Departamento de informática de sistemas y computadores

Universitat Politècnica de València

Diseño, Implementación y evaluación mediante simulación de un protocolo de enrutamiento geográfico para redes VANET

Trabajo Fin de Máster

Master en Ingeniería de Computadores y Redes

Autor: Javier Moraga Matoque

Tutor: Juan Vicente Capella Hernández

Tutor Experimental: José Navarro Alabarta

2017/2018

Resumen

En un mundo cada vez más conectado, el concepto de red vehicular suma importancia y actualidad. Las VANET requieren tanto de infraestructuras físicas como mecanismos para asegurar que la información de las aplicaciones que se montan sobre este tipo de redes puedan entregarse bajo una tasa de entrega y retardo aceptables.

Este trabajo se centra en los mecanismos de enrutamiento utilizados en las redes tipo VANET. Se han estudiado exhaustivamente los protocolos más relevantes, implementándose un protocolo geográfico, que ha sido evaluado posteriormente mediante simulación utilizando el simulador ns3.

La peculiaridad del protocolo propuesto reside en su toma de decisiones mediante mecanismos tipo *machine learning*. Se ha utilizado por un lado un sistema experto basado en reglas, mientras que por el otro la toma de decisión vendrá dada por una red neuronal previamente entrenada. En cualquier caso, la información inicial que usarán ambos mecanismos estará basada en las coordenadas cartesianas de los nodos, así como parámetros relativos a la calidad de la señal.

Los resultados obtenidos muestran que el mecanismo de enrutamiento propuesto basado en redes neuronales presenta unas prestaciones similares a otras propuestas relevantes actuales pero ofreciendo más posibilidades, principalmente en cuanto a poder contemplar un gran número de parámetros en la toma de decisiones de enrutamiento, que permitirán mejorar las prestaciones y fiabilidad, y que además ampliarán el abanico de posibles aplicaciones.

Palabras clave: Redes VANET, *ns3*, enrutamiento geográfico, Sistema experto basado en reglas, redes neuronales.

Abstract

In a world more and more connected, vehicular network concept gains importance progressively. VANET networks need to use physical infrastructures as well as mechanisms to ensure that the information from the apps running on top of this networks can be delivered under a correct delivery ratio and delay.

This document focuses on the routing mechanism used in them. Distinct state of the art protocols will be presented, and another will be implemented for its evaluation inside ns3 simulator.

The peculiarity of the protocol under study resides in its different decision making process using machine learning mechanisms. On the one hand, an expert system based on rules is proposed. On the other hand a neural network, previously trained, will be deciding. In any case, initial information used by both of them will be based on the cartesian coordinates of the nodes and signal quality parameters.

The obtained results show that the routing mechanism proposed based on neural networks presents similar capabilities to other current relevant proposals but offering more possibilities, principally in respect of being able to take in account a bigger number of parameters in the

routing decision taking system, that will allow to improve the functionalities and reliability, and will enlarge the enlarge the range of possible applications.

Keywords: VANET, ns3, routing, geographical, coordinates, snr, expert systems, machine learning.

Tabla de contenidos

1. Introducción

1.1 Motivación de este Trabajo de fin de master

El campo de las redes vehiculares gana fuerza a medida que evolucionamos hacia una sociedad más conectada, donde el Internet de las Cosas (IoT) empieza a ser una realidad. La seguridad en carretera es un tema crítico siempre que hablamos de automoción, y gran parte de las mejoras en este campo vienen de parte del sector tecnológico, con intención de reducir el número de accidentes que se producen en carretera.

Este trabajo se enmarca en este contexto, estudiándose alternativas que mejoren las redes vehiculares mediante la implementación, evaluación y comparación de distintos protocolos de enrutamiento que se encargan de posibilitar la entrega de paquetes de manera eficiente y robusta. El modelado y simulación de las propuestas se realizará mediante el simulador de redes "Network Simulator 3", con él se llevará a cabo una amplia experimentación modelando escenarios lo más realistas posible, de forma que lo implementado en el simulador pueda llevarse a implementaciones en nodos físicos con mayores garantías, tras los ciclos de modelado, evaluación y mejora de las propuestas después análisis de resultados.

Con el protocolo a implementar se busca conseguir mayor robustez en las comunicaciones móviles, que se adapten mejor a los entornos dinámicos característicos de las redes vehiculares manteniendo una productividad, *overhead* y retardos aceptables.

Se ha optado por la implementación de un protocolo de enrutamiento geográfico, que utiliza en una de sus variantes un sistema experto de decisiones emulando en cierto modo el razonamiento humano. En su otra variante, se propone la utilización de una red neuronal previamente entrenada para la toma de decisiones de enrutamiento.

1.2 Objetivos

Los principales objetivos de este trabajo son:

- Analizar las técnicas y tendencias actuales a la hora de implementar redes vehiculares.
- Conocer el estado del arte en cuanto a protocolos de enrutamiento orientados a redes vehiculares.
- Estudiar las técnicas inteligentes más apropiadas para su integración en protocolos para VANET.
- Diseñar la implantación adecuada de los mecanismos de *machine learning* que serán el núcleo para la toma de decisiones del protocolo.
- Modelar correctamente el protocolo.
- Evaluar mediante simulación el comportamiento del protocolo a implementar.
- Llevar a cabo una completa experimentación que permita estudiar las prestaciones de las técnicas propuestas.
- Implementar correctamente el protocolo.
- Comparar los resultados con protocolos de referencia.

- Proponer futuras líneas de mejora en esta línea a partir de los resultados obtenidos.

1.3 Estructura memoria

El presente documento tiene como objetivo la explicación de forma clara y concisa de todos los aspectos de redes vehiculares que han sido clave a la hora de desarrollar este trabajo, así como acotar que se esperaba del mismo. En los siguientes párrafos se va a detallar la organización del documento.

En el primer apartado se hace la presentación y motivación para realizar este TFM, así como se nos explican los motivos que han propiciado su implementación, y los objetivos que se marcan de cara a la finalización del mismo.

El siguiente apartado expone las tecnologías relacionadas con el proyecto, ayudando así a su desarrollo correcto y permitiendo un funcionamiento dentro de nuestras expectativas. Estas tecnologías son también estudiadas, de forma que podamos entender que función realiza cada una en el ámbito global.

Comentadas las tecnologías usadas, se continuará describiendo los aspectos sobre la implementación y modelado llevados a cabo en el simulador. Se describirá el funcionamiento del protocolo a alto nivel, para poder posteriormente profundizar en cómo se ha desarrollado cada parte del mismo.

Se detallaran y comentaran los módulos implementados de forma abstracta, sin entrar en detalles del código. También se expondrán en este apartado los que han ido surgiendo y las soluciones aportadas.

Con la implementación ya completamente funcional, se procede a la realización de los experimentos, de esta forma se evalúa si el protocolo responde como se espera en distintos escenarios.

Finalmente se concluirá en base a los resultados obtenidos y al funcionamiento del protocolo. También se definirán posibles futuros trabajos en esta línea.

2. Tecnologías relacionadas

Este trabajo involucra un gran número de herramientas y tecnologías, que propiciarán una implementación conforme a los objetivos que se han establecido al inicio del mismo. Con este apartado se describen las más importantes, aquellas que constituyen la base, sin las cuales este proyecto no podría realizarse.

2.1 Protocolos VANET

Las redes de computadores han evolucionado a lo largo del tiempo, intentando satisfacer necesidades tecnológicas que han surgido con el paso de los años. Es por ello que dentro de este campo podremos encontrarnos con multitud de tipos de redes, desde cableadas hasta submarinas, pasando por las inalámbricas.

Dentro del marco de las redes inalámbricas se encuentran las VANET, motivo de estudio de este trabajo, para las que se presenta un protocolo de enrutamiento que aproveche sus propiedades más características con intención de ofrecer un buen servicio.

Las VANET, son un subtipo de las MANET (*Mobile Ad-hoc Network*) con ciertas características que las hacen peculiares. Ambas comparten el hecho de que los nodos se comunican de manera inalámbrica e intercambian datos actuando como servidores y clientes al mismo tiempo.

Ninguna utiliza regularmente infraestructuras de red fijas, por tanto precisan de mecanismos que las ayuden a auto regularse y organizarse; esta implementación aporta su grano de arena en este aspecto, al permitir crear una red mediante la acción combinada de cada nodo que ejecute el protocolo diseñado.

El intercambio de información entre vehículos es la característica más diferencial entre los dos tipos de redes. En las redes VANET, los nodos son muy dinámicos, provocando cambios en la topología, conexiones y desconexiones de manera frecuente, rápida y brusca. En las MANET los nodos son móviles, pero su rango de movimiento es más limitado y la topología no sufre tantas variaciones a lo largo de su vida útil.

En VANET tenemos dos esquemas de comunicación básicos, los cuales son V2V (vehículo con vehículo) y V2I (vehículo con infraestructura). Se emplean ambos para conseguir afrontar el reto de diseminar la información en una red que sufre los problemas derivados del dinamismo.

Como punto débil de estas redes se puede destacar el bajo ancho de banda que suelen alcanzar, además del limitado radio inalámbrico que alcanzan sus señales.

Uno de los aspectos más positivos que tienen es el hecho de que no se necesite un nodo centralizado que coordine el intercambio de datos en la red. Esto les otorga una cierta tolerancia a fallos, ya que no se dispone de un único punto de fallo que pudiese dejar la red inservible, y un fallo en un nodo puntual no supone un problema grave para la transmisión de información.

Por otro lado, al ser todos los nodos móviles y haber tantos cambios en la topología la transmisión de datos dista mucho de ser perfecta.

2.1.1 Arquitecturas de red en VANET

La información en VANET se puede diseminar de distintas formas, cada una corresponde a una topología de red posible distinta [2]:

- ***Pure cellular* o WLAN**: Basadas en *"cellular gateways"* fijas en combinación con puntos de acceso WLAN o *WiMax* repartidos en las intersecciones de tráfico, que suelen resultar más problemáticas a la hora de enviar señales. Estos puntos de acceso, además de actuar como *routers* entre los nodos, proveen a los vehículos con acceso a internet o dan información sobre el tráfico utilizando el *gateway*. Suelen estar más orientados a aplicaciones de *infotainment* [15] y se observan principalmente enlaces de tipo V2I.
- **Ad-hoc:** En esta arquitectura los propios nodos actúan como *routers*, de forma que pueden intercambiarse información utilizándose unos a otros. En este tipo de red también es común que aparte de nodos móviles hayan otros dispositivos fijos a los lados de la calzada conocidos como RSU (*Roadside units*) que pueden intervenir en el intercambio de información. Se observan principalmente conexiones de tipo V2V y son especialmente útiles para aplicaciones de emergencia, en lugares donde las redes tipo infraestructura son inexistentes u operan con dificultad [15]
- **Híbridas:** Más flexibles al contar con los beneficios de las dos anteriores. Sus nodos se comunican de forma ad-hoc, además de poder intercambiar información con puntos de acceso en modo infraestructura. El abanico de aplicaciones posible aumenta respecto a las anteriores.

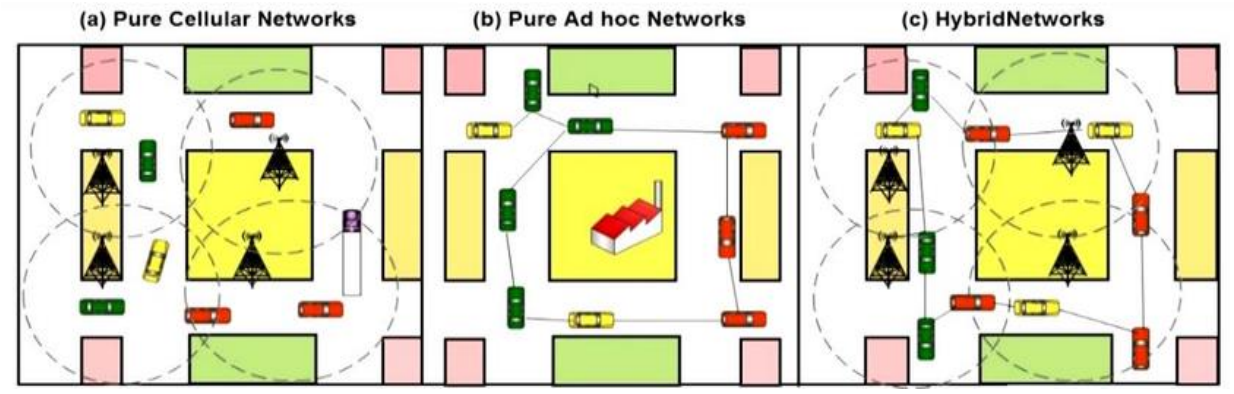


Ilustración 1. Tipos de infraestructura en una red VANET

2.1.2 Características de las VANET

A pesar de ser prácticamente subtipo de las MANET, las redes VANET poseen ciertas características que dificultan su creación de forma fiable y robusta. En este apartado presentamos las siguientes:

- Hay demasiado dinamismo, la topología de la red cambia frecuentemente creándose y destruyéndose enlaces continuamente.
- Un aspecto positivo es su naturaleza distribuida, todos los nodos que utilizan la red contribuyen a su creación, ya que cada uno constituye un *router* más.
- Las conexiones y desconexiones con nodos vecinos son muy frecuentes a causa del elevado dinamismo. Un nodo puede tener información de una cantidad de nodos, y en un instante cercano esa cantidad podrá ser mucho mayor o mucho menor, dependiendo del número de nodos cercanos de los que esté recibiendo señal.
- Al estar embebidos en vehículos, los nodos de estas redes se presuponen con energía "ilimitada", además de poseer grandes cantidades de almacenamiento.
- Podemos aprovechar la multitud de sensores que podemos adjuntar a los nodos, para obtener información que tratada correctamente pueda ayudar a mejorar la seguridad en carretera o permitir que los conductores puedan realizar su función de una forma más cómoda y sencilla.
- La calidad de la señal es muy variable, demasiado dependiente del entorno en el que se encuentra el nodo. Entornos densos, con muchos obstáculos para la señal, provocan que los enlaces sean mucho peores y de menor alcance. Es complicado modelar de forma real la variabilidad de la señal, por ello se han diseñado modelos de propagación que abstraigan este fenómeno según el entorno en el que se mueven los nodos.
- Algunas aplicaciones (se destacan algunas relativas a los *Intelligent Transport Systems*[24]) tienen requerimientos más acercados al tiempo real, precisando que la información se entregue con el mínimo retardo posible. Por ejemplo, cuando un coche realiza un cambio de carril o pulsa el freno, la información tendrá que llegar a sus vecinos de forma inmediata para evitar así colisiones entre vehículos. A parte de intercambiar esta información los vehículos están dotados de sensorización avanzada (sistemas de visión o laser 3D entre otros).
- Están llamadas a ser la base de la infraestructura de seguridad en carretera del futuro.

2.1.3 Distribución de la información

Para diseminar datos dentro de cualquier red, es muy importante que cada elemento que la constituye sepa cómo debe tratar la información. Por ello se diseñaron las pilas de protocolos, que regulan el comportamiento de cada nodo, para darles la capacidad de actuar de forma coordinada con el resto de elementos de la red.

Una parte importante de esta pila de protocolos reside en el protocolo de enrutamiento que se decida utilizar para repartir la información desde su origen a su destino. Este componente actuará en cada nodo enrutando los datos, acercándolos con cada salto más a su destino.

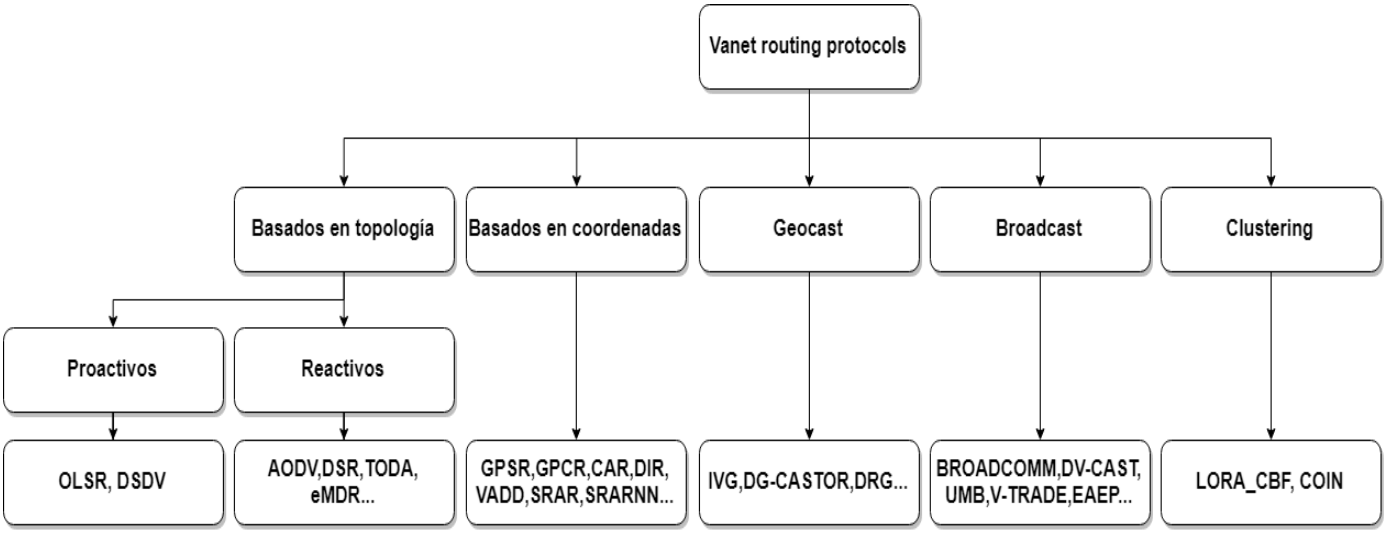


Ilustración 2. Taxonomía de los protocolos de enrutamiento orientados a VANET

Según el tipo de red al que nos enfrentemos y sus rasgos más característicos será más conveniente el uso de ciertos protocolos. En el caso de las redes VANET, estudiando la bibliografía, viendo diferentes experimentaciones realizadas y publicaciones se puede observar cómo, principalmente, nos encontramos ante los siguientes subgrupos[1][3][4][16]:

Protocolos basados en la topología de red

La topología se construye en base a la información de los enlaces que cada nodo tiene a su alrededor, que le servirá para tomar decisiones de enrutamiento adaptadas a las situaciones que puedan presentarse.

Se apoyan en estructuras en forma de tablas para almacenar la información de los distintos enlaces.

Dentro de los protocolos de enrutamiento basados en la topología podemos ver dos tipos:

- **Protocolos proactivos:** Guardan la información relativa al enrutamiento (como el *next hop* o los vecinos) en caché para ser utilizada por el protocolo. En este tipo de protocolos no es necesario realizar un descubrimiento de ruta de forma muy frecuente ya que una vez se descubre una buena ruta ésta es usada hasta que deja de ser válida.
  Sin embargo, esta clase de protocolos no dan resultados muy favorables en nuestro caso de estudio, ya que las VANET tienen una naturaleza dinámica que añadiría demasiado *overhead* en la red al tener que descubrir rutas nuevas con frecuencia.

  Ejemplos: OLSR, DSDV…

- **Protocolos reactivos o ad-hoc:** Calculan una ruta nueva para cada paquete que se quiera enviar desde un origen hasta un destino. Por tanto se comportan mejor en entornos dinámicos como los de las VANET.

Debido a la similitud de las VANET con las MANET se ve muy frecuentemente la prueba de protocolos típicos como AODV o DSR, sin éxito, al ser las primeras mucho más cambiantes y dinámicas.
Por ello se pensó que tal vez estos protocolos podrían adaptarse al nuevo entorno, teniendo en cuenta sus características, para así resolver los problemas que nos aparecen.
Un ejemplo de estas adaptaciones destinadas a combatir la pérdida de enlaces provocada por la movilidad es la creación de protocolos de predicción como PRAODV y PRAODVM. Estos predicen cuanto va a durar un enlace en base a la velocidad relativa entre los nodos y la distancia de los mismos. De esta forma nos anticipamos a la pérdida del enlace calculando una nueva ruta hacia el destino. Así el fallo que se daría en el protocolo AODV normal se vería cubierto con esta ruta más fresca.

