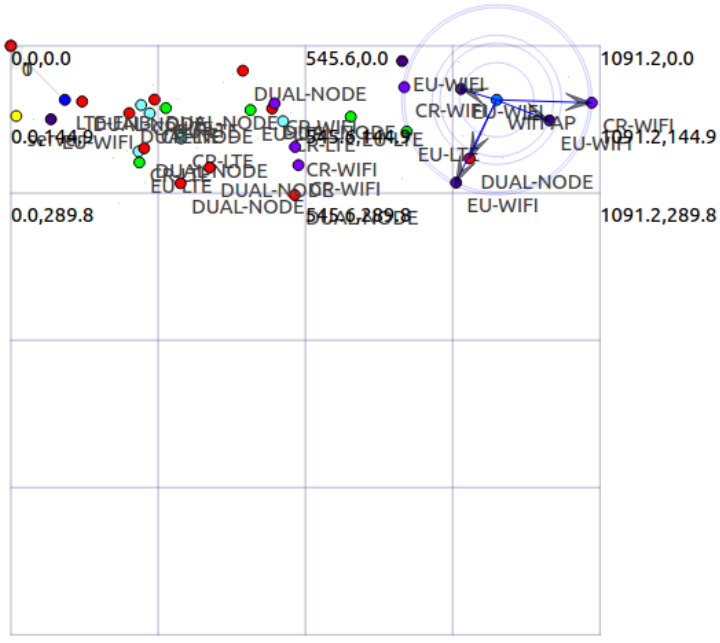
```
wscript:66: Warning: (in applications) Requested to build modular python bindings, but apidefs dir not found => skipped the bindings.
  # these variables are mandatory ('/' are converted automatically)
wscript:310: Warning: (in lte) Requested to build modular python bindings, but apidefs dir not found => skipped the bindings.
  env.append_value('LINKFLAGS', '-lgcov')
wscript:172: Warning: (in wifi) Requested to build modular python bindings, but apidefs dir not found => skipped the bindings.
  help=('DEPRECATED (run ./waf shell)'),
Waf: Leaving directory `/home/pablo/ns-allinone-3.23/ns-3.23/build'
'build' finished successfully (11.880s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:25 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:25 Does not have a mobility model. Use SetConstantPosition if it is stationary
```

- Esperar hasta que la simulación haya culminado, el tiempo de real de la simulación será determinado por el tiempo de simulación escogido, tal como se muestra en el texto anterior.
- Ejecutar el comando ./NetAnim en el terminal
- Una vez abierto NetAnim, dar clic en la parte superior izquierda del programa, en el ícono de carpeta, y escoger el archivo LTE-wifi.xml
- Iniciar el ambiente gráfico de la simulación dando clic en el ícono de play, se podrá observar el movimiento de los terminales a medida que la simulación transcurre.

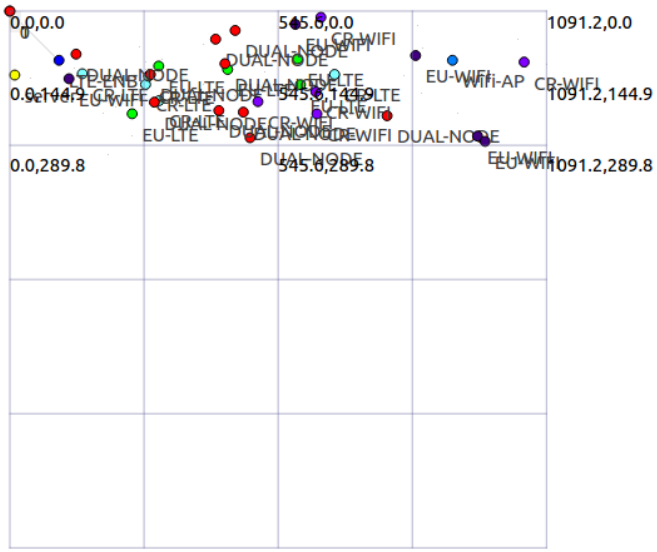
Movimiento de los terminales en NetAmin para ejemplo 1



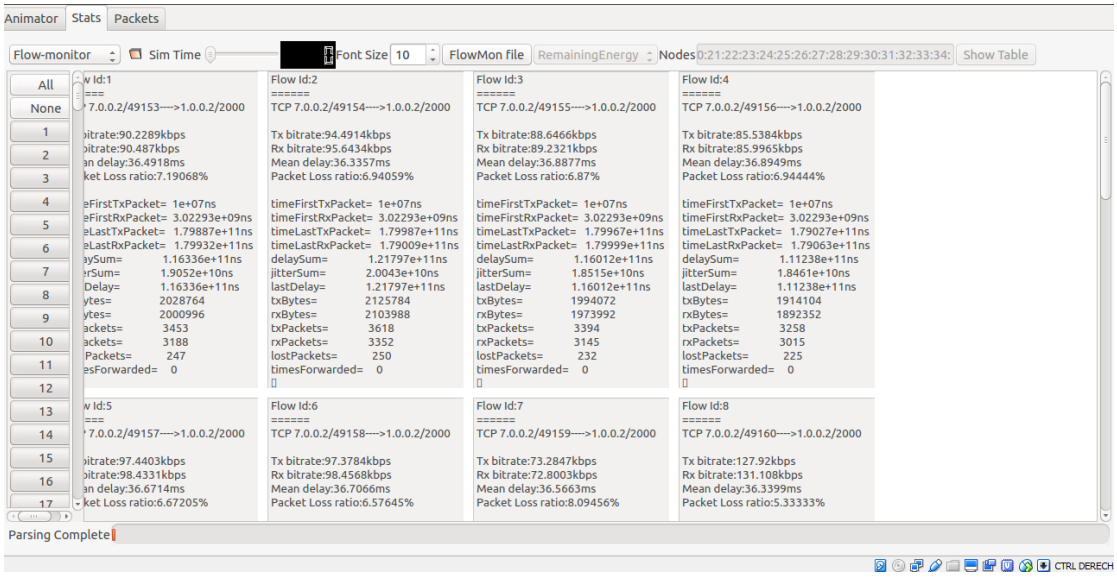
Movimiento de los terminales en NetAmin para ejemplo 1



Movimiento de los terminales en NetAmin para ejemplo 1



- En la pestaña de Stats se puede escoger la opción Flow- monitor, para observar estadística descriptiva y datos ip (jitter, delay, packet loss) de cada uno de los nodos de la simulación



- En la ruta /home/pablo/ns-allinone-3.23 se encuentran todos los archivos de resultados de la simulación, los cuales pueden ser usados para la investigación y pruebas que se quieran realizar.

Segundo ejemplo:

- Configuramos los parámetros como lo muestra la figura de acuerdo a la siguiente tabla, observando cambios con el ejemplo uno:

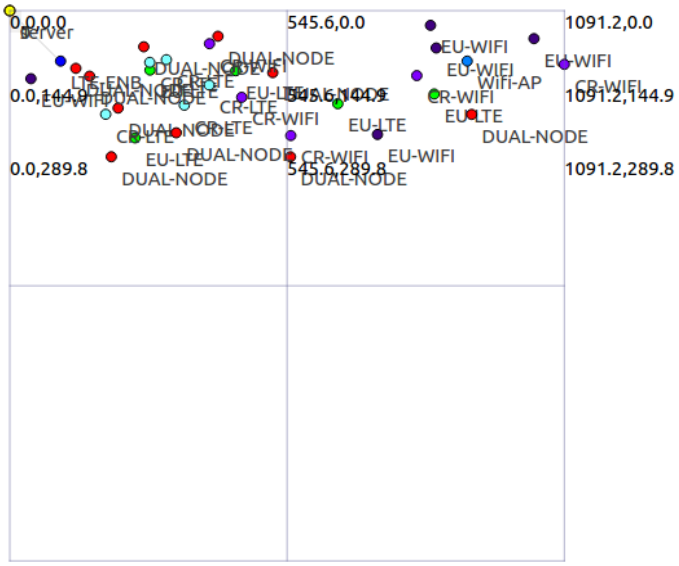
Parámetros	Configuración	Mínimo	Máximo
Frecuencia LTE	729Mhz	729Mhz	2170Mhz
Frecuencia WIFI	2400Mhz	2400Mhz	5875Mhz
Celdas en Enb	3	1	3
Ancho de Banda LTE	20Mhz	10Mhz	60Mhz
Ancho de Banda WIFI	20Mhz	20Mhz	150Mhz
Ruido dbm	-15dbm	0dbm	-250dbm
Potencia de transmisión	0.037mW	0.025mW	1.27mW
Potencia de recepción	0.06mW	0.025mW	1.27mW
Usuarios CR LTE	3	0	100
Usuarios CR WIFI	3	0	100
Usuarios duales CR	15	0	100
Usuarios primarios LTE	8	0	100
Usuarios primarios WIFI	8	0	100
Muestras para detección	16000	1000	30000
Rango de cobertura AP	300m	200m	500m
Rango de cobertura eNB	1000m	350m	5000m
Tiempo de simulación	10 seg	0	120 horas

```
#!/bin/bash

#-------------------------------------------------#

#     Run the LTE-WIFI project using this Script
```

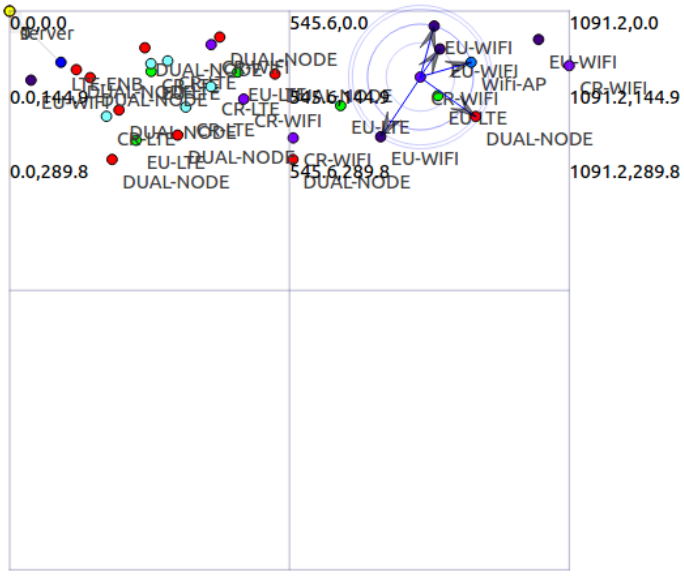
```
# NOTE:

#          Please set all parameters correctly!

#-----------------------------------------------------#

rm -f *.txt

./waf --run "LTE-WIFI --ENB=3 --UserCrLte=3 --UserCrWifi=3 --DualUserCr=15 --
PrimaryUserLte=8 --PrimaryUserWifi=8 --simTime=10 --NumberOfSample=16000 --
trafficName=tcp --LteFrequency=729 --WifiFrequency=2.4 --LteUpBandWidth=20.0 --
LteDownBandWidth=20.0 --WifiBandWidth=20.0 --TX=0.037 --RX=0.06 --Noise=-15 --
ApX=200.0 --ApY=400.0 --ApZ=0.0 --eNBX=200.0 --eNBY=100.0 --eNBZ=0.0 --ApRange=300 --
EnbRange=1000"
```

- Guardamos los cambios y ejecutamos como en el ejemplo 1
- Esperamos hasta finalizar la simulación
- Ejecutamos NetAnim como en el ejemplo 1
- Iniciar el ambiente gráfico de la simulación dando clic en el ícono de play, se podrá observar la diferencia con el ejemplo 1 en cuanto a cantidad de usuarios y ubicación.

