UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"PROTOTIPO 5G NR V2X SOBRE OPENAIRINTERFACE"

TRABAJO FIN DE GRADO

Septiembre - 2022

AUTOR: Kaiming Chen

DIRECTOR/ES: Baldomero Coll Perales

Miguel Sepulcre Ribes

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"PROTOTIPO 5G NR V2X SOBRE OPENAIRINTERFACE"

TRABAJO FIN DE GRADO

Septiembre - 2022

AUTOR: Kaiming Chen

DIRECTOR/ES: Baldomero Coll Perales

Miguel Sepulcre Ribes

RESUMEN

La seguridad vial es un aspecto crucial en el tráfico en carreteras y autopistas. La conducción de un vehículo es una habilidad que exige altos niveles de concentración al conductor en todo tipo de situaciones. Con esto en mente, la comunidad científica y la industria están trabajando en el diseño y desarrollo de vehículos autónomos para una conducción segura, fluida y adecuada ante cualquier posible situación de peligro que pueda darse en las carreteras. Los vehículos autónomos tienen incorporados sensores (cámaras, radares, LIDAR, etc.) para detectar el entorno que los rodea y así poder controlar de forma autónoma la conducción. Sin embargo, la eficacia de estos sensores puede verse afectada por obstáculos físicos o condiciones meteorológicas adversas. En este contexto, los sistemas de comunicación V2X (*Vehicle to Everything*) pueden complementar las capacidades de los sensores gracias al intercambio de información entre los vehículos y entre los vehículos y la infraestructura. Por ejemplo, las comunicaciones V2X pueden utilizarse para intercambiar información sobre posición y velocidad, o para coordinar maniobras como adelantamientos.

El organismo de estandarización 3GPP ha desarrollado el estándar LTE V2X para la implementación de los sistemas de comunicación V2X entre vehículos por medio de la red celular, que cubre las capas física y de acceso al medio. Esta tecnología es posteriormente evolucionada en el estándar 5G NR V2X, que soporta casos de usos más avanzados. Ambas tecnologías permiten la comunicación directa entre vehículos (V2V, *Vehicle-to-Vehicle*), y por lo tanto no requieren tener cobertura de la infraestructura celular.

Dada la reciente publicación del estándar del 3GPP que define la tecnología 5G NR V2X, existen escasas implementaciones para su estudio y optimización. Las principales implementaciones existentes se utilizan en estudios de simulación software. En este proyecto se pretende dar un paso más y realizar una implementación sobre la plataforma OpenAirInterface (OAI). OAI es una plataforma SDR (*Software Defined Radio*) ampliamente utilizada para la emulación de redes celulares y no dispone de una implementación de 5G NR V2X. De este modo, en este proyecto se ha implementado las funcionalidades de la capa MAC del estándar 5G NR V2X y se ha integrado el código en la plataforma de emulación OAI. La implementación realizada se ha validado y se ha utilizado para evaluar el rendimiento para diferentes configuraciones y escenarios.

ABSTRACT

Road safety is a crucial aspect in traffic on roads and highways. Driving a vehicle is a skill that requires high levels of concentration from the driver in all kinds of situations. With this in mind, autonomous vehicles are being designed and deployed by the industry and the research community for safe, smooth and efficient driving in any possible danger situation on the roads. Autonomous vehicles have sensors (cameras, radars, LIDAR, etc.) built in to detect the environment around them and thus be able to autonomously control the driving. However, the effectiveness of these sensors may be affected by physical obstacles or adverse weather conditions. In this context, V2X (*Vehicle to Everything*) communication systems can complement the capabilities of sensors thanks to the exchange of information between vehicles and between vehicles and infrastructure. For example, V2X communications can be used to exchange information about the position and speed of the vehicles, or to coordinate their maneuvers.

The 3GPP standardization body has developed the LTE V2X standard for the implementation of V2X communication systems between vehicles through the cellular network, that covers the physical and access layers of the protocol stack. This technology has been recently evolved in the 5G NR V2X standard, which includes more advanced use cases. Both technologies allow direct communication between vehicles (V2V, *Vehicle-to-Vehicle*), even if there is no coverage from the cellular infrastructure.

Given the recent publication of the 5G NR V2X standard, the number of implementations is openly available to study its performance and operation. The main existing implementations are used for network simulations. In this project, the goal is to go one step further by implementing it over the OpenAirInterface (OAI) platform. OAI is a SDR (*Software Defined Radio*) platform widely used for the emulation of cellular networks and it does not include 5G NR V2X yet. In this context, in this project, the MAC layer of the 5G NR V2X standard has been implemented and the code has been integrated into the OAI platform. The implementation has been validated and has been used to evaluate the performance of the technology under different configuration scenarios.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En la actualidad hay un creciente progreso en la investigación de los sistemas de comunicaciones entre vehículos autónomos (V2X o *Vehicle to Everything*) como complemento a las detecciones directas de los peligros del entorno por los sensores incorporados en dichos vehículos. La evolución de estos sensores (cámaras, radares, LIDAR, etc.) a lo largo del tiempo y sus constantes mejoras han contribuido en parte a la construcción de vehículos cada vez más adaptados a los requerimientos de una conducción autónoma. Sin embargo, la percepción del entorno por parte de los sensores puede verse limitada por obstáculos físicos o condiciones meteorológicas adversas que dificulten o disminuyen sus capacidades sensoriales de detección. Ante la necesidad de proveer una conducción segura y eficiente ante las adversidades en cualquier tipo de escenario, se requiere la implementación y la aplicación de los sistemas de comunicación V2X entre vehículos (V2V) y entre vehículos y la infraestructura (V2I). La cooperación conjunta de la percepción de los sensores junto con la transmisión de paquetes en las comunicaciones V2X satisface la fiabilidad y robustez exigidas para vehículos autónomos en el control de su conducción de forma autónoma y segura en cualquier escenario. Este intercambio de datos de los sensores (llamado sensado cooperativo) permite ampliar las capacidades de detección de otros vehículos o nodos/usuarios en los vehículos autónomos más allá de los límites de alcance de los sensores instalados, por lo tanto, se consigue una detección temprana y una mayor anticipación para evitar accidentes. En las comunicaciones V2X los vehículos también pueden intercambiar sus intenciones de maniobras de conducción para una coordinación colectiva (llamado conducción cooperativa) que evite accidentes y mejore la fluidez del tráfico. Los sistemas de comunicación V2X ofrecen alta fiabilidad y baja latencia, consiguiendo altas tasas de transmisión entre los vehículos autónomos.

La primera versión del sistema de comunicación V2X (IEEE 802.11p) surgió a partir de adaptaciones de la tecnología WiFi. Sin embargo, en los análisis de los resultados de numerosos estudios se observa un bajo rendimiento en los escenarios de alta densidad de vehículos y altos niveles de automatización, sin satisfacer los estándares de calidad exigidos. La tecnología *Cellular* V2X, C-V2X o LTE-V surge como alternativa al uso de IEEE 802.11p. La tecnología C-V2X es una adaptación de LTE (tecnología móvil 4G) para las comunicaciones V2X en la banda sub-6GHz, e introduce nuevos sistemas de

transmisión, denominados comunicaciones V2V (*Vehicle-to-Vehicle*) con o sin cobertura de red celular empleando comunicación *Sidelink* (SL) o basada en la interfaz PC5. El 3GPP desarrolló la tecnología C-V2X en la *Release* 14, que posteriormente fue mejorada ligeramente en la *Release* 15. En la *Release* 15 también se ha estudiado y evaluado el funcionamiento de los posibles casos de uso más avanzados para la conducción autónoma que se incluirán en la *Release* 16. En la *Release* 16 se ha desarrollado el estándar 5G NR V2X, que introduce la tecnología 5G a las comunicaciones V2X, permitiendo proporcionar servicios de conducción conectada y autónoma más avanzados.

En el estándar 5G NR V2X se introduce casos de uso más avanzados y también permite la comunicación *Sidelink* (SL) o basada en la interfaz PC5. Para lograr el objetivo de la comunicación V2V sin cobertura celular 5G se requieren protocolos que configuran y gestionan automáticamente sus comunicaciones de forma distribuida y se le conoce técnicamente como modo 2. Esta capacidad de gestión autónoma es esencial para garantizar la comunicación V2X ubicua para vehículos autónomos y garantizar los servicios de percepción y conducción cooperativa.

Hoy en día se usa ampliamente herramientas de simulación de software para implementar, validar y testear los sistemas de comunicación V2X basados en el estándar 5G NR V2X debido a que es altamente configurable y versátil. Sin embargo, los escenarios de simulación se basan en suposiciones y simplificaciones introducidas durante el modelado de los sistemas de comunicación, lo que puede conducir a conclusiones que no se corresponden con la realidad. La Figura 1 resume el modelado de diferentes agentes o capas de un sistema de comunicación dependiendo de la plataforma utilizada (simulación y prototipado). En la Figura 1 se puede observar que solo los estudios basados en prototipos pueden evaluar todas las capas de un sistema de comunicación en condiciones reales. Sin embargo, el uso de prototipos no está exento de desafíos asociados con las limitaciones existentes de los productos disponibles (como las de cómputo) y la dificultad para realizar testeos a gran escala (incluso económicas).

En este contexto, este proyecto tiene como objetivo desarrollar la capa MAC (*Medium Access Control*) de 5G NR V2X en una arquitectura de emulación que tenga en cuenta los inconvenientes existentes en las plataformas de simulación y de prototipado. Para lograr este objetivo, se utilizará y evolucionará una plataforma de emulación basada en OpenAirInterface (OAI). En esta plataforma, todas las capas del sistema de comunicación se implementan para su uso en sistemas reales, excepto la capa PHY, que es modelada.

Esta arquitectura híbrida entre simulación y prototipado es lo suficientemente flexible como para evaluar el rendimiento de 5G NR V2X en diferentes escenarios, lo que permite el desarrollo y la evaluación en pruebas de laboratorio. Esta arquitectura fue diseñada e implementada en un proyecto anterior en colaboración con EURECOM (Francia) para evaluar LTE-V2X. Este proyecto se ha evolucionado para integrar la capa MAC de 5G NR V2X.

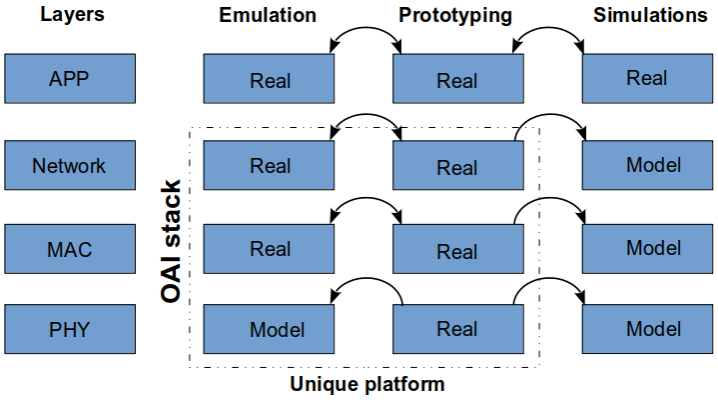

Figura 1. Modelado de los diferentes agentes o capas utilizando plataforma de simulación, emulación, y prototipado

Este trabajo se estructura en las siguientes secciones. En la sección 2 se presenta un análisis del estado del arte que explica las diferentes plataformas que se usan para la implementación y evaluación del rendimiento de los estándares de comunicación V2X en diferentes escenarios, así como la evolución de estos estándares. En la sección 3 se describe la capa física y la capa MAC del estándar 5G NR V2X y las funcionalidades a implementar. En la sección 4 se hace una breve descripción de otros proyectos de OpenAirInterface (OAI), la descripción de los archivos en el proyecto de OAI y los comandos necesarios para preparar el equipo para la compilación y ejecución de OAI. En la sección 5 se explica con gran nivel de detalle toda la implementación realizada y su posterior integración en OAI, así como los principales problemas encontrados durante la implementación y las resoluciones aplicadas y también las automatizaciones implementadas para las emulaciones. La sección 6 presenta un análisis de los resultados obtenidos y su comparativa con los resultados en el simulador ns-3. En la sección 7 se expone las conclusiones. Y en la sección 8 está la bibliografía.

El trabajo realizado en este TFG ha formado parte del proyecto de I+D CENID titulado “5G y Conducción Autónoma para la Digitalización de la Movilidad” financiado por la Diputación de Alicante y dirigido por el Dr. Baldomero Coll Perales.

2. ESTADO DEL ARTE

2.1. PLATAFORMAS DE SIMULACIÓN Y EMULACIÓN

Los simuladores de comunicaciones (como ns-3 y OMNeT++) se utilizan para modelar y evaluar el comportamiento de redes de comunicaciones. Estos simuladores permiten explorar diferentes de escenarios de red, tales como el rendimiento de la red, el análisis de protocolos y el comportamiento en condiciones de error. Los simuladores de comunicación pueden proporcionar mucha información útil, pero tienen algunas limitaciones. Los simuladores en particular a menudo no cumplen con los estándares de comunicación, lo que puede conducir a resultados incompletos o inexactos.

Las pruebas de campo mediante prototipos con chips reales (como Quectel) se utilizan para probar aplicaciones y servicios de vehículos en implementaciones reales que cumplen con los estándares. Estas pruebas nos permiten evaluar si la aplicación o servicio en cuestión está funcionando correctamente y detectar posibles errores o problemas. Sin embargo, los chips no existen hasta varios años después de que se establezca un estándar y además los chips no se pueden modificar como el software para su investigación en la materia.

En este contexto, las plataformas SDR (*Software Defined Radio*) proporcionan un compromiso entre la simulación y el prototipado, puesto que implementan en software la pila de protocolos al completo, siguiendo los estándares. Existen diferentes plataformas SDR que implementan los estándares de 3GPP para las comunicaciones celulares, como srsLTE, OpenLTE y OAI, las cuales se presentan a continuación.

La plataforma srsLTE es una solución de software ampliamente usada para implementar redes LTE. Proporciona una arquitectura de red completamente funcional y una interfaz fácil de usar para la gestión y el control de la red. srsLTE también es compatible con una amplia gama de dispositivos y se puede utilizar para su implementación en una variedad de plataformas.

OpenLTE es también una plataforma de software para construir redes LTE avanzadas. El objetivo es simplificar el diseño e implementación de este tipo de redes para que los operadores de telecomunicaciones y otras empresas puedan ofrecer servicios LTE de una manera más eficiente y rentable. OpenLTE consta de una serie de módulos que se pueden usar de forma independiente o en conjunto para formar una red LTE completa. La

plataforma incluye una capa de aplicación para el control y la gestión de la red y una capa de soporte de hardware para acceder a los recursos de la red.

OAI es una plataforma software de código abierto configurable y flexible para el desarrollo de 4G LTE y 5G *New Radio* (NR). La plataforma OAI puede ejecutarse en ordenadores de uso común o propósito general de arquitectura x86 y para la transmisión radio inalámbrica deben estar equipados con tarjetas radio SDR. Una de las tarjetas radio más comunes para ser usadas en la plataforma OAI es el periférico USRP (*Universal Software Radio Peripheral*) de ETTUS. La plataforma OAI es muy utilizada por la comunidad investigadora para la realización de experimentos y pruebas de campo de la red de acceso radio de las tecnologías 4G y 5G, y está continuamente siendo evolucionada para incorporar las funcionalidades necesarias para realizar testeos de comunicaciones. OAI es la plataforma seleccionada en el presente proyecto, dado que el trabajo parte de una implementación previa del grupo de investigación. En la sección 5 se profundiza en su descripción.

2.2. TECNOLOGÍAS DE COMUNICACIÓN

En la *Release* 12 el organismo de estandarización 3GPP (*3rd Generation Partnership Project*) definió por primera vez un estándar para las comunicaciones directas entre dispositivos o UEs, denominada comunicaciones D2D (*Device-to-Device*). La tecnología D2D se basa en LTE y su uso permite la interacción de dispositivos próximos entre sí (*ProSe, Proximity Services*) [1]. El desarrollo de la tecnología LTE D2D permitía sentar las bases para la posibilidad de implementar aplicaciones de redes sociales que impliquen la interacción de usuarios que se encuentren cerca, aunque no formaba parte de los objetivos de la *Release* 12 de 3GPP.

Además del estándar LTE D2D, 3GPP ha desarrollado posteriormente el estándar LTE V2X en la *Release* 14 como evolución de LTE D2D, que es mejorado en la *Release* 15 [2] y el estándar 5G NR V2X en la *Release* 16. El estándar LTE V2X, también denominado *Cellular* V2X o C-V2X, se desarrolló a partir de los avances en LTE D2D e introduce las comunicaciones V2X basado en tecnologías celulares. El 3GPP en la *Release* 15 introduce un nuevo modo de sistema de comunicación V2X basado en la nueva interfaz de radio denominada *New Radio* (NR), con uso limitado para comunicaciones con la red sin la posibilidad de comunicaciones directas entre vehículos. En el estándar 5G NR V2X los UEs pueden referirse tanto a vehículos como a

infraestructura vial ubicada en la carretera (*RoadSide Units*) o a dispositivos móviles transportados por peatones. El 3GPP ha desarrollado el estándar 5G NR V2X en la *Release* 16 para comunicaciones V2X basadas en la nueva interfaz radio 5G NR. Este estándar incorpora un nuevo modo de transmisión que permite la comunicación directa entre vehículos sin la necesidad de que la información transmitida pase por la red celular.

LTE V2X es un estándar que puede establecer comunicación V2X basada en tecnología LTE. LTE V2X está diseñado para operar en la banda de frecuencia de 5.9 GHz reservada para el uso de servicios ITS (*Intelligent Transportation Services*) en la mayoría de las regiones del mundo. Estos servicios se basan en el envío de pequeños mensajes en modo broadcast, como CAMs (*Cooperative Awareness Messages*) o BSMs (*Basic Safety Messages*). A través de estos mensajes, los vehículos pueden intercambiar información básica como la posición, dirección de movimiento, la velocidad y la aceleración. Esto es muy útil para implementar servicios cooperativos de seguridad vial.

Para adaptarse a las necesidades y especificaciones de la comunicación V2X, el estándar LTE V2X define nuevas capas física y MAC como evolución de LTE. La capa física se basa en la tecnología SC-FDMA (*Single-Carrier Frequency Division Multiple Access*) y organiza los recursos radio en una forma de rejilla en función del tiempo y la frecuencia. Estos recursos radio de LTE V2X se denominan *Resource Blocks* (RB). Un RB consta de 12 subportadoras con un ancho de banda fijo de 15 kHz. En el eje del tiempo, se divide en subtramas de 1 ms. La Figura 2 muestra un ejemplo de la estructura tiempo-frecuencia de la capa física de LTE V2X. Se puede observar en la figura que los subcanales están formados por un conjunto de RBs. Los subcanales se utilizan para la transmisión de mensajes o *Transport Blocks* (TB) e información de control, denominada *Sidelink Control Information* (SCI), asociada con la transmisión. El SCI siempre ocupa 2 RBs y se ubica en la parte superior del primer subcanal utilizado en la transmisión del paquete.

El estándar LTE V2X incluye los modos 3 y 4 para la asignación de recursos o subcanales para realizar las comunicaciones V2V. En el modo 3, la estación base LTE (eNB) administra los recursos. Por lo tanto, este modo solo puede aplicarse si los vehículos se encuentran dentro del rango de alcance de una estación base LTE. En cambio, el modo 4 funciona incluso cuando los vehículos están fuera del alcance de una estación base LTE. De hecho, en el modo 4, el vehículo selecciona de forma autónoma los recursos radio para la transmisión de paquetes. Para el modo 4, el estándar LTE V2X incluye un mecanismo de selección de recursos conocido como *sensing-based Semi-Persistent*

Scheduling (*sensing-based SPS*) (ver la Figura 2). El modo de gestión de recursos radio permite que el vehículo seleccione recursos para futuras transmisiones. De hecho, la selección de recursos radio se mantiene de forma semipersistente durante un cierto número de transmisiones consecutivas (*Reselection Counter* o contador de reselección). Esta selección semipersistente está destinada a cumplir con los requisitos de tráfico V2X esperados para LTE V2X. Se caracteriza por el envío de mensajes periódicos (ej. CAM) con aplicaciones específicas (ej. *Cooperative Awareness Services*). El proceso de selección de un nuevo recurso en el modo 4 se resume en la Figura 2. Como ejemplo, un vehículo genera un nuevo paquete en el instante t_G sin recurso seleccionado previamente. A continuación, el vehículo crea una ventana de selección (*Selection Window*) que contiene todos los recursos libres. Es decir, recursos no reservados por otros vehículos y que satisface el requisito de latencia máxima desde la creación del paquete. Este requisito de latencia máxima se emplea para definir el límite superior de la ventana de selección ($t_G + T_2$ en la Figura 2). Se define una ventana de detección (*Sensing Window*) para encontrar recursos libres dentro de la ventana de selección. Dentro de la ventana de detección, el vehículo escucha o detecta las transmisiones de otros vehículos en cada subcanal. Tras la detección de la transmisión, el vehículo utiliza la información de control contenida en el SCI para identificar las reservas realizadas por otros vehículos para recursos futuros (ver los recursos resaltados en violeta en la Figura 2). Además, también se registra la potencia recibida en cada recurso para comprobar el nivel de ocupación (ver los recursos resaltados en rojo en la Figura 2). Como se muestra en la Figura 2, la ventana de detección precede a la ventana de selección y tiene una duración de 1000 ms. Se selecciona aleatoriamente un recurso de los identificados como libres (no reservados) dentro de la ventana de selección. Esta selección se realiza después de preseleccionar el 20 % de los recursos que están disponibles y que tengan una potencia media recibida más baja durante la ventana de detección. El recurso seleccionado se reserva de forma semipersistente para que los paquetes generados posteriormente (hasta el límite del contador de reselección) puedan utilizar el mismo recurso.

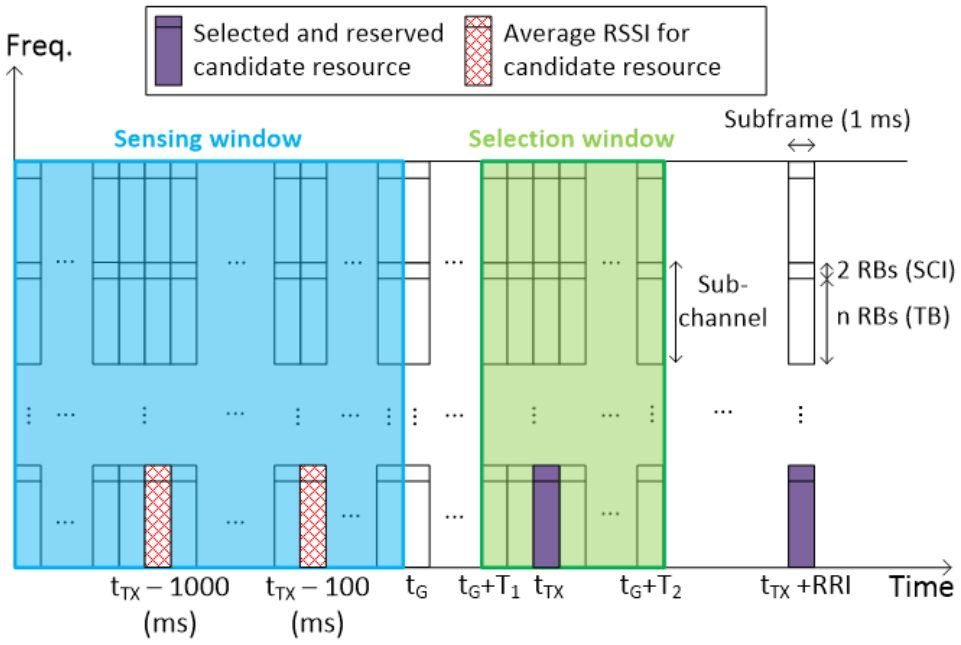

Figura 2. Estructura tiempo-frecuencia de la capa física de LTE V2X y ejemplo del uso del mecanismo de scheduling sensing-based SPS de modo 4.

3GPP publicó la *Release* 16 (https://www.3gpp.org/release-16) a mediados de 2020, que es el primer estándar para comunicación V2X basado en la nueva interfaz de radio 5G NR. El estándar 5G NR V2X ofrece casos de uso avanzados destinadas a dar soporte a servicios críticos de vehículos autónomos conectados, que se caracterizan por requisitos estrictos en términos de latencia, fiabilidad, ancho de banda, etc. Esta sección se enfoca en describir las funcionalidades añadidas en 3GPP *Release* 16 para comunicaciones 5G NR V2X. La *Release* 16 de 3GPP establece la comunicación V2X entre vehículos o V2V, entre vehículos y la red o V2N (*Vehicle-to-Network*), entre vehículos y la infraestructura o V2I (*Vehicle-to-Infrastructure*) y entre vehículos y otros usuarios de la red como peatones o V2P (*Vehicle-to-Pedestrians*). En la terminología 3GPP, V2V también se denomina comunicaciones de enlace lateral (*Sidelink* o SL) o comunicaciones basadas en la interfaz PC5.

La tecnología 5G NR V2X y sus funcionalidades se basan en desarrollos realizados en anteriores *Releases* de 3GPP. Dada la importancia de esta tecnología para el presente trabajo, su funcionamiento se detalla en el siguiente capítulo.

3. TECNOLOGÍA 5G NR V2X

3.1. CAPA FÍSICA

Como en LTE V2X (ver la sección 2.2), las transmisiones *sidelink* (SL) de 5G NR V2X utilizan multiplexación por división de frecuencia ortogonal (OFDM) con un prefijo cíclico (CP) en la capa física (PHY). La estructura de trama SL consta de tramas radio con una duración de 10 ms. Una trama radio se divide en 10 subtramas (SF, *SubFrame*) con una duración de 1 ms cada una. A diferencia de LTE V2X, la cantidad de ranuras por subtrama para 5G NR V2X y el espacio entre subportadoras (SCS, *Sub Carrier Spacing*) para OFDM son flexibles. Para admitir diferentes requisitos y operaciones en diferentes rangos de frecuencia, 5G NR V2X utiliza una numerología OFDM escalable basada en el estándar *Release* 15 NR Uu. Cada numerología OFDM está definida por un SCS que es un múltiplo de 15 kHz (es decir, el SCS para LTE V2X). Por lo tanto, la numerología en 5G NR V2X permite un SCS escalable que se puede expresar como $2\ \mu \times 15$ kHz. En 5G NR V2X, el valor del factor μ puede tomar valores de 0, 1, 2 y 3, lo que da como resultado un SCS de 15 kHz, 30 kHz, 60 kHz y 120 kHz, respectivamente.

Los recursos radio de nivel PHY se agrupan en una rejilla denominada *resource pool* (grupo de recursos). Una *resource pool* consta de *Resource Blocks* o RBs (que a su vez están formados por 12 subportadoras) contiguos en el dominio de la frecuencia y de ranuras en el dominio del tiempo, que no necesitan ser contiguos. La *resource pool* resultante define los recursos disponibles para las transmisiones SL. En el dominio de la frecuencia, el grupo de recursos se divide en *L* subcanales consecutivos. Aquí, un subcanal consta de un grupo de RBs consecutivos dentro de una ranura [3]. NR V2X SL permite tamaños de subcanal de 10, 15, 20, 25, 50, 75 o 100 PRBs. Un subcanal representa la unidad más pequeña de transmisión y recepción de datos SL dentro de una *resource pool*. Las transmisiones SL pueden usar uno o más subcanales.

En NR V2X SL, los datos se transmiten en *Transport Blocks* (TBs) [3] y a cada TB se le asigna un *Sidelink Control Information* (SCI). Los TBs se envían a través del canal físico de datos (PSSCH). El SCI contiene información sobre los recursos utilizados por el PSSCH que transporta el TB asociado e información adicional requerida para decodificar el TB. El SCI en NR V2X se transmite en dos etapas (1st-stage SCI y 2nd-stage SCI) en comparación con una sola etapa para LTE V2X [4]. El SCI de primera etapa en NR V2X

se transmite por el canal físico de control (PSCCH) y el SCI de segunda etapa se transmite por el PSSCH correspondiente.

3.2. CAPA MAC

La *Release* 16 de 3GPP define los modos 1 y 2 para la selección de recursos o subcanales en las comunicaciones de vehículo a vehículo (V2V) en 5G NR V2X basadas en el enlace lateral (SL). Estos dos modos son similares a los modos 3 y 4 del estándar LTE V2X, pero LTE V2X solo admite comunicaciones V2V basada en el enlace SL de tipo broadcast, mientras que 5G NR V2X admite *groupcast* y *unicast*.

En el modo 1 de 5G NR V2X, como en el modo 3 de LTE V2X, la estación base celular (denominada gNB) es la encargada de gestionar y asignar los recursos radio que usarán los vehículos en las comunicaciones SL. En el modo 2 de 5G NR V2X, los vehículos son capaces de gestionar de forma autónoma y distribuida los recursos radio para las comunicaciones directas entre ellos. Por lo tanto, el modo 2 permite la comunicación 5G NR V2X SL sin cobertura celular. Este trabajo se centra en el modo 2, que se describe a continuación.

El modo 2 de 5G NR V2X tiene similitudes con el modo 4 de LTE V2X, pero con algunas diferencias. El modo 4 funciona de acuerdo con el esquema *sensing-based semi-persistent scheduling* (SPS) [5][6]. El modo 2, por otro lado, puede operar con un esquema de *scheduling* dinámico o semipersistente (*dynamic* o *semi-persistent scheduling*) diferente al implementado en el modo 4 de LTE V2X. El esquema dinámico selecciona un nuevo recurso para cada TB o paquete y solo reserva recursos para las retransmisiones de ese paquete. El esquema de *scheduling* semipersistente permite que un UE reserve recursos para las transmisiones de múltiples paquetes (el número está definido por la variable *Reselection Counter*) y de sus retransmisiones [4]. Una distinción importante para comprender correctamente el modo 2 es la diferencia entre recursos seleccionados y reservados. En el modo 2 los recursos se seleccionan utilizando el esquema de *scheduling* dinámico o semipersistente. Los recursos seleccionados son reservados para transmisiones futuras[1] cuando los informa el UE en el SCI de primera etapa.

[1] En el modo 2 de 5G NR V2X la información de control se intercambia entre los vehículos utilizando el elemento SCI (*Sidelink Control Information*). A diferencia de LTE V2X, donde toda la información de control está contenida en un SCI, 5G NR V2X divide la información de control en dos SCIs (primera etapa y segunda etapa). El SCI de primera etapa contiene información útil para decodificar el TB e información sobre las reservas realizadas para transmisiones futuras. El SCI de segunda etapa contiene información de

El modo 2 usa casi el mismo procedimiento para seleccionar recursos para los esquemas de *scheduling* dinámico y semipersistente [4]. La diferencia principal consiste en que el esquema de *scheduling* dinámico solo selecciona recursos para las retransmisiones de un paquete y en el esquema de *scheduling* semipersistente se selecciona recursos para un número de *Reselection Counter* paquetes consecutivos. La separación temporal entre los recursos seleccionados en el esquema de *scheduling* semipersistente está definida por el *Resource Reservation Interval* (RRI) que puede tomar valores de {0, [1:99], 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000} ms. Tanto en el esquema de *scheduling* dinámico como el semipersistente, durante el procedimiento para seleccionar recursos, el UE primero crea la ventana de selección donde identifica recursos candidatos para la transmisión de un TB (ver Figura 3). La ventana de selección incluye todos los recursos entre las ranuras $t = n + T_1$ y $t = n + T_2$, donde la ranura n se refiere al instante en el cual se inicia el proceso de selección de nuevos recursos [3]. Esta selección se puede activar por la generación de un nuevo paquete o la necesidad de seleccionar nuevos recursos porque un nuevo paquete supera en número de subcanales a los recursos previamente reservados. T_1 es el tiempo de procesamiento requerido por el UE para identificar recursos candidatos y seleccionar nuevos recursos para transmitir en el enlace SL. El valor de T_2 se deja a la implementación del UE, pero debe estar en el rango $T_{2min} \leq T_2 \leq$ PDB. Aquí, PDB (*Packet Delay Budget*) representa el límite de latencia para enviar el paquete, y T_{2min} se determina en función de la prioridad del paquete a transmitir. Una vez que se define la ventana de selección, el UE necesita identificar los recursos candidatos dentro de la ventana de selección. Cada recurso candidato está definido por una ranura de tiempo y un número (*LPSSCH*) consecutivo de subcanales de frecuencia para adaptarse al tamaño del paquete que se transmitirá.

control útil para dar soporte a los mecanismos introducidos en 5G NR V2X para mejorar la fiabilidad de las transmisiones groupcast y unicast.

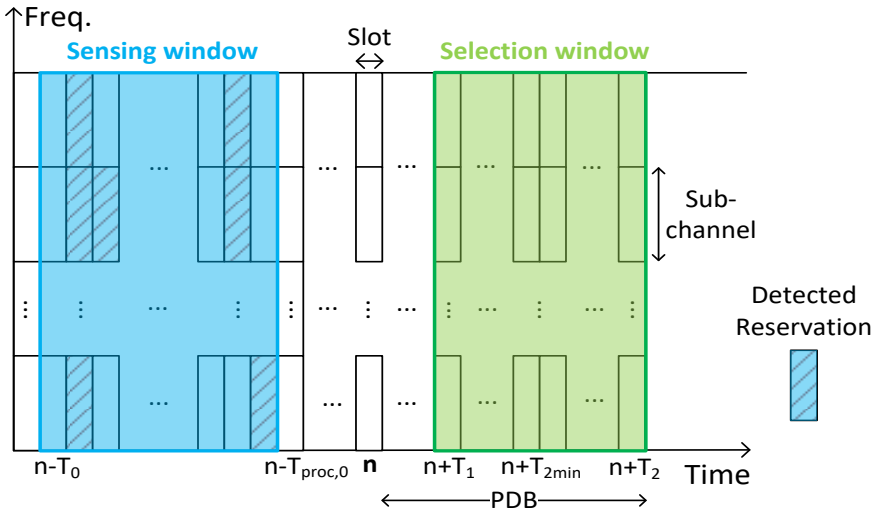

Figura 3. Representación de las ventanas de selección (selection window) y de sensado (sensing window) del modo 2 de asignación de recursos de 5G NR V2X (ejemplo para el caso de que T_2= PDB)

La información que un UE puede obtener de transmisiones vecinas se usa para identificar los recursos candidatos dentro de la ventana de selección. De hecho, cuando no está transmitiendo, el UE escucha continuamente los canales y recursos de SL captando información importante utilizada para identificar recursos candidatos. Esto se llama sensado del canal y da como resultado una ventana de detección (ver Figura 3, el tamaño de la ventana de detección viene dado por el parámetro *T0* y puede tomar valores de 1100 ms o 100 ms). Durante el proceso de detección, el UE decodifica la información de control contenida en el SCI (*Sidelink Control Information*) del TB recibido. El SCI indica los recursos de SL reservados por otros UE para la transmisión de TB. Además, el UE que realiza el proceso de detección mide el nivel de la señal o RSRP (*Received Signal Received Power*) de los TBs recibidos de otros UEs. Un UE almacena información recibida de otros UEs para determinar su ventana de detección y selecciona recursos de la ventana de selección cuando inicia un nuevo proceso de selección de recursos.

Para el proceso de selección de nuevos recursos, el modo 2 de gestión de recursos de 5G NR V2X define un algoritmo de 2 pasos que utiliza esquemas de *scheduling* dinámico y semipersistente.

3.2.1. PASO 1: EXCLUSIÓN DE RECURSOS EN LA VENTANA DE SELECCIÓN

En el paso 1 se descartan los recursos de la ventana de selección que se corresponden a las hipotéticas reservas realizadas por los vehículos circundantes. Estas reservas hacen referencia a las anunciadas en los paquetes que son transmitidos simultáneamente que las transmisiones pasadas del vehículo que está seleccionando recursos para la transmisión

del nuevo paquete, cuando éste no puede recibirlos, debido a las características de la transmisión half-dúplex. En concreto, el vehículo descarta todos los recursos pertenecientes a las ranuras resultantes de la expresión $s_i + q$ x RRI_i. En la expresión citada, s_i es cualquier ranura de la ventana de detección en la cual el vehículo estaba transmitiendo, RRI_i representa todos los posibles valores de la lista preconfigurada de RRI permitidos, hasta un máximo de 16 valores. Y la variable q puede tomar valores entre 1 $\leq q \leq Q$. El valor de Q se calcula como T_2 / RRI_i si se cumplen las siguientes condiciones: 1) $RRI_i < T_2$ (en ms) y 2) $n - s_i \leq RRI_i$ (en ranuras), siendo n la ranura en la que el vehículo selecciona recursos para la transmisión del paquete actual. En caso contrario, Q toma el valor 1. El valor de Q representa el número de transmisiones periódicas estimadas de otros vehículos para cada potencial RRI, de forma que puedan colisionar con los recursos de la ventana de selección. Si se está usando el esquema de *scheduling* semipersistente, adicionalmente se descartan los recursos correspondientes a las ranuras s_j de la ventana de selección para las cuales se cumple la igualdad $s_j + j$ x $RRI_{TX} = s_i + q$ x RRI_i. En la ecuación anterior RRI_{TX} es el RRI que usará el vehículo que está seleccionando recursos para la transmisión del paquete actual y j puede tomar valores entre $1 \leq j \leq 10$ x *ReselectionCounter* - 1.

El UE (o equipo de usuario) también tiene que descartar los recursos de la ventana de selección reservados por otros vehículos (usando el 1st-stage SCI). La exclusión de los recursos en la ventana de selección en este caso son las correspondientes a las reservas incluidas en los paquetes recibidos, cuyo nivel de señal o RSRP medido es superior a un umbral. Este proceso se procede de la siguiente forma. Se ha recibido un paquete (1st-stage SCI) con un nivel de señal superior al umbral de RSRP en la ranura s_k de la ventana de detección o sensado. El recurso reservado se corresponde a la ranura $s_k + RRI_{RX}$, siendo RRI_{RX} el RRI incluido en el paquete recibido. Entonces el vehículo receptor del paquete descarta los recursos (subcanales) de la ventana de selección que se superponen en frecuencia con el paquete recibido en las ranuras $s_k + q$ x RRI_{RX}, la variable q es incluida en la expresión, porque los otros vehículos podrían mantener los mismos recursos para sucesivas transmisiones periódicas de diferentes paquetes. Adicionalmente, al igual que en la operación half-dúplex, si se trata del esquema semipersistente, el UE que selecciona recursos también tendrá que descartar los recursos de la ventana de selección cuyas reservas futuras ($s_p + j$ x RRI_{TX}) puedan coincidir, fuera de la ventana de selección, con

las posibles transmisiones futuras (s_k + q x RRI_{RX}) del vehículo del cual se recibió el paquete, siendo j y q las variables definidas anteriormente.

El porcentaje de recursos candidatos restantes en la ventana de selección después del proceso de descarte de recursos debe ser superior al umbral de 20, 35 o 50 %. El porcentaje aplicado depende de la prioridad del paquete actual del vehículo que está seleccionando recursos para la transmisión. Si no se cumple el caso, entonces se repite el proceso de exclusión de recursos del paso 1, incrementando en 3 dB el umbral de RSRP de referencia para la exclusión de los recursos reservados por otros vehículos. Este incremento se repite mientras no se alcance el porcentaje requerido.

3.2.2. PASO 2: SELECCIÓN DE RECURSOS

Tras la finalización del proceso de descarte de recursos del paso 1, se incluyen los recursos libres de la ventana de selección a la lista de recursos candidatos. Entonces el vehículo selecciona aleatoriamente un recurso de la lista de recursos candidatos para la transmisión del paquete. Si está activada las retransmisiones, el proceso para continuar seleccionando aleatoriamente recursos es de la siguiente forma. A partir de la ranura m_1 del primer recurso seleccionado aleatoriamente como primera referencia, el segundo recurso seleccionado debe situarse entre las ranuras [m_1-31, m_1+31] y el tercer recurso seleccionado debe situarse entre las ranuras [m_1-31, m_1+31] y [m_2-31, m_2+31] y así sucesivamente. Es decir, para seguir seleccionando recursos aleatoriamente, se toma como referencia las ranuras de todos los recursos previamente seleccionados. En caso de tener que seguir seleccionando recursos, pero ningún recurso libre cumple con las restricciones anteriores, entonces el resto de los recursos se seleccionan de forma totalmente aleatoria.

3.2.3. RETRANSMISIONES

En el modo 2 del estándar 5G NR V2X se ha añadido la posibilidad de retransmitir el mismo paquete dentro de la misma ventana de selección. En este modo, que se aplica tanto al esquema dinámico como al esquema semipersistente, el vehículo selecciona N recursos candidatos ($N \leq N_{MAX}$ con $1 \leq N_{MAX} \leq 32$) para la primera transmisión del paquete y sus (N - 1) retransmisiones. Este proceso sigue el algoritmo de 2 pasos explicado en las subsecciones anteriores. La elección de N depende de la implementación del UE y debe ser menor que el número de recursos candidatos disponibles después del paso 1 del modo 2. Por otro lado, el valor de N_{MAX} se puede ajustar según la carga del canal. En cada

transmisión o retransmisión del paquete puede incluir hasta N_{SCI} recursos. El valor de N_{SCI} puede estar configurado a 2 o 3 recursos. El primer recurso es la ranura de transmisión del paquete y los siguientes recursos son las reservas para las siguientes retransmisiones del mismo paquete. La diferencia máxima entre la ranura de transmisión y la última reserva de retransmisión incluida en el paquete son de 31 ranuras (ver Figura 4). Es decir, para poder reservar las siguientes retransmisiones programadas durante la transmisión de un paquete, estas retransmisiones no pueden distar más de 31 ranuras respecto a la ranura de transmisión del paquete.

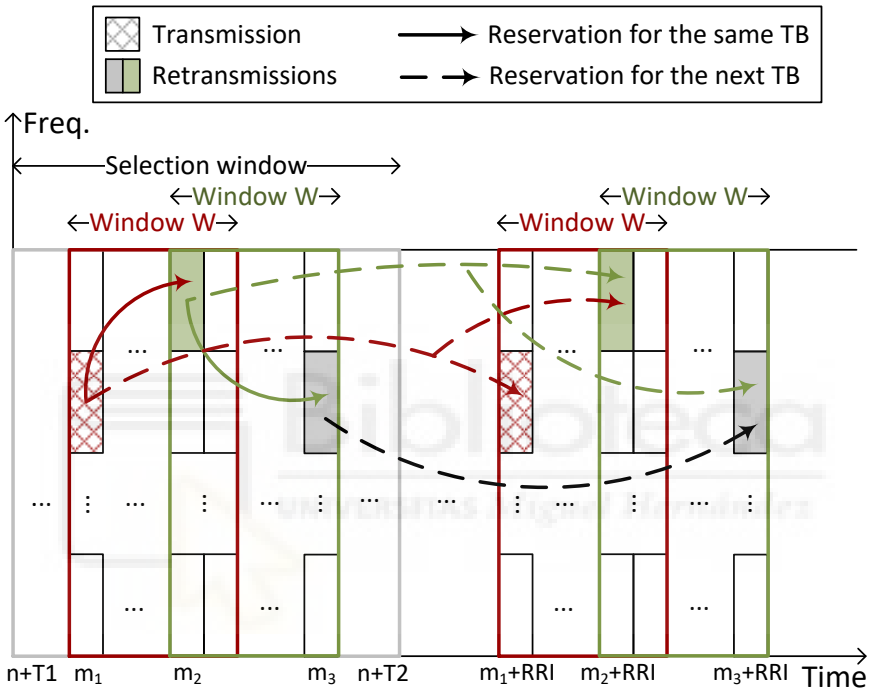

Figura 4. Proceso de selección de recursos radio para las N transmisiones (transmisión inicial y N-1 retransmisiones) en modo 2

3.2.4. RE-EVALUATION

Un UE que ha seleccionado un nuevo recurso SL (por ejemplo, en la ranura *m*) continúa detectando las transmisiones de otros UE durante la ventana de selección. El UE puede decidir si aplicar otra vez el paso 1 para comprobar si el recurso seleccionado aún sigue disponible. El UE puede volver a ejecutar el paso 1 en la ranura m - T_3 o antes. T_3 es el tiempo máximo permitido (en ranuras) para que un UE complete el proceso de selección de recursos. La Figura 5 muestra el proceso de ejecutar nuevamente el paso 1. En la Figura 5, n' representa la ranura en la que el UE inicia una nueva ejecución del paso 1. El UE define una nueva ventana de selección (SW') que comienza en la ranura $n' + T_1$ y termina en la ranura $n' + T_2'$. T_2' debe estar en el rango $T_{2min} \leq T_2' \leq n$ + PDB - n'. donde n es el

instante en el que se inició el proceso de selección de recursos (ver Figura 3). Entonces el UE realiza el paso 1 sobre los recursos candidatos en SW' para identificar qué recursos están disponibles dentro de esa ventana. Si se excluye el recurso previamente seleccionado en la ranura *m*, esta detección se le conoce como *re-evaluation* [3] en los estándares 3GPP. La detección de reevaluación puede ocurrir en esquemas de *scheduling* dinámico o semipersistente. Cuando se detecta *re-evaluation*, el UE procede a ejecutar el paso 2 nuevamente (ver sección 3.2.2) y selecciona un nuevo recurso SL entre los recursos disponibles en la ventana SW' [7]. Si el recurso seleccionado originalmente (en la ranura *m*) todavía está disponible cuando se reinicia el paso 1, el UE no volverá a ejecutar el paso 2.

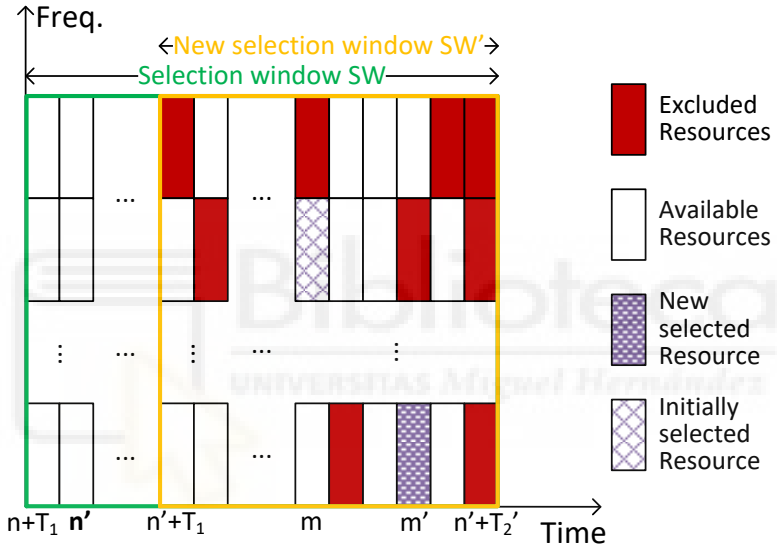

Figura 5. Nueva ejecución del paso 2 cuando se detecta re-evaluation en el modo 2 de gestión de recursos de 5G NR V2X

4. PLATAFORMA OPENAIRINTERFACE

4.1. INTRODUCCIÓN

OAI es un proyecto de código abierto destinado a proporcionar software de radio digital para comunicaciones inalámbricas de próxima generación. OAI se creó en octubre de 2012 por iniciativa de Eurecom, un instituto de investigación de telecomunicaciones con sede en Francia.

Hoy en día, el proyecto OAI está liderado por Eurecom, con la colaboración de numerosas instituciones y empresas de todo el mundo. Las principales instituciones involucradas en el proyecto OAI son el Instituto Fraunhofer Heinrich Hertz (HHI) y la Universidad Técnica de Berlín (TU Berlín) en Alemania y la Universidad de Cantabria en España.

El software de radio OAI se usa cada vez más en una variedad de aplicaciones y entornos, desde la creación de prototipos y análisis de redes de próxima generación hasta la enseñanza y la formación en telecomunicaciones.

La plataforma OAI implementa en lenguaje C los protocolos de las diferentes capas de comunicación del terminal (UE) y también los diferentes componentes del núcleo de la red, así como las estaciones base. En este proyecto trabajaremos solo sobre la implementación del UE, dado que las comunicaciones *Sidelink* o PC5 no requieren comunicación del terminal con la estación base y la infraestructura.

4.2. PROYECTOS

Diversos proyectos e iniciativas están en marcha en el marco de desarrollo de OAI.

El proyecto 5G CORE NETWORK de OAI es una iniciativa para construir una red de comunicaciones inalámbricas de código abierto de quinta generación (5G). Las redes 5G deberían ofrecer una mayor capacidad y velocidades de datos, así como una mejor eficiencia energética. Las redes 5G deberían ser más rápidas y más eficientes energéticamente que el 4G LTE actual.

El proyecto MOSAIC5G de OAI es una iniciativa de investigación y desarrollo de software destinada a desarrollar redes de comunicaciones inalámbricas de quinta generación (5G) de código abierto. Las redes 5G están diseñadas para ofrecer una mayor capacidad de ancho de banda y un rendimiento mejorado en comparación con las redes 4G y anteriores.

4.3. INSTALACIÓN

El entorno empleado para la ejecución de OAI es el sistema operativo Linux versión Ubuntu 16.04.7 LTS (*Xenial Xerus*) con el kernel 4.8.0-53-lowlatency. Una vez descargado la ISO ubuntu-16.04.7-desktop-amd64.iso e instalado Ubuntu, se procederá a descargar actualizaciones, a instalar el kernel citado anteriormente y después de reiniciar Ubuntu desde el nuevo kernel instalado, se borrarán los kernels instalados por defecto en el sistema operativo, para ello se introducirá por terminal los siguientes comandos:

sudo apt update && sudo apt upgrade
sudo apt-get update

sudo apt-get install linux-image-4.8.0-53-lowlatency
sudo apt-get install linux-headers-4.8.0-53-lowlatency
dpkg -l | grep linux-image
sudo update-grub
sudo reboot now

dpkg -l | grep linux-image
sudo apt-get remove linux-image-4.15.0-112-generic
sudo apt-get purge linux-modules-4.15.0-112-generic
sudo apt-get purge linux-modules-extra-4.15.0-112-generic
sudo apt-get purge linux-image-4.15.0-112-generic
sudo apt-get autoremove

sudo apt-get remove linux-image-4.15.0-142-generic
sudo apt-get purge linux-modules-4.15.0-142-generic
sudo apt-get purge linux-modules-extra-4.15.0-142-generic
sudo apt-get purge linux-image-4.15.0-142-generic
sudo apt-get autoremove

Para trabajar con OAI, se instala la herramienta git y se descarga su código del repositorio *https://github.com/KaimingChen95/OAI_5G_NR_V2X.git*. Una vez instalado git y clonado el repositorio, se requiere instalar una serie de funcionalidades y librerías

necesarias para la compilación y ejecución de OAI. A continuación, se muestran los comandos a introducir en la terminal:

#Instalación git y clonación de repositorio
sudo apt-get install git
git clone https://github.com/KaimingChen95/OAI_5G_NR_V2X.git

#Instalación de las funcionalidades y las librerías necesarias
cd OAI_5G_NR_V2X/v2x_mac/openairinterface5g
source oaienv
cd cmake_targets
sudo apt-get install cmake libblkid-dev e2fslibs-dev libboost-all-dev libaudit-dev
sudo apt-get install mosquitto
sudo apt-get install mosquitto mosquitto-clients
sudo apt-get install libmosquitto-dev
sudo apt-get install libjson-c-dev
sudo add-apt-repository ppa:ettusresearch/uhd
sudo apt-get update
sudo ./build_oai -I -w USRP

#Compilación
cd ../../../
sudo ./compile
cd v2x_mac/openairinterface5g/cmake_targets/lte_build_oai/build
sudo make lte-uesoftmodem –j4

#Ejecución
cd ../../../../../
sudo ./run.sh

5. DISEÑO E IMPLEMENTACIÓN

5.1. METODOLOGÍA DE TRABAJO

En la implementación del código, primero se programa las funcionalidades de la capa MAC del estándar 5G NR V2X de forma aislada para testear y verificar el correcto funcionamiento antes de integrarlo en OAI. Este código, que denominaremos *código local* integra la capa MAC de 5G NR V2X en un simulador ligero computacionalmente que permite simular la transmisión y recepción de paquetes por múltiples vehículos, considerando un modelo de propagación y las interferencias que puedan generarse entre ellos. De esta forma, el *código local* nos permite validar completamente el código de la capa MAC 5G NR V2X implementado, no solo mediante una depuración más sencilla, sino también a través de su ejecución para realizar simulaciones y compararlas con otras simulaciones o modelos analíticos. La implementación se ha realizado de forma que la capa MAC implementada pueda extraerse de ese *código local* para su integración posterior en OAI.

Una vez comprobado su correcto funcionamiento, se procede a integrar la capa MAC de 5G NR V2X de ese *código local* en el repositorio del proyecto original de OAI. En esta integración se ha introducido nuevos parámetros de configuración correspondientes al estándar 5G NR V2X, se ha redefinido las estructuras empleadas y añadidas otras nuevas y las funciones para gestionar la memoria dinámica de las nuevas estructuras, se incluye flujos de ejecución alternativos para 5G NR V2X, también se añade el registro de la PDR para la capa aplicación, que se detallará más en adelante.

Durante el proceso de desarrollo se ha realizado un seguimiento de las tareas implementadas en una planificación online. Esta planificación ha servido para identificar y organizar los trabajos a realizar y ha sido muy útil para marcar objetivos y cumplir hitos. En la Figura 6 se muestra la planificación online dividida en 4 fases. Las tareas a implementar en cada nueva fase tienen una complejidad incremental.

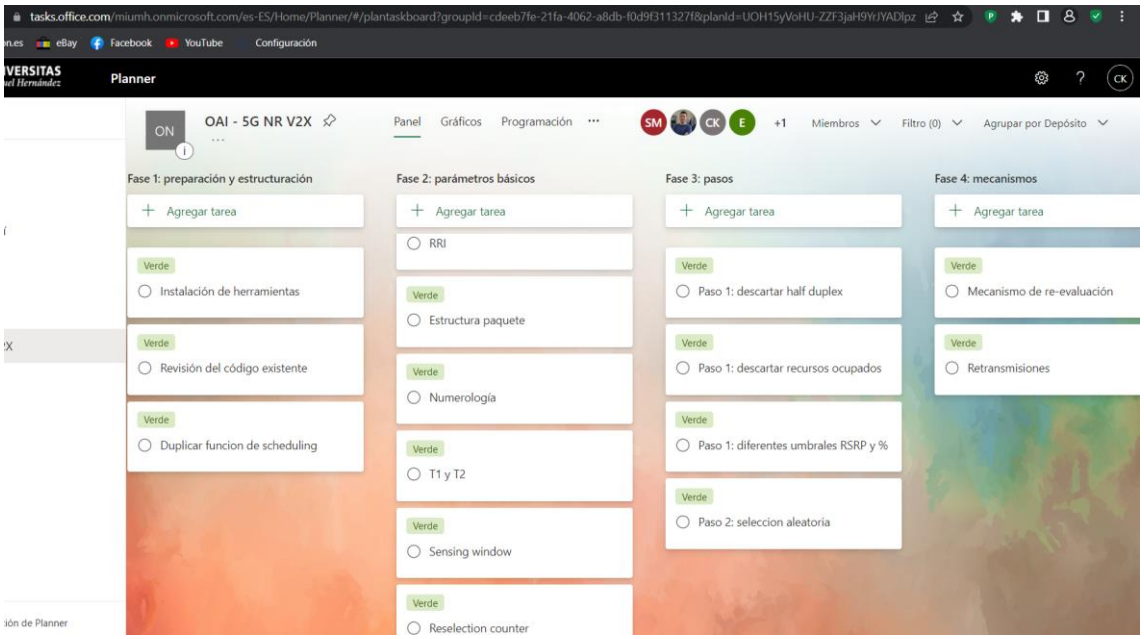

Figura 6. Planificación online de las tareas a implementar

5.2. IMPLEMENTACIÓN DE LA CAPA MAC DE 5G NR V2X

La función `nr_v2x_scheduler` es la función principal en la que se ha implementado el mecanismo *sensing-based semi-persistent scheduling* del modo 2 de 5G NR V2X. Esta función básicamente se utiliza para seleccionar aleatoriamente recursos disponibles para las transmisiones del paquete actual en función de las reservas realizadas por los vehículos del entorno. A continuación, se mostrará un diagrama de flujo de alto nivel de una serie de llamadas internas de las funciones principales a partir de la función `nr_v2x_scheduler`. Como puede verse en la Figura 7, las funciones siguen una jerarquía de llamadas, de forma que una función de un nivel superior llama a una o varias funciones del nivel inmediatamente inferior. La función `nr_v2x_scheduler` puede continuarse por 2 flujos, que se detallarán en las subsecciones siguientes. El recuadro que engloba a las funciones `calculate_vectorQ` y `nr_v2x_halfduplex` indica que se ejecutan en conjunto y pertenecen al mismo nivel dentro de la jerarquía. La función `nr_v2x_halfduplex`, al igual que la función `nr_v2x_discard_reserved_resources`, llama a las funciones `calculate_qMin`, `discard_resources` y `discard_overlapping_resources` para el descarte de recursos de la ventana de selección en el paso 1 del algoritmo, cuyos detalles de funcionamiento se explicarán en las subsecciones siguientes.

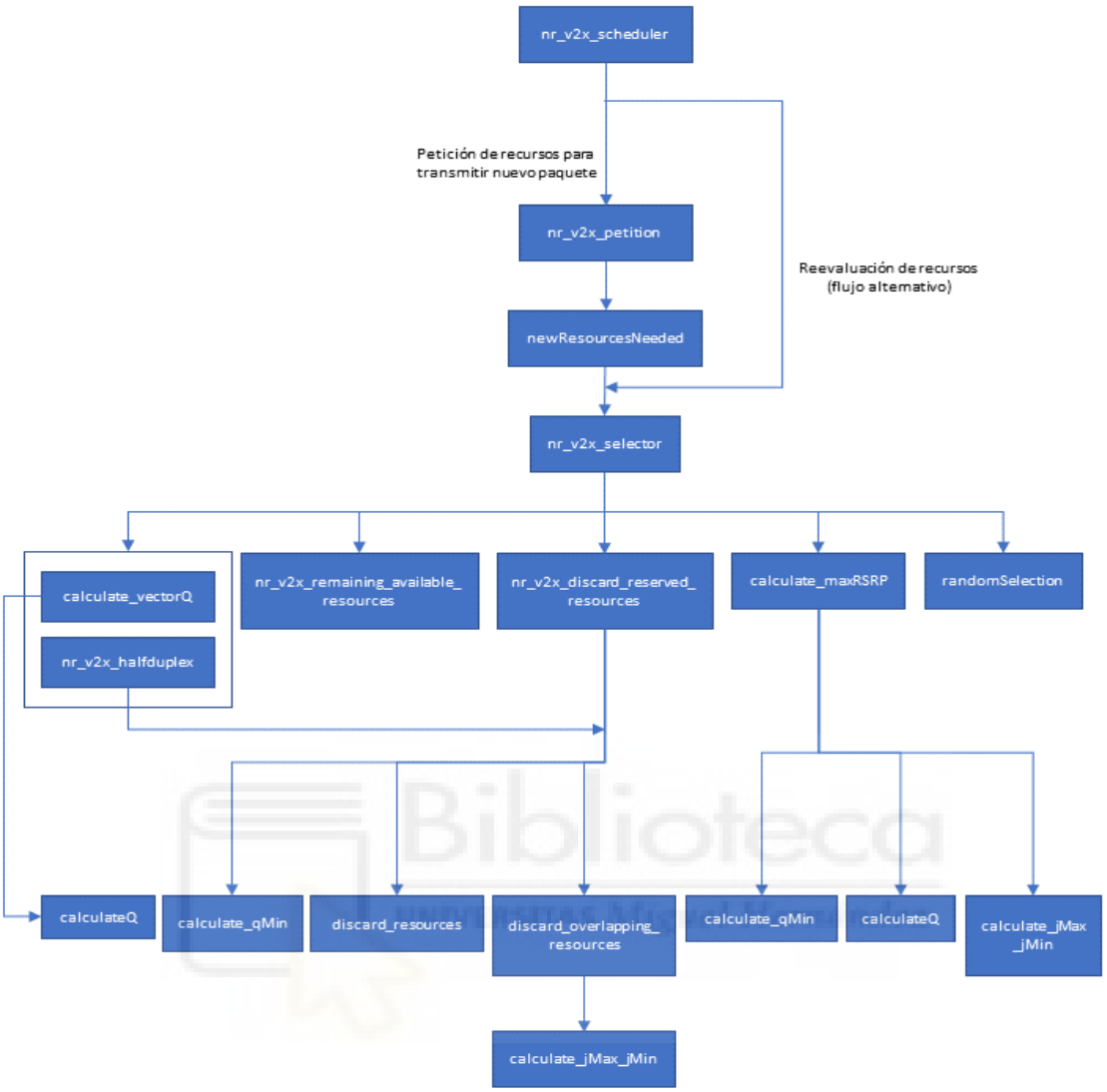

Figura 7. Flujo de llamadas de las funciones principales

La función `nr_v2x_scheduler` puede ser llamada en dos casos en la implementación propuesta. En el primer caso, la función `nr_v2x_scheduler` es llamada cuando llega una petición de reservar recursos desde capas superiores a la capa MAC para realizar la primera transmisión y las retransmisiones de un nuevo paquete. En el primer caso, la función `nr_v2x_scheduler` es llamada si el instante actual coincide con la variable `tPetition` o instante de petición de nuevos recursos (i.e., `tActual == tPetition`), que se irá actualizando en función de si es tráfico periódico o tráfico aperiódico. En el segundo caso, la función `nr_v2x_scheduler` es llamada cuando se ejecuta el mecanismo de reevaluación para identificar la disponibilidad de los recursos seleccionados previamente en el primer caso. Este caso se produce cuando el instante actual (representado en la

implementación de OAI mediante la variable `tick_proxy`) coincide con el instante de reevaluación (representado en la implementación de OAI mediante la variable `tReevaluation`). En la Figura 8 se observa el flujo interno de ejecución de la función `nr_v2x_scheduler`, así como las llamadas internas a otras funciones.

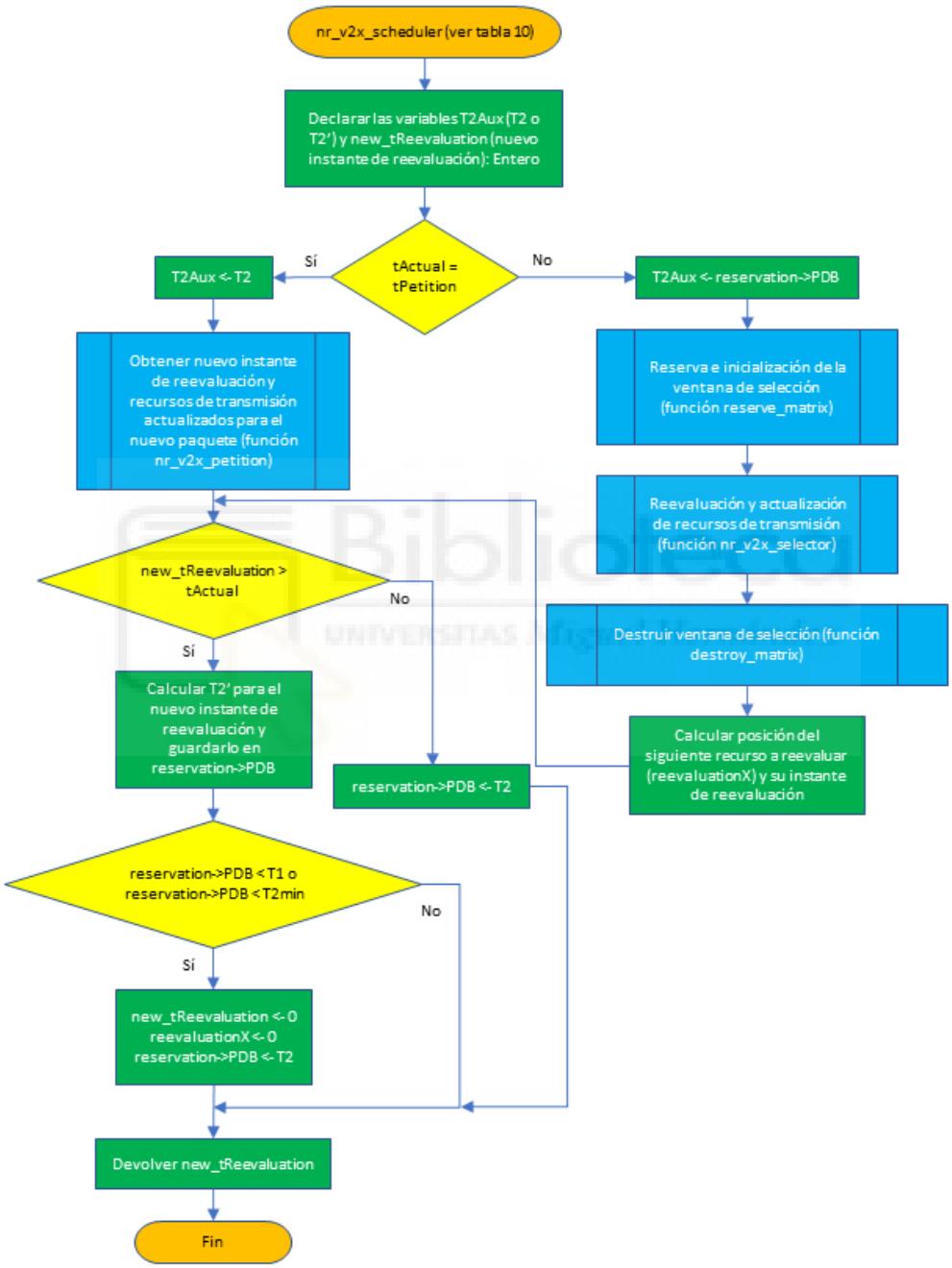

Figura 8. Diagrama de flujo de la función nr_v2x_scheduler

La función `nr_v2x_scheduler` recibe catorce parámetros de entrada los cuales se pueden agrupar según su uso/significado en los siguientes grupos (Tabla 1):

- Las variables `tPetition` y `tActual` son el instante de petición de nuevos recursos y el instante actual, respectivamente.
- Las estructuras `sensingWindow`, `reservation`, `retransmissions`, `retransmissionsPrevious`: el primero guarda los paquetes recibidos de otros vehículos, el segundo guarda el siguiente paquete a transmitir, el tercero guarda la lista de recursos seleccionados para las retransmisiones del paquete y el último guarda los recursos transmitidos de paquetes anteriores.
- Las variables `T1`, `T2` y `T3` almacenan valores relativos que se utilizan para: los dos primeros delimitan la ventana de selección (en la magnitud tiempo) a partir de `tActual` como referencia [`tActual + T1`, `tActual + T2`], el tercero permite calcular el instante último de reevaluación de un recurso seleccionado a partir de su instante de transmisión (i.e., `retransmissions->list[reevaluationX].subframe - T3`).
- Las variables `reevaluationX` y `retransmissionX` indican las posiciones de los recursos de la lista `retransmissions` para su reevaluación y la siguiente retransmisión del mismo paquete, respectivamente.
- Las variables `packetLength` y `Nsubchannel` que hacen referencia al tamaño del paquete en subcanales y el número de subcanales configurados en la capa MAC de 5G NR V2X.
- Por último, la variable `last_RSRPthreshold` es el umbral de RSRP del cual se obtiene la lista de recursos candidatos con RSRP medido menor o igual que el umbral en la previa selección de recursos.

Por otro lado, la función `nr_v2x_scheduler` devuelve el instante de reevaluación de los recursos seleccionados (i.e., `tReevaluation`). Además, dentro de la función se actualiza la lista de recursos para las retransmisiones del paquete.

Tipo Dato	Parámetro	Funciones	Uso/Significado
int	tPetition	nr_v2x_scheduler	Instante de petición de nuevos recursos
int	tActual	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Instante actual
listResource*	sensingWindow	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Lista de paquetes recibidos

reservation_t*	reservation	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Siguiente paquete para transmitir
listRetransmission*	retransmissions	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Lista de recursos seleccionados
listRetransmission_prev*	retransmissionsPrevious	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Lista de recursos transmitidos de paquetes anteriores
int	T1	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Ranura inicial de la ventana de selección (valor relativo)
int	T2	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Ranura final de la ventana de selección (valor relativo)
int	T3	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Último instante de reevaluación de un recurso (valor relativo)
int*	reevaluationX	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded	Posición del siguiente recurso para reevaluar
int	reevaluationX	nr_v2x_selector	Posición del siguiente recurso para reevaluar
int*	retransmissionX	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded	Posición del siguiente recurso para retransmitir
int	retransmissionX	nr_v2x_selector	Posición del siguiente recurso para retransmitir
int	packetLength	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded	Tamaño del nuevo paquete en subcanales
int	Nsubchannel	nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded, nr_v2x_selector	Número total de subcanales configurados
stSelectionWindow**	selectionWindow	nr_v2x_selector	Ventana de selección
bool	Reev	nr_v2x_selector	Indica si es (o no) una reevaluación

Tabla 1. Parámetros de entrada de las funciones nr_v2x_scheduler, nr_v2x_petition, newResourcesNeeded y nr_v2x_selector

5.2.1. CASOS POSIBLES EN UNA PETICIÓN DE NUEVOS RECURSOS

La implementación dentro de la función `nr_v2x_scheduler` identifica primero la razón o el caso por el que la función ha sido llamada, es decir, se comprueba si es una petición de recursos para la transmisión de un nuevo paquete o una reevaluación de recurso. Si se trata de una petición para la transmisión de un nuevo paquete (i.e., `tActual == tPetition`), entonces se ejecuta la función `nr_v2x_petition`. La función `nr_v2x_petition` primero decrementa en uno el contador *reselection counter* (identificado con la variable `RC` dentro de la estructura `reservation`, i.e., `reservation->RC`). Después se contempla tres escenarios o casos generales posibles:

- El primer caso se produce cuando el *reselection counter* llegue a cero y el RRI usado en la transmisión del paquete anterior sea igual a cero también o si el tamaño del nuevo paquete (i.e., `packetLength`) es mayor al tamaño del paquete anterior (i.e., `reservation->length`). El RRI es el intervalo de reserva de recursos representado en ranuras.
- Para el segundo caso se cumple que al menos una de las reservas en las retransmisiones del paquete anterior se queda fuera de la nueva ventana de selección.
- En el tercer o último caso se mantienen todos los recursos anunciados en las reservas realizadas en las retransmisiones del paquete anterior.

Estos casos pueden observarse en la Figura 9.

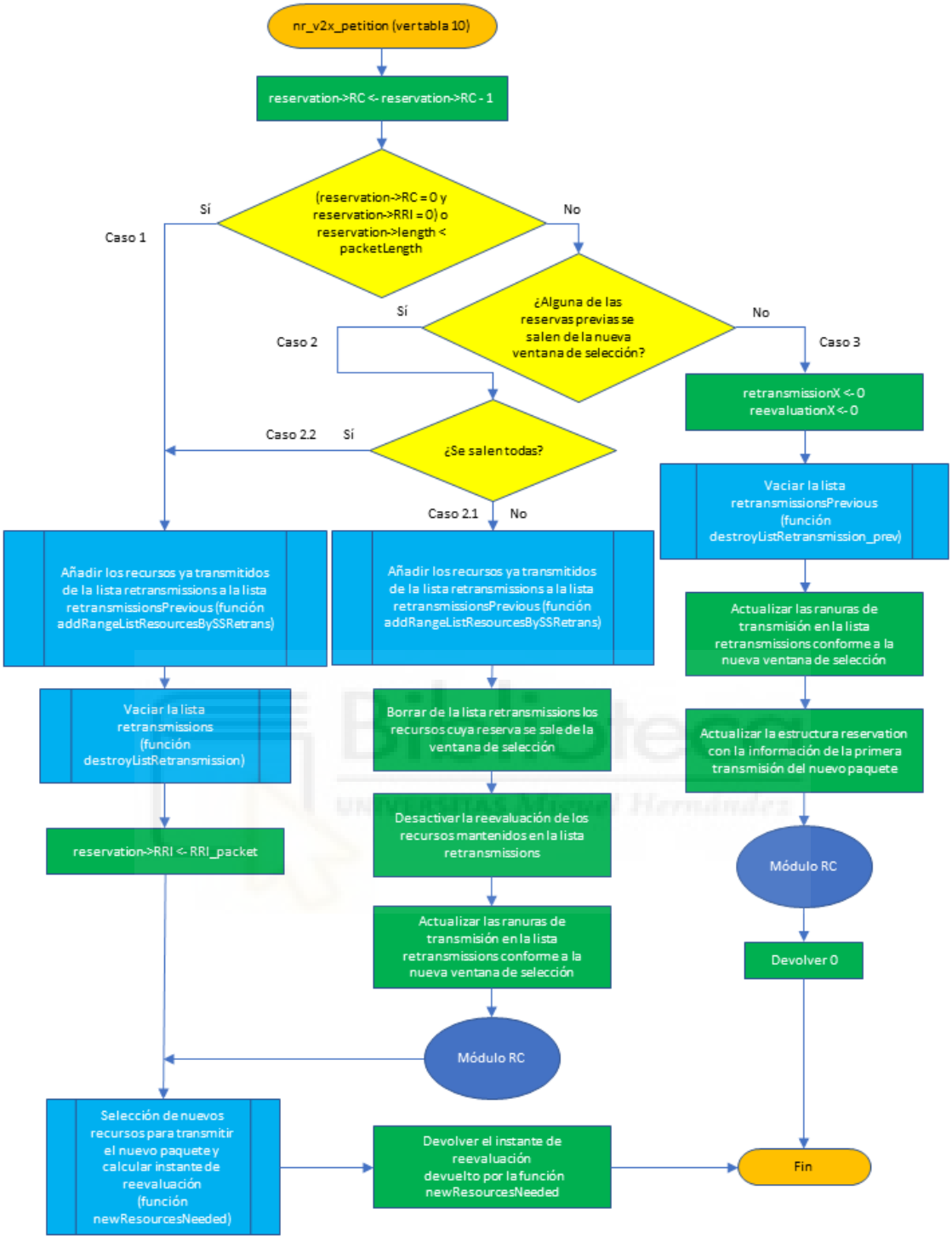

Figura 9. Diagrama de flujo de la función nr_v2x_petition

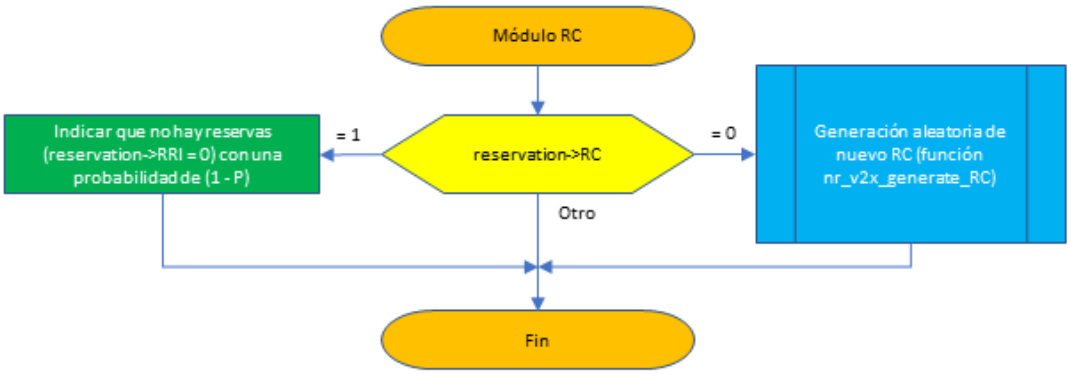

Figura 10. Módulo contador de reselección

Hay tres situaciones a partir de las cuales se puede llegar al primer caso:

- La primera se da una única vez por vehículo y ocurre cuando se hace una petición de nuevos recursos por primera vez, es decir, al llamar a la función `nr_v2x_petition` para seleccionar nuevos recursos para la transmisión del primer paquete. Para ello, al inicio del programa se ha inicializado las variables `RC`, `RRI` y `length` de la estructura `reservation` a 1, 0 y 0, respectivamente.
- La segunda situación se llega cuando en la penúltima llamada a la función `nr_v2x_petition` el *reselection counter* es 1 y el número aleatorio (i.e., `reselProb`) generado entre 0 y 10 (ambos incluidos) sea menor o igual a la variable `P_const`, en tal caso el RRI asignado al anterior paquete transmitido es cero (Figura 10). El valor de `P_const` se calcula con la siguiente expresión: 10 * (1 - *P*), donde (1 - *P*) es la probabilidad de seleccionar nuevos recursos cuando el *reselection counter* llegue a cero. Cada vehículo puede configurar *P* entre 0 y 0,8 (ambos incluidos). Un RRI igual a cero indica a los demás vehículos que no se reserva los mismos recursos para las retransmisiones del paquete siguiente. Si `reselProb` es mayor a `P_const`, entonces se mantendrá los mismos recursos para la transmisión del paquete siguiente.
- En la tercera situación el tamaño del nuevo paquete es mayor que el último paquete transmitido.

Una vez comprobado que se trata del primer caso, primero se calcula el índice del último recurso transmitido (i.e., `posMax`) de la lista retransmisiones, es decir, el último cuyo subframe es menor al instante actual. Entonces se añaden todos los recursos de retransmisiones con subframe menor al instante actual a la lista

`retransmissionsPrevious`, se borran todos los recursos de la lista de retransmisiones previa (i.e., `retransmissions`), se actualiza el RRI de las reservas (i.e., `reservation->RRI = RRI_packet`) y al final devuelve el instante de reevaluación devuelto por la llamada a la función `newResourcesNeeded`.

En el segundo caso hay dos posibilidades. En la primera de ellas, parte de las reservas anunciadas en el paquete anterior se queda fuera de la nueva ventana de selección y la otra parte se queda dentro de la ventana. En esta primera posibilidad primero se actualiza la lista `retransmissionsPrevious` como en el primer caso, después se mueve los recursos de la lista de retransmisiones, cuya reserva se queda dentro de la nueva ventana de selección, al principio de la lista y al final se redimensiona la lista a solo esos elementos o recursos. Se libera el vector `reevActiv` y se reserva un vector `reevActiv` con el mismo número de elementos que la lista anterior, inicializado a ceros. El vector `reevActiv` indica si está (o no) activada la reevaluación para cada recurso de la lista de retransmisiones. El valor *false* o 0, indica que está desactivada la reevaluación, en otro caso, la reevaluación está activada. También se actualiza el subframe o ranura de transmisión de los recursos mantenidos en la lista con los anunciados en las reservas (i.e., `retransmissions->list[i].subframe += reservation->RRI`, siendo `i = [0, retransmissions->size - 1]`). Antes de llamar a la función `newResourcesNeeded` y devolver el instante de reevaluación, si el *reselection counter* es 1, se realiza los mismos pasos que aquello que conduce a la segunda situación del primer caso, en otro caso, si el *reselection counter* es 0, se genera un nuevo valor para el *reselection counter* (Figura 10). En la segunda posibilidad, todas las reservas anunciadas en las retransmisiones del paquete anterior se salen de la nueva ventana de selección, entonces se realizan los mismos pasos que si se tratase del primer caso.

En el tercer o último caso, se resetea a cero los índices `retransmissionX` y `reevaluationX`, se borra la lista de recursos transmitidos por paquetes previos (i.e., `destroyListRetransmission_prev(retransmissionsPrevious)`), se actualiza el subframe de los recursos de la lista retransmisiones al igual que en la primera posibilidad del segundo caso, se asigna el nuevo tamaño del paquete (i.e., `reservation->length = packetLength`) y también se actualiza la estructura `reservation` con los primeros recursos de la lista retransmisiones para la primera transmisión del paquete actual. Los vectores `subframe` y `subchannel` de `reservation` pueden almacenar hasta un máximo de N_{SCI} valores. El valor máximo de N_{SCI} puede ser igual a 2 o 3. La posición 0 de esos

vectores indican el `subframe` y el `subchannel` o subcanal usado en la propia transmisión y las posiciones siguientes son las reservas para las siguientes retransmisiones del mismo paquete. Entre el primer subframe y cualquiera de los siguientes no puede haber una separación superior a 31 ranuras. La variable `sizeSub` de `reservation` guarda el número de posiciones usadas en los vectores `subframe` y `subchannel`. Cada subframe o ranura de transmisión de un paquete más el valor de RRI es una reserva de la ranura de una de las retransmisiones del paquete siguiente. Finalmente, antes de devolver el valor 0, para indicar que no habrá reevaluación de los recursos a transmitir, también se comprueba si el *reselection counter* es 1 o 0 y se aplican los mismos pasos que en la primera posibilidad del segundo caso (Figura 10). En la Figura 11 se observa la matriz de la ventana de selección, formada por `Nsubchannel` filas (o subcanales) y (`T2 - T1 + 1`) columnas (o ranuras).

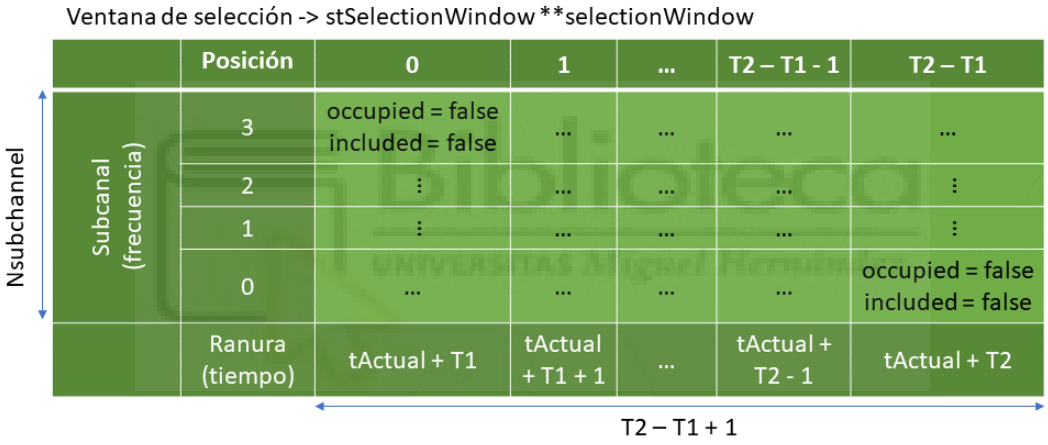

Figura 11. Ventana de selección inicializada

5.2.2. CÁLCULO DEL INSTANTE DE REEVALUACIÓN

La función `newResourcesNeeded` se encarga principalmente de calcular el nuevo instante de reevaluación. Primero crea una ventana de selección de `Nsubchannel` filas (o subcanales) y (`T2 - T1 + 1`) columnas (o ranuras), actualiza el nuevo tamaño del paquete (i.e., `reservation->length = packetLength`). Si todos los recursos de retransmisiones fueron descartados para el paquete actual (i.e., `retransmissions->size == 0`), genera un nuevo *reselection counter*. También se resetea a cero la variable `retransmissionX`. A continuación, se ejecuta la función `nr_v2x_selector`, que es la encargada de descartar los recursos de la ventana de selección aplicando las operaciones half-dúplex y descarte de recursos reservados por otros vehículos, de seleccionar aleatoriamente de la lista de

recursos candidatos los recursos que se usarán para las retransmisiones del paquete actual y de actualizar la información de la siguiente transmisión en la estructura `reservation`. Después de finalizar la ejecución de la función `nr_v2x_selector`, se libera la memoria reservada para crear (se destruye) la ventana de selección. Una vez finalizado la actualización de la lista retransmisiones con los nuevos recursos para la transmisión del paquete actual, se procede a calcular la posición (i.e., `reevaluationX`) del siguiente recurso a reevaluar, así como su instante de reevaluación. Para ello, primero se resetea a cero la variable `reevaluationX` y se comprueba si está activada la reevaluación de recursos. Si la reevaluación está desactivada (i.e., `reevaluation == false`), la función `newResourcesNeeded` devuelve 0, para indicar que no hay reevaluación. Por el contrario, si la reevaluación está activada, se recorrerá cada recurso de la lista retransmisiones para buscar el primer recurso a reevaluar. El recurso elegido para reevaluar cumple las siguientes condiciones:

- Tiene activada la reevaluación (i.e., `retransmissions->reevActiv[*reevaluationX] == true`, para `*reevaluationX = [0, retransmissions->size - 1]`).
- Su instante de reevaluación es posterior al instante actual (i.e., `retransmissions->list[*reevaluationX].subframe - T3 > tActual`).
- La diferencia entre su ranura de transmisión y la ranura de transmisión del recurso previo en la lista retransmisiones es superior a 31 ranuras (i.e., `retransmissions->list[*reevaluationX].subframe > retransmissions->list[*reevaluationX - 1].subframe + 31`). Esta última condición no se aplica al recurso de la posición 0.

Si ningún recurso cumple las condiciones anteriores, el índice `reevaluationX` toma el valor 0 y la función `newResourcesNeeded` devuelve 0, para indicar que no hay reevaluación. En la Figura 12 se muestra un ejemplo del cálculo del instante de reevaluación para el recurso 2 de la lista de recursos seleccionados para las retransmisiones del paquete actual.

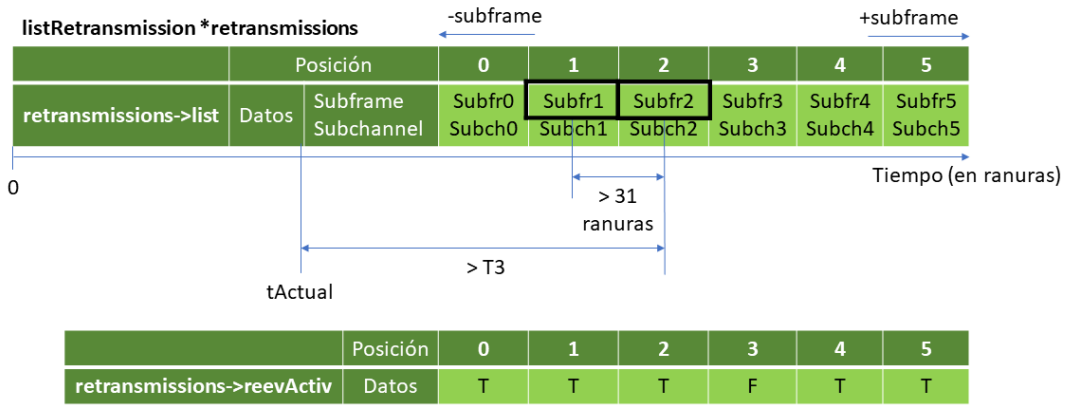

Figura 12. Ejemplo del cálculo del instante de reevaluación. Instante de reevaluación (tReevaluation = Subfr2 – T3) del recurso 2 (reevaluationX = 2)

5.2.3. PROCESO PARA EL DESCARTE DE RECURSOS

La función `nr_v2x_selector` es llamada tanto si hay petición de nuevos recursos como si hay reevaluación de recursos. Antes de aplicar las operaciones de descarte de recursos (half-dúplex y descarte de recursos reservados por otros vehículos), se actualiza la lista de recursos transmitidos de los paquetes previos (i.e., `retransmissionsPrevious`), borrando los recursos más antiguos de la lista. Se borran aquellos recursos cuyo subframe (en ranuras) más el RRI máximo de 1000 ms (pasado a ranuras) no llegue a la ranura del límite inferior (i.e., `tActual + T1`) de la ventana de selección.

Después de actualizar la lista `retransmissionsPrevious`, es ejecutado la operación half-dúplex (función `nr_v2x_halfduplex`) por primera vez y se comprueba la cantidad de recursos todavía libres en la ventana de selección (función `nr_v2x_remaining_available_resources`). Si no queda ningún recurso libre, entonces se desactiva la operación half-dúplex y se reinicia la ventana de selección, marcando todos los recursos como disponibles. Por otro lado, si la cantidad restante de recursos libres es inferior a X % (i.e., `occupiedCounter > occupiedThreshold`, el primero representa la cantidad de recursos ocupados y el segundo la cantidad máxima esperada de recursos ocupados), entonces se registra que no hay suficientes recursos disponibles (i.e., `enough_available_resources = false`) tras aplicar half-dúplex. El valor de X puede ser 20, 35 o 50, dependiendo de la prioridad del paquete del que está seleccionando recursos para la transmisión del paquete. En el mejor de los casos, si el porcentaje de recursos restantes tras aplicar half-dúplex es superior a X %, entonces se descarta los recursos reservados por otros vehículos (función

nr_v2x_discard_reserved_resources) cuyo RSRP medido del paquete recibido sea superior al RSRP umbral (i.e., RSRPthreshold).

La función nr_v2x_remaining_available_resources se encarga principalmente de actualizar la variable occupiedCounter y si el parámetro addAvailableResources es igual a *true*, también incluye los recursos libres de la ventana de selección a la lista de recursos candidatos (i.e., listRetransmission_prev listCSR).

A partir de aquí hay dos flujos de ejecución en función de si se trata de una petición de nuevos recursos o si es una reevaluación de recursos, que después de finalizar todo el proceso de descarte de recursos (paso 1 del algoritmo para seleccionar recursos), vuelven a converger en el paso 2, cuando se obtiene la lista de recursos candidatos para la selección aleatoria de los recursos que se incluirán en la lista retransmisiones.

En este punto, si es una petición de nuevos recursos, se ejecuta la función nr_v2x_remaining_available_resources para incluir los recursos libres a la lista de recursos candidatos. Con la lista listCSR actualizada, si enough_available_resources es igual a verdadero (i.e., true) se comprueba si el porcentaje de recursos candidatos es inferior a X %, mientras se cumple esta última condición, en bucle se incrementa en 3 el umbral de RSRP (i.e., RSRPthreshold), se resetea todos los recursos de la ventana de selección como libres, se vuelve a aplicar el proceso de descarte de recursos (funciones nr_v2x_halfduplex y nr_v2x_discard_reserved_resources) con el nuevo valor del umbral de RSRP, al final del bucle se ejecuta nr_v2x_remaining_available_resources para actualizar la variable occupiedCounter y la lista listCSR. Después de obtener los suficientes recursos candidatos y de salir del bucle, el valor último del umbral de RSRP se guarda en la variable last_RSRPthreshold, esta variable se usará en el proceso de reevaluación de recursos de la lista retransmisiones. Por el contrario, si enough_available_resources es igual a falso, entonces se ejecuta la función calculate_maxRSRP, esta función es llamada únicamente si inicialmente, tras aplicar solo la operación half-dúplex en el descarte de recursos, la cantidad de recursos restantes disponibles en la ventana de selección es inferior a X %. La función calculate_maxRSRP permite calcular el umbral de RSRP que deja por debajo a la RSRP de todas las reservas de otros vehículos que coincidan con los recursos restantes disponibles en la ventana de selección.

5.2.4. PROCESO PARA LA SELECCIÓN DE RECURSOS

Antes de comenzar con la selección aleatoria de recursos de la lista `listCSR`, se recorrerá dicha lista para borrar aquellos elementos cuyo *subframe* (o ranura) coincide con la ranura de transmisión de algún recurso de la lista retransmisiones. Para ello, dos índices recorren cada posición de las listas `listCSR` y `retransmissions`. Es incrementado en uno el índice cuyo el *subframe* del recurso es menor al del *subframe* de la otra lista en cada iteración del bucle. Y si los subframes de las dos listas coinciden, se borra el recurso de la lista `listCSR`.

En la selección aleatoria de recursos, primero se comprueba que la lista de recursos candidatos no esté vacía. Si la lista retransmisiones está vacía, se ejecuta la función `randomSelection` para seleccionar aleatoriamente de la lista `listCSR` el primer recurso para añadirlo a la lista retransmisiones. Este primer recurso se usará como referencia inicial para seleccionar aleatoriamente el segundo recurso, de forma que la diferencia en ranuras no sea superior a 31 ranuras. Si las retransmisiones están activadas (i.e., `retransmissionActivated == true`) y el número de recursos de la lista retransmisiones no llega a N_{MAX} (i.e., `retransmissions->size < NMAX`), se recorrerá cada recurso de la lista `listCSR` y aquellos cuyo subframe cumple con las restricciones del SCI (diferencia máxima de 31 ranuras) con al menos un recurso de la lista retransmisiones, se añadirán a una segunda lista (i.e., `listRetransmission_prev listRandom`) y después se quita ese recurso de la lista original (i.e., `listCSR`). Tras la finalización del proceso anterior, si la lista `listRandom` no está vacía (i.e., `listRandom.size != 0`), se ejecuta la función `randomSelection` para seleccionar aleatoriamente un recurso de la lista `listRandom` y añadirlo a la lista retransmisiones. Cada vez que se añade un recurso procedente de la lista `listRandom` a la lista retransmisiones, se vuelve a repetir el mismo proceso desde que se comprueba si el número de recursos en la lista retransmisiones es suficiente (i.e., `retransmissions->size < NMAX`), se busca de nuevo recursos de la lista `listCSR` que cumplan con las restricciones del SCI con la lista retransmisiones actualizada, se actualiza la lista `listRandom` y si ésta no está vacía, se vuelve a seleccionar otro recurso aleatoriamente. Este proceso se repite mientras el tamaño de la lista retransmisiones no sea igual a `NMAX` y aún quede algún recurso en la lista `listRandom` después de recorrer la lista `listCSR`. Si al salir del proceso anterior, los recursos incluidos en la lista retransmisiones siguen siendo insuficientes, se seleccionará aleatoriamente los recursos restantes de la lista

`listCSR`, mientras quede algún recurso en dicha lista. Al final de la función `nr_v2x_selector` se actualiza la información de la siguiente transmisión en los vectores `subframe` y `subchannel` de la estructura `reservation` con un máximo de N_{SCI} recursos consecutivos de la lista retransmisiones, a partir de la posición `retransmissionX`, teniendo en cuenta que en la estructura `reservation`, entre la primera ranura del vector `subframe` y cualquiera de las siguientes no puede haber una diferencia mayor a 31 ranuras.

5.2.5. OPERACIÓN HALF-DÚPLEX

El modo de transmisión usado en los vehículos es half-dúplex, por lo tanto, un vehículo que está transmitiendo no puede a su vez sensar los paquetes transmitidos por otros vehículos en la misma ranura de tiempo. En la operación half-dúplex se descarta los recursos de las posibles reservas de otros vehículos anunciadas en las mismas ranuras de transmisiones previas del vehículo que está seleccionando recursos para la transmisión del paquete actual. Se ha implementado dos funciones principales para aplicar el algoritmo half-dúplex: `calculate_vectorQ` y `nr_v2x_halfduplex`. En el proceso half-dúplex se recorre cada recurso de la lista `retransmissionsPrevious` y para cada recurso son ejecutadas las dos funciones anteriores.

Para calcular el índice *tIndex_si* correspondiente a las ranuras de tiempo en la ventana de selección de las posibles reservas, se usará la expresión $s_i + q * RRI_i$, donde s_i representa el *subframe* o la ranura de cualquier transmisión previa registrada en la lista `retransmissionsPrevious`, RRI_i representa todos los posibles valores de RRI en ranuras basados en la lista de RRI permitidos en el grupo de recursos (i.e., `int vRRI_NR[] = {0, 5, 10, 20, 25, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000}` ms) y q es un número entero que toma valores desde `qMin` hasta Q (ambos incluidos). Q indica el número estimado de transmisiones periódicas de otros vehículos para cada potencial RRI.

Primero la función `calculate_vectorQ` calcula el valor de Q de todos los RRI_i para una ranura de transmisión de la lista `retransmissionsPrevious`. En el código estos valores de Q son guardados en el vector de enteros `Q` (i.e., `int Q[16]`), que posteriormente es pasado como parámetro a la función `nr_v2x_halfduplex`. Para calcular los valores de Q, la función `calculate_vectorQ` recorre cada RRI_i de la lista y se ejecuta la función `calculateQ`, el valor devuelto por `calculateQ` es guardado en el vector `Q` en la posición

`i` (i.e., `Q[i]`, para `i` = [0, 15]). La función `calculateQ` devuelve el valor de *Q* calculado en función del RRI y de la última ranura (valor relativo) de la ventana de selección (i.e., `T2`):

- Para un RRI_i igual a 0, devuelve 0.
- Si el RRI_i es menor a `T2` en ms (i.e., `RRIms < T2ms`) y la distancia o diferencia en ranuras entre el instante actual y la ranura de transmisión del recurso de la lista `retransmissionsPrevious` es menor o igual a RRI_i (i.e., `n_slots - si <= RRI_slots`, donde `n_slots = tActual` y `RRI_slots` representa el RRI_i en ranuras), devuelve `T2 / RRI_slots`.
- En otro caso devuelve 1.

La función `nr_v2x_halfduplex` calcula el índice *tIndex_si* de todas las ranuras resultantes de la expresión $s_i + q * RRI_i$, todos los recursos o subcanales de cada ranura calculada son marcados como ocupados en la ventana de selección. Primero se recorre cada RRI_i del vector `vRRI_NR` y para cada RRI_i (pasado a ranuras, véase Tabla 2), en otro bucle interno la variable `q` toma valores entre `qMin` y `Q[i]` (ambos incluidos). La variable `qMin` es el valor mínimo de `q` para que el índice *tIndex_si* obtenido alcance como mínimo la primera ranura de la ventana de selección (i.e., *tIndex_si* `>= 0`). La función `calculate_qMin` es la encargada de calcular `qMin` en función de s_i y RRI_i:

- Devuelve 1 si el RRI_i es 0.
- Si el índice *tIndex_si* es menor al índice de la primera ranura de la ventana de selección (i.e., *tIndex_si* `< 0`), se divide la diferencia entre esas dos ranuras por el RRI_i (en ranuras) y es devuelto este valor (i.e., `(0 - tIndex_si) / RRI_slots`) redondeado hacia arriba.
- En otro caso devuelve 1.

Dentro del bucle interno se calcula el índice *tIndex_si*, este índice es pasado como parámetro a las funciones `discard_resources` y `discard_overlapping_resources`.

La función `discard_resources` marca como ocupados los recursos en un rango de subcanales consecutivos correspondiente al índice *tIndex_si* en la ventana de selección. Y la función `discard_overlapping_resources` marca como ocupadas las ranuras s_j de la ventana de selección, cuyas reservas consecutivas a partir de esas ranuras con el RRI del vehículo que está seleccionando recursos coincidan con cualquier ranura de la expresión anterior, es decir, se descarta un rango de subcanales consecutivos de las

ranuras s_j que cumplan con la igualdad $s_j + j * RRI_{TX} = s_i + q * RRI_i$. El número *j* puede tomar valores entre 1 y (`10 * reservation->RC - 1`) y RRI_{TX} es el RRI (en ranuras, i.e., `reservation->RRI`) seleccionado por el vehículo que está seleccionando recursos para transmitir el paquete actual.

Para obtener las ranuras s_j que se descartan en la ventana de selección, se despeja s_j de la ecuación o igualdad entre las dos expresiones: $s_j = s_i + q * RRI_i - j * RRI_{TX}$. El índice *tIndex_si* se obtiene restando (`tActual + T1`) a las ranuras (véase correspondencia entre posiciones y ranuras en la Figura 11). La función `discard_overlapping_resources` recibe el índice *tIndex_si*, si éste es mayor que 0 (índice de la primera ranura de la ventana de selección), se ejecuta la función `calculate_jMax_jMin` para calcular los valores mínimo y máximo (i.e., `jMin` y `jMax`) de la variable `j`. La variable `jMin` representa el valor mínimo de `j` para que el índice *tIndex_sj* de la ranura s_j sea menor o igual al índice de la última ranura de la ventana de selección (i.e., *tIndex_sj* `<= T2 - T1`) y la variable `jMax` representa el valor máximo de `j` del cual el índice *tIndex_sj* sea mayor o igual al índice de la primera ranura de la ventana de selección (i.e., *tIndex_sj* `>= 0`). Si el valor de `jMax` supera al valor límite de (`10 * reservation->RC - 1`), `jMax` es actualizado a dicho valor límite. Si el RRI_{TX} es 0, la función `calculate_jMax_jMin` asigna los valores 0 y 1 a las variables `jMax` y `jMin`, respectivamente. En otro caso, el valor de `jMax` es el cociente de la división entera de *tIndex_si* entre RRI_{TX} y es actualizado al valor límite si lo supera. Si el índice *tIndex_si* es mayor a `T2 - T1`, es decir, si el índice es posterior en el tiempo a la ventana de selección, se divide la diferencia entre esos dos índices por el RRI_{TX} y este valor (i.e., `(tIndex_si - (T2 - T1)) / RRI_TX`) redondeado hacia arriba es asignado a la variable `jMin`, por el contrario, si *tIndex_si* es menor o igual a `T2 - T1`, la variable `jMin` toma el valor 1.

En half-dúplex las funciones `discard_resources` y `discard_overlapping_resources` reciben los valores `0` y `Nsubchannel` en los parámetros `firstSubchannel` y `lastSubchannel`, respectivamente. El parámetro `firstSubchannel` representa el primer subcanal de las ranuras a descartar en la ventana de selección y (`lastSubchannel - 1`) representa el último subcanal a descartar. Es decir, en la operación half-dúplex se descartan los recursos de todos los subcanales de las ranuras $s_i + q * RRI_i$ y s_j en la ventana de selección (i.e., `selectionWindow[k][tIndex_si_sj].occupied = true`, para `k = [0, lastSubchannel - 1]` y *tIndex_si_sj* representa el índice *tIndex_si* o el índice *tIndex_sj*).

5.2.6. DESCARTE DE RECURSOS RESERVADOS EN PAQUETES RECIBIDOS

En el proceso de descarte de recursos reservados por otros vehículos, se reutiliza las funciones usadas en la operación half-dúplex. Este proceso tiene bastantes similitudes con la operación half-dúplex y también ciertas diferencias. Para llevar a cabo el proceso, se ejecuta la función `nr_v2x_discard_reserved_resources`, esta función recorre cada paquete recibido en la ventana de detección hasta la posición `posTproc0`. El índice `posTproc0` es la posición del último paquete de la ventana de detección cuya ranura de transmisión (i.e., `sensingWindow->list[posTproc0].subframe[0]`) es menor o igual al instante `tActual - Tproc0[mu]`, donde `Tproc0` es un vector que contiene los valores {1, 1, 2, 4} ranuras para un factor de configuración de `mu` igual a {0, 1, 2, 3}. Los valores de `Tproc0` representan el tiempo necesario para completar el procesamiento de un paquete recientemente recibido. Para cada paquete primero se comprueba si el nivel de RSRP medido es superior al umbral de RSRP (i.e., `sensingWindow->list[res].RSRP > RSRPthreshold`, donde `res = [0, posTproc0]`), si no se cumple esta condición, se ignora el paquete y se pasa al siguiente paquete de la lista en la ventana de detección. Por el contrario, si el nivel de RSRP es superior al umbral, se procederá a descartar los recursos de las reservas realizadas en el paquete recibido.

De forma análoga a la estructura `reservation` (véase último párrafo de la sección 5.2.1), cada paquete recibido también consta del vector `subframe`, con la ranura de transmisión y las ranuras de las retransmisiones, el vector `subchannel` y la variable `sizeSub`, que indica el tamaño de estos. Por cada ranura de transmisión del vector subframe hay tres fases de descarte de recursos, las mismas que en la operación half-dúplex (con algunas diferencias) y una adicional. Una diferencia con la operación half-dúplex es el rango de subcanales de los recursos a descartar en la ventana de selección. A diferencia de la operación half-dúplex, que descartaba todos los subcanales, en el descarte de recursos reservados los parámetros `firstSubchannel` y `lastSubchannel` reciben los valores `fIndex` y (`fIndex + length`), es decir, se descarta únicamente los subcanales reservados en el paquete recibido. La variable `fIndex` representa el primer subcanal a descartar y el valor (`fIndex + length - 1`) representa el último subcanal del rango. La variable `length` es el tamaño del paquete recibido y la variable `fIndex` toma su valor del vector `subchannel` en la misma posición que la ranura de transmisión asociada en el vector `subframe`.

En la primera fase se ejecuta la función `discard_resources` para descartar una de las ranuras s_k del vector `subframe` del paquete recibido[2]. Al igual que en la operación half-dúplex, las ranuras son pasadas al índice correspondiente en la ventana de selección. En la segunda fase se ejecuta también la función `discard_resources` para descartar recursos de las ranuras de la expresión $s_k + q * RRI_{RX}$, donde s_k sustituye a s_i y RRI_{RX} a RRI_i en la expresión $s_i + q * RRI_i$, s_k hace referencia a cualquier ranura del vector subframe, RRI_{RX} es el RRI asociado al paquete recibido[3] y los valores mínimo y máximo (i.e., `qMin` y `Q`) de q se calculan de la misma manera que en la operación half-dúplex, teniendo en cuenta las dos sustituciones en la expresión. En la tercera fase se descarta cualquier ranura s_p en la ventana de selección que cumpla con la igualdad $s_p + j * RRI_{TX} = s_k + q * RRI_{RX}$, de forma análoga a la operación half-dúplex se ejecuta la función `discard_overlapping_resources`, despejando s_p.

5.2.7. CALCULAR EL UMBRAL DE RSRP DE LOS RECURSOS LIBRES

En la sección 5.2.3 se ha expuesto la condición para ejecutarse la función `calculate_maxRSRP`. En esta sección se hará una comparación del algoritmo empleado en la función `calculate_maxRSRP` con la función `nr_v2x_discard_reserved_resources`. Las dos funciones recorren cada paquete en la ventana de detección hasta la posición `posTproc0` de la lista de paquetes recibidos. Y por cada paquete se obtienen las ranuras s_k, $s_k + q * RRI_{RX}$ y s_p. La única diferencia es que la función `nr_v2x_discard_reserved_resources` tiene el objetivo de marcar como ocupados los recursos reservados por otros vehículos cuyo nivel de RSRP medido del paquete recibido es superior al umbral de RSRP pasado como parámetro, mientras que la función `calculate_maxRSRP` calcula el umbral de RSRP para los recursos disponibles en la ventana de selección. Hay que recordar que la función `calculate_maxRSRP` se ejecuta únicamente si después de ejecutarse la función `nr_v2x_halfduplex` el porcentaje de recursos libres en la ventana de selección es inferior a X (20, 35 o 50) %.

En la función `calculate_maxRSRP` primero se inicializa la variable `maxRSRP` a un valor inicial de RSRP, este valor se corresponde al parámetro `RSRPthresholdMain`, que es pasado en sentencia de ejecución a la función *main* (Tabla 3). Por cada paquete, si el nivel

[2] En la primera fase no se aplica a la ranura de transmisión (i.e., sensingWindow->list[res].subframe[0]) del paquete, ya que esa ranura es anterior al instante actual, por lo tanto, se sitúa fuera de la ventana de selección.
[3] A diferencia del RRI_i, que representa la lista de valores permitidos, el RRI_{RX} es un único valor por paquete.

de RSRP medido del paquete es mayor que la variable `maxRSRP`, se comprobará si alguna de las ranuras s_k, $s_k + q * RRI_{RX}$ o s_p coincide con algún recurso libre de la ventana de selección (i.e., `selectionWindow[0][tIndex_sk].occupied == false`, el segundo índice puede ser `tIndex_sk`, `tIndex_selWin` o `tIndex_sp` según a cuál de las tres ranuras anteriores pertenece), que, en caso de cumplirse la condición, se actualizará la variable `maxRSRP` con la RSRP del paquete y se pasará al paquete siguiente. Si la diferencia entre `maxRSRP` y `RSRPthresholdMain` no es divisible entre 3, se aumentará `maxRSRP` al valor más cercano para cumplirse esa condición, tal como el valor del umbral de RSRP, incrementado en 3 a partir del valor inicial de `RSRPthresholdMain`, cada vez que el porcentaje de recursos libres en la ventana de selección es inferior a X (20, 35 o 50) % después de ejecutarse las funciones `nr_v2x_halfduplex` y `nr_v2x_discard_reserved_resources` (último párrafo de la sección 5.2.3). Al final la función `calculate_maxRSRP` devolverá el valor de la variable `maxRSRP` como el umbral de RSRP para la siguiente reevaluación de recursos. El valor último del umbral de RSRP, tanto si el umbral fue incrementado en 3 como si el valor del umbral es devuelto por la función `calculate_maxRSRP`, es guardado en la variable `last_RSRPthreshold`, que es un parámetro pasado por referencia.

5.2.8. MECANISMO DE REEVALUACIÓN

El mecanismo de reevaluación de recursos se produce si el instante actual coincide con el instante de reevaluación. En tal caso, en la función `nr_v2x_scheduler` se crea la ventana de selección, que es pasado como parámetro a la función `nr_v2x_selector`, el parámetro `reev` de esta función recibe el valor `true` para indicar que hay reevaluación de recursos. En las secciones anteriores a partir de la sección 5.2.3 se ha explicado con detalles el algoritmo de la función `nr_v2x_selector`. En esta sección se va a centrar en las diferencias en el flujo de ejecución de una reevaluación de recursos en comparación con una petición de nuevos recursos.

Después de ejecutarse la función `nr_v2x_halfduplex` por primera vez, si el porcentaje de recursos es suficiente (mayor a X %), se ejecuta la función `nr_v2x_discard_reserved_resources`, para descartar los recursos reservados por otros vehículos en función de un umbral de RSRP. La primera diferencia es el valor del umbral de RSRP, que en una petición de nuevos recursos se inicializa con el valor mínimo de `RSRPthresholdMain`, en cambio, en una reevaluación de recursos, toma el valor de la

variable `last_RSRPthreshold`, que es el umbral de RSRP de la ejecución anterior de la función `nr_v2x_selector`. En finalizar la ejecución de las funciones `nr_v2x_halfduplex` y `nr_v2x_discard_reserved_resources`, si es una reevaluación de recursos, se comprueba si el porcentaje de recursos libres es suficiente tras aplicar half-dúplex (i.e., `enough_available_resources == true`), si no es así, se ejecuta la función `calculate_maxRSRP` para calcular el umbral de RSRP para una posible siguiente reevaluación de recursos y se sale de la función `nr_v2x_selector`. Por el contrario, si había suficientes recursos libres, se reevaluará todos los recursos de la lista retransmisiones a partir de la posición `reevaluationX`, que tengan la reevaluación activada (véase las secciones 5.2.1 y 5.2.2 para más detalles). La ventana de selección es una matriz de `Nsubchannel` filas y (`T2 - T1 + 1`) columnas, en el proceso de reevaluación de un recurso, se calculan los índices del primer subcanal y la ranura de transmisión del recurso y la variable `reservation->length` indica cuántos subcanales contiguos se usarán para la transmisión. Un recurso es reevaluado como disponible si todos esos subcanales contiguos de la ranura de transmisión están marcados como disponibles en la ventana de selección (i.e., `selectionWindow[fIndex + k][selectedSlotIndex].occupied == false`, `fIndex` es el índice del primer subcanal, `selectedSlotIndex` es el índice de la ranura de transmisión y `k = [0, reservation->length - 1]`). Si todos los recursos reevaluados siguen disponibles, se sale de la función `nr_v2x_selector`. En otro caso, se borra cada recurso ocupado de la lista retransmisiones y al igual que en una petición de nuevos recursos se ejecutan las funciones `nr_v2x_halfduplex` y `nr_v2x_discard_reserved_resources` para descartar recursos en la ventana de selección a partir de un umbral inicial de RSRP con el valor de `RSRPthresholdMain`, que es incrementado en 3 cada vez que el porcentaje de recursos libres es insuficiente. Y el paso 2, selección de recursos, es exactamente igual que en una petición de nuevos recursos.

Al final de la función `nr_v2x_scheduler`, tanto si es una petición de recursos para la transmisión de un nuevo paquete como si es una reevaluación de recursos, si el instante de reevaluación calculado es posterior al instante actual (i.e., `new_tReevaluation > tActual`), se calcula la porción restante de la ventana de selección (i.e., T_2') en función del nuevo instante de reevaluación, y el valor calculado se guarda en la variable `PDB` de la estructura `reservation` (i.e., `reservation->PDB = tActual + T2Aux - new_tReevaluation`, `T2Aux` representa el valor de T_2 o T_2' usado en la última creación

de la ventana de selección y `reservation->PDB` guarda el nuevo valor de T_2' para la siguiente reevaluación). Si el valor de `reservation->PDB` es inferior a T_1 o a T_{2min}, el instante de reevaluación y la posición del siguiente recurso a reevaluar son actualizados a 0 y `reservation->PDB` toma el valor total de T_2. Si el instante de reevaluación no era posterior al instante actual, es decir, si no hay reevaluación, se actualiza `reservation->PDB` con el valor total de T_2.

5.2.9. PROBLEMAS Y DIFICULTADES DURANTE LA IMPLEMENTACIÓN

Las funcionalidades del estándar 5G NR V2X son para complementar y no sustituir a LTE V2X. En la medida de lo posible se ha intentado que todo el nuevo código sea retrocompatible con el estándar previo. Durante el proceso de implementación de las funcionalidades del estándar 5G NR V2X, se ha detectado diversos problemas en el código arrastrados de la implementación de las funcionalidades del estándar LTE V2X. Los problemas encontrados han generado muchas dificultades en el progreso o avance en la implementación de las nuevas funcionalidades y también en la depuración y en las simulaciones realizadas. Estos problemas tienen como consecuencias principales la violación de memoria y la baja automatización en la ejecución del código, entre otras.

El objetivo de esta sección es exponer las principales dificultades encontradas y también la resolución de éstas. Estos problemas han supuesto un aumento muy considerable en la complejidad de desarrollo que se estimó en las fases iniciales del proyecto, lo que implicó tiempo y esfuerzo adicional para solucionarlos, antes de poder seguir programando los nuevos detalles y funcionalidades del estándar 5G NR V2X y de lanzar las simulaciones correctamente. A continuación, se enumerarán los puntos principales de los problemas junto a una descripción o ejemplo de cada uno:

- Funcionalidades incompletas: este punto hace especial referencia a las funcionalidades del estándar 5G NR V2X que comparten algunas propiedades comunes con las especificaciones del estándar LTE V2X. En las fases iniciales del proyecto se ha informado de que las funcionalidades como la operación half-dúplex y el descarte de recursos reservados por otros vehículos estaban ya implementadas en el código LTE V2X, pero tras un análisis profundo del código se ha podido observar que hay una carencia casi total en el nivel de detalles. Como recordatorio, en la operación half-dúplex se descarta los recursos de la ventana de selección para las ranuras $s_i + q * RRI_i$ y las ranuras s_j que cumplan la igualdad s_j

$+ j * RRI_{TX} = s_i + q * RRI_i$ (ver sección 5.2.5 para más detalles), sin embargo, en el código de LTE V2X no se aprecia el descarte de las ranuras s_j y en lo relativo a la primera expresión se ha podido apreciar que solo se descarta la ranura $s_i + 100$[4]. Por otro lado, en el descarte de recursos reservados por otros vehículos, se descartan las ranuras $s_k + q * RRI_{RX}$ y las ranuras s_p que cumplan la igualdad $s_p + j * RRI_{TX} = s_k + q * RRI_{RX}$ (ver sección 5.2.5 para más detalles), al igual que en la operación half-dúplex, tampoco se aprecia en el código de LTE V2X el descarte de las ranuras s_p y en lo relativo a la primera expresión solo se descarta la ranura $s_k + RRI_{RX}$.

- Funcionalidad inexacta: la actualización del valor del contador de reselección (RC), así como la gestión de la probabilidad $(1 – P)$ de seleccionar nuevos recursos son incorrectas en la implementación del código LTE V2X. Según las especificaciones [7], el valor del RC es decrementado en uno después de la transmisión de un paquete, es decir, antes de una nueva petición de recursos para transmitir el nuevo paquete. Sin embargo, en el código de LTE V2X, el valor de RC no llega decrementado (actualizado) para la nueva petición, sino que es decrementado su valor justo después de mantener el mismo recurso, con lo cual el RC no es decrementado después del primer uso (transmisión) del recurso seleccionado para la transmisión, sino que, a partir del segundo uso, el RC es decrementado previamente en la iteración anterior. Por otra parte, la probabilidad $(1 – P)$ de seleccionar nuevos recursos si el RC es 0, debe determinarse cuando el RC es 1, porque el valor del RRI para la transmisión debe conocerse para indicar si se reservará el mismo recurso para nueva transmisión (RRI > 0) o no (RRI = 0), es decir, se genera el número aleatorio para la probabilidad cuando el RC es 1 y para cuando el RC es 0, el valor del RRI usado en la última transmisión determinará si hay selección de nuevos recursos [7]. Si el RC es 0 y el RRI es mayor a 0, es decir, si se mantiene el mismo recurso, se generará un nuevo valor aleatorio para RC. Sin embargo, en el código de LTE V2X, la probabilidad de seleccionar nuevos recursos es determinado cuando el RC ya ha llegado a cero, pudiendo derivarse en dos situaciones incongruentes. La primera es mantener el mismo recurso para transmitir, cuando se trata de la primera petición de nuevos

[4] Se deduce que el valor de 100 hace referencia al RRI para una simulación con una tasa media (i.e., `MsgRate`) de 10 paquetes por segundo.

recursos para la transmisión del primer paquete, es decir, se mantiene el mismo recurso, cuando no ha habido selección previa de recurso para transmitir. La segunda es la probabilidad de seleccionar nuevos recursos (para RC = 0), cuando en la última transmisión (para RC = 1) ya se reservó el mismo recurso para la transmisión del nuevo paquete (RRI > 0). También se observa en el código, que de forma errónea se asigna el valor 0 al RRI, si después de mantener el mismo recurso, el valor del RC decrementado es 0.

- Uso ineficiente de la memoria: en la reserva de memoria dinámica se reserva mucha más memoria de lo necesario, como ejemplo, en la creación de la ventana de selección se requiere reservar memoria para ($T_2 - T_1 + 1$) ranuras, en cambio, en el código LTE V2X se reserva siempre T_2 ranuras (ver Figura 11). Otro problema agregado, mucho más grave, es utilizar una única estructura para 3 cosas distintas, como son la ventana de detección o sensado (*sensing window*), la ventana de selección (*selection window*) y la lista de recursos candidatos, sin que éstas compartan apenas variables en común, desperdiciando una cantidad elevada de memoria.
- Poca abstracción del código: hay valores numéricos que son introducidos directamente a lo largo del programa, en lugar de usar una variable que contenga el valor, esto impide la escalabilidad del programa, debido a que esos valores son diferentes en el estándar 5G NR V2X o para una configuración diferente de simulación.
- Ausencia de control de reserva correcta de memoria: en todo el programa hay una ausencia total en la comprobación posterior de la reserva, ampliación o reducción exitosa de la memoria dinámica (a excepción de la matriz de valores RSSI). Esta ausencia junto a un uso ineficiente de la memoria causa constantes cuelgues en las simulaciones debido a las violaciones de memoria o *segmentation fault*.
- Imprecisión en el tamaño de los vectores: el tamaño de los vectores que registran la tasa de paquetes recibidos correctamente no concuerda con la distancia máxima en número de vehículos configurada para registrar la transmisión-recepción de paquetes entre dos vehículos. Se ha observado que únicamente se hace un registro cuando el vehículo receptor es uno de los 5 vehículos centrales de la carretera, pero solo hay garantía de que el registro no se salga de las posiciones de los

vectores si el vehículo receptor es el vehículo central, pudiendo salirse hasta dos posiciones fuera de los vectores.

- Inicialización de variables: la variable instante actual (i.e., `tActual`) es inicializada con el valor 1000, por lo tanto, el tiempo de simulación pasado por parámetro no se corresponde con el tiempo simulado. Al actualizarlo con el valor 0, surge otro error en la inicialización de la variable `subframe` o ranura de transmisión de la estructura `reservation`, que al coincidir con el valor inicial de `tActual`, se produce una transmisión de paquete sin haber seleccionado previamente un recurso para transmitir. Las variables `RRI` y `RC` de la estructura `reservation` no están inicializadas con los valores correctos de 0 y 1 (ver primera situación del primer caso de la sección 5.2.1).

En los puntos anteriores se ha expuesto los principales problemas detectados en la implementación del estándar LTE V2X, en los siguientes párrafos se expondrán las resoluciones:

- Para abordar las dos funcionalidades anteriores, se ha visto en la necesidad de implementar 8 funciones: `nr_v2x_halfduplex`, `nr_v2x_discard_reserved_resources`, `calculate_vectorQ`, `calculate_qMin`, `calculateQ`, `discard_resources`, `calculate_jMax_jMin`, y `discard_overlapping_resources`. Las dos primeras funciones hacen referencia a las funciones principales de las funcionalidades half-dúplex y descarte de recursos reservados, respectivamente. Estas dos funcionalidades comparten la ejecución de las 5 últimas funciones de la lista anterior. Y la función `calculate_vectorQ` se ejecuta solo en la operación half-dúplex. Las funciones `calculate_qMin` y `calculateQ` calculan los valores mínimo y máximo para *q*, respectivamente, en la primera expresión de las dos funcionalidades y la función `discard_resources` descarta los recursos respectivos en la ventana de selección. La función `calculate_jMax_jMin` calcula los valores mínimo y máximo de *j* en la igualdad de la segunda expresión y la función `discard_overlapping_resources` descarta los recursos para las ranuras s_j o s_p. Como se puede observar, hay una clara división del código en funciones específicas, lo que permite aprovechar al máximo la reutilización de éstas y un mayor nivel de abstracción o modularidad de los detalles. Gracias a las funciones reutilizables, en futuras revisiones del código LTE V2X, se podrá subsanar de

forma casi inmediata las carencias en los detalles de la implementación, sustituyendo una línea de código en cada caso por las funciones mencionadas para descartar los recursos acordes a las especificaciones del estándar LTE V2X.

- En la Figura 9 y la Figura 10 se muestran el flujo correcto de actualización del RC, así como la correcta gestión de la probabilidad (1 - *P*) de seleccionar nuevos recursos. Este flujo será extrapolable para futuras revisiones del código LTE V2X.
- Para hacer un uso más eficiente de la memoria, además de reservar el número de posiciones estrictamente necesario, se ha definido nuevas estructuras diferenciadas, específicas para cada uso (ver Tabla 4 y Tabla 1), esto ha hecho necesario también otras modificaciones en el código común con LTE V2X, para poder compilar y ejecutar el programa.
- Los valores que pueden cambiar son sustituidos por variables.
- Se ha incluido comprobaciones de reserva exitosa de la memoria dinámica, así como su gestión en caso de fallar la reserva de memoria. Cada función que reserva o redimensiona memoria devolverá un código de error si esa operación falla, la función superior que recibe el código de error también devolverá un código de error, así hasta llegar a la función *main*. En la función *main* se liberará toda la memoria dinámica reservada hasta el momento y después devuelve el valor -1 para indicar que ha fallado la simulación. Dentro de cada función también se libera memoria reservada, si ésta es reservada temporalmente dentro de la función, inaccesible desde funciones superiores.
- Se ha ajustado el tamaño de los vectores que registran la tasa de paquetes recibidos correctamente. Ahora los vectores también registran la tasa de paquetes transmitidos entre vehículos que están a una distancia de hasta 1 km, sin la necesidad de que el vehículo receptor sea uno de los vehículos centrales, para ello se obtiene la distancia en número de vehículos entre el emisor y el receptor en valor absoluto y se verifica si este valor se sitúa dentro de las dimensiones de los vectores.
- Se ha realizado un análisis en profundidad de la inicialización de todas las variables del programa.

5.3. INTEGRACIÓN EN CÓDIGO LOCAL

La implementación realizada de la capa MAC de 5G NR V2X se ha integrado en el denominado *código local* para su depuración y validación antes de la integración en OAI.

El *código local* es capaz de simular un escenario de autovía, con una densidad de tráfico preconfigurada, en el que todos los vehículos emplean el *scheduler* de la capa MAC implementada. En la simulación, se lleva a cabo un control del tiempo y del instante en el que cada vehículo ha de realizar una llamada al *scheduler*, ha de transmitir o recibir un mensaje. En el diagrama de flujo de la Figura 13 se representa el flujo de ejecución de alto nivel de cada vehículo de las simulaciones. Se puede observar en la figura que la ejecución comienza con la declaración e inicialización de variables, el instante actual (i.e., `tActual`), representado en ranuras, es inicializado a cero y siempre se incrementa en uno al final del bucle `while` mientras que `tActual` sea menor que el tiempo total de simulación (i.e., `tActual` < `durationTime`). El valor de `durationTime` es pasado por parámetro en la función *main* en ms y posteriormente es pasado a ranuras. La conversión de ms a ranuras se realiza multiplicando por el número de ranuras/ms (Tabla 2) en función del valor del factor de configuración μ (i.e., `mu`). Los 4 valores de ranuras/ms son almacenados en el vector `Nslot` y la variable `mu` contiene el valor del índice para el acceso a la posición correspondiente del vector `Nslot` (i.e., Ranuras = ms x `Nslot[mu]`).

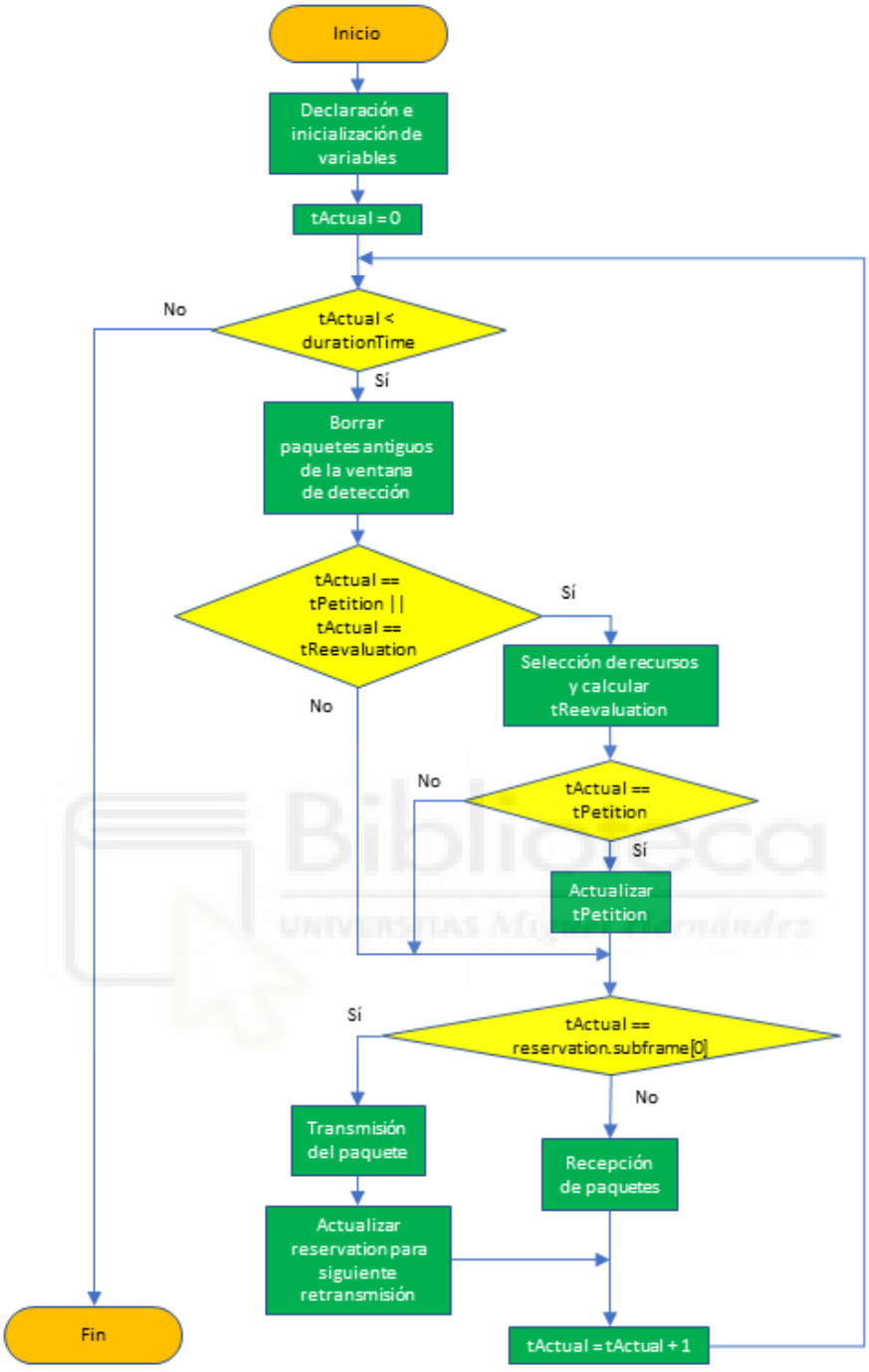

Figura 13. Flujo de ejecución de un vehículo

μ	Ranuras por ms (2^{μ})
0	1
1	2
2	4
3	8

Tabla 2. Relación ranuras por milisegundo

Dentro del bucle `while` el vehículo primero comprueba los paquetes de la ventana de detección (i.e., `listResource sensingWindow`) y borra de la lista aquellos paquetes cuyos todos los valores del vector `subframe` (ranura de transmisión del paquete y las posibles reservas), representado en ranuras, sean menores a `tActual - senWin_tam`. En 5G NR V2X la variable `senWin_tam` (en ranuras) puede tomar dos valores: 1100 ms o 100 ms. Estos dos valores son almacenados en el vector `T0` y el valor elegido es el del índice `T0Idx`, que es pasado por parámetro a la función *main*. El valor elegido de T_0 es la diferencia en ms entre `tActual` y la primera ranura de la ventana de detección, que es transformado en ranuras al asignar el valor a la variable `senWin_tam`.

Parámetro	Significado	Valores	Unidad
Density	Densidad de vehículos	60, 120	vehículos/km
MsgRate	Número medio de paquetes “nuevos” para transmitir por segundo	10, 20, 50	Hz
TxPower	Potencia de transmisión	23	dBm
Nsubchannel	Número total de subcanales por ranura	4	-
PktSize	Tamaño del paquete	190	bytes
durationTime	Tiempo total de simulación	3200 ms	ranuras
RSRPthresholdMain	Umbral de la potencia recibida de la señal de referencia	-128	dB
T1	Tiempo de procesamiento requerido para seleccionar nuevos recursos	2	ranuras
version	Estándar LTE o 5G NR V2X	0, 1	-
mu	Factor de configuración en función del SCS	1	-
T0Idx	Índice del valor seleccionado	0	-
v2x_priority	Prioridad del paquete	2	-
P	1 – P es la probabilidad para seleccionar nuevos recursos cuando RC = 0	0.0	-
reevaluation	Activar / desactivar reevaluación	0, 1	-
traffic_type	Tipo de tráfico	0, 1, 2	-
retransmissionActivated	Activar / desactivar retransmisiones	0, 1	-

NMAX	Número máximo de transmisiones del mismo paquete	1-32	-

Tabla 3. Parámetros de entrada de la función main

La estructura interna de `sensingWindow` (Tabla 4) tiene 2 variables, `list` es el vector con los paquetes recibidos correctamente y `size` guarda el tamaño del vector `list`. La lista de paquetes recibidos está ordenada en función de `subframe[0]` (ranura de transmisión) y en segundo lugar de `subchannel[0]` (primer subcanal usado en la transmisión).

Nombre de la estructura	Datos en la estructura
reservation_t	int sizeSub; int RRI; int RC; int length; int subframe[NTRANSMISIONES]; int subchannel[NTRANSMISIONES]; int priority; int PDB;
resource_t	int sizeSub; int subframe[NTRANSMISIONES]; int RRI; double RSRP; int subchannel[NTRANSMISIONES]; int length; int priority; int txID;
listResource	resource_t *list; int size;
stRetransmission	int subframe; int subchannel;
listRetransmission	stRetransmission *list; bool *reevActiv; int size;
stSelectionWindow	bool occupied; bool included; double RSSI;
listRetransmission_prev	stRetransmission *list; int size;

Tabla 4. Las estructuras struct empleadas en el código

Con la ventana de detección ya actualizada, si el instante actual coincide con el instante de petición de nuevos recursos (i.e., `tPetition`) o el instante de reevaluación de recursos (i.e., `tReevaluation`), entonces es llamado la función `nr_v2x_scheduler`, que es la encargada principalmente de actualizar la lista de recursos seleccionados para las retransmisiones (i.e., `listRetransmission retransmissions`) y el siguiente instante

de reevaluación de recursos. Si es una petición de recursos también se actualizará la variable `tPetition` en función de si se trata de tráfico periódico o tráfico aperiódico.

Si no hay petición de recursos ni reevaluación de recursos, entonces se comprueba si es el instante de (re)transmisión del paquete (i.e., `tActual == reservation[veh].subframe[0]`, `veh` es el id del vehículo y `reservation` es de tipo `reservation_t`), si se cumple, se incluye el paquete (se transmite el paquete) en la ventana de detección de los vehículos receptores que no cumplan esa misma condición (debido a la transmisión half-dúplex) y se actualiza la estructura `reservation` (del transmisor) con los siguientes recursos de la lista de retransmisiones. En la recepción de paquetes, se calcula el nivel de RSRP (potencia recibida de la señal de referencia) del paquete recibido en función de la distancia entre el emisor y el receptor y la potencia de transmisión (`TxPower`) y se actualiza la tabla de RSSI (indicador de intensidad de señal recibida) del vehículo receptor en la ranura actual, sumándole el valor de RSRP calculado en la recepción. Para sumar esos dos valores, son transformados primero de dBm a W y el resultado de la suma es convertido de nuevo a dBm. Previamente a la transmisión-recepción de paquetes se actualiza la tabla de RSSI de todos los vehículos en la ranura actual asignándole un valor aleatorio. En el escenario simulado se dispone de una longitud de 2 km de carretera, en la que los vehículos están colocados estáticamente y repartidos de forma equidistante a lo largo de la carretera. La distancia entre dos vehículos consecutivos en metros se calcula dividiendo los 1000 metros entre la densidad de vehículos por km.

$$RSSI = \frac{(rand()\ \%\ 10000)}{10000.0} + thermalNoise\_dB$$

Ecuación 1. Asignación de valor aleatorio de RSSI

La tabla de RSSI es una matriz bidimensional formada por `Nsubchannel` filas (o subcanales) y `senWin_tam` columnas (o ranuras). En la declaración e inicialización de variables se asigna inicialmente valores aleatorios de RSSI en toda la tabla de cada vehículo. La actualización de la tabla se hace de manera circular. En cada iteración del bucle `while` se calcula el (segundo) índice de la matriz RSSI correspondiente a la ranura actual, dicho cálculo se realiza haciendo módulo del instante actual entre el tamaño en ranuras de la ventana de detección en la dimensión tiempo (i.e., `tActual % senWin_tam`), el resto de la división entera se guarda en la variable `tIdx`. En cada instante de tiempo (i.e., `tActual`), para cada vehículo, se actualiza todos los subcanales de la ranura `tIdx`

(segundo índice) de su tabla RSSI con valores aleatorios y posteriormente se le suma la RSRP de las recepciones, de forma que en cada nueva iteración del bucle `while` se tiene todos los valores de RSSI de las últimas `senWin_tam` ranuras previas, siendo `tActual % senWin_tam` el índice de la primera ranura (instante `tActual - senWin_tam`) y `(tActual - 1 + senWin_tam) % senWin_tam` la última ranura (instante `tActual - 1`) de la ventana de detección.

Tras la finalización del proceso de transmisión-recepción de paquetes, se recorre la lista de paquetes recibidos de cada vehículo empezando por el final. Para los paquetes recibidos con ranura de transmisión en el instante actual se calcula si han sido recibidos correctamente. El paquete es recibido correctamente si el nivel de RSRP es mayor al umbral `Psen` y la función `pkt_received` devuelve verdadero. La función `pkt_received` recibe tres parámetros de entrada: `MCS`, `SINR` y `mu`. La variable `MCS` es el esquema de codificación y modulación, `SINR` es la relación entre RSRP y el ruido total. En función de esos tres valores se obtiene una tasa de error `BLER` y se genera un valor aleatorio `x` entre [0, 1) con 5 decimales. Si `x` es mayor que `BLER`, la función `pkt_received` devolverá verdadero, en caso contrario, devolverá falso. Finalmente, si el paquete ha sido recibido correctamente se mantendrá en la ventana de detección, pero si no, se borrará de la lista de paquetes recibidos. En la Tabla 5, la Tabla 6 y la Tabla 7 se muestra la relación entre los valores de `MCS`, `SINR` y `mu` con las tasas de errores (`BLER`) correspondientes.

SINR	BLER
< -9	1
< -8	0.9624
< -7	0.9261
< -6	0.7683
< -5	0.5995
< -4	0.4393
< -3	0.2862
< -2	0.1711
< -1	0.0903
< 0	0.0419
< 1	0.0151
< 2	0.0070
< 3	0.0029
< 4	0.0014
>= 4	0.0006

Tabla 5. Tasas de error 1, con MCS = 13 y mu = 0

SINR	BLER
< -2	1
< -1	0.97501
< 0	0.6132
< 1	0.4755
< 2	0.3781
< 3	0.2182
< 4	0.1321
< 5	0.0540
< 6	0.0230
< 7	0.0059
>= 7	0.0015

Tabla 6. Tasas de error 2, con MCS = 13 y mu = 1

SINR	BLER
< -10	1
< -9	0.8081
< -8	0.6598
< -7	0.4867
< -6	0.3243
< -5	0.2034
< -4	0.1302
< -3	0.0499
< -2	0.0189
< -1	0.0076
< 0	0.0018
< 1	0.0005
>= 1	0.0001

Tabla 7. Tasas de error 3, con MCS = 13 y mu = 2

La principal métrica de rendimiento almacenada para el análisis de resultados es la PDR (*Packet Delivery Ratio*) que se define como la probabilidad de correcta recepción de paquetes en función de la distancia entre transmisor y receptor. Para obtener esta métrica, por cada paquete recibido (correctamente o no) es incrementado en uno el vector `TotalNumPktRx` en la posición `i`, la variable `i` representa la distancia entre el emisor y el receptor en número de vehículos y se calcula obteniendo el valor absoluto de la diferencia entre el ID del receptor con el ID del transmisor (i.e., `abs(veh - txID)`[5]). En el vector `NumPktRx` también es incrementado en uno de forma análoga al vector `TotalNumPktRx` si el paquete es correctamente recibido (Tabla 8). Para almacenar los resultados se emplean también los vectores `TotalNumPktRxAplication` y `NumPktRxAplication`. Estos dos últimos también registran los paquetes recibidos y los recibidos de forma correcta, respectivamente. Los dos primeros (`TotalNumPktRx y NumPktRx`) representan

[5] En el bucle de transmisión-recepción, veh es el ID del vehículo transmisor. En el bucle de comprobación de la correcta recepción del paquete, veh es el ID del receptor y txID es el ID del transmisor.

la capa MAC y se hace un registro por cada paquete recibido. En cambio, los dos últimos representan la capa aplicación y de todas las retransmisiones recibidas del mismo paquete por el mismo receptor, se hace el registro de únicamente de una de las retransmisiones. El vector `NumPktRxAplication` hace un único registro si el mismo paquete es recibido correctamente más de una vez por el mismo receptor (Tabla 9).

Capa MAC	
Vectores	Uso (en función de la distancia)
TotalNumPktRx	Registra cada paquete recibido
NumPktRx	Registra cada paquete recibido correctamente

Tabla 8. Registro de los paquetes recibidos en capa MAC

Capa Aplicación	
Vectores	Uso (en función de la distancia)
`TotalNumPktRxAplication`	Registra cada paquete "diferente" recibido por cada receptor
`NumPktRxAplication`	Registra cada paquete "diferente" recibido correctamente por cada receptor

Tabla 9. Registro de los paquetes recibidos en capa Aplicación

El registro de los paquetes se realiza en los archivos con los prefijos 5G_PDR_Application y 5G_PDR_MAC y con extensión CSV y se almacenan los datos en el formato que se muestra en la Figura 14. Por cada escenario simulado son generados dos archivos cuyos nombres, además de los prefijos, contienen los pares nombre-valor de los parámetros pasados en sentencia de ejecución, que son diferentes en función de la configuración del escenario. En el contenido de cada archivo, en la primera fila está la cabecera con los nombres de los parámetros y las distancias de las cuales se registran los valores de la PDR. A partir de la segunda fila, se registra los valores de los parámetros de configuración y la tasa de paquetes recibidos correctamente en función de la distancia.

N	O	P	Q	R	S	T	U	V
reevaluation	traffic_type	retransmissionActivated	NMAX	17	33	50	67	83
0	0	0	1	0.998138	0.999465	0.997969	0.996720	0.994499
0	0	0	1	0.997623	0.995574	0.992818	0.987768	0.985765
0	0	0	1	0.998538	0.997987	0.998646	0.997821	0.997931

Figura 14. Ejemplo del contenido de un archivo resultado en formato CSV

5.4. INTEGRACIÓN EN OAI

5.4.1. ARQUITECTURA

La implementación de la capa MAC de 5G NR V2X se ha integrado en OAI adaptando y evolucionando la arquitectura representada en la Figura 15. Esta arquitectura fue

implementada para LTE V2X en un anterior proyecto, y en este proyecto se ha adaptado y evolucionado para hacerla compatible con 5G NR V2X.

En esta integración se ha añadido nuevos parámetros en sentencia de ejecución correspondientes a las nuevas configuraciones del estándar 5G NR V2X. Estos nuevos parámetros hacen referencia a las nuevas funcionalidades implementadas en 5G NR V2X (ver Tabla 3 a partir del parámetro `RSRPthresholdMain`). Al igual que en el *código local*, en el código de OAI también presentaba los mismos problemas de la sección 5.2.9 y se ha aplicado las mismas resoluciones. En la Tabla 10 se presenta la correspondencia de variables del *código local* con OAI que tengan distintos nombres.

Local	OAI	Significado
durationTime	simul_time	Tiempo de simulación / emulación
sensingRSSI	tableRSSI	Tabla de valores RSSI
sensingWindow	sched_algo	Ventana de detección
NumPktRx	pktsCorrect	Registra cada paquete recibido correctamente
TotalNumPktRx	pktsTotal	Registra cada paquete recibido
NumPktRxAplication	pktsCorrectApplication	Registra cada paquete “diferente” recibido correctamente por cada receptor
TotalNumPktRxAplication	pktsTotalApplication	Registra cada paquete “diferente” recibido por cada receptor
tIdx	count_Meas	Índice de la ranura actual en la matriz de RSSI

Tabla 10. Correspondencia de variables entre el código local y OAI

Cada *Transceiver* que se muestra en el diagrama es un nodo UE implementado en OAI, por lo que cada uno es un vehículo en el sistema. OAI UE implementa capas PDCP (*Packet Data Convergence Protocol*), RLC (*Radio Link Control*) y MAC (*Medium Access Control*) en su pila de protocolos. En este trabajo se integra la capa MAC del *scheduler* 5G NR V2X.

En las capas inferiores, el OAI UE también implementa la capa PHY (*Physical*). Dada la complejidad de implementar esta capa PHY y el tiempo de desarrollo requerido, establecimos una colaboración con el centro EURECOM en Francia, el autor y principal desarrollador de OAI. En esta colaboración, EURECOM desarrollará la capa PHY y la integrará en el prototipo. Una implementación de capa PHY permite la transmisión de mensajes a través del medio radio utilizando el USRP apropiado. Esta implementación está siendo desarrollada actualmente por EURECOM, pero aún no está completa. Para resolver este problema, utilizamos un módulo que permite la abstracción de la capa PHY, llamado proxy o *Channel Abstraction*, como se muestra en la Figura 15. Este módulo permite que múltiples UE OAI se comuniquen y establece un canal de radio y las correspondientes capas físicas de cada nodo. Gracias a este módulo proxy, no estábamos limitados por la cantidad de dispositivos USRP (hardware) disponibles en nuestro laboratorio, por lo que pudimos emular la comunicación entre cientos de nodos. Otro aspecto importante de la arquitectura es la interfaz para intercambiar paquetes entre el proxy y varios UE. Esta interfaz se basa en Eclipse Mosquitto MQTT Message Broker y utiliza el formato de texto JSON para el intercambio de mensajes, lo que la hace muy flexible. La arquitectura también cuenta con una aplicación V2X externa para generar mensajes y un módulo de control de emulación que permite sincronizar específicamente el reloj interno del proxy con el reloj de la aplicación V2X.

Otro aspecto importante de la arquitectura es la interfaz para el intercambio de paquetes entre el *Proxy* y los múltiples UEs. Dicha interfaz está basada en el *broker* MQTT de mensajes Eclipse Mosquitto, haciendo uso además del formato de texto para intercambio de mensajes JSON, el cual ofrece una gran flexibilidad. La arquitectura cuenta también con una aplicación V2X externa para la generación de mensajes, y un módulo para el control de la emulación, que permite, entre otras cosas, sincronizar el reloj interno del *Proxy* con el de las aplicaciones V2X.

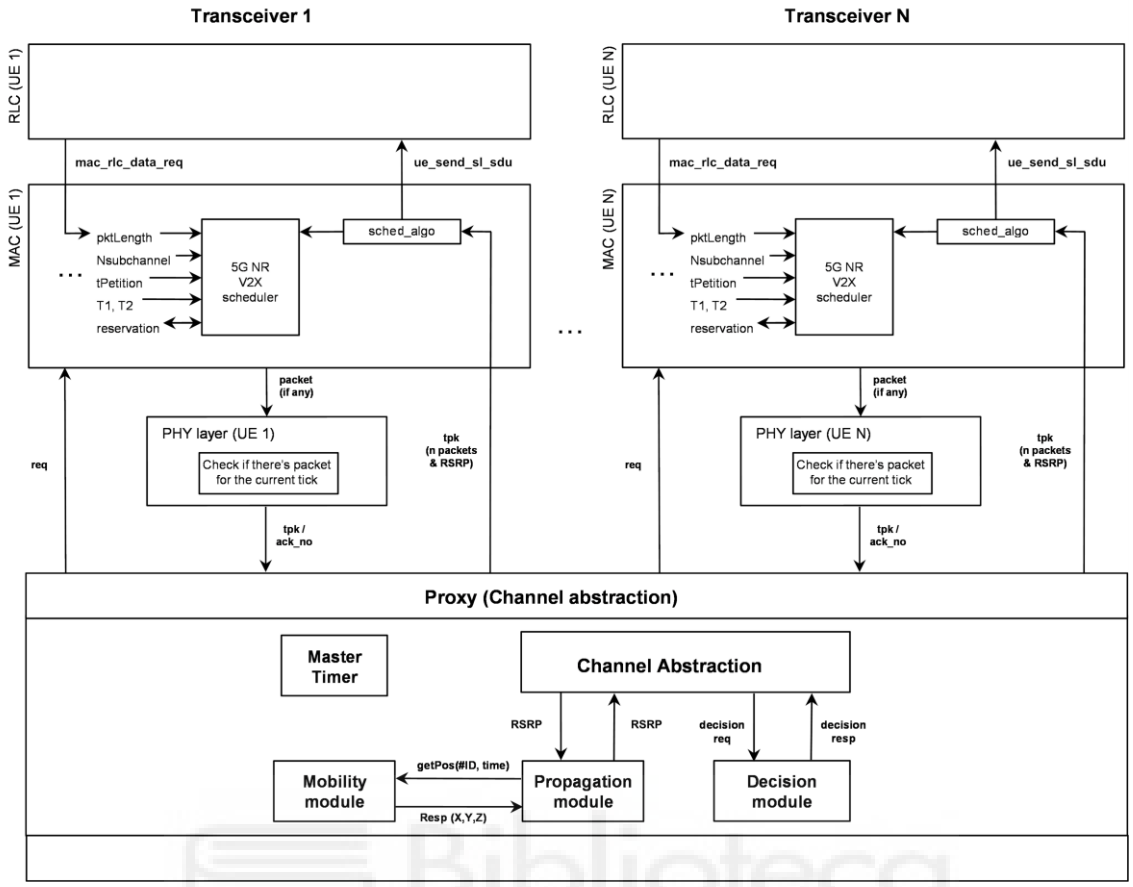

Figura 15. Arquitectura software general del prototipo implementado y su plataforma de emulación

5.4.2. FLUJO

La función *scheduler* del estándar 5G NR V2X se ha integrado en la capa MAC de OAI. En el momento que se recibe un paquete en la capa MAC proveniente de capas superiores, a través de la función `mac_rlc_data_req`, el *scheduler* debe indicar qué recursos radio se ha seleccionado para transmitir el nuevo paquete, teniendo en cuenta si se dispone de recursos preseleccionados válidos (reservados en la transmisión del último paquete), que de no disponerse, analizando los paquetes recibidos en la ventana de detección (se aplica el algoritmo de 2 pasos detallados en las secciones anteriores).

Este proceso se implementó creando un conjunto de hilos paralelos de ejecución. Los hilos principales son `mqtt_thread`, `timer_thread` y `UE_phy_stub_single_thread_rxn_txnp4`. El flujo de ejecución de estos hilos se explica brevemente a continuación. Esta descripción utiliza como referencia el diagrama de la Figura 16 y las estructuras, funciones e hilos de la Tabla 4 y la Tabla 11, que incluyen como referencia las estructuras de datos, los hilos y los archivos utilizados en la implementación:

1. Al inicio, el hilo `mqtt_thread` se queda a la espera de la recepción de un mensaje procedente del proxy. El proceso *Proxy* se encarga de controlar y gestionar cuándo se puede enviar paquetes desde la capa MAC hacia las capas inferiores para su transmisión. El formato del mensaje puede ser de tipo `req` (solicitud) o de tipo `tpk` (paquete), tal como se muestra en la Figura 16. La función `on_message` es la encargada de procesar los mensajes que recibe un nodo OAI UE. Con respecto a LTE V2X, en la estructura de los paquetes se han modificado las variables `subframe` y `subchannel` a vectores, porque en 5G NR V2X se puede incluir hasta un máximo de 2 reservas para las siguientes retransmisiones del mismo paquete. Y la variable `sizeSub` indica el tamaño de estos vectores. También se ha incluido la variable `priority` (prioridad del paquete) que servirá para implementar el mecanismo de congestión de tráfico.
2. El flujo de ejecución inicia de la siguiente forma. Primero el *Proxy* envía un mensaje de solicitud (`req`) al nodo OAI UE para saber si éste tiene algún paquete para transmitir. Entonces el OAI UE ejecuta la función `on_message`. Esta función se encargará de borrar los paquetes antiguos recibidos en la ventana de detección mediante la ejecución de la función `removeRangeFromListResources` y de actualizar el contador interno, que es representado por la variable `tick_proxy` y se refiere al instante actual. Después enviará una señal de activación al hilo `timer_thread`, tal como puede verse en la Figura 16.
3. Posteriormente el hilo `timer_thread` actualiza algunos contadores locales, responsables de verificar que las capas física y MAC están sincronizadas. A continuación, envía una señal de activación al hilo `UE_phy_stub_single_thread_rxn_txnp4` y éste llama a la función `phy_procedures_UE_SL_TX`. En este momento, si hay disponible un paquete para transmitir procedente de capas superiores, la función anterior primero llamará al *scheduler* (función `nr_v2x_scheduler`). En tal caso, se ejecuta el algoritmo de 2 pasos para identificar los recursos radio que se usarán para la transmisión del paquete.
4. Después de ejecutarse el *scheduler*, la función `phy_procedures_UE_SL_TX` verifica si el instante actual está reservado para transmitir algún paquete generado previamente en las capas superiores y puesto en cola hasta que lleguen los recursos o las ranuras seleccionadas para su transmisión y retransmisiones. El paquete para transmitir es enviado al *Proxy* y éste lo difunde al resto de OAI UEs. Este paquete se

envía en un mensaje de tipo `tpk`. Si no hay paquete, responde con un mensaje tipo `ack_no`. Para registrar la PDR de la capa aplicación, en el envío de paquetes de los UEs al proxy se incluye un valor para indicar si se trata de la primera transmisión del paquete o si se trata de una de las retransmisiones, cuando el *Proxy* recibe el valor 1 junto a la clave `first_trans`, entonces actualiza la matriz `first_transmission` con el valor *true* para todos los vehículos receptores (primer índice) y la posición `txIdx` del segundo índice (ID del vehículo transmisor). Entonces cada vez que el *Proxy* registra la recepción de un paquete (ya sea correctamente o no) para un vehículo con ID `rxIdx` (que no estaba también transmitiendo) en el vector `pktsTotal` (en función de la distancia entre transmisor y receptor) para la capa MAC, comprueba en la matriz `first_transmission` si en la posición `rxIdx-txIdx` el valor es true, en tal caso, también se incrementa en uno la posición correspondiente del vector `pktsTotalApplication` (capa aplicación). Después de actualizar el vector `pktsTotalApplication`, se marca a *false* la posición `rxIdx-txIdx` de la matriz `first_transmission` para indicar que las siguientes recepciones de paquetes en el vehículo `rxIdx` provenientes del vehículo `txIdx` se trata de retransmisiones del mismo paquete y que, por lo tanto, se actualiza el vector `pktsTotal` para la capa MAC, pero no se actualizaría el vector `pktsTotalApplication` (capa aplicación). Además de registrar la recepción de paquete en la capa aplicación, en la misma posición `rxIdx-txIdx` de la matriz `already_received_correctly` se inserta el valor *false*, que significa que todavía no se ha recibido correctamente el paquete en al menos una de sus retransmisiones. Posteriormente, si ese paquete es recibido "correctamente" por el vehículo `rxIdx`, entonces se actualiza el vector `pktsCorrect` como en el vector `pktsTotal` (en función de la distancia) para la capa MAC. De forma análoga al vector `pktsCorrect`, también se actualiza el vector `pktsCorrectApplication` si en la posición `rxIdx-txIdx` de la matriz `already_received_correctly` el valor es *false* y después se cambia el valor a *true*, porque en la capa aplicación solo se registra el paquete recibido correctamente una única vez para todas sus retransmisiones.

5. Una vez que todos los OAI UEs han respondido con el *tpk/ack_no* correspondiente, el *Proxy* procede a procesar todos estos mensajes. Primero cada paquete transmitido por los OAI UEs se replican al resto de los OAI UEs con el nivel de RSRP determinado. A continuación, en función del RSRP y el SINR, se decide qué paquetes

llegaron correctamente y qué paquetes se descartaron. Entonces el *Proxy* envía a cada OAI UE los paquetes que debe recibir y los niveles de RSSI que deben haber detectado.

6. Cada OAI UE recibe entonces un mensaje de tipo `tpk` del *Proxy*. Este mensaje es recibido por el hilo `mqtt_thread`, que también era el responsable de recibir los mensajes de tipo `req`. Como antes, este hilo llama a la función `on_message`. Esta función examina todos los paquetes recibidos correctamente por el OAI UE en este momento. Dichos paquetes se envían primero a las capas superiores a través de la función `ue_send_sl_sdu` y luego se agregan a la lista de paquetes (ventana de detección) del *scheduler* a través de la función `addListResourcesBySS`. Además, los niveles de ruido detectados actualmente se almacenan en la tabla RSSI del *scheduler*.
7. Más tarde el *Proxy* enviará mensajes `req` en la siguiente ranura y el OAI UE repetirá el proceso anterior. Después de que hayan pasado suficientes ranuras, el OAI UE ha acumulado los paquetes recibidos durante los últimos 1100 o 100 milisegundos (según la configuración) en `sched_algo` y los niveles de ruido en `tableRSSI`. Cuando llega el momento, se recibe un paquete de las capas superiores y la función `phy_procedures_UE_SL_TX` llama al *scheduler*. La función recibe como entrada la ventana de detección que consta de una lista de paquetes recibidos y una matriz con los valores RSSI en los últimos 1100 o 100 milisegundos. También obtiene los parámetros que definen la ventana de selección. Además, recibe el tamaño del paquete que se enviará e información de reservas anteriores. Finalmente, el *scheduler* devuelve información sobre la reserva.

Hilos de ejecución y funciones	**Archivos de OAI UE relacionados**
mqtt_thread	lte_ue.c
timer_thread	lte_ue.c
UE_phy_stub_single_thread_rxn_txnp4	lte_ue.c
phy_procedures_UE_SL_TX()	phy_procedures_lte_ue.c
ue_get_slsch()	ue_procedures.c

Tabla 11. Hilos de ejecución, funciones y archivos del OAI UE donde se encuentra la implementación

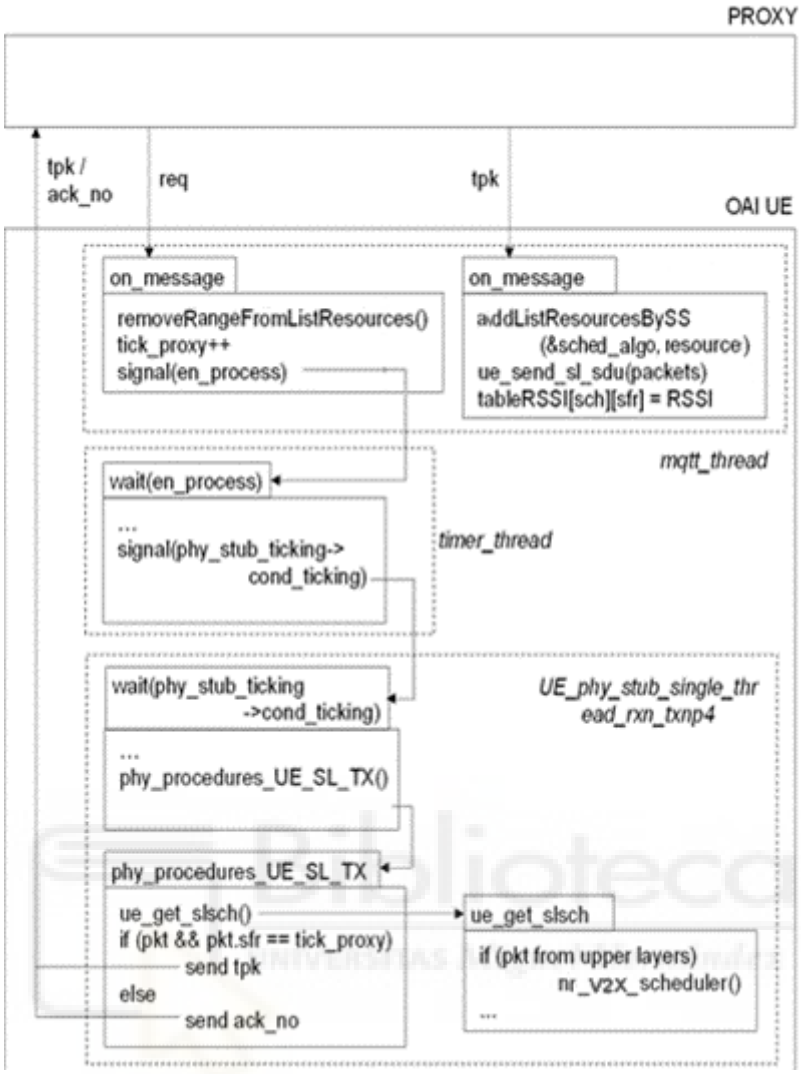

Figura 16. Flujo de datos e interfaz del scheduler 5G NR V2X con OAI

5.4.3. LIGHTWEIGHT UE

Se necesita el uso de múltiples nodos o vehículos para verificar y evaluar el correcto funcionamiento general del sistema. Esto requiere ejecutar múltiples OAI UEs, dada la arquitectura propuesta. Los escenarios de vehículos a gran escala pueden incluir fácilmente cientos de vehículos, por lo que se ha utilizado la herramienta Lightweight UE o LUE implementada en proyectos anteriores. LUE actúa como un nodo UE adicional en la emulación e implementa toda la lógica de comunicación e intercambio de mensajes con el *Proxy* y el *scheduler* (exactamente la misma implementación que la capa MAC de OAI UE). De hecho, el Proxy no distingue entre mensajes OAI UE y LUE. En este proyecto, se ha integrado y adaptado la implementación de la MAC de 5G NR V2X en el LUE. A diferencia del código de LTE V2X, la función *scheduler* de 5G NR V2X también puede ser ejecutada si el instante actual coincide con el instante de reevaluación de recursos. El

siguiente instante de petición de nuevos recursos solo es calculado (actualizado) cuando se trata de una petición de nuevos recursos, después de ejecutarse la función *scheduler*.

El LUE se implementa en el cliente Mosquitto y espera recibir mensajes JSON del proxy a través del hilo `listenning`. Por lo tanto, la función de `listenning` es equivalente a `mqtt_thread` de OAI. Similar a `mqtt_thread`, cuando `listening` detecta que se ha recibido un paquete, llama a `on_message` para procesarlo. El resto de la ejecución, es decir, la llamada al *scheduler*, las interacciones con el *Proxy*, el rellenado de la ventana de detección y demás, es igual que en el OAI UE. Sin embargo, no se emulan todas las capas del protocolo de comunicación.

Una de las principales ventajas del nodo LUE es que no tiene la complejidad de OAI UE y es suficiente para evaluar y emular la evaluación del *scheduler*. La Figura 17 muestra la configuración para evaluar y obtener resultados a gran escala utilizando un OAI UE, múltiples LUEs (número configurable en función de la densidad de tráfico a emular) y el *Proxy*.

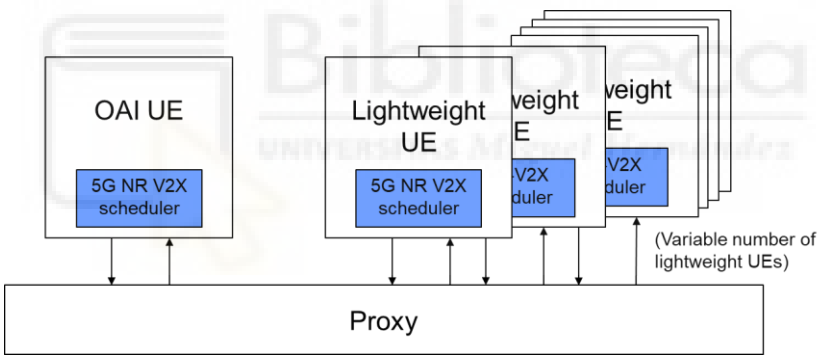

Figura 17. Emulación mediante un OAI UE y múltiple Lightweight UEs

5.5. AUTOMATIZACIÓN EN EL PROCESO DE EMULACIÓN

En esta subsección se expondrá qué tipos de automatizaciones se ha implementado en el código para el proceso de emulación tanto en *código local* como en OAI. En el proceso de emulación se puede destacar varios aspectos en referencia a la automatización implementada, estas automatizaciones suponen un posterior gran ahorro en el tiempo total de emulación, en el esfuerzo en lanzar las emulaciones y en revisar los ficheros resultado y en el tiempo y el esfuerzo en calcular la media de los resultados de todas las repeticiones para cada configuración distinta de emulación.

Se ha implementado un *script* de ejecución con todas las configuraciones de parámetros de escenario precisadas para lanzar las emulaciones de forma automatizada. Esto tiene

grandes ventajas, ya que permite al programador abstraerse del esfuerzo de tener que introducir manualmente todos y cada uno de los valores de los parámetros de configuración, evitando posibles confusiones en la introducción de los valores numéricos, así como su modificación para emular un escenario diferente. Otra ventaja es la ausencia de la necesidad del programador de estar pendiente de la finalización de una emulación para lanzar la siguiente repetición o ejecutar una emulación diferente, reduciendo al mínimo el tiempo de espera entre emulaciones consecutivas.

Una de las nuevas funcionalidades añadidas del estándar 5G NR V2X es la posibilidad de poder retransmitir el mismo paquete, en términos de emulación implica un mayor tiempo de ejecución al aumentar el número total de retransmisiones. Para repartir la carga de varias emulaciones simultáneas entre varios ordenadores[6], se ha añadido dos parámetros al script de ejecución que permite configurar el número total de emulaciones simultáneas, así como el número de emulación o proceso de ejecución. Esto permite dividir el tiempo total de todas las emulaciones, repartiendo la carga a una emulación por ordenador.

El reparto equitativo de la carga se procede de forma que el primer proceso o proceso 1, de un número determinado de X procesos, ejecuta todas las configuraciones y repeticiones para 1 transmisión por paquete, el proceso 2 ídem para 2 transmisiones por paquete, el proceso 3 para 3 transmisiones y así sucesivamente. Una vez finalizado las emulaciones para el primer grupo de X números consecutivos de transmisiones, los procesos anteriores pasan a ejecutar las emulaciones del siguiente grupo con la misma cantidad de números consecutivos de transmisiones, de forma que el proceso 1 pasa de emular con el primer número del primer grupo a emular con el último número del segundo grupo para el número total de transmisiones por paquete, el proceso 2 pasa del segundo número al penúltimo número, el proceso 3 del tercer número al antepenúltimo número y así sucesivamente. Cada proceso se va alternando entre esos dos grupos, teniendo en cuenta que, a mayor número de retransmisiones, mayor es el tiempo de emulación. Por lo tanto, el proceso 1 terminará primero y para balancear la carga, en el siguiente grupo pillará las emulaciones que demoran más tiempo, con un mayor número de retransmisiones.

Para un ejemplo de 4 procesos, el primer grupo lo forma {1, 2, 3, 4} transmisiones por paquete, el segundo grupo con {5, 6, 7, 8} transmisiones, el tercero con {9, 10, 11, 12},

[6] En el caso del *código local*, el reparto de la carga se puede realizar entre varios núcleos del procesador.

el cuarto con {13, 14, 15, 16} y así sucesivamente. El proceso 1 tomaría los números 1, 8, 9, 16…, el proceso 2 tomaría 2, 7, 10, 15… Esto permite que todos los procesos finalicen en un tiempo total similar.

Otra capa adicional de automatización en el proceso de emulación es la comprobación automática del número de repeticiones de emulaciones completadas para una determinada configuración. Antes de ejecutar un determinado número de repeticiones para un escenario de emulación determinado, en el script de ejecución se ejecuta otro programa en lenguaje C, para comprobar cuántas repeticiones son completadas para esa emulación. Este programa recibe como parámetros los mismos que el programa de emulación y devuelve el número de repeticiones completadas para esa emulación, esto le permite al script saltarse esas repeticiones y ejecutar únicamente las repeticiones restantes. Esto es una ventaja importante, ya que, si se produce un apagón, el programador no necesita relanzar las emulaciones desde cero o pasar mucho tiempo y esfuerzo en revisar manualmente los ficheros resultado y reconfigurar el script de ejecución. El proceso *Proxy* de OAI puede devolver 3 posibles valores, que son 0, -1 y 1, que significan todo correcto, memoria insuficiente y error en el número de parámetros, respectivamente. Si devuelve 0, el script incrementa en uno el contador del número de repeticiones, si devuelve -1, se mantiene igual el contador de repeticiones, por lo tanto, se reinicia esa repetición, y si devuelve 1, el script ejecuta la instrucción `break` para salir del bucle de repeticiones. En el segundo caso, el programa de emulación muestra un mensaje por consola indicando que ha fallado alguna reserva o redimensión de memoria dinámica y en el último caso, también se muestra un mensaje por consola indicando el error.

Por último, se ha implementado otro programa en lenguaje C para calcular la media de todas las PDRs de cada configuración de emulación distinta. Por cada nueva emulación es generado dos nuevos ficheros CSV que contendrá todas las PDRs de la misma, una fila por cada repetición. Este programa funciona por comandos, uno de los comandos calcula la media de las PDRs de todos los ficheros CSV cuyos nombres tienen un prefijo común, este prefijo es pasado como parámetro al comando. Todas las medias calculadas son guardadas en un mismo fichero generado con el nombre del prefijo y con la extensión CSV. Durante el proceso son generados 4 ficheros CSV, es decir, el comando es ejecutado 4 veces con 4 prefijos diferentes. Por un lado, los ficheros CSV son divididos en densidad 60 y densidad 120, ya que para densidad 120 se genera la PDR para una mayor cantidad de distancias. Por otro lado, los ficheros CSV son divididos en capa aplicación y capa

MAC, debido a que cada emulación genera dos ficheros CSV, uno para la capa aplicación y otro para la capa MAC, para evitar posibles futuras confusiones.

5.6. CÓDIGO

En esta subsección se hará una breve descripción de archivos de las capas MAC y física de OAI que se han empleado, así como los archivos empleados para el lanzamiento de las emulaciones. El código de OAI está organizado en los siguientes archivos:

- En la carpeta raíz se encuentran los archivos para la compilación (archivo compile) y la ejecución (archivos run.sh y apps.sh) del proyecto de OAI, así como instrucciones para preparar el equipo para lanzar las emulaciones, descargar el repositorio e instalar las librerías y funcionalidades necesarias y las sentencias de compilación y ejecución de OAI (archivo Instructions). También se incluye el código fuente en lenguaje C (archivo times_completed_simulation.c) de un programa que es ejecutado por el archivo de ejecución run.sh antes de lanzar cada escenario para comprobar el número de emulaciones finalizadas en anteriores ejecuciones. Todos estos archivos y los archivos de los siguientes puntos, así como los directorios donde se ubican son anotados en el archivo “Edited files”.
- El nodo OAI está implementado en múltiples archivos. En los archivos lte-uesoftmodem.c y lte-softmodem.h se encuentran la función main y los parámetros de la configuración de escenario. En los archivos transport_common.h, mac_proto.h y pre_v2x_processor.c están las estructuras definidas (información de la reserva, paquete a transmitir, ventana de detección, lista de recursos seleccionados, ventana de selección, lista de recursos transmitidos…), los prototipos de las funciones del *scheduler* y de las gestiones de memoria para las estructuras definidas y las implementaciones de las funciones anteriores, respectivamente. Y finalmente en los archivos ue_procedures.c, phy_procedures_lte_ue.c y lte-ue.c se encuentran la integración de la función *scheduler* y el código con la transmisión y la recepción de paquetes con el proceso *Proxy*, respectivamente.
- Ahora se describirán los archivos del nodo Lightweight UE. El archivo client.c contiene la función main, la declaración e inicialización de variables y estructuras necesarias (información de la reserva, ventana de detección, lista de recursos seleccionados, lista de recursos transmitidos…), los parámetros de configuración

de escenario pasados en sentencia de ejecución, la integración de la función *scheduler*, las comunicaciones con el proceso *Proxy*… Los archivos scheduler_fb2.h y scheduler_fb2.c incluyen las estructuras definidas, los prototipos y las implementaciones de las funciones del *scheduler*. Y los archivos resourceList.h y resourceList.c incluyen los prototipos y las implementaciones de las funciones para la gestión de memoria de las estructuras definidas en scheduler_fb2.h.

- Por último, están los archivos del proceso *Proxy*. En los archivos proxy_mac.c, common.c y common.h incluyen código para las comunicaciones con los nodos UE, así como la gestión de los paquetes que tiene que recibir cada nodo UE y el registro de la tasa de paquetes recibidos correctamente tanto en la capa MAC como en la capa aplicación en archivos en formato CSV.

6. RESULTADOS

6.1. CONDICIONES Y PARÁMETROS DE CONFIGURACIÓN

La Tabla 12 muestra las condiciones y los parámetros de configuración utilizados para evaluar la implementación del *scheduler* 5G NR V2X. Es importante tener en cuenta que estas condiciones de evaluación son las mismas que las especificadas por 3GPP como método de evaluación para simulaciones en el sistema. El análisis de los resultados se basa principalmente en la métrica PDR. Esta métrica refleja la tasa de paquetes recibidos correctamente por los vehículos en función de la distancia entre el transmisor y el receptor. En la sección 6.2 se presenta la validación de la implementación realizada en varios escenarios comparando los resultados con una implementación existente. En las secciones 6.3 a 6.7 se emplea la implementación realizada para estudiar el impacto de diferentes factores en el rendimiento de 5G NR V2X. En la sección 6.8 se muestran resultados empleando la integración en OAI. Se proporcionan condiciones de evaluación específicas en cada una de las siguientes subsecciones, que son diseñadas para validar y evaluar el impacto de los factores clave que pueden afectar al rendimiento de 5G NR V2X.

Escenario y modelo de tráfico	
Escenario	Autopista 3GPP
Densidad de vehículos [veh. /km]	60 – 120
Modelos de tráfico	Periódico y aperiódico
Configuración del resource pool	
Ancho de banda (BW)	20 MHz
SCS	30 kHz
RBs/BW	51 RBs
Tamaño del subcanal	12 RBs/subcanal
MAC	
T0 (sensing Window)	1100 ms
Tproc,0 (sensing Window)	1 slot (SCS 30 kHz)
T1 (select. W)	2 slots
Tproc,1 (select. Window)	5 slots (SCS 30 kHz)
T2 (selection Window)	PDB
% min. recursos	20
Umbral RSRP	-128 dBm
RRI	100 ms
Probability P	0
PHY	
MCS	16QAM-r0.5
1st-stage SCI format	12 RBs and 2 symbols
2nd-stage SCI format	48 bits (Huawei, R1-2005796)
in band emissions	TS 38.101-1

LUTs TB	Huawei: R1-1900852 & R1-1901542
LUTs SCI (1st&2nd stage)	Ericsson: R1-1903180
Sensitivity	-89.4 dBm (30 kHz & 20 MHz)
Tx Power	23 dBm
Noise (dBm)	-174 + 10 * log10(BW_RX) + noiseFigure
Noise Figure	9 dBm
Antenna model	Omnidirectional, Gain: 0dB
Channel model	
Pathloss & shadowing models	TR 37.885
Frequency	5.9 GHz
Decorrelation distance	25 meters
Fast fading	LUTs

Tabla 12. Parámetros de evaluación utilizados en la plataforma hardware OAI

6.2. VALIDACIÓN DEL SCHEDULER 5G NR V2X

El *scheduler* implementado se ha validado comparando las curvas de PDR obtenidas en varias configuraciones con las curvas obtenidas mediante una implementación en el simulador ns-3 disponible en el grupo de investigación. Esta implementación en ns-3 modela en detalle la capa MAC y ha sido extensamente validada y testeada. El objetivo de esta subsección es contrastar los resultados de la PDR del *código local* de OAI[7] y el simulador ns-3. En concreto se ha seleccionado 4 escenarios con densidades de vehículos de 60 veh/km y 120 veh/km y para 1 o 2 transmisiones por paquete generado. Estas curvas PDR muestran los resultados para la capa aplicación. Las diferencias entre la capa MAC y la capa aplicación se expondrá en la sección 6.7. Los 4 escenarios tienen una configuración de una tasa de generación de paquetes de 10 paq/s, con el mecanismo de reevaluación desactivado y tráfico de tipo aperiódico. La Figura 18 y la Figura 19 muestran los resultados para los escenarios con una densidad de vehículos de 60 veh/km y para 1 y 2 transmisiones, respectivamente. Y la Figura 20 y la Figura 21 reflejan la PDR para configuraciones de una densidad de 120 veh/km y también para 1 y 2 transmisiones, respectivamente. Se puede observar que las curvas PDR son similares en los 4 escenarios. La tasa de paquetes recibidos correctamente en OAI es ligeramente superior en la zona central de los 4 escenarios y posteriormente las dos curvas se cruzan a una distancia de aproximadamente de 800-950 metros, a partir de la cual la PDR es ligeramente superior en el simulador ns-3 hasta la distancia de 1 km. Las diferencias producidas pueden

[7] En el *código local* se ha realizado un mayor número de repeticiones en las simulaciones de cada escenario, por lo tanto, las curvas PDR están más suavizadas que las de OAI. En la subsección 6.8 se verá que las curvas PDR del *código local* tienen una coincidencia total con las curvas PDR de OAI.

deberse al modelado en el simulador de efectos de propagación de forma más precisa, como las IBE (*In-Band Emissions*), que son interferencias producidas en RBs adyacentes.

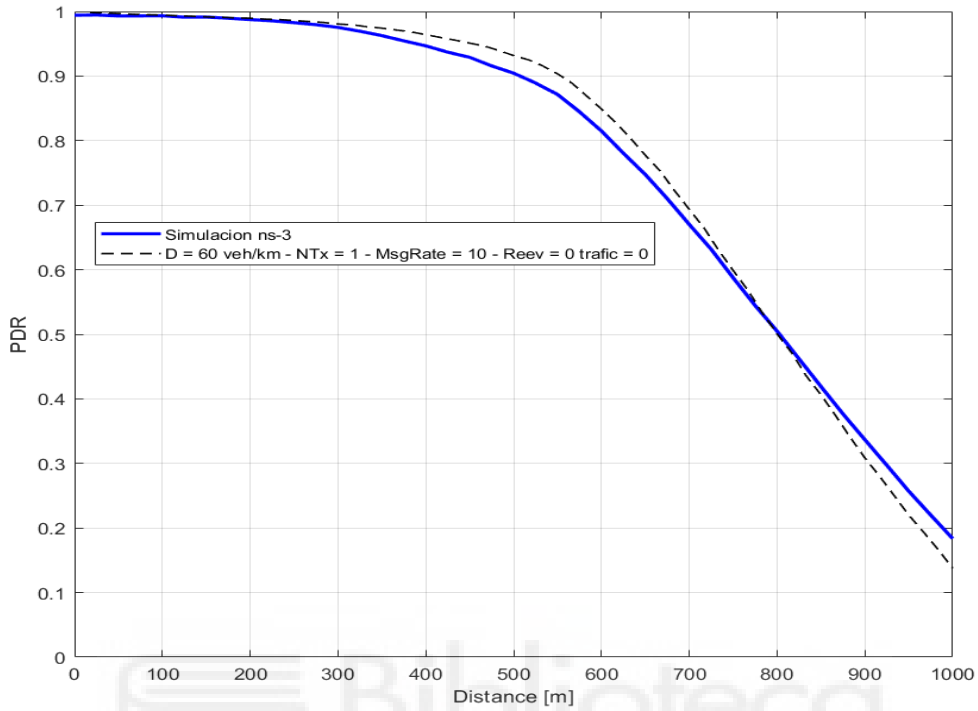

Figura 18. Comparativa de la PDR con ns-3 en función de la distancia para densidades de vehículos de 60 veh/km con NTx = 1

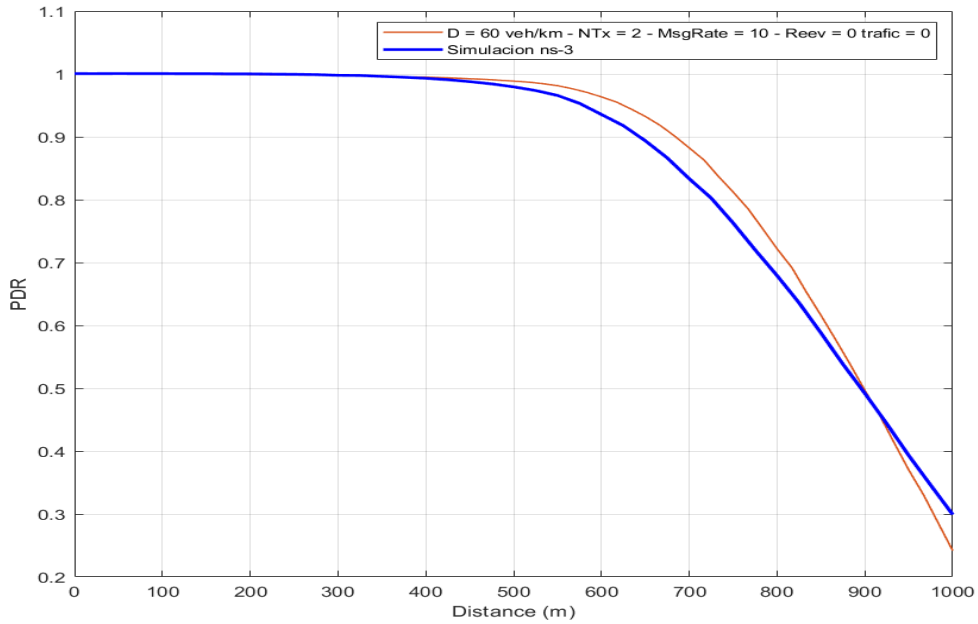

Figura 19. Comparativa de la PDR con ns-3 en función de la distancia para densidades de vehículos de 60 veh/km con NTx = 2

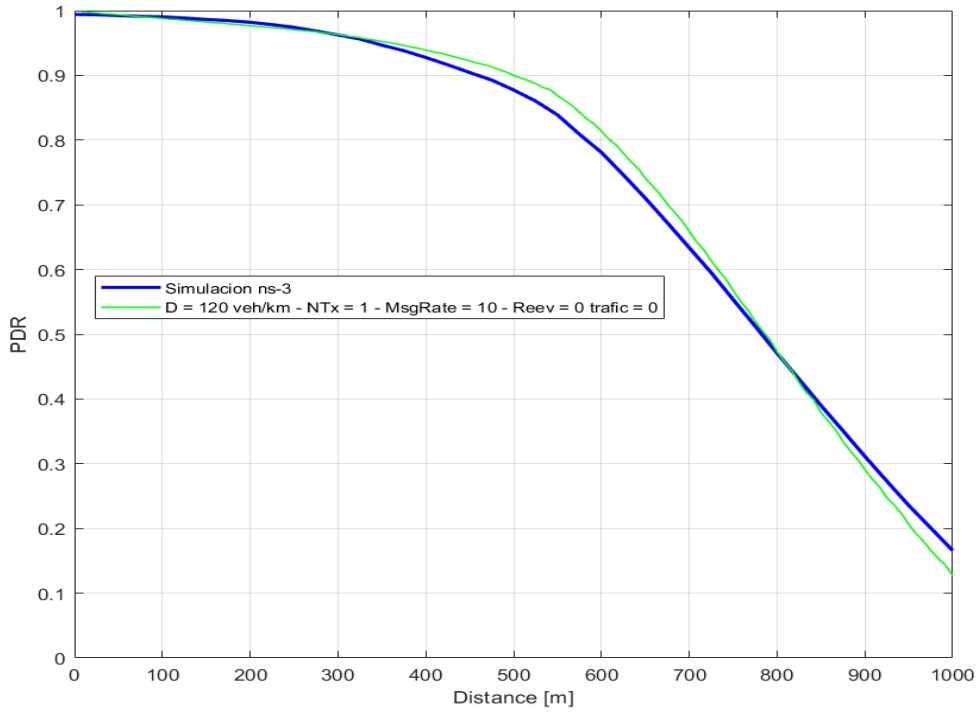

Figura 20. Comparativa de la PDR con ns-3 en función de la distancia para densidades de vehículos de 120 veh/km con NTx = 1

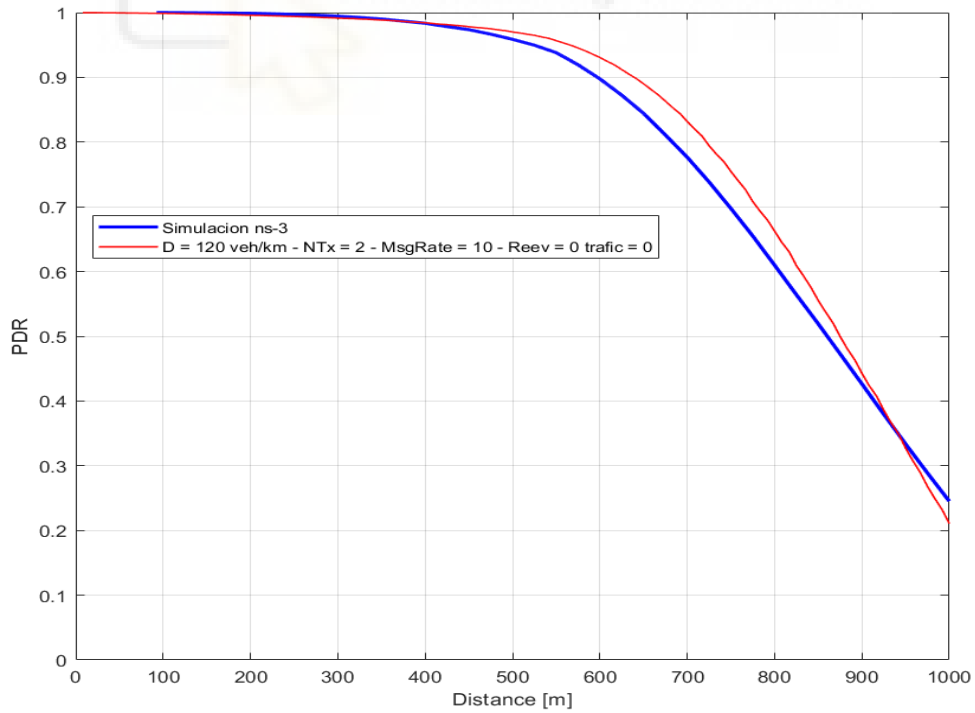

Figura 21. Comparativa de la PDR con ns-3 en función de la distancia para densidades de vehículos de 60 veh/km con NTx = 2

6.3. IMPACTO DE LA DENSIDAD DE VEHÍCULOS

Esta subsección se centra en analizar el impacto del aumento de la densidad de vehículos en la PDR en 5G NR V2X. En la Figura 22 se compara la PDR para escenarios con densidades de 60 y 120 veh/km. Se puede observar en la figura cómo el aumento de la densidad de vehículos en el escenario afecta a la PDR. Como ejemplo, para una distancia de separación de 300 m entre el emisor y el receptor, la PDR es de 98,06 % para una densidad de 60 veh/km, que es reducido al 96,29 % para una densidad de 120 veh/km. Este menor rendimiento en la PDR, como es lógico, es debido a una mayor carga del canal. Al aumentar la densidad de vehículos hay una mayor probabilidad de que varios vehículos inicien el proceso de selección de recursos en instantes de tiempo similares, de forma que sus ventanas de selección se solapen y que al seleccionar aleatoriamente el mismo recurso para transmitir, produzca colisiones.

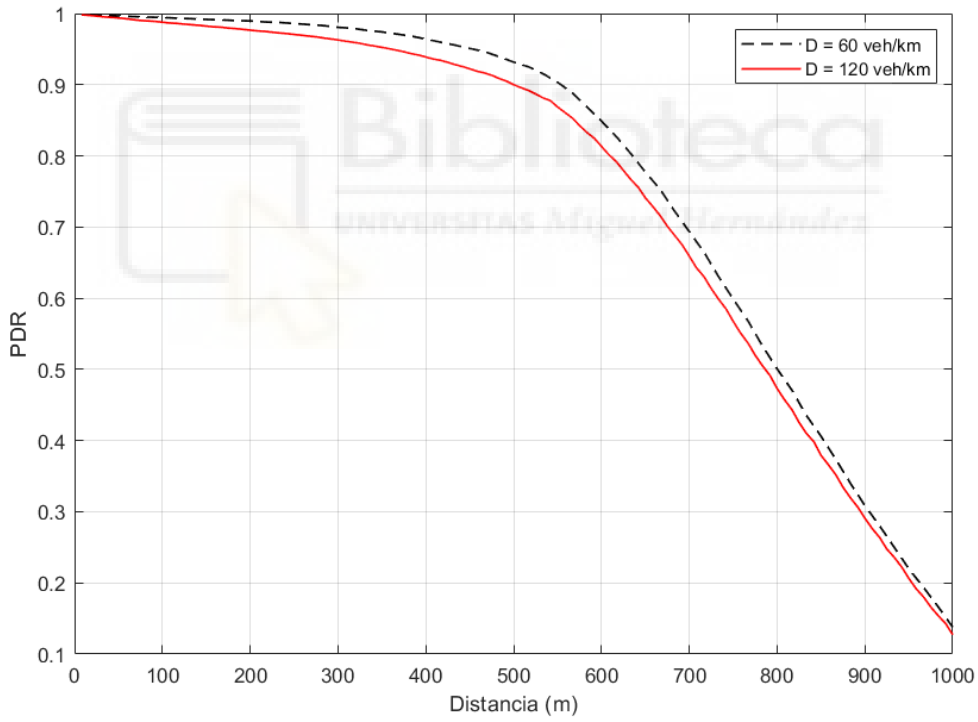

Figura 22. PDR en función de la distancia para densidades de vehículos de 60 veh/km y 120 veh/km

6.4. IMPACTO DE LA TASA DE GENERACIÓN DE TRÁFICO

Al igual que la subsección anterior, también se produce una bajada del rendimiento en la PDR, cuando se genera una mayor tasa de paquetes para transmitir por segundo. En este caso, en la Figura 23 se compara las PDRs obtenidas para escenarios donde los vehículos generan tasas de paquetes de 10 paq/s, 20 paq/s y 50 paq/s para una densidad de 60

veh/km. En este caso, el aumento de la carga del canal supone una reducción de la PDR al 96,99 % y 92,48 % para la misma distancia de 300 m. y tasas de 20 paq/s y 50 paq/s, respectivamente. Como recordatorio, en la subsección anterior, el doble de densidad de vehículos para la misma tasa de generación de paquetes (10 paq/s), la PDR a 300 m. era del 96,29 %. Por otro lado, en la Figura 23 se observa que al duplicar la tasa de generación de paquetes, la PDR a 300 m. se reduce del 98,06 % a 96,99 % y que al quintuplicar la tasa se reduce al 92,48 % a la misma distancia.

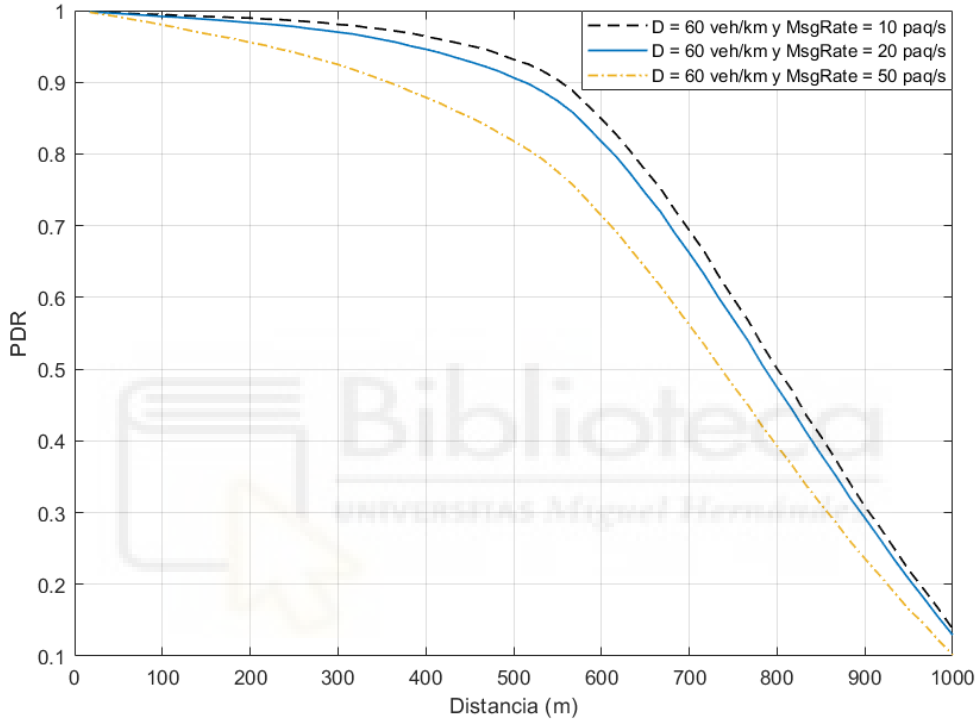

Figura 23. PDR en función de la distancia para densidades de vehículos de 60 veh/km y tasa de generación de tráfico de 10 paq/s y 50 paq/s

6.5. IMPACTO DEL TIPO DE TRÁFICO

Además de la carga de tráfico, otro aspecto importante a tener en cuenta del rendimiento de 5G NR V2X es el tipo de tráfico generado por los vehículos. En el modo 2 de 5G NR V2X, el esquema semipersistente para la selección de recursos para la transmisión permite reutilizar los recursos usados en las transmisiones del último paquete, actualizando las ranuras de transmisión pasadas sumándole un valor de RRI. Esta periodicidad en las transmisiones se adecua mejor a un patrón de tráfico periódico en la generación de nuevos paquetes para transmitir. Sin embargo, la generación de paquetes en el estándar 5G NR V2X sigue un ciclo aperiódico. Por consiguiente, el organismo 3GPP recomienda realizar

las emulaciones de 5G NR V2X también en escenarios con tráfico aperiódico. En la Figura 24 se observa una reducción de la PDR al 96,96 % a una distancia de 300 m. al tratarse de un escenario con tráfico aperiódico. Esto se debe a que parte de las reservas realizadas en las transmisiones previas no se puede reutilizar, teniendo que seleccionar nuevos recursos que puedan colisionar con los recursos de otros vehículos que también ejecuten el mismo proceso de selección de nuevos recursos.

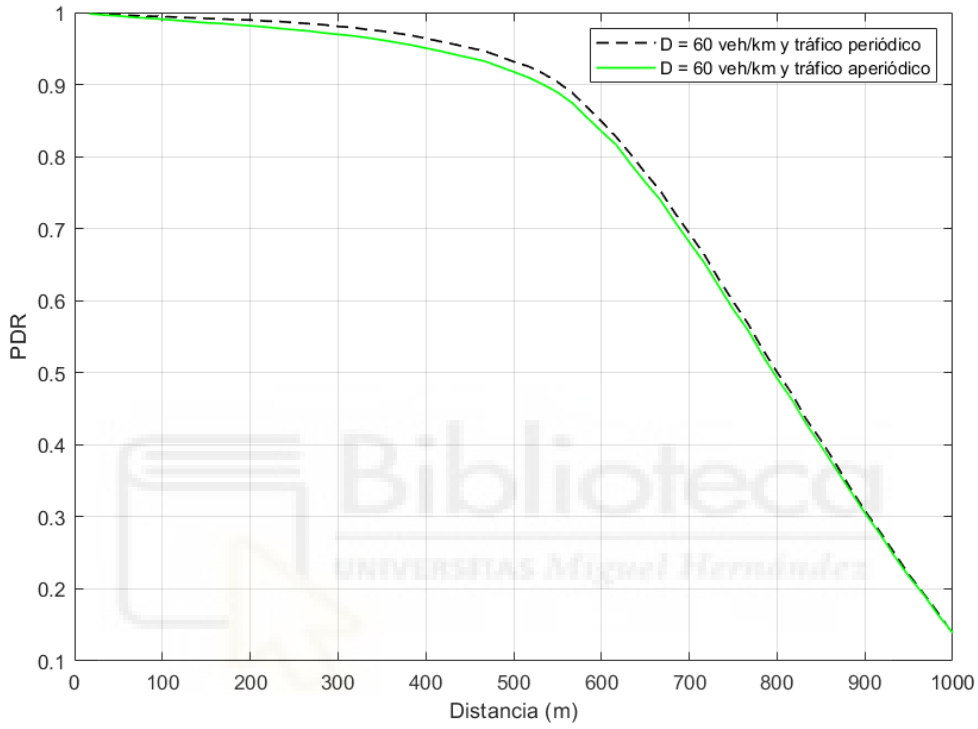

Figura 24. PDR en función de la distancia para densidades de vehículos de 60 veh/km y tráfico periódico y aperiódico (tasa de generación de tráfico de 10 paq/s)

6.6. IMPACTO DEL MECANISMO DE REEVALUACIÓN

Esta sección pretende ofrecer un análisis complementario mediante el análisis del rendimiento del mecanismo de reevaluación utilizando la implementación realizada sobre la plataforma OAI. En concreto, se ha seleccionado un escenario con una densidad de 60 veh/km, con una tasa de paquetes de 10 paq/s y 3 transmisiones por paquete y se compara el rendimiento cuando el mecanismo de reevaluación está activado y desactivado. En la Figura 25 se deduce que la activación del mecanismo de reevaluación permite detectar los recursos seleccionados que entran en conflicto con las reservas de otros vehículos. Una nueva reselección de los recursos mejora la PDR por una menor cantidad de colisiones en las transmisiones. En la Figura 25 puede verse que se produce un aumento

de la PDR del 95,39 % al 97,13 % a una distancia de 300 m. cuando el mecanismo de reevaluación pasa de estar desactivado a activado, respectivamente. Debido a una tasa de PDR muy elevada para un escenario de una transmisión por paquete, se aprecia una mejora de rendimiento en la activación del mecanismo de reevaluación a partir de una configuración de 2 transmisiones por cada paquete nuevo generado. La Figura 25 muestra la PDR para un escenario de 3 transmisiones por cada paquete.

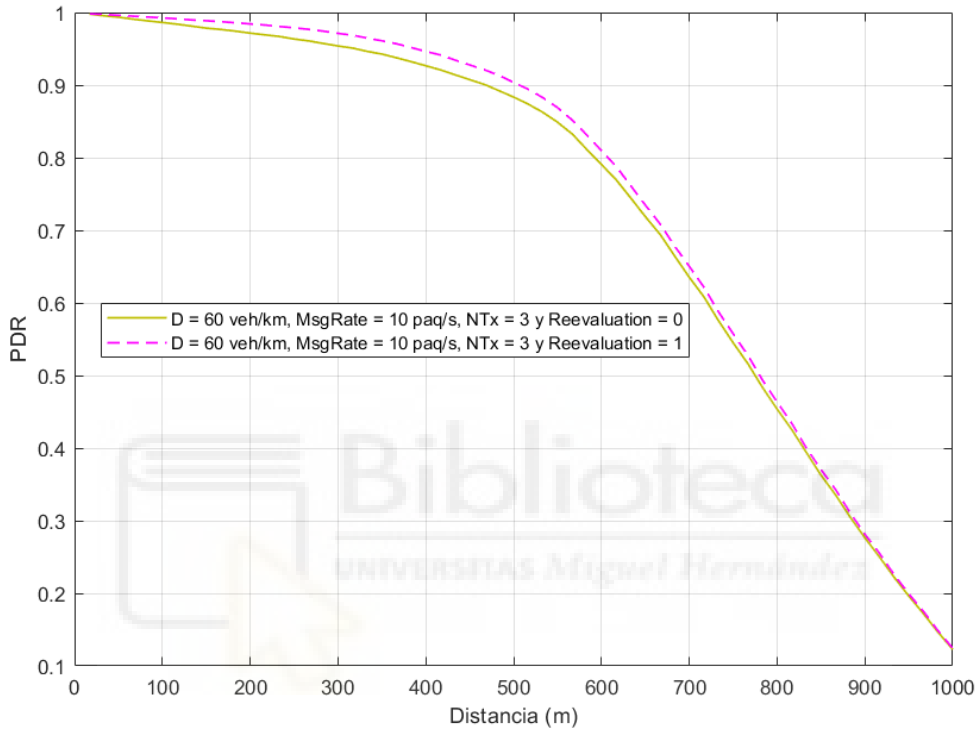

Figura 25. PDR en función de la distancia para densidades de vehículos de 60 veh/km cuando el mecanismo de reevaluación está activado y desactivado (tasa de generación de tráfico de 10 paq/s, tráfico periódico y 3 transmisiones/paq)

6.7. IMPACTO DEL NÚMERO DE RETRANSMISIONES

En esta subsección se analizará el impacto de las retransmisiones en el rendimiento de 5G NR V2X. En la sección 5.2 se ha descrito la implementación del mecanismo de scheduler del modo 2 de 5G NR V2X que permite la posibilidad de realizar hasta un total de 32 transmisiones por cada nuevo paquete generado. Todas las retransmisiones de un mismo paquete se realizan en un intervalo inferior a RRI, que es el intervalo de reserva de recursos. El objetivo de las retransmisiones es maximizar la posibilidad de que un paquete generado se reciba correctamente al menos una vez en una de esas retransmisiones. En la Figura 26 se muestra la PDR a nivel MAC para escenarios con NTx = {1-5, 10, 15, 20,

25, 32} transmisiones por cada nuevo paquete generado. La PDR a nivel MAC mide la ratio de paquetes recibidos correctamente independientemente de que varias transmisiones pertenezcan al mismo paquete. Por lo tanto, a nivel MAC el incremento del número de retransmisiones por paquete disminuye el rendimiento de 5G NR V2X, debido a una mayor carga del canal, al igual que en las secciones 6.3 y 6.4. Este efecto en el rendimiento puede apreciarse en la Figura 26, donde se muestra que para una distancia de separación de 300 m. entre el transmisor y el receptor, la PDR medida cuando el número de transmisiones es NTx = {1, 2, 3, 4, 5, 10, 15, 20, 25, 32} es {98.06, 96.62, 95.39, 93.96, 92.30, 82.00, 64.20, 47.76, 34.98, 22.97} %, respectivamente.

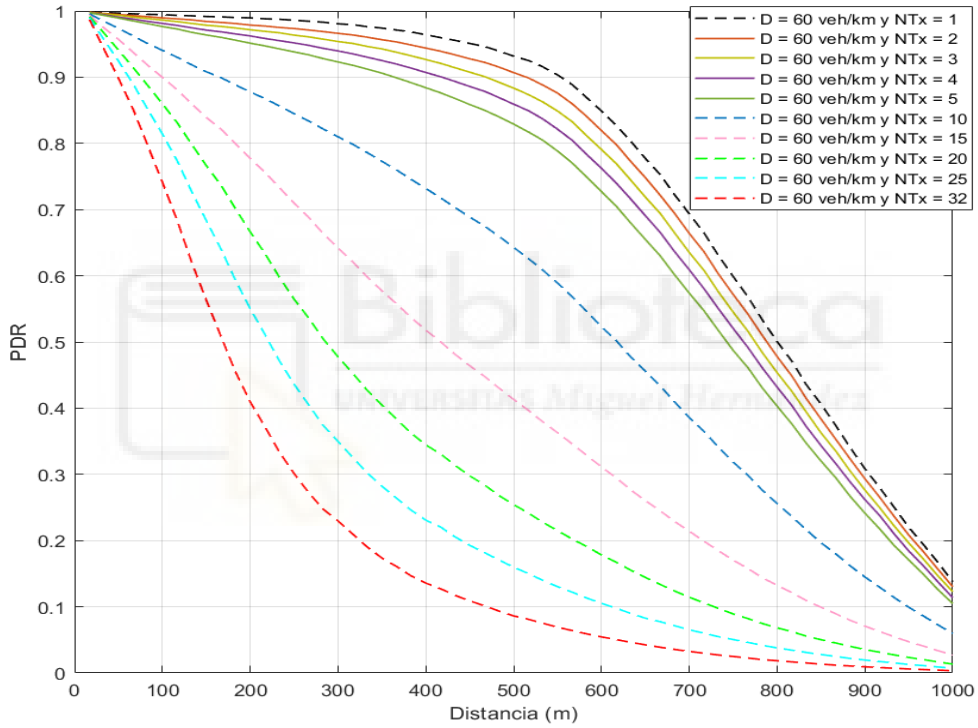

Figura 26. PDR en función de la distancia para densidades de vehículos de 60 veh/km cuando se realizan NTx = [1, 5] y NTx = {10, 15, 20, 25, 32} transmisiones del mismo paquete (tasa de generación de tráfico de 10 paq/s)

Sin embargo, a nivel aplicación es suficiente que se reciba correctamente el paquete en una de las retransmisiones para determinar que el paquete se ha recibido correctamente, ya que se trata de copias del mismo paquete. En la Figura 27 y la Figura 28 se muestran la PDR a nivel aplicación para un escenario con una densidad de 60 veh/km. Por otro lado, la Figura 29 y la Figura 30 también muestran la PDR a nivel aplicación, pero para un escenario con una densidad de 120 veh/km. Los resultados obtenidos demuestran una mejoría en la PDR al incrementar el número de retransmisiones por paquete hasta un

punto óptimo. Para el escenario de 60 veh/km el punto óptimo es de 8 transmisiones por paquete (NTx = 8). Y para el escenario de 120 veh/km el punto óptimo es de 4 transmisiones (NTx = 4), aunque la PDR para 9 transmisiones por paquete (NTx = 9) sigue siendo superior que cuando no se realiza retransmisiones (NTx = 1).

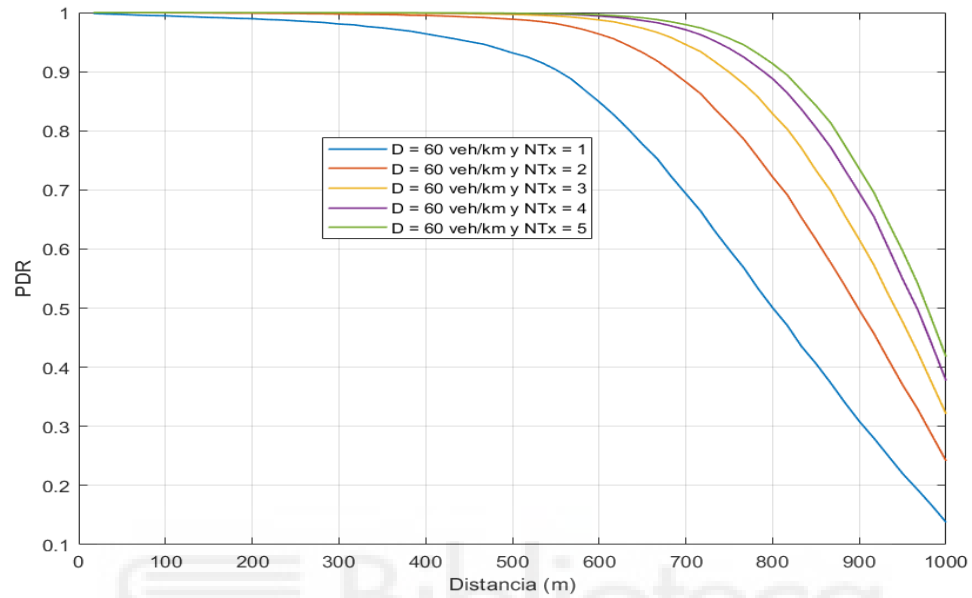

Figura 27. PDR a nivel de APP en función de la distancia para densidades de vehículos de 60 veh/km cuando se realizan NTx = [1, 5] transmisiones del mismo paquete (tasa de generación de tráfico de 10 paq/s)

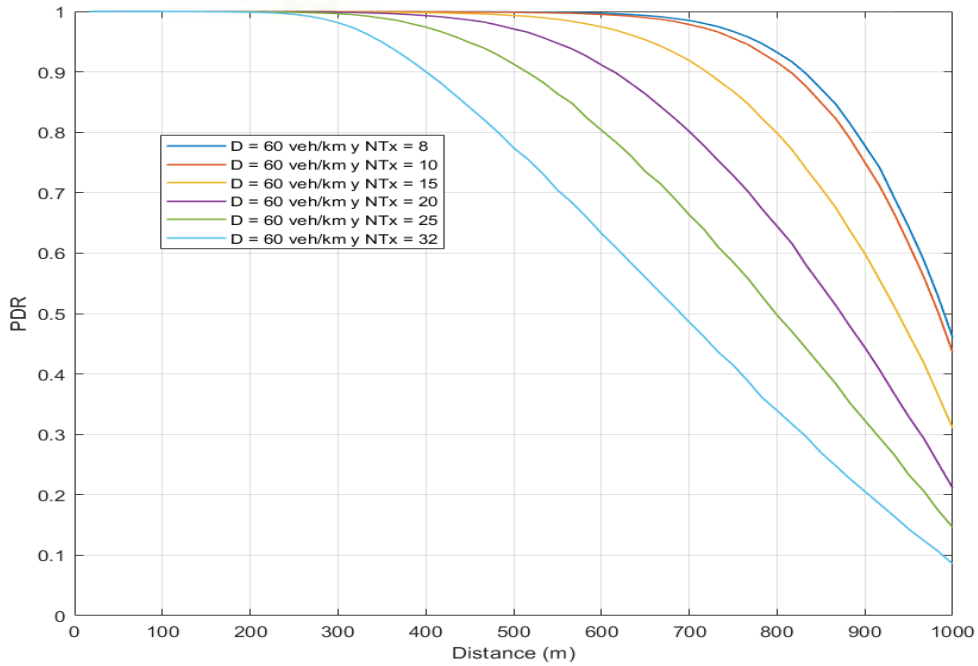

Figura 28. PDR a nivel de APP en función de la distancia para densidades de vehículos de 60 veh/km cuando se realizan NTx = {8, 10, 15, 20, 25, 32} transmisiones del mismo paquete (tasa de generación de tráfico de 10 paq/s)

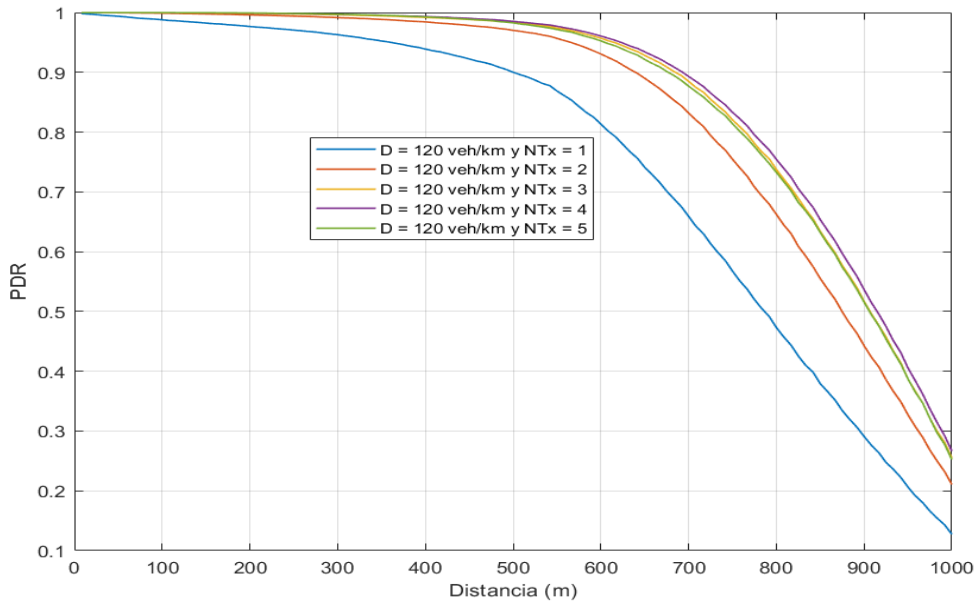

Figura 29. PDR a nivel de APP en función de la distancia para densidades de vehículos de 120 veh/km cuando se realizan NTx = [1, 5] transmisiones del mismo paquete (tasa de generación de tráfico de 10 paq/s)

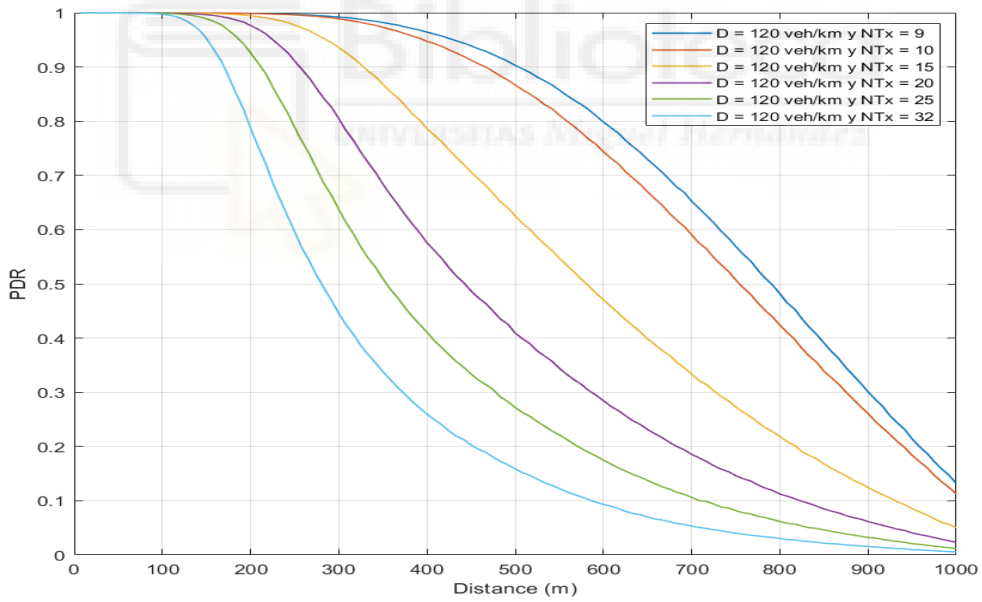

Figura 30. PDR a nivel de APP en función de la distancia para densidades de vehículos de 120 veh/km cuando se realizan NTx = {9, 10, 15, 20, 25, 32} transmisiones del mismo paquete (tasa de generación de tráfico de 10 paq/s)

6.8. RESULTADOS EMPLEANDO LA INTEGRACIÓN EN OAI

Como se ha mencionado anteriormente, la evaluación detallada de las funcionalidades del estándar 5G NR V2X en la capa MAC se ha realizado primero en *código local*, que permite depurar, testear y validar el código de una manera más sencilla y rápida en

comparación con el código de OAI. Gracias a esta simplificación del código de OAI, se ha podido detectar y corregir muchos errores del código tanto en tiempo de compilación como violaciones de memoria en tiempo de ejecución. Una vez comprobado el correcto funcionamiento del *código local* y evaluado que el impacto en la PDR para una enorme configuración de escenarios sea correcto, se procede a integrar dicho código en el proyecto de OAI. El paso final para verificar que la integración del código en OAI se ha realizado de manera correcta es haciendo una comparación de las curvas PDR del *código local* con las de OAI. En concreto se ha seleccionado 4 tipos de escenarios. El primer escenario se corresponde con la configuración de 60 veh/km de densidad de vehículos de la sección 6.3, se puede observar en la Figura 31 que la coincidencia de las 2 curvas PDR es absoluta. Lo mismo ocurre con el resto de las gráficas. El segundo escenario (Figura 32) se corresponde con un escenario con tráfico aperiódico, en concreto es el mismo escenario aperiódico de la sección 6.5. En la Figura 33 está activado el mecanismo de reevaluación para una densidad de vehículos de 60 veh/km, 1 transmisión por paquete, una tasa de generación de paquetes de 10 paq/s y tráfico periódico. Y en la Figura 34 tiene una configuración de 4 transmisiones por paquete, una tasa también de 10 paq/s, con el mecanismo de reevaluación desactivado, tráfico periódico y para una densidad de 60 veh/km.

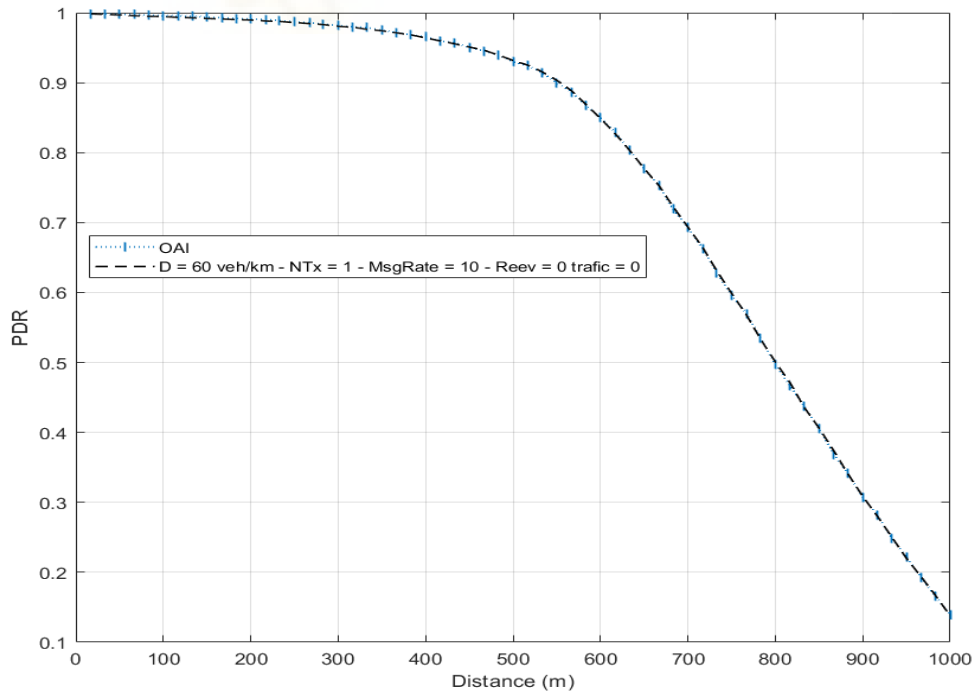

Figura 31. Comparativa de la PDR con código local en función de la distancia para una densidad de vehículo de 60 veh/km con NTx = 1

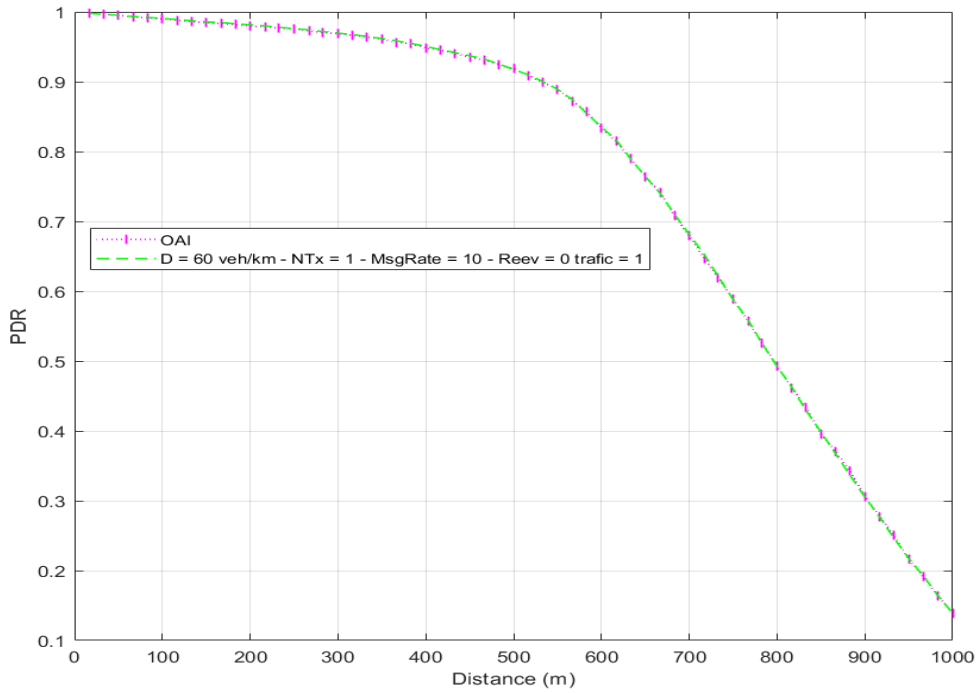

Figura 32. Comparativa de la PDR con código local en función de la distancia para densidades de vehículos de 60 veh/km y tráfico aperiódico (tasa de generación de tráfico de 10 paq/s)

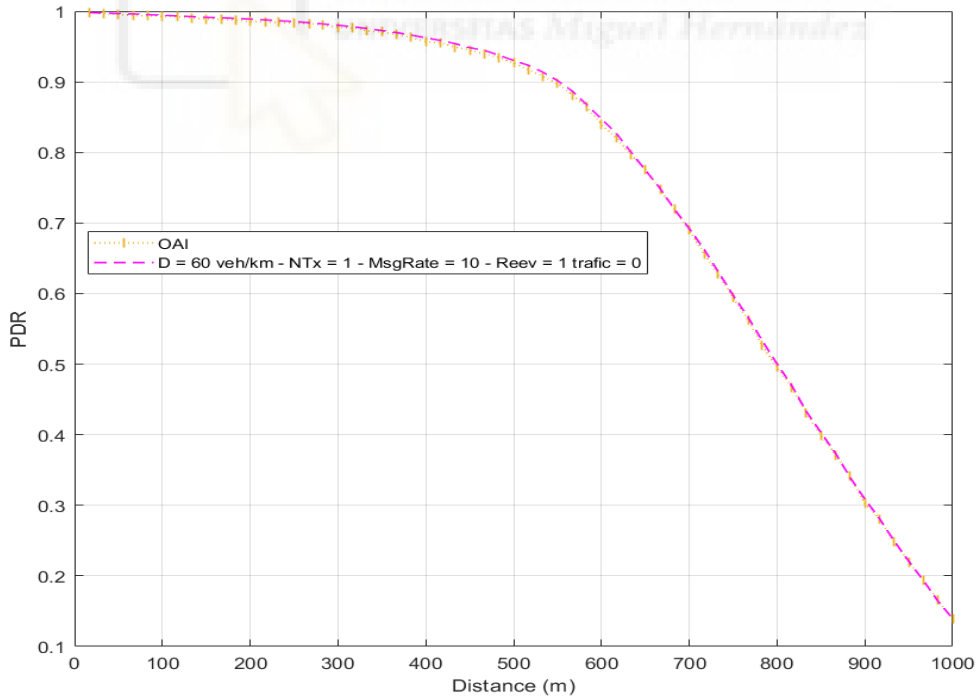

Figura 33. Comparativa de la PDR en función de la distancia con código local para densidades de vehículos de 60 veh/km cuando el mecanismo de reevaluación está activado (tasa de generación de tráfico de 10 paq/s, tráfico periódico y 1 transmisión/paq)

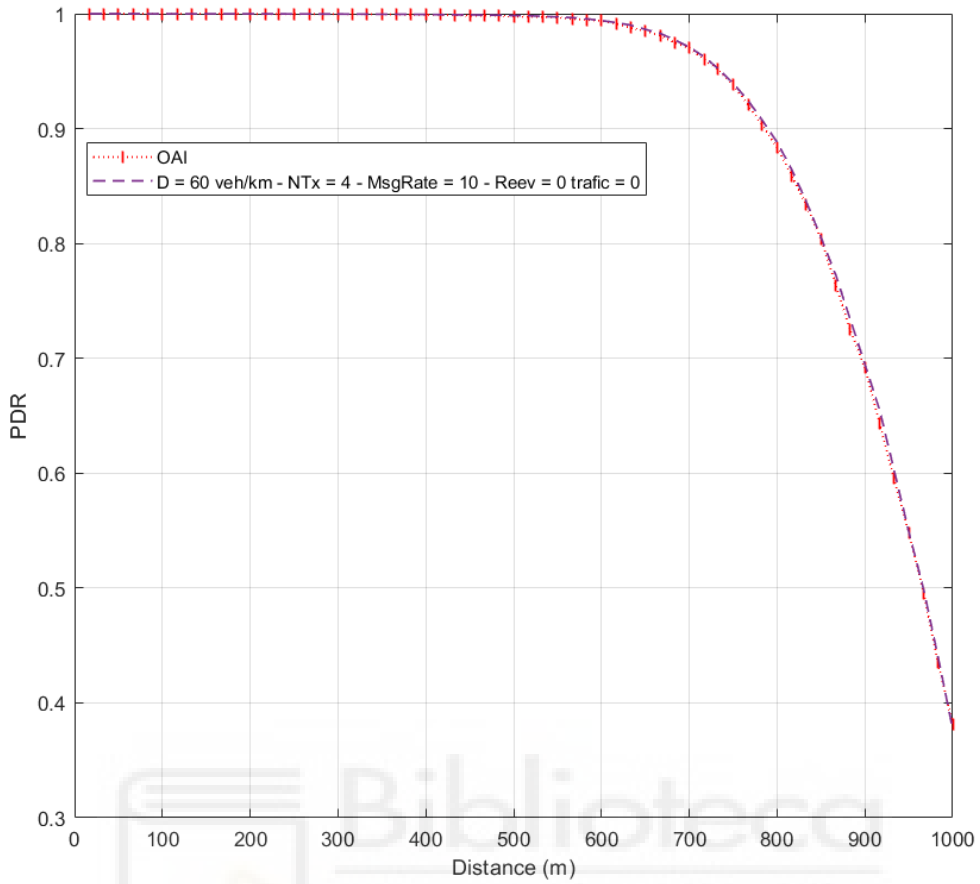

Figura 34. Comparativa de la PDR a nivel de APP en función de la distancia con código local para densidades de vehículos de 60 veh/km cuando se realizan NTx = 4 transmisiones del mismo paquete (tasa de generación de tráfico de 10 paq/s)

7. CONCLUSIONES

En este trabajo se ha implementado la capa MAC de 5G NR V2X y se ha validado y evaluado extensamente para analizar su rendimiento. La implementación se ha integrado sobre una arquitectura de emulación basada en OpenAirInterface (OAI) en la que todas las capas del sistema de comunicaciones son implementadas para su utilización en sistemas reales, excepto la capa PHY que es modelada.

Durante el desarrollo de las funcionalidades en la implementación del estándar 5G NR V2X se ha añadido nuevas características respecto al estándar LTE V2X, como la reevaluación de recursos, la posibilidad de realizar retransmisiones de un mismo paquete, soporte de tráfico aperiódico, entre muchas otras, lo que ha permitido lanzar emulaciones con una mayor variedad de configuraciones. Asimismo, se ha enmendado muchos errores derivados de la implementación del estándar LTE V2X, que causaban problemas o que dificultaban el mantenimiento y la escalabilidad del código implementado y también dificultades en la legibilidad o comprensión del mismo. También se ha realizado muchas mejoras en el código, así como un mayor nivel de detalles, abstracción y modularización, haciendo el código mucho más mantenible, escalable y con mayor precisión y dinamismo.

En el estudio realizado se ha obtenido un gran volumen de datos que, de forma automatizada, han permitido generar múltiples gráficas para el análisis del rendimiento. Una vez obtenidos todos los resultados y generado las gráficas, se ha realizado un análisis comparativo del impacto de los diferentes escenarios. En el proceso de emulación se ha añadido muchos niveles de automatización que facilitarán en tiempo y esfuerzo también en futuras emulaciones. Los resultados demuestran la validez de la implementación realizada, y el extenso estudio realizado deja patente la necesidad de estudiar en detalle el rendimiento de la tecnología para optimizar su configuración.

En futuras ampliaciones, para mejorar el rendimiento de la PDR en escenarios con un alto número de transmisiones por paquete, se podría introducir el mecanismo de congestión del tráfico. Este mecanismo permitirá una reducción en la carga del canal, a través de estrategias que adaptan el MCS, por ejemplo, para reducir el número de subcanales utilizados para transmitir un paquete, y también mediante estrategias que reduzcan el número máximo de retransmisiones, o la potencia de transmisión.

Como trabajo futuro queda también la integración de los desarrollos realizados con una implementación SDR de la capa física, para poder emplear dispositivos SDR y

transmisiones inalámbricas reales. Ello permitiría validar la implementación realizada incluyendo la capa física, y el intercambio de información con chips reales de la tecnología 5G NR V2X.

8. BIBLIOGRAFÍA

[1] Lin, Xingqin, J. G. Andrews, A. Ghosh and R. Ratasuk, "An overview of 3GPP device-to-device proximity services," *IEEE Communications Magazine, vol. 52, no. 4, pp. 40-48,* Abril 2014.

[2] Chen et al., "Vehicle-to-Everything (v2x) Services Supported by LTE-Based Systems and 5G," *IEEE Communications Standards Magazine, vol. 1, no. 2, pp. 70-76*, Julio 2017.

[3] 3GPP, "TS 38.214 NR; Physical layer procedure for data (v16.3.0, Release 16), " 3GPP, Tech. Spec., Septiembre 2020.

[4] 3GPP, "TR 37.985 Overall description of Radio Access Network (RAN) aspects for Vehicle-to-everything (V2X) based on LTE and NR (v16.0.0, Release 16), " 3GPP, Tech. Rep., Septiembre. 2020.

[5] R. Molina-Masegosa, J. Gozalvez, "System Level Evaluation of LTE-V2V Mode 4 Communications and its Distributed Scheduling", *Proceedings of the IEEE 85th Vehicular Technology Conference (VTC2017-Spring)*, 4-7 Junio 2017, Sydney (Australia).

[6] R. Molina-Masegosa, J. Gozalvez, "LTE-V for Sidelink 5G V2X Vehicular Communications: A New 5G Technology for Short-Range Vehicle-to-Everything Communications", *IEEE Vehicular Technology Magazine, vol. 12, no. 4, pp. 30-39*, Diciembre 2017.

[7] Mario H. Castañeda Garcia, Alejandro Molina-Galan, Mate Boban, Javier Gozalvez, Baldomero Coll-Perales, Taylan Şahin, Apostolos Kousaridas, "A Tutorial on 5G NR V2X Communications", *pp. 30-31*, Febrero 2021.