Ejemplos: PRAODV, PRAODVM, AODV, TODA, DSR, eMDR…

Protocolos basados en coordenadas geográficas

Las decisiones de enrutamiento basadas en el conocimiento de la localización geográfica de los nodos, con el apoyo de mapas, modelos de tráfico y sistemas de navegación suelen dar mejores resultados que los protocolos basados en topología.

Esto se debe al hecho de que el movimiento de los nodos en una VANET suele venir definido por las calles en las que se mueven, y normalmente puede darse en dos direcciones. Además no se precisa de un conocimiento o mantenimiento de una ruta global de inicio a destino[17].

A pesar de este hecho, este tipo de protocolos no están exentos de problemas. La información de la localización no es suficiente a la hora de tomar decisiones que permitan modelar un protocolo robusto ante entornos tan cambiantes.

Con el paso del tiempo, los protocolos de este tipo han ido mejorando, obteniendo cada vez resultados más positivos. Un ejemplo de esto es la evolución de los protocolos *greedy*, consistente en añadir progresivamente nuevas técnicas o datos de entrada con intención de mejorar el rendimiento:

- ***Greedy routing*:** Protocolo consistente en enviar los paquetes al *next-hop* más cercano al destino.
- ***Greedy perimeter stateless routing (GPSR)[18]:***
  Sufre de problemas al usarse en escenarios urbanos, debido a los obstáculos que pueden haber en las calles que impiden que la señal llegue al destino correctamente. Aquí tendremos dos modos, el modo *greedy* ya comentado, además del modo *perimeter* que se encarga usar la *right hand rule* en los casos que ningún *nexthop* consiga acercar más el paquete hacia el destino.
- ***Greedy perimeter coordinator routing (GPCR)*:** Aquí se combate el problema de los obstáculos añadiendo nodos coordinadores en las zonas susceptibles a problemas como pueden ser las intersecciones. Los coordinadores toman decisiones de enrutamiento y se eligen mediante heurísticas.

Ejemplos: GPSR,GPCR ,*Connective aware routing* (CAR), *Diagonal-Intersection-Based Routing Protocol* (DIR) y protocolos tolerantes a retrasos (MOVE, VADD, SADV).

Enrutamiento usando clusters

En esquemas como este agruparemos los nodos móviles en infraestructuras de red virtuales, que conocemos como *clusters*.

Dentro tendremos dos roles:

- **Coordinador o *cluster head***, encargado de regular las comunicaciones dentro de la estructura y de comunicarse con el resto de *clusters*.
- **Nodo genérico**, que trata de comunicarse con el resto de nodos. Lo hace de forma ad-hoc con los de su mismo *cluster* y usará el coordinador para comunicar con otros *clusters*.

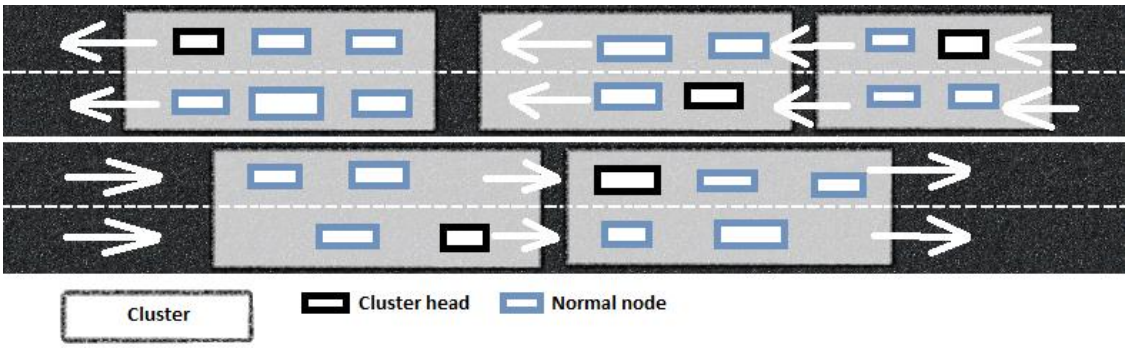


Ilustración 3. Descripción de estructura de red en *clusters*

Se ha demostrado empíricamente que los protocolos que parten de esta idea obtienen buenos resultados en redes MANET. No obstante, las características específicas de las redes VANET hacen que no se obtengan resultados tan positivos en estas. La creación de *clusters* en entornos tan variables no consigue los objetivos buscados y no se consigue una buena escalabilidad al disolverse estos muy rápidamente.

Protocolos basados en *clustering* para VANET:

- ***Clustering for open IVC Networks* (COIN):**
  Head escogido por el protocolo, que tomará su decisión entre todos los nodos de cierto *cluster* basándose en entradas como el movimiento de cada vehículo o las intenciones que tiene su conductor.
  Se adapta bien a los cambios en la distancia entre los vehículos de la red y obtiene una red con *clusters* más estables, con la penalización de añadir un mayor *overhead* en la red que otros protocolos.
- **LORA_CBF:**
  Protocolo de localización reactivo, caracterizado por usar *flooding* dentro de los *clusters*. Para este protocolo existen tres tipos de nodos:
  - Head:
    Encargados de portar la información sobre que miembros son los encargados de comunicar con otros *clusters* u otros nodos *head*.

- Gateway:
  Se encarga de la comunicación de los nodos de su *cluster* con otros *clusters* de la red.
- Miembro normal:
  Nodos que se encargarán de simplemente enviar mensajes al resto de nodos, dejando el *forwarding* en manos del *Gateway* del *cluster* del que forman parte.

Modo de funcionamiento:
Si queremos enviar a un destino del que no se tiene información sobre su localización aparece en escena el mensaje *Location Request* (LREQ).
Este mensaje será respondido mediante un *Location Replay* (LREP) dado por algún *cluster head* vecino que tenga la información solicitada.
Los únicos nodos capaces de hacer *forwarding* hasta el destino de los dos mensajes comentados son los nodos *head* o *Gateway*, consiguiendo así que no todos los nodos intervengan en esta diseminación y disminuyendo el *overhead* de manera notable respecto a los protocolos *broadcast*.

Se ha demostrado que este protocolo funciona bastante mejor que AODV o DSR en escenarios típicos (urbano, carretera…). Además, es mucho más escalable que otros protocolos y soporta mejor los escenarios donde nos encontremos ante una mayor movilidad de los nodos.

No obstante, el proceso de creación y modificación de un *cluster* penaliza mucho en redes VANET. Al haber tantos cambios, el mantenimiento de la estructura de *clusters* supone un aumento en el *delay* y en el *overhead* debido a los paquetes de control.

Ejemplos: COIN, LORA-CBF

Protocolos basados en Broadcast

Hay información (tráfico, meteorológica, emergencias…) que resulta de utilidad para todos los nodos que forman la red VANET. Por ello es frecuente el uso del *broadcast* cuando se necesita diseminarla.

Hay protocolos de *routing unicast* donde se usa *broadcasting* para diseminar información que ayude a encontrar una ruta eficiente hacia cada destino. Cuando el nodo destino no se encuentre dentro del rango inalámbrico se usarán estas rutas mediante *multi-hopping*.

El *flooding* es la técnica más simple y usada para realizar *broadcasting*. El paquete enviado es recibido por los nodos vecinos, que a su vez lo reenvían a sus nodos adyacentes. De esta forma, todos los nodos de la red deberían recibir el paquete siempre y cuando la red no se saturase.

El re-emitir el mensaje puede ocasionar bucles que provoquen que un paquete pueda retransmitirse eternamente, por ello los números de secuencia se introducen en los paquetes. Así un paquete queda identificado por la dupla <nodoEmisor,numeroSecuencia> y los nodos que lo reciben sabrán si es un duplicado y por tanto deben descartarlo. También mide la "frescura" del mismo, descartando así paquetes con un número de secuencia menor al último recibido.

Pero usar *flooding* no es recomendable en ciertas situaciones. Si escalamos la red, el ancho de banda necesario aumenta de manera exponencial y aparecen colisiones y contenciones a causa de los intentos de emisión simultáneos que se dan por toda la red.

Se han ideado otras técnicas como *broadcasting* acotado (*Geocast,* a analizar en el siguiente punto) por zonas o por saltos, que aligerarán esta sobrecarga en toda la red.

A continuación se van a comentar algunos protocolos basados en *broadcast*:

- **BROADCOMM [6]**
  Este protocolo sigue una estructura jerárquica a usar en autopistas.
  Los nodos se organizan en células virtuales y en dos posibles niveles de jerarquía. El primer nivel incluye todos los nodos en una célula, el segundo nivel está compuesto por los reflectores de célula (*cell reflectors*) que son unos pocos nodos situados cerca del centro de la célula.

  Los reflectores de célula actúan durante cierto tiempo como lo haría un *head cluster*, tratando los mensajes de emergencia de los miembros de la célula o miembros cercanos de otra.
  Además sirve para enrutar dentro de la célula mensajes de emergencia que vienen de otras células.

  Este protocolo deja en evidencia a otros protocolos de *flooding* similares, pero al ser tan simple sólo resulta adecuado para un escenario de autopista.

- ***DV-CAST: Distributed vehicular broadcast protocol[17]:***
  La información sobre la topología local es obtenida mediante el uso de los mensajes HELLO que se envían por *broadcast*.

  Cada nodo tiene un *flag* para sus paquetes que le permite conocer si cierto paquete entrante es redundante o no.

  Distingue entre tres tipos de vehículos dependiendo de su conectividad local, por tanto tendremos nodos fuertemente conectados, otros dispersamente conectados y otros totalmente desconectados de sus vecinos.

  Para los fuertemente conectados se usa un esquema de persistencia.

  Para los dispersamente conectados después de recibir el *broadcast*, los nodos pueden reenviar este a los nodos que se mueven en su misma dirección.

  Para los totalmente desconectados, almacenarán el mensaje hasta que un nuevo vehículo entre en su rango o se acabe su periodo de validez.

  Es bastante deficiente en redes altamente escaladas por su alto *overhead* de control y un gran *delay* en la ruta desde inicio hasta destino.

- ***Urban Multi-Hop Broadcast Protocol* (UMB)**
  Destinado a combatir interferencias, colisiones de paquetes y el problema del nodo oculto cuando hacemos *broadcasts* de más de un hop.

Los nodos emisores intentan seleccionar el nodo más lejano en la dirección del *broadcast* para asignarle la tarea de enviar y comprobar el envío del paquete sin tener ninguna información sobre la topología de red.

En las intersecciones disponemos de repetidores para poder enviar paquetes a todas las direcciones en la intersección. Este protocolo tiene mayor éxito cuando hay mucha carga de paquetes y la densidad del tráfico que otros protocolos 802.11 basados en aleatoriedad y distancias.

- ***Vector-based Tracking detection* (V-TRADE) y *History-enhanced V-Trade* (HV-TRADE)** son protocolos de *broadcast* basados en GPS.[7]

  Un nodo que implemente este protocolo de enrutamiento se basará en la información sobre la posición y movimiento que tiene de sus vecinos para clasificarlos en distintos grupos.
  Dentro de cada grupo sólo los que se encuentran más en la periferia del mismo (*border vehicles*) serán los encargados de reenviar el *broadcast*.

  Así conseguimos una utilización más eficiente del ancho de banda a costa de perder un poco de alcance de señal, consecuencia de no usar todos los vehículos como repetidores de la señal. No obstante, el proceso realizado para seleccionar el nodo emisor en cada salto añade bastante *overhead* en la red.

- ***EAEP: Edge-aware epidemic protocol[17][19]:***

  Reduce el *overhead* de paquetes de control eliminando el exceso de mensajes HELLO entre los nodos de diferentes *clusters*. Es un protocolo de diseminación probabilística de información que no precisa de conocimiento de la topología.

  Los *epidemic protocols* convencionales asumen conectividad continua de punto a punto, por tanto este tipo de protocolo orientado a VANET añade características que hagan frente a las desconexiones frecuentes.

  Cada nodo conoce su posición, añadiendo la misma al mensaje HELLO, que podrá a su vez contener un parámetro indicando la dirección de propagación del mismo.

  Tendrá dos modos de funcionamiento según la dirección que tomará el HELLO en cada envío, distinguiendo entre envíos unidireccionales y envíos hacia una dirección específica.

  El protocolo realizará cálculos estadísticos en cada caso basados en parámetros configurables, y en el caso de los envíos omnidireccionales se tendrá en cuenta la urgencia de los mensajes en el cálculo de la probabilidad de envío. Se usará esta probabilidad en cada nodo para saber si retransmitir o no el *broadcast*.

Ejemplos: BROADCOMM, UMB, V-TRADE, V-CAST

Routing basado en geocast

Estos protocolos nacen como respuesta a los problemas que acarrea el uso de *flooding*.

Se idea un tipo de grupo *multicast* basándose en la localización de los nodos. Cada nodo tendrá asociada una *Zone Of Relevance (ZOR)* donde enviar sus paquetes.

Especialmente indicado para casos en los que la información únicamente es útil en ciertas zonas de la carretera.

Por ejemplo, en un accidente, el vehículo puede aprovechar sus capacidades de cómputo para detectar situaciones de riesgo o accidentes mediante sus sensores. La información interpretada por este puede resultar de utilidad para vehículos que todavía no han llegado al epicentro del peligro o el problema. La información será enviada mediante *flooding* por el protocolo basado en *geocast*, pero sólo a la zona de interés. Así ahorramos en la sobrecarga que provocaría un *broadcast* general.

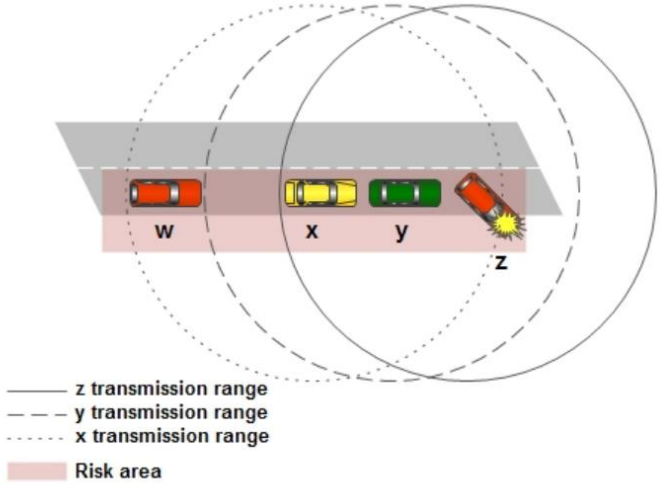


Ilustración 4. Zona de riesgo donde se realizan envíos GEOCAST marcada en rojo

De esta manera se evitan muchas colisiones en el acceso al medio inalámbrico y el número de *rebroadcasts* se reduce drásticamente.

Se presenta con esto solución al problema de las *broadcast storms[20]*, ya que además de reenviar en zonas acotadas, cada nodo se espera un tiempo para tomar la decisión de enviar el *rebroadcast*. La velocidad a la hora de tomar esta decisión es directamente proporcional a la distancia entre el nodo que ha recibido el paquete y el emisor.

Cuando este tiempo de espera expira se comprueba que no se ha recibido el mismo mensaje por parte de otro nodo, y si se afirma se procede al *rebroadcast*.

***Intervehicles geocast protocol* (IVG):**
Pensado principalmente para utilizar el *geocasting* de forma que se puedan enviar alertas dentro de un área de riesgo.

Si tenemos en cuenta la naturaleza cambiante de las VANET, se llega a la conclusión de que el hecho de mantener paquetes que no hayan podido ser enviados dentro de cierta ZOR no es una mala idea. Es por ello que en algunas implementaciones se añaden cachés a la capa de *routing*. Se han hecho estudios que refuerzan el uso de esta idea, y dejan ver que son útiles para el envío de paquetes que pudiesen perderse por el particionamiento de la red o situaciones puntuales de conexión deficiente.

Ejemplos: IVG, DG-CASTOR, DRG

Clasificación del protocolo	Puntos fuertes	Puntos débiles
A)Protocolos basados en topología		
A1)Proactivos	- Rutas predefinidas. - No hay descubrimiento de rutas. Se ahorra el *delay* asociado.	- El dinamismo provoca una excesiva actualización de la tabla de enrutamiento. - Mantener los nodos no usados provoca que se acabe sobrecargando la capacidad de la red. - Alto uso del ancho de banda. - Degradación del rendimiento de la red por los tres factores anteriores.
A2)Reactivos	- Reducción del tráfico de la red, ahorrando ancho de banda. - Con ayuda de tabla de enrutamiento para mantener las rutas desde origen hasta receptor.	- *Delay* de descubrimiento de ruta es alto. - Si la comunicación sobrepasa las capacidades de la red, el *flooding* de paquetes puede causar la suspensión de sus nodos.
Geográficos	- No se mantienen rutas ni tablas de enrutamiento. - Se evita el *delay* de descubrimiento de rutas.	- Necesidad de GPS y sistema de localización de nodos. - La señal GPS no alcanza dentro de los túneles. - Se puede provocar *deadlock* en el sistema de localización de nodos.
Cluster	- Alta escalabilidad. - Apto para redes grandes.	- *Delay* de formación de clústers.
Broadcast	- Fácil de implementar. - Apto para una pequeña	- Redes muy grandes implican alto uso de ancho de banda.

	porción de red.	- Todos los nodos reciben el broadcast en instantes similares, aumentando las posibilidades de colisión y congestión.
Geocast	- Se reducen las colisiones respecto a los de *broadcast*	- Necesidad de tecnologías externas, como GPS y un sistema de localización para el resto de nodos.

Tabla 1. Comparativa de posibles protocolos de enrutamiento para VANET[16]

2.1.4 Aplicaciones en VANET

La inclusión de tecnología en cualquier ámbito aporta por norma general muchos beneficios. En este caso se tiene la motivación de querer comunicar vehículos de manera eficiente y cada vez más fiable, de forma que tenga un impacto positivo en muchas áreas, sobre todo en la seguridad en carretera.

Las características de los servicios que implementemos son tenidas en cuenta para mejorar aspectos dónde la inclusión de aplicaciones VANET resulta beneficiosa.

En [21] se resalta la importancia de muchas aplicaciones VANET en materias como eficiencia y seguridad. Se nos pone un ejemplo práctico con el que queda demostrado que en este tipo de redes estas dos características están realmente ligadas, y lo que se acaba buscando es un balance entre ambas.

Las aplicaciones VANET permitirán a los nodos monitorizar su entorno para obtener distintos tipos de datos que ayuden a montar una infraestructura global más segura y eficiente.

Vehicular Network	**Tipo de aplicación**	• Aplicación de seguridad • Transporte inteligente • Destinada al confort • Sensorización urbana
	Calidad de servicio	• Tiempo real relajado • Tiempo real estricto • Tolerante a retardos
	Alcance	• Local • *Wide area*
	Arquitectura de red	• Ad hoc • Infraestructura • Híbrida
	Tipo de comunicación	• V2I (Vehículo-Infraestructura) • V2V(Vehículo-Vehículo)

Tabla 2. Características de red vehicular[9]

Las aplicaciones se pueden clasificar atendiendo a diferentes criterios.

Si incidimos en el tipo de aplicación, podremos observar que según el servicio al que estén orientadas o el fin que tengan se clasificarán en:

- **Aplicaciones de seguridad [8]:**
  Constituyen el campo más importante, sobre el que más investigación se ha realizado con intención de obtener un mejor servicio. Facilitan las condiciones para la conducción tratando de reducir la probabilidad de accidente. Sus medidas de seguridad, entre otras, podrían ser el envío de alertas al conductor sobre situaciones de riesgo cercanas. Además los vehículos ya vienen dotados con cierta lógica que le permita anticiparse a situaciones de riesgo y aplicar los actuadores necesarios para evitar que el riesgo se materialice, gracias a asistentes de conducción.

  Dentro de las aplicaciones de seguridad podemos encontrar distintas funciones, se enumeran las de mayor relevancia:

  - Alerta de colisión cooperativa
  - Manejo de incidentes
  - *Streaming* de video de emergencia
  - Asistente de giros[23].
  - Alertas por cambio de carril potencialmente peligroso.[23]
  - Alertas por condiciones peligrosas de la calzada.
  - Asistente para detención ante señales de STOP.

- **Aplicaciones orientadas a crear *Intelligent Transport Systems (ITS)*[24]**:
  El campo de los *Intelligent Transport Systems (ITS)*, está basado en un estándar de comunicación especialmente pensado para redes del estilo de las VANET, conocido como *DSRC (Dedicated Short Range Communications)*.

