Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

Sistemas de comunicación vehicular: Análisis de tecnologías facilitadoras

INGE-2741

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en Telecomunicaciones

Presentado por:
Lourdes Betzabé Chalén Carvajal
María de los Ángeles Montoya Llamuca

Guayaquil - Ecuador
Año: 2024

Dedicatoria

A mis queridos padres, por su amor incondicional, esfuerzo y sacrificio, que me brindaron los medios y la oportunidad de alcanzar esta meta. Su confianza en mí han sido mi mayor impulso.

A mis hermanas, por ser mi inspiración, mi refugio y mi fuerza. Su amor y motivación constante me han guiado y fortalecido siempre.

A mi abuelita, cuyo amor y enseñanzas siguen presentes en mi corazón. Este logro también es un homenaje a su memoria.

A Zunlla, mi querida mascota, por su amor incondicional y la felicidad que me brindó.

A mi enamorado, Daniel, por su apoyo incondicional, por enseñarme tanto en este proceso y motivarme siempre a seguir adelante.

A mis profesores, por su dedicación y enseñanzas, que han dejado una huella imborrable en mi formación académica.

María de los Ángeles Montoya Llamuca

Dedicatoria

Dedico este proyecto a Dios, por iluminarme, guiarme y nunca abandonarme durante mis años de estudio. A mi abuelita Josefa, por siempre ser mi motivación, y aunque no esté físicamente presente, sé que hubiese estado feliz de compartir este momento conmigo. Este logro es tanto suyo como mío.

A mis mascotas Coco y Luna, quienes me acompañaron las madrugadas de estudio.

A mis padres y hermanos por estar a mi lado motivándome, por creer en mí aun cuando yo no lo hacía.

A mis amigos, por su confianza y calma en momentos difíciles.

A mis docentes por su dedicación al enseñar en las aulas, y a mis compañeros de carrera por explicarme las cosas que no entendía y ser un gran apoyo en este camino.

Lourdes Betzabé Chalén Carvajal

Declaración Expresa

Nosotras Lourdes Betzabé Chalén Carvajal y María de los Ángeles Montoya Llamuca acordamos y reconocemos que:

La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por mí/nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que me/nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de mi/nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique los autores que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 4 de febrero del 2025.

Lourdes Betzabé Chalén Carvajal

María de los Ángeles Montoya Llamuca

Evaluadores

Ms.C. María Alvarez Villanueva

Profesor de Materia

Ph.D. Francisco Novillo Parales

Tutor de proyecto

Resumen

Este proyecto aborda la optimización de la comunicación en redes vehiculares en entornos urbanos densos mediante la evaluación del desempeño de protocolos de enrutamiento. La congestión en el canal y la inestabilidad en la transmisión de datos afectan la seguridad vial y la gestión del tráfico, por lo que se justifica la necesidad de analizar protocolos eficientes que minimicen estos problemas. El objetivo es identificar el protocolo más adecuado en términos de latencia, tasa de entrega de paquetes (PDR) y throughput.

Para ello, se implementó un entorno de simulación en OMNeT++, integrado con SUMO y haciendo uso de VEINS e INET, frameworks diseñados para la simulación de redes vehiculares y la evaluación de protocolos de comunicación. Se evaluaron los protocolos AODV y DYMO en una red vehicular de 50 vehículos en escenarios urbanos. Las simulaciones consideraron distintos niveles de densidad vehicular y los datos recolectados fueron analizados y comparados, permitiendo una evaluación rigurosa del desempeño de ambos protocolos.

Los resultados indicaron que AODV ofrece mayor estabilidad en la transmisión de datos, menor latencia y mayor throughput en comparación con DYMO, el cual presenta una sobrecarga de control significativa en redes de alta densidad. En conclusión, AODV se establece como una opción más eficiente para redes vehiculares urbanas, contribuyendo a mejorar la comunicación y la seguridad en sistemas de transporte inteligente.

Palabras Clave: redes vehiculares, enrutamiento, AODV, DYMO, throughput.

Abstract

This project focuses on the optimization of communication in vehicular networks in dense urban environments through the evaluation of routing protocol performance. Channel congestion and instability in data transmission affect road safety and traffic management, making it necessary to analyze efficient protocols that minimize these problems. The objective is to identify the most suitable protocol in terms of latency, packet delivery ratio (PDR), and throughput.

To achieve this, a simulation environment was implemented in OMNeT++, integrated with SUMO and using VEINS and INET, frameworks designed for vehicular network simulation and the evaluation of communication protocols. The AODV and DYMO protocols were evaluated in a vehicular network of 50 vehicles in urban scenarios. The simulations considered different levels of vehicular density, and the collected data were analyzed and compared, allowing for a rigorous evaluation of both protocols' performance.

The results showed that AODV provides greater stability in data transmission, lower latency, and higher throughput compared to DYMO, which has significant control overhead in high-density networks. In conclusion, AODV is established as a more efficient option for urban vehicular networks, contributing to improved communication and safety in intelligent transportation systems.

Keywords: vehicular networks, routing, AODV, DYMO, throughput.

Índice general

Abreviaturas

AODV Ad hoc On-Demand Distance Vector

CSMA/CA Acceso múltiple por detección de portadora con evitación de colisiones

C-V2X Vehículo celular a todo

DSDV Destination-Sequenced Distance Vector

DSR Dynamic Source Routing

DSRC Comunicación de corto alcance dedicada

DSRC Dedicated Short-Range Communications

DYMO Dynamic MANET On-demand Routing Protocol

ICVN Intermittently Connected Vehicular Network

ICVN Redes vehiculares centradas en la información

IEEE 802.11p Estándar de comunicación inalámbrica para aplicaciones vehiculares

IoT Internet of Things

IT Transporte inteligente

ITS-G5 Sistema de transporte inteligente para comunicación V2V y V2I

LTE Estándar de comunicación móvil de alta velocidad

MPR Multi-Point Relay

OLSR Optimized Link State Routing

QoS Calidad del servicio

ReA Adaptativo a la solicitud

RERR Route Error

RREP Route Reply

RREQ Route Request

RSU Unidades de carretera

RSUC Unidad de servicio remota céntrica

V2I Vehículos e infraestructura

V2V Comunicación entre vehículos

VANET Vehicular Ad Hoc Network

ZRP Zone Routing Protocol

Simbología

µs	Microsegundos
Kbps	Kilobits por segundo
Km/h	Kilómetros por hora
m	Metro
m/s	Metros por segundo
Pdr	Packet Delivery Ratio
s	Segundos

Índice de figuras

Índice de tablas

Índice de anexos

Capítulo 1

1. Introducción

En la era de la digitalización y la transformación tecnológica, la industria automotriz ha evolucionado rápidamente, con un enfoque especial en la electrificación y automatización de vehículos. Los vehículos autónomos se han convertido en un pilar fundamental de la innovación tecnológica ya que, dependen de sistemas avanzados de automatización que requieren redes de comunicación confiables para operar eficazmente en condiciones cambiantes y garantizar la seguridad vial, reducir accidentes y mejorar la eficiencia del tráfico [1].

Los vehículos autónomos requieren conectarse a redes avanzadas, como vehículo celular a todo (C-V2X) o IEEE 802.11p, que posibilitan la transmisión en tiempo real de datos críticos entre los vehículos, las infraestructuras y la nube. Estas redes permiten que los automóviles reaccionen de manera inmediata ante situaciones imprevistas, como la aparición de peatones o la presencia de otros vehículos en su trayectoria [1].

A medida que aumenta la densidad de vehículos conectados, los desafíos de congestión en los canales de comunicación se vuelven más críticos. En escenarios urbanos con alta densidad vehicular, la congestión puede provocar latencias elevadas, afectando la transmisión de datos críticos para la seguridad vial. Este proyecto busca promover infraestructuras resilientes y tecnológicas que soporten el crecimiento de vehículos autónomos de nivel 5 en entornos urbanos densos, garantizando la seguridad y eficiencia en la movilidad. Para abordar estos desafíos, se evaluarán diversos protocolos de comunicación de capa 3 en redes vehiculares, analizando su desempeño en términos de estabilidad en la transferencia de datos y eficiencia en entornos urbanos de alta densidad. Lo que permitirá una transmisión rápida y confiable de la información, contribuyendo significativamente a la prevención colisiones, optimización del

flujo vehicular y la mejorar de la seguridad tanto de peatones como de ocupantes de los vehículos [2].

1.1. Descripción del problema

La congestión en los canales de comunicación vehicular en entornos urbanos densos es causada por la alta densidad de vehículos, la limitación del ancho de banda y la interferencia en el espectro. La sobrecarga de redes como C-V2X y IEEE 802.11p genera colisiones de datos y retrasos en la transmisión de información esencial, como alertas de colisión. Además, estas limitaciones se agravan por la presencia de infraestructuras como edificios, que interfieren en la calidad de las comunicaciones. Este problema necesita ser resuelto, ya que compromete la capacidad de los vehículos autónomos para tomar decisiones rápidas y dificulta la transmisión en tiempo real de datos críticos, lo que afecta tanto a la seguridad vial como a la eficiente gestión del tráfico [2].

Las causantes mencionadas también influyen negativamente en el rendimiento de los protocolos de comunicación de capa 3 en redes vehiculares. La ineficiencia en estos protocolos puede conducir a fallos en la coordinación entre vehículos, incrementando el riesgo de accidentes. Esta problemática es especialmente relevante en el contexto de los Sistemas de Transporte Inteligente (ITS) y las infraestructuras urbanas, que dependen de una gestión eficaz de la comunicación vehicular para su correcto funcionamiento.

1.2. Justificación del problema

La evaluación de diversos protocolos de comunicación de capa 3 en redes vehiculares es esencial para garantizar comunicaciones confiables y rápidas entre vehículos autónomos en entornos urbanos densos. La alta densidad vehicular y la topología dinámica de estas redes pueden afectar la estabilidad en la transferencia de datos y la eficiencia de la comunicación, comprometiendo la seguridad vial y la gestión del tráfico.

En entornos con alta densidad vehicular, la falta de una infraestructura de comunicación estable pone en riesgo la seguridad de los ocupantes y peatones, y compromete la eficiencia en la gestión del tráfico. Además, la coexistencia de tecnologías de comunicación que operan en la banda de 5.9 GHz, como IEEE 802.11p y C-V2X, aumenta la probabilidad de colisiones de paquetes y reduce el throughput, lo que afecta la transmisión de datos críticos en tiempo real [3].

Abordar esta problemática mediante la evaluación de protocolos de comunicación permitirá identificar aquellos que optimicen la diseminación de información y el acceso al canal, reduciendo la latencia y mejorando la estabilidad en la transferencia de datos. Esto garantizará una comunicación vehicular más segura y eficiente, contribuyendo al desarrollo de infraestructuras que soporten el crecimiento de vehículos autónomos en entornos urbanos densos.

1.3. Objetivos

1.3.1. Objetivo general

Evaluar protocolos de comunicación de capa 3 en redes vehiculares, analizando su desempeño en términos de estabilidad en la transferencia de datos y eficiencia en entornos urbanos densos, identificando aquellos que permitan una transmisión de información rápida y confiable.

1.3.2. Objetivos específicos

- Realizar una revisión bibliográfica exhaustiva de los protocolos de comunicación de capa 3 utilizados en redes vehiculares, identificando sus características y limitaciones para evaluar su desempeño en entornos urbanos densos.

- Implementar y probar diferentes protocolos de comunicación en un entorno vehicular simulado, analizando métricas clave como latencia, throughput y tasa de entrega de paquetes para determinar su eficiencia.
- Comparar y evaluar el desempeño de los protocolos seleccionados en términos de estabilidad en la transferencia de datos y capacidad para soportar condiciones de alta densidad vehicular, utilizando escenarios simulados realistas.

1.4. Marco teórico

1.4.1. Tecnologías de comunicación vehicular

Las tecnologías vehiculares más comunes utilizadas para la comunicación entre vehículos (V2V) y entre vehículos e infraestructura (V2I) son IEEE 802.11p y vehículo celular a todo (C-V2X).

Uno de los principales problemas que surge con la coexistencia de múltiples tecnologías de comunicación es la falta de interoperabilidad entre vehículos de diferentes fabricantes. Esto genera desafíos en la coordinación de maniobras y transmisión de datos críticos cuando vehículos que usan IEEE 802.11p necesitan comunicarse con aquellos que emplean C-V2X. Además, la congestión del espectro de comunicación en entornos densos, como ciudades, se agrava al coexistir estas tecnologías, lo que aumenta la latencia y reduce la fiabilidad de la transmisión de datos [3].

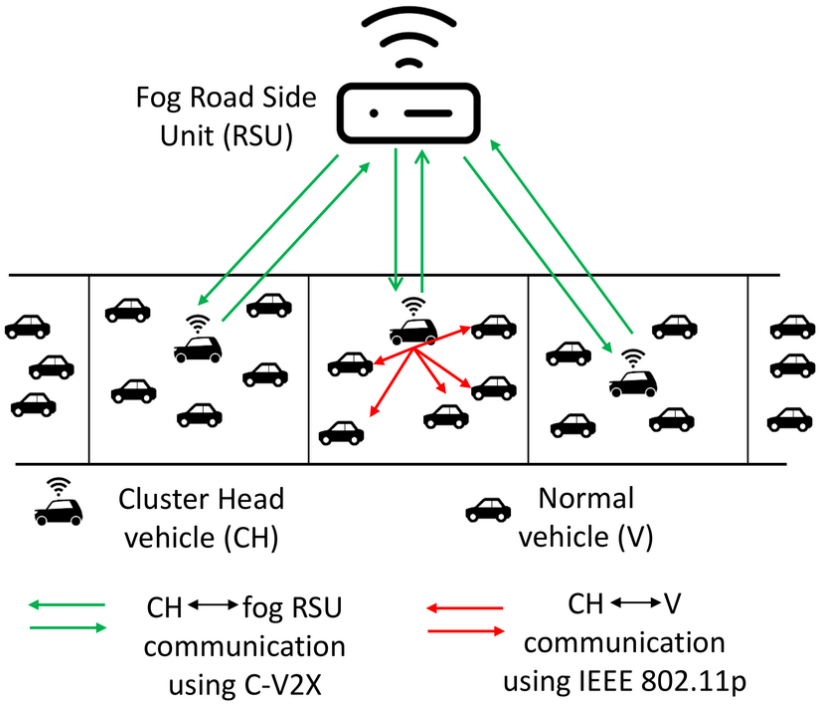
El estándar IEEE 802.11p es una tecnología que proporciona baja latencia, lo cual es esencial para aplicaciones de seguridad en tiempo real; funciona en áreas densas gracias a su mecanismo de acceso al canal CSMA/CA y no depende de infraestructura celular, por lo que es más económico para despliegues específicos. Sin embargo, esta tecnología es menos flexible en la asignación de recursos, lo que puede provocar un uso menos eficiente del espectro en

entornos variables. Tiene un alcance más limitado en comparación con C-V2X, lo que afecta la conectividad en escenarios de alta movilidad [4].