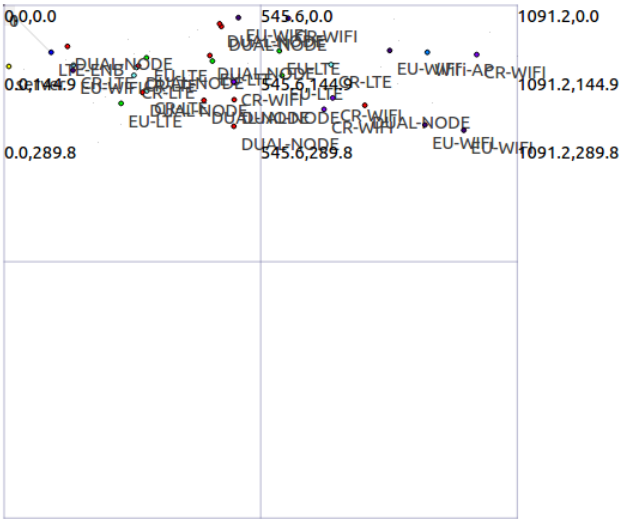
Movimiento de los terminales en NetAmin para ejemplo 2



Movimiento de los terminales en NetAmin para ejemplo 2



Movimiento de los terminales en NetAmin para ejemplo 2



- Los archivos de datos de resultados obtenidos en la simulación son sobrescritos en la ruta /home/pablo/ns-allinone-3.23

Tercer ejemplo:

- En este ejemplo se van a variar la cantidad de usuarios primarios y secundarios, esto es para demostrar la extensión del simulador y futuros trabajos.
- Configuramos los parámetros como lo muestra la figura de acuerdo a la siguiente tabla:

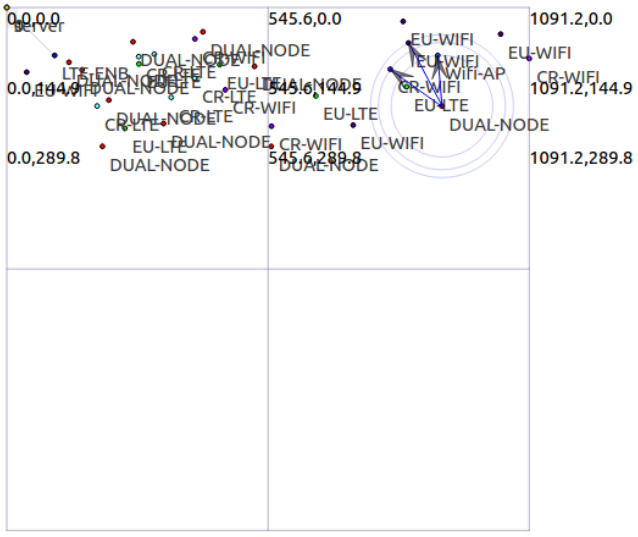
Parámetros	Configuración	Mínimo	Máximo
Frecuencia LTE	729Mhz	729Mhz	2170Mhz
Frecuencia WIFI	2400Mhz	2400Mhz	5875Mhz
Celdas en Enb	3	1	3
Ancho de Banda LTE	20Mhz	10Mhz	60Mhz
Ancho de Banda WIFI	20Mhz	20Mhz	150Mhz
Ruido dbm	-15dbm	0dbm	-250dbm
Potencia de transmisión	0.037mW	0.025mW	1.27mW
Potencia de recepción	0.06mW	0.025mW	1.27mW
Usuarios CR LTE	5	0	100
Usuarios CR WIFI	5	0	100
Usuarios duales CR	20	0	100
Usuarios primarios LTE	10	0	100
Usuarios primarios WIFI	10	0	100
Muestras para detección	16000	1000	30000
Rango de cobertura AP	300m	200m	500m
Rango de cobertura eNB	1000m	350m	5000m
Tiempo de simulación	15 seg	0	120 horas

```
#!/bin/bash

#--------------------------------------------------#

#     Run the LTE-WIFI project using this Script
```

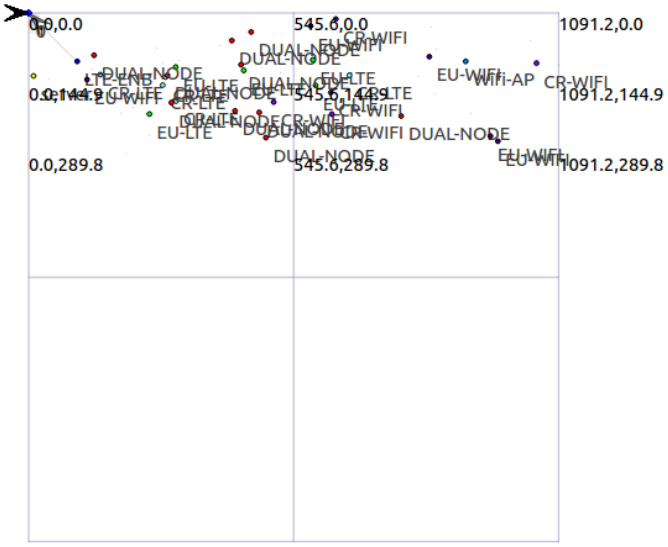
```
# NOTE:

#          Please set all parameters correctly!

#-----------------------------------------------------#

rm -f *.txt

./waf --run "LTE-WIFI --ENB=3 --UserCrLte=5 --UserCrWifi=5 --DualUserCr=20 --PrimaryUserLte=10 --PrimaryUserWifi=10 --simTime=15 --NumberOfSample=16000 --trafficName=tcp --LteFrequency=729 --WifiFrequency=2.4 --LteUpBandWidth=20.0 --LteDownBandWidth=20.0 --WifiBandWidth=20.0 --TX=0.037 --RX=0.06 --Noise=-15 --ApX=200.0 --ApY=400.0 --ApZ=0.0 --eNBX=200.0 --eNBY=100.0 --eNBZ=0.0 --ApRange=300 --EnbRange=1000"
```

- Guardamos los cambios y ejecutamos como en los ejemplos anteriores.
- Esperamos hasta finalizar la simulación.
- Ejecutamos NetAnim como en los ejemplos anteriores
- Iniciar el ambiente gráfico de la simulación dando clic en el ícono de play.

Movimiento de los terminales en NetAmin para ejemplo 3



Movimiento de los terminales en NetAmin para ejemplo 3



- Además observamos los resultados en el archivo de salida /home/pablo/ns-allinone-3.23/ns-3.23/svd-result.txt, con estos datos podríamos obtener modelos del método de detección variando el número de usuarios y su generalización.

Svd-result.txt

Time	Type of User	Detection (0 not detect, 1 detect)	Primary User
0.220000	CR-Lte-0	1	EU-Lte-5
1.105523	CR-Dual-8	1	EU-Lte-8
1.994839	CR-Wifi-0	1	EU-Wifi-5
2.488930	CR-Wifi-0	1	EU-Wifi-5
2.921562	CR-Dual-15	1	EU-Lte-9
3.266540	CR-Lte-2	1	EU-Lte-3
3.538929	CR-Lte-2	0	EU-Lte-4
3.958482	CR-Lte-2	1	EU-Lte-8
4.743278	CR-Dual-6	1	EU-Wifi-1
4.964039	CR-Dual-17	1	EU-Wifi-3
5.211961	CR-Dual-4	1	EU-Lte-1
5.536223	CR-Dual-16	1	EU-Lte-8
6.354093	CR-Lte-3	1	EU-Lte-7
7.085685	CR-Lte-4	1	EU-Lte-6
7.628540	CR-Dual-0	1	EU-Wifi-1
7.889481	CR-Dual-4	1	EU-Wifi-7
8.159196	CR-Lte-3	0	EU-Lte-7
8.872944	CR-Wifi-0	0	EU-Wifi-4
9.249632	CR-Dual-3	0	EU-Wifi-3
9.888745	CR-Wifi-3	1	EU-Wifi-9

10.661858	CR-Dual-14	1	EU-Wifi-2
11.550767	CR-Lte-3	1	EU-Lte-2
12.237540	CR-Dual-11	0	EU-Lte-5
12.619910	CR-Dual-12	1	EU-Lte-4
13.548115	CR-Lte-0	1	EU-Lte-1
13.762521	CR-Wifi-2	0	EU-Wifi-8
14.207478	CR-Wifi-4	1	EU-Wifi-3