  DSRC permite comunicaciones inalámbricas a corto y medio alcance que proporcionen altas tasas de transferencia para información crítica y con un retardo reducido [25].

  Este campo de las VANET está destinado al envío de información crítica relacionada con el tráfico y la conducción, que hagan de esta una actividad más segura y sostenible medioambientalmente.

  La información permitirá, en combinación con la capacidad de cómputo de los nodos VANET, conseguir ver aspectos del tráfico que no se aprecian a simple vista, evitando así muchas situaciones de riesgo por anticipación.

  Ejemplos de características que pueden formar parte de un ITS:
  - Tener una idea global del tráfico que se tiene alrededor de una zona, lo que conocemos como **monitorización del tráfico**.
  - Aprovechando esta monitorización, se pueden diseñar sistemas que regulen el tráfico en zonas conflictivas, centrándose en la **gestión del tráfico**.
  - Aprovechando las capacidades anteriores, en ciertas situaciones, como por ejemplo el transporte de mercancías, puede ser interesante realizar el transporte de forma única, es decir, formar caravanas con los

vehículos que llevan la mercancía o crear autovías automatizadas[26]. Esto se conoce como ***platooning***.

- ***Tracking* de vehículos.**
- Alertar a los conductores o al resto de nodos de ciertas circunstancias mediante el uso de **servicios de notificación**.
- Automatización del Pago de peajes.

La creación de este tipo de sistemas, que integra distintos tipos de aplicaciones, ayuda a mejorar la seguridad en carretera, a conseguir una circulación más eficiente y a disminuir las emisiones nocivas para el medioambiente.

- **Aplicaciones de *infotainment*[27]:**
  La tecnología suele estar asociada con una mejora en la facilidad de uso de ciertos sistemas o la mejora en comodidad a la hora de realizar algunas actividades. Las aplicaciones de *infotainment* están enfocadas en conseguir precisamente esto, haciendo el viaje de los pasajeros más entretenido y llevadero.
  - Gestión de plazas de aparcamiento.
  - Juegos distribuidos o comunicación por voz.
  - Aplicaciones *Peer-to-Peer*.
  - Conexión a perfiles sociales de los pasajeros.
  - Información relativa al ocio según la zona que atraviesa el vehículo.

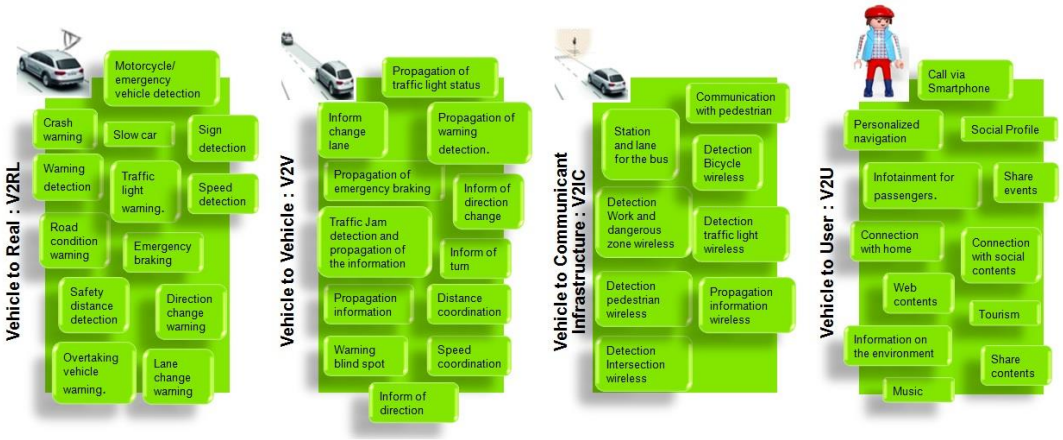


Ilustración 5. Posibles utilidades de aplicaciones VANET

- **Aplicaciones de sensorización urbana [21]:**
  Una red vehicular a gran escala es vista como una oportunidad para monitorizar datos urbanos y compartir datos de común interés. Con estas aplicaciones se podrá monitorizar condiciones del entorno y las actividades sociales de las áreas urbanas. Estas habilitarán la creación de redes VSN (*Vehicular Sensor Networks*).

Aunque se enmarquen en cuatro grupos distintos, todas estas aplicaciones definirán claramente a la hora de su desarrollo cuáles son sus requisitos para un funcionamiento correcto. Estos requisitos clasificarán a las mismas en base al resto de criterios mostrados en la tabla 2.

Como criterios podemos observar:

- Calidad de servicio, marca cómo ha de llegar la señal al destino, cuanto porcentaje de pérdida de información y cuanto retardo puede tolerar la aplicación.
- Alcance de la señal, dependiendo del ámbito al que van dirigidas y al tamaño de la porción de red sobre la que operarán.
- Tipo de arquitectura sobre el que funcionan dependiendo del tipo de enlaces que se forman entre los nodos que las utilizan.

2.1.5 Entornos de simulación en VANET

Nos encontramos ante un campo bastante crítico, las aplicaciones que queramos aportar al mismo no pueden ser probadas directamente en los escenarios reales a los que van a enfrentarse, ya que un fallo podría acarrear consecuencias bastante graves.

Por ello, es frecuente el uso de modelos y simulaciones que nos permitan hacer el *testing* de las mismas en situaciones "virtuales" lo más realistas posibles sin necesidad de invertir en infraestructuras físicas reales, que suponen un gasto inasumible en la mayoría de situaciones. Además, se consigue una mayor flexibilidad y rapidez a la hora de hacer las pruebas[28].

La base principal para el desarrollo en este campo hace uso de simuladores que tienen implementados múltiples modelos (en representación de características de los entornos reales, dispositivos y/o protocolos) que los dotan con capacidades para emular con la mayor precisión posible la inserción de las aplicaciones VANET en distintas situaciones.

2.1.5.1 Modelos de movilidad

Sirven para marcar en nuestras simulaciones cómo es el movimiento de todos los nodos que forman la red. Podemos hacer que las implementaciones puedan probarse en situaciones muy distintas, controlando el dinamismo de los nodos del escenario, la densidad de los mismos por área, las zonas u obstáculos para la correcta transmisión de la señal u otros aspectos que puedan verse en el mundo real.

Tenemos distintos modelos, por ejemplo:

- ***Random WayPoint mobility model [10]:*** Su funcionamiento es tan simple como ir diciéndole al nodo destinos aleatorios a los que acudir. Se menciona por ser ampliamente utilizado a pesar de no representar ninguna situación real.
  Pasado un tiempo este modelo fue modificado para añadirle parámetros como la longitud de la carretera, sus carriles o la distancia inter-vehicular.
- **Saha&Johnson[28] *o Street random waypoint* (STRAW)** son ejemplos de modelos que se apoyan en mapas virtuales convertidos en grafos, sobre los que operar para calcular las mejores rutas en base a diferentes criterios, siendo el más común el de distancia más corta. Se implementan de forma que puedan tener en cuenta factores realistas, como pueden ser la velocidad de las calles en el caso del primero y el tráfico urbano o la interacción del vehículo con los controles de tráfico.

- **Usar de trazas reales de vehículos**, permite escoger escenarios basados en situaciones reales que se ajusten más al escenario de simulación en el que queremos evaluar la aplicación.

2.1.5.2 Simuladores de redes[29]

La simulación de redes es la metodología más común y eficiente en términos de coste a la hora de evaluar distintas topologías de red. No precisar de implementaciones físicas supone un ahorro económico y de tiempo importantes.

La necesidad de lanzar al mercado nuevos protocolos que mejoren a los actuales es un hecho, pero estos han de ser probados previamente para asegurar su buen comportamiento. Es por esto que su uso es común entre la comunidad investigadora para evaluar sus nuevas teorías e hipótesis.

En el terreno de la evaluación de topologías VANET, las trazas de movilidad obtenidas sirven como *input* para los simuladores de redes. Estos cuentan con representaciones virtuales de los componentes de las capas ISO/OSI necesarios. También implementan otro tipo de modelos, como el de propagación de la señal o herramientas que nos permitan sacar resultados y estadísticas de manera más sencilla.

El modelado analítico que se obtiene con estas herramientas puede dar resultados que no tengan una precisión al detalle, pero sirvan para dar una idea del rendimiento de los protocolos a evaluar de una forma rápida y más *cost effective*.

Existen diversas alternativas a la hora de elegir un simulador de redes, cada uno con sus ventajas y sus puntos flojos, a continuación se describen los más comunes en este campo:

- ***Network simulator 2:***
  Simulador de eventos discretos de red, de tipo *open source*.
  Usado principalmente para la evaluación de protocolos de red utilizando distintas topologías. Capaz de simular redes de tipo cableado y también inalámbricas. Construido sobre C++ y capaz de proveer la interfaz de simulación mediante *OTcl*. Los escenarios se crean con scripts de este último lenguaje.

- ***Network simulator 3:***
  Proyecto iniciado a mitades de 2006 que ha ido desarrollándose a grandes pasos desde entonces. Igual que ns-2, ns-3 es un simulador de eventos discretos de red con carácter *open source*. Llamado a ser el reemplazo de ns-2.
  Está escrito en C++ y *Python*, esto nos habilita para ser capaces de diseñar nuestras simulaciones íntegramente en C++, y a su vez poder usar algunas partes de *Python*.

- **OMNET++:**
  Presentado al público en Septiembre del 1997, llevar tanto tiempo en el mercado le ha permitido obtener una gran cuota de usuarios.
  Una de sus ventajas es que no se limita únicamente a las simulaciones de red, ya que puede usarse para modelar multiprocesadores y hardware distribuido o

evaluar el rendimiento de sistemas software complejos.
A pesar de esto su uso más común es la simulación de redes. Se encaja también en el marco de los simulaciones de eventos discretos.

2.2 Modelos y módulos del simulador ns3 empleados

2.2.1 *Network simulator 3* - NS3[31]

El objetivo que se tenía al desarrollar este simulador de eventos discretos de red era el abarcar los campos de la educación y la investigación. El proyecto parte en septiembre del 2006, con carácter *open source* y con intenciones de sustituir a su predecesor, el simulador ns2. Se deja bien claro que no es retrocompatible con este último, que se mantendrá hasta que las funcionalidades de ns3 estén completas y se haya realizado una transición e integración estudiada al detalle.

Su carácter *open source* facilita el mantenimiento de un entorno abierto a nuevos investigadores, que pueden contribuir en el desarrollo del simulador, y también pueden utilizarlo para propósitos personales o profesionales donde resulte útil.

Su uso principal se da en sistemas Linux, aunque se puede utilizar en Windows mediante *Cygwin*.

El soporte viene dado por sus propios usuarios, a través de su *mailing-list*, al no ser un producto oficialmente apoyado por ninguna compañía. Se recomienda a los usuarios de ns2 la transición a este nuevo simulador, que tiene un mayor soporte por parte de los usuarios de la comunidad.

Sus modelos están escritos en C++ y la mayor parte de su API está disponible también en Python. Para utilizar un simulador de este tipo se necesitan al menos conocimientos básicos sobre *socket programming*.

En [30] se nos da una idea sobre qué aspectos se han mejorado respecto a ns-2. Entre sus mejoras destacamos que tiene unos modelos rediseñados para acercarlos más a la realidad. El haber nacido cómo una síntesis de otros simuladores le permite aprovechar los puntos fuertes de cada uno. Por ejemplo, el *debugging* es mucho más eficaz en esta versión.

El proceso de instalación se encuentra detallado en su web oficial, teniendo como prerrequisitos la instalación de ciertas librerías, así como algunas herramientas necesarias para su despliegue.

Una vez instalado, se nos facilita la herramienta *waf*, que nos permite hacer un *build* correcto de los módulos y scripts que deseemos del simulador de manera sencilla e intuitiva.

Con *waf* también lanzaremos nuestros scripts de simulación.

2.2.1.1 Abstracciones clave de ns3 a tener en cuenta en la experimentación

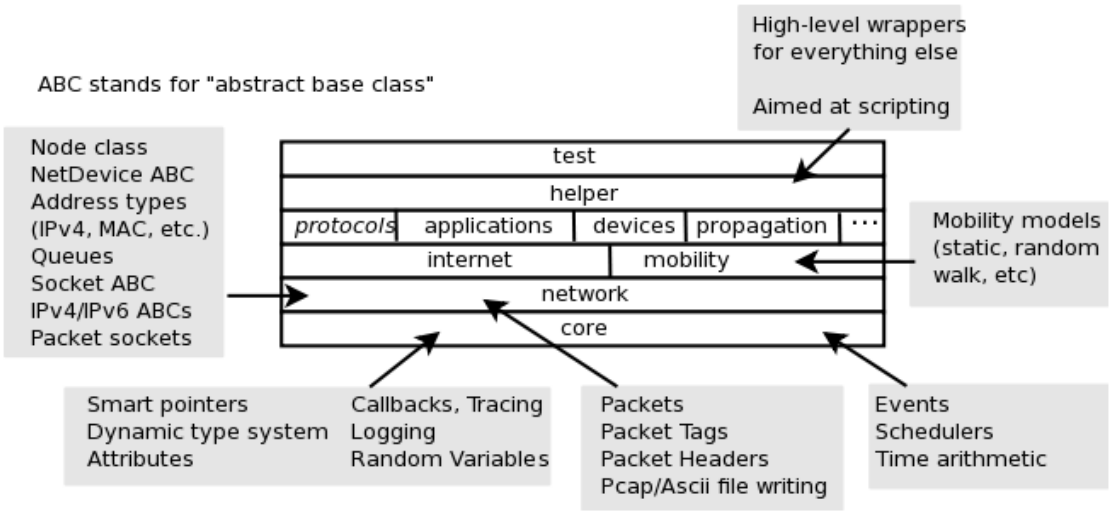


Ilustración 6. Arquitectura del simulador ns3

Dentro del simulador, tenemos varios modelos y abstracciones básicas, cuyo conocimiento es necesario a la hora de desarrollar scripts correctamente:

- **Nodo:** Abstracción más básica que se le da a un dispositivo de cómputo dentro del simulador. Representado por una clase *C++* que le dota de métodos para gestionar los elementos que lo componen.

- **Aplicación:** Representa la entidad que genera alguna actividad a ser simulada y evaluada. Su abstracción se representa con la clase *Application*, que contiene métodos para controlar su funcionamiento mediante parámetros.
  Ejemplos: *UdpEchoServerApplication*, *UdpEchoClientApplication*...

- **Canal:** Los dispositivos se pueden conectar a la red. El medio por el que se transmiten los flujos de datos es lo que se conoce como canal. Se proporciona una clase llamada *Channel* con métodos para gestionar los objetos de comunicación en las subredes y para controlar las conexiones de los nodos al mismo.
  Ejemplos: *CsmaChannel*, *PointToPointChannel*, *WifiChannel*...

    - **Modelo de propagación de la señal:** Al programar simulaciones basadas en comunicaciones inalámbricas, como es el caso de las VANET, el canal de transmisión llevará asociado un modelo de propagación de la señal que represente de qué forma se transmite de un punto a otro.
      Con este modelo se hará constancia de las interferencias u obstáculos que se tiene a la hora de transmitir por el medio inalámbrico.

      Para realizar la evaluación de este experimento se quiso estudiar cómo responden los protocolos de enrutamiento a los efectos de atenuación de la señal según la distancia. Por ello se implementó el modelo de propagación conocido como *log-normal shadowing*[44].

- **Net Device:** Modela la tarjeta de red o *Network Interface Card (NIC)*.
  En implementaciones reales necesita un controlador software para que comunique y maneje su interfaz hardware habilitándola para funcionar en los nodos.
  La abstracción dada en ns3 de este concepto cubre tanto la parte del hardware como la parte del controlador.
  Estos dispositivos se instalan en un nodo para dotar al mismo con capacidad para comunicarse con el resto de nodos adheridos al canal.
  Se representan en *C++* mediante la clase *NetDevice*, que contiene métodos para gestionar conexiones entre objetos *Node* y *Channel*.
  Ejemplos: *CsmaNetDevice*, *PointToPointNetDevice*, *WifiNetDevice*…

- **Helpers:** Ayudan a gestionar las redes y los nodos creados de una manera más sencilla y centralizada. Encargados de tareas como conectar *NetDevices* con *Channels*, asignar direcciones IP o configurar la pila de protocolos, entre otras.

- **Protocolo diseñado, que heredará de *Ipv4RoutingProtocol*:** Se encarga de la lógica de enrutamiento descrita a fondo en los siguientes apartados.

- **Modelo de movilidad, que hereda de *MobilityModel*:** Se encarga de representar el movimiento de los nodos dentro del escenario de simulación.
  El simulador ns3 se alimenta de trazas SUMO para conocer en tiempo de ejecución dónde se encuentra cada nodo.

- **Clase Simulator:** Planifica el flujo de eventos que se dan en nuestra simulación. Cuenta con un elaborado sistema de *callbacks* que permiten programar los eventos discretos que deban darse en la simulación de manera precisa.

2.2.2 *Simulator of urban mobility, SUMO[32][33]*

Las simulaciones de tráfico fueron pensadas para evaluar los cambios en las infraestructuras de comunicación antes de llevarlos a cabo. Por ejemplo, se puede evaluar la efectividad de los algoritmos de control de luces de tráfico, haciendo que estos puedan ser optimizados antes de llevarlos a un escenario real.

Aunque fuese ideado para tareas directamente relacionadas con la fluidez del tráfico, en la investigación relacionada con las redes VANET su uso se ha popularizado, al estar estas investigaciones evaluando protocolos y técnicas cuyo funcionamiento se ve directamente afectado por el movimiento de los nodos.

El simulador SUMO se presenta destacando su portabilidad y su carácter *open source*, además de su capacidad para generar simulaciones de tráfico continuo y microscópico, pudiendo manejarlas con una enorme cantidad de nodos en movimiento interactuando para no colisionar mientras respetan ciertas normas de circulación expuestas en la simulación. Está programado en C++ y hace uso únicamente de librerías *portable*.

Es la herramienta utilizada para generar el escenario dónde se moverán los nodos en cada experimento. Se opta por este simulador al quedar comprobada su compatibilidad con ns3 y por lo fiables que resultan sus trazas.

Se ha de seguir un proceso a la hora de generar escenarios de tráfico, creando una serie de ficheros de configuración que los modelen dentro del simulador.

2.2.2.1 Proceso de generación de escenarios y posterior exportación a ns3

Es necesario crear un escenario virtual, donde los nodos de la simulación de tráfico puedan circular, por ello la primera herramienta a la que se recurre en este proceso es *NETEDIT*.

Esta aplicación permite controlar gráficamente como son las carreteras del escenario y que interconexiones se realizan entre ellas. Como resultado final se obtiene un fichero XML con la representación del modelo creado.

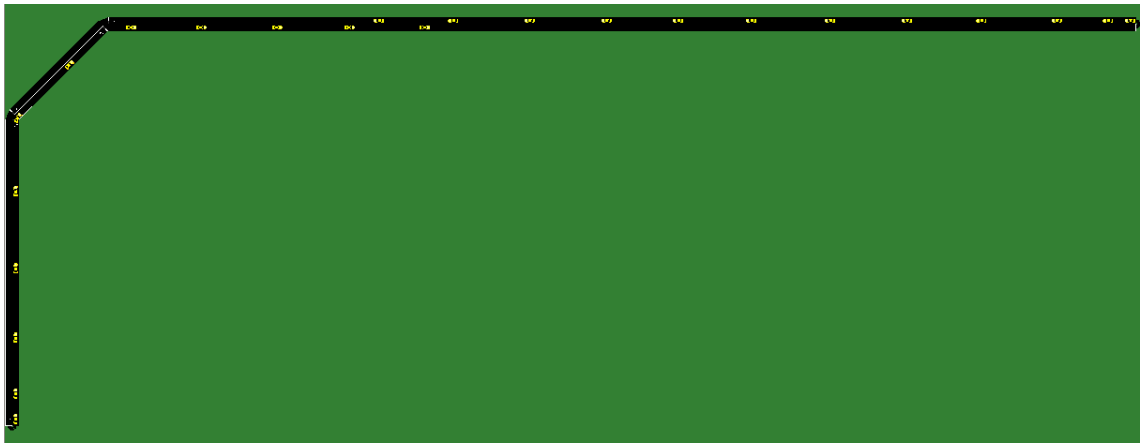


Ilustración 7. Escenario resultante del XML creado con Netedit

Ya con el XML del escenario, se necesita regular como se inyecta el tráfico dentro del mismo. Surge por tanto el llamado fichero de rutas, que contiene aspectos sobre cómo se moverán los vehículos, que tipos de coches hay en nuestra simulación, si hay o no pasajeros…

En el desarrollo de este trabajo, fue necesario crear un *script* generador de rutas para controlar en la medida de lo posible que los nodos no se alejasen más de la cuenta. Así se puede evaluar el protocolo bajo una red que se mantiene conexa durante toda la simulación.

```
<routes>
<vType id="car" type="passenger" length="5" accel="3.5" decel="2.2" departSpeed="25" arrivalSpeed="25" maxSpeed="25" />
<route id="routeIn1" color="1,1,0" edges="InputToCurveE CurveToLongE CurveToOutputE"/>
<route id="routeIn2" color="1,1,0" edges="CurveToLongE CurveToOutputE"/>
<route id="routeIn3" color="1,1,0" edges="CurveToOutputE "/>
<route id="routeOut" color="1,1,0" edges="OutputToCurveE CurveToShortE CurveToInputE"/>
<vehicle id="0" type="car" route="routeIn1" departPos="0" depart="0" color="1,0,0"/>
<vehicle id="1" type="car" route="routeIn1" departPos="20" depart="0" color="1,0,0"/>
<vehicle id="2" type="car" route="routeIn1" departPos="40" depart="0" color="1,0,0"/>
<vehicle id="3" type="car" route="routeIn1" departPos="60" depart="0" color="1,0,0"/>
...
<vehicle id="148" type="car" route="routeOut" departPos="1460" depart="0" color="1,0,0"/>
<vehicle id="149" type="car" route="routeOut" departPos="1480" depart="0" color="1,0,0"/>
</routes>
```

Ilustración 8. Fichero de rutas generado por script (id 4 a 147 suprimidos en la imagen)

Hecho lo anterior, se procede a crear un fichero de configuración para SUMO. Mediante este fichero se apunta a los dos ficheros anteriores, de esta forma el programa creará el escenario a simular que hemos preparado.

```
<configuration>
    <input>
       <net-file value="Highway500_4000.net.xml"/>
       <route-files value="flow150Nodos25ms.rou.xml"/>
    </input>
</configuration>
```

Ilustración 9. Fichero de configuración a utilizar por SUMO para lanzar la simulación

Seguidamente se usa la herramienta *Tracexporter* del simulador. Permite generar una traza que sirve como *input* a nuestra simulación ns3.

Con la traza resultante aplicada a los nodos de la simulación ns3 se puede observar como los nodos se mueven acorde a lo generado en SUMO.

2.3 Técnicas inteligentes

2.3.1 Sistemas expertos basados en reglas

Los sistemas expertos basados en reglas[48][49] toman decisiones siguiendo un conjunto de reglas de entrada. Tratan de emular el comportamiento humano, al interpretar distintos tipos de información que influirán en la toma de decisión final.

Se comportan bien en procesos de control, predicciones relativamente sencillas, algunas estimaciones o en sistemas de decisión, como en este caso.

2.3.2 Redes neuronales[35][36]

Un *Artificial Neural Network* (ANN) es un paradigma del procesamiento de datos inspirado por la manera que tienen los sistemas nerviosos biológicos para procesar información.

Su clave reside en la estructura que se ha de formar, constituida por elementos altamente interconectados, conocidos como neuronas. Estos trabajan en conjunto para resolver problemas después de haber aprendido por exposiciones continuadas a los mismos (técnica del aprendizaje por ejemplo).

Una red neuronal entrenada puede considerarse como un sistema “experto” en la clase de información que ha analizado mediante su entrenamiento.

Las decisiones de enrutamiento del protocolo implementado se toman de forma razonada en tiempo de ejecución, es por ello que se ha pensado en evaluar un protocolo adaptado para que tome acción mediante una red neuronal previamente entrenada.

Entre sus ventajas tenemos:

1. **Aprendizaje a partir de *datasets* de entrenamiento**, que incluyen entradas con información sobre cómo se realiza la tarea que debe aprender. Estos se componen de datos de entrada, acompañados de sus correspondientes salidas después de la tarea realizada.

2. **Auto-organización**, ya que crean su representación de la información que reciben durante el mismo proceso de entrenamiento, así cada iteración del mismo les otorga una mayor precisión en la tarea que resuelven.
3. **Operación a tiempo real:** Las computaciones en un ANN pueden ser llevadas a cabo en paralelo, el hardware diseñado orientado a la computación paralela explota mejor las capacidades de este tipo de redes.
4. **Tolerancia a fallos mediante codificación redundante de información:** Gracias a esta técnica una red neuronal podrá soportar la disfunción de una de sus partes asumiendo únicamente una degradación en su rendimiento y la posible pérdida de ciertas capacidades.

La red se compone por un gran número de perceptrones, que modelan de forma básica el comportamiento de las neuronas humanas, planteado en 1940 por *McCulloch y Pitts*, que combinaron sus conocimientos en neurociencia y matemáticas[34]. Por si solos, estos elementos no realizan una tarea de utilidad, sin embargo, se ha demostrado que en combinación con otros perceptrones se pueden diseñar sistemas con habilidad para aprender a base de datos de ejemplo.

La red neuronal, por tanto, estará compuesta por cierto número de elementos de procesamiento (neuronas o perceptrones) fuertemente interconectados unos con otros y trabajando en paralelo para resolver un problema específico.

Este tipo de redes aprenden por ejemplo, y por tanto no se pueden programar explícitamente para llevar a cabo una tarea específica, algo que es característico de los sistemas computacionales. Los ejemplos escogidos deben ser lo suficientemente representativos para que la red pueda dar resultados válidos en cualquier caso que se le presente, si esto no se cumple aparecerá el denominado *overfitting*[37] y se habrá gastado tiempo y recursos de procesamiento en vano.

Cada vez es más frecuente el uso de ANN para realizar tareas en combinación con sistemas de computación tradicional. Cada una de las dos partes realizará tareas para las que sea más eficiente. De esta forma se consiguen sistemas que puedan operar con una mayor eficiencia.

Es un hecho que algunos problemas están fuera del alcance de la computación tradicional por no tener la potencia suficiente para llevarse a cabo, o bien suponen un consumo energético excesivo. Con esta nueva aproximación, en muchas ocasiones estos problemas se pueden resolver y con un consumo energético mucho menor, sin sobrecargar tanto al computador que los ejecuta.

2.3.3 OpenNN[38][39]

Este es el *framework* utilizado para el desarrollo de la red neuronal que se encargará de tomar las decisiones de enrutamiento en la propuesta correspondiente al protocolo basado en un ANN.

Escrito en C++ y de carácter *open source*, es capaz de implementar cualquier número de capas de perceptrones para generar una red con amplias capacidades para el aprendizaje supervisado.

Se destaca su alto rendimiento al estar implementada en C++, decisión tomada para gestionar a grano fino la gestión de la memoria y obtener mejores velocidades de procesamiento, además implementan paralelismo mediante *OpenMP* y *NVIDIA Cuda*.

2.3.3.1 Partes básicas para desarrollo de Artificial Neural Networks

A la hora de desarrollar un *ANN*, se cuenta con tres piezas fundamentales, que constituyen los pasos básicos a la hora de diseñarlas:

- **Elección de la arquitectura:** Supondrá la base de la función, parametrizada con varias variables, de forma que tenga unas muy buenas capacidades para aproximar los resultados que se esperan.
  - Criterios de elección:
    - Número de capas
    - Número de neuronas por capa
    - Conectividad entre capas
  - Extensiones añadidas en la arquitectura del perceptrón multicapa:
    - *Scaling layer*
    - *Unscaling layer*
    - *Bounding layer*
    - *Probabilistic layer*
    - *Conditions layer*

- **Elección del *Loss Index:*** Define la tarea a realizar, y provee al sistema de una medida de la calidad de la representación que se obtiene mediante la red neuronal. Así se decide cuál es la calidad final que buscamos en caso de ser alcanzable.
  - Términos que lo componen:
    - **Error:** Necesario y suficiente en la mayoría de los casos. Mide la distancia obtenida entre la muestra deseada y la salida de nuestro sistema.
    - **Regularización:** Puede ser necesario según los requisitos de la solución. Consiste en añadir términos adicionales para penalizar a los componentes de la red que tengan un mayor peso y por tanto una mayor importancia, que probablemente acabase empeorando la toma de decisiones al no tener en cuenta los componentes de menor importancia [40].
    - **Restricciones o *constraints*:** Según los requisitos de la solución, la salida de nuestro sistema puede estar restringida en ciertos rangos.

  - Evaluación distinta según el problema al que se enfrenten:
    - **Evaluación mediante *datasets***
      - *Function regression*
      - *Pattern matching*
    - **Evaluación mediante modelos matemáticos**
      - Control óptimo
      - *Shape design*
    - **Evaluación mediante *datasets* y modelos matemáticos**

- Problemas inversos

- **Elección de la estrategia de entrenamiento:** Se elige la mejor dependiendo del problema que se tiene delante.
  - Es la solución al problema de optimización, consistente en conseguir el mejor conjunto de parámetros de configuración para nuestra red neuronal.
  - La más usada por su velocidad y eficacia es la estrategia de *Quasi-Newton*. Es la elección tomada para este desarrollo.
  - Hay otras para problemas más complejos, como por ejemplo el algoritmo revolucionario.

Tan necesarios son los tres elementos formulados arriba como la representación del conocimiento que queremos que nuestra red neuronal sea capaz de emular. Esta representación viene dada en forma de *datasets*, que contienen información en forma de variables, clasificables en tres tipos distintos:

- ***Inputs:*** Variables de entrada, independientes en el modelo.
- ***Outputs:*** Variables de salida, dependientes de las variables de entrada.
- **Variables no usadas:** No se tienen en cuenta en la representación del conocimiento a emular.

Se buscará que la red neuronal aprenda habiéndose enfrentado al problema que busca resolver un gran número de veces, de forma que haya podido encontrar unos parámetros que le permitan obtener una salida más precisa a la hora de predecir los *Outputs* de cada muestra.

Podemos realizar una clasificación de los *dataset*, que se englobarían en cuatro grupos:

- **Instancias de entrenamiento:** Ayudan a construir el modelo.
- **Instancias de selección:** Seleccionan el orden óptimo del modelo.
- **Instancias de test:** Validan el correcto funcionamiento del modelo desarrollado.
- **Instancias no usadas:** No se tienen en cuenta en todo el desarrollo del modelo.

2.3.3.2 Problemas del tipo function regression

También llamados *modelling*, son una de las tareas más frecuentes a resolver por las ANN. Pretenden conseguir una función que aproxime lo máximo posible sus *outputs* al valor deseado correspondiente a los *inputs* que se le han pasado.

En este tipo de problemas se utilizan los *datasets* donde distinguiremos entre variables de entrada y variables de tipo *target*, que serán las variables a aproximar por el modelo de red neuronal.

Las variables de entrada podrán tener un carácter cualitativo o cuantitativo, mientras que los *target* únicamente podrán ser cuantitativas, dado el hecho de que nos encontramos ante un problema de aproximación de valores continuos.

El *loss index* para este tipo de problemas se suele basar en la suma de errores entre las salidas del ANN y las salidas que esperábamos contenidas en el *dataset* de entrenamiento.

En caso de tener datos de entrenamiento deficientes y que no de la misma importancia a todos los casos, puede ser necesario un término de regularización, cuyo concepto ha sido planteado anteriormente.

3. Diseño e Implementación

3.1 Diseño del protocolo

El protocolo propuesto está basado en el protocolo *Stability and reliability aware routing (SRR)* descrito en [11] y del que se presenta una aproximación en [12]. Está adaptado a la comunicación del tipo *V2V* típica de las redes *VANET*.

Requiere conocer la localización de los nodos que forman la red, al ser un **protocolo geográfico**. No mantiene información sobre rutas en su tabla de vecinos, el siguiente salto se calcula de manera *ad-hoc*, es decir, cada vez que se necesita encaminar un paquete hacía un destino se pone en marcha el sistema de decisión, algo característico de un **protocolo reactivo**. Además, está especialmente pensado para redes inalámbricas de tipo *ad-hoc*.

La ruta que seguirá cada paquete del que se haga cargo el protocolo se calculará en cada salto hacia el destino. La decisión sobre que nodo constituirá el siguiente salto será tomada en base a tres parámetros:

- **Distancia nodo intermedio-destino:** A normalizar dentro del radio de alcance inalámbrico de cada nodo.
- **Orientación relativa entre nodos origen, intermedio y destino:** A normalizar utilizando el coseno del ángulo formado entre los tres.
- ***Signal-to-noise ratio (SNR)*:** Influyente en la decisión final, es un medidor de la calidad de la señal.

El protocolo cuenta con dos modos de funcionamiento:

- **El método SRR que se usa en zonas de tráfico denso:**
  El portador del paquete verifica que su tabla de vecinos no esté vacía. En ese caso comprueba si el destino del paquete está entre sus vecinos para entregarle el paquete de forma directa.
  En caso de no contar con el destino en la tabla, se pondrá a funcionar el motor de decisión del protocolo descrito para conocer el siguiente nodo en la ruta hacia el receptor.
  - En la implementación de este trabajo el motor de decisión se ha realizado atendiendo a dos propuestas distintas:
    - **Propuesta basada en sistema experto basado en reglas(SRAR):**
      El motor de inferencia utilizará un sistema experto basado en reglas, emulando el sistema de *fuzzy logic* usado en [14] que con una serie de reglas tratará de imitar a grandes rasgos el razonamiento humano en una situación dada.

      Este motor, instalado en el nodo que quiere conocer el mejor salto hacia el destino, calculará el valor indicador de calidad como *nexthop* para cada nodo vecino, quedándose con el que mayor resultado obtenga.

El valor vendrá dado en base a los tres parámetros comentados anteriormente, obteniendo mejor puntuación para los nodos que estén a distancia media respecto al emisor y que tengan una mejor orientación hacia el destino. Atendiendo a esta distancia y orientación, se clasificará a cada nodo dentro de una regla de salida.

Ahora supongamos que hemos evaluado a todos los vecinos a nuestro alcance y se han obtenido varios nodos dentro de la mejor regla de salida. Es aquí donde entra en juego el ya comentado *SNR*.

Cada regla de salida tendrá un rango de valores continuo [x,y], y el SNR se normaliza dentro de ese rango, de forma que dentro de la misma tendrán preferencia los nodos para los que se haya observado una mayor calidad de la señal.

El paquete podrá ser reenviado con el uso de este sistema tantas veces como lo permita su *Time-to-Live (TTL)*.

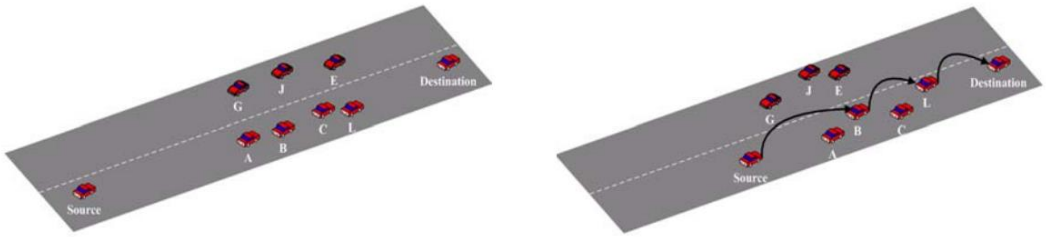
- **Propuesta basada en red neuronal (SRARNN):**
  Con intención de observar si se podría obtener algún beneficio en cuanto a uso de recursos o consumo energético a la vez que se posibilita ampliar considerablemente el número de parámetros a considerar, se ha entrenado una red neuronal capaz de distinguir los nodos que tienen una mejor predisposición a enviar correctamente cada paquete a su destino final. Esta red neuronal será el motor de decisión del protocolo.

- Se prepara la red neuronal mediante scripts que generen muestras de entrenamiento representando los criterios comentados anteriormente, es decir, dar prioridad a los nodos a distancia media del emisor, con mejor direccionalidad hacia el destino y con una calidad de señal aceptable.

  Los scripts generarán, basándose en estos tres *inputs,* la salida que la propia red neuronal tendrá que tratar de aprender, para calcularla de forma directa después de haber sido entrenada.

  En este caso también se cuenta con la restricción del número de reenvíos máximo para cada paquete marcada por el *Time to Live (TTL)*.

- **El método *carry and forward*:** Cuando un nodo que no sea el destino recibe el paquete, en caso de verificar que no tiene datos sobre ningún nodo vecino en su tabla, pasará a este modo, que consistirá en guardar el paquete en una cola que servirá como cache, donde se guardará un tiempo determinado hasta ser descartado o enviado cuando el nodo entre en conectividad.



(a) A source node uses carry-and-forward mechanism to keep packets in unconnected wireless links

(b) A source node switches to SRR protocol when a vehicle has been found within its radio range

Ilustración 10. Modos de funcionamiento del protocolo según escenario

3.1.1 Métricas de entrada al sistema de decisión

La decisión tomada a la hora de calcular el siguiente salto en la ruta hacia el destino necesita de ciertas métricas de entrada que nos permitan caracterizar el valor de cada nodo como *next hop* de forma inmediata. Estas han sido comentadas con anterioridad, y ahora se pasa a realizar un análisis más exhaustivo de las mismas:

3.1.1.1 Métrica relativa a la distancia entre el nodo vecino y el destino:

Se dará mayor valor a los nodos que se encuentren a una distancia intermedia entre origen y destino. Se podría pensar que un nodo con una mayor cercanía al receptor supondría una mejor decisión para el protocolo (de hecho hay protocolos que se basan en este fundamento).

Pero la realidad es bien distinta, y en este desarrollo se valoran mejor las distancias medias por las siguientes razones:

1. Cuanto más nos acercamos al destino, más nos alejamos del emisor, esto supone peligro de atenuación de la señal, que será más susceptible de verse afectada por factores como el ruido o los obstáculos para su propagación.
2. Por otro lado, si nos centramos en las distancias cortas, se tiene que observar que el número de saltos necesarios para alcanzar el destino aumentará, y por tanto también el tiempo de procesamiento del paquete invertido en los saltos intermedios será mayor. Además, distancias más cortas suponen una mayor probabilidad de interferencias, ya que los nodos emitirán en áreas más pequeñas y habrá más probabilidades de colisión.
3. En caso de que un nodo sea vecino en un determinado momento, decidamos enviarle el paquete y este haya cambiado de posición a la hora del envío, no tendremos problema, ya que al haber primado las distancias medias, la probabilidad de que haya salido de la zona de cobertura será mucho menor.

Para el cálculo de distancia entre los nodos se ha utilizado el teorema de Pitágoras, ya que en el simulador el tipo de coordenadas utilizadas es el cartesiano. En el caso de haberse tratado de coordenadas geográficas se debería haber utilizado una fórmula más precisa (fórmula de Haversine[13]) orientada al cálculo de distancias de este tipo.

$$h^2 = x^2 + y^2$$

$$Tenemos\ (x_0, y_0)\ y\ (x_1, y_1)\ como\ coordenadas\ emisor\ e\ intermedio$$

$$Distancia_{emisor-intermedio} = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

Ecuación 1. Fórmula de Pitágoras adaptada a la distancia emisor – intermedio

$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) * \cos(\varphi_2) * sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 * atan2(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R * c$$

Ecuación 2. Fórmula de Haversine donde φ *es la latitud,* λ *es la longitud,* R *es el radio de la tierra (radio = 6,371km)*

3.1.1.2 Métrica relativa a la dirección relativa:

Se calcula entre cada nodo intermedio y destino, respecto al nodo origen. Nos dice como de dirigido hacia el destino está cada nodo vecino. Se valorarán más aquellos nodos que cuenten con una mejor direccionalidad.

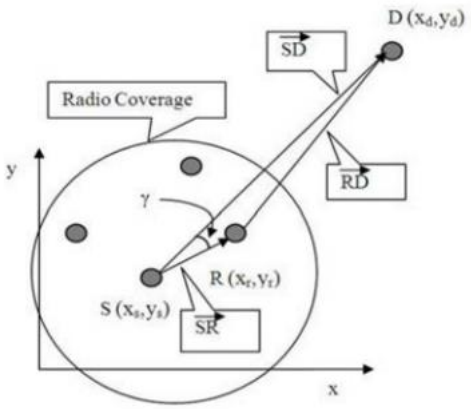


Ilustración 11. Esquema de los nodos involucrados en el cálculo de dirección relativa

Como se aprecia en la imagen, para calcular el *score* del nodo intermedio R, necesitaremos el valor para su dirección relativa con el nodo D desde el punto de vista del nodo origen S.