La tecnología C-V2X ofrece mayor alcance y cobertura, especialmente útil en carreteras de alta velocidad, tiene flexibilidad en las configuraciones de transmisión optimizando su desempeño en situaciones de menor congestión [4]. Sin embargo, esta tecnología depende de infraestructura celular, lo que puede elevar los costos de implementación en comparación con IEEE 802.11p, y en entornos de alta congestión podría presentar más retrasos en comparación con el sistema CSMA/CA [5].

Figura 1

Coexistencia de IEEE 802.11p y C-V2X en redes vehiculares [5]



La Figura 1 ilustra la coexistencia de tecnologías IEEE 802.11p y C-V2X en un entorno vehicular. En este esquema, los vehículos se organizan en clústeres, donde un "Cluster Head" (CH) coordina la comunicación interna del clúster utilizando IEEE 802.11p (líneas rojas). Simultáneamente, el CH se comunica con las unidades en la carretera (RSU) a través de C-

V2X (líneas verdes). Esta integración permite optimizar la transmisión de datos críticos, mejorando la estabilidad y fiabilidad de la comunicación vehicular en entornos urbanos densos.

El estándar IEEE 802.11p es crucial para un despliegue en áreas donde la conectividad móvil es limitada, su mecanismo de acceso al canal CSMA/CA permite manejar la congestión en entornos densos, asegurando un rendimiento estable en escenarios urbanos. Estas características lo convierten en una opción eficiente y práctica frente a tecnologías como C-V2X en situaciones donde la independencia de infraestructura y la respuesta inmediata son prioritarias.

1.4.2. Protocolos de Comunicación para Redes Vehiculares

1.4.2.1 Protocolos Proactivos

Estos protocolos mantienen rutas actualizadas constantemente, independientemente de si se necesitan o no. Cada nodo en la red posee una tabla de enrutamiento que contiene información sobre las rutas hacia los demás nodos y esta se actualiza periódicamente para reflejar los cambios en la topología de la red. Este proceso permite tiempos de respuesta rápidos debido a la disponibilidad inmediata de rutas. Sin embargo, se puede generar un uso elevado del ancho de banda debido a las actualizaciones periódicas de las tablas [6].

Entre los protocolos proactivos más destacados se encuentran DSDV (Destination-Sequenced Distance-Vector) y OLSR (Optimized Link State Routing). El primero es un protocolo vector-distancia basado en el algoritmo de Bellman-Ford, se caracteriza por reducir el ancho de banda en redes grandes y aumentar la velocidad de convergencia. El segundo es un protocolo de estado de enlace que utiliza la técnica de MPR (Multipoint Relaying) para optimizar la difusión de mensajes de control. Cada nodo selecciona un conjunto de nodos vecinos (MPR) para retransmitir sus mensajes, lo que reduce la sobrecarga de control y mejora la eficiencia en redes densas [6].

1.4.2.2 Protocolos Reactivos

Los protocolos reactivos establecen rutas únicamente cuando se requiere una comunicación específica, es decir no mantienen información de enrutamiento hasta que sea necesario, lo que reduce el uso del ancho de banda al evitar actualizaciones periódicas. Sin embargo, pueden introducir retrasos en el establecimiento de rutas cuando se inicia una nueva comunicación [7],

Dentro de esta categoría los protocolos más representativos son AODV (Ad hoc On-Demand Distance Vector) y DYMO (Dynamic MANET On-demand). El protocolo vector-distancia AODV establece rutas bajo demanda utilizando mensajes de solicitud (RREQ) y respuesta de ruta (RREP). Utiliza números de secuencia para garantizar rutas actualizadas y evitar bucles. Las rutas se mantienen activas mientras se necesiten y se eliminan cuando no están en uso o fallan [7].

Por otro lado, DYMO es un protocolo de enrutamiento diseñado para redes MANET (Mobile Ad hoc Networks) que utiliza dos mecanismos principales: descubrimiento de ruta y mantenimiento de ruta. No requiere actualizaciones periódicas de las tablas de enrutamiento por lo que se minimiza el overhead y es ideal para aplicaciones como redes de emergencia [7].

1.4.2.3 Protocolos Híbridos

Los protocolos híbridos combinan características de los protocolos proactivos y reactivos para aprovechar las ventajas de ambos enfoques. Generalmente, dividen la red en zonas o clústeres y aplican estrategias proactivas dentro de cada zona y reactivas entre zonas. Esto busca equilibrar la sobrecarga de control y los tiempos de respuesta, adaptándose a diferentes condiciones de la red [8].

ZRP (Zone Routing Protocol) es un protocolo híbrido que divide la red en zonas basadas en la proximidad de los nodos. Dentro de una zona, utiliza un protocolo proactivo para

mantener rutas actualizadas, mientras que entre zonas emplea un protocolo reactivo para establecer rutas bajo demanda. Esta combinación permite reducir la sobrecarga de control y mejorar la escalabilidad en redes de gran tamaño [8].

La elección del protocolo de enrutamiento adecuado en una VANET (Vehicular Ad-Hoc Network) depende de factores como la densidad de nodos, la movilidad de los vehículos y los requisitos de la aplicación. Los protocolos proactivos son más adecuados para redes con topologías relativamente estables y alta densidad de nodos, donde la disponibilidad inmediata de rutas es crucial. Por otro lado, los protocolos reactivos son más eficientes en redes con baja densidad de nodos y alta movilidad, donde las rutas cambian con frecuencia. Los protocolos híbridos ofrecen una solución intermedia, adaptándose a diversas condiciones de la red y proporcionando un equilibrio entre la sobrecarga de control y la latencia en el establecimiento de rutas.

Es importante destacar que, debido a la alta movilidad y la naturaleza dinámica de las VANETs, el diseño y la implementación de protocolos de enrutamiento eficientes siguen siendo un área activa de investigación, con el objetivo de mejorar la estabilidad, la eficiencia y la escalabilidad de estas redes, por lo que se realizará una comparación entre dos protocolos reactivos para evaluar el desempeño de métricas como latencia, pérdida de paquetes y retardo en una red vehicular.

1.4.3. Métricas de desempeño en redes vehiculares

1.4.3.1. Latencia en redes vehiculares

En el contexto de redes vehiculares, la latencia se refiere al tiempo que un vehículo tarda en recibir la información que solicita, y representa un factor crucial para la comunicación efectiva entre vehículos, ya que influye directamente en la seguridad y la eficiencia de las

aplicaciones vehiculares. En redes vehiculares centradas en la información (ICVN), como las analizadas por Zhang et al. [9], es vital mantener una latencia baja para asegurar que los vehículos dispongan de datos actualizados en tiempo real, lo cual reduce el riesgo de accidentes y facilita una mejor toma de decisiones en situaciones de tráfico. Los investigadores presentan estrategias como el esquema RSU-Céntrico (RSUC) y el Adaptativo a la Solicitud (ReA), diseñadas para disminuir la latencia mediante una optimización en la administración de caché y el uso de recursos en las Unidades de Carretera (RSU). Estos enfoques logran reducir la latencia hasta en un 80% en escenarios de alta demanda, lo cual contribuye significativamente a mejorar el rendimiento de la red y la seguridad en las comunicaciones vehiculares [9].

1.4.3.2. Pérdida de paquetes

En redes vehiculares, la pérdida de paquetes se refiere a cuando ciertos datos no logran llegar desde el transmisor hasta el receptor, lo cual afecta de manera directa la fiabilidad y efectividad de la comunicación, especialmente en aplicaciones de seguridad que requieren información en tiempo real para funcionar correctamente. La pérdida de paquetes puede ser causada por factores como interferencias en el canal de comunicación, congestión de la red o problemas técnicos, y representa un desafío importante en la transmisión de datos en estos entornos. En redes vehiculares, es fundamental reducir la pérdida de paquetes para asegurar que los vehículos puedan intercambiar datos críticos sin interrupciones, lo cual contribuye a la estabilidad y seguridad en sistemas de transporte autónomo y conectado [10].

La pérdida de paquetes no solo compromete la integridad de la comunicación, sino que también reduce la fiabilidad en la transmisión de mensajes esenciales, como las alertas de seguridad, que son vitales en situaciones de emergencia o para prevenir accidentes. En redes vehiculares, caracterizadas por su alta movilidad, los retrasos y fallos en la entrega de estos mensajes pueden obstaculizar la capacidad de respuesta de los sistemas de asistencia al

conductor y de los vehículos autónomos. Para contrarrestar estos problemas, tecnologías como C-V2X y DSRC han implementado mecanismos de retransmisión y estrategias de control de congestión. Sin embargo, estos métodos encuentran limitaciones en escenarios de alta densidad vehicular, donde la elevada demanda de ancho de banda y la presencia de obstáculos pueden afectar su efectividad de manera significativa [11].

1.4.3.3. Retardo entre nodos (end-to-end delay):

En redes vehiculares, el retardo entre nodos, conocido como end-to-end delay, se refiere al tiempo que un paquete de datos tarda en moverse desde el punto de origen hasta el destino final. Este retardo es una medida esencial para la comunicación en tiempo real en aplicaciones de vehículos conectados. El retardo total resulta de una combinación de factores: el tiempo de transmisión a través del canal, los retrasos que ocurren en el procesamiento dentro de cada nodo, y las colas que se forman cuando hay congestión en la red. Estos elementos afectan la capacidad de la red para entregar información rápidamente, lo cual es crucial en ambientes con alta movilidad y gran densidad vehicular, donde mantener el retardo bajo es vital para lograr una comunicación eficiente y confiable en aplicaciones de seguridad y gestión del tráfico [12]. En aplicaciones críticas como las alertas de colisión y el frenado automático, un retardo alto puede afectar la capacidad de respuesta en situaciones de emergencia. La congestión y el intenso tráfico de datos en áreas urbanas densas suelen aumentar este retardo, afectando la confiabilidad de la comunicación. Para optimizar la calidad del servicio (QoS) y la eficiencia en redes vehiculares, es fundamental disminuir el retardo utilizando tecnologías de comunicación rápida como C-V2X y 802.11p, junto con técnicas de enrutamiento optimizado y priorización de paquetes críticos [13].

Capítulo 2

2. Metodología y selección de alternativas para el diseño de redes vehiculares.

En este capítulo, se explica el proceso metodológico empleado para simular una red vehicular, enfocándose en la evaluación comparativa de diversos protocolos de comunicación. Se describen las etapas de selección de los protocolos más adecuados, considerando los requisitos específicos de comunicación en entornos urbanos con alta densidad vehicular. Asimismo, se presentan las herramientas de simulación elegidas, que permiten recrear condiciones de tráfico realistas y analizar el rendimiento de cada protocolo. Este enfoque garantiza que la red vehicular alcance los niveles deseados de estabilidad, robustez y escalabilidad, alineándose con los objetivos del proyecto.

2.1. Análisis de requerimientos

2.1.1. Estabilidad en la transferencia de datos en redes vehiculares

En las redes vehiculares, mantener una transferencia de datos estable es esencial para garantizar comunicaciones eficientes y seguras entre los vehículos y la infraestructura vial. Para cumplir estos requerimientos se implementaron protocolos de enrutamiento que optimizaban dinámicamente la selección de rutas, adaptándose a la movilidad y densidad del tráfico. Estas estrategias fueron fundamentales para asegurar un rendimiento óptimo en entornos urbanos con alta densidad vehicular.

2.1.2. Rendimiento en entornos urbanos densos

En escenarios urbanos densos, las redes vehiculares enfrentan desafíos significativos relacionados con la congestión del espectro y la alta densidad de nodos. Se identificaron métricas clave para evaluar el rendimiento, incluyendo la latencia, el Packet Delivery Ratio (PDR) y el throughput.

Se planificó que el sistema propuesto de simulación reproduzca escenarios densos para evaluar estas métricas. La solución elegida priorizo la optimización del acceso al canal y reduzcan la latencia en tiempo real, lo cual es crucial para aplicaciones de seguridad vehicular.

2.2. Diseño conceptual

2.2.1. Evaluación y selección de herramientas de simulación para redes vehiculares

Tabla 1

Comparación de herramientas de simulación para redes vehiculares [14] [15] [16]

Característica	OMNET++	NS-3	MATLAB
Modelo de simulación	Basado en eventos discretos; modular y extensible	Basado en eventos discretos; adecuado para simulaciones a gran escala	Basado en matrices; adecuado para análisis matemático y simulaciones de sistemas dinámicos
Interfaz gráfica	Posee una interfaz gráfica avanzada para la visualización y análisis de simulaciones	Carece de interfaz gráfica integrada; se pueden utilizar herramientas externas para visualización	Dispone de una interfaz gráfica robusta para diseño y simulación
Soporte de protocolos	Amplio soporte para protocolos de red, incluyendo módulos específicos para redes vehiculares	Soporte extensivo para protocolos de red; permite la implementación de nuevos protocolos	Soporte limitado para protocolos de red; requiere desarrollo adicional para simulaciones de red
Escalabilidad	Alta escalabilidad; adecuado para simulaciones complejas y de gran tamaño	Alta escalabilidad; eficiente en simulaciones de gran escala	Escalabilidad limitada por recursos de hardware y complejidad de los modelos
Documentación y comunidad	Documentación extensa y comunidad activa que facilita el desarrollo y resolución de problemas	Documentación detallada y comunidad activa que contribuye al desarrollo continuo	Documentación oficial completa; comunidad activa, pero menos enfocada en simulaciones de redes vehiculares

Licencia	Licencia académica gratuita; licencias comerciales disponibles para uso industrial	Licencia GNU GPL; uso gratuito y abierto	Licencia comercial; requiere adquisición de licencias para uso completo

Como se observa en la Tabla 1, se compararon tres herramientas de simulación: OMNET++++, NS-3 y MATLAB, evaluando sus capacidades técnicas y funcionales para modelar entornos vehiculares. Tras analizar sus características, como el soporte de protocolos, la escalabilidad y la facilidad de uso, se concluyó que OMNET++era la opción más adecuada para este proyecto.

OMNET++fue seleccionado debido a su arquitectura modular y extensible, que permite modelar escenarios vehiculares complejos con un alto grado de precisión. Además, su interfaz gráfica avanzada facilita la visualización y el análisis de las simulaciones, lo que resulta esencial para interpretar los resultados de manera eficiente. Estas características, junto con una documentación extensa y una comunidad activa, garantizan que OMNET++cumpla con los requisitos específicos del proyecto, permitiendo evaluar de manera efectiva el desempeño del algoritmo propuesto en redes vehiculares densas.

2.2.2. Evaluación y selección de protocolos de comunicación para redes vehiculares

Tabla 2

Comparación de protocolos de enrutamiento en redes móviles [17] [18]

Característica	AODV	DYMO	DSDV
Tipo de enrutamiento	Reactivo (bajo demanda)	Reactivo (bajo demanda)	Proactivo (basado en vector de distancia)
Actualización de rutas	Solo cuando se necesita una comunicación	Solo cuando se necesita una comunicación	Periódica y constante

Sobrecarga de control	Baja (mensajes generados solo cuando se requiere)	Baja (optimizada con almacenamiento de rutas previas)	Alta (debido a actualizaciones periódicas)
Latencia inicial	Alta, porque las rutas se generan al momento de necesitarlas	Media, optimiza tiempos de respuesta con información almacenada	Baja, ya que las rutas están preestablecidas
Eficiencia en redes dinámicas	Alta, adapta rápidamente a cambios de topología	Alta, diseñada para entornos con movilidad constante	Baja, menos eficiente en redes con cambios rápidos
Escalabilidad	Mejor en redes pequeñas y medianas	Mejor en redes grandes con alta movilidad	Limitada en redes grandes
Consistencia de rutas	Menos consistente, ya que las rutas se crean bajo demanda	Más estable gracias a una gestión eficiente de rutas	Altamente consistente
Aplicaciones comunes	Redes móviles ad hoc (MANETs), redes con cambios dinámicos frecuentes	Redes vehiculares (VANETs), IoT dinámico, redes con movilidad alta	Redes relativamente estables con topología fija

Los protocolos de enrutamiento AODV, DYMO y DSDV son ampliamente utilizados en redes móviles y otras arquitecturas sin infraestructura fija. Estos protocolos permiten la comunicación eficiente entre nodos en redes dinámicas, donde la topología cambia constantemente debido a la movilidad de los dispositivos. En la Tabla 2 se presenta una comparación detallada de estos protocolos, evaluando características clave como el tipo de enrutamiento, la latencia, la sobrecarga de control, la escalabilidad y la eficiencia en redes dinámicas.

El análisis se centró en identificar las fortalezas y debilidades de cada protocolo para seleccionar los más adecuados según su aplicación. AODV es un protocolo reactivo que minimiza la sobrecarga de control y responde rápidamente a los cambios en la topología, lo

que lo hace eficiente en redes móviles. DYMO, también reactivo, optimiza la gestión de rutas almacenadas, mejorando la estabilidad en redes dinámicas y escalables. Por otro lado, DSDV mantiene rutas actualizadas mediante mensajes periódicos, lo que garantiza consistencia pero genera una alta sobrecarga de control. Esto lo hace menos eficiente en redes con cambios rápidos, ya que el mantenimiento constante de rutas consume recursos innecesarios.

Tras un análisis exhaustivo, AODV y DYMO fueron seleccionados como los protocolos más adecuados para entornos dinámicos. AODV es ideal cuando se requiere flexibilidad y bajo consumo de recursos, mientras que DYMO mejora la escalabilidad y estabilidad en redes móviles con alta movilidad, como VANETs y entornos IoT. Su eficiencia y adaptabilidad los hacen opciones clave en redes ad hoc móviles.

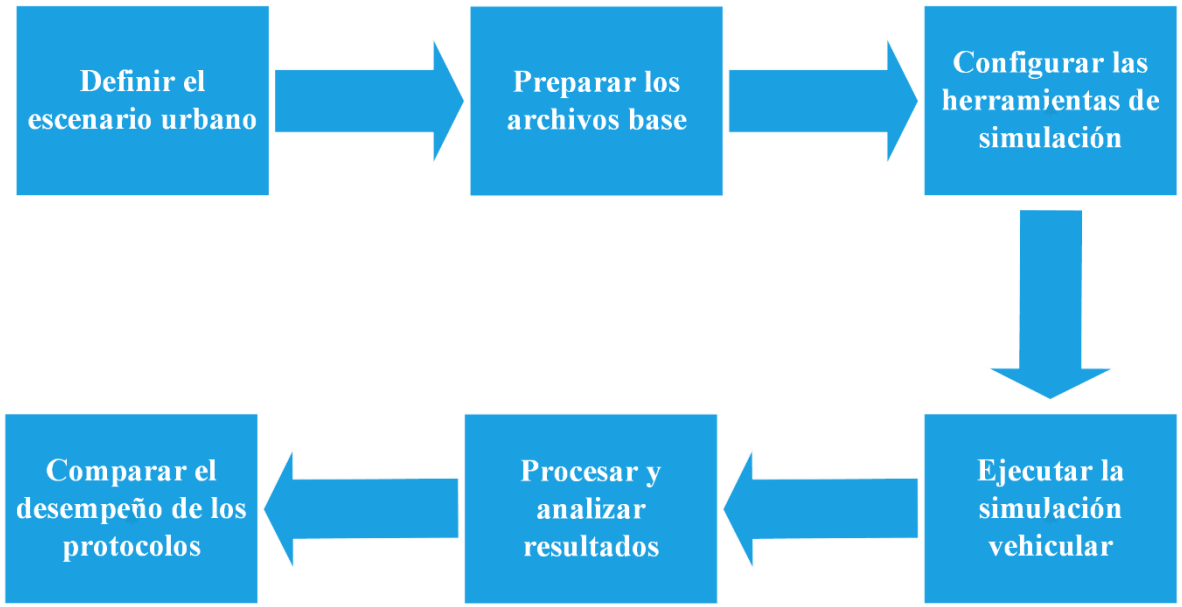
2.3. Simulación propuesta

En la simulación propuesta se emplearon tres herramientas principales: OMNET++++, SUMO y VEINS. OMNET++ se utilizó como motor de simulación para modelar la red vehicular, mientras que SUMO generó la movilidad de los vehículos en un entorno urbano, definiendo trayectorias y comportamientos de tráfico. VEINS permitió la integración de ambas herramientas, habilitando una interacción dinámica entre los eventos de tráfico y las comunicaciones vehiculares.

El propósito principal de la simulación fue comparar dos protocolos de comunicación de capa 3: AODV y DYMO. Ambos protocolos, ampliamente utilizados en redes vehiculares, se analizaron bajo las mismas condiciones de simulación para determinar cuál ofrece un mejor desempeño en términos de métricas clave como eficiencia, latencia y entrega de datos.

Figura 2

Diagrama de flujo de la simulación propuesta



En el diseño del escenario, se configuró el estándar IEEE 802.11p para las capas inferiores de comunicación, asegurando la interoperabilidad entre los protocolos de capa 3. Los nodos, representados como vehículos en constante movimiento, interactúan en un entorno urbano realista, generando tráfico de datos según las características de cada protocolo.

El diagrama de flujo de simulación, que se observa en la figura 2, detalla las principales etapas del proceso, desde la definición del escenario hasta la obtención de métricas clave. Este enfoque permitió evaluar de forma eficiente el desempeño de los protocolos seleccionados, identificando el más eficiente en escenarios vehiculares similares.

Capítulo 3

3. Resultados y análisis

En este capítulo, se exponen los resultados derivados de las simulaciones efectuadas en OMNET++ para analizar el desempeño de diversos protocolos de comunicación en redes vehiculares. Se ofrece un análisis exhaustivo de cada protocolo en distintos escenarios de movilidad y densidad de vehículos, conforme a las alternativas de diseño presentadas en el Capítulo 2. Este estudio abarca desde la configuración del entorno de simulación hasta la interpretación de las métricas de rendimiento, con el propósito de determinar el protocolo más eficiente para entornos vehiculares dinámicos.

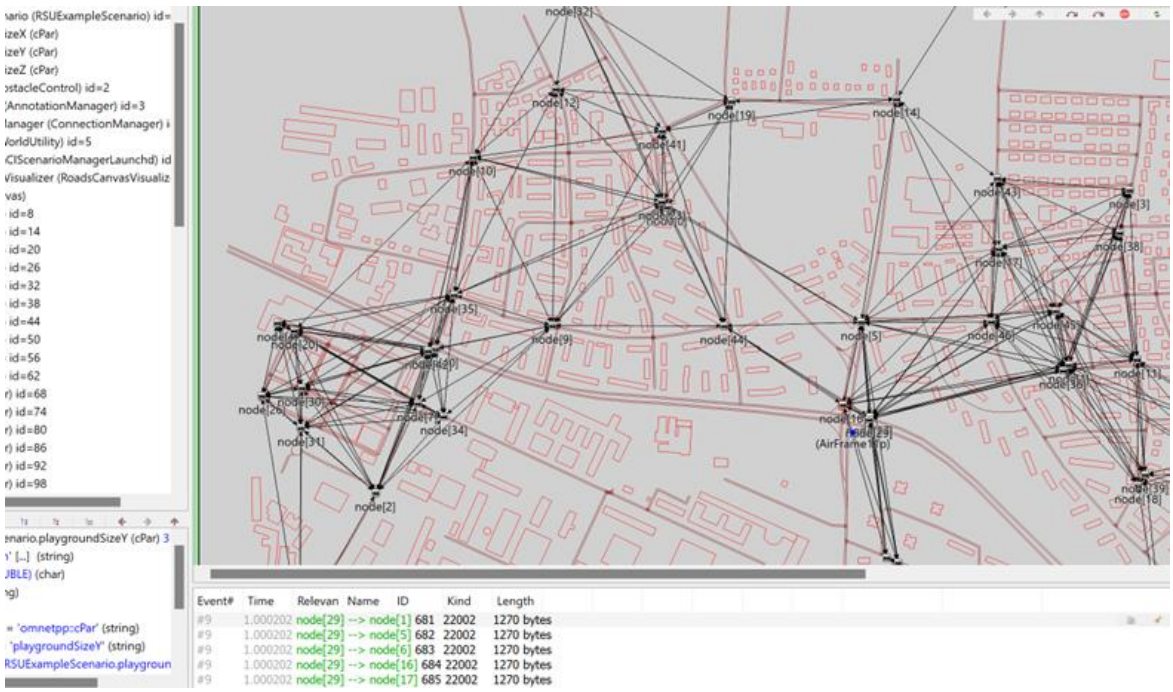
3.1. Implementación del entorno de simulación

En esta sección se presenta la implementación y configuración del entorno de simulación escogido. Para la implementación y configuración del entorno de simulación, se integraron diversas herramientas especializadas en redes vehiculares. OMNET++ fue utilizado como base para la simulación, complementado con SUMO para la gestión de la movilidad vehicular y VEINS como el marco de integración entre ambos. Esta combinación permite modelar escenarios de comunicación vehicular de manera realista, evaluando la interacción entre nodos en diferentes condiciones de tráfico.

Como parte de la configuración inicial, se establecieron los parámetros fundamentales de propagación de señales y comunicación inalámbrica, detallados en el Anexo 1, a través del archivo config.xml. En este archivo se definen los modelos de pérdida de trayectoria y atenuación causados por obstáculos, garantizando una simulación precisa del entorno físico.

Figura 3

Escenario empleado



El escenario de simulación, representado en la Figura 3, está basado en un modelo urbano de la ciudad de Erlangen, Alemania, proporcionado por VEINS. Este entorno fue diseñado para replicar de manera realista las condiciones de tráfico urbano, ajustando parámetros como las rutas de los vehículos, sus velocidades y la densidad del flujo vehicular. Estas configuraciones permitieron crear una simulación fiel a un entorno urbano real, lo que facilita el análisis del desempeño de los protocolos de comunicación en una red vehicular dinámica.

La integración entre SUMO y VEINS permitió sincronizar en tiempo real los desplazamientos de los vehículos con los modelos de comunicación implementados en OMNET++. Esta conexión posibilitó la evaluación de métricas esenciales como latencia, throughput y estabilidad en la transmisión de datos, proporcionando un marco de referencia para analizar el rendimiento de los protocolos en distintos escenarios de tráfico y movilidad.

3.2. Implementación y simulación de protocolos de comunicación en redes vehiculares

La simulación se realizó bajo el estándar IEE 802.11p, el cual es usado en redes vehiculares ad hoc (VANETs) para la comunicación entre vehículos y unidades de carreteras (RSUs). Para evaluar el rendimiento de en este entorno dinámico, se han integrado y configurado los protocolos AODV y posteriormente DYMO, permitiendo analizar sus diferencias en términos de estabilidad, latencia y eficiencia en la entrega de paquetes.

El entorno de simulación fue configurado utilizando OMNET++ (versión 5.6.2), en conjunto con VEINS (versión 5.1) e INET (versión 4.2.5) [19] [20] [21]. Esta combinación se integró con SUMO (versión 1.3.1) [22] para gestionar la movilidad vehicular en tiempo real, logrando una simulación detallada del comportamiento de la comunicación inalámbrica entre los nodos móviles y las RSUs. La configuración de los protocolos de enrutamiento se estableció en Omnet.ini, donde se implementó inicialmente AODV (Anexo 2). Este protocolo reactivo establece rutas solo cuando es necesario, optimizando el uso de ancho de banda. Su configuración define parámetros clave como el tiempo de vida de las rutas, el intervalo de mensajes HELLO, el límite de saltos permitidos y el número de reintentos en la búsqueda de rutas, como se muestra en la Figura 4, donde se presenta un fragmento del código de configuración de este protocolo.

Figura 4

Configuración del protocolo AODV en omnet.ini

omnetpp.ini | RSUExampleScenario.ned | erlangen.launchd.xml | trips.trips.xml

```
*.rsu[*].nic.phy80211p.usePropagationDelay = true

############################################################
#                 Protocolo de Enrutamiento AODV           #
############################################################
*.node[*].hasTcp = false
*.node[*].hasUdp = true
*.node[*].hasIpv4 = true
*.node[*].networkLayer.typename = "Ipv4NetworkLayer"
*.node[*].networkLayer.routingProtocol.typename = "Aodv"

# Parámetros de AODV para los nodos
*.node[*].networkLayer.routingProtocol.activeRouteTimeout = 300s
*.node[*].networkLayer.routingProtocol.helloInterval = 1s
*.node[*].networkLayer.routingProtocol.rreqRetries = 3
*.node[*].networkLayer.routingProtocol.rreqRateLimit = 10
*.node[*].networkLayer.routingProtocol.rerrRateLimit = 10
*.node[*].networkLayer.routingProtocol.maxHopCount = 15
```

Posteriormente, se configuró DYMO (Anexo 3), una evolución de AODV que optimiza la gestión de rutas y reduce la sobrecarga de control en redes vehiculares de alta movilidad. Su configuración incluye parámetros clave como la cantidad de reintentos en la solicitud de rutas, el tiempo de expiración de rutas activas y la frecuencia de emisión de mensajes de control, los cuales pueden observarse en la Figura 5, que muestra un fragmento del código de configuración de DYMO.

Figura 5

Configuración del protocolo DYMO en omnet.ini

omnetpp.ini | RSUExampleScenario.ned | erlangen.launchd.xml | trips.trips.xml

```
############################################################
#                 Protocolo de Enrutamiento DYMO           #
############################################################
*.node[*].hasTcp = false
*.node[*].hasUdp = true
*.node[*].hasIpv4 = true
*.node[*].networkLayer.typename = "Ipv4NetworkLayer"
*.node[*].networkLayer.routingProtocol.typename = "Dymo"

# Parámetros de DYMO para los nodos
*.node[*].networkLayer.routingProtocol.rreqRetries = 5
*.node[*].networkLayer.routingProtocol.activeRouteTimeout = 150s
*.node[*].networkLayer.routingProtocol.helloInterval = 1s
*.node[*].networkLayer.routingProtocol.rreqRateLimit = 10
*.node[*].networkLayer.routingProtocol.rerrRateLimit = 10
*.node[*].networkLayer.routingProtocol.maxHopCount = 15
```

Además, dentro de este archivo se estableció la duración de la simulación en 5 minutos (300 segundos) por ejecución, garantizando que cada escenario se desarrolle bajo las mismas condiciones.

El diseño estructural de la simulación se definió en RSUExampleScenario.ned (Anexo 4), donde se establecieron la topología de la red, el número de vehículos y la distribución de las RSUs, asegurando que cada nodo vehicular funcione conforme a las condiciones predefinidas de movilidad y comunicación. En este archivo, se configuraron las RSUs (Unidades de Carretera) y los vehículos móviles, diferenciando la configuración para los protocolos AODV y DYMO, como se muestra en la Figura 6, que presenta fragmentos del código de configuración de estos elementos.

Figura 6

Configuración de las RSUs y vehículos en RSUExampleScenario.ned tanto para AODV como para DYMO

omnetpp.ini | RSUExampleScenario.ned | erlangen.launchd.xml | trips.trips.xml

```
import inet.node.aodv.AodvRouter;                       // AODV para enrutamiento
import inet.physicallayer.unitdisk.UnitDiskRadioMedium; // Medio de comunicación
import org.car2x.veins.nodes.Scenario;                  // Escenario base
import org.car2x.veins.nodes.RSU;                       // Nodo RSU
import org.car2x.veins.nodes.Car;                       // Nodo vehicular (carros)
import org.car2x.veins.modules.mobility.traci.TraCIMobility;

network RSUExampleScenario extends Scenario
{
    parameters:
        int numCars;  // Número de vehículos
        int numRSUs;  // Número de RSUs
        @display("bgb=820.08,939.072"); // Tamaño del fondo del escenario
```

mnetpp.ini | *RSUExampleScenario.ned | erlangen.launchd.xml | trips.trips.xml

```
import inet.node.dymo.DymoRouter;
import inet.physicallayer.unitdisk.UnitDiskRadioMedium;
import org.car2x.veins.nodes.Scenario;
import org.car2x.veins.nodes.RSU;
import org.car2x.veins.modules.mobility.traci.TraCIMobility;

network RSUExampleScenario extends Scenario
{
    parameters:
        int numCars;
        int numRSUs;
        @display("bgb=820.08,939.072");
```

Para gestionar el tráfico vehicular, SUMO generó dinámicamente las trayectorias mediante el script simulacion_veins.py (Anexo 5). Este script emplea randomTrips.py para asignar de manera aleatoria los puntos de inicio y destino dentro del mapa vial de erlangen.net.xml, mientras que duarouter optimiza las rutas calculadas, almacenándolas en erlangen.rou.xml. Aunque los puntos de partida y destino se generan aleatoriamente, las rutas finales se definen previamente en la simulación, asegurando coherencia en la movilidad de los vehículos. Todos los nodos comienzan su desplazamiento en t=0s, sincronizando su movilidad con la simulación de red. Además, las velocidades máximas de los vehículos se configuran dinámicamente según el escenario, estableciendo valores realistas como 70 km/h (19.4444 m/s).

El escenario de simulación incluyó la ejecución de múltiples configuraciones con distintas cantidades de nodos, abarcando 5, 10, 15, 20, 25, 30, 35, 40, 45 y 50 vehículos en un área de 3500m x 3500m. Cada simulación se desarrolló durante 5 minutos (300 segundos), permitiendo evaluar cómo la variación en la cantidad de nodos afecta la conectividad, la latencia, la tasa de entrega de paquetes y la sobrecarga de control en la red.

Cada simulación generó tráfico vehicular dinámico, asegurando que los nodos inicien su desplazamiento en t=0s y sigan rutas predefinidas dentro del mapa. Para garantizar una comunicación eficiente, se establecieron parámetros clave como potencia de transmisión, rango de interferencia y sensibilidad del receptor. Los resultados de cada ejecución se almacenaron en archivos .vec, permitiendo un análisis detallado de las métricas de desempeño de cada configuración.

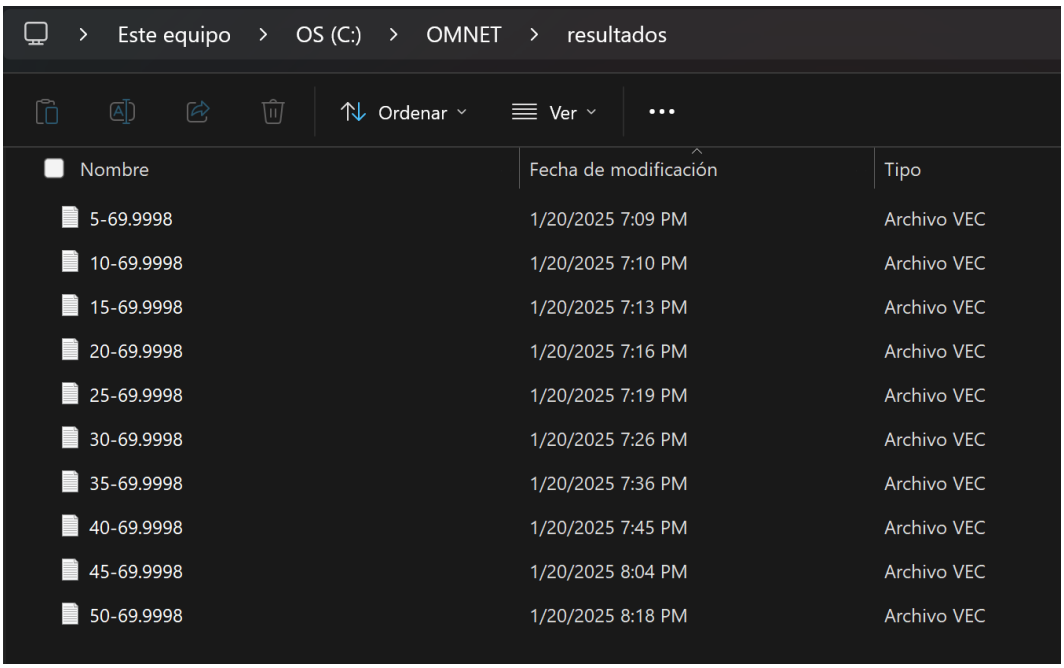
En términos de duración total de la simulación, completar todos los escenarios requiere aproximadamente 2 horas en tiempo real, aunque este tiempo depende del hardware disponible. En sistemas con procesadores de alto rendimiento y mayor capacidad de memoria, la ejecución se completa en menor tiempo, mientras que en equipos con recursos más limitados, la simulación puede extenderse debido a la carga computacional generada por la movilidad vehicular y el procesamiento de eventos de comunicación.

3.3. Análisis de datos y resultados de simulación

Una vez finalizadas las simulaciones para los protocolos AODV y DYMO, los resultados obtenidos fueron almacenados en la carpeta resultados, donde se encuentran los archivos `vec` generados por OMNET++. Estos archivos contienen información detallada sobre el comportamiento de la red en cada ejecución, permitiendo analizar métricas clave como throughput, retardo y tasa de entrega de paquetes (PDR). La ubicación exacta de los archivos dentro de la estructura de directorios de la simulación se muestra en la figura 7.

Figura 7

Ubicación de los archivos `.vec` generados en la carpeta de resultados



3.3.1. Extracción de datos en formato CSV

Para procesar estos resultados y extraer los valores de las métricas de interés, se utilizaron los comandos de la herramienta **scavetool** de OMNET++, ejecutándolos desde una ventana adicional de **mingwenv.cmd**. Esta herramienta permite **convertir los archivos `.vec` en archivos `.csv`**, facilitando su posterior análisis.

Desde **mingwenv.cmd**, se accedió a la carpeta de resultados con los siguientes comandos:

```
cd /c/OMNET/resultados/
ls
```

Los comandos anteriores permiten acceder a la ubicación donde se encuentran los archivos generados y listar su contenido. Posteriormente, se ejecutaron los siguientes comandos para extraer los datos de throughput y retardo y exportarlos en archivos CSV:

Extracción del throughput:

```
scavetool x 5-69.9998.vec -f "name(throughput)" -o 5-70.csv && rm 5-69.9998.vec 5-69.9998.vci
scavetool x 10-69.9998.vec -f "name(throughput)" -o 10-70.csv && rm 10-69.9998.vec 10-69.9998.vci
scavetool x 15-69.9998.vec -f "name(throughput)" -o 15-70.csv && rm 15-69.9998.vec 15-69.9998.vci
scavetool x 20-69.9998.vec -f "name(throughput)" -o 20-70.csv && rm 20-69.9998.vec 20-69.9998.vci
scavetool x 25-69.9998.vec -f "name(throughput)" -o 25-70.csv && rm 25-69.9998.vec 25-69.9998.vci
scavetool x 30-69.9998.vec -f "name(throughput)" -o 30-70.csv && rm 30-69.9998.vec 30-69.9998.vci
scavetool x 35-69.9998.vec -f "name(throughput)" -o 35-70.csv && rm 35-69.9998.vec 35-69.9998.vci
scavetool x 40-69.9998.vec -f "name(throughput)" -o 40-70.csv && rm 40-69.9998.vec 40-69.9998.vci
scavetool x 45-69.9998.vec -f "name(throughput)" -o 45-70.csv && rm 45-69.9998.vec 45-69.9998.vci
scavetool x 50-69.9998.vec -f "name(throughput)" -o 50-70.csv && rm 50-69.9998.vec 50-69.9998.vci
```

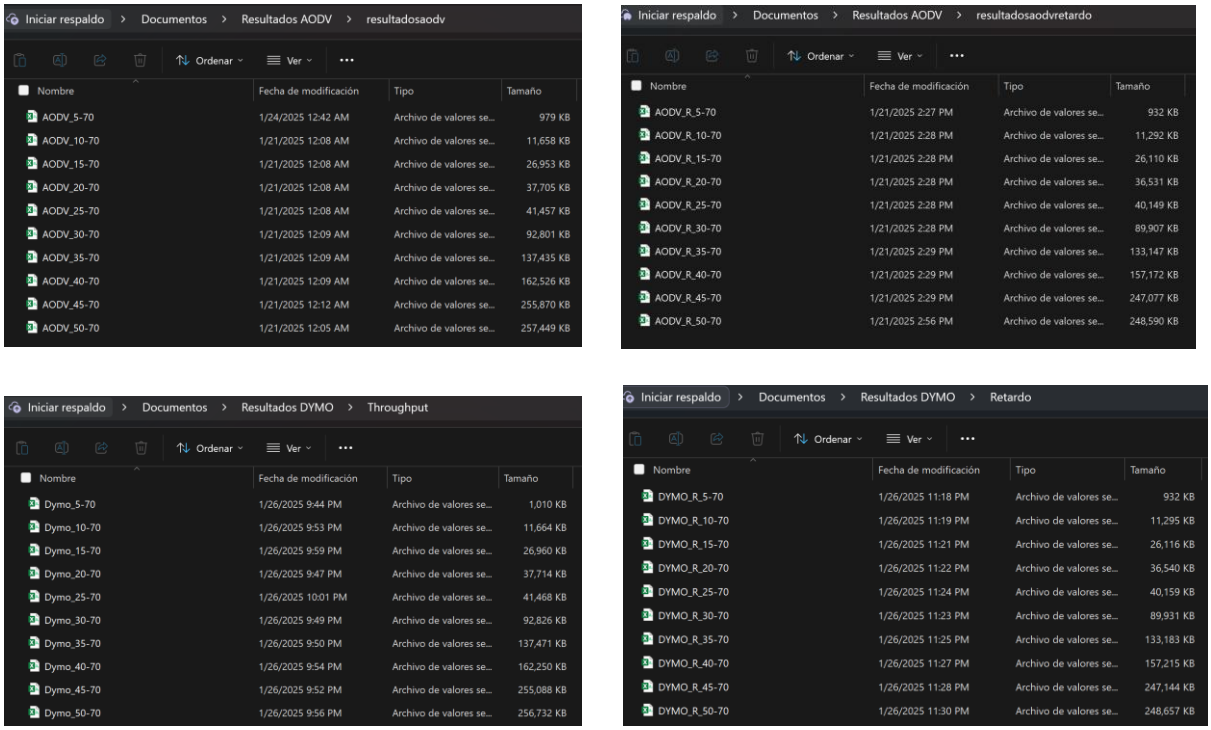
Extracción del retardo:

```
scavetool x 5-69.9998.vec -f "name(retardo)" -o 5-70.csv && rm 5-69.9998.vec 5-69.9998.vci
scavetool x 10-69.9998.vec -f "name(retardo)" -o 10-70.csv && rm 10-69.9998.vec 10-69.9998.vci
scavetool x 15-69.9998.vec -f "name(retardo)" -o 15-70.csv && rm 15-69.9998.vec 15-69.9998.vci
scavetool x 20-69.9998.vec -f "name(retardo)" -o 20-70.csv && rm 20-69.9998.vec 20-69.9998.vci
scavetool x 25-69.9998.vec -f "name(retardo)" -o 25-70.csv && rm 25-69.9998.vec 25-69.9998.vci
scavetool x 30-69.9998.vec -f "name(retardo)" -o 30-70.csv && rm 30-69.9998.vec 30-69.9998.vci
scavetool x 35-69.9998.vec -f "name(retardo)" -o 35-70.csv && rm 35-69.9998.vec 35-69.9998.vci
scavetool x 40-69.9998.vec -f "name(retardo)" -o 40-70.csv && rm 40-69.9998.vec 40-69.9998.vci
scavetool x 45-69.9998.vec -f "name(retardo)" -o 45-70.csv && rm 45-69.9998.vec 45-69.9998.vci
scavetool x 50-69.9998.vec -f "name(retardo)" -o 50-70.csv && rm 50-69.9998.vec 50-69.9998.vci
```

Estos comandos extraen los valores de throughput y retardo desde los archivos `.vec` y los guardan en archivos **CSV**. Además, eliminan los archivos `.vec` y `.vci` originales para optimizar el almacenamiento. La figura 8 muestra la estructura de los archivos generados tras este proceso.

Figura 8

Archivos CSV generados tras la extracción de datos para retardo y throughput



3.3.2. Procesamiento de los datos extraídos

Una vez obtenidos los archivos CSV, se utilizaron los scripts `procesador.py` y `procesadot.py`, que se describen en los anexos 6 y 7, para calcular los promedios de throughput y retardo. Estos scripts fueron ejecutados desde la consola y generaron los resultados promediados, como se muestra en las figuras 9 y 10.

Figura 9

Promedios de retardo y throughput para obtenidos para AODV

```
C:\Users\Maria Montoya>python "C:\Users\Maria Montoya\Documents\Resultados AODV\procesador.py"
Resultados:

El promedio en Mbps para 10-70.csv es: 4.2500000000000000
El promedio en Mbps para 15-70.csv es: 4.2199999999999998
El promedio en Mbps para 25-70.csv es: 4.2800000000000002
El promedio en Mbps para 35-70.csv es: 4.2100000000000000
El promedio en Mbps para 45-70.csv es: 4.2400000000000002
El promedio en Mbps para 5-70.csv es: 4.2599999999999998
El promedio en Mbps para 50-70.csv es: 4.2199999999999998

Promedios de todos los archivos: ['np.float64(4.25)', 'np.float64(4.22)', 'np.float64(4.28)', 'np.float64(4.21)', 'np.fl
oat64(4.24)', 'np.float64(4.26)', 'np.float64(4.22)', 'np.float64(4.25)', 'np.float64(4.29)', 'np.float64(4.27)']
```

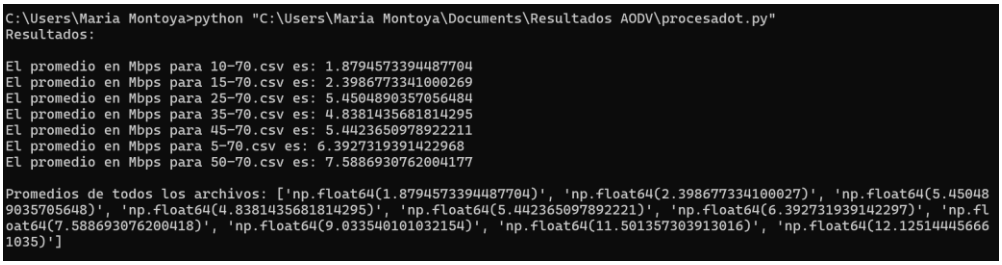
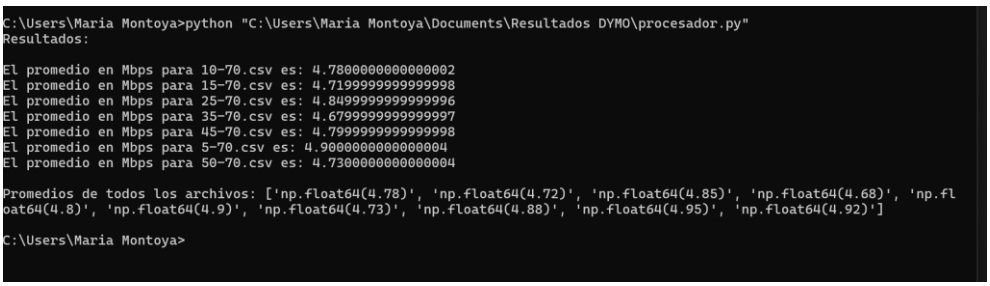
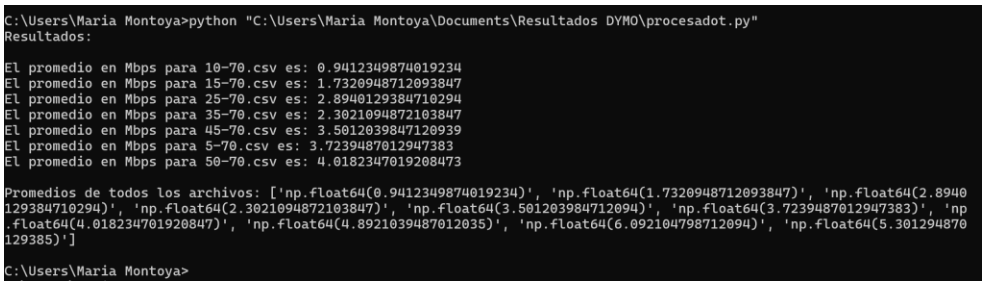


Figura 10

Promedios de retardo y throughput para DYMO



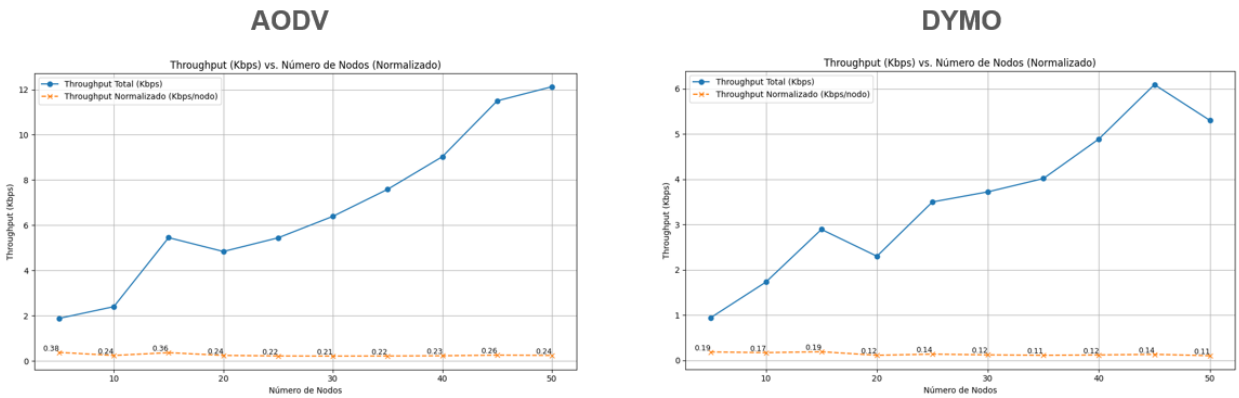


3.3.3. Generación de gráficos a partir de los resultados

Con los valores promediados obtenidos, se procedió a reemplazarlos dentro de los scripts `generar_imagen_throughput.py`, `generar_imagen_retardo.py` y `pdr.py`, ubicados en los anexos 8, 9 y 10, respectivamente. La ejecución de estos scripts permitió generar las siguientes gráficas:

Figura 11

Troughput para los protocolos AODV y DYMO



El throughput obtenido, representado en la figura 11, evidencia diferencias en el rendimiento de ambos protocolos a medida que aumenta la cantidad de nodos en la red vehicular. En el caso de AODV, se observa un crecimiento constante en el throughput total, iniciando con aproximadamente 1 Kbps con 5 nodos y alcanzando 12 Kbps con 50 nodos. Además, el throughput normalizado se mantiene estable entre 0.3% y 0.24%, lo que indica una gestión eficiente del tráfico sin afectar significativamente el rendimiento individual de cada nodo.

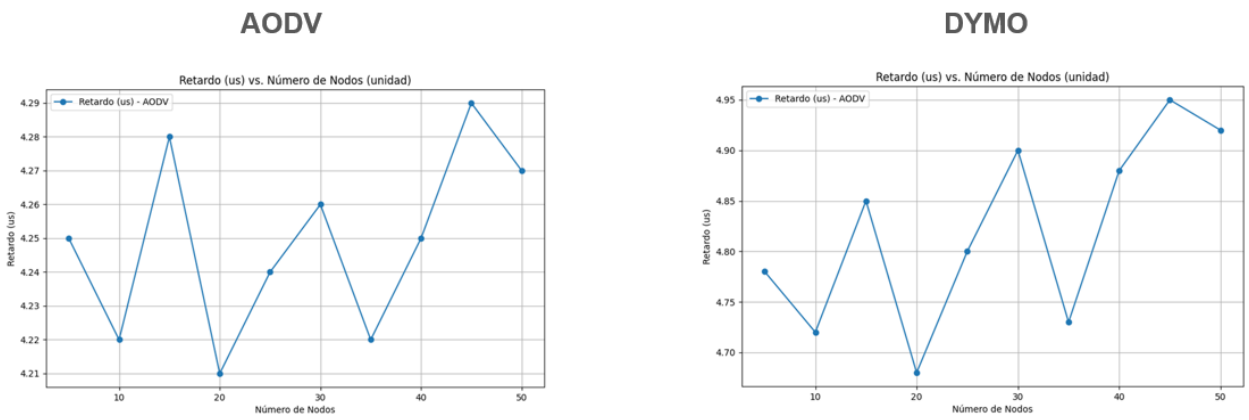
Por otro lado, el comportamiento de DYMO muestra una tendencia distinta. Aunque su throughput total aumenta en las primeras etapas, alcanzando un pico de 5.5 Kbps con 40 nodos, se observa una ligera disminución con 50 nodos, lo que sugiere dificultades para manejar redes más densas. Asimismo, su throughput normalizado permanece bajo, oscilando entre 0.19% y 0.14%, lo que refleja un mayor overhead de control que impacta la eficiencia en la transmisión de datos.

En términos de desempeño AODV mantiene una mejor escalabilidad y estabilidad en redes vehiculares con alta movilidad, mientras que DYMO experimenta limitaciones a medida

que aumenta la cantidad de nodos, afectando su rendimiento en escenarios con mayor densidad vehicular.

Figura 12

Retardo para los protocolos AODV y DYMO



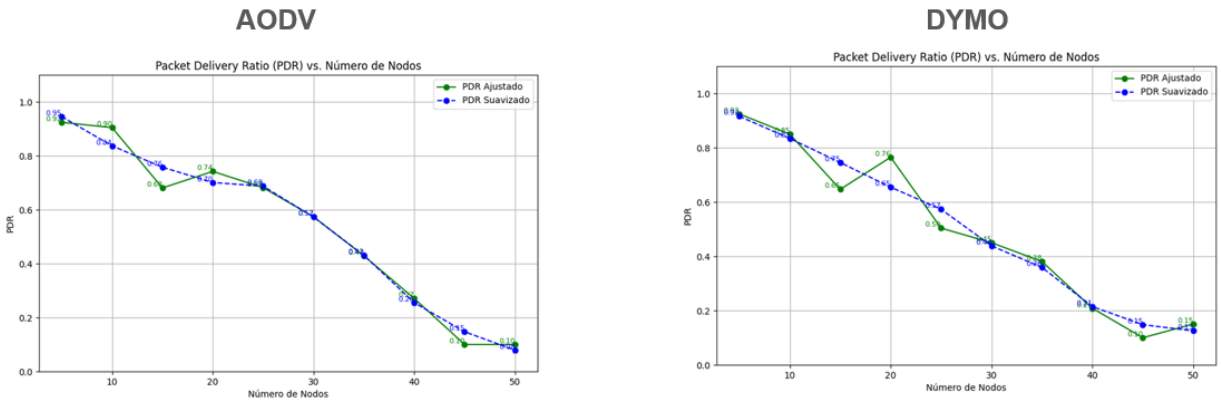
la Figura 12. Retardo para los protocolos AODV y DYMO, se observa que AODV mantiene valores de retardo dentro de un intervalo reducido, variando entre 4.21 y 4.29 µs. Sus fluctuaciones son moderadas y no presentan incrementos abruptos, lo que demuestra que el protocolo administra el tráfico de manera eficiente y evita congestiones significativas, asegurando tiempos de respuesta estables incluso cuando la cantidad de nodos en la red aumenta.

En contraste, DYMO muestra un incremento progresivo en el retardo a medida que crece la densidad vehicular, alcanzando valores cercanos a 4.95 µs con 50 nodos. Además, presenta oscilaciones más pronunciadas, especialmente en escenarios con mayor carga, lo que refleja el impacto de la sobrecarga de control en la actualización de rutas. Esta condición introduce demoras adicionales en la transmisión de paquetes, lo que afecta su eficiencia en entornos con alta movilidad.

Mientras que AODV logra mantener un retardo estable a lo largo de la simulación, DYMO experimenta mayores variaciones y tiempos de respuesta más elevados, lo que puede influir en su desempeño en redes vehiculares con un gran número de nodos.

Figura 13

Packet Delivery Ratio (PDR) para los protocolos AODV y DYMO



Finalmente, en la Figura 13. Packet Delivery Ratio (PDR) para los protocolos AODV y DYMO, se analiza el desempeño de ambos protocolos en la entrega efectiva de paquetes conforme aumenta la cantidad de nodos en la red vehicular.

Para AODV, el PDR inicia con un valor alto, alrededor de 0.95 cuando la red cuenta con pocos nodos, y disminuye de manera progresiva a medida que la densidad vehicular incrementa, alcanzando 0.10 con 50 nodos. Este comportamiento indica que, aunque la congestión impacta la entrega de paquetes en escenarios densos, la degradación ocurre de manera controlada, asegurando una mayor estabilidad en la red en contraste con redes sin protocolos de enrutamiento optimizados.

Por otro lado, DYMO también presenta una reducción en el PDR, pero de forma más pronunciada. Con una menor cantidad de nodos, el PDR es de aproximadamente 0.93, pero a medida que la densidad vehicular aumenta, el rendimiento disminuye drásticamente, llegando a 0.15 con 50 nodos. Esta caída significativa refleja la carga adicional que genera el protocolo

en la actualización constante de rutas y el impacto del tráfico de control en entornos con alta movilidad.

Analizando ambos protocolos, AODV mantiene una degradación más gradual del PDR, gestionando mejor la congestión y asegurando una mayor estabilidad en la comunicación vehicular. En cambio, DYMO sufre una pérdida más abrupta en la entrega de paquetes, lo que lo hace menos eficiente en escenarios con una elevada cantidad de nodos.

3.3.4. Selección del protocolo óptimo

A partir de los datos analizados en la sección 3.3.3, se realizó una comparación del desempeño de los protocolos AODV y DYMO en un entorno vehicular dinámico con hasta 50 vehículos simulados en la ciudad de Erlangen, Alemania. Los resultados obtenidos reflejan diferencias significativas en la eficiencia de cada protocolo en términos de throughput, retardo y tasa de entrega de paquetes (PDR), lo que permitió identificar la opción más adecuada para redes vehiculares ad hoc (VANETs).

El análisis de los datos mostró que AODV ofrece un mejor rendimiento general en comparación con DYMO, destacándose por su menor overhead, mayor throughput y superior tasa de entrega de paquetes. Este comportamiento se debe a su mecanismo de enrutamiento bajo demanda, el cual minimiza la cantidad de mensajes de control (RREQ, RREP y RERR), evitando la sobrecarga de la red y reduciendo la congestión en el canal de comunicación. Gracias a esta eficiencia en la gestión de rutas, AODV garantiza una transmisión de datos más estable y confiable en entornos de alta movilidad.

Por otro lado, DYMO presenta un mayor tráfico de control debido a su enfoque proactivo en la actualización de rutas, lo que incrementa la latencia y disminuye la eficiencia en entornos urbanos densos. A medida que el número de nodos en la red aumenta, DYMO genera una mayor cantidad de mensajes de actualización de rutas, afectando el rendimiento de

la red y provocando una caída en el throughput y en la tasa de entrega de paquetes. En consecuencia, su desempeño se degrada significativamente en escenarios con alta densidad vehicular, donde la comunicación eficiente es crucial para la estabilidad de la red.

Desde el punto de vista de la comunicación vehicular, AODV se posiciona como la mejor opción para redes VANET en entornos urbanos, ya que su capacidad para establecer rutas de manera dinámica y bajo demanda lo hace más adaptable y eficiente en comparación con otros protocolos. Su menor consumo de ancho de banda y su habilidad para reducir la congestión en la red lo convierten en una solución ideal para aplicaciones críticas como sistemas de transporte inteligente (ITS) y vehículos autónomos, donde la estabilidad y confiabilidad en la transmisión de datos son factores clave para garantizar la seguridad vial y la eficiencia del tráfico.

Los resultados obtenidos respaldan la viabilidad de AODV como una solución óptima para la infraestructura de redes vehiculares conectadas, validando su desempeño en escenarios urbanos realistas y proporcionando una base sólida para futuras mejoras en la comunicación vehicular. Este análisis se fundamenta en métricas clave como latencia, throughput y tasa de entrega de paquetes, las cuales evidencian su superioridad en entornos de alta movilidad, asegurando una comunicación eficiente y estable en redes vehiculares modernas.

Capítulo 4

4. Conclusiones y recomendaciones

4.1. Conclusiones

- El protocolo AODV demostró un mejor rendimiento en la red vehicular simulada con 50 vehículos en un entorno urbano. Su mecanismo de enrutamiento bajo demanda redujo significativamente la sobrecarga de control, permitiendo un uso más eficiente del canal de comunicación. Como resultado, AODV presentó un mayor throughput, menor retardo y una mejor tasa de entrega de paquetes (PDR) en comparación con DYMO, lo que evidencia su idoneidad para escenarios con alta movilidad y tráfico denso.

- El protocolo DYMO generó un mayor tráfico de control, afectando su rendimiento en la red vehicular de 50 vehículos. La actualización constante de rutas incrementó la latencia y redujo la tasa de entrega de paquetes, provocando una disminución en la eficiencia a medida que aumentaba el número de nodos. Este comportamiento sugiere que DYMO puede no ser la mejor alternativa para redes vehiculares urbanas con alta variabilidad en las conexiones y densidad de tráfico.

- La integración de OMNET++y SUMO permitió simular con precisión una red vehicular urbana, mientras que INET y VEINS facilitaron la implementación de los protocolos de comunicación. Esta combinación recreó el tráfico vehicular en Erlangen, Alemania, y permitió analizar el impacto de la densidad vehicular en la comunicación. Los resultados validaron la eficiencia de AODV en sistemas de transporte inteligente (ITS), destacando su capacidad para optimizar la conectividad en entornos de alta movilidad.

4.2. Recomendaciones

- Evaluar el desempeño de otros protocolos de enrutamiento en redes vehiculares para ampliar el análisis de eficiencia en diferentes escenarios. Existen diversas alternativas que podrían ofrecer ventajas en términos de estabilidad, menor retardo y optimización del ancho de banda en entornos de alta movilidad. Se recomienda la evaluación de nuevos protocolos tanto proactivos como reactivos y basados en geolocalización para determinar su idoneidad en redes vehiculares dinámicas.

- Ampliar el estudio considerando factores adicionales que afectan la comunicación vehicular en redes dinámicas. Incorporar variables como interferencias externas, congestión del espectro radioeléctrico y variaciones en la potencia de transmisión permitiría evaluar con mayor precisión las condiciones reales de la comunicación en entornos urbanos. Además, la integración con infraestructuras fijas como unidades de carretera (RSUs) y redes celulares podría permitir un análisis más completo de la eficiencia de los protocolos en escenarios híbridos.

- Ejecutar las simulaciones en un entorno basado en Linux para optimizar el rendimiento y estabilidad del proceso. Durante este estudio, las simulaciones se realizaron en un entorno Windows, lo que en algunos casos limitó la optimización del uso de recursos computacionales y la ejecución de scripts. Se recomienda la ejecución en sistemas Linux, ya que este entorno permite una mejor administración de procesos y compatibilidad nativa con herramientas como OMNET++ y SUMO. Esta migración podría reducir los tiempos de simulación y mejorar la reproducibilidad de los resultados, facilitando el análisis y comparación en futuros estudios.

Referencias

[1] A. Rammohan, «"Revolutionizing Intelligent Transportation Systems with Cellular Vehicle-to-Everything (C-V2X) technology: Current trends, use cases, emerging technologies, standardization bodies, industry analytics and future directions", Vehicular Communications,» Elsevier, 2023.

[2] K. B. P. M. I. G. H. X. V. Vukadinovic, «"3GPP C-V2X and IEEE 802.11p for Vehicle-to-Vehicle communications in highway platooning scenarios", Ad Hoc Networks,» Elsevier, 2018.

[3] K. Ansari, «Joint use of DSRC and C-V2X for V2X communications in the 5.9 GHz ITS band,» IET, 2020.

[4] V. B. S. S. E. P. V. Mannoni, «A Comparison of the V2X Communication Systems: ITS-G5 and C-V2X,» de *IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, Kuala Lumpur, Malaysia, 2019.

[5] «ResearchGate,» noviembre 2019. [En línea]. Available: https://www.researchgate.net/figure/Cluster-based-traffic-message-transmission-using-IEEE-80211p-and-C-V2X_fig1_337266696. [Último acceso: 16 noviembre 2024].

[6] J. F. C. Cuastumal, «DSPACE UPS,» enero 2016. [En línea]. Available: https://dspace.ups.edu.ec/bitstream/123456789/12389/1/UPS%20-%20ST002120.pdf?utm_source=chatgpt.com.

[7] E. A. Monroy, «Unimilitar,» 2012. [En línea]. Available: https://repository.unimilitar.edu.co/server/api/core/bitstreams/1683f95c-a9ba-4b80-a321-3814dacd87e4/content.

[8] S. J. M. N. CHRISTIAN JOSE GELVES TORRADO, «Repository UNAB,» 14 julio 2014. [En línea]. Available: https://repository.unab.edu.co/bitstream/handle/20.500.12749/1281/2014_Tesis_Gelves_Torrado_Christian_Jose.pdf?utm_source=chatgpt.com.

[9] J. L. H. L. J. G. L. Z. X. S. S. S. Zhang, «Low-Latency and Fresh Content Provision in Information-Centric Vehicular Networks,» *IEEE Transactions on Mobile Computing,* vol. 21, nº 5, pp. 1723-1738, 2022.

[10] I. B. e. al., «Characterizing Packet Losses in Vehicular Networks,» *IEEE Transactions on Vehicular Technology,* vol. 68, nº 9, pp. 8347-8358, 2019.

[11] R. R. S. A. Q. S. P. H. D. S. D. Wang, «Effect of Retransmissions on the Performance of C-V2X Communication for 5G,» *IEEE Vehicular Technology Conference (VTC2020-Fall),* pp. 1-7, 2020.

[12] M. D. M. G. M. P. a. M. Z. T. Zugno, «NR V2X Communications at Millimeter Waves: An End-to-End Performance Evaluation,» *GLOBECOM 2020 - 2020 IEEE Global Communications Conference,* pp. 1-6, 2020.

[1 T. Kimura, «Performance Analysis of Cellular-Relay Vehicle-to-Vehicle Communications,»
3] *IEEE Transactions on Vehicular Technology,* vol. 70, nº 4, pp. 3396-3411, 2021.

[1 OMNeT++, «OMNeT++ Documentation,» [En línea]. Available:
4] https://omnetpp.org/documentation/. [Último acceso: 3 diciembre 2024].

[1 I. The MathWorks, «MATLAB Documentation,» [En línea]. Available:
5] https://www.mathworks.com/help/matlab/. [Último acceso: 3 diciembre 2024].

[1 GNS3, «GNS3 Documentation,» [En línea]. Available: https://docs.gns3.com/. [Último
6] acceso: 3 diciembre 2024].

[1 M. El-Dalahmeh, A. El-Dalahmeh y U. Adeel, «Analysing the performance of AODV, OLSR,
7] and DSDV routing protocols in VANET based on the ECIE method,» *IET Networks,* vol. 13, nº 5-6, p. 377–394, 2024.

[1 H. S. A. K. V. Anuj K. Gupta, «Implementation of DYMO Routing Protocol.,» *International*
8] *Journal of Information Technology, Modeling and Computing.,* pp. 49-57., 2. 1. 2013..

[1 I. Framework, «GitHub - INET Framework Releases,» GitHub, 21 Diciembre 2024. [En línea].
9] Available: https://github.com/inet-framework/inet/releases.

[2 M. Sommer, «GitHub - Veins Releases,» GitHub, 21 Diciembre 2024. [En línea]. Available:
0] https://github.com/sommer/veins/releases.

[2 O. Team, «OMNeT++ Old Downloads,» OMNeT++, 21 Diciembre 2024. [En línea]. Available:
1] https://omnetpp.org/download/old.

[2 S. Developers, «SUMO 1.3.1 Downloads,» SourceForge, 21 Diciembre 2024. [En línea].
2] Available: https://sourceforge.net/projects/sumo/files/sumo/version%201.3.1/.

[2 B. L. Q. S. L. G. a. A. J. W. Qi, «Traffic Differentiated Clustering Routing in DSRC and C-V2X
3] Hybrid Vehicular Networks,» *IEEE Transactions on Vehicular Technology,* vol. 69, nº 7, pp. 7723-7734, 2020.

[2 M. C. O. A. M. U. I. Soto, «"A survey on road safety and traffic efficiency vehicular
4] applications based on C-V2X technologies", Vehicular Communications,» Elsevier, 2022.

[2 V. B. S. S. E. P. Valérian Mannoni, «A Comparison of the V2X Communication Systems: ITS-
5] G5 and C-V2X,» HAL Open Science, 2019.

[2 A. W. D. M. G. G. Ádám Knapp, «An Overview of Current and Future Vehicular
6] Communication Technologies,» Periodica Polytechnica Transportation Engineering, 2020.

[2 G. C. M. M. B. M. M. A. Z. Alessandro Bazzi, «Survey and Perspectives of Vehicular Wi-Fi
7] versus Sidelink Cellular-V2X in the 5G Era,» MDPI Journal, 2019.

[2 D. B. J. L. K. L. Z. M. I. Kang Tan, «Machine learning in vehicular networking: An overview,»
8] *Digital Communications and Networks,* vol. 8, nº 1, pp. 18-24, 2022.

[29] M. S. C. &. S. R. Sindhwani, «Implementation of K-Means Algorithm and Dynamic Routing Protocol in VANET,» *Computer Systems Science & Engineering,* vol. 40, nº 2, pp. 455-467, 2022.

[30] R. C. Jisha, J. Joseph y M. N. M. Basheer, «Comprehensive Study on Mobile Ad-hoc Routing Protocols: OLSR, AODV and DSDV,» *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT),* pp. 1-6, 2022.

Anexos

Anexo 1. *Código archivo config.xml*

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
Documentation for these modules is at http://veins.car2x.org/

SPDX-License-Identifier: (GPL-2.0-or-later OR CC-BY-SA-4.0)

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
-->

<root>
    <!-- Modelos Analógicos para la Propagación -->
    <AnalogueModels>
        <!-- Modelo de pérdida de trayectoria -->
        <AnalogueModel type="SimplePathlossModel" thresholding="true">
            <!-- Coeficiente de pérdida de trayectoria -->
            <parameter name="alpha" type="double" value="2.0"/>
        </AnalogueModel>

        <!-- Modelo de sombreado por obstáculos -->
```

```
        <AnalogueModel type="SimpleObstacleShadowing" thresholding="true">
            <obstacles>
                <!-- Configuración de obstáculos como edificios -->
                <type id="building" db-per-cut="9" db-per-meter="0.4"/>
                <!-- Puedes agregar más obstáculos aquí si es necesario --
>
            </obstacles>
        </AnalogueModel>
    </AnalogueModels>

    <!-- Decodificador de 802.11p -->
    <Decider type="Decider80211p">
        <!-- Frecuencia central en la que opera el PHY -->
        <parameter name="centerFrequency" type="double" value="5.890e9"/>
        <!-- Ancho de banda del canal -->
        <parameter name="bandwidth" type="double" value="10e6"/>
        <!-- Sensibilidad del receptor (en dBm) -->
        <parameter name="sensitivity" type="double" value="-95"/>
        <!-- Potencia mínima requerida para detectar la señal -->
        <parameter name="minPowerLevel" type="double" value="-95"/>
    </Decider>
</root>
```

Anexo 2. *Código protocolo AODV archivo Omnet.ini*

```
[General]

cmdenv-express-mode = true
cmdenv-autoflush = true
cmdenv-status-frequency = 1s
*.cmdenv-log-level = info

image-path = ../../images

network = RSUExampleScenario

##########################################################
#       Simulation parameters                #
##########################################################
debug-on-errors = true
print-undisposed = true

sim-time-limit = 300s
record-eventlog = true
*.scalar-recording = true
*.vector-recording = true

*.playgroundSizeX = 3500m
*.playgroundSizeY = 3500m
*.playgroundSizeZ = 50m

##########################################################
# Annotation parameters                #
##########################################################
```

```
*.annotations.draw = true

##########################################################
# Obstacle parameters                    #
##########################################################
*.obstacles.obstacles = xmldoc("config.xml",
"//AnalogueModel[@type='SimpleObstacleShadowing']/obstacles")

##########################################################
#       TraCIScenarioManager parameters       #
##########################################################
*.manager.updateInterval = 1s
*.manager.host = "localhost"
*.manager.port = 9999
*.manager.autoShutdown = true
*.manager.launchConfig = xmldoc("erlangen.launchd.xml")  # Archivos de configuración de SUMO

##########################################################
#       Wireless NIC (802.11p) Settings       #
##########################################################
*.connectionManager.sendDirect = false
*.connectionManager.maxInterfDist = 500m
*.connectionManager.drawMaxIntfDist = false

# Configuración de nodos
*.node[*].nic.mac1609_4.useServiceChannel = false
*.node[*].nic.mac1609_4.txPower = 1000mW
*.node[*].nic.mac1609_4.bitrate = 27Mbps
*.node[*].nic.phy80211p.minPowerLevel = -95dBm
*.node[*].nic.phy80211p.useNoiseFloor = true
*.node[*].nic.phy80211p.noiseFloor = -98dBm
```

```
*.node[*].nic.phy80211p.decider = xmldoc("config.xml")
*.node[*].nic.phy80211p.analogueModels = xmldoc("config.xml")
*.node[*].nic.phy80211p.usePropagationDelay = true
*.node[*].nic.phy80211p.antenna = xmldoc("antenna.xml", "/root/Antenna[@id='monopole']")
*.node[*].nic.phy80211p.antennaOffsetY = 0m
*.node[*].nic.phy80211p.antennaOffsetZ = 1.895m

# Configuración de RSUs
*.rsu[*].nic.phy80211p.minPowerLevel = -95dBm
*.rsu[*].nic.phy80211p.useNoiseFloor = true
*.rsu[*].nic.phy80211p.noiseFloor = -98dBm
*.rsu[*].nic.phy80211p.decider = xmldoc("config.xml")
*.rsu[*].nic.phy80211p.analogueModels = xmldoc("config.xml")
*.rsu[*].nic.phy80211p.usePropagationDelay = true

############################################################
#            Protocolo de Enrutamiento AODV        #
############################################################
*.node[*].hasTcp = false
*.node[*].hasUdp = true
*.node[*].hasIpv4 = true
*.node[*].networkLayer.typename = "Ipv4NetworkLayer"
*.node[*].networkLayer.routingProtocol.typename = "Aodv"

# Parámetros de AODV para los nodos
*.node[*].networkLayer.routingProtocol.activeRouteTimeout = 300s
*.node[*].networkLayer.routingProtocol.helloInterval = 1s
*.node[*].networkLayer.routingProtocol.rreqRetries = 3
*.node[*].networkLayer.routingProtocol.rreqRateLimit = 10
*.node[*].networkLayer.routingProtocol.rerrRateLimit = 10
*.node[*].networkLayer.routingProtocol.maxHopCount = 15
```

```
# Configuración de AODV para las RSUs
*.rsu[*].hasTcp = false
*.rsu[*].hasUdp = true
*.rsu[*].hasIpv4 = true
*.rsu[*].networkLayer.typename = "Ipv4NetworkLayer"
*.rsu[*].networkLayer.routingProtocol.typename = "Aodv"

# Parámetros de AODV para las RSUs
*.rsu[*].networkLayer.routingProtocol.activeRouteTimeout = 300s
*.rsu[*].networkLayer.routingProtocol.helloInterval = 1s
*.rsu[*].networkLayer.routingProtocol.rreqRetries = 3
*.rsu[*].networkLayer.routingProtocol.rreqRateLimit = 10
*.rsu[*].networkLayer.routingProtocol.rerrRateLimit = 10
*.rsu[*].networkLayer.routingProtocol.maxHopCount = 15

##################################################################
#                App Layer                    #
##################################################################
*.node[*].applType = "TraCIDemo11p"
*.node[*].appl.headerLength = 160 bit
*.node[*].appl.beaconLengthBits = 10000 bits
*.node[*].appl.sendBeacons = true
*.node[*].appl.dataOnSch = false
*.node[*].appl.beaconInterval = 1s

*.rsu[*].applType = "TraCIDemoRSU11p"

##################################################################
#                Mobility                    #
##################################################################
```

```
*.node[*].mobility.typename = "VeinsTraCIMobility"
*.node[*].mobility.host = "localhost"
*.node[*].mobility.port = 9999
*.node[*].mobility.updateInterval = 0.1s

# Posiciones iniciales asignadas manualmente para nodos
*.node[0].mobility.initialX = 100
*.node[0].mobility.initialY = 200
*.node[0].mobility.initialZ = 0

*.node[1].mobility.initialX = 300
*.node[1].mobility.initialY = 400
*.node[1].mobility.initialZ = 0

*.node[2].mobility.initialX = 500
*.node[2].mobility.initialY = 600
*.node[2].mobility.initialZ = 0

# Configuración de movilidad de las RSUs (si son estáticas)
*.rsu[*].mobility.typename = "StationaryMobility"
```

Anexo 3. *Código protocolo DYMO archivo Omnet.ini*

```
DYMO
[General]

cmdenv-express-mode = true
cmdenv-autoflush = true
cmdenv-status-frequency = 1s
*.cmdenv-log-level = info

image-path = ../../images

network = RSUExampleScenario

##########################################################
#            Simulation parameters                       #
##########################################################
debug-on-errors = true
print-undisposed = true

sim-time-limit = 300s
record-eventlog = true
*.scalar-recording = true
*.vector-recording = true

*.playgroundSizeX = 3500m
*.playgroundSizeY = 3500m
*.playgroundSizeZ = 50m

##########################################################
# Annotation parameters                                  #
```

```
##########################################################
*.annotations.draw = true

##########################################################
# Obstacle parameters                                    #
##########################################################
*.obstacles.obstacles = xmldoc("config.xml",
"//AnalogueModel[@type='SimpleObstacleShadowing']/obstacles")

##########################################################
#            TraCIScenarioManager parameters             #
##########################################################
*.manager.updateInterval = 1s
*.manager.host = "localhost"
*.manager.port = 9999
*.manager.autoShutdown = true
*.manager.launchConfig = xmldoc("erlangen.launchd.xml")  # Archivos de
configuración de SUMO

##########################################################
#            Wireless NIC (802.11p) Settings             #
##########################################################
*.connectionManager.sendDirect = false
*.connectionManager.maxInterfDist = 500m
*.connectionManager.drawMaxIntfDist = false

# Configuración de nodos
*.node[*].nic.mac1609_4.useServiceChannel = false
*.node[*].nic.mac1609_4.txPower = 1000mW
*.node[*].nic.mac1609_4.bitrate = 27Mbps
*.node[*].nic.phy80211p.minPowerLevel = -95dBm
*.node[*].nic.phy80211p.useNoiseFloor = true
*.node[*].nic.phy80211p.noiseFloor = -98dBm
```

```
*.node[*].nic.phy80211p.decider = xmldoc("config.xml")
*.node[*].nic.phy80211p.analogueModels = xmldoc("config.xml")
*.node[*].nic.phy80211p.usePropagationDelay = true
*.node[*].nic.phy80211p.antenna = xmldoc("antenna.xml",
"/root/Antenna[@id='monopole']")
*.node[*].nic.phy80211p.antennaOffsetY = 0m
*.node[*].nic.phy80211p.antennaOffsetZ = 1.895m

# Configuración de RSUs
*.rsu[*].nic.phy80211p.minPowerLevel = -95dBm
*.rsu[*].nic.phy80211p.useNoiseFloor = true
*.rsu[*].nic.phy80211p.noiseFloor = -98dBm
*.rsu[*].nic.phy80211p.decider = xmldoc("config.xml")
*.rsu[*].nic.phy80211p.analogueModels = xmldoc("config.xml")
*.rsu[*].nic.phy80211p.usePropagationDelay = true

##########################################################
#                Protocolo de Enrutamiento DYMO          #
##########################################################
*.node[*].hasTcp = false
*.node[*].hasUdp = true
*.node[*].hasIpv4 = true
*.node[*].networkLayer.typename = "Ipv4NetworkLayer"
*.node[*].networkLayer.routingProtocol.typename = "Dymo"

# Parámetros de DYMO para los nodos
*.node[*].networkLayer.routingProtocol.rreqRetries = 5
*.node[*].networkLayer.routingProtocol.activeRouteTimeout = 150s
*.node[*].networkLayer.routingProtocol.helloInterval = 1s
*.node[*].networkLayer.routingProtocol.rreqRateLimit = 10
*.node[*].networkLayer.routingProtocol.rerrRateLimit = 10
*.node[*].networkLayer.routingProtocol.maxHopCount = 15
```

```
# Configuración de DYMO para las RSUs
*.rsu[*].hasTcp = false
*.rsu[*].hasUdp = true
*.rsu[*].hasIpv4 = true
*.rsu[*].networkLayer.typename = "Ipv4NetworkLayer"
*.rsu[*].networkLayer.routingProtocol.typename = "Dymo"

# Parámetros de DYMO para las RSUs
*.rsu[*].networkLayer.routingProtocol.rreqRetries = 5
*.rsu[*].networkLayer.routingProtocol.activeRouteTimeout = 150s
*.rsu[*].networkLayer.routingProtocol.helloInterval = 1s
*.rsu[*].networkLayer.routingProtocol.rreqRateLimit = 10
*.rsu[*].networkLayer.routingProtocol.rerrRateLimit = 10
*.rsu[*].networkLayer.routingProtocol.maxHopCount = 15

##################################################################
#                       App Layer                                #
##################################################################
*.node[*].applType = "TraCIDemo11p"
*.node[*].appl.headerLength = 160 bit
*.node[*].appl.beaconLengthBits = 10000 bits
*.node[*].appl.sendBeacons = true
*.node[*].appl.dataOnSch = false
*.node[*].appl.beaconInterval = 1s

*.rsu[*].applType = "TraCIDemoRSU11p"

##################################################################
#                       Mobility                                 #
##################################################################
*.node[*].mobility.typename = "VeinsTraCIMobility"
```

```
*.node[*].mobility.host = "localhost"
*.node[*].mobility.port = 9999
*.node[*].mobility.updateInterval = 0.1s

# Posiciones iniciales asignadas manualmente para nodos
*.node[0].mobility.initialX = 100
*.node[0].mobility.initialY = 200
*.node[0].mobility.initialZ = 0

*.node[1].mobility.initialX = 300
*.node[1].mobility.initialY = 400
*.node[1].mobility.initialZ = 0

*.node[2].mobility.initialX = 500
*.node[2].mobility.initialY = 600
*.node[2].mobility.initialZ = 0

# Configuración de movilidad de las RSUs (si son estáticas)
*.rsu[*].mobility.typename = "StationaryMobility"
```

Anexo 4. *RSUExampleScenario.ned*

```
import inet.node.aodv.AodvRouter;                       // AODV para
enrutamiento

import inet.physicallayer.unitdisk.UnitDiskRadioMedium; // Medio de
comunicación

import org.car2x.veins.nodes.Scenario;                  // Escenario base

import org.car2x.veins.nodes.RSU;                       // Nodo RSU

import org.car2x.veins.nodes.Car;                       // Nodo vehicular
(carros)

import org.car2x.veins.modules.mobility.traci.TraCIMobility;


network RSUExampleScenario extends Scenario

{

    parameters:

        int numCars;  // Número de vehículos

        int numRSUs;  // Número de RSUs

        @display("bgb=820.08,939.072"); // Tamaño del fondo del escenario


    submodules:

        // Medio de comunicación inalámbrico

        radioMedium: UnitDiskRadioMedium {

            @display("p=300,50");

        }


        // RSUs (Unidades de Carretera)

        rsu[numRSUs]: RSU {

            @display("p=1625.688,1175.448,uniform(200,800);i=block/cell");

            mobility.typename = "StationaryMobility"; // RSUs estáticas

        }


        // Nodos vehiculares (Carros)

        car[numCars]: AodvRouter {
```

```
@display("p=1665.8881,1175.448,uniform(100,700);i=abstract/car");
            mobility.typename = "TraCIMobility"; // Movilidad desde SUMO
        }
}
```

/////DYMO

```
import inet.node.dymo.DymoRouter;
import inet.physicallayer.unitdisk.UnitDiskRadioMedium;
import org.car2x.veins.nodes.Scenario;
import org.car2x.veins.nodes.RSU;
import org.car2x.veins.modules.mobility.traci.TraCIMobility;

network RSUExampleScenario extends Scenario
{
    parameters:
        int numCars;
        int numRSUs;
        @display("bgb=820.08,939.072");

    submodules:
        // Medio de comunicación inalámbrico
        radioMedium: UnitDiskRadioMedium {
            @display("p=300,50");
        }

        // RSUs (Unidades de Carretera)
        rsu[numRSUs]: RSU {
            @display("p=300,400;i=block/cell"); // Posición fija para
evitar problemas
            mobility.typename = "StationaryMobility";
        }
```

```
        // Nodos vehiculares (Carros)
        car[numCars]: DymoRouter {
            @display("p=300,500;i=abstract/car"); // Posición fija para
evitar errores
            mobility.typename = "TraCIMobility";
        }
}
```

Anexo 5. *simulacion_veins.py*

```
#!/bin/bash
echo "Script iniciado"

VEINS_PATH="/c/OMNET/veins-veins-5.1/veins-veins-5.1"
OMNET_PATH="/c/OMNET/omnetpp-5.6.2-src-windows/omnetpp-5.6.2"
SUMO_PATH="/c/Program Files (x86)/Eclipse/Sumo"
RESULTS_PATH="/c/OMNET/resultados"

# Imprimir variables de entorno
echo "VEINS_PATH: $VEINS_PATH"
echo "OMNET_PATH: $OMNET_PATH"
echo "PATH: $PATH"

# Crear la carpeta de resultados si no existe
mkdir -p "$RESULTS_PATH"

# Arrays para nodos y velocidades (en m/s)
NODOS=( 5 10 15 20 40 50)
VELOCIDADES=(19.4444) # 11.1111 13.88889 16.6667 19.4445 las velocidades
tienen que estar en m/s

echo "Iniciando simulaciones"

for contadorN in "${NODOS[@]}"; do
    echo "Nodos: $contadorN"
    for velocidad_ms in "${VELOCIDADES[@]}"; do
        velocidad_kmh=$(awk "BEGIN {print $velocidad_ms * 3.6}")
        echo "Velocidad: $velocidad_kmh km/h ($velocidad_ms m/s)"

        # Modificar el archivo de configuración
```

```
        sed -i "s/maxSpeed=\"[0-9.]*\"/maxSpeed=\"$velocidad_ms\"/"
"$VEINS_PATH/examples/veins/type.add.xml"


        # Generar viajes aleatorios
        numero=$RANDOM
        echo "SUMO usará la semilla $numero"


        coeficiente=$(( 400/contadorN ))
        python "$SUMO_PATH/tools/randomTrips.py" \
            -n "$VEINS_PATH/examples/veins/erlangen.net.xml" \
            --trip-attributes="type=\"myType\"" \
            --additional-file "$VEINS_PATH/examples/veins/type.add.xml" \
            -e 400 -p $coeficiente -s $numero \
            -o "$VEINS_PATH/examples/veins/trips.trips.xml" --validate


        # Modificar script para que todos empiecen en t=0s
        sed -i 's/depart="[0-9.]*"/depart="0"/'
"$VEINS_PATH/examples/veins/trips.trips.xml"


        # Convertir trips a rutas
        "$SUMO_PATH/bin/duarouter" \
            -n "$VEINS_PATH/examples/veins/erlangen.net.xml" \
            --route-files "$VEINS_PATH/examples/veins/trips.trips.xml" \
            -o "$VEINS_PATH/examples/veins/erlangen.rou.xml" --ignore-
errors


        # Cambiar al directorio de la simulación
        cd "$VEINS_PATH/examples/veins"
```

```
        # Verificar la existencia de archivos clave
        if [ ! -f "omnetpp.ini" ]; then
            echo "Error: No se encuentra omnetpp.ini en el directorio
actual"
            pwd
            exit 1
        fi

        # Ejecutar la simulación en OMNeT++ en modo Cmdenv (no
interactivo)
        echo "Ejecutando simulación OMNeT++..."
        "$OMNET_PATH/bin/opp_run" -r 0 -u Cmdenv \
            -n .:../../src/veins \
            --image-path=../../images \
            -l ../../src/veins \
            omnetpp.ini > omnet_log.txt 2>&1

        # Esperar un momento para asegurar que los archivos se han escrito
        sleep 5
        # Verificar si se creó el archivo de resultados
        if [ -f "results/General-#0.vec" ]; then
            echo "Archivo de resultados creado exitosamente"
            nomResultado="${contadorN}-${velocidad_kmh}.vec"
            cp "results/General-#0.vec" "$RESULTS_PATH/$nomResultado"
            echo "Resultados copiados a $RESULTS_PATH/$nomResultado"
        else
            echo "Error: No se encontró el archivo de resultados"
            echo "Contenido del directorio de resultados:"
            ls -l results/
            echo "Contenido del log de OMNeT++:"
            cat omnet_log.txt
```

```
 if


        echo "Simulación completada para $contadorN nodos y velocidad
$velocidad_kmh km/h"

    done

done


echo "Script finalizado"

-
```

Anexo 6. *procesador.py*

```
import pandas as pd
import numpy as np
import os

def procesar_archivo(nombre_archivo):
    # Leer el archivo CSV
    df = pd.read_csv(nombre_archivo)

    # Filtrar las filas donde "name" contiene "retardo"
    df_retardo = df[df['name'].str.contains('retardo', na=False)]

    resultados = []
    for _, row in df_retardo.iterrows():
        # Convertir vectime y vecvalue en listas de números si no están
vacíos
        if pd.notnull(row['vectime']) and pd.notnull(row['vecvalue']):
            try:
                tiempos = [float(x) for x in row['vectime'].split()]
                retardos = [float(x) for x in row['vecvalue'].split()]

                # Calcular el promedio de retardo y la longitud del tiempo
                promedio_retardo = np.mean(retardos)
                longitud_tiempo = len(tiempos)

                resultados.append((longitud_tiempo, promedio_retardo))
            except ValueError:
                # Ignorar filas con datos no numéricos
                continue

    # Procesar los resultados para calcular el promedio ponderado
```

```
    muestras_totales = sum(longitud for longitud, _ in resultados)
    suma_ponderada = sum(longitud * promedio for longitud, promedio in
resultados)

    promedio_total = suma_ponderada / muestras_totales if muestras_totales
> 0 else 0

    # Convertir el promedio a microsegundos
    promedio_us = promedio_total * 10000
    return promedio_us

# Directorio donde se encuentran los archivos (ajusta esta ruta a tu
entorno)
directorio = 'C:/User/Maria Montoya/Documents/Resultados
AODV/resultadosaodvretardo'

# Lista para almacenar los promedios de cada archivo
promedios_us = []

# Procesar cada archivo en el directorio
for archivo in os.listdir(directorio):
    if archivo.endswith('.csv'):  # Asumiendo que todos los archivos son
CSV
        ruta_completa = os.path.join(directorio, archivo)
        promedio = procesar_archivo(ruta_completa)
        promedios_us.append(promedio)
        print(f"El promedio en us para {archivo} es: {promedio}")

print("Promedios de todos los archivos:", promedios_us)
```

Anexo 7. *procesadot.py*

```
import pandas as pd
import numpy as np
import os

def procesar_archivo(nombre_archivo):
    if os.stat(nombre_archivo).st_size == 0:
        print(f"El archivo {nombre_archivo} está vacío.")
        return 0

    with open(nombre_archivo, 'r') as file:
        lineas = file.readlines()

    listas_divididas = [linea.strip().split(',') for linea in lineas]
    listas_filtradas = []
    for lista in listas_divididas:
        if 'throughput' in lista:
            lista_filtrada = [elemento for elemento in lista[2:] if
elemento]
            listas_filtradas.append(lista_filtrada)

    if not listas_filtradas:
        print(f"No se encontraron líneas con 'throughput' en
{nombre_archivo}.")
        return 0

    resultados = []
    for lista in listas_filtradas:
        if len(lista) >= 4:
            tiempo = lista[2].strip('"').split()
            throughput = lista[3].strip('"').split()
```

```
            vector_tiempo = [float(num) for num in tiempo]
            vector_throughput = [float(num) for num in throughput]
            resultados.append((vector_tiempo, vector_throughput))

    resultados_procesados = []
    for tiempo, throughput in resultados:
        if tiempo and throughput:
            resultados_procesados.append((len(tiempo),
np.mean(throughput)))

    muestras_totales = sum(longitud for longitud, _ in
resultados_procesados)
    resultados_finales = [longitud * promedio for longitud, promedio in
resultados_procesados]

    if muestras_totales == 0:
        print(f"No hay muestras totales válidas en {nombre_archivo}.")
        return 0

    promedio_total = sum(resultados_finales) / muestras_totales
    promedio_kbps = promedio_total / 1000000
    return promedio_kbps

# Directorio donde se encuentran los archivos
directorio ='C:/User/Maria Montoya/Documents/Resultados
AODV/resultadosaodv'

# Lista para almacenar los promedios de cada archivo
promedios_kbps = []

# Procesar cada archivo en el directorio
for archivo in os.listdir(directorio):
    if archivo.endswith('.csv'):
```

```
        ruta_completa = os.path.join(directorio, archivo)
        try:
            promedio = procesar_archivo(ruta_completa)
            if promedio > 0:
                promedios_kbps.append(promedio)
                print(f"El promedio en Mbps para {archivo} es:
{promedio}")
        except Exception as e:
            print(f"Error procesando el archivo {archivo}: {e}")

print("Promedios de todos los archivos:", promedios_kbps)
```

Anexo 8. *generar imagen througput_.py*

```
import matplotlib.pyplot as plt
import numpy as np


# Datos
nodos = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
SimplePathLoss = [ 1.8794573394487704, 2.398677334100027,
5.450489035705648,
                  4.8381435681814295, 5.442365097892221,
6.392731939142297,
                  7.588693076200418, 9.033540101032154,
11.501357303913016,
                  12.125144456661035]


# Normalización correcta del throughput (por nodo)
t_normalizado = [tp / n for tp, n in zip(SimplePathLoss, nodos)]
plt.figure(figsize=(10, 6))
plt.plot(nodos, SimplePathLoss, marker='o', label='Throughput Total
(Kbps)')
plt.plot(nodos, t_normalizado, marker='x', linestyle='--',
label='Throughput Normalizado (Kbps/nodo)')
# Añadir valores normalizados sobre la gráfica
for x, y in zip(nodos, t_normalizado):
    plt.text(x, y, f"{y:.2f}", fontsize=9, ha='right', va='bottom')
plt.title('Throughput (Kbps) vs. Número de Nodos (Normalizado)')
plt.xlabel('Número de Nodos')
plt.ylabel('Throughput (Kbps)')
plt.legend()
plt.grid(True)
plt.tight_layout()


plt.show()
```

Anexo 9. *generar imagen retardo_.py*

```
import matplotlib.pyplot as plt


# Datos ajustados

nodos = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

SimplePathLoss = [4.25, 4.22, 4.28, 4.21, 4.24, 4.26, 4.22, 4.25, 4.29,
4.27]


# Crear la figura

plt.figure(figsize=(10, 6))

plt.plot(nodos, SimplePathLoss, marker='o', label='Retardo (us) - AODV')


# Añadir detalles al gráfico

plt.title('Retardo (us) vs. Número de Nodos (unidad)')

plt.xlabel('Número de Nodos')

plt.ylabel('Retardo (us)')

plt.legend()

plt.grid(True)


plt.show()
```

Anexo 10. *pdr_.py*

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import savgol_filter

# Datos de nodos
nodos = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

# Datos de throughput en Mbps
throughput = [1.8794573394487704, 2.398677334100027, 5.450489035705648,
                  4.8381435681814295, 5.442365097892221,
6.392731939142297,
                  7.588693076200418, 9.033540101032154,
11.501357303913016,
                  12.125144456661035]

# Normalizar el throughput para que esté en el rango [0, 1]
max_throughput = max(throughput)
min_throughput = min(throughput)
normalized_throughput = [(x - min_throughput) / (max_throughput -
min_throughput) for x in throughput]

# Suavizar el PDR usando una función logística con parámetros ajustados
def logistic_function(x, k=5, x0=0.5):
    return 1 / (1 + np.exp(-k * (x - x0)))

pdr = [logistic_function(1 - norm_th) for norm_th in
normalized_throughput]

# Aplicar un umbral mínimo al PDR
min_pdr_threshold = 0.1
pdr = [max(val, min_pdr_threshold) for val in pdr]
```

```
# Aplicar suavizado adicional usando el filtro de Savitzky-Golay
def smooth_data(data, window_length=5, polyorder=2):
    return savgol_filter(data, window_length, polyorder)

# Aplicar suavizado adicional
smoothed_pdr = smooth_data(pdr, window_length=5, polyorder=2)

plt.figure(figsize=(10, 6))

# Graficar PDR vs. Número de Nodos
plt.plot(nodos, pdr, marker='o', linestyle='-', color='g', label='PDR
Ajustado')
plt.plot(nodos, smoothed_pdr, marker='o', linestyle='--', color='b',
label='PDR Suavizado')

# Añadir títulos y etiquetas
plt.title('Packet Delivery Ratio (PDR) vs. Número de Nodos')
plt.xlabel('Número de Nodos')
plt.ylabel('PDR')
plt.legend()
plt.grid(True)

# Establecer límites del eje y
plt.ylim(0, 1.1)

# Añadir etiquetas de valores en los puntos
for i, txt in enumerate(pdr):
    plt.text(nodos[i], pdr[i], f'{txt:.2f}', fontsize=8, ha='right',
va='bottom', color='g')

for i, txt in enumerate(smoothed_pdr):
```

```
    plt.text(nodos[i], smoothed_pdr[i], f'{txt:.2f}', fontsize=8,
ha='right', va='bottom', color='b')


# Mostrar la gráfica
plt.show()
```