Probability of Detect: 0.777778 Probability of false Detect: 0.222222

D. Códigos y Scripts

1. Script para inicialización de parámetros del simulador

```
#!/bin/bash
#-------------------------------------------------------#
#      Run the LTE-WIFI project using this Script
# NOTE:
# Please set all parameters correctly!
#-------------------------------------------------------#
rm -f *.txt
./waf --run "LTE-WIFI --ENB=3 --UserCrLte=3 --UserCrWifi=3
   --DualUserCr=15 --PrimaryUserLte=8 --PrimaryUserWifi=8
   --simTime=10 --NumberOfSample=16000 --trafficName=tcp
   --LteFrequency=729 --WifiFrequency=2.4 --LteUpBandWidth=20.0
   --LteDownBandWidth=20.0 --WifiBandWidth=20.0 --TX=0.037 --RX=0.06
   --Noise=-15 --ApX=200.0 --ApY=400.0 --ApZ=0.0 --eNBX=200.0
   --eNBY=100.0 --eNBZ=0.0 --ApRange=300 --EnbRange=1000"
```

codigo/Lte–Wifi.C

2. Script principal para generación de simulación

```
#include <iostream>
#include "ns3/netanim-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/aodv-module.h"
#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/spectrum-module.h>
#include <ns3/cognitive-module.h>
#include <cmath>
#include <string>
```

```
#include <cassert>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "ns3/basic-energy-source.h"
#include "ns3/wifi-radio-energy-model.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/ipv4-flow-classifier.h"
#include "ns3/uinteger.h"

using namespace ns3;
using namespace std;

#define max(a, b) ((a) > (b) ? (a) : (b))

NS_LOG_COMPONENT_DEFINE ("LTE-WIFI");

/*
 * Main Function
 */
int main(int argc, char *argv[]) {
  uint16_t numberOfDualNodes = 10;// Number of dual nodes
  uint16_t numberOfEUL = 5; // Number of Primary User LTE
  uint16_t numberOfCUL = 5; // Number of User CR LTE
  uint16_t numberOfEUW = 5; // Number of Primary User WIFI
  uint16_t numberOfCUW = 5; // Number of User CR WIFI

  // The number of dual nodes will be 10
  uint16_t numberOfeNBNodes = 1;  // Number of eNB node for LTE
  uint16_t numberOfapNodes = 1; // Number of AP node for WIFI
  double simTime = 5;   // Simulating Time
  double distance = 200.0;    // Distance between eNB and AP
  double interPacketInterval = 100; // Interval time for sending
     packet
  int nodeSpeed = 100;      // Speed of Nodes in m/s
  int nodePause = 0;      // Pause time in sec
  uint8_t bandwidth = 25;     // BandWidth of Network
  uint8_t upBW = 25;      // Uplink BandWidth
  uint8_t downBW = 25;      // Downlink BandWidth
  bool generateSpectrumTrace = false;
  double eNBX, eNBY, eNBZ;    // position of eNB node for LTE
  double ApX, ApY, ApZ;     // position of AP node for WIFI
  eNBX = eNBY = eNBZ = 100.0;
  ApX = ApY = ApZ = 100.0;

  double tx = 0.37;   // transmisstion data power
  double rx = 0.6;    // received data power
```

```
72  double noise = 1e-15;   // Noise is the -120dBm
73  string trafficName, traffic;
74
75  uint32_t wifiFrequency = 2.4;
76  uint32_t lteFrequency = 100;
77  double ApRange = 250.0;
78  double EnbRange = 500.0;
79  time_t startTd, endTd;
80
81  int numberOfSample = 16000;
82
83  char dtbuf[256] = { 0 };
84  FILE* timeFp = fopen("simulating-time.txt", "a+");
85  time(&startTd);
86        strftime(dtbuf, sizeof dtbuf, "%A %b %d %H:%M:%S %Y",
            localtime(&startTd));
87  fprintf(timeFp, "StartTime: %s\n", dtbuf);
88
89  SeedManager::SetSeed((unsigned int)time(NULL));
90
91  // Command line arguments
92  CommandLine cmd;
93  cmd.AddValue("ENB", "Number of ENB", numberOfeNBNodes);
94  cmd.AddValue("UserCrWifi", "Number of UserCrWifi", numberOfCUW);
95  cmd.AddValue("UserCrLte", "Number of UserCrLte", numberOfCUL);
96  cmd.AddValue("DualUserCr", "Number of DualUserCr",
      numberOfDualNodes);
97  cmd.AddValue("PrimaryUserWifi", "Number of PrimaryUserWifi",
      numberOfEUW);
98  cmd.AddValue("PrimaryUserLte", "Number of PrimaryUserLte",
      numberOfEUL);
99
100 cmd.AddValue("simTime", "Total duration of the simulation [s])",
      simTime);
101 cmd.AddValue("NumberOfSample", "Number of the sample for SVD
      Matrix)", numberOfSample);
102 cmd.AddValue("trafficName", "Traffic Name of Network (TCP or
      UDP)", traffic);
103 cmd.AddValue("eNBX", "X position of eNB node", eNBX);
104 cmd.AddValue("eNBY", "Y position of eNB node", eNBY);
105 cmd.AddValue("eNBZ", "Z position of eNB node", eNBZ);
106 cmd.AddValue("LteUpBandWidth", "Upload bandwidth in LTE", upBW);
107 cmd.AddValue("LteDownBandWidth", "Download bandwidth in LTE",
      downBW);
108
109 cmd.AddValue("ApX", "X position of AP node", ApX);
110 cmd.AddValue("ApY", "Y position of AP node", ApY);
111 cmd.AddValue("ApZ", "Z position of AP node", ApZ);
112 cmd.AddValue("WifiBandWidth", "BandWidth in WIFI", bandwidth);
```

```
113
114   cmd.AddValue("TX", "Transmission data power", tx);
115   cmd.AddValue("RX", "Received data power", rx);
116   cmd.AddValue("Noise", "Noise of Network", noise);
117
118   cmd.AddValue("WifiFrequency", "Wifi Frequency (GHz)",
          wifiFrequency);
119   cmd.AddValue("LteFrequency", "Lte Frequency (MHz)", lteFrequency);
120   cmd.AddValue("ApRange", "Ap cover area (m)", ApRange);
121   cmd.AddValue("EnbRange", "Enb cover area (m)", EnbRange);
122
123   cmd.Parse(argc, argv);
124   trafficName = "udp";
125
126   // set the SrsPeriodicity of Lte
127   Config::SetDefault("ns3::LteEnbRrc::SrsPeriodicity",
          UintegerValue(320));
128
129   // create the Lte helper
130   Ptr < LteHelper > lteHelper = CreateObject<LteHelper>();
131   Ptr < PointToPointEpcHelper > epcHelper =
132       CreateObject<PointToPointEpcHelper>();
133   lteHelper->SetEpcHelper(epcHelper);
134
135   ConfigStore inputConfig;
136   inputConfig.ConfigureDefaults();
137   cmd.Parse(argc, argv);
138
139   Ptr < Node > pgw = epcHelper->GetPgwNode();
140
141   // create the remote host container
142   NodeContainer remoteHostContainer;
143   remoteHostContainer.Create(1);
144   Ptr < Node > remoteHost = remoteHostContainer.Get(0);
145   InternetStackHelper internet;   // create the internet stack helper
146   internet.Install(remoteHostContainer);  // install the remote host
          to internet stack
147
148 //====================================
149   // set the properties to Lte helper
150   lteHelper->SetSchedulerType("ns3::PfFfMacScheduler");
151   lteHelper->SetSchedulerAttribute("UlCqiFilter",
152       EnumValue(FfMacScheduler::PUSCH_UL_CQI));
153   lteHelper->SetEnbDeviceAttribute("DlBandwidth",
          UintegerValue(downBW));
154   lteHelper->SetEnbDeviceAttribute("UlBandwidth",
          UintegerValue(upBW));
155
156 #ifdef ENB
```

```
157   lteHelper->SetEnbDeviceAttribute("Range", DoubleValue(EnbRange));
158   SVD::L = numberOfSample;
159 #endif
160
161   // Create the Internet
162   PointToPointHelper p2ph;
163   p2ph.SetDeviceAttribute("DataRate",
          DataRateValue(DataRate("100Gb/s")));
164   p2ph.SetDeviceAttribute("Mtu", UintegerValue(1500));
165   p2ph.SetChannelAttribute("Delay", TimeValue(Seconds(0.010)));
166   NetDeviceContainer internetDevices = p2ph.Install(pgw, remoteHost);
167
168   // create the Ipv4Address Helper
169   Ipv4AddressHelper ipv4h;
170   ipv4h.SetBase("1.0.0.0", "255.0.0.0");  // set the base network
          address
171   Ipv4InterfaceContainer internetIpIfaces =
          ipv4h.Assign(internetDevices);
172   Ipv4Address remoteHostAddr = internetIpIfaces.GetAddress(1);  //
          set the remote host address
173   Ipv4StaticRoutingHelper ipv4RoutingHelper;
174   Ptr < Ipv4StaticRouting > remoteHostStaticRouting =   // set the
          routing of host
175       ipv4RoutingHelper.GetStaticRouting(remoteHost->GetObject<Ipv4>());
176   remoteHostStaticRouting->AddNetworkRouteTo(Ipv4Address("7.0.0.0"),
          // add the routing table to host
177       Ipv4Mask("255.0.0.0"), 1);
178   p2ph.EnablePcapAll("Lte-module"); // enable the pcap log
179
180   // node containers (dual nodes, primary lte user and cr lte user)
181   NodeContainer EULNodes;
182   NodeContainer CRLNodes;
183   NodeContainer dualNodes;
184   EULNodes.Create(numberOfEUL);   // create the primary lte nodes
185   CRLNodes.Create(numberOfCUL);   // create the cr lte nodes
186   dualNodes.Create(numberOfDualNodes);  // create the dule nodes
187
188   NodeContainer ueNodes(EULNodes, CRLNodes, dualNodes); // create
          the ue nodecontainer
189   NodeContainer enbNodes;         // create the enb nodescontainer
190   enbNodes.Create(numberOfeNBNodes);      // create the enb nodes
191
192   // set the schedular to Lte helper
193   lteHelper->SetSchedulerType("ns3::PfFfMacScheduler");
194   std::string frAlgorithmType = lteHelper->GetFfrAlgorithmType();
195   NS_LOG_DEBUG("FrAlgorithmType: " << frAlgorithmType);
196
197   // create and initial the coalition module
198   if (frAlgorithmType == "ns3::coalition") {
```

```
199     lteHelper->SetSchedulerAttribute("UlCqiFilter",
200         EnumValue(FfMacScheduler::PUSCH_UL_CQI));
201     lteHelper->SetEnbDeviceAttribute("DlBandwidth",
           UintegerValue(downBW));
202     lteHelper->SetEnbDeviceAttribute("UlBandwidth",
           UintegerValue(upBW));
203   }
204
205   lteHelper->SetHandoverAlgorithmType
        ("ns3::A2A4RsrqHandoverAlgorithm");
206   lteHelper->SetHandoverAlgorithmAttribute ("ServingCellThreshold",
207                                             UintegerValue (30));
208   lteHelper->SetHandoverAlgorithmAttribute ("NeighbourCellOffset",
209                                             UintegerValue (1));
210
211   // create the mobility model
212   MobilityHelper mobility1;
213
214   // Install Mobility Model
215   Ptr < ListPositionAllocator > positionAlloc = CreateObject<
216       ListPositionAllocator>();
217   for (uint16_t i = 0; i < numberOfeNBNodes; i++) {
218     positionAlloc->Add(Vector(distance * i, 0, 0));
219   }
220
221   mobility1.SetMobilityModel("ns3::ConstantPositionMobilityModel");
222   mobility1.SetPositionAllocator(positionAlloc);
223   MobilityHelper mobility;
224   int64_t streamIndex = 0; // used to get consistent mobility across
        scenarios
225
226   ObjectFactory pos;
227   pos.SetTypeId("ns3::RandomRectanglePositionAllocator");
228
229   char buf[512] = { 0 };
230   sprintf(buf, "ns3::UniformRandomVariable[Min=0.0|Max=%f]",
        max(ApX, eNBX) + 200);
231   pos.Set("X", StringValue(buf));
232
233   memset(buf, 0, 512);
234   sprintf(buf, "ns3::UniformRandomVariable[Min=0.0|Max=%f]",
        max(ApY, eNBY) + 200);
235   pos.Set("Y", StringValue(buf));
236
237   Ptr < PositionAllocator > taPositionAlloc =
        pos.Create()->GetObject<
238       PositionAllocator>();
239   streamIndex += taPositionAlloc->AssignStreams(streamIndex);
240   std::stringstream ssSpeed;
```

```
241   ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << nodeSpeed
         << "]";
242   std::stringstream ssPause;
243   ssPause << "ns3::ConstantRandomVariable[Constant=" << nodePause <<
         "]";
244   mobility.SetMobilityModel("ns3::RandomWaypointMobilityModel",
         "Speed",
245       StringValue(ssSpeed.str()), "Pause",
             StringValue(ssPause.str()),
246       "PositionAllocator", PointerValue(taPositionAlloc));
247   mobility.SetPositionAllocator(taPositionAlloc); // set the
         position allocation
248   mobility.Install(ueNodes);  // install the ue nodes to mobility
         model
249   mobility1.Install(enbNodes);  // install the enb node to mobility
         model
250   NetDeviceContainer enbLteDevs =
         lteHelper->InstallEnbDevice(enbNodes);  // install the enb node
         device
251   NetDeviceContainer ueLteDevs =
         lteHelper->InstallUeDevice(ueNodes); // install the ue nodes
         device
252   internet.Install(ueNodes);
253   Ipv4InterfaceContainer ueIpIface;
254   ueIpIface =
         epcHelper->AssignUeIpv4Address(NetDeviceContainer(ueLteDevs));
255
256   // install the routing table to ue nodes
257   for (uint32_t u = 0; u < ueNodes.GetN(); ++u) {
258     Ptr < Node > ueNode = ueNodes.Get(u);
259     Ptr < Ipv4StaticRouting > ueStaticRouting =
260         ipv4RoutingHelper.GetStaticRouting(ueNode->GetObject<Ipv4>());
261     ueStaticRouting->SetDefaultRoute(
262         epcHelper->GetUeDefaultGatewayAddress(), 1);
263   }
264
265   // add the ue nodes to enb node
266   for (uint32_t i = 0; i < numberOfeNBNodes; i ++) {
267     uint32_t numUe = ueNodes.GetN();
268     for (uint16_t j = numUe / numberOfeNBNodes * i; j < numUe /
            numberOfeNBNodes * (i + 1); j++) {
269       lteHelper->Attach(ueLteDevs.Get(j), enbLteDevs.Get(i));
270     }
271   }
272
273   // create the spectrum Analyzer Node
274   NodeContainer spectrumAnalyzerNodes;
275   spectrumAnalyzerNodes.Create(1);
276   SpectrumAnalyzerHelper spectrumAnalyzerHelper;
```

```
277
278   if (generateSpectrumTrace) {
279     Ptr < ListPositionAllocator > positionAlloc = CreateObject<
280         ListPositionAllocator>();
281
282     positionAlloc->Add(Vector(distance * 0.5, distance * 0.866,
          0.0)); // eNB3
283
284     // create the mobility model of spectrumAnalyzerNode
285     MobilityHelper mobility;
286     mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
287     mobility.SetPositionAllocator(positionAlloc);
288     mobility.Install(spectrumAnalyzerNodes);
289
290     Ptr < LteSpectrumPhy > enbDlSpectrumPhy =
291         enbNodes.Get(0)->GetObject<LteEnbNetDevice>()->GetPhy()->GetDownlinkSp
292             LteSpectrumPhy>();
293     Ptr < SpectrumChannel > dlChannel =
          enbDlSpectrumPhy->GetChannel();
294
295     // set channel to spectrumAnalyzerNode
296     spectrumAnalyzerHelper.SetChannel(dlChannel);
297     Ptr < SpectrumModel > sm =
          LteSpectrumValueHelper::GetSpectrumModel(lteFrequency,
298         bandwidth);
299     spectrumAnalyzerHelper.SetRxSpectrumModel(sm);
300     spectrumAnalyzerHelper.SetPhyAttribute("Resolution",
301         TimeValue(MicroSeconds(10)));
302     spectrumAnalyzerHelper.SetPhyAttribute("NoisePowerSpectralDensity",
303         DoubleValue(noise));
304     spectrumAnalyzerHelper.EnableAsciiAll("spectrum-analyzer-output");
305     spectrumAnalyzerHelper.Install(spectrumAnalyzerNodes);
306   }
307
308   lteHelper->EnableTraces();  // enable the trace of lte
309
310   // create the nodecontainer (primary wifi user, cr wifi user)
311   NodeContainer EUWNodes;
312   NodeContainer CRWNodes;
313   EUWNodes.Create(numberOfEUW); // create the primary wifi user nodes
314   CRWNodes.Create(numberOfCUW); // create the cr wifi user nodes
315
316   Config::SetDefault ("ns3::RangePropagationLossModel::MaxRange",
        DoubleValue (ApRange));
317
318   // create the wifi nodes
319   NodeContainer wifiNodes(EUWNodes, CRWNodes, dualNodes);
320
321   // create the wifi Ap nodecontainer
```

```
322   NodeContainer wifiApNode;
323   wifiApNode.Create(numberOfapNodes); // create the AP node
324
325   // nodecontainer of all nodes for wifi
326   NodeContainer allWifiNodes(EUWNodes, CRWNodes, wifiApNode);
327
328   YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
          // create the wifi channel helper
329   channel.AddPropagationLoss ("ns3::RangePropagationLossModel");
330
331   YansWifiPhyHelper phy = YansWifiPhyHelper::Default();     //
         create the wifi phy helper
332   phy.SetChannel(channel.Create()); // set the channel to phy
333   phy.Set("Frequency", UintegerValue(wifiFrequency * 1024 * 1024));
334
335   WifiHelper wifi = WifiHelper::Default();  // create the wifi helper
336   wifi.SetRemoteStationManager("ns3::AarfWifiManager"); // set the
         remote station manager for wifi
337   NqosWifiMacHelper mac = NqosWifiMacHelper::Default(); // create
         the wifi mac layer for wifi
338   Ssid ssid = Ssid("ns-3-ssid");        // set the ssid for wifi
339   mac.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssid),
         "ActiveProbing",
340       BooleanValue(false));
341
342   // create the sta device for wifi nodes
343   NetDeviceContainer staDevices;
344   staDevices = wifi.Install(phy, mac, wifiNodes);   // install the
         wifi nodes to sta device
345   mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssid)); // set
         type to mac layer
346
347   // create the ap device for wifi node
348   NetDeviceContainer apDevices;
349   apDevices = wifi.Install(phy, mac, wifiApNode); // install the ap
         node to ap device
350   mobility1.Install(wifiApNode);  // install the wifi AP to mobility
         model
351   mobility.Install(EUWNodes); // install the primary wifi user to
         mobility model
352   mobility.Install(CRWNodes); // install the cr wifi user to
         mobility model
353
354   InternetStackHelper stack;  // create the internet stack
355   stack.Install(allWifiNodes);  // install the wifi nodes to
         internet stack
356
357   // create the Ipv4AddressHelper
358   Ipv4AddressHelper address;
```

```
359   address.SetBase("10.1.1.0", "255.255.255.0"); // set the base
          network address
360   Ipv4InterfaceContainer staInterfaces;
361
362   // assign all nodes to ipv4
363   staInterfaces = address.Assign(staDevices);
364   Ipv4InterfaceContainer apInterface;
365   apInterface = address.Assign(apDevices);
366
367   // create the basic energy source
368   Ptr < BasicEnergySource > energySource =
          CreateObject<BasicEnergySource>();
369   Ptr < WifiRadioEnergyModel > energyModel =
370       CreateObject<WifiRadioEnergyModel>();
371   energySource->SetInitialEnergy(3000);     // init the energy model
372   energyModel->SetEnergySource(energySource);   // set the energy
          source to nodes
373   energySource->AppendDeviceEnergyModel(energyModel); // append the
          energy model
374   energyModel->SetTxCurrentA(tx);       // set the transmisstion
          power
375   energyModel->SetRxCurrentA(rx);       // set the received power
376
377   phy.EnablePcapAll("Wifi-module");     // enable the pcap log
378   uint16_t dlPort = 1000;   // download port
379   uint16_t ulPort = 2000;   // upload port
380   uint16_t otherPort = 3000;  // other port
381   ApplicationContainer clientApps;// applications container for
          client app
382   ApplicationContainer serverApps;// applications container for
          server app
383
384   // install the application to nodes
385   for (uint32_t u = 0; u < ueNodes.GetN(); ++u) {
386     if (trafficName == "udp") { // traffic mode UDP
387       ++ulPort;
388       ++otherPort;
389       PacketSinkHelper dlPacketSinkHelper("ns3::UdpSocketFactory",
390           InetSocketAddress(Ipv4Address::GetAny(), dlPort),
                numberOfDualNodes, numberOfEUL, numberOfCUL,
                numberOfEUW, numberOfCUW, simTime);
391       PacketSinkHelper ulPacketSinkHelper("ns3::UdpSocketFactory",
392           InetSocketAddress(Ipv4Address::GetAny(), ulPort),
                numberOfDualNodes, numberOfEUL, numberOfCUL,
                numberOfEUW, numberOfCUW, simTime);
393       PacketSinkHelper packetSinkHelper("ns3::UdpSocketFactory",
394           InetSocketAddress(Ipv4Address::GetAny(), otherPort),
                numberOfDualNodes, numberOfEUL, numberOfCUL,
                numberOfEUW, numberOfCUW, simTime);
```

```
395         serverApps.Add(dlPacketSinkHelper.Install(ueNodes.Get(u)));
396         serverApps.Add(ulPacketSinkHelper.Install(remoteHost));
397         serverApps.Add(packetSinkHelper.Install(ueNodes.Get(u)));
398
399         UdpClientHelper dlClient(ueIpIface.GetAddress(u), dlPort);
400         dlClient.SetAttribute("Interval",
401             TimeValue(MilliSeconds(interPacketInterval)));
402         dlClient.SetAttribute("MaxPackets", UintegerValue(1000000));
403
404         UdpClientHelper ulClient(remoteHostAddr, ulPort);
405         ulClient.SetAttribute("Interval",
406             TimeValue(MilliSeconds(interPacketInterval)));
407         ulClient.SetAttribute("MaxPackets", UintegerValue(1000000));
408
409         UdpClientHelper client(ueIpIface.GetAddress(u), otherPort);
410         client.SetAttribute("Interval",
411             TimeValue(MilliSeconds(interPacketInterval)));
412         client.SetAttribute("MaxPackets", UintegerValue(1000000));
413
414         clientApps.Add(dlClient.Install(remoteHost));
415         clientApps.Add(ulClient.Install(ueNodes.Get(u)));
416         if (u + 1 < ueNodes.GetN()) {
417           clientApps.Add(client.Install(ueNodes.Get(u + 1)));
418         } else {
419           clientApps.Add(client.Install(ueNodes.Get(0)));
420         }
421       } else if (trafficName == "tcp") {  // traffic mode TCP
422         PacketSinkHelper dlPacketSinkHelper("ns3::TcpSocketFactory",
423             InetSocketAddress(Ipv4Address::GetAny(), dlPort),
                numberOfDualNodes, numberOfEUL, numberOfCUL,
                numberOfEUW, numberOfCUW, simTime);
424         PacketSinkHelper ulPacketSinkHelper("ns3::TcpSocketFactory",
425             InetSocketAddress(Ipv4Address::GetAny(), ulPort),
                numberOfDualNodes, numberOfEUL, numberOfCUL,
                numberOfEUW, numberOfCUW, simTime);
426         PacketSinkHelper packetSinkHelper("ns3::TcpSocketFactory",
427             InetSocketAddress(Ipv4Address::GetAny(), otherPort),
                numberOfDualNodes, numberOfEUL, numberOfCUL,
                numberOfEUW, numberOfCUW, simTime);
428         serverApps.Add(dlPacketSinkHelper.Install(ueNodes.Get(u)));
429         serverApps.Add(ulPacketSinkHelper.Install(remoteHost));
430         serverApps.Add(packetSinkHelper.Install(ueNodes.Get(u)));
431
432         BulkSendHelper dlClientHelper ("ns3::TcpSocketFactory",
433                     InetSocketAddress(Ipv4Address::GetAny(), dlPort));
434         dlClientHelper.SetAttribute ("MaxBytes", UintegerValue (0));
435         clientApps.Add (dlClientHelper.Install (remoteHost));
436         serverApps.Add (dlPacketSinkHelper.Install (ueNodes));
437         BulkSendHelper ulClientHelper ("ns3::TcpSocketFactory",
```

```
438                     InetSocketAddress (remoteHostAddr, ulPort));
439         ulClientHelper.SetAttribute ("MaxBytes", UintegerValue (0));
440         clientApps.Add (ulClientHelper.Install (ueNodes));
441         serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
442                   }
443   }
444
445   serverApps.Start(Seconds(0.01));  // start the server application
446   clientApps.Start(Seconds(0.01));  // start the client application
447
448   serverApps.Stop(Seconds(simTime - 0.1));  // start the server
         application
449   clientApps.Stop(Seconds(simTime - 0.1));  // start the client
         application
450
451   // create the flow monitor
452   FlowMonitorHelper flowmon;
453   Ptr < FlowMonitor > monitor = flowmon.InstallAll(); // install all
         the nodes to flow monitor
454 //============================================================================
455   AnimationInterface anim("LTE-WIFI.xml");  // create the animation
         for netanim
456
457   // set the color, description and size of primary lte user
458   for (uint32_t i = 0; i < EULNodes.GetN(); ++i) {
459     anim.UpdateNodeDescription(EULNodes.Get(i), "EU-LTE");  //
         Optional
460     anim.UpdateNodeColor(EULNodes.Get(i), 0, 255, 0);   // Optional
461     anim.UpdateNodeSize(EULNodes.Get(i)->GetId(), 10, 10);
462   }
463
464   // set the color, description and size of cr lte user
465   for (uint32_t i = 0; i < CRLNodes.GetN(); ++i) {
466     anim.UpdateNodeDescription(CRLNodes.Get(i), "CR-LTE"); //
         Optional
467     anim.UpdateNodeColor(CRLNodes.Get(i), 128, 255, 255);   //
         Optional
468     anim.UpdateNodeSize(CRLNodes.Get(i)->GetId(), 10, 10);
469   }
470
471   // set the color, description and size of primary wifi user
472   for (uint32_t i = 0; i < EUWNodes.GetN(); ++i) {
473     anim.UpdateNodeDescription(EUWNodes.Get(i), "EU-WIFI"); //
         Optional
474     anim.UpdateNodeColor(EUWNodes.Get(i), 64, 0, 128); // Optional
475     anim.UpdateNodeSize(EUWNodes.Get(i)->GetId(), 10, 10);
476   }
477
478   // set the color, description and size of cr wifi user
```

```
479   for (uint32_t i = 0; i < CRWNodes.GetN(); ++i) {
480     anim.UpdateNodeDescription(CRWNodes.Get(i), "CR-WIFI"); //
          Optional
481     anim.UpdateNodeColor(CRWNodes.Get(i), 128, 0, 255); // Optional
482     anim.UpdateNodeSize(CRWNodes.Get(i)->GetId(), 10, 10);
483   }
484
485   // set the color, description and size of dual user
486   for (uint32_t i = 0; i < dualNodes.GetN(); ++i) {
487     anim.UpdateNodeDescription(dualNodes.Get(i), "DUAL-NODE");  //
          Optional
488     anim.UpdateNodeColor(dualNodes.Get(i), 255, 0, 0);    // Optional
489     anim.UpdateNodeSize(dualNodes.Get(i)->GetId(), 10, 10);
490   }
491
492   // set the color, description and size of enb node
493   for (uint32_t i = 0; i < enbNodes.GetN(); ++i) {
494     anim.UpdateNodeDescription(enbNodes.Get(i), "LTE-ENB"); //
          Optional
495     anim.UpdateNodeColor(enbNodes.Get(i), 0, 0, 255); // Optional
496     anim.UpdateNodeSize(enbNodes.Get(i)->GetId(), 10, 10);
497   }
498
499   // set the color, description and size of wifi AP node
500   for (uint32_t i = 0; i < wifiApNode.GetN(); ++i) {
501     anim.UpdateNodeDescription(wifiApNode.Get(i), "Wifi-AP"); //
          Optional
502     anim.UpdateNodeColor(wifiApNode.Get(i), 0, 128, 255); // Optional
503     anim.UpdateNodeSize(wifiApNode.Get(i)->GetId(), 10, 10);
504   }
505
506   // set the color, description and size of spectrumAnalyzerNode
507   anim.UpdateNodeDescription(spectrumAnalyzerNodes.Get(0),
        "server"); // Optional
508   anim.UpdateNodeColor(spectrumAnalyzerNodes.Get(0), 250, 255, 0);
        // Optional
509   anim.UpdateNodeSize(spectrumAnalyzerNodes.Get(0)->GetId(), 10,
        10); // Optional
510
511   AnimationInterface::SetConstantPosition(spectrumAnalyzerNodes.Get(0),
        10,
512       130);
513
514
515   // set the position of enb node
516   AnimationInterface::SetConstantPosition(enbNodes.Get(0), eNBX,
        eNBY);
517
518   // set the position of AP node
```

```
519   AnimationInterface::SetConstantPosition(wifiApNode.Get(0), ApX,
          ApY);
520
521   Simulator::Stop(Seconds(simTime));  // stop simulation
522   Simulator::Run();     // run the simulation
523
524   Ptr < Ipv4FlowClassifier > classifier = DynamicCast <
          Ipv4FlowClassifier
525       > (flowmon.GetClassifier());
526   std::map < FlowId, FlowMonitor::FlowStats > stats =
          monitor->GetFlowStats();
527
528   // output the results of simulation
529   uint32_t total_tx = 0;
530   uint32_t total_rx = 0;
531   double total_through = 0.0;
532   double total_delay = 0.0;
533   double total_jitter = 0.0;
534   uint32_t first_count = 0;
535   uint32_t second_count = 0;
536   uint32_t count = 1;
537
538   ofstream fs;
539   fs.open("graph-information.txt", ios_base::app);
540   fs <<
          "No\tThroughPut(Kbps)\tDelayTime(sec)\tJitterTime(sec)\tPacketLoss(%)"
          << endl;
541
542   for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i =
543       stats.begin(); i != stats.end(); ++i) {
544     total_rx += i->second.rxPackets;
545     total_tx += i->second.txPackets;
546
547     // calculate the throughput, delay time, jitter time and packet
          loss
548     total_through += i->second.rxBytes * 8.0
549             / (i->second.timeLastRxPacket.GetSeconds()
550                 - i->second.timeFirstTxPacket.GetSeconds())
551             / 1024;
552
553     if (i->second.rxPackets > 0) {
554       total_delay += i->second.delaySum.GetSeconds() /
             i->second.rxPackets;
555       total_jitter += i->second.jitterSum.GetSeconds() /
             i->second.rxPackets;
556       second_count ++;
557     }
558
559     first_count ++;
```

```
560
561     if (i->second.timeLastRxPacket.GetSeconds() -
           i->second.timeFirstTxPacket.GetSeconds() > 0.0) {
562       fs << count;
563       fs << "\t" << i->second.rxBytes * 8.0
564               / (i->second.timeLastRxPacket.GetSeconds()
565                   - i->second.timeFirstTxPacket.GetSeconds())
566               / 1024;
567       fs << "\t" << i->second.delaySum.GetSeconds() /
              i->second.rxPackets;
568       fs << "\t" << i->second.jitterSum.GetSeconds() /
              i->second.rxPackets;
569       fs << "\t" << (i->second.txPackets - i->second.rxPackets) *
              100 / i->second.txPackets << endl;
570       count ++;
571     }
572   }
573
574   fs.close();
575
576   // output the results to file
577   FILE* fp = fopen("wifi-network_perf.txt", "a+");
578   fprintf(fp, "ThroughPut(Kbps): %.3f\n", total_through /
         first_count);
579   fprintf(fp, "DelayTime(sec): %.3f\n", total_delay / second_count);
580   fprintf(fp, "JitterTime(sec): %.3f\n", total_jitter /
         second_count);
581   fprintf(fp, "PacketLoss: %.3f\n", ((double)(total_tx) -
         (double)(total_rx)) * 100.0 / (double)total_tx);
582   fclose(fp);
583
584   memset(dtbuf, 0, 256);
585   time(&endTd);
586         strftime(dtbuf, sizeof dtbuf, "%A %b %d %H:%M:%S %Y",
              localtime(&endTd));
587   fprintf(timeFp, "EndTime: %s\n", dtbuf);
588
589   time_t durTime = endTd - startTd;
590         fprintf(timeFp, "SimulatingTime(sec): %d\n", (int)durTime);
591   fclose(timeFp);
592
593
594   // serialize results to XML file
595   flowmon.SerializeToXmlFile("Lte-Wifi.flomon", false, false);
596   Simulator::Destroy();
597   return 0;
598 }
```

codigo/LTE–WIFI.C

3. Código en C para implementación de algoritmo de detección SVD

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <algorithm>
#include <list>
#include "svd.h"

using namespace std;

#define SIGN(a, b) ((b) >= 0.0 ? fabs(a) : -fabs(a))  // get the
    abs(a) by b
#define MAX(x,y) ((x)>(y)?(x):(y))      // get the max value between
    a and b

////////////////////////////////////////////////////////
// sort the matrix of data
Vector<float> SVD(Matrix<float> &m) {
  Vector<float> w(m.GetRows());       // create the 1 matrix
  Matrix<float> v(m.GetRows(), m.GetCols());    // create the 2
     matrix
  dsvd(m, m.GetRows(), m.GetCols(), w.GetRawData(), v);// create the
     3 matrix
  m.SortCols(w);            // sort the matrix
  return w;
}

////////////////////////////////////////////////////////
// get the pythag of 2 variable (a and b)
static double PYTHAG(double a, double b) {
  double at = fabs(a), bt = fabs(b), ct, result;

  if (at > bt) {
    ct = bt / at;
    result = at * sqrt(1.0 + ct * ct);
  } else if (bt > 0.0) {
    ct = at / bt;
    result = bt * sqrt(1.0 + ct * ct);
  } else
    result = 0.0;
  return (result);
}

// main function for SVD algorithm
// a ,v, w: output matrix
// m, n: input parameter
int svd(Matrix<float> &a, int m, int n, float *w, Matrix<float> &v) {
  int flag, i, its, j, jj, k, l, nm;
```

```
  double c, f, h, s, x, y, z;
  double anorm = 0.0, g = 0.0, scale = 0.0;
  double *rv1;

  // if m < n, this is the error of matrix
  if (m < n) {
    fprintf(stderr, "#rows must be > #cols \n");
    return (0);
  }

  // create the area of memory
  rv1 = (double *) malloc((unsigned int) n * sizeof(double));

  /* Householder reduction to bidiagonal form */
  for (i = 0; i < n; i++) {
    /* left-hand reduction */
    l = i + 1;
    rv1[i] = scale * g;
    g = s = scale = 0.0;
    if (i < m) {
      for (k = i; k < m; k++)
        scale += fabs((double) a[k][i]);
      if (scale) {
        for (k = i; k < m; k++) {
          a[k][i] = (float) ((double) a[k][i] / scale);
          s += ((double) a[k][i] * (double) a[k][i]);
        }

        f = (double) a[i][i];
        g = -SIGN(sqrt(s), f);
        h = f * g - s;
        a[i][i] = (float) (f - g);
        if (i != n - 1) {
          for (j = l; j < n; j++) {
            for (s = 0.0, k = i; k < m; k++)
              s += ((double) a[k][i] * (double) a[k][j]);
            f = s / h;
            for (k = i; k < m; k++)
              a[k][j] += (float) (f * (double) a[k][i]);
          }
        }
        for (k = i; k < m; k++)
          a[k][i] = (float) ((double) a[k][i] * scale);
      }
    }
    w[i] = (float) (scale * g);

    /* right-hand reduction */
    g = s = scale = 0.0;
```

```
93      if (i < m && i != n - 1) {
94        for (k = l; k < n; k++)
95          scale += fabs((double) a[i][k]);
96        if (scale) {
97          for (k = l; k < n; k++) {
98            a[i][k] = (float) ((double) a[i][k] / scale);
99            s += ((double) a[i][k] * (double) a[i][k]);
100         }
101         f = (double) a[i][l];
102         g = -SIGN(sqrt(s), f);
103         h = f * g - s;
104         a[i][l] = (float) (f - g);
105         for (k = l; k < n; k++)
106           rv1[k] = (double) a[i][k] / h;
107         if (i != m - 1) {
108           for (j = l; j < m; j++) {
109             for (s = 0.0, k = l; k < n; k++)
110               s += ((double) a[j][k] * (double) a[i][k]);
111             for (k = l; k < n; k++)
112               a[j][k] += (float) (s * rv1[k]);
113           }
114         }
115         for (k = l; k < n; k++)
116           a[i][k] = (float) ((double) a[i][k] * scale);
117       }
118     }
119     anorm = MAX(anorm, (fabs((double )w[i]) + fabs(rv1[i])));
120   }
121
122   /* accumulate the right-hand transformation */
123   for (i = n - 1; i >= 0; i--) {
124     if (i < n - 1) {
125       if (g) {
126         for (j = l; j < n; j++)
127           v[j][i] =
128               (float) (((double) a[i][j] / (double) a[i][l]) / g);
129         /* double division to avoid underflow */
130         for (j = l; j < n; j++) {
131           for (s = 0.0, k = l; k < n; k++)
132             s += ((double) a[i][k] * (double) v[k][j]);
133           for (k = l; k < n; k++)
134             v[k][j] += (float) (s * (double) v[k][i]);
135         }
136       }
137       for (j = l; j < n; j++)
138         v[i][j] = v[j][i] = 0.0;
139     }
140     v[i][i] = 1.0;
141     g = rv1[i];
```

```
142       l = i;
143     }
144
145     /* accumulate the left-hand transformation */
146     for (i = n - 1; i >= 0; i--) {
147       l = i + 1;
148       g = (double) w[i];
149       if (i < n - 1)
150         for (j = l; j < n; j++)
151           a[i][j] = 0.0;
152       if (g) {
153         g = 1.0 / g;
154         if (i != n - 1) {
155           for (j = l; j < n; j++) {
156             for (s = 0.0, k = l; k < m; k++)
157               s += ((double) a[k][i] * (double) a[k][j]);
158             f = (s / (double) a[i][i]) * g;
159             for (k = i; k < m; k++)
160               a[k][j] += (float) (f * (double) a[k][i]);
161           }
162         }
163         for (j = i; j < m; j++)
164           a[j][i] = (float) ((double) a[j][i] * g);
165       } else {
166         for (j = i; j < m; j++)
167           a[j][i] = 0.0;
168       }
169       ++a[i][i];
170     }
171
172     /* diagonalize the bidiagonal form */
173     for (k = n - 1; k >= 0; k--) { /* loop over singular values */
174       for (its = 0; its < 30; its++) { /* loop over allowed iterations
           */
175         flag = 1;
176         for (l = k; l >= 0; l--) { /* test for splitting */
177           nm = l - 1;
178           if (fabs(rv1[l]) + anorm == anorm) {
179             flag = 0;
180             break;
181           }
182           if (fabs((double) w[nm]) + anorm == anorm)
183             break;
184         }
185         if (flag) {
186           c = 0.0;
187           s = 1.0;
188           for (i = l; i <= k; i++) {
189             f = s * rv1[i];
```

```
            if (fabs(f) + anorm != anorm) {
              g = (double) w[i];
              h = PYTHAG(f, g);
              w[i] = (float) h;
              h = 1.0 / h;
              c = g * h;
              s = (-f * h);
              for (j = 0; j < m; j++) {
                y = (double) a[j][nm];
                z = (double) a[j][i];
                a[j][nm] = (float) (y * c + z * s);
                a[j][i] = (float) (z * c - y * s);
              }
            }
          }
        }
        z = (double) w[k];
        if (l == k) { /* convergence */
          if (z < 0.0) { /* make singular value nonnegative */
            w[k] = (float) (-z);
            for (j = 0; j < n; j++)
              v[j][k] = (-v[j][k]);
          }
          break;
        }
        if (its >= 30) {
          free((void*) rv1);
          fprintf(stderr, "No convergence after 30,000! iterations
             \n");
          return (0);
        }

        /* shift from bottom 2 x 2 minor */
        x = (double) w[l];
        nm = k - 1;
        y = (double) w[nm];
        g = rv1[nm];
        h = rv1[k];
        f = ((y - z) * (y + z) + (g - h) * (g + h)) / (2.0 * h * y);
        g = PYTHAG(f, 1.0);
        f = ((x - z) * (x + z) + h * ((y / (f + SIGN(g, f))) - h)) / x;

        /* next QR transformation */
        c = s = 1.0;
        for (j = l; j <= nm; j++) {
          i = j + 1;
          g = rv1[i];
          y = (double) w[i];
          h = s * g;
```

```
238           g = c * g;
239           z = PYTHAG(f, h);
240           rv1[j] = z;
241           c = f / z;
242           s = h / z;
243           f = x * c + g * s;
244           g = g * c - x * s;
245           h = y * s;
246           y = y * c;
247           for (jj = 0; jj < n; jj++) {
248             x = (double) v[jj][j];
249             z = (double) v[jj][i];
250             v[jj][j] = (float) (x * c + z * s);
251             v[jj][i] = (float) (z * c - x * s);
252           }
253           z = PYTHAG(f, h);
254           w[j] = (float) z;
255           if (z) {
256             z = 1.0 / z;
257             c = f * z;
258             s = h * z;
259           }
260           f = (c * g) + (s * y);
261           x = (c * y) - (s * g);
262           for (jj = 0; jj < m; jj++) {
263             y = (double) a[jj][j];
264             z = (double) a[jj][i];
265             a[jj][j] = (float) (y * c + z * s);
266             a[jj][i] = (float) (z * c - y * s);
267           }
268         }
269         rv1[l] = 0.0;
270         rv1[k] = f;
271         w[k] = (float) x;
272       }
273     }
274     free((void*) rv1);
275     return (1);
276   }
```

codigo/svd.C

4. Código en C para implementación de algoritmo de teoría de juegos de coaliciones

```
#include "coalition.h"
#include <ns3/log.h>

namespace ns3 {

NS_LOG_COMPONENT_DEFINE ("coalition");

NS_OBJECT_ENSURE_REGISTERED (coalition);

// initialize the DefaultConfiguration for FrHardDownlink
static const struct FrHardDownlinkDefaultConfiguration {
  uint8_t m_cellId;
  uint8_t m_dlBandwidth;
  uint8_t m_dlOffset;
  uint8_t m_dlSubBand;
} g_frHardDownlinkDefaultConfiguration[] = { { 1, 15, 0, 4 }, { 2,
   15, 4, 4 }, {
    3, 15, 8, 6 }, { 1, 25, 0, 8 }, { 2, 25, 8, 8 }, { 3, 25, 16, 9
       }, { 1,
    50, 0, 16 }, { 2, 50, 16, 16 }, { 3, 50, 32, 18 }, { 1, 75, 0,
       24 }, {
    2, 75, 24, 24 }, { 3, 75, 48, 27 }, { 1, 100, 0, 32 },
    { 2, 100, 32, 32 }, { 3, 100, 64, 36 } };

// initialize the DefaultConfiguration for FrHardUplink
static const struct FrHardUplinkDefaultConfiguration {
  uint8_t m_cellId;
  uint8_t m_ulBandwidth;
  uint8_t m_ulOffset;
  uint8_t m_ulSubBand;
} g_frHardUplinkDefaultConfiguration[] = { { 1, 15, 0, 5 }, { 2, 15,
   5, 5 }, {
    3, 15, 10, 5 }, { 1, 25, 0, 8 }, { 2, 25, 8, 8 }, { 3, 25, 16, 9
       }, { 1,
    50, 0, 16 }, { 2, 50, 16, 16 }, { 3, 50, 32, 18 }, { 1, 75, 0,
       24 }, {
    2, 75, 24, 24 }, { 3, 75, 48, 27 }, { 1, 100, 0, 32 },
    { 2, 100, 32, 32 }, { 3, 100, 64, 36 } };

// init number of downlink configure
const uint16_t NUM_DOWNLINK_CONFS(
    sizeof(g_frHardDownlinkDefaultConfiguration)
        / sizeof(FrHardDownlinkDefaultConfiguration));
// init number of downlink configure
const uint16_t NUM_UPLINK_CONFS(
```

```
40      sizeof(g_frHardUplinkDefaultConfiguration)
41          / sizeof(FrHardUplinkDefaultConfiguration));
42
43  /*
44   * initialize the coalition
45   */
46  coalition::coalition() :
47      m_ffrSapUser(0), m_ffrRrcSapUser(0), m_dlOffset(0),
           m_dlSubBand(0), m_ulOffset(
48           0), m_ulSubBand(0) {
49    NS_LOG_FUNCTION(this);
50    m_ffrSapProvider = new MemberLteFfrSapProvider<coalition>(this);
51    m_ffrRrcSapProvider = new
         MemberLteFfrRrcSapProvider<coalition>(this);
52  }
53
54  coalition::~coalition() {
55    NS_LOG_FUNCTION(this);
56  }
57
58  void coalition::DoDispose() {
59    NS_LOG_FUNCTION(this);
60    delete m_ffrSapProvider;
61    delete m_ffrRrcSapProvider;
62  }
63
64  TypeId coalition::GetTypeId() {
65    static TypeId tid =
66        TypeId("ns3::coalition").SetParent<LteFfrAlgorithm>().SetGroupName(
67            "Lte").AddConstructor<coalition>().AddAttribute(
68            "UlSubBandOffset",
69            "Uplink Offset in number of Resource Block Groups",
70            UintegerValue(0),
71            MakeUintegerAccessor(&coalition::m_ulOffset),
72            MakeUintegerChecker<uint8_t>()).AddAttribute(
73            "UlSubBandwidth",
74            "Uplink Transmission SubBandwidth Configuration in number
                 of Resource Block Groups",
75            UintegerValue(25),
76            MakeUintegerAccessor(&coalition::m_ulSubBand),
77            MakeUintegerChecker<uint8_t>()).AddAttribute(
78            "DlSubBandOffset",
79            "Downlink Offset in number of Resource Block Groups",
80            UintegerValue(0),
81            MakeUintegerAccessor(&coalition::m_dlOffset),
82            MakeUintegerChecker<uint8_t>()).AddAttribute(
83            "DlSubBandwidth",
84            "Downlink Transmission SubBandwidth Configuration in
                 number of Resource Block Groups",
```

```
85            UintegerValue(25),
86            MakeUintegerAccessor(&coalition::m_dlSubBand),
87            MakeUintegerChecker<uint8_t>());
88    return tid;
89  }
90
91  void coalition::SetLteFfrSapUser(LteFfrSapUser* s) {
92    NS_LOG_FUNCTION(this << s);
93    m_ffrSapUser = s;
94  }
95
96  LteFfrSapProvider*
97  coalition::GetLteFfrSapProvider() {
98    NS_LOG_FUNCTION(this);
99    return m_ffrSapProvider;
100 }
101
102 void coalition::SetLteFfrRrcSapUser(LteFfrRrcSapUser* s) {
103   NS_LOG_FUNCTION(this << s);
104   m_ffrRrcSapUser = s;
105 }
106
107 LteFfrRrcSapProvider*
108 coalition::GetLteFfrRrcSapProvider() {
109   NS_LOG_FUNCTION(this);
110   return m_ffrRrcSapProvider;
111 }
112
113 void coalition::DoInitialize() {
114   NS_LOG_FUNCTION(this);
115   LteFfrAlgorithm::DoInitialize();
116
117   NS_ASSERT_MSG(m_dlBandwidth > 14,
118       "DlBandwidth must be at least 15 to use coalition algorithms");
119   NS_ASSERT_MSG(m_ulBandwidth > 14,
120       "UlBandwidth must be at least 15 to use coalition algorithms");
121
122   if (m_frCellTypeId != 0) {
123     SetDownlinkConfiguration(m_frCellTypeId, m_dlBandwidth);
124     SetUplinkConfiguration(m_frCellTypeId, m_ulBandwidth);
125   }
126
127 }
128
129 /*
130  * reconfiguration
131  */
132 void coalition::Reconfigure() {
133   NS_LOG_FUNCTION(this);
```

```
134   if (m_frCellTypeId != 0) {
135     SetDownlinkConfiguration(m_frCellTypeId, m_dlBandwidth);
136     SetUplinkConfiguration(m_frCellTypeId, m_ulBandwidth);
137   }
138
139   // init the downlink rbg maps
140   InitializeDownlinkRbgMaps();
141
142   // init the downlink rbg maps
143   InitializeUplinkRbgMaps();
144   m_needReconfiguration = false;
145 }
146
147 /*
148  * set the downlink configuration
149  */
150 void coalition::SetDownlinkConfiguration(uint16_t cellId, uint8_t
       bandwidth) {
151   NS_LOG_FUNCTION(this);
152   for (uint16_t i = 0; i < NUM_DOWNLINK_CONFS; ++i) {
153     if ((g_frHardDownlinkDefaultConfiguration[i].m_cellId == cellId)
154         && g_frHardDownlinkDefaultConfiguration[i].m_dlBandwidth
155             == m_dlBandwidth) {
156       m_dlOffset =
             g_frHardDownlinkDefaultConfiguration[i].m_dlOffset;
157       m_dlSubBand =
             g_frHardDownlinkDefaultConfiguration[i].m_dlSubBand;
158     }
159   }
160 }
161
162 /*
163  * set the uplink configuration
164  */
165 void coalition::SetUplinkConfiguration(uint16_t cellId, uint8_t
       bandwidth) {
166   NS_LOG_FUNCTION(this);
167   for (uint16_t i = 0; i < NUM_UPLINK_CONFS; ++i) {
168     if ((g_frHardUplinkDefaultConfiguration[i].m_cellId == cellId)
169         && g_frHardUplinkDefaultConfiguration[i].m_ulBandwidth
170             == m_ulBandwidth) {
171       m_ulOffset = g_frHardUplinkDefaultConfiguration[i].m_ulOffset;
172       m_ulSubBand =
             g_frHardUplinkDefaultConfiguration[i].m_ulSubBand;
173     }
174   }
175 }
176
177 /*
```

```
178  * Initialize downlink rbg maps
179  */
180 void coalition::InitializeDownlinkRbgMaps() {
181   m_dlRbgMap.clear();
182
183   // get the rbg size
184   int rbgSize = GetRbgSize(m_dlBandwidth);
185   m_dlRbgMap.resize(m_dlBandwidth / rbgSize, true); // resize the
        rbg map
186
187   // assert the message
188   NS_ASSERT_MSG(m_dlOffset <= m_dlBandwidth,
189       "DlOffset higher than DlBandwidth");
190   NS_ASSERT_MSG(m_dlSubBand <= m_dlBandwidth,
191       "DlBandwidth higher than DlBandwidth");
192   NS_ASSERT_MSG((m_dlOffset + m_dlSubBand) <= m_dlBandwidth,
193       "(DlOffset+DlSubBand) higher than DlBandwidth");
194
195   // add the data to map
196   for (uint8_t i = m_dlOffset / rbgSize;
197       i < (m_dlOffset / rbgSize + m_dlSubBand / rbgSize); i++) {
198     m_dlRbgMap[i] = false;
199
200   }
201 }
202
203 /*
204  * Initialize uplink rbg maps
205  */
206 void coalition::InitializeUplinkRbgMaps() {
207   m_ulRbgMap.clear(); // map clear
208
209   // check the flag
210   if (!m_enabledInUplink) {
211     m_ulRbgMap.resize(m_ulBandwidth, false); // resize the map FALSE
212     return;
213   }
214
215   // resize the map (TRUE)
216   m_ulRbgMap.resize(m_ulBandwidth, true);
217
218   // assert the messages
219   NS_ASSERT_MSG(m_ulOffset <= m_ulBandwidth,
220       "UlOffset higher than UlBandwidth");
221   NS_ASSERT_MSG(m_ulSubBand <= m_ulBandwidth,
222       "UlBandwidth higher than UlBandwidth");
223   NS_ASSERT_MSG((m_ulOffset + m_ulSubBand) <= m_ulBandwidth,
224       "(UlOffset+UlSubBand) higher than UlBandwidth");
225
```

```
226   // add the data to map
227   for (uint8_t i = m_ulOffset; i < (m_ulOffset + m_ulSubBand); i++) {
228     m_ulRbgMap[i] = false;
229   }
230 }
231
232 // callback function for get available dl rbg
233 std::vector<bool> coalition::DoGetAvailableDlRbg() {
234   NS_LOG_FUNCTION(this);
235
236   // check the flag
237   if (m_needReconfiguration) {
238     Reconfigure();
239   }
240
241   // check the rbg map
242   if (m_dlRbgMap.empty()) {
243     InitializeDownlinkRbgMaps();
244   }
245
246   return m_dlRbgMap;
247 }
248
249 /*
250  * Is Dl RbgAvailable for ue
251  */
252 bool coalition::DoIsDlRbgAvailableForUe(int rbId, uint16_t rnti) {
253   NS_LOG_FUNCTION(this);
254   // get the item using id
255   return !m_dlRbgMap[rbId];
256 }
257
258 /*
259  * callback function for get available ul rbg
260  */
261 std::vector<bool> coalition::DoGetAvailableUlRbg() {
262   NS_LOG_FUNCTION(this);
263   // check the ulRbgMap
264   if (m_ulRbgMap.empty()) {
265
266     // initialize the uplink RBG map
267     InitializeUplinkRbgMaps();
268   }
269
270   return m_ulRbgMap;
271 }
272
273 /*
274  * Is Ul Rbg Available For Ue
```

```
 */
bool coalition::DoIsUlRbgAvailableForUe(int rbId, uint16_t rnti) {
  NS_LOG_FUNCTION(this);

  // check the uplink flag
  if (!m_enabledInUplink) {
    return true;
  }

  return !m_ulRbgMap[rbId];
}

/*
 * print the report for dl Cqi information
 */
void coalition::DoReportDlCqiInfo(
    const struct FfMacSchedSapProvider::SchedDlCqiInfoReqParameters&
       params) {
  NS_LOG_FUNCTION(this);
  NS_LOG_WARN("Method should not be called, because it is empty");
}

/*
 * print the report for ul Cqi information-1
 */
void coalition::DoReportUlCqiInfo(
    const struct FfMacSchedSapProvider::SchedUlCqiInfoReqParameters&
       params) {
  NS_LOG_FUNCTION(this);
  NS_LOG_WARN("Method should not be called, because it is empty");
}

/*
 * print the report for ul Cqi information-2
 */
void coalition::DoReportUlCqiInfo(
    std::map<uint16_t, std::vector<double> > ulCqiMap) {
  NS_LOG_FUNCTION(this);
  NS_LOG_WARN("Method should not be called, because it is empty");
}

/*
 * print the topic
 */
uint8_t coalition::DoGetTpc(uint16_t rnti) {
  NS_LOG_FUNCTION(this);
  return 1; // 1 is mapped to 0 for Accumulated mode, and to -1 in
     Absolute mode TS36.213 Table 5.1.1.1-2
}
```

```
321
322 /*
323  * get the min continuous ul bandwidth
324  */
325 uint8_t coalition::DoGetMinContinuousUlBandwidth() {
326   NS_LOG_FUNCTION(this);
327
328   if (!m_enabledInUplink) {
329     return m_ulBandwidth;
330   }
331
332   return m_ulSubBand;
333 }
334
335 /*
336  * print the report for user measuer
337  */
338 void coalition::DoReportUeMeas(uint16_t rnti,
339     LteRrcSap::MeasResults measResults) {
340   NS_LOG_FUNCTION(this << rnti << (uint16_t) measResults.measId);
341   NS_LOG_WARN("Method should not be called, because it is empty");
342 }
343
344 /*
345  * print the receive load information
346  */
347 void
    coalition::DoRecvLoadInformation(EpcX2Sap::LoadInformationParams
    params) {
348   NS_LOG_FUNCTION(this);
349   NS_LOG_WARN("Method should not be called, because it is empty");
350 }
351
352 } // end of namespace ns3
```

codigo/coalition.C

5. Código en R para obtención de modelos de métodos de detección

```
1 ######
2 #Script para construccion de CDF de los metodos de deteccion SVD
    propuesto, SVD teorico y
3 #Energy Detection##
4 #Autor: Pablo Palacios
5 #Fecha:07/03/2017
6 #######
7
```

```
8
9  ####################CDF para el metodo SVD Teorico############
10 fs = 48e3;
11 Am=10;
12 time = (0:fs-1)/fs;
13 count=0:100;
14 max_threshold=500;
15 time_series=matrix(0,20,16);
16 threshold=matrix(0,1,50);
17 SNR=-24:25;
18 Ns=10000;
19 L=16;
20 threshold[1]=0.1;
21 for( no in c(-23:25)){
22   noise=matrix(runif(320),20,16);
23   for( i in c(1:320)){
24     time_series[i]=Am*cos(2*pi/320*i)+10^((no-25)/100)*noise[i];
25   }
26   s=svd(time_series);
27   ts_max=max(max(s$d),max(s$u),max(s$v));
28   ts_min=min(min(s$d),min(s$v),min(s$u));
29   threshold[no+25]=max_threshold/(ts_max/ts_min);
30   r=(sqrt(Ns)+sqrt(16))/(sqrt(Ns)-sqrt(16))*
31     (1+((sqrt(Ns)+sqrt(16))^(-2/3))/((Ns*L)^(1/6)));
32   Nsym = 6;
33   beta = 0.5;
34   sampsPerSym = 8;
35   threshold[no+25]=threshold[no+24]+(runif(1)*runif(1))/13;
36 }
37
38
39
40 ####################CDF para el metodo Energy Detection############
41 rnd=rnorm(1000)
42 obj=ecdf(rnd)
43 x=c(-2500:2500)/100
44 y=double(5001)
45 for(i in c(0:5001)){
46   y[i]=obj(x[i])
47 }
48
49
50 ####################Datos de simulaciones para CDF de SVD
      propuesto############
51 Pd=c(0.130483, 0.2090776, 0.3116048, 0.4189724, 0.5393132,
      0.6527466, 0.761855, 0.8157188, 0.8518388, 0.8565692, 0.91309075)
52 SNR_P=c(-25,-20,-15,-10,-5,0,5,10,15,20,25)
53
54
```

```
55 ####################Grafico de los CDFs de los metodos############
56 plot(SNR_P,Pd,main="CDF's",xlab="SNR (dB)",
57      ylab="Probabilidad de Deteccion (Pd)",col="blue",type="l",
           ylim=c(0,1))
58 lines(x,y,col="red", type = "l")
59 lines(SNR,threshold,col="green",type="l")
60 legend(x=-25, y=1, legend=c("SVD propuesto","Energy Detector", "SVD
    Teorico"),cex=0.5, lty=c(1:2),col=c("blue","red", "green"))
```

codigo/metodo.R

6. Código en R para obtención de MLE de los métodos

```
1  #######
2  #Script para comparacion de metodos de deteccion mediante MLE##
3  #Autor: Pablo Palacios
4  #Fecha:07/03/2017
5  #######
6
7
8  ############################Seleccion de paquete para uso de
       MLE######
9  local({pkg <- select.list(sort(.packages(all.available =
       TRUE)),graphics=TRUE)
10 if(nchar(pkg)) library(pkg, character.only=TRUE)})
11 mu = 0;
12 sigma = 1;
13 pd = rnorm(1);
14
15 #=======================Datos de los metodos=================#
16 ysvd=c(0.108472663, 0.141566552, 0.146318109, 0.20832853,
       0.429086015,
17 0.540720545, 0.663368778, 0.731325518, 0.889684802, 0.918901025,
       0.995247747);
18 ypropsvd=c(0.130483, 0.2090776, 0.3116048, 0.4189724, 0.5393132,
19 0.6527466, 0.761855, 0.8157188, 0.8518388, 0.8565692, 0.91309075);
20 ycdf=c(0.000012, 0.000034, 0.000125, 0.000689, 0.008365, 0.467326,
21 0.983261, 0.996968, 0.998972, 0.999535, 0.999752);
22
23
24 #=======================Aplicacion de
       MLE============================#
25 pdf_svd = density(ysvd);
26 pdf_svd
27 pdf_prop=density(ypropsvd);
28 pdf_prop
29 pdf_cdf=density(ycdf);
```

```
pdf_cdf

ys=ysvd*1000000000;
yp=ypropsvd*1000000000;
yc=ycdf*1000000000;
nLL <- function(lambda) -sum(stats::dpois(ys, lambda, log = TRUE))
phat_svd=mle( nLL, start = list(lambda = 5), nobs = NROW(ys),
method = "Brent", lower = 1, upper = 9999999990);
nLL <- function(lambda) -sum(stats::dpois(yp, lambda, log = TRUE))
phat_prop=mle( nLL, start = list(lambda = 5), nobs = NROW(yp),
method = "Brent", lower = 1, upper = 9999999990);
nLL <- function(lambda) -sum(stats::dpois(yc, lambda, log = TRUE))
phat_cdf=mle( nLL, start = list(lambda = 5), nobs = NROW(yc),
method = "Brent", lower = 1, upper = 9999999990);


mle_prop_svd=coef(phat_prop)/1000000000
mle_svd=coef(phat_svd)/1000000000
mle_energy_detect=coef(phat_cdf)/1000000000

############Desviacion estandard del parametro MLE##################
SD_prop=sd(ypropsvd)
SD_svd=sd(ysvd)
SD_cdf=sd(ycdf)

############Error estandard del parametro MLE#################

SDerror_pro=SD_prop/sqrt(length(ypropsvd))
SDerror_svd=SD_svd/sqrt(length(ysvd))
SDerror_ener=SD_cdf/sqrt(length(ycdf))

#################Presentacion de resultados###############3
mle_prop_svd
mle_svd
mle_energy_detect

SDerror_pro
SDerror_svd
SDerror_ener

SD_prop
SD_svd
SD_cdf
```

codigo/MLE.R