Para ello debemos comenzar calculando los siguientes vectores

$$\overrightarrow{SR} = < (x_r - x_s), (y_r - y_s) >$$
$$\overrightarrow{SR} = < (x_d - x_r), (y_d - y_r) >$$
$$\overrightarrow{SR} = < (x_d - x_s), (y_d - y_s) >$$

Ecuación 3. Cálculo de los vectores necesarios

Con los vectores calculados anteriormente podremos ser capaces de calcular el ángulo necesario para conocer la métrica de la dirección relativa

$$\cos\gamma = \frac{\overrightarrow{SR} * \overrightarrow{SD}}{|\overrightarrow{SR}| * |\overrightarrow{SD}|}$$

Ecuación 4. Cálculo de dirección relativa

3.1.1.3 Métrica relativa a la calidad de la señal[41]

En las redes inalámbricas la conectividad juega un papel importante a la hora de garantizar un buen servicio, por ello, muchos protocolos y tecnologías de red de la capa de enlace y enrutamiento tienen en cuenta distintas métricas que midan la calidad de la señal.

Una de las más conocidas es el *Signal-to-noise ratio*, que en combinación con otras técnicas [42], puede resultar interesante para conocer la calidad de los enlaces. En este caso, utilizarlo con los nodos que tengan un mejor *score* nos acerca a devolver como mejor vecino al que mejor cualidades tenga para una transmisión efectiva.

Una vez hayamos clasificado el nodo en su clase, dentro de la misma se evaluará que nodos serán mejores que otros utilizando su valor *signal-to-noise-ratio (SNR)* obtenido la última vez que se recibió una señal del mismo.

Por definición es calculado dividiendo la señal recibida entre el ruido, de esta manera observaremos que mayores valores para esta variable supondrán una mayor calidad de la señal. Como se comenta en [41], este valor es clave en cualquier transmisión de señal y debería ser monitorizado para asegurar la calidad de los datos.

De esta forma podemos asignar como mejor vecino a un nodo que será más probable que nos permita un envío correcto, al usar esta medida que nos indica sobre la calidad del enlace por el que se transmiten los datos ambos nodos.

3.1.2 Diseño del sistema experto de decisiones basado en reglas

En el protocolo diseñado, la aplicación de un sistema experto basado en reglas resulta muy útil a la hora de discernir cuál es el nodo vecino con mejores condiciones para enviar el paquete hacia su destino.

3.1.2.1 Adaptación de entradas y salidas

Siendo las métricas anteriores ya conocidas y calculadas correctamente, tendrán que adaptar sus valores a un rango que resulte más fácilmente manejable para el sistema de decisión.

En el caso de la direccionalidad sus valores estarán dentro del rango [-1,1], indicando una mejor direccionalidad cuanto mayor sea su valor. Este valor adaptado está representado por el coseno del ángulo obtenido.

En el caso de la distancia, el rango a manejar será [0.05,0.96] y el valor se adapta teniendo en cuenta el radio de cobertura de la señal. Todos los nodos vecinos estarán dentro del mismo. El realizar la división de distancia emisor-vecino entre el radio de cobertura nos dará un valor que represente la cercanía entre estos dos nodos y por tanto la lejanía entre el vecino y el destino.

$$RDistance = 1 - \frac{D}{R}$$

Ecuación 5. Cálculo del parámetro *RDistance*. R es el radio de alcance *wireless* definido

En base a los dos parámetros se obtiene la regla de salida del sistema experto.

Cada regla de salida tiene asignado un rango continuo de valores. Conociendo el que abarca cada una, podemos normalizar el SNR dentro de cada rango si conocemos sus valores máximo y mínimo.

Dada una regla de salida con un conjunto continuo de valores [x,y] y conocidos los [min,max] del SNR podremos normalizar el segundo valor en el rango continuo de la regla.

Por ejemplo, supongamos que tenemos un valor x de SNR, que supone el recorrido de un 25 % del rango [min,max] comentado. Sabiendo esto, la normalización a realizar corresponde a recorrer un 25 % del rango asociado a la regla de salida correspondiente.

Hecho esto se consigue, que dentro de una misma regla de salida, en la que los nodos en principio tienen condiciones similares, podamos ordenarlos por calidad de enlace, reduciendo así la posibilidad de un enlace defectuoso.

3.1.2.2 Sistema experto basado en reglas:

	IF		**THEN**
Regla	**Distancia**	**Direccionalidad**	**Regla de *salida***
1	Lejos	Baja	Bajo
2	Lejos	Media	Medio
3	Lejos	Alta	Alto
4	Intermedio	Baja	Medio
5	Intermedio	Media	Alto
6	Intermedio	Alta	Muy alto
7	Cerca	Baja	Muy bajo
8	Cerca	Media	Bajo
9	Cerca	Alta	Medio

Tabla 3. Reglas del motor de inferencia

Cada par de reglas de entrada el motor de inferencia calcula una regla de salida, usada para obtener un valor numérico dentro de la misma en función del SNR.

Direccionalidad, es una regla de entrada con tres rangos:

1. Baja: *DegreeDirectness* en el rango [-1,-0.05[
2. Media: *DegreeDirectness* en el rango [-0.05,0.85[
3. Alta: *DegreeDirectness* en el rango [0.62,1]

Distancia, es una regla de entrada con tres rangos:

1. Lejos: *RDistance* en el rango [0,0.35[
2. Intermedio: *RDistance* en el rango [0.35,0.62[
3. Cerca: *RDistance* en el rango [0.62,1]

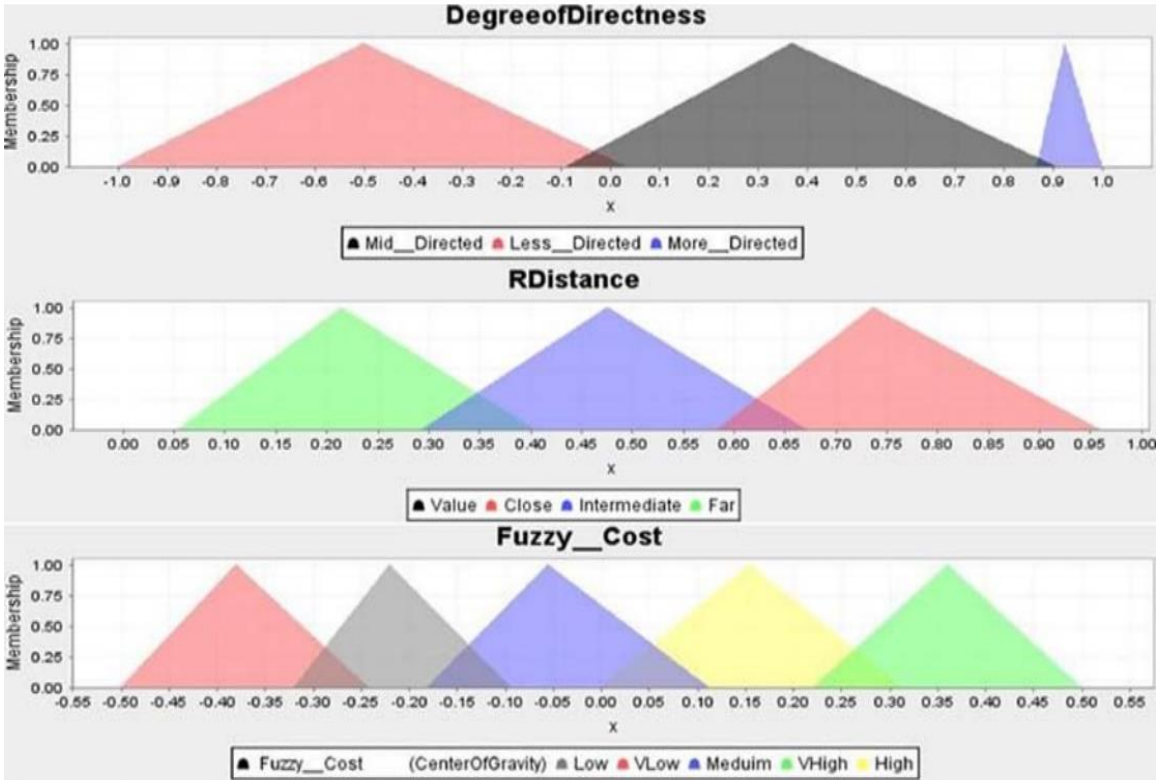


Ilustración 12. Valores de las reglas de entrada y salida en función del valor de sus parámetros

El motor, en base a los valores de distancia y direccionalidad dará una regla de salida. Con todos los vecinos encasillados en una clase de salida, se podrá pasar a la normalización del SNR ya comentada, para los nodos de la regla de salida más prioritaria.

3.1.3 Flujo de procesamiento de paquetes en el protocolo

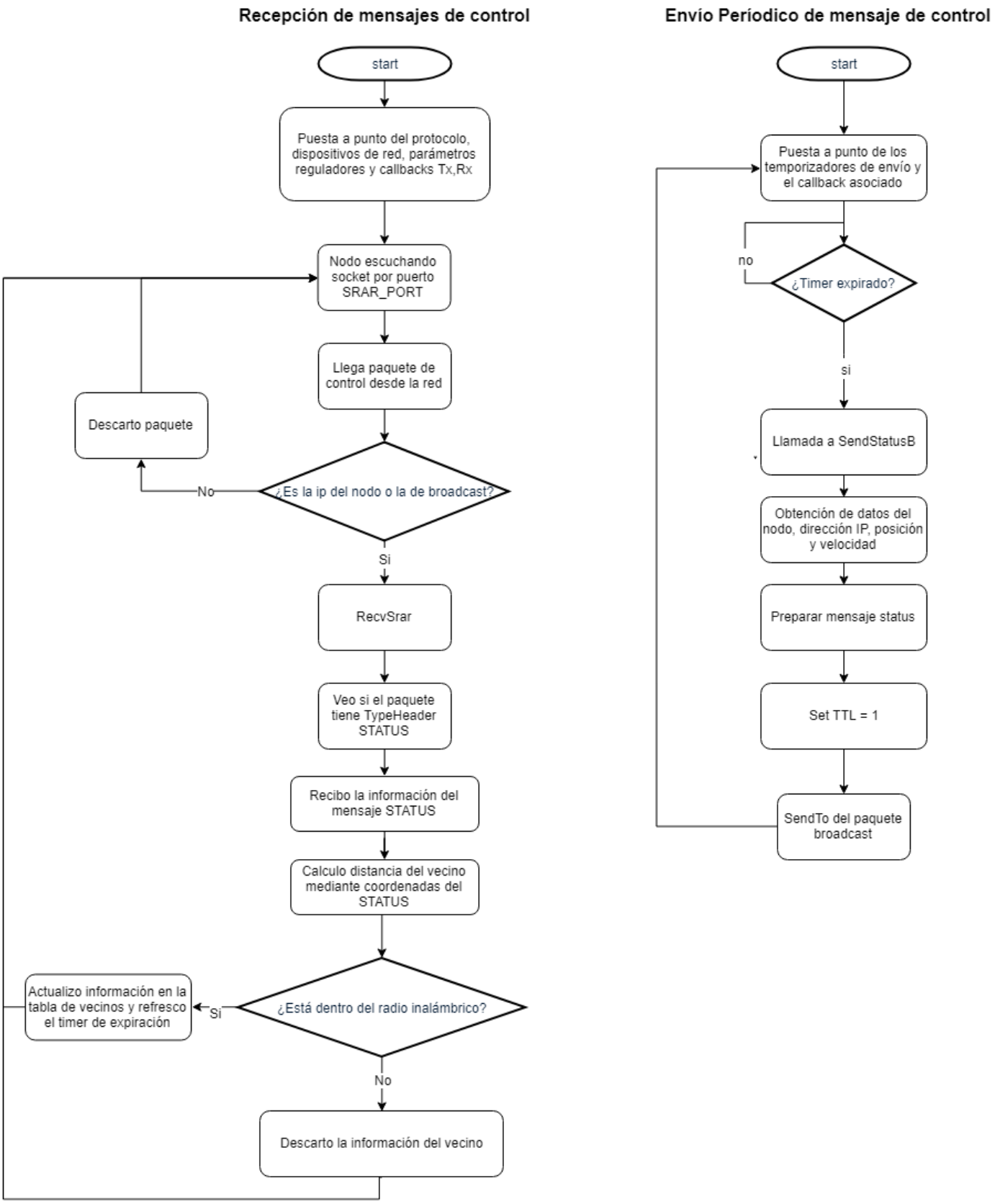


Ilustración 13. Trato de paquetes de control de SRAR y SRARNN

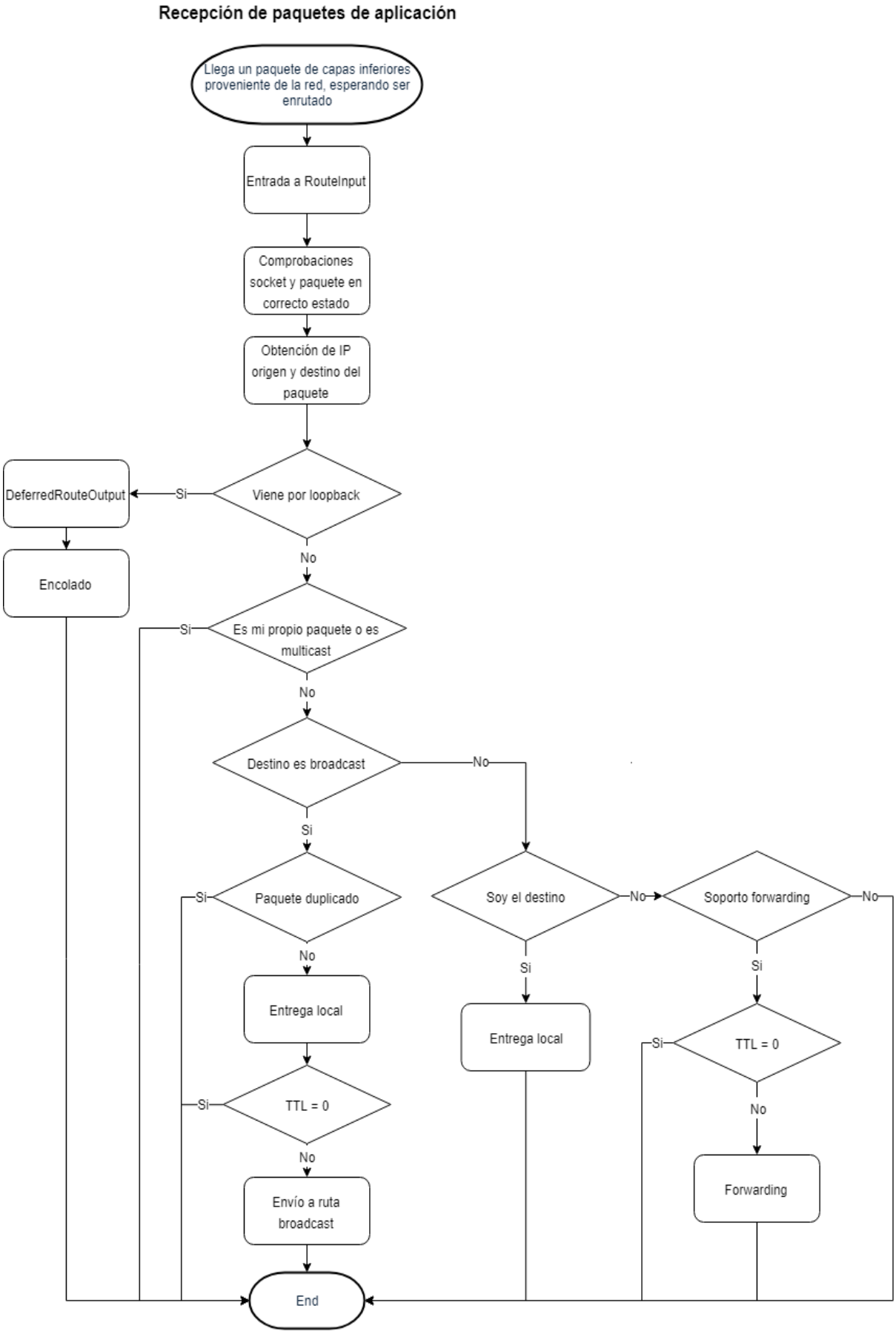


Ilustración 14. Recepción de paquetes de aplicación

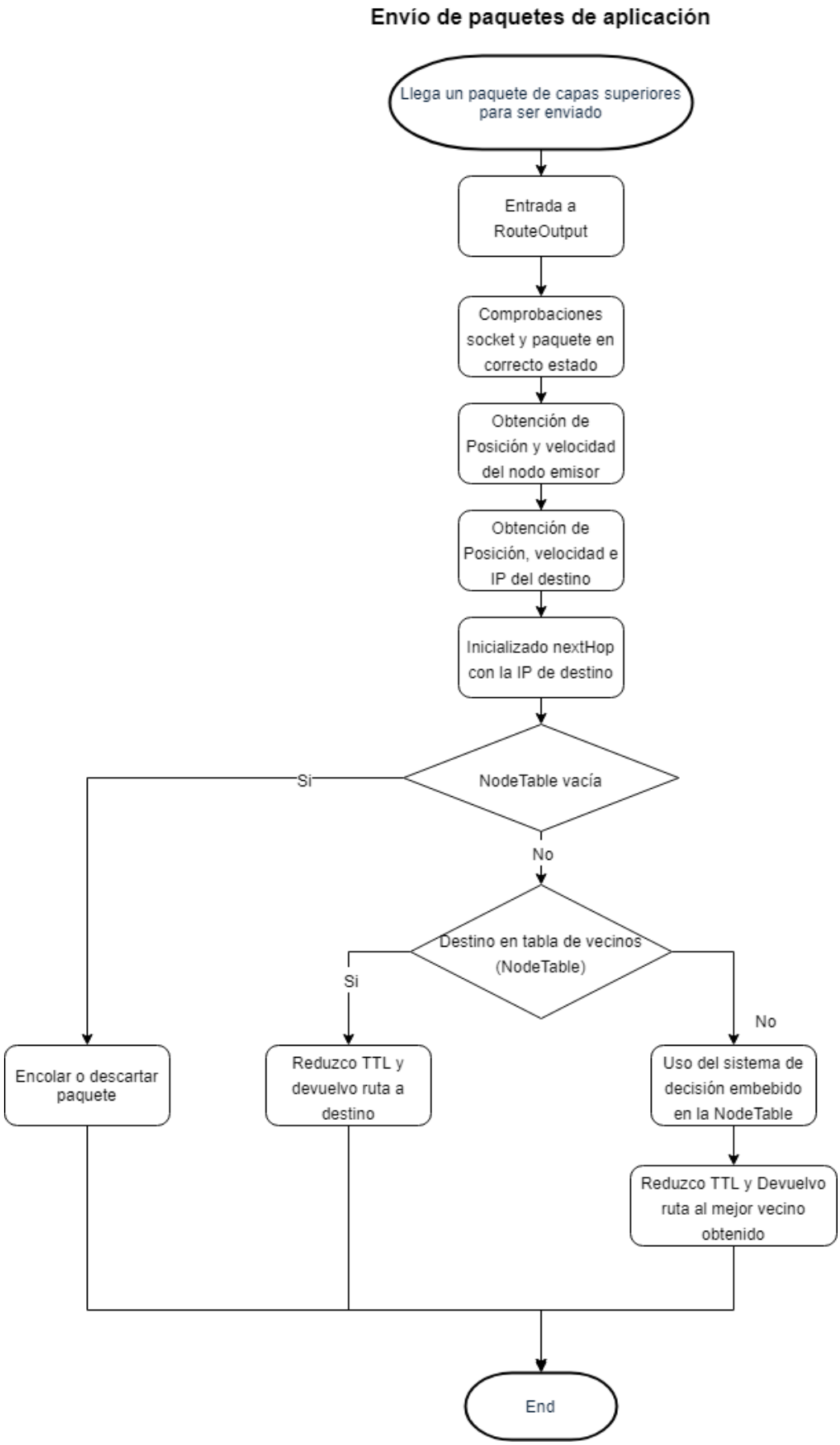


Ilustración 15. Envío de paquetes de aplicación por los protocolos

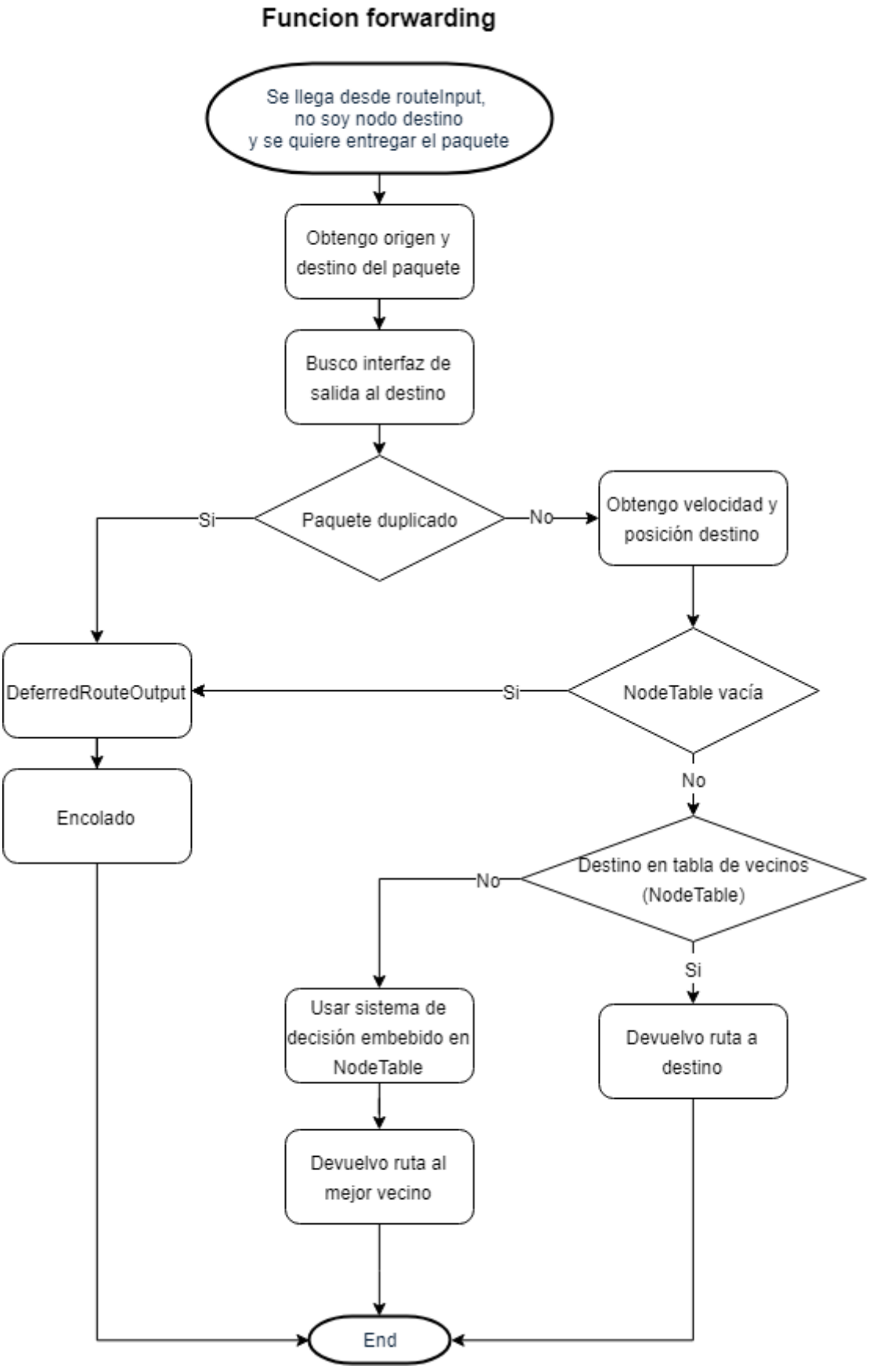


Ilustración 16. Proceso de *forwarding* de los protocolos

3.2 Aspectos de la implementación

Se lleva a cabo la implementación de este protocolo utilizando C++ como lenguaje de programación, desarrollando el correspondiente modelo dentro del entorno de simulación de ns3.

Se puede así generar escenarios con clases capaces de representar modelos que los acerquen a crear situaciones simuladas que se acerquen en la medida de lo posible a la realidad.

El desarrollo se llevó a cabo poco a poco, analizando que elementos serían necesarios para un protocolo de este tipo:

- Se precisa de una estructura de datos que albergue en su interior los datos relativos a los vecinos del nodo que la tiene en memoria, se le llama ***nodetable***.
    - Cuenta con las funciones necesarias para manejar sus entradas de una manera eficiente: evitando entradas duplicadas, refrescando datos y tiempos de expiración a cada mensaje nuevo que llega, eliminando entradas a las que se les ha agotado el tiempo…
    - Contiene dentro un objeto representando al sistema de decisión escogido, por tanto será capaz de devolvernos mediante una función el mejor nodo de sus entradas.
- Para los mensajes de control necesarios para enviar información a los vecinos, que son enviados por el protocolo de forma periódica, se necesita también tener una representación mediante una clase en memoria. La clase donde he creado estos modelos es ***srar-packet***.
- El motor de toma de decisiones se implementa en una clase externa, de forma que pueda ser utilizado en otras implementaciones. La clase asociada es ***srar-routingEngine***.
    - Esta clase tiene en su interior el sistema experto basado en reglas o la red neuronal previamente entrenada. Se usará la opción que se active por línea de comandos.
- La clase central, y la más importante, alberga la lógica central del protocolo en cuanto al tratamiento de los paquetes entrantes y se encarga de envíos y recepciones. Localizada en ***srar.cc***, esta clase merece una mención aparte en los siguientes apartados.
- Una clase que modele de manera abstracta un sistema de localización global de nodos, de manera que podamos obtener información de nodos lejanos sin necesidad de obtener esta información sobrecargando la red con mensajes enviados por técnicas poco deseables, como el *flooding*, que no serían viables ya que restarían mucha escalabilidad al protocolo.
  En este caso, se deja como una abstracción aprovechando el conocimiento global sobre los nodos que tiene el simulador, que nos podrá dar en cada momento la localización de cada nodo.
    - Este sistema debería aportarnos la información en un tiempo razonable y con datos frescos sobre la localización del nodo que se le demanda.
    - La localización de los vecinos sí que tiene que venir dada mediante los paquetes de control del protocolo, para saber con una mayor exactitud dónde se sitúa el vecino, teniendo seguridad sobre si está o no dentro del rango de alcance.

- Otra implementación que fue necesaria de realizar, fue la creación del modelo de propagación *log normal shadowing*[44] dentro del simulador, necesario para la experimentación. Con esto se puede evaluar cómo responde el protocolo al fenómeno de atenuación de la señal con la distancia.

3.2.1 Mensajes utilizados

En el protocolo propuesto se ha optado por el uso de un único paquete de control, que albergará la información necesaria para los vecinos que lo reciben.

Será enviado mediante un *broadcast* con un *Time to Live* puesto a uno, de manera que únicamente sus vecinos lo recibirán, y la red no sufrirá la sobrecarga asociada al reenvío de *broadcasts*. El nombre que se le ha dado a este paquete es *STATUS*

3.2.1.1 Mensaje Status

Creado con la intención de comunicar a los nodos vecinos la información que el protocolo, y más concretamente el motor de decisión (sin importar si hace uso de la red neuronal o del sistema experto) necesitan para funcionar.

Este mensaje está limitado a un salto para evitar que la red se inunde con paquetes innecesarios, y contiene los datos mostrados en el siguiente esquema.

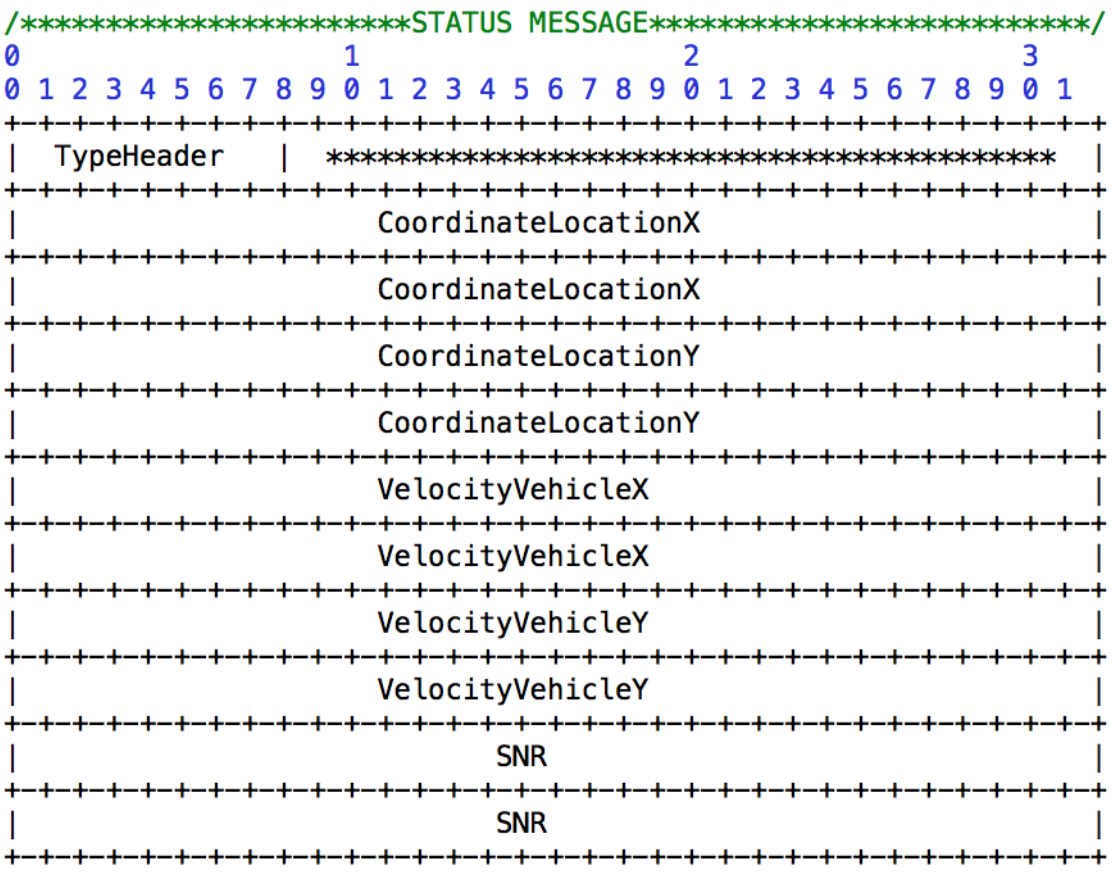


Ilustración 17. Esquema de datos del paquete STATUS

El campo *typeheader* se ha dejado en el paquete por si en el futuro se pretende añadir alguna funcionalidad más al protocolo que requiera de nuevos tipos de paquetes, de

esta forma los podríamos distinguir mediante este campo y no sería necesario cambiar la estructura ya existente.

El resto de información ya es conocida, además del *SNR* y las coordenadas, ambos necesarios para la toma de decisiones, se añade el parámetro de la velocidad de los nodos, que también podría resultar interesante a la hora de predecir distancias futuras y tomar prevención en ciertas situaciones, en todo caso este campo no tiene todavía un uso determinado dentro del protocolo.

3.2.2 Clase central del protocolo SRAR

Dentro del simulador ns3 vienen implementados por defecto múltiples protocolos de enrutamiento, con propósitos de todo tipo y orientados a distintos tipos de redes. Sin embargo, hay algo que comparten todas, heredan de una clase padre llamada *Ipv4RoutingProtocol* que proporciona una *API* necesaria siempre que se quiera desarrollar un elemento de este tipo dentro del simulador.

La api necesaria se resume en estas funciones:

- **Clases destructoras** del objeto necesarias para la gestión correcta de la memoria dentro del simulador.
- **RouteInput** como función encargada de conectar la capa de *routing* en nuestro nodo con las capas inferiores.
  Recibe paquetes que vienen del canal inalámbrico y decide si le corresponde al protocolo de enrutamiento que lo implementa aplicar su lógica. De ser así, dicta si le corresponde entregarlo localmente, realizar reenvío al siguiente salto o por el contrario es un paquete destinado a otro tipo de protocolos del mismo nivel.
- **RouteOutput** sirve para buscar la ruta de salida para el paquete que quiere salir a la red pasando por las capas inferiores. Se pregunta a la cache de rutas cual es la idónea para enviar al destino que se plantea.
- **NotifyInterfaceUp/Down** cuentan con la lógica necesaria para notificar al protocolo que se cuenta con una alta/baja de una interfaz de red utilizada por el protocolo.
- **NotifyAddAddress y NotifyRemoveAddress** para notificar las asignaciones de direcciones IP en las interfaces de red del nodo usadas por el protocolo.
- **SetIpv4** se encarga de realizar el paso necesario de asociar el protocolo de enrutamiento con el objeto de tipo ipv4 del nodo.
- **PrintRoutingTable**, una función que en el marco de este trabajo permite obtener información en tiempo real sobre el estado de los nodos dentro del radio inalámbrico.

Además, al protocolo se han añadido más funciones que ayudan a dar forma a la funcionalidad que se espera obtener:

- **Start:** Arranca el protocolo de enrutamiento, se encarga de poner a punto los *callbacks* destinados al envío periódico de los mensajes de control y de poner los parámetros necesarios al valor correspondiente para que el protocolo funcione.
- **DeferredRouteOutput:** Llamada cuando se desea añadir un paquete a la cola de mensajes.

- **Forwarding:** Usada en el caso en el que el nodo receptor no es el destino final del paquete, y necesita utilizar el sistema de decisión creado para calcular el siguiente salto.
- **IsMyOwnAddress, findSocketWithInterfaceAddress, findInterfaceWithIpv4Address, FindSocketWithIpv4Address, FindSocketWithIpv4BroadcastAddress, FindInterfaceForAGivenDst:** Destinadas a conocer las interfaces por las que debemos enviar o por la que hemos recibido un determinado paquete.
- **LoopbackRoute, LoopbackRouteForwarding:** Devuelven la ruta necesaria en caso de que queramos guardar el paquete en la cola de mensajes que no han podido ser enviados, de forma que el nodo receptor se envía el paquete a sí mismo para poder encolarlo utilizado su *callback* asociado a la recepción de un paquete *loopback*.
- **RecvSrar:** Se encarga de la entrada del paquete de control cuando el protocolo lo recibe en el nodo destino, de esta forma sus datos serán redirigidos a la función que los extraerá para añadirlos al conocimiento sobre los vecinos que tiene el protocolo.
- **RecvStatus:** Trata los mensajes de este tipo y añade o refresca la entrada relativa al emisor del paquete con nuevos datos relativos a la velocidad, posición y SNR del mismo.
- **CalculateDistance:** Ayuda a conocer la distancia actual del nodo con el emisor del paquete, de esta forma se puede hacer que si el paquete ha llegado fuera de rango de cobertura esta información no sea añadida a la tabla de vecinos, ya que resultaría un vecino problemático.
- **SendStatusB:** Envía los mensajes *status* mediante *broadcasts* de un salto, se prepara un *callback* con un *timer* que lo lance en intervalos periódicos regulables por parámetro.
- **SendTo:** Permite escoger el socket por el que queremos enviar un paquete y su destino.
- **StatusTimerExpire:** *Timer* que tiene asociado el *callback* que lance la función SendStatusB de forma periódica.

Todas estas funciones modelan el comportamiento general del protocolo.

En cuanto a las variables internas del mismo tenemos:

- **m_range** define el rango de cobertura que manejan los nodos que utilizan el protocolo.
- **m_lastStatusTime** se usa para controlar que los status se están enviando en periodos correctos y no enviar más de los necesarios.
- **m_statusTimer** es el *timer* para el envío periódico del mensaje de control
- **m_nodetable** representa la tabla de vecinos de cada nodo.
- **m_locationService** será objeto de una clase que se encargará de aprovechar el conocimiento global sobre los nodos que tiene el simulador para crear la abstracción comentada del sistema de localización global de nodos.
- **LocationServiceName** es una variable que contiene el modelo de localización de nodos empleado, en caso de tener añadir otras opciones posibles en la implementación.

- **m_lo** es la interfaz de *loopback*.
- **m_socketAddresses** sirve como estructura para guardar las direcciones IP asociadas a cada interfaz en el nodo.
- **m_socketSubnetBroadcastAddresses** es una estructura para guardar las direcciones de *broadcast* orientados a subredes de cada interfaz en el nodo.
- **m_ipv4** es el objeto ipv4 con el que debe comunicarse el protocolo de enrutamiento en el simulador.
- **m_statusInterval** es una variable que regula el periodo de envío de los mensajes STATUS de control.
- **m_ttlStart** con esto se indica el TTL inicial de los paquetes de control del protocolo, que en nuestro caso al estar pensado para hacer *broadcasts* de un salto tendrá un valor de uno.
- **m_neighborTimeout** indicará la validez en términos de tiempo de cada entrada en la tabla de vecinos. Si se supera su valor sin recibir noticias del nodo vecino asociado a la entrada, sus datos serán borrados de la caché de vecinos.
- **SRAR_PORT** es el puerto usado por el protocolo para el envío de sus paquetes de control.

3.2.3 Clase RoutingEngine.cc

Su API permite asignar un *score* a cada nodo intermedio mediante su función costeNodo, que recibe como parámetros estructuras representando a los nodos origen, intermedio y destino, que contienen la información necesaria (SNR, localización y dirección IP) para realizar los cálculos y saber a quién pertenece la puntuación obtenida.

Como funciones a destacar tenemos:

- **costeNodo:** Se la llama desde dentro de la tabla de vecinos para obtener el score que tiene cada nodo respecto a un origen y un destino, todos pasados como parámetro en forma de estructuras albergando en su interior la información necesaria para que el sistema de decisión pueda desarrollar su función.
- **calculaDistanciaEntreNodos**
- **deg2rad:** convierte grados a radianes, algo necesario según la lógica descrita.
- **calcula_RDistance:** calculará el parámetro al que se refiere.
- **calcula_DegreeDirectness:** calculará el parámetro al que se refiere.
- **function_expertSystemCost:** función llamada en el caso de querer utilizar el sistema experto basado en reglas.
- **function_neuralNetworkCost:** función llamada en el caso de querer utilizar la red neuronal para tomar la decisión.
- **normalizeValue:** para normalizar el valor del SNR dentro del rango asociado a cada regla de salida indicativa de la calidad de un nodo para ser elegido como vecino.

Como parámetro importante tenemos m_range, que marca el alcance inalámbrico al que llegan los nodos de la red y que normaliza el valor *RDistance* al rango [0,1].

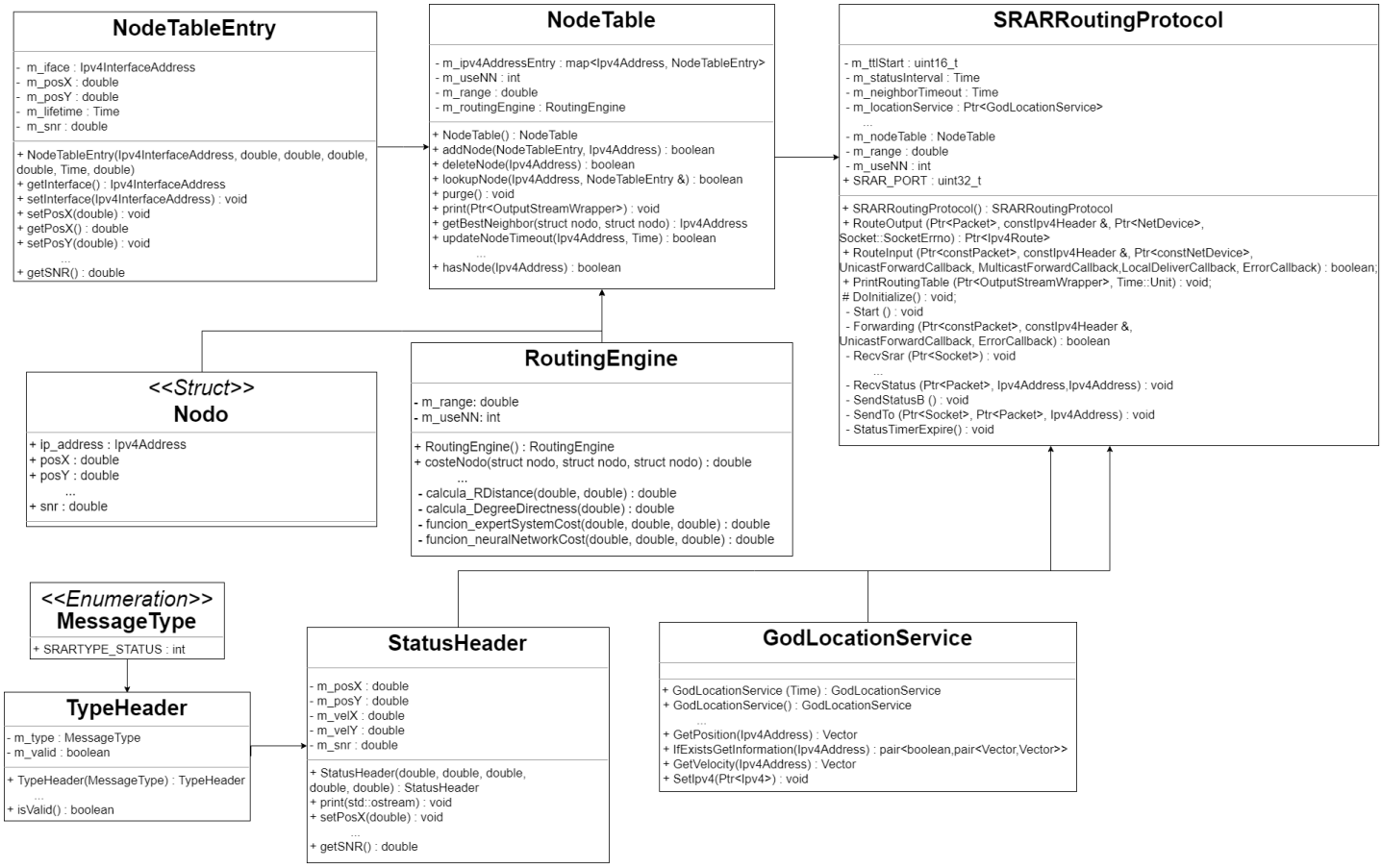


Ilustración 18. Diagrama de clases de los protocolos de enrutamiento implementados

3.2.4 Implementación del modelo de propagación de la señal *log normal shadowing*

El simulador ns3 cuenta en su código con diferentes modelos de propagación de la señal, que representan las pérdidas que sufre la señal al ser transmitida por el canal.

En este caso, no se ha optado por el uso de ningún modelo de propagación que viniese implementado por el simulador, ya que uno de los intereses principales de la experimentación era la evaluación del impacto del efecto de atenuación de la señal sobre el rendimiento de cada protocolo.

El modelo más ajustado a representar este efecto de atenuación, resultó ser el llamado *log normal shadowing*[44].

3.2.4.1 Ficheros donde se ha realizado la implementación

El simulador tiene el código organizado por módulos, cada uno representando componentes que realizan una función similar. Es por ello que la implementación de este modelo de propagación se ubica en los ficheros propagation-loss-model.cc y propagation-loss-model.h.

Estos ficheros implementan distintos modelos de propagación por defecto, pero no se ajustaban lo suficiente al comportamiento esperado para el tipo de experimento a

realizar. El .h contiene la declaración de atributos y funciones usadas en el modelo. El .cc los atributos configurables y lo necesario para configurarlos, así como la implementación de las funciones más importantes.

3.2.4.2 Código asociado a la implementación

Fórmula asociada

El código ha sido implementado en base a la siguiente fórmula:

$$P_{rx} = P_{rx0} - 10 * n * \log\left(\frac{d}{d0}\right) + X_{\sigma}$$

$Siendo$: P_{rx0} $la\ potencia\ de\ recepción\ a\ la\ distancia\ de\ referencia$
$n\ el\ path\ loss\ exponent$[45]
$d\ la\ distancia\ para\ la\ que\ se\ calcula\ P_{rx}$
$d0\ la\ distancia\ de\ referencia\ asociada\ a\ P_{rx0}$
$X_{\sigma}\ la\ variable\ gaussiana\ aleatoria, con\ desviación\ típica\ \sigma$

Ecuación 6. Modelo de propagación *log normal shadowing*

Cálculo del *reference loss* a la distancia de referencia

Se usa la ecuación de *Friis* para calcular la potencia de recepción de referencia, al ser esta la asociada al espacio libre, con menores dificultades para la transmisión de datos. Este valor supone el tope de potencia de recepción. Los factores asociados al *shadowing* lo reducirán, en función de la calidad del canal y la distancia.

$$P_{rx} = P_{tx} + G_{tx} + G_{rx} - 20 * \log(\frac{4 * \pi * d}{lambda})$$

$Siendo$: $G_{tx} y\ G_{rx} las\ ganancias\ de\ transmisión\ y\ recepción$
$lambda\ la\ longitud\ de\ onda\ de\ la\ señal$

Ecuación 7. Modelo de propagación de radio de *Friis*

Generación de la variable aleatoria con distribución normal

Se hace uso de la clase *NormalRandomVariable* para generar números aleatorios que sigan una distribución Gaussiana. Esta generación de números se verá afectada por el parámetro σ del modelo de propagación implementado. Que generará mayores dificultades para la transmisión a medida que se aumenta.

Cálculo de la potencia de recepción asociada al modelo *log normal shadowing*

Como se observa en la ecuación 6, la combinación de lo anterior es lo que proporciona el valor de la potencia recibida en un receptor que se encuentra en un medio con los efectos del modelo de atenuación de la señal con la distancia.

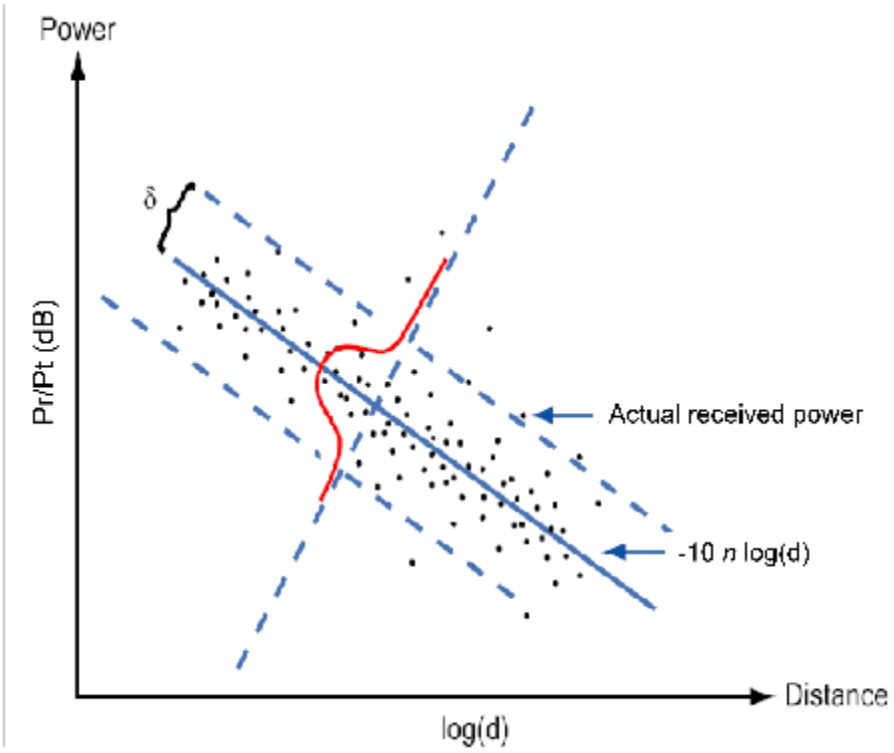


Ilustración 19. Resultados del modelo de propagación *log normal shadowing*

En la imagen se ve el comportamiento obtenido con este modelo de propagación.
La distancia penaliza a la potencia de recepción, y el valor de σ contribuye a ensanchar la campana de *gauss*, alejando el valor de P_{rx} del esperado.

3.2.5 Funcionamiento de la aplicación: script asociado

Una vez realizada la implementación del protocolo de enrutamiento que irá embebido dentro de la pila de protocolos de cada nodo, para probarlo es necesario crear un script de simulación que genere un escenario donde realizar pruebas y recolectar los datos resultado de las mismas.

En este caso el script asociado tiene el nombre *srr-log-normal-shadowing.cc*, y dentro modela un escenario con las siguientes características y posibilidades.

3.2.5.1 Pila de protocolos asociada a los nodos: configuración del nodo

CreateNodes

Los nodos se crean mediante las funciones que proporciona ns3, se tiene un *NodeContainer* donde se alojan todos los nodos de la simulación.

Una vez creado el número de nodos que se necesita en el escenario del experimento se procede a la instalación del modelo de movilidad asociado a los nodos, en este caso, al usar trazas provenientes de SUMO es necesario usar el *helper* llamado *Ns2MobilityHelper*, con capacidad para interpretar los ficheros de traza exportados desde SUMO. Desde este momento los nodos del escenario ya tienen un modelo de movilidad asociado. El escenario creado ha de concordar en tiempo de simulación y número de nodos con la traza SUMO generada.

CreateDevices

Teniendo los nodos ya creados, es necesario crear los dispositivos encargados de las comunicaciones, por ello se les instala la capa física y la capa de acceso al medio (*Medium Access Layer - MAC*).

En el experimento, los dispositivos se asociarán mediante enlaces inalámbricos del tipo *adhoc*. Se necesita instalar interfaces acorde a este hecho.

La interfaz física escogida corresponde a *YansWifiPhy*, y en ella hay que jugar con las ganancias de transmisión y recepción, así como con la potencia de transmisión hasta dar con una buena combinación que resulte efectiva para transmitir correctamente a distancias de hasta 200 metros.

El estándar de transmisión 802.11 usado en este caso ha sido el 80211b, se ha optado por este al contar con estudios anteriores que habían realizado sus experimentos utilizando este estándar, así es posible comparar resultados de manera más justa, con escenarios en condiciones muy parecidas.

El modelo de acceso al medio en este caso corresponde al modelado dentro del simulador como *Adhocwifimac* necesario para las conexiones directas entre nodos sin necesidad de una infraestructura subyacente.

La información se transmite con un *datamode* de tipo DsssRate2Mbps, esto habla de la modulación que se usa para emitir en el espectro inalámbrico la información. Esta modulación está especialmente pensada para reducir las interferencias entre los nodos que emiten en el mismo canal[47].

También se define aquí el canal mediante el cual se comunicarán todos los nodos, que será de tipo *WifiChannel* y en el incorporaremos la versión creada del modelo de propagación *log normal shadowing*.

InstallInternetStack

Habiendo definido ya la capa física, el medio de transmisión y la capa de acceso a dicho medio, se puede proceder con la asignación de direcciones IP mediante el uso de los *helpers* proporcionados por el simulador.

Es ahora cuando se instala el *routingHelper* asociado al protocolo de enrutamiento a evaluar. También se ponen a punto parámetros usados a la hora de instalarlo (intervalo entre mensajes de control o *timeout* para las entradas de nodos vecinos, entre otros).

InstallApplications

Como último paso, queda aportar al experimento una aplicación que sea capaz de generar el tráfico a enviar desde cada origen hasta los distintos destinos.

En este caso se opta por el uso de la aplicación de tipo *OnOff* que se activa periódicamente para enviar datos, durante intervalos regulares marcados por el usuario que la configure. La pondremos a funcionar sobre el protocolo de transporte *UDP*, ya que se desea que la información se entregue lo más rápido posible y se evaluará cómo reacciona el protocolo a la hora de transmitir en cuanto a tiempos de envío o a entrega de paquetes.

Si se utilizase *TCP* no se apreciaría la pérdida de paquetes real al tener capacidades de retransmisión a costa de pagar con *overhead* y tiempo de procesamiento añadido.

En el experimento también se evalúa el consumo energético realizado por los nodos, asociado a sus interfaces de red. El modelo energético escogido, existente en el simulador, ha sido *wifiradioenergymodel[43]*.

3.2.5.2 Posibles protocolos a probar

Se ha dotado al *script* con las capacidades necesarias para probar 4 protocolos:

- Aproximación a SRR mediante sistema experto basado en reglas, añadiendo el *SNR* como indicador de calidad del enlace.
- Aproximación a SRR utilizando una red neuronal para seleccionar el mejor vecino en cada caso.
- Dos protocolos típicos de redes MANET como son OLSR y AODV.

3.2.5.3 Trazas de movilidad asociadas al escenario

Dentro del propio escenario de simulación se accede a los ficheros que correspondan a los escenarios de movilidad necesarios para llevar a cabo los experimentos.

La generación de estos escenarios, representados mediante ficheros de trazas que se pasan al simulador ns3, ya ha sido comentada en apartados anteriores.

3.2.5.4 Recolección de datos

El uso de las herramientas de ns3 conocidas como *traceSources* y *traceSinks* resulta muy útil a la hora de recoger datos.

Los resultados del experimento han sido obtenidos mediante la creación de *callbacks* lanzados cuando los eventos asociados a los *traceSources* de envío o recepción de los paquetes de aplicación se iban dando. De esta forma ha sido posible calcular los retardos, *throughput* de aplicación enviado correctamente y cantidad de paquetes transmitidos y recibidos. También se crearon ciertas funciones que aprovechasen estos *traceSources* para conocer la energía media consumida por los nodos de cada experimento.

Con todos estos datos se pueden crear estadísticas indicativas de cómo ha funcionado cada protocolo evaluado en el escenario creado.

Además también es posible crear otro tipo de documentos como por ejemplo capturas de tipo *.pcap* con todos los paquetes enviados y recibidos durante la transmisión, o ficheros de log que pueden resultar útiles para estudiar la solución a un posible fallo manifiesto en el desarrollo.

3.2.5.5 Parámetros regulables

Se otorga cierta flexibilidad al escenario, para sacar más partido al mismo, conseguida mediante los siguientes parámetros para la simulación:

- **RunID:** Parámetro que marcará cuantas veces realizaremos el experimento para el escenario lanzado. Así se puede obtener unos resultados más fiables al calcular la media de los datos obtenidos en cada run, y se sabe con mayor exactitud como se desenvuelve el protocolo.

- **Número de nodos fuente:** Marcará el número de nodos del escenario encargados de emitir datos mediante la aplicación *OnOff*.
    - Se eligen aleatoriamente a cada *run* en la simulación.
- **Número de nodos sumidero:** Marca cuantos nodos utilizan el *Packet sink* de ns3 para recibir los paquetes enviados por los nodos fuentes.
    - Se eligen aleatoriamente entre todos los nodos del escenario.
- **Protocolo a evaluar:** Indica al script de simulación cual de todos los protocolos mencionados en el apartado anterior se ha de usar en el experimento actual.
- ***Datarate* o tasa de envío en la capa de aplicación:** Indica a la aplicación *OnOff* a qué velocidad debe generar paquetes.
- **Número de nodos y velocidad de los mismos**
    - En este caso la combinación de ambos hará que el escenario escoja en tiempo de ejecución el escenario SUMO asociado.
- **Sigma para las experimentaciones con *Log normal shadowing*:** Este parámetro regula como penaliza la distancia a la hora de transmitir la información por el medio inalámbrico. Va asociada al modelo de propagación creado.
- **Duración de la simulación**
- **Semilla de la simulación:** Hay que cambiarla en cada *run*, de esta forma el simulador obtendrá distintos resultados a cada iteración, ya que si todos empiezan con la misma semilla el simulador funcionará todas las veces que se lance el experimento de la misma manera, lanzando los mismos eventos.
- **Alcance inalámbrico de los nodos**
- **Puerto de la aplicación *OnOff***

3.2.6 Aspectos sobre la red neuronal

La librería OpenNN permite crear fácilmente *scripts* de simulación en el lenguaje C++ con clases y métodos que se encargan de recolectar la información necesaria para la configuración y el entrenamiento correcto de una red neuronal.

El desarrollo de esta parte se ha basado en conocer que arquitectura, *loss index* y método de entrenamiento resultan más adecuados para el problema que se tiene enfrente.

En la propia web de la librería se orienta sobre este hecho, por tanto, teniendo en cuenta que el problema que se busca solucionar con la red neuronal es de regresión, se deciden los aspectos más importantes en su desarrollo.

Se prueban distintas combinaciones que den los menores índices de pérdida y quedándonos con el mejor posible. El análisis sobre que combinación funciona mejor se realiza mediante un script de test, al que se le pasa un margen de error máximo para considerar la salida de la red neuronal como válida.

3.2.6.1 Método entrenamiento

Se ha de definir dentro del programa la forma que tienen los *dataset* de entrenamiento que se usa para resolver el problema de optimización.

De esta manera, se carga el fichero de entrenamiento, creado mediante el *script* generador de trazas de entrenamiento. Es necesario indicar que separador se usa entre cada variable, para que el programa sea capaz de reconocer donde empieza y termina cada valor de entrada, así como la salida esperada.

También será necesario utilizar un puntero llamado *Variables*, para proporcionar información referente a la forma que tienen las muestras:

- Nombre de estas variables e identificador
- El número de variables que actúan como *input*
- El número de variables que actúan como output

Es entonces cuando se define la arquitectura de capas que utilizará la red, uno de los aspectos más importantes de la misma. En este caso no se ha sobre-escalado la red, ya que se busca obtener un rendimiento de la misma aceptable, pero no a costa de generar una red muy profunda que acabe consumiendo muchos recursos de procesamiento.

Se probaron varias arquitecturas, siempre con una o dos capas intermedias, y se llegó a la conclusión empírica de que la fórmula que mejor resultados ofrecía era el usar una red formada por dos capas intermedias, que contasen con cuatro y dos perceptrones, respectivamente.

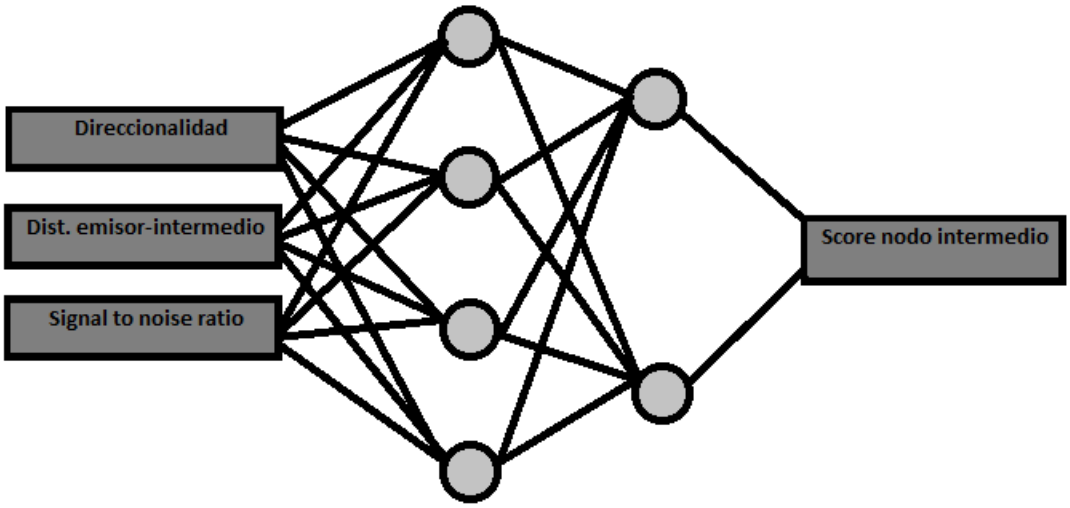


Ilustración 20. Arquitectura de la red neuronal creada

Habiendo definido ya la forma que tendrá la red neuronal, se procede a escoger un índice de pérdida adecuado al problema. Buscando información sobre el tema, se observa que para un problema de este tipo uno de los métodos que suele dar buenos resultados es el llamado *mean squared error*.

Se realizan pruebas de forma empírica probando los distintos índices que nos proporciona el simulador a parte del mencionado (*root mean squared error, sum squared error, normalized squared error*...). Se concluye que efectivamente el índice ya destacado es el que mejores resultados obtiene.

Por último, queda escoger una estrategia de entrenamiento, se escoge una de las más populares por su velocidad y su buen funcionamiento, la estrategia de *Quasi Newton*. Una vez más, se comprueba empíricamente que los resultados obtenidos con esta estrategia superan a los obtenidos con las alternativas ofrecidas en el *framework*.

La combinación de estos elementos básicos le permiten al *script* de creación formar una red neuronal que prediga el *score* correspondiente al nodo intermedio del que se ha pasado la información como parámetro. Se obtiene un error final de 0,0042, aceptable para el propósito del trabajo, y sin haber creado una red que arrasase con los recursos de procesamiento.

3.2.6.2 Script generación de trazas de entrenamiento

Se desarrolla un *script* que genere todas las muestras de entrenamiento necesarias para que la red neuronal aprenda del problema que tiene que resolver.

Este *script* al ser lanzado pide el número de muestras a generar. Al introducirlo, genera variables aleatorias uniformes para cada variable de entrada y dentro de los rangos acotados para cada una.

Con los tres *inputs* (Grado de direccionalidad, distancia emisor-intermedio y *SNR*) generados aleatoriamente, se vuelve a utilizar el criterio que da prioridad a los nodos a distancias medias y bien dirigidos para generar un *score* de salida.

De esta forma se genera un *output* para cada muestra de entrenamiento y crea con los cuatro valores un fichero legible para el programa, con sus líneas representando muestras con los valores separados con comas, como se le ha indicado al *script* de generación de la red neuronal.

3.2.6.3 Script testing de red neuronal

Integra la misma lógica que el *script* de generación de trazas de entrenamiento, con el añadido de que ahora se añade en su interior la red neuronal ya entrenada para poder observar si el error entre el resultado esperado (usando el mismo sistema calculador del *score* que en el generador de trazas de entrenamiento) y el que da la red neuronal está en un margen aceptable para cada muestra.

El número de muestras que estén dentro del rango de error aceptable serán considerados aciertos de la red, así podremos calcular la relación aciertos-muestras que se utiliza como un indicador de la calidad de la red neuronal.

3.2.6.4 Implantación dentro de la clase srar-routingEngine

La propia librería se encarga de generar, una vez terminado el proceso de creación y *test* de la red neuronal, una expresión matemática fácilmente adaptable a cualquier lenguaje de programación que sirve para generar las salidas correspondientes a cada trío de valores de entrada que se le pase.

Esta fórmula matemática se añade a la clase *srar-routingEngine* para crear el método *neuralNetworkCost* que recoge como parámetros de entrada los tres *inputs* necesarios para devolver el *score* del nodo. Así se completaría la integración de la red neuronal en el protocolo.

4. Experimentación

Acabada la implementación, se procede a validar y evaluar los protocolos mediante simulación, analizando los resultados obtenidos. Utilizando el *script* de generación de escenarios de ns3 se crean los distintos entornos a los que va a tener que enfrentarse el protocolo para comprobar su operatividad.

Se van a comparar las dos aproximaciones implementadas en este trabajo con los protocolos clásicos orientados a redes del tipo MANET, como son AODV y OLSR, de esta forma se puede observar las mejoras obtenidas respecto a estos en un entorno más dinámico.

4.1 Diseño de los experimentos a realizar

Con la experimentación se busca evaluar cómo responde el protocolo diseñado a algunas circunstancias que pueden darse en escenarios reales. Se desea observar como algunas variaciones que se pueden dar en el escenario de simulación pueden influir en la calidad de funcionamiento de cada protocolo, medida con parámetros que den una idea de la misma.

En esta experimentación distinguiremos entre cuatro evaluaciones:

- Evaluación del impacto del número de nodos
- Evaluación del impacto de la velocidad de los nodos
- Evaluación del impacto del *datarate* o tasa de envío de los emisores
- Evaluación del impacto del parámetro sigma del modelo de propagación *log normal shadowing*

Cada una de estas evaluaciones ha sido llevada a cabo con los siguientes protocolos:

- SRAR con sistema experto basado en reglas (SRAR)
- SRAR basado en red neuronal (SRARNN)
- AODV
- OLSR

Para obtener unos resultados más fiables, para cada evaluación realizada se han obtenido los resultados correspondiente a la media de las diez veces que se ha tenido que repetir cada uno.

En la tabla siguiente resumo los experimentos llevados a cabo y sus parámetros más significativos.

Exp.	Fuentes|Sumid.	Num.nodos	Datarate (kbps)	$X\sigma$	Velocidad(m/s)
Ch. Shadowing	5|1	150	16	2,8,12	25
Densidad	5|1	30,100,150, 200,250	16	8	25
Velocidad	5|1	150	16	8	15,20,25,30
Datarate	5|1	150	16,24,32,50, 72	8	25

Tabla 4. Datos de los experimentos

Se estudiará el impacto de las variaciones en los siguientes parámetros:

- *Packet Delivery Ratio* de paquetes de datos
- *End to end delay* medio de los paquetes de datos recibidos
- Consumo energético medio de la red por enviar/recibir paquetes de datos y control.

Las simulaciones tendrán una duración de 160 segundos en la mayoría de los casos, siempre se prima la conectividad de la red (que al menos haya una ruta posible desde cada nodo de la red hasta otro), ya que se pretende evaluar el comportamiento del protocolo de forma general, sin tener en cuenta su modo de encolado de paquetes.

La idea es comprobar a que niveles de escalabilidad pueden llegar las implementaciones sin verse muy perjudicadas por los retardos en la transmisión o la pérdida de paquetes.

Algunos escenarios realizan simulaciones de menor tiempo al no poder garantizarse una red conexa durante el tiempo estipulado para las demás. Esto se debe a que, según el número de nodos o la velocidad que lleven los mismos, el escenario puede quedarse corto para los 160 segundos. En el anexo se encuentran detallados todos los escenarios.

La aplicación instalada en los nodos fuente tendrá 30 segundos al inicio de la simulación antes de activarse. De esta forma se le da tiempo al nodo de conocer el entorno en el que se está moviendo y cuáles son sus vecinos.

Del mismo modo, se detendrá el envío de paquetes de aplicación 30 segundos antes de acabar la simulación, asegurando así que los paquetes que puedan haber sufrido un mayor retardo hayan llegado al destino acabada la simulación.

4.2 Creación de escenarios de movilidad

Se utiliza el simulador *SUMO* presentado previamente con la finalidad de crear las trazas de movilidad necesarias para conseguir un escenario parecido al que se utilizó en [11].

La creación de un escenario, como ya se ha comentado anteriormente, requiere de dos ficheros principales:

- Definición del escenario donde se moverán los nodos
- Creación de los nodos y definición de las características del tráfico

Para crear el escenario donde se mueven los vehículos, se utiliza el entorno gráfico proporcionado por *netedit*, el resultado utilizado en la mayoría de casos es el siguiente:

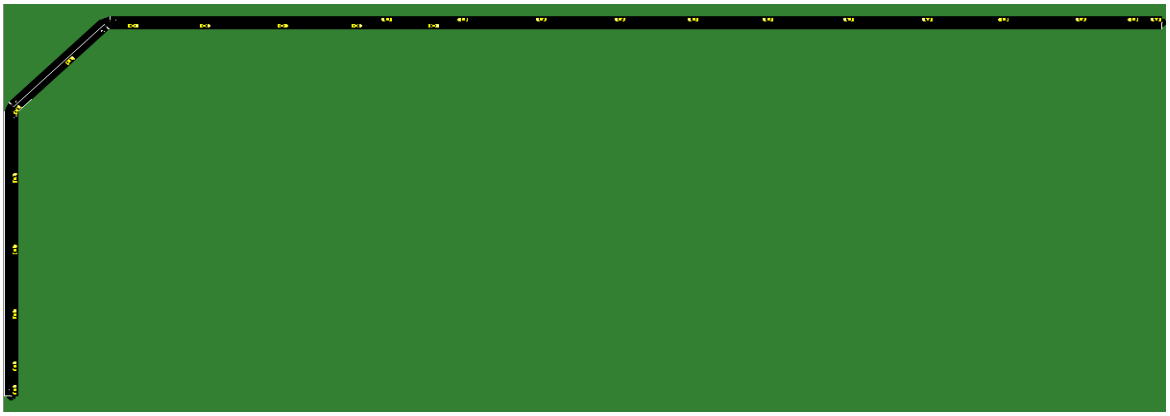


Ilustración 21. Escenario utilizado en el experimento

Se trata de un escenario simple, con curva y carriles en las dos direcciones. Sus dimensiones generalmente serán de 4000 m de ancho por 500 m de alto, que podrán verse modificadas en algunos escenarios al necesitar un mayor espacio para que los nodos no salgan del mismo durante el tiempo que dura la simulación.

Seguidamente se alimenta este escenario con los vehículos representando a los nodos de la simulación, esto se realiza mediante un fichero especial de extensión *rou.xml*.

La necesidad de formar una red conexa provoca que el comportamiento de los nodos a crear resulte muy costoso de modelar a mano; las herramientas que se facilitan, como los llamados *flows*, no aseguran una distancia entre los vehículos menor de los 200 metros de cobertura que se requieren.

Por este hecho, se tuvo que desarrollar un *script* en C++, que siguiendo el formato indicado para construir el fichero de rutas, podrá situar el número de nodos correspondiente al experimento a distancias regulares.

Este *script* preguntará por tres parámetros básicos para la generación del fichero de rutas; como son el número de nodos de la simulación, la velocidad a la que se moverán y la distancia inicial a la que parten unos de otros. De esta forma resultó más sencillo controlar que la red se mantuviese conexa.

Las herramientas que se ofrecen desde el propio simulador *SUMO* no aseguran la distancia inter-vehicular máxima que se necesita, dando lugar a escenarios que no resultaban válidos para el propósito de la experimentación.

La manera que tiene este script para ubicar los nodos es la siguiente, internamente se le han puesto los datos relativos a la longitud de los carriles de nuestro escenario, de forma que conociendo la longitud de todos pueda ubicar los nodos en intervalos de espacio regulares, controlando además que no se añadan nodos en posiciones inválidas (en posiciones que superen la longitud máxima de cada carril).

En el anexo se detallan todos los escenarios de simulación diferentes creados y como se han distribuido los nodos que se le añaden.

4.3 Scripts programados para lanzar todos los posibles experimentos

El proceso de lanzamiento de cada simulación de *ns3* resulta bastante tedioso de realizar, al tener que pasarle por línea de comandos los diferentes parámetros relativos a cada escenario a simular.

Al observarse lo incómodo que resultaba tener que lanzar cada experimento por separado, se genera un *bash script* para cada tipo de experimento a realizar, que lanzará todas las simulaciones necesarias con sus correspondientes parámetros, ahorrando el proceso de tener que lanzarlos a mano.

4.4 Proceso de recolección de datos y representación de los mismos

Recolección

- El propio *script* de simulación cuenta con la lógica necesaria para controlar los envíos y recepciones que ocurren durante el funcionamiento de la aplicación. El uso de los *callbacks* permite conocer el momento en el que se envía y recibe un determinado paquete, de forma que se pueda realizar los registros necesarios para calcular las estadísticas que evalúen la calidad del protocolo de enrutamiento.
- El cálculo del PDR se realiza de manera sencilla. Cada experimento lanzado pondrá dos variables a cero, que serán indicativas del número de paquetes enviados y recibidos por la aplicación. Con estos valores se calcula lo correspondiente a cada experimento.
- El cálculo del *delay* no resulta tan trivial, se crea una estructura dentro del simulador que pueda registrar para cada paquete su instante de envío, de manera que cuando sea recibido se calcule el tiempo transcurrido por paquete. Con esos tiempos se podrá sacar la media global de todos los envíos.
- La energía se ha calculado de manera similar, usando los *callbacks* para el cálculo de la energía. Cada vez que se produce una transmisión o una recepción por un nodo de la red, se captura el evento, aumentando las variables que contienen el consumo de cada nodo. Al acabar se saca la media de todos los consumos.

Representación

- Los datos a representar para cada experimento se encuentran en dos ficheros, uno relativo a la energía y el otro relativo a las métricas de calidad de la comunicación entre los nodos.
- Ambos ficheros deberán ser tratados de forma automática para extraer los datos importantes y representarlos de una forma más visual, para así poder realizar una comparativa entre los distintos protocolos. Además, se puede observar el impacto sobre el funcionamiento del protocolo que tienen los cambios en ciertas variables.
- Se desarrolla un *bash script* encargado de acceder a cada experimento de cada protocolo y hacer *parsing* de los resultados, de manera que los obtenga automáticamente y aproveche el software de *gnuplot* para pasarle estos mismos datos ordenados y que proporcione las representaciones gráficas de los resultados. Los resultados obtenidos muestran de forma comparativa el rendimiento de los distintos

protocolos, con una gráfica distinta para cada tipo de experimento y cada parámetro a estudiar.

4.5 Resultados de la experimentación

Realizadas las simulaciones y obtenidos los resultados de forma comparativa, se procede a evaluar cómo ha operado cada protocolo en los distintos escenarios.

4.5.1 Evaluación respecto al consumo

En estos experimentos se ha calculado el consumo energético que han supuesto las transmisiones y recepciones realizadas por todos los nodos de la red. Con los datos relativos al consumo de cada nodo, se ha representado el consumo medio a nivel de red.

4.5.1.1 Consumo variando datarate

En este apartado se presentan los resultados obtenidos en la simulación relacionada con consumo energético de la red variando el *datarate* o tasa de envío de paquetes de aplicación entre los valores 16 y 72 kbps. En dicha campaña el escenario ha estado compuesto de 150 nodos a 25 m/s, y se ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

Consumo de red

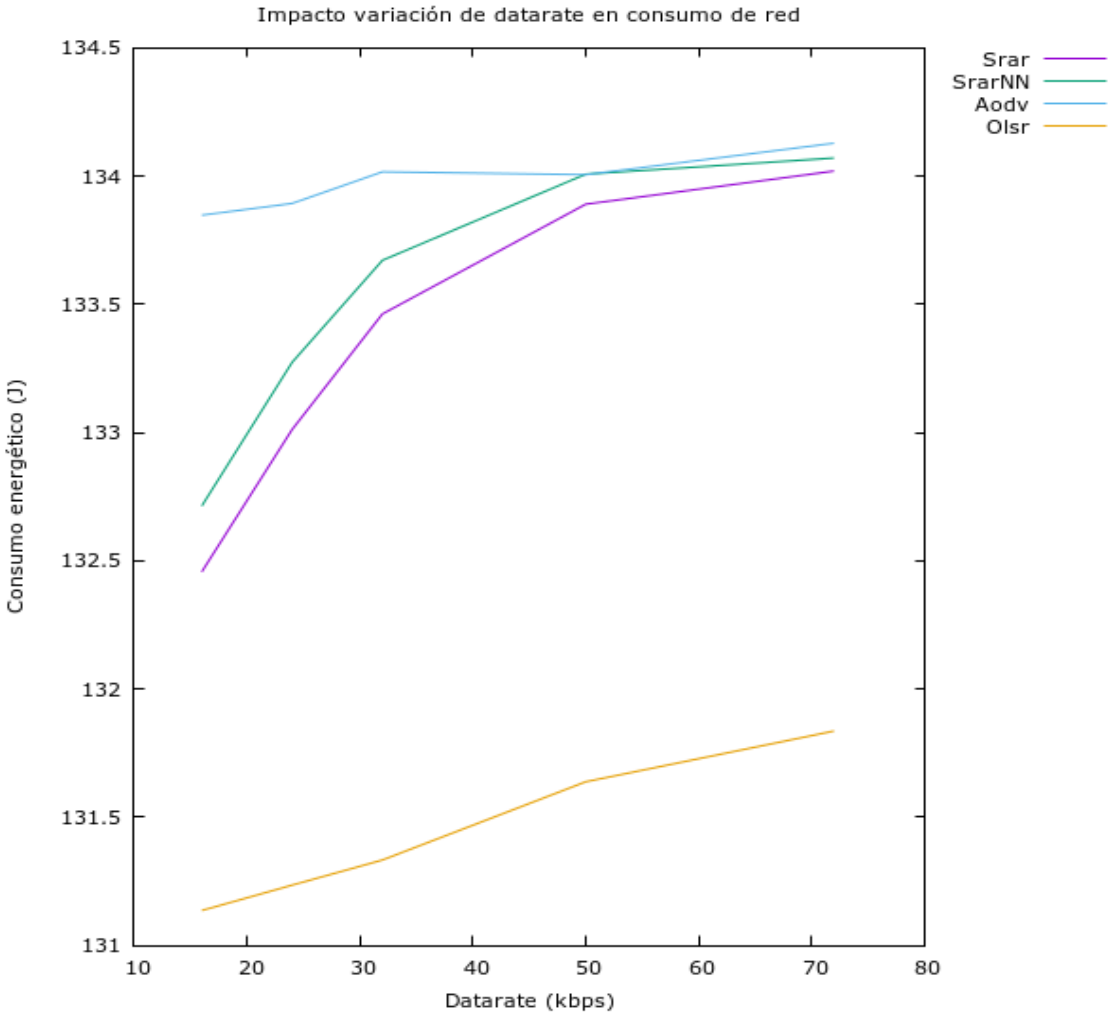


Ilustración 22. Gráfico comparativo sobre el consumo de red variando el *datarate*

Aumentar la tasa de envío provoca que el consumo en todos los protocolos se incremente en mayor o menor medida. SRAR y SRARNN notan mucho más el efecto de la tasa de envío, ya que cada paquete enviado por la red tendrá un mayor recorrido por la misma que si se utilizasen AODV u OLSR. Cada salto en el recorrido supone consumo extra.

AODV obtiene unos consumos ligeramente más elevados, llegando a igualarse en los peores escenarios con los protocolos implementados.

La diferencia, es que el mecanismo de descubrimiento de rutas de AODV inunda mediante *flooding* la red con paquetes RREQ (*Route Request*), hecho que dispara los consumos asociados al envío y recepción de estos. Por tanto, gran parte del consumo de AODV proviene de sus paquetes de control. Los paquetes de datos usando este protocolo no tienen mucho recorrido en una red tan dinámica como las VANET, por tanto el consumo asociado a ellos es menor en el caso de AODV.

Sin embargo, para los protocolos implementados el envío de paquetes de control se reduce a un broadcast con TTL = 1 cada segundo, aunque el periodo sea configurable. Es por ello que el consumo asociado a los paquetes de control de SRAR y SRARNN es menor que el que se asocia a los paquetes de datos. Los consumos llegan a igualarse en los escenarios con mayor tasa de envío, pero como ya se ha comentado, gran parte de los paquetes de SRAR y SRARNN llegan a su destino, habiendo dado un buen número de saltos, y consumiendo energía por cada salto. En el caso de AODV gran parte de los paquetes no llegan al destino por no encontrar una ruta válida.

OLSR está ligeramente por debajo en consumo que AODV, SRAR y SRARNN. Este protocolo obtiene mejor rendimiento que AODV, no inunda tanto la red con paquetes de control. Sus broadcast de paquetes de control tienen también un TTL=1 y aparece la selección de nodos MPR (*Multipoint relay*) como emisores de los broadcast, evitando así broadcasts innecesarios. De esta forma, aunque se necesite ciertos paquetes de control para escogerlos, se reduce el consumo asociado al *flooding* que sí se tiene en AODV. Por ello se sitúa ligeramente por debajo de AODV en cuanto consumos.

En cuanto a consumo debido a paquetes de control, OLSR supera a SRAR y SRARNN, pero en los resultados se obtienen consumos mayores en las propuestas de este trabajo, esto se debe nuevamente al consumo debido a paquetes de datos. En el caso de OLSR gran parte de los paquetes se pierden por el camino. En el caso de SRAR y SRARNN, pocos paquetes se pierden en la ruta al destino, y se entregan después de haber dado un buen número de saltos en la red.

Si comparamos las dos propuestas del trabajo, la aproximación por red neuronal supone consumos levemente superiores, al realizar en algunos casos una elección del mejor vecino ligeramente más imprecisa que acaba penalizando levemente después de todos los saltos hasta el destino. Ello es debido por un lado a que se trata de una primera aproximación que se está refinando y mejorando su entrenamiento, y por otro que deberían contemplarse más parámetros para la toma de decisiones, aspecto en estudio actualmente.

4.5.1.2 Consumo variando parámetro del modelo de propagación

En el presente apartado se presentan los resultados obtenidos en las simulaciones relativas al consumo de red variando el parámetro *sigma* del modelo de propagación con valores entre 2 y 12. En dicha campaