Examen Final: Red C-V2X modo 2 Sidelink (PC5) sobre OMNET++ y NS3 para eventos de emergencia

Martín Vinces
Facultad de Ingeniería
Universidad de Cuenca
carlosm.vinces@ucuenca.edu.ec

Erick Ramón
Facultad de Ingeniería
Universidad de Cuenca
erick.ramon@ucuenca.edu.ec

Resumen

Este trabajo presenta la implementación y evaluación de una red 5G NR V2X operando en modo Sidelink (PC5), orientada a soportar aplicaciones de seguridad crítica en Sistemas de Transporte Inteligente (ITS). El estudio se enfoca en la viabilidad técnica del sistema de Frenado de Emergencia Autónomo (AEB), una aplicación que impone requisitos estrictos en términos de comunicaciones ultra confiables de baja latencia (URLLC).

Con el fin de obtener una evaluación integral, el escenario propuesto fue implementado y analizado utilizando dos entornos de simulación a nivel sistema complementarios. Por un lado, se empleó el simulador ns-3 junto con el módulo 5G-LENA para modelar una red vehicular operando en Modo 2, donde los vehículos gestionan de forma autónoma la asignación de recursos de radio sin depender de infraestructura celular. Por otro lado, se desarrolló una implementación independiente en OMNeT++, integrando los frameworks Simu5G, Veins, INET y SUMO, lo que permitió modelar de manera conjunta la movilidad vehicular, la comunicación NR Sidelink y la lógica de aplicación asociada a la seguridad vial.

Ambos entornos de simulación incorporan dinámicas vehiculares realistas y la pila de protocolos 5G NR, posibilitando la detección proactiva de colisiones y la ejecución automática de maniobras evasivas. Los resultados obtenidos validan la robustez del estándar NR Sidelink, reportando retardos extremo a extremo del orden de pocos milisegundos y tasas de entrega de paquetes superiores al 99 %, lo que demuestra que esta tecnología cumple con los requisitos de desempeño necesarios para aplicaciones V2X críticas y escenarios de conducción autónoma cooperativa.

Index Terms

5G NR, C-V2X, Sidelink PC5, Modo 2, URLLC, AEB, ns-3, Latencia, Seguridad Vehicular, OMNET++, simu5g, inet, SUMO.

I. Introducción

La rápida evolución de los Sistemas de Transporte Inteligente (ITS) ha catalizado un cambio de paradigma fundamental en las comunicaciones vehiculares, transitando desde una conectividad básica hacia la conducción automatizada cooperativa. Si bien los estándares iniciales basados en IEEE 802.11p sentaron las bases para la comunicación V2X (*Vehicle-to-Everything*), los estrictos requisitos de las aplicaciones modernas de seguridad crítica exigen un rendimiento superior en términos de ancho de banda y latencia. El estándar 5G NR (*New Radio*), desarrollado por el 3GPP, emerge como el habilitador clave para esta transformación, ofreciendo comunicaciones ultra confiables de baja latencia (URLLC) y banda ancha móvil mejorada (eMBB). A diferencia de sus predecesores, 5G NR V2X introduce numerologías flexibles y estructuras de capa física avanzadas diseñadas específicamente para manejar la alta movilidad y las condiciones de canal variables típicas de los entornos viales. Esta tecnología no es meramente una actualización incremental, sino una infraestructura indispensable para soportar casos de uso avanzados como la prevención cooperativa de colisiones, el intercambio de datos de sensores y la conducción remota, donde los tiempos de reacción del orden de milisegundos son vitales para salvaguardar la vida humana.

Un elemento central y distintivo en la arquitectura 5G V2X es la interfaz *Sidelink* (PC5), la cual facilita la comunicación directa entre vehículos (V2V) sin la necesidad de enrutar datos a través del núcleo de la red celular (Core Network). Esta capacidad es fundamental para garantizar una operación continua y robusta en áreas con cobertura celular deficiente o inexistente, así como para reducir drásticamente la latencia de transmisión. Específicamente, el estándar NR V2X define la operación en Modo 2, donde los vehículos seleccionan autónomamente sus recursos de transmisión a partir de un *pool* preconfigurado mediante mecanismos inteligentes de *sensing* y reserva, eliminando la dependencia de un planificador central o estación base. Este proyecto se centra en evaluar rigurosamente este modo autónomo dentro de un contexto de seguridad crítica: la implementación de un sistema de Frenado de Emergencia Autónomo (AEB). Aprovechando la baja latencia y alta confiabilidad de 5G NR Sidelink, los vehículos pueden intercambiar información de trayectoria y estado cinemático en tiempo real, permitiendo la detección proactiva de colisiones inminentes y la ejecución automatizada de maniobras evasivas antes de que el conductor humano pueda reaccionar.

Para validar el rendimiento teórico de estos protocolos de comunicación avanzados en escenarios realistas antes de su despliegue físico, las herramientas de simulación de redes desempeñan un papel indispensable en la ingeniería de telecomunicaciones. Este informe presenta una implementación y análisis exhaustivos de una red 5G NR V2X Sidelink utilizando el simulador de eventos discretos ns-3, evaluando específicamente su capacidad para soportar los estrictos requisitos de calidad de servicio (QoS) de una aplicación AEB. Además, para proporcionar una perspectiva más amplia sobre la fidelidad de simulación y los enfoques arquitectónicos, este estudio se complementa con una implementación paralela desarrollada en el entorno de simulación OMNeT++. Este enfoque de doble plataforma permite una verificación robusta de los algoritmos de seguridad propuestos y proporciona información valiosa sobre el comportamiento del estándar NR V2X bajo diferentes núcleos de simulación y modelos de movilidad. Las secciones subsiguientes detallan el desarrollo técnico, la configuración de parámetros y el análisis comparativo de los resultados obtenidos, demostrando la viabilidad técnica de 5G NR como la columna vertebral para los sistemas de seguridad vehicular de próxima generación.

II. Fundamento Teórico

El despliegue de sistemas de transporte inteligente (ITS) requiere tecnologías de comunicación capaces de garantizar baja latencia y alta confiabilidad para aplicaciones críticas de seguridad, como el Frenado de Emergencia Autónomo (AEB). En este contexto, la tecnología 5G NR (*New Radio*) y el estándar C-V2X (*Cellular Vehicle-to-Everything*) representan la evolución fundamental frente a sus predecesores basados en 802.11p.

II-A. Redes 5G y Comunicaciones Ultra-Confiables (URLLC)

La quinta generación de redes móviles (5G) no solo ofrece un aumento en el ancho de banda, sino que introduce un cambio de paradigma mediante la especificación de servicios URLLC (*Ultra-Reliable Low Latency Communications*). Para aplicaciones vehiculares, el 3GPP (*3rd Generation Partnership Project*) ha definido en sus *Releases* 16 y 17 especificaciones que permiten latencias del orden de milisegundos y confiabilidades superiores al 99.999 % [1].

La arquitectura 5G se basa en la flexibilidad del espectro mediante el uso de diferentes numerologías (μ), que definen el espaciado entre subportadoras (SCS - *Subcarrier Spacing*). Un SCS mayor (ej. 60 kHz o 120 kHz) reduce la duración del símbolo OFDM y, consecuentemente, la latencia de la trama, lo cual es crítico para la seguridad vial [2].

II-B. Arquitectura de Protocolos: Red 5G vs. Sidelink

El ecosistema C-V2X se sustenta en dos interfaces de radio distintas. Es crucial diferenciar entre la comunicación con la red (V2N) y la comunicación directa entre vehículos (V2V).

La Figura 1 ilustra la pila de protocolos de la **Red 5G convencional (Interfaz Uu)**. En esta arquitectura, el UE (*User Equipment*) se conecta a la estación base (gNB) y al núcleo de la red (representado por el AMF - *Access and Mobility Management Function*). Esta pila es utilizada para el intercambio de señalización de control (RRC/NAS) y tráfico de datos hacia internet o servidores en la nube.

Por el contrario, para la aplicación de prevención de colisiones simulada en este proyecto, se utiliza la interfaz **PC5 (Sidelink)**. A diferencia de la Figura 1, el tráfico Sidelink fluye horizontalmente entre los UEs a través de las capas PHY, MAC, RLC y PDCP, sin intervención activa del gNB ni del núcleo de la red durante la transmisión de datos.

II-C. Modos de Operación en C-V2X

El estándar define distintos modos de asignación de recursos para la comunicación directa. Aunque la terminología evoluciona entre LTE y NR, los conceptos fundamentales de gestión centralizada vs. autónoma se mantienen.

- **Modo 1 (NR) / Modo 3 (LTE):** La red celular (gNB) utiliza la interfaz Uu (Figura 1) para planificar y asignar los recursos que los vehículos utilizarán en el Sidelink. Requiere cobertura de red.
- **Modo 2 (NR) / Modo 4 (LTE):** Los vehículos seleccionan sus propios recursos de radio de un *Resource Pool* preconfigurado de manera autónoma, utilizando la interfaz PC5. Este es el modo utilizado en la presente simulación, ya que es el único capaz de operar en escenarios fuera de cobertura (*out-of-coverage*) y es el más robusto para seguridad [1].

La Figura 2 ilustra la diferencia topológica entre estos modos. En el Modo 2 (equivalente funcional al Modo 4 de LTE mostrado), la comunicación es puramente *ad-hoc*.

II-D. Estructura de Recursos Físicos en NR Sidelink

La gestión del espectro en NR Sidelink es compleja debido a la necesidad de multiplexar datos y control. La unidad básica de asignación no es el Bloque de Recursos Físicos (PRB) individual, sino el **Subcanal**.

- **Subcanales:** Un subcanal es un grupo de PRBs contiguos en frecuencia. El tamaño del subcanal es configurable y determina la granularidad con la que se reservan los recursos.
- **Canales Físicos:**

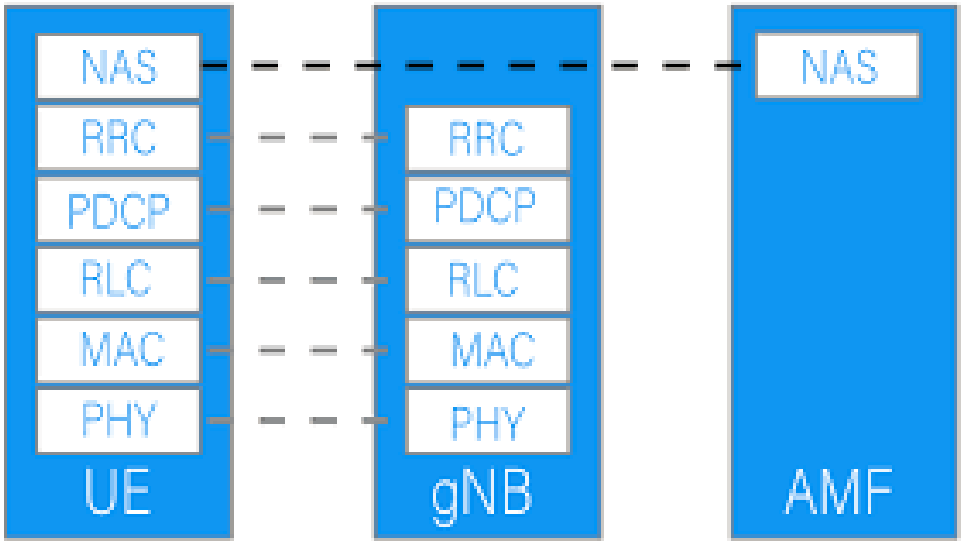

Figura 1: Pila de protocolos de la red 5G NR (Interfaz Uu). Se observa la conexión vertical entre el UE, el gNB (Radio Access Network) y el AMF (Core Network), utilizada para comunicaciones V2N o configuración del Modo 1 [4].

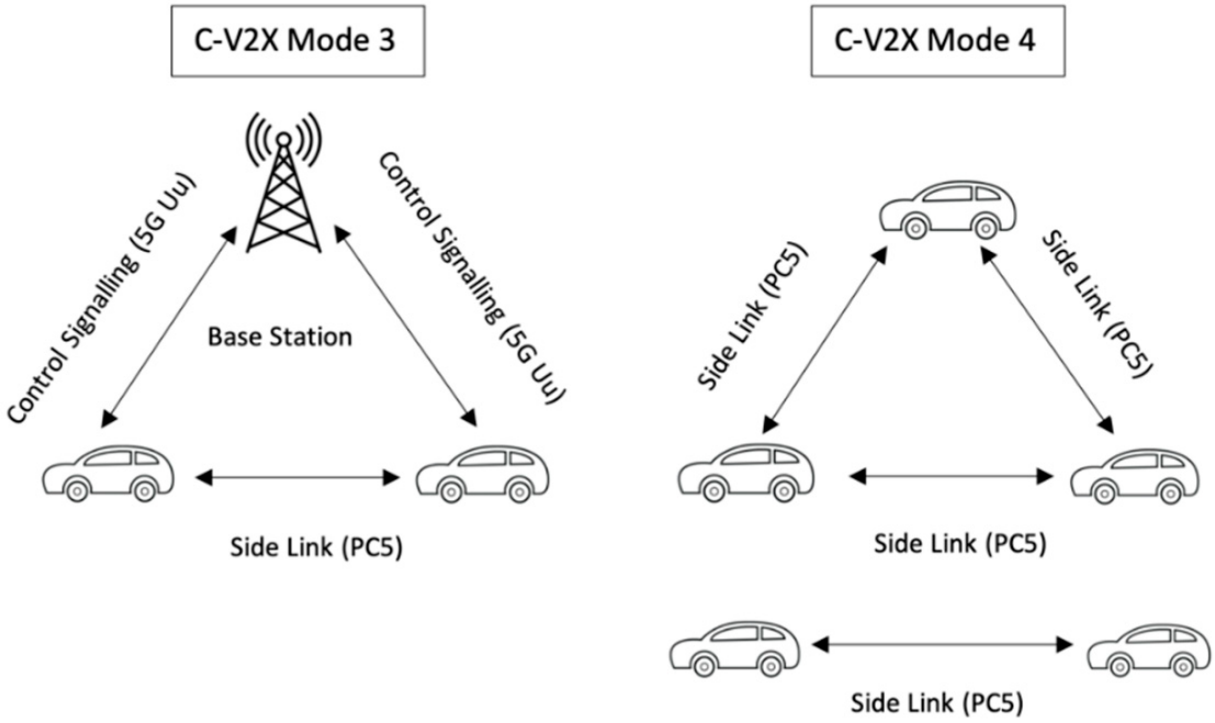

Figura 2: Comparación de modos de operación. A la izquierda, asignación controlada por la estación base (Modo 1/3) usando la red 5G. A la derecha, asignación autónoma mediante PC5 (Modo 2/4), utilizada en este desarrollo [4].

- **PSCCH (Physical Sidelink Control Channel):** Transporta la Información de Control de Sidelink (SCI - *Stage 1*), necesaria para que los receptores decodifiquen los datos.
- **PSSCH (Physical Sidelink Shared Channel):** Transporta los datos del usuario (TB - *Transport Block*) y la segunda parte del control (SCI - *Stage 2*).
- **PSFCH (Physical Sidelink Feedback Channel):** Canal físico para retroalimentación HARQ (ACK/NACK), intro-

ducido en NR [3].

La Figura 3 detalla la jerarquía de recursos, desde el símbolo OFDM y la subportadora hasta la conformación del subcanal.

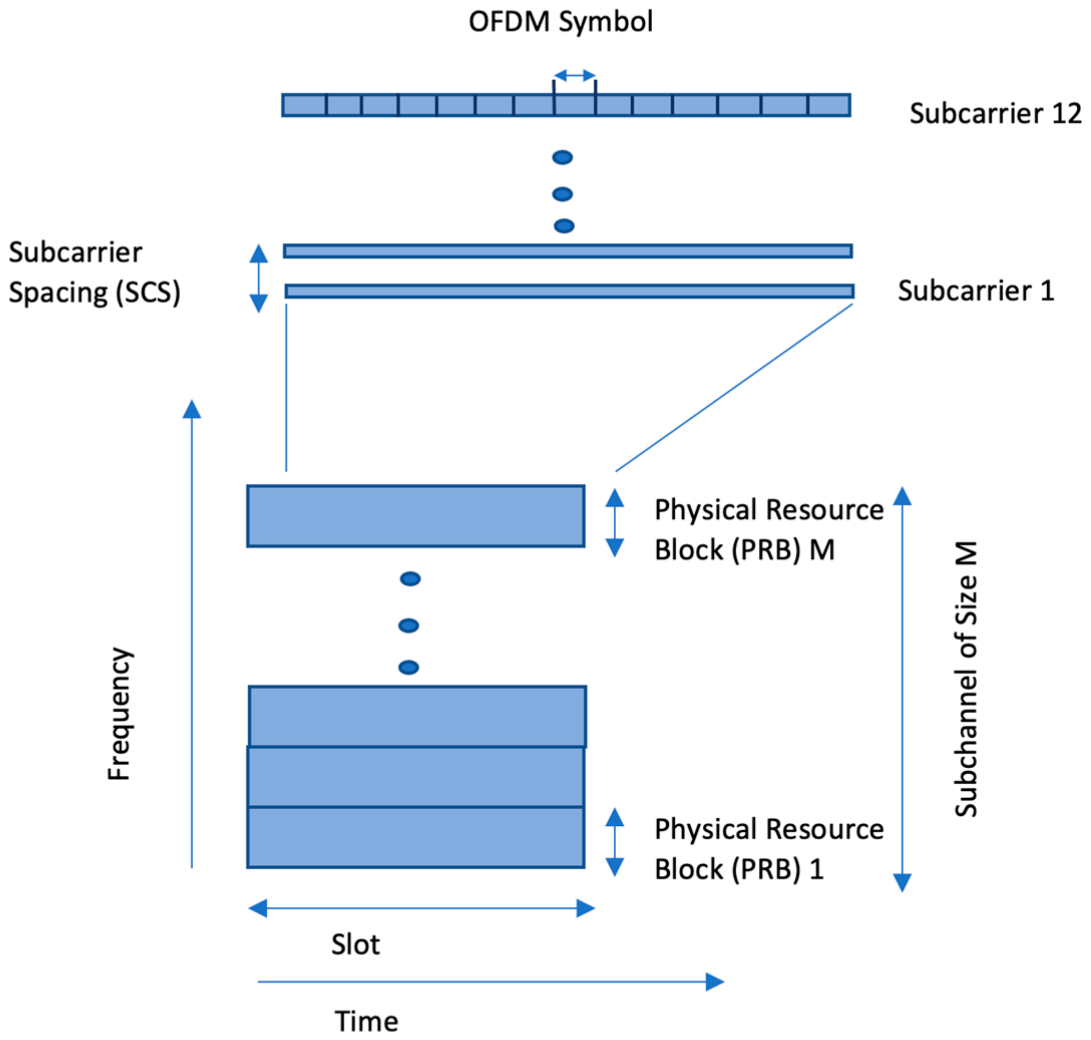

Figura 3: Jerarquía de recursos en NR Sidelink: Símbolos OFDM, Subportadoras, Bloques de Recursos Físicos (PRB) y agrupación en Subcanales [4].

II-E. Asignación de Recursos: Resource Pool y Sensing

En el Modo 2, los vehículos operan sobre un *Resource Pool* (Figura 4), que es un conjunto de recursos tiempo-frecuencia permitidos para Sidelink. Para evitar colisiones de paquetes sin una estación base central, se emplea el mecanismo de *Sensing* y *Resource Selection*:

1. **Sensing (Escucha):** El vehículo monitorea continuamente el canal, decodificando los SCI de otros vehículos para saber qué recursos están ocupados o reservados para el futuro. 2. **Ventana de Selección:** Cuando se genera un paquete (ej. alerta AEB), el vehículo selecciona un recurso libre dentro de una ventana de selección futura, basándose en la información recolectada durante el sensing.

Este mecanismo es crucial para garantizar que, incluso en alta densidad vehicular, los mensajes de seguridad críticos tengan una alta probabilidad de entrega (PDR) y baja latencia.

III. Desarrollo

III-A. Implementación en NS-3

Esta sección detalla el procedimiento técnico y la arquitectura de software utilizada para la implementación de la simulación de una red C-V2X (*Cellular Vehicle-to-Everything*) en modo *Sidelink* (PC5) bajo el estándar 5G NR (*New Radio*). El desarrollo

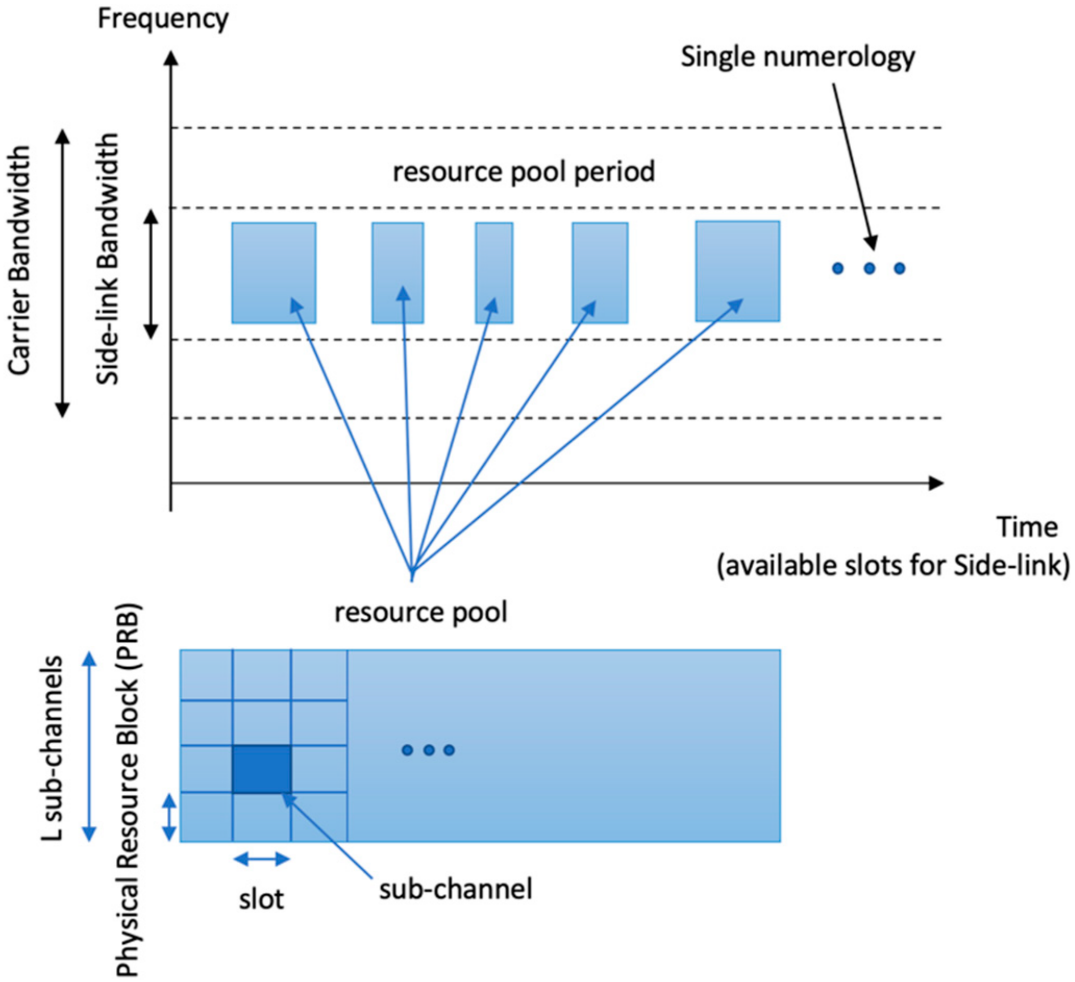

Figura 4: Estructura del Resource Pool en el dominio del tiempo y la frecuencia. Se observa cómo los subcanales se distribuyen a lo largo de los slots disponibles para Sidelink [4].

se centra en la evaluación de una aplicación de seguridad crítica, específicamente el Frenado de Emergencia Autónomo (AEB), validando tanto la lógica de control vehicular como las métricas de desempeño de la red de comunicaciones.

La simulación se ha desarrollado sobre el simulador de redes discretas ns-3, utilizando el módulo 5G-LENA (NR module). Este módulo ha sido extendido específicamente para soportar las funcionalidades de NR V2X, permitiendo la simulación de comunicaciones directas entre vehículos sin necesidad de infraestructura celular activa, conocido como operación fuera de cobertura (*out-of-coverage*) o Modo 2.

A continuación, se describen los bloques funcionales que componen el script de simulación (`v2v-collision-avoidance.cc`), explicando la lógica operativa y la configuración de los parámetros del estándar 3GPP.

III-A1. Arquitectura de la Simulación y Configuración del Escenario: El entorno de simulación se inicializa definiendo los parámetros fundamentales de la capa física y la topología de la red.

- **Gestión de Nodos y Movilidad:** Se utiliza el contenedor `NodeContainer` para instanciar los UE (*User Equipment*) vehiculares. La movilidad se gestiona mediante el modelo `ConstantVelocityMobilityModel`, lo que permite actualizar dinámicamente el vector de velocidad y posición de cada vehículo. En el escenario planteado, se configuran dos vehículos: un líder (Vehicle 1) y un seguidor (Vehicle 0), separados inicialmente por una distancia parametrizable (por defecto 20 metros) y con velocidades relativas que propician un escenario de alcance.
- **Configuración del Espectro y NR Stack:** Se instancia el `NrHelper` junto con la configuración de las partes de ancho de banda (BWP - *Bandwidth Parts*). Se ha seleccionado una frecuencia central de 5.89 GHz, correspondiente a la banda ITS (*Intelligent Transport Systems*). Se utiliza una numerología $\mu = 2$ (Subcarrier Spacing de 60 kHz), optimizada para soportar la latencia requerida en escenarios vehiculares dinámicos.

III-A2. Configuración del Canal Sidelink (PC5): La comunicación V2X se basa en el enlace lateral o *Sidelink*, gestionado en el código mediante el `NrSlHelper`. A diferencia de las comunicaciones celulares tradicionales (Uplink/Downlink), el Sidelink en NR V2X requiere una pre-configuración exhaustiva para operar en Modo 2, donde los vehículos seleccionan autónomamente sus recursos de transmisión.

- **Recursos de Radio (Resource Pool):** Se define un *Resource Pool* mediante `LteRrcSap::SlResourcePoolNr`. Esto incluye la configuración del mapa de bits (*bitmap*) que dicta qué ranuras de tiempo (*slots*) están disponibles para la transmisión Sidelink, así como la definición de subcanales en el dominio de la frecuencia.
- **Pre-configuración RRC:** Dado que se simula un entorno sin cobertura de gNB, es necesario inyectar una pre-configuración RRC (*Radio Resource Control*) en los UEs. Esto se realiza mediante la estructura `SidelinkPreconfigNr`, la cual define los parámetros para la selección de recursos, incluyendo los patrones TDD y la configuración de los canales físicos PSCCH (*Physical Sidelink Control Channel*) y PSSCH (*Physical Sidelink Shared Channel*).

III-A3. Pila de Protocolos IP y Flujos de Tráfico: Para habilitar la comunicación de datos, se instala la pila de internet (`InternetStackHelper`) en los vehículos.

- **Direccionamiento:** Se asignan direcciones IPv4 a la interfaz de red `NrUeNetDevice`. Es crucial destacar el uso de una dirección de grupo (*Multicast/Groupcast*) para emular la naturaleza de difusión de los mensajes de seguridad (CAM/BSM).
- **Gestión de Portadores (Bearers):** Se configuran los *Traffic Flow Templates* (TFT) mediante `LteSlTft` para mapear el tráfico generado por las aplicaciones hacia los portadores Sidelink adecuados, definiendo prioridades y el ID de destino de capa 2 (L2 ID).
- **Generación de Tráfico:** Se emplea la aplicación `OnOffApplication` con protocolo UDP para generar un flujo constante de paquetes de 300 bytes a una tasa de 24 kbps. Esto simula la carga útil de los mensajes de estado del vehículo enviados periódicamente.

III-A4. Lógica de Aplicación: AEB con Recuperación: El núcleo lógico de la seguridad reside en la clase personalizada `V2vAebApp`, que hereda de la clase base `Application` de ns-3. Esta clase implementa un bucle de control con un intervalo de actualización de 0.1 segundos.

- **Cálculo de TTC:** La aplicación accede a los modelos de movilidad para obtener la posición y velocidad instantánea de los vehículos vecinos. Con estos datos, calcula la distancia euclidiana y la velocidad relativa para determinar el Tiempo Hasta la Colisión (TTC - *Time To Collision*).
- **Máquina de Estados:**
  1. **Monitoreo:** Si $TTC < 1{,}5$ segundos, se activa el estado de emergencia.
  2. **Frenado:** Se fuerza el vector de velocidad del vehículo a $(0, 0, 0)$, deteniendo la unidad y registrando el evento.
  3. **Espera y Reanudación:** Tras el frenado, se programa un evento (`Simulator::Schedule`) con un temporizador de 20 segundos. Al expirar, se restaura el vector de velocidad original, simulando una recuperación de la marcha tras la disipación del peligro.

III-A5. Sistema de Métricas y Trazas: Para validar el desempeño de la red de manera profesional, se implementó un sistema de extracción de métricas basado en trazas (*Trace-based metrics*) en lugar de utilizar el módulo `FlowMonitor`, debido a las particularidades del enrutamiento en Sidelink.

Se conectaron funciones de *callback* a las trazas físicas `TxWithSeqTsSize` y `RxWithSeqTsSize`. Estas funciones realizan lo siguiente:

- **Cálculo de Latencia E2E:** Se extrae la marca de tiempo (*timestamp*) de creación del paquete desde su cabecera y se compara con el tiempo actual de simulación (`Simulator::Now()`) en el momento de la recepción.
- **Registro CSV:** Se exportan datos granulares (Tiempo, Latencia, Tamaño de Paquete, Intervalo de Llegada) a archivos CSV (`v2x_metrics.csv`) para su posterior procesamiento y visualización.
- **Análisis de Pérdidas:** Se comparan los contadores globales de transmisión y recepción para determinar el *Packet Delivery Ratio* (PDR).

III-A6. Ejecución de la Simulación: Para ejecutar la simulación, es necesario compilar el proyecto dentro del entorno `ns-3-dev` utilizando el sistema de construcción `ns3` (basado en CMake/Ninja). Los comandos utilizados en la terminal para la configuración y ejecución son los siguientes:

```
# Configuración del entorno de construcción (solo la primera vez)
./ns3 configure --enable-examples --enable-tests

# Compilación del script de simulación
./ns3 build

# Ejecución del escenario V2V con Sidelink
```

```
./ns3 run scratch/v2v-collision-avoidance
```

Tras la ejecución exitosa, el simulador genera los archivos de trazas (`.csv`, `.xml` para NetAnim) y muestra en la salida estándar (consola) un reporte resumido con las métricas de confiabilidad (PDR), eficiencia (Throughput) y latencia media, validando si se cumplen los requisitos de seguridad V2X.

III-B. Implementación en OMNET++

III-B1. Entorno de simulación y herramientas empleadas: La implementación del escenario de comunicación vehicular se desarrolló utilizando un entorno de simulación basado en OMNeT++, el cual permite la modelación modular de redes de comunicación y la integración con herramientas externas de movilidad. El objetivo principal de este entorno fue evaluar el comportamiento de la comunicación NR Sidelink en un escenario vehicular dinámico, manteniendo un nivel de abstracción adecuado para estudios a nivel de sistema.

Como núcleo de la simulación se empleó OMNeT++, en conjunto con el framework INET, el cual provee los modelos necesarios para las capas de red y transporte basadas en IP. INET fue utilizado para la gestión de direcciones IPv4, el enrutamiento básico y el soporte de aplicaciones basadas en UDP, permitiendo que los mensajes de alerta se transmitan mediante tráfico multicast sobre la interfaz celular.

Para la modelación de la red celular 5G se utilizó Simu5G, un framework orientado a simulaciones a nivel sistema de tecnologías LTE y NR. Simu5G proporciona los módulos necesarios para representar nodos de usuario, estaciones base y la interfaz de acceso celular, así como modelos abstractos de las capas MAC y PHY. En este proyecto, Simu5G fue el encargado de gestionar la comunicación NR Sidelink entre los vehículos, incluyendo la selección de parámetros de transmisión, el cálculo de tasas efectivas y la estimación de errores de enlace.

La movilidad vehicular se integró mediante el framework Veins, el cual actúa como un puente entre OMNeT++ y el simulador de tráfico SUMO. A través de la interfaz TraCI, Veins permite la creación y destrucción dinámica de nodos vehiculares en OMNeT++, así como la actualización continua de su posición, velocidad y estado. Esta integración posibilita que el comportamiento de la red de comunicaciones esté directamente influenciado por la dinámica del tráfico vehicular.

SUMO fue utilizado para definir el escenario de movilidad, incluyendo la topología vial, las rutas de los vehículos y las características visuales del entorno. Los archivos de configuración asociados a SUMO permiten describir de manera detallada el mapa del escenario, los flujos de tráfico y los parámetros de simulación temporal, los cuales son posteriormente interpretados por Veins durante la ejecución conjunta con OMNeT++.

Todo el entorno de simulación fue ejecutado sobre un sistema operativo GNU/Linux, asegurando compatibilidad con las herramientas utilizadas y facilitando la compilación de los distintos frameworks involucrados. La combinación de OMNeT++, INET, Simu5G, Veins y SUMO permitió construir un entorno de simulación coherente, extensible y adecuado para el análisis de comunicaciones vehiculares basadas en NR Sidelink.

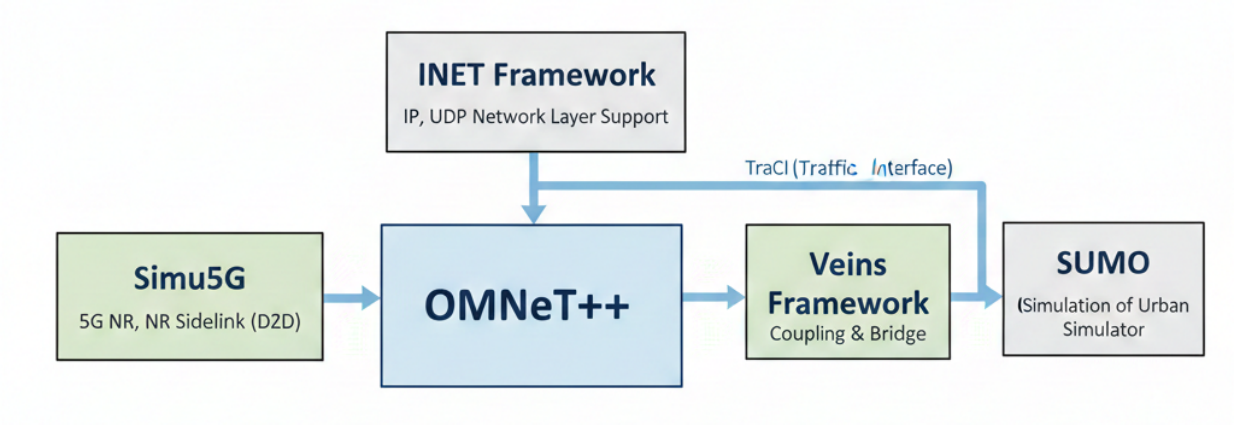

Figura 5: Ecosistema de herramientas empleadas para la simulación vehicular, mostrando la integración entre OMNeT++, Simu5G, INET, Veins y SUMO.

III-B2. Integración OMNeT++–SUMO mediante Veins: La integración entre la simulación de red y la simulación de movilidad se realizó mediante el framework Veins, el cual permite acoplar OMNeT++ con el simulador de tráfico SUMO a través de la interfaz TraCI (Traffic Control Interface). Esta integración es fundamental para escenarios vehiculares, ya que posibilita que el comportamiento de la red de comunicaciones esté directamente condicionado por la dinámica realista del tráfico.

En este proyecto, SUMO se utilizó para definir el escenario de movilidad vehicular, incluyendo la topología vial, las rutas de circulación y las características geométricas del entorno. Dicho escenario se describe mediante un conjunto de archivos de configuración en formato XML, los cuales especifican de manera separada la red vial, los flujos de tráfico, los elementos gráficos y los parámetros generales de simulación. Estos archivos son cargados por SUMO al inicio de la simulación y controlan el movimiento de los vehículos a lo largo del tiempo.

Veins actúa como un intermediario entre SUMO y OMNeT++, manteniendo una comunicación continua mediante TraCI. A través de esta interfaz, Veins obtiene información en tiempo real sobre la posición, velocidad y estado de cada vehículo simulado en SUMO, y la traduce en actualizaciones de movilidad para los nodos correspondientes en OMNeT++. De esta manera, cada vehículo en SUMO tiene una representación directa como un nodo de red dentro del simulador de comunicaciones.

La creación y destrucción de nodos vehiculares en OMNeT++ se gestiona dinámicamente a través del módulo `VeinsInetManager`. Este componente se encarga de instanciar nodos del tipo definido para los vehículos, asociarlos con la movilidad proporcionada por SUMO y eliminarlos cuando el vehículo abandona el escenario. Este mecanismo permite simular escenarios con tráfico variable sin necesidad de definir de forma estática todos los nodos de red desde el inicio.

La sincronización temporal entre OMNeT++ y SUMO garantiza que los eventos de movilidad y los eventos de red se desarrollen de forma coherente. Cada avance en el tiempo de simulación implica una actualización del estado vehicular y, simultáneamente, la evaluación del comportamiento de la red de comunicaciones. Como resultado, fenómenos como la variación de distancias entre nodos, cambios de conectividad o aparición de nuevos enlaces influyen directamente en la transmisión de mensajes de alerta.

La utilización conjunta de OMNeT++, Veins y SUMO permitió construir un escenario vehicular dinámico y realista, en el cual la movilidad no es un elemento externo o simplificado, sino una parte integral del proceso de simulación. Esta integración constituye la base sobre la cual se desarrollan las capas superiores de la red y la lógica de comunicación NR Sidelink analizada en las siguientes subsecciones.

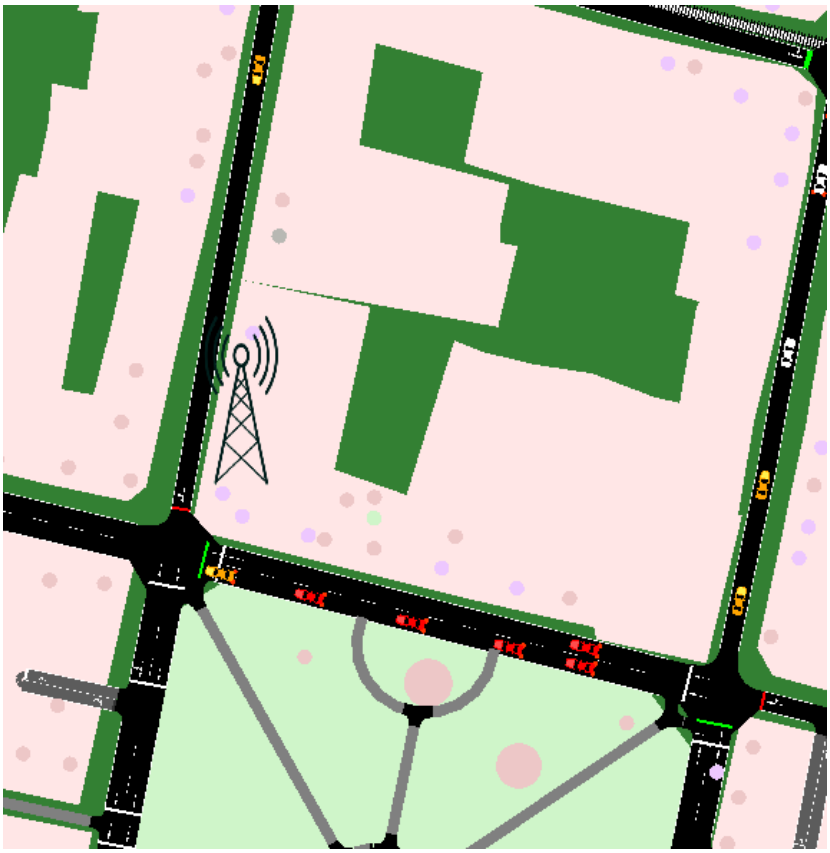

Figura 6: Escenario de movilidad vehicular definido en SUMO y sincronizado con OMNeT++ mediante la interfaz TraCI.

III-B3. Definición del escenario de red en OMNeT++: El escenario de red fue definido en OMNeT++ mediante archivos en lenguaje NED, los cuales permiten describir de forma jerárquica la composición de los nodos, sus interconexiones y los módulos funcionales que los conforman. Esta definición constituye la base estructural sobre la cual se integran la movilidad vehicular y la comunicación NR Sidelink.

El archivo principal del escenario describe la red completa de simulación e incluye los elementos necesarios para el funcionamiento de la red celular y la gestión de los nodos móviles. En este archivo se instancian los nodos vehiculares, una estación base NR, un nodo de tipo RSU y los módulos de soporte requeridos por Simu5G, tales como el binder, el control del canal y los mecanismos de configuración de red IP. La inclusión de estos módulos permite que los nodos compartan información global del estado del enlace radio y que la simulación mantenga coherencia a nivel de sistema.

Los vehículos son modelados como nodos de usuario NR mediante un módulo específico que extiende la funcionalidad básica de un UE celular. Cada vehículo incorpora una interfaz celular NR, una pila de protocolos de red basada en INET y un conjunto de aplicaciones configurables. Estos nodos no definen movilidad propia, sino que reciben su información de posición y velocidad desde Veins, lo que garantiza su sincronización con el escenario de tráfico definido en SUMO.

Adicionalmente, se incluyó un nodo de tipo RSU modelado como un nodo NR estático. Aunque desde el punto de vista de la red celular este nodo comparte características con un UE, su rol funcional dentro del escenario es diferente, ya que actúa como punto intermedio para la retransmisión de mensajes de alerta. Este enfoque permite evaluar mecanismos de diseminación multi-salto sin introducir infraestructura adicional más allá de la comunicación sidelink.

La estación base NR se incluye en el escenario con el fin de proporcionar el contexto celular requerido por Simu5G, aun cuando la comunicación principal entre vehículos se realiza de forma directa mediante NR Sidelink. Su presencia permite inicializar correctamente los identificadores de celda y los parámetros asociados al acceso radio, sin intervenir activamente en la transmisión de los mensajes de alerta.

La asignación de direcciones IP y la configuración básica de la red se realizan mediante los módulos de configuración automática proporcionados por INET. De esta forma, los nodos vehiculares y el RSU pueden intercambiar tráfico IP sin necesidad de definir rutas de manera manual, facilitando el uso de aplicaciones basadas en UDP sobre la interfaz celular.

En conjunto, la definición del escenario de red en OMNeT++ establece una estructura modular y flexible, capaz de soportar la integración con la movilidad vehicular y la comunicación NR Sidelink. Esta estructura sirve como soporte para la configuración detallada del enlace celular y la implementación de la lógica de aplicaciones descritas en las subsecciones siguientes.

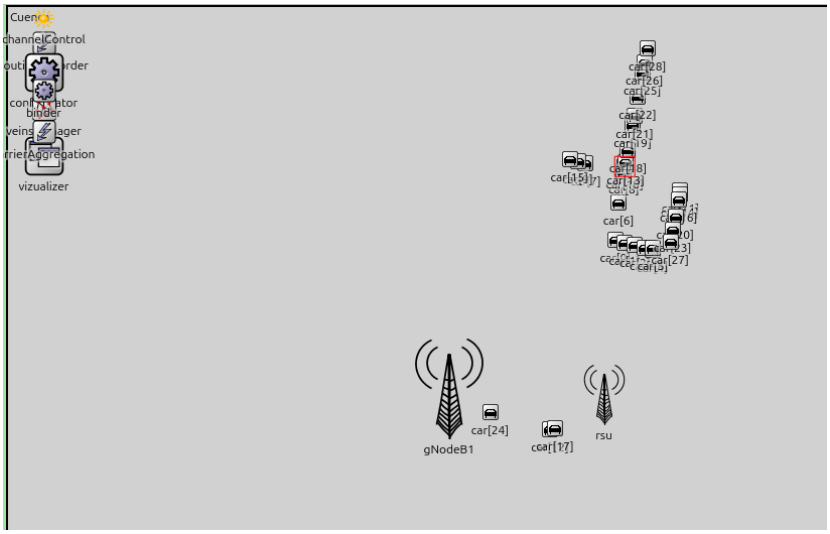

Figura 7: Estructura del escenario de red definida en OMNeT++, mostrando los nodos vehiculares, el RSU y los módulos de soporte del framework Simu5G.

III-B4. Configuración del enlace NR Sidelink: La configuración del enlace de comunicación NR Sidelink se realizó principalmente a través del archivo `omnetpp.ini`, el cual permite definir parámetros globales y específicos para los nodos de la simulación. En este archivo se establecieron las condiciones bajo las cuales los nodos vehiculares operan en modo de comunicación directa, sin intervención del núcleo de red ni de la estación base para la transmisión de mensajes de alerta.

Uno de los primeros aspectos configurados corresponde a la activación del modo de comunicación directa entre nodos. Para ello, se definió que los nodos vehiculares inicien la simulación con el modo D2D habilitado, lo que permite que los paquetes sean transmitidos directamente a través del enlace sidelink desde el inicio de la simulación. Este comportamiento se configura de manera explícita en el archivo de inicialización, como se muestra a continuación.

```
*.car[*].cellularNic.d2dInitialMode = true
```

Listing 1: Activación del modo D2D en los nodos vehiculares

La selección del modo de adaptación de enlace se realizó indicando que la transmisión se lleve a cabo bajo un esquema D2D con parámetros preconfigurados. En particular, se deshabilitó la adaptación dinámica de modulación y codificación basada en retroalimentación del canal, optando por un esquema fijo que prioriza la robustez y la reproducibilidad del escenario de simulación.

```
**.amcMode = D2D
**.usePreconfiguredTxParams = true
**.enableD2DCqiReporting = false
**.d2dCqi = 7
```

Listing 2: Configuración del esquema de transmisión preconfigurado para NR Sidelink

En esta configuración, el valor fijo de CQI establece de manera indirecta la modulación y la tasa de codificación utilizadas durante la transmisión, según las tablas internas de modulación y codificación del framework Simu5G. Al deshabilitar el reporte dinámico de CQI, los nodos no ajustan sus parámetros de transmisión en función de variaciones instantáneas del canal, lo cual resulta adecuado para el estudio de diseminación de mensajes de seguridad, donde la confiabilidad tiene mayor relevancia que la eficiencia espectral.

Adicionalmente, se configuraron los parámetros asociados a la capa física, en particular la potencia de transmisión utilizada por los nodos vehiculares durante la comunicación sidelink. Este parámetro influye directamente en el alcance de la comunicación y en la relación señal a interferencia y ruido percibida por los nodos receptores.

```
*.car[*].cellularNic.phy.d2dTxPower = 20
*.rsu.cellularNic.phy.d2dTxPower = 30
```

Listing 3: Configuración de la potencia de transmisión para NR Sidelink

Finalmente, se ajustaron parámetros globales del modelo físico abstracto proporcionado por Simu5G, los cuales influyen en la tasa efectiva de transmisión y en la probabilidad de error del enlace. Estos parámetros permiten calibrar el comportamiento estadístico del canal sin necesidad de modelar explícitamente los procesos físicos de modulación y demodulación.

```
*.binder.maxDataRatePerRb = 1.16Mbps
*.binder.blerShift = 5
```

Listing 4: Parámetros globales del modelo físico abstracto

En conjunto, la configuración del enlace NR Sidelink establece un entorno de comunicación directa autónoma entre los nodos vehiculares, caracterizado por el uso de parámetros de transmisión fijos y un modelo físico de tipo estadístico. Este enfoque permite analizar el comportamiento del sistema a nivel de red y aplicación, manteniendo un balance adecuado entre realismo y complejidad computacional.

III-B5. Implementación de la lógica de aplicación para la diseminación de alertas: La diseminación de mensajes de alerta en el escenario vehicular se implementó a nivel de aplicación mediante módulos desarrollados en `C++`, los cuales se ejecutan sobre la pila de protocolos proporcionada por INET. Esta lógica de aplicación se diseñó con el objetivo de representar un mecanismo simple y reproducible de difusión de eventos críticos entre nodos vehiculares utilizando comunicación NR Sidelink.

Para este propósito, se implementaron tres tipos de aplicaciones: un emisor de alertas, un receptor de alertas y un nodo intermedio encargado de la retransmisión de mensajes. Cada una de estas aplicaciones se encapsula en un módulo independiente, lo que facilita su reutilización y su asignación selectiva a los distintos nodos del escenario.

El módulo `AlertSender` se encarga de generar y transmitir mensajes de alerta cuando ocurre un evento específico dentro del escenario. En la simulación, este evento se asocia a la detención abrupta de un vehículo, lo cual representa una situación de riesgo que debe ser comunicada al resto de los nodos. La aplicación inicia la transmisión de mensajes luego de un tiempo de arranque definido y envía paquetes de forma periódica durante un intervalo determinado.

La comunicación entre nodos se realiza utilizando sockets UDP y direccionamiento multicast, lo que permite que un único mensaje sea recibido por todos los nodos suscritos al grupo de difusión. El uso de multicast simplifica la lógica de la aplicación y resulta adecuado para escenarios de difusión de información de seguridad. La inicialización del socket y la configuración del grupo multicast se realizan explícitamente en la aplicación emisora, como se muestra en el siguiente fragmento de código.

```
socket.setOutputGate(gate(socketOut));
socket.bind(localPort);
socket.setMulticastOutputInterface(multicastInterfaceId);
socket.joinMulticastGroup(multicastAddress);
```

Listing 5: Inicialización del socket UDP multicast en la aplicación emisora

Cada mensaje de alerta encapsula información relevante asociada al evento, incluyendo un identificador de secuencia, una marca temporal y la posición del vehículo emisor. Esta información se almacena en una estructura de paquete personalizada, la cual se define mediante archivos de cabecera y se inserta en el paquete de red antes de su transmisión. Este diseño permite que los nodos receptores puedan identificar el origen del mensaje y evaluar su pertinencia en función del contexto espacial y temporal.

Los nodos vehiculares que actúan como receptores ejecutan la aplicación `AlertReceiver`, la cual permanece a la espera de mensajes multicast durante toda la simulación. A diferencia de un receptor pasivo, esta aplicación implementa una lógica de decisión basada en el tipo de mensaje recibido y en su contenido, permitiendo modelar distintas reacciones del vehículo ante eventos de alerta.

En particular, la aplicación contempla tres casos principales de recepción. En el primer caso, cuando el mensaje corresponde a una alerta generada por el propio nodo, el paquete es descartado, evitando procesar información redundante. En el segundo caso, cuando se recibe una alerta válida proveniente de otro vehículo y el evento aún no ha sido procesado previamente, el nodo ejecuta acciones asociadas a una situación de riesgo, tales como la reducción o detención de la velocidad del vehículo. Finalmente, en el tercer caso, si el mensaje recibido corresponde a una alerta ya procesada con anterioridad, el paquete es ignorado para evitar reacciones repetidas ante el mismo evento.

Esta lógica permite representar un comportamiento vehicular más realista, en el cual los nodos no solo reciben información, sino que reaccionan de forma diferenciada según el contexto del mensaje, influyendo directamente en la dinámica del tráfico y en la evolución de la simulación.

Adicionalmente, se implementó un módulo `AlertRelay`, asignado a un nodo de tipo RSU, cuyo objetivo es retransmitir los mensajes de alerta recibidos. Este módulo permite evaluar escenarios de diseminación multi-salto sin introducir complejidad adicional en la lógica de red. Para evitar la propagación indefinida de mensajes duplicados, el relay mantiene un registro local de los identificadores de mensajes previamente procesados y descarta aquellos que ya han sido retransmitidos.

La lógica de temporización de las aplicaciones se controla mediante eventos internos del simulador, lo que permite ajustar con precisión los instantes de inicio y finalización de la transmisión de alertas. Este enfoque facilita la repetición de experimentos bajo condiciones controladas y la comparación de resultados entre distintos escenarios de simulación.

En conjunto, la implementación de la lógica de aplicación proporciona un mecanismo sencillo pero efectivo para la diseminación de alertas vehiculares, permitiendo centrar el análisis en el comportamiento del enlace NR Sidelink y en la influencia de la movilidad sobre la propagación de la información.

III-B6. Archivos auxiliares analizados: Además de los archivos directamente modificados para la implementación del escenario y de la lógica de aplicación, fue necesario analizar diversos archivos pertenecientes a los frameworks utilizados, con el fin de comprender su funcionamiento interno y validar las decisiones de configuración adoptadas. Estos archivos no fueron alterados, pero resultaron fundamentales para interpretar el comportamiento observado durante la simulación.

En el contexto de Simu5G, se analizaron archivos asociados a los mecanismos de adaptación de enlace y a las tablas internas de modulación y codificación. En particular, se revisaron los módulos encargados de la gestión del AMC y de la selección de parámetros de transmisión a partir del indicador de calidad de canal. Este análisis permitió identificar cómo un valor fijo de CQI se traduce internamente en una modulación y una tasa de codificación específicas, así como comprender el efecto de deshabilitar la adaptación dinámica del enlace en escenarios de comunicación sidelink.

Asimismo, se estudiaron los archivos relacionados con la definición de la interfaz celular NR, comúnmente referida como *cellular NIC*. Estos módulos implementan la encapsulación de paquetes IP provenientes de la capa de red en la pila de protocolos celulares, gestionando su paso por las capas MAC y PHY de forma abstracta. El análisis de estos archivos fue clave para entender cómo los paquetes generados por las aplicaciones son finalmente transmitidos a través del enlace NR Sidelink.

Desde el lado de la movilidad, se revisaron archivos internos de Veins relacionados con la sincronización mediante TraCI y la actualización de los parámetros de movilidad de los nodos. Esto permitió comprender el mecanismo mediante el cual los cambios en la posición y velocidad de los vehículos en SUMO se reflejan de manera inmediata en los nodos de OMNeT++, influyendo directamente en las condiciones del enlace radio.

Finalmente, se consultaron archivos de configuración y documentación del framework INET para comprender el manejo de tráfico multicast sobre interfaces específicas y la interacción entre las capas de transporte y red. Este análisis resultó necesario para asegurar que los mensajes de alerta fueran correctamente encaminados a través de la interfaz celular y no por otras interfaces disponibles en los nodos.

Como resultado de este análisis, fue posible interpretar de manera concreta los parámetros de transmisión utilizados en el escenario actual. En particular, al fijar `d2dCqi = 7` y habilitar `usePreconfiguredTxParams = true`, la selección de parámetros de enlace se realiza mediante las tablas internas de modulación y codificación, donde el valor de CQI se mapea a un perfil fijo de modulación y tasa de codificación. Para el caso evaluado, el CQI configurado corresponde a una modulación 16QAM y a un valor de tasa de codificación nominal equivalente a $378/1024 \approx 0{,}369$, lo cual representa un esquema relativamente robusto para la transmisión de mensajes de seguridad. Esta interpretación permite relacionar directamente los

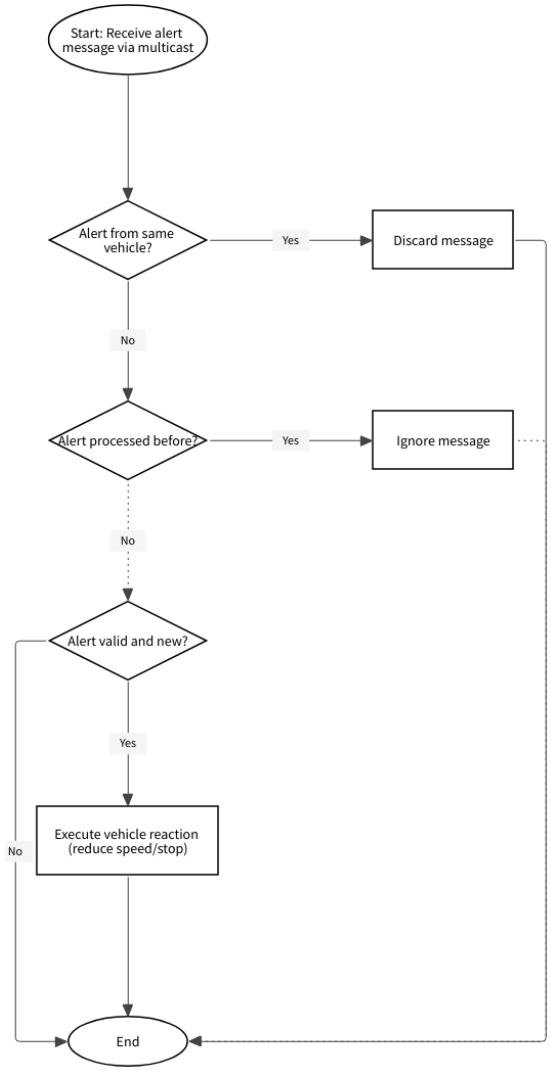

Figura 8: Flujo de diseminación de mensajes de alerta entre el nodo emisor, los nodos receptores y el nodo de retransmisión.

parámetros definidos en `omnetpp.ini` con el comportamiento observado en la simulación, sin requerir adaptación dinámica del enlace.

III-B7. Instrumentación y métricas observadas: Con el fin de evaluar el desempeño del escenario de comunicación NR Sidelink implementado, se instrumentó la simulación para registrar métricas a nivel de aplicación que permitan caracterizar la confiabilidad y la latencia en la diseminación de mensajes de alerta. La selección de estas métricas se realizó considerando el enfoque del escenario, orientado a comunicaciones vehiculares de seguridad, donde la correcta recepción de los mensajes y su tiempo de entrega resultan aspectos críticos.

La métrica principal considerada fue el *Packet Delivery Ratio* (PDR), definido como la relación entre el número de mensajes de alerta recibidos correctamente por un nodo y el número total de mensajes transmitidos durante el intervalo de simulación. Esta métrica se registró de forma individual para cada nodo vehicular receptor, lo que permite observar el comportamiento heterogéneo de la red y evaluar cómo la comunicación directa sidelink se ve influenciada por la topología dinámica del escenario.

Adicionalmente, se construyó la función de distribución acumulada (CDF) del PDR a partir de los valores obtenidos para todos los nodos. Esta representación permite analizar el desempeño global del sistema de forma agregada, proporcionando una visión compacta del porcentaje de nodos que alcanzan determinados niveles de confiabilidad en la recepción de mensajes de alerta.

Como métrica complementaria, se registró el retardo de los mensajes de alerta a nivel de aplicación, definido como la diferencia entre el instante de transmisión del mensaje por el nodo emisor y el instante de recepción en el nodo destinatario. Este retardo incluye los efectos combinados de la pila de protocolos, el acceso al medio, el modelo físico abstracto y la posible

retransmisión de mensajes, constituyendo una medida representativa del tiempo de reacción del sistema ante eventos de riesgo.

La recolección de estas métricas se realizó mediante registros generados por las aplicaciones desarrolladas y por los mecanismos de trazado de OMNeT++. Los valores obtenidos fueron posteriormente procesados de forma externa para su análisis y visualización, dando lugar a gráficas que permiten interpretar el comportamiento del sistema tanto a nivel individual de nodo como a nivel global.

La instrumentación descrita establece la base para el análisis de resultados presentado en la siguiente sección, donde se discute el impacto de la configuración del enlace NR Sidelink y de la movilidad vehicular sobre la confiabilidad y el retardo en la diseminación de mensajes de alerta.

IV. Análisis de resultados

IV-A. Resultados en NS-3

Esta sección presenta la evaluación técnica de la simulación, abordando tanto el comportamiento de la pila de protocolos de comunicaciones 5G NR V2X como la respuesta de la lógica de aplicación de seguridad. Los resultados se han obtenido mediante la ejecución del escenario en el simulador ns-3, utilizando trazas de consola en tiempo real, archivos de registro (.csv) y herramientas de visualización.

El análisis se divide en tres componentes: la validación del procedimiento de acceso al medio (MAC) en el enlace lateral, la respuesta temporal del sistema de frenado (AEB) y el rendimiento global de la red (QoS).

IV-A1. Análisis de Trazas MAC y Operación Sidelink: Para validar que la comunicación *Sidelink* (PC5) opera bajo los estándares del 3GPP para 5G NR, se inspeccionaron los registros generados por el componente `NrSlUeMac`. La Figura 9 muestra la actividad detallada del planificador y la capa MAC instantes antes y durante el evento de frenado.

```
erick@ErickLAPTOP: ~/works
[ CellId 0, bwpId 0, rnti 1] Adding destination 255 with priority 0 to list of sidelink Tx destinations
[ CellId 0, bwpId 0, rnti 2] Adding destination 255 to list of sidelink Rx destinations
[ CellId 0, bwpId 0, rnti 1] Reporting for Sidelink. Tx Queue size = 335
[ CellId 0, bwpId 0, rnti 1] Received grant to dstL2Id 255 on HARQ ID 0 containing 1 slots and RRI +100ms
[ CellId 0, bwpId 0, rnti 1] Notifying NR SL RLC of TX opportunity for LC id 5 for TB size 335
[ CellId 0, bwpId 0, rnti 1] Grant at : Frame = 70 SF = 2 slot = 1
[ CellId 0, bwpId 0, rnti 1] Sending PSSCH MAC PDU (1st Tx) dstL2Id: 255 harqId: 0 Packet Size: 335
[ CellId 0, bwpId 0, rnti 2] IMSI 2 adding LC from 1 to 255 lcId 5 dynamic true pdb +20ms
[ CellId 0, bwpId 0, rnti 2] SL PDU reception on LC 5 from src: 1 to dst: 255
[ CellId 0, bwpId 0, rnti 1] Reporting for Sidelink. Tx Queue size = 335
[ CellId 0, bwpId 0, rnti 1] Received grant to dstL2Id 255 on HARQ ID 0 containing 1 slots and RRI +100ms
[ CellId 0, bwpId 0, rnti 1] Notifying NR SL RLC of TX opportunity for LC id 5 for TB size 335
[ CellId 0, bwpId 0, rnti 1] Grant at : Frame = 80 SF = 2 slot = 1
[ CellId 0, bwpId 0, rnti 1] Sending PSSCH MAC PDU (1st Tx) dstL2Id: 255 harqId: 0 Packet Size: 335
[ CellId 0, bwpId 0, rnti 2] SL PDU reception on LC 5 from src: 1 to dst: 255
[ CellId 0, bwpId 0, rnti 1] Reporting for Sidelink. Tx Queue size = 335
[ CellId 0, bwpId 0, rnti 1] Received grant to dstL2Id 255 on HARQ ID 0 containing 1 slots and RRI +100ms
[ CellId 0, bwpId 0, rnti 1] Notifying NR SL RLC of TX opportunity for LC id 5 for TB size 335
[ CellId 0, bwpId 0, rnti 1] Grant at : Frame = 90 SF = 2 slot = 1
[ CellId 0, bwpId 0, rnti 1] Sending PSSCH MAC PDU (1st Tx) dstL2Id: 255 harqId: 0 Packet Size: 335
[ CellId 0, bwpId 0, rnti 2] SL PDU reception on LC 5 from src: 1 to dst: 255
[ CellId 0, bwpId 0, rnti 1] Reporting for Sidelink. Tx Queue size = 335
[ CellId 0, bwpId 0, rnti 1] Received grant to dstL2Id 255 on HARQ ID 0 containing 1 slots and RRI +100ms
[ CellId 0, bwpId 0, rnti 1] Notifying NR SL RLC of TX opportunity for LC id 5 for TB size 335
[ CellId 0, bwpId 0, rnti 1] Grant at : Frame = 100 SF = 2 slot = 1
[ CellId 0, bwpId 0, rnti 1] Sending PSSCH MAC PDU (1st Tx) dstL2Id: 255 harqId: 0 Packet Size: 335
[ CellId 0, bwpId 0, rnti 2] SL PDU reception on LC 5 from src: 1 to dst: 255

[t=1.1s] 🚨 PELIGRO (AEB) -> Frenando.
 [ CellId 0, bwpId 0, rnti 1] Reporting for Sidelink. Tx Queue size = 335
```

Figura 9: Trazas de la capa MAC Sidelink y activación del evento AEB en $t = 1{,}1s$.

Del análisis de estas trazas se extrae la siguiente secuencia operativa del protocolo:

- **Solicitud de Recursos (Buffer Status Reporting):** Se observa la línea `Reporting for Sidelink. Tx Queue size = 335`. Esto indica que la aplicación ha depositado un paquete (de 300 bytes más cabeceras IP/UDP) en la cola del RLC, y el UE (Vehículo 0, identificado como `rnti 1`) solicita recursos al planificador autónomo para transmitirlo.
- **Asignación de Grant (Resource Allocation):** La traza `Received grant to dstL2Id 255 on HARQ ID 0` confirma que el planificador del UE ha seleccionado exitosamente recursos en tiempo y frecuencia (Modo 2). El `dstL2Id 255` confirma que es una transmisión *Broadcast* (dirigida a todos los vecinos). El uso de `HARQ ID 0` indica el proceso de solicitud de repetición automática híbrida asignado.

- **Estructura de Trama NR (Timing):** La línea `Grant at : Frame = 80 SF = 2 slot = 1` revela la sincronización precisa de la red 5G. El paquete es programado para transmitirse en una ubicación específica de la cuadrícula de recursos (Frame 80, Subframe 2, Slot 1), respetando la numerología configurada ($\mu = 2$).
- **Transmisión Física (PSSCH):** Finalmente, la instrucción `Sending PSSCH MAC PDU` confirma que la unidad de datos del protocolo MAC se ha pasado a la capa física para su emisión por el Canal Compartido de Enlace Lateral Físico (*Physical Sidelink Shared Channel*).

Esta secuencia se repite periódicamente (cada 100ms aprox.), demostrando que la pila de protocolos está funcional y operando de manera cíclica y estable antes de la emergencia.

IV-A2. Validación de la Lógica de Frenado (AEB): En la misma Figura 9, se observa la intersección entre la lógica de control y la comunicación. En el instante de simulación $t = 1{,}1s$, el sistema detecta una condición crítica:

- **Detección:** El algoritmo calcula un Tiempo Hasta la Colisión (TTC) de $0{,}33s$, violando el umbral de seguridad de $1{,}5s$.
- **Acción:** Se imprime la alerta `PELIGRO (AEB) ->Frenando`.
- **Persistencia de Red:** Es crucial notar que, inmediatamente después de la línea de frenado, aparece nuevamente `Reporting for Sidelink`. Esto valida que el frenado de emergencia no interrumpe la capacidad del vehículo para seguir transmitiendo mensajes de alerta a otros usuarios de la vía, un requisito fundamental para la seguridad cooperativa.

IV-A3. Métricas de Calidad de Servicio (QoS): La validación cuantitativa de la red se sustenta en las métricas obtenidas al finalizar la simulación, presentadas en la Figura 10.

Figura 10: Reporte final de métricas de red NR V2X.

- **Confiabilidad (PDR):** Se alcanzó un *Packet Delivery Ratio* de **99.34 %**. De 408 paquetes transmitidos, solo 3 se perdieron (típicamente durante la fase de descubrimiento inicial), lo cual excede los requisitos de confiabilidad para servicios V2X críticos.
- **Latencia (Delay):** El retardo promedio extremo a extremo fue de **5.48 ms**. Este valor es extremadamente bajo comparado con tecnologías anteriores (como LTE-V2X o 802.11p), validando la capacidad de NR Sidelink para soportar aplicaciones de tiempo real estricto.

IV-A4. Comportamiento Temporal de la Red: Para verificar la estabilidad de la red a lo largo del tiempo y bajo la dinámica de movilidad del escenario (frenado y reanudación), se generaron gráficas de desempeño temporal para las métricas clave de la interfaz Sidelink (Figura 11).

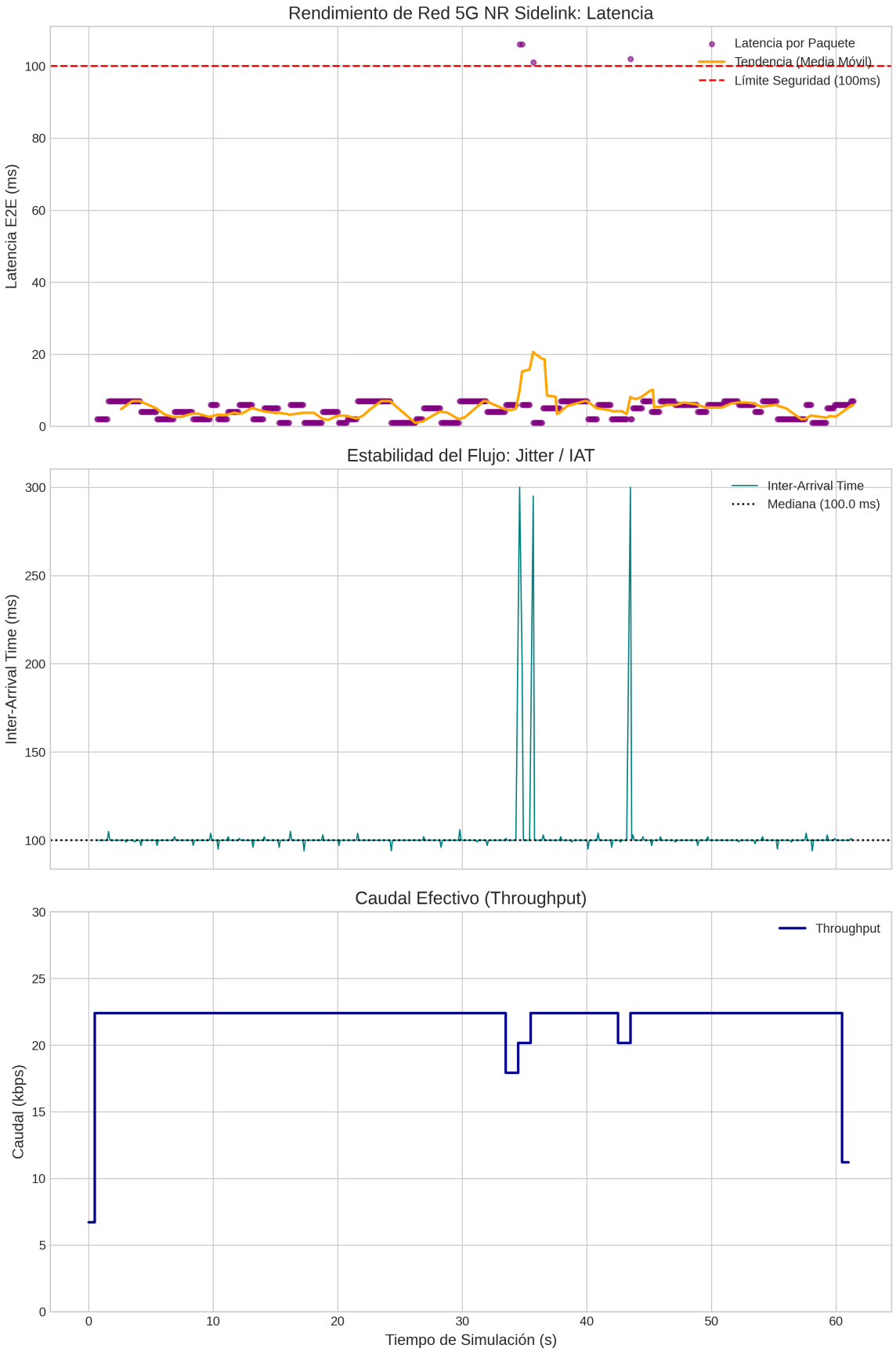

Figura 11: Comportamiento temporal: Latencia E2E, Jitter y Throughput.

Del análisis detallado de estas gráficas se extraen las siguientes observaciones técnicas:

- **Latencia E2E (Gráfica Superior):** El retardo extremo a extremo no es aleatorio, sino que presenta un comportamiento discreto o escalonado, oscilando estrictamente entre los **1 ms y 7 ms**. Este patrón valida la correcta operación de la numerología 5G NR ($\mu = 2$), donde los paquetes deben esperar a la siguiente ranura de tiempo (*slot*) disponible para su transmisión. La tendencia media (línea amarilla) se estabiliza en torno a los 5 ms, manteniéndose órdenes de magnitud por debajo del límite crítico de seguridad de 100 ms (línea roja discontinua), lo que confirma la aptitud de la red para servicios de baja latencia. La única vez en la que esto falla es cuando existe una distancia considerable entre vehículos, es decir cuando el vehículo que se freno se queda atrás mientras el otro vehículo avanza, llega un punto en el que se alejan lo suficiente y existe perdida de paquetes, pero una vez el vehículo vuelve a moverse, alcanza una distancia menor y vuelve a haber buena recepción de paquetes.
- **Estabilidad del Flujo / Jitter (Gráfica Central):** La gráfica del intervalo de llegada (*Inter-Arrival Time*) muestra una línea base perfectamente centrada en la mediana de **100 ms**. Esto corrobora que la aplicación genera tráfico a una tasa constante de 10 Hz. Se observan picos simétricos esporádicos de aproximadamente ± 5 ms (llegando a 105 ms o bajando a 95 ms), los cuales son atribuibles a variaciones mínimas en la programación de recursos del planificador MAC, sin representar una desincronización crítica del flujo de datos.
- **Caudal Efectivo / Throughput (Gráfica Inferior):** El caudal de recepción se mantiene plano y constante en **22.5 kbps** durante la inmensa mayoría de la simulación (desde $t = 0$ hasta $t \approx 33s$). Esta estabilidad demuestra que ni el evento de frenado brusco en $t = 1{,}1s$ ni la reanudación de la marcha afectaron la capacidad del canal. Las fluctuaciones y caídas observadas en los últimos segundos de la simulación ($t > 33s$) coinciden con los instantes donde se registraron los 3 paquetes perdidos reportados en las estadísticas globales, probablemente debido a condiciones de borde al finalizar el tiempo de simulación o a una degradación puntual del canal al aumentar la distancia.

IV-A5. Validación Espacial: Finalmente, la topología fue verificada visualmente mediante NetAnim (Figura 12), confirmando que la distancia inicial inter-vehicular y la trayectoria lineal se correspondieron con los parámetros de diseño.

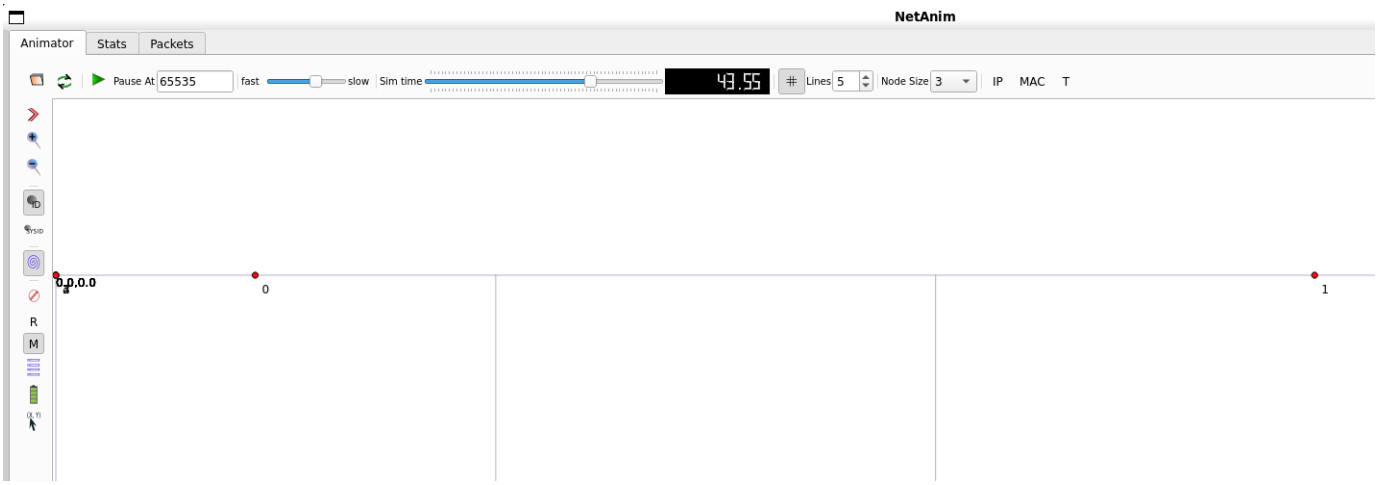

Figura 12: Visualización de la topología en NetAnim.

IV-B. Implementación en OMNET++

En esta subsección se presentan los resultados obtenidos a partir de la simulación del escenario vehicular implementado en OMNeT++, haciendo énfasis en el desempeño de la comunicación NR Sidelink para la diseminación de mensajes de alerta. El análisis se centra en métricas a nivel de aplicación, las cuales permiten evaluar tanto la confiabilidad en la entrega de los mensajes como el retardo asociado a su transmisión.

El código fuente completo de la implementación desarrollada, junto con los archivos de configuración del escenario de simulación y los scripts asociados, se encuentra disponible en un repositorio público, con el objetivo de facilitar la reproducibilidad de los resultados presentados en este trabajo.

El repositorio puede consultarse en: https://github.com/carlosmvinces/5g_sidelink

IV-B1. Desempeño en la entrega de mensajes de alerta: La confiabilidad del sistema se evaluó inicialmente mediante el *Packet Delivery Ratio* (PDR) calculado de forma individual para cada nodo vehicular receptor. Esta métrica permite observar el comportamiento heterogéneo de la red y analizar cómo la comunicación directa sidelink se ve afectada por la dinámica del escenario vehicular.

La Figura 13 muestra el PDR obtenido para cada uno de los nodos considerados en la simulación. Se observa que un grupo de vehículos alcanza valores de PDR cercanos a 1, lo que indica una recepción prácticamente completa de los mensajes de alerta transmitidos. En estos casos, la pérdida de paquetes es mínima y el sistema logra mantener una alta confiabilidad en la entrega de la información.

Por otro lado, se identifican nodos con valores de PDR intermedios, típicamente en el rango de 0.4 a 0.7, así como algunos nodos con valores inferiores. Esta variabilidad refleja la influencia de la topología dinámica y de las condiciones cambiantes del enlace inalámbrico en un escenario vehicular. No obstante, incluso en presencia de esta dispersión, una fracción significativa de los nodos logra recibir una proporción considerable de los mensajes de alerta, lo cual resulta adecuado para aplicaciones de difusión de información de seguridad.

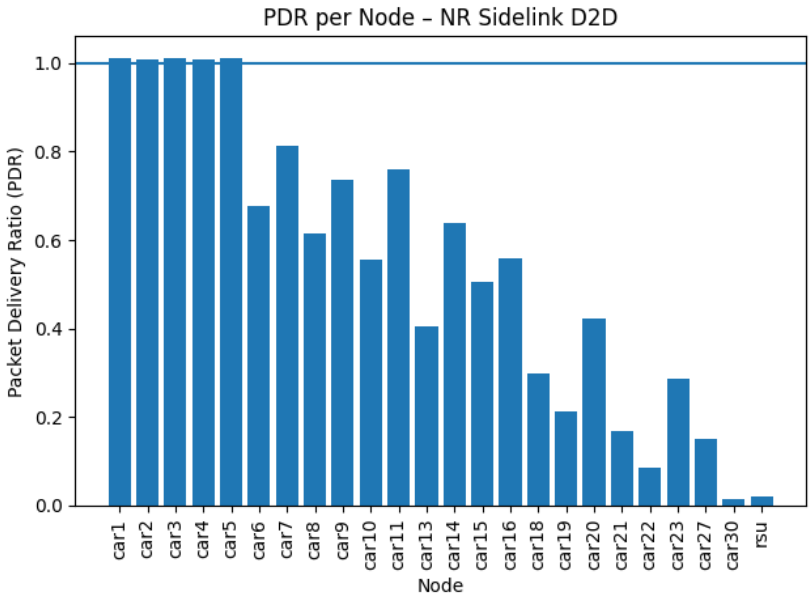

Figura 13: Relación de entrega de paquetes (PDR) obtenida para cada nodo vehicular en el escenario NR Sidelink.

IV-B2. Análisis estadístico global de confiabilidad: Con el objetivo de obtener una visión global del desempeño del sistema, se construyó la función de distribución acumulada (CDF) del PDR a partir de los valores individuales obtenidos para cada nodo. Esta representación permite analizar de manera agregada el porcentaje de nodos que alcanzan determinados niveles de confiabilidad en la recepción de mensajes de alerta.

La Figura 14 muestra la CDF del PDR correspondiente al escenario simulado. A partir de esta gráfica, se observa que aproximadamente la mitad de los nodos alcanza valores de PDR superiores a 0.6, mientras que una fracción relevante de los vehículos presenta valores cercanos o superiores a 0.8. Este comportamiento indica que el sistema, en términos globales, logra una entrega confiable de mensajes de alerta para una porción significativa de los nodos presentes en el escenario.

Asimismo, la forma de la curva evidencia la existencia de nodos con desempeños más degradados, lo cual es consistente con la naturaleza del entorno vehicular y con el uso de comunicación directa sidelink sin control centralizado. La CDF permite así complementar el análisis individual por nodo, proporcionando una medida clara del desempeño agregado del sistema.

Cabe destacar que los valores de PDR se calculan sobre toda la duración de la simulación, por lo que nodos que se alejan progresivamente de la zona del evento presentan valores acumulados menores, reflejando la naturaleza dinámica del escenario vehicular y no un fallo del mecanismo de comunicación.

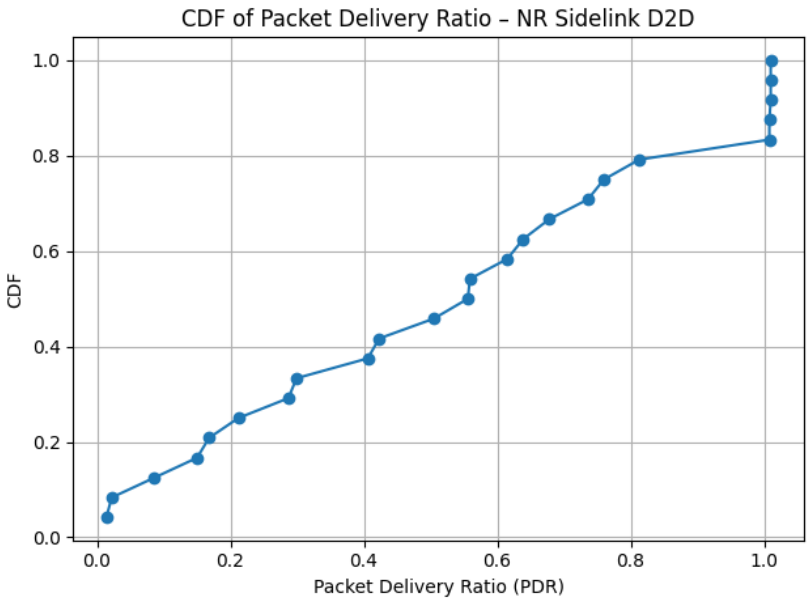

Figura 14: Función de distribución acumulada (CDF) del PDR para los nodos vehiculares en el escenario NR Sidelink.

IV-B3. Retardo en la diseminación de alertas: Además de la confiabilidad en la entrega de mensajes, se analizó el retardo de los mensajes de alerta a nivel de aplicación, métrica especialmente relevante en escenarios de seguridad vehicular. Este retardo se definió como la diferencia temporal entre el instante de transmisión del mensaje por el nodo emisor y el instante de recepción por parte del nodo receptor.

La Figura 15 presenta la distribución de los retardos observados durante la simulación. Los resultados muestran que la mayoría de los mensajes de alerta se entrega con retardos comprendidos aproximadamente entre 10 y 20 ms, centrados en 14ms, con valores máximos que no superan el orden de algunas decenas de milisegundos. Esta concentración de los retardos en un intervalo reducido indica un comportamiento estable y predecible del sistema.

Desde el punto de vista de aplicaciones de seguridad vehicular, estos valores de latencia resultan adecuados, ya que permiten que los nodos receptores reaccionen en tiempos cortos ante la ocurrencia de eventos críticos. La baja dispersión observada en los retardos sugiere, además, que la configuración del enlace NR Sidelink utilizada es capaz de sostener una diseminación rápida de mensajes incluso en un escenario dinámico con múltiples nodos.

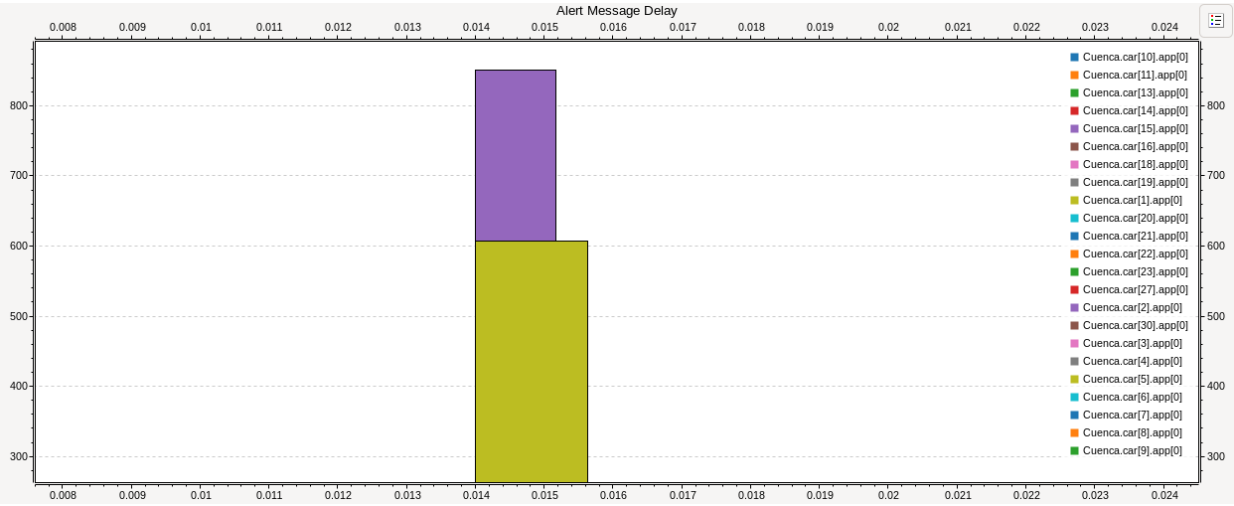

Figura 15: Distribución del retardo de los mensajes de alerta a nivel de aplicación en el escenario NR Sidelink.

IV-B4. Discusión general de los resultados obtenidos: En conjunto, los resultados obtenidos evidencian que la configuración implementada para la comunicación NR Sidelink permite una diseminación eficiente de mensajes de alerta en el escenario vehicular considerado. Los valores de PDR alcanzados por una fracción significativa de los nodos, junto con los retardos

del orden de decenas de milisegundos, indican que el sistema satisface los requerimientos básicos de confiabilidad y latencia asociados a aplicaciones de seguridad.

La variabilidad observada entre nodos es consistente con la dinámica propia del escenario y con las limitaciones inherentes a la comunicación inalámbrica directa. No obstante, el comportamiento global del sistema se mantiene dentro de márgenes aceptables, lo que valida el enfoque adoptado para la configuración del enlace y la lógica de diseminación de alertas.

V. Conclusiones

El desarrollo y análisis de esta simulación han permitido validar que la interfaz PC5 bajo el estándar 5G NR es plenamente capaz de satisfacer los estrictos requisitos de las comunicaciones ultra-confiables de baja latencia (URLLC). Los resultados obtenidos, con un retardo extremo a extremo promedio de 5.48 ms, demuestran que la red proporciona una velocidad de respuesta significativamente superior al límite de seguridad de 100 ms establecido para aplicaciones V2X críticas, confirmando que la tecnología es apta para soportar sistemas de tiempo real estricto como el Frenado de Emergencia Autónomo, permitiendo reacciones vehiculares casi instantáneas incluso a velocidades de autopista.

Adicionalmente, se ha verificado la robustez del Modo 2 para la operación fuera de cobertura, donde los vehículos gestionan su propio acceso al medio sin dependencia de una estación base. Al alcanzar un *Packet Delivery Ratio* (PDR) del 99.34 %, se concluye que los algoritmos de *Sensing* y selección de recursos del estándar NR mantienen la integridad del canal de comunicación de manera eficaz, evitando colisiones de paquetes críticas en escenarios de seguimiento vehicular y asegurando la disponibilidad del servicio de seguridad sin infraestructura celular de soporte.

Por otro lado, el análisis del comportamiento temporal de la red reveló una alta estabilidad en la capa física y MAC frente a la dinámica vehicular. A pesar de las variaciones bruscas en el vector de movilidad causadas por el frenado de emergencia y la posterior reanudación de la marcha, el enlace Sidelink mantuvo una transmisión constante sin degradación por efecto Doppler o *Jitter* significativo. Esto garantiza que la ”visibilidad electrónica.entre vehículos se mantiene ininterrumpida y predecible tanto en situaciones de flujo libre como en eventos de parada crítica.

Finalmente, el proyecto demostró la integración exitosa y la interoperabilidad entre la lógica de aplicación de seguridad y la pila de protocolos de red. La arquitectura implementada probó que es posible procesar la cinemática de los vehículos para detectar colisiones inminentes y ejecutar maniobras de evasión automáticas mientras la red continúa operando en segundo plano, validando el diseño del sistema como una solución integral y viable para la prevención de accidentes en entornos de Sistemas de Transporte Inteligente.

Como complemento a los resultados obtenidos, este trabajo permitió evaluar el uso de simuladores a nivel sistema como herramienta válida para el estudio de comunicaciones vehiculares basadas en NR Sidelink. La implementación desarrollada en OMNeT++ junto con Simu5G, Veins y SUMO demostró que es posible modelar de forma coherente la interacción entre movilidad vehicular, lógica de aplicación y comportamiento del enlace radio, manteniendo un balance adecuado entre realismo y complejidad computacional.

En particular, el enfoque de simulación adoptado permitió abstraer los detalles físicos de la transmisión sin perder la capacidad de analizar métricas relevantes para aplicaciones V2X, tales como la confiabilidad y el retardo a nivel de aplicación. Esta característica resulta especialmente útil en etapas tempranas de diseño y evaluación de sistemas, donde el interés principal radica en el comportamiento global de la red y en la viabilidad funcional de los mecanismos de comunicación propuestos.

Asimismo, el desarrollo de la lógica de aplicación de alertas y la reacción diferenciada de los nodos vehiculares ante distintos tipos de mensajes evidenció la importancia de considerar la capa de aplicación como un elemento activo dentro del sistema V2X. Los resultados muestran que la correcta integración entre la red de comunicaciones y el comportamiento de los vehículos es clave para evaluar de manera realista el impacto de las tecnologías NR Sidelink en escenarios de seguridad vial.

Finalmente, el trabajo establece una base sólida para futuras extensiones, tales como la comparación directa con otros entornos de simulación, la incorporación de configuraciones dinámicas del enlace o el análisis de escenarios de mayor densidad vehicular. De esta forma, la implementación presentada no solo valida el uso de NR Sidelink en aplicaciones de seguridad, sino que también constituye un marco flexible para investigaciones posteriores en el ámbito de los Sistemas de Transporte Inteligente.

Con el fin de promover la transparencia y la reutilización académica, el código fuente de la implementación desarrollada se ha puesto a disposición en un repositorio público, permitiendo que futuros trabajos puedan extender o comparar los resultados obtenidos en este estudio. El repositorio puede consultarse en: https://github.com/carlosmvinces/5g_sidelink

Referencias

[1] Z. Ali, S. Lagén, L. Giupponi, and R. Rouil, “NR V2X Tutorial: Models, Implementation, and Examples,” *Workshop on ns-3 (WNS3)*, Jun. 2022.

[2] T. Zugno, M. Drago, S. Lagén, Z. Ali, and M. Zorzi, “Extending the ns-3 Spatial Channel Model for Vehicular Scenarios,” *Workshop on ns-3 (WNS3)*, Jun. 2021.

[3] T. Henderson, S. Gamboa, A. Ben-Mosbah, W. Garey, C. Liu, and R. Rouil, “5G NR Sidelink, ProSe, and Public Safety Communications models,” *ns-3 Annual Meeting*, Jun. 2024.

[4] Preprints.org, “Cellular V2X Sidelink Resource Allocation,” [Online]. Disponible: https://www.preprints.org/manuscript/202310.1212. Accedido: 2024.

VI. ANEXOS

VI-A. Código de v2v-collision-avoidance.cc para NS3

```
/* -*- Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/nr-module.h"
#include "ns3/lte-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/netanim-module.h"
#include "ns3/antenna-module.h"

#include <iostream>
#include <fstream> // <--- IMPORTANTE: Necesaria para guardar archivos
#include <vector>
#include <bitset>
#include <set>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("V2vAebDemo");

/* ================================================
 * VARIABLES GLOBALES
 * ================================================ */
uint32_t g_txPackets = 0;
uint32_t g_rxPackets = 0;
static uint32_t gAebEvents = 0;

// Variables para métricas
Time g_delaySum = Seconds(0);
uint32_t g_rxBytes = 0;

// Archivo de salida CSV
std::ofstream g_csvFile;
Time g_lastRxTime = Seconds(0); // Para calcular Jitter/IAT

/* ================================================
 * TRAZAS DE RED (AQUÍ SE GUARDAN LOS DATOS)
 * ================================================ */
void TxTrace(Ptr<const Packet> p, const Address& src, const Address& dst, const SeqTsSizeHeader& header)
{
    g_txPackets++;
}

void RxTrace(Ptr<const Packet> p, const Address& src, const Address& dst, const SeqTsSizeHeader& header)
{
    g_rxPackets++;
    g_rxBytes += p->GetSize();

    // 1. Calcular Delay (Latencia)
    Time now = Simulator::Now();
    Time delay = now - header.GetTs();
    g_delaySum += delay;

    // 2. Calcular Inter-Arrival Time (Jitter proxy)
    Time iat = Seconds(0);
    if (g_rxPackets > 1) { // Necesitamos al menos 2 paquetes para calcular intervalo
        iat = now - g_lastRxTime;
    }
    g_lastRxTime = now;

    // 3. Escribir en el CSV: Tiempo, Delay(ms), Tamaño(bytes), IAT(ms)
    // Formato: Time,DelayMs,PacketSize,JitterMs
    g_csvFile << now.GetSeconds() << ","
              << delay.GetMilliSeconds() << ","
              << p->GetSize() << ","
              << iat.GetMilliSeconds() << std::endl;
}

void ReceivePacket(Ptr<const Packet> packet, const Address&) {
    // Ya contamos en RxTrace
}

/* ================================================
 * LOGICA AEB (ORIGINAL)
 * ================================================ */
class V2vAebApp : public Application
{
```

```
public:
  static TypeId GetTypeId (void) {
      static TypeId tid = TypeId ("ns3::V2vAebApp")
        .SetParent<Application> ()
        .SetGroupName ("Applications")
        .AddConstructor<V2vAebApp> ();
      return tid;
  }
  void Setup (Ptr<Node> node, double interval) {
      m_node = node;
      m_interval = interval;
  }

private:
  void StartApplication () override {
      m_event = Simulator::Schedule (Seconds (m_interval), &V2vAebApp::CheckTtc, this);
  }
  void StopApplication () override {
      if (m_event.IsPending ()) Simulator::Cancel (m_event);
  }

  void ResumeMovement () {
      auto myMob = m_node->GetObject<ConstantVelocityMobilityModel> ();
      if (myMob) {
          myMob->SetVelocity (m_savedVelocity);
          m_hasBraked = false;
          std::cout << "[t=" << Simulator::Now ().GetSeconds () << "s] Reanudando marcha...\n";
          m_event = Simulator::Schedule (Seconds (m_interval), &V2vAebApp::CheckTtc, this);
      }
  }

  void CheckTtc () {
      if (m_hasBraked) return;
      auto myMob = m_node->GetObject<ConstantVelocityMobilityModel> ();
      if (!myMob) return;

      Vector myPos = myMob->GetPosition ();
      Vector myVel = myMob->GetVelocity ();

      for (uint32_t i = 0; i < NodeList::GetNNodes (); ++i) {
          Ptr<Node> other = NodeList::GetNode (i);
          if (other == m_node) continue;
          auto otherMob = other->GetObject<ConstantVelocityMobilityModel> ();
          if (!otherMob) continue;

          Vector otherPos = otherMob->GetPosition ();
          Vector otherVel = otherMob->GetVelocity ();

          if (myPos.x >= otherPos.x) continue;

          double distance = CalculateDistance (myPos, otherPos);
          double relSpeed = myVel.x - otherVel.x;

          if (relSpeed <= 0.1) continue;

          double ttc = distance / relSpeed;

          if (ttc < 1.5) {
              gAebEvents++;
              std::cout << "\n[t=" << Simulator::Now ().GetSeconds () << "s]  PELIGRO (AEB) -> Frenando.\n";
              m_savedVelocity = myVel;
              myMob->SetVelocity (Vector (0.0, 0.0, 0.0));
              m_hasBraked = true;
              Simulator::Schedule(Seconds(40.0), &V2vAebApp::ResumeMovement, this);
              break;
          }
      }
      if (!m_hasBraked) {
          m_event = Simulator::Schedule (Seconds (m_interval), &V2vAebApp::CheckTtc, this);
      }
  }

  Ptr<Node> m_node;
  EventId m_event;
  double m_interval;
  bool m_hasBraked{false};
  Vector m_savedVelocity;
};

/* ================================================
 * MAIN
 * ================================================ */
int main(int argc, char* argv[])
{
    // ABRIR EL ARCHIVO CSV
    g_csvFile.open("v2x_metrics.csv");
```

```
    g_csvFile << "Time,DelayMs,PacketSize,IAT" << std::endl; // Header

    uint16_t interUeDistance = 20;
    bool logging = false;
    uint32_t udpPacketSizeBe = 300;
    Time simTime = Seconds(60);
    Time slBearersActivationTime = Seconds(0.5);

    // Configuración Física
    uint16_t numerologyBwpSl = 2;
    double centralFrequencyBandSl = 5.89e9;
    uint16_t bandwidthBandSl = 400;
    double txPower = 23;

    CommandLine cmd(__FILE__);
    cmd.AddValue("logging", "Enable logging", logging);
    cmd.Parse(argc, argv);

    Time finalSimTime = simTime + slBearersActivationTime + Seconds(1.0);

    if (logging) { LogComponentEnable("V2vAebApp", LOG_LEVEL_INFO); }
    LogComponentEnable("NrSlUeMac", LOG_LEVEL_INFO);

    // 1. Nodos
    NodeContainer ueVoiceContainer;
    ueVoiceContainer.Create(2);

    MobilityHelper mobility;
    mobility.SetMobilityModel("ns3::ConstantVelocityMobilityModel");
    Ptr<ListPositionAllocator> positionAllocUe = CreateObject<ListPositionAllocator>();
    positionAllocUe->Add(Vector(0.0, 0.0, 1.5));
    positionAllocUe->Add(Vector(interUeDistance, 0.0, 1.5));
    mobility.SetPositionAllocator(positionAllocUe);
    mobility.Install(ueVoiceContainer);

    ueVoiceContainer.Get(0)->GetObject<ConstantVelocityMobilityModel>()->SetVelocity(Vector(27.7, 0.0, 0.0));
    ueVoiceContainer.Get(1)->GetObject<ConstantVelocityMobilityModel>()->SetVelocity(Vector(13.8, 0.0, 0.0));

    // 2. NR Stack
    Ptr<NrPointToPointEpcHelper> epcHelper = CreateObject<NrPointToPointEpcHelper>();
    Ptr<NrHelper> nrHelper = CreateObject<NrHelper>();
    nrHelper->SetEpcHelper(epcHelper);

    CcBwpCreator ccBwpCreator;
    const uint8_t numCcPerBand = 1;
    CcBwpCreator::SimpleOperationBandConf bandConfSl(centralFrequencyBandSl, bandwidthBandSl, numCcPerBand,
    BandwidthPartInfo::V2V_Highway);
    OperationBandInfo bandSl = ccBwpCreator.CreateOperationBandContiguousCc(bandConfSl);
    nrHelper->InitializeOperationBand(&bandSl);
    auto allBwps = CcBwpCreator::GetAllBwps({bandSl});

    epcHelper->SetAttribute("S1uLinkDelay", TimeValue(MilliSeconds(0)));
    nrHelper->SetUeAntennaAttribute("NumRows", UintegerValue(1));
    nrHelper->SetUeAntennaAttribute("NumColumns", UintegerValue(2));
    nrHelper->SetUeAntennaAttribute("AntennaElement", PointerValue(CreateObject<IsotropicAntennaModel>()));
    nrHelper->SetUePhyAttribute("TxPower", DoubleValue(txPower));
    nrHelper->SetUeMacTypeId(NrSlUeMac::GetTypeId());
    nrHelper->SetUeMacAttribute("EnableSensing", BooleanValue(false));
    nrHelper->SetUeMacAttribute("ActivePoolId", UintegerValue(0));

    uint8_t bwpIdForGbrMcptt = 0;
    nrHelper->SetBwpManagerTypeId(TypeId::LookupByName("ns3::NrSlBwpManagerUe"));
    nrHelper->SetUeBwpManagerAlgorithmAttribute("GBR_MC_PUSH_TO_TALK", UintegerValue(bwpIdForGbrMcptt));

    std::set<uint8_t> bwpIdContainer;
    bwpIdContainer.insert(bwpIdForGbrMcptt);

    NetDeviceContainer ueVoiceNetDev = nrHelper->InstallUeDevice(ueVoiceContainer, allBwps);
    for (auto it = ueVoiceNetDev.Begin(); it != ueVoiceNetDev.End(); ++it) {
        DynamicCast<NrUeNetDevice>(*it)->UpdateConfig();
    }

    // 3. IP Stack
    InternetStackHelper internet;
    internet.Install(ueVoiceContainer);

    NodeContainer epcNodes;
    for (uint32_t i = 0; i < NodeList::GetNNodes(); ++i) {
        if (NodeList::GetNode(i)->GetNDevices() > 0 && !ueVoiceContainer.Contains(i)) {
            epcNodes.Add(NodeList::GetNode(i));
        }
    }
    MobilityHelper epcMobility;
    epcMobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
    epcMobility.Install(epcNodes);
```

```
    Ipv4Address groupAddress4("225.0.0.1");
    Address remoteAddress = InetSocketAddress(groupAddress4, 8000);
    Address localAddress = InetSocketAddress(Ipv4Address::GetAny(), 8000);
    epcHelper->AssignUeIpv4Address(ueVoiceNetDev);

    Ipv4StaticRoutingHelper ipv4RoutingHelper;
    for (uint32_t u = 0; u < ueVoiceContainer.GetN(); ++u) {
        Ptr<Node> ueNode = ueVoiceContainer.Get(u);
        Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting(ueNode->GetObject<Ipv4>());
        ueStaticRouting->SetDefaultRoute(epcHelper->GetUeDefaultGatewayAddress(), 1);
    }

    // 4. Sidelink
    Ptr<NrSlHelper> nrSlHelper = CreateObject<NrSlHelper>();
    nrSlHelper->SetEpcHelper(epcHelper);
    nrSlHelper->SetSlErrorModel("ns3::NrEesmIrT1");
    nrSlHelper->SetNrSlSchedulerTypeId(NrSlUeMacSchedulerFixedMcs::GetTypeId());
    nrSlHelper->SetUeSlSchedulerAttribute("Mcs", UintegerValue(14));
    nrSlHelper->PrepareUeForSidelink(ueVoiceNetDev, bwpIdContainer);

    LteRrcSap::SlResourcePoolNr slResourcePoolNr;
    Ptr<NrSlCommResourcePoolFactory> ptrFactory = Create<NrSlCommResourcePoolFactory>();
    std::vector<std::bitset<1>> slBitmap = {1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1};
    ptrFactory->SetSlTimeResources(slBitmap);
    ptrFactory->SetSlSensingWindow(100);
    ptrFactory->SetSlSelectionWindow(5);
    ptrFactory->SetSlFreqResourcePscch(10);
    ptrFactory->SetSlSubchannelSize(10);
    ptrFactory->SetSlMaxNumPerReserve(3);
    std::list<uint16_t> resourceReservePeriodList = {0, 100};
    ptrFactory->SetSlResourceReservePeriodList(resourceReservePeriodList);
    slResourcePoolNr = ptrFactory->CreatePool();

    LteRrcSap::SlResourcePoolConfigNr slresoPoolConfigNr;
    slresoPoolConfigNr.haveSlResourcePoolConfigNr = true;
    slresoPoolConfigNr.slResourcePoolId.id = 0;
    slresoPoolConfigNr.slResourcePool = slResourcePoolNr;

    LteRrcSap::SlBwpPoolConfigCommonNr slBwpPoolConfigCommonNr;
    slBwpPoolConfigCommonNr.slTxPoolSelectedNormal[0] = slresoPoolConfigNr;

    LteRrcSap::Bwp bwp;
    bwp.numerology = numerologyBwpSl;
    bwp.symbolsPerSlots = 14;
    bwp.rbPerRbg = 1;
    bwp.bandwidth = bandwidthBandSl;

    LteRrcSap::SlBwpGeneric slBwpGeneric;
    slBwpGeneric.bwp = bwp;
    slBwpGeneric.slLengthSymbols = LteRrcSap::GetSlLengthSymbolsEnum(14);
    slBwpGeneric.slStartSymbol = LteRrcSap::GetSlStartSymbolEnum(0);

    LteRrcSap::SlBwpConfigCommonNr slBwpConfigCommonNr;
    slBwpConfigCommonNr.haveSlBwpGeneric = true;
    slBwpConfigCommonNr.slBwpGeneric = slBwpGeneric;
    slBwpConfigCommonNr.haveSlBwpPoolConfigCommonNr = true;
    slBwpConfigCommonNr.slBwpPoolConfigCommonNr = slBwpPoolConfigCommonNr;

    LteRrcSap::SlFreqConfigCommonNr slFreConfigCommonNr;
    for (const auto& it : bwpIdContainer) {
        slFreConfigCommonNr.slBwpList[it] = slBwpConfigCommonNr;
    }

    LteRrcSap::SidelinkPreconfigNr slPreConfigNr;
    slPreConfigNr.slPreconfigFreqInfoList[0] = slFreConfigCommonNr;
    slPreConfigNr.slPreconfigGeneral.slTddConfig.tddPattern = "DL|DL|DL|F|UL|UL|UL|UL|UL|UL|";
    LteRrcSap::SlUeSelectedConfig slUeSelectedPreConfig;
    slUeSelectedPreConfig.slProbResourceKeep = 0;
    LteRrcSap::SlPsschTxParameters psschParams;
    psschParams.slMaxTxTransNumPssch = 2;
    LteRrcSap::SlPsschTxConfigList pscchTxConfigList;
    pscchTxConfigList.slPsschTxParameters[0] = psschParams;
    slUeSelectedPreConfig.slPsschTxConfigList = pscchTxConfigList;
    slPreConfigNr.slUeSelectedPreConfig = slUeSelectedPreConfig;
    nrSlHelper->InstallNrSlPreConfiguration(ueVoiceNetDev, slPreConfigNr);

    SidelinkInfo slInfo;
    slInfo.m_castType = SidelinkInfo::CastType::Groupcast;
    slInfo.m_dstL2Id = 255;
    slInfo.m_rri = MilliSeconds(100);
    slInfo.m_dynamic = false;

    Ptr<LteSlTft> tftTx = Create<LteSlTft>(LteSlTft::Direction::TRANSMIT, groupAddress4, slInfo);
    NetDeviceContainer txDev; txDev.Add(ueVoiceNetDev.Get(0));
    nrSlHelper->ActivateNrSlBearer(slBearersActivationTime, txDev, tftTx);

```

```
    Ptr<LteSlTft> tftRx = Create<LteSlTft>(LteSlTft::Direction::RECEIVE, groupAddress4, slInfo);
    NetDeviceContainer rxDev; rxDev.Add(ueVoiceNetDev.Get(1));
    nrSlHelper->ActivateNrSlBearer(slBearersActivationTime, rxDev, tftRx);

    // 5. Apps
    OnOffHelper onOff("ns3::UdpSocketFactory", remoteAddress);
    onOff.SetConstantRate(DataRate("24kb/s"), udpPacketSizeBe);
    onOff.SetAttribute("EnableSeqTsSizeHeader", BooleanValue(true));

    ApplicationContainer clientApps = onOff.Install(ueVoiceContainer.Get(0));
    clientApps.Start(slBearersActivationTime + Seconds(0.1));
    clientApps.Stop(finalSimTime);

    PacketSinkHelper sink("ns3::UdpSocketFactory", localAddress);
    sink.SetAttribute("EnableSeqTsSizeHeader", BooleanValue(true));
    ApplicationContainer serverApps = sink.Install(ueVoiceContainer.Get(1));
    serverApps.Start(Seconds(0.0));

    // 6. Apps AEB
    for (uint32_t i = 0; i < ueVoiceContainer.GetN (); ++i) {
      Ptr<V2vAebApp> app = CreateObject<V2vAebApp> ();
      app->Setup (ueVoiceContainer.Get (i), 0.1);
      ueVoiceContainer.Get (i)->AddApplication (app);
      app->SetStartTime (Seconds (1.0));
      app->SetStopTime (finalSimTime);
    }

    clientApps.Get(0)->TraceConnectWithoutContext("TxWithSeqTsSize", MakeCallback(&TxTrace));
    serverApps.Get(0)->TraceConnectWithoutContext("RxWithSeqTsSize", MakeCallback(&RxTrace));
    Config::ConnectWithoutContext("/NodeList/*/ApplicationList/*/$ns3::PacketSink/Rx", MakeCallback(&ReceivePacket));

    std::cout << "Iniciando Simulación AEB NR-V2X..." << std::endl;
    AnimationInterface anim ("v2v-aeb-animation.xml");
    anim.SetMobilityPollInterval (Seconds (0.1));

    Simulator::Stop(finalSimTime);
    Simulator::Run();

    // Reporte
    double pdr = (g_txPackets > 0) ? ((double)g_rxPackets / g_txPackets) * 100.0 : 0.0;
    double avgDelayMs = (g_rxPackets > 0) ? (g_delaySum.GetSeconds() / g_rxPackets) * 1000.0 : 0.0;

    std::cout << "\n=== RESULTADOS NR-V2X ===\n";
    std::cout << "PDR: " << pdr << " %\n";
    std::cout << "Delay Medio: " << avgDelayMs << " ms\n";
    std::cout << "Datos guardados en 'v2x_metrics.csv'\n";

    g_csvFile.close(); // Cerrar archivo
    Simulator::Destroy();
    return 0;
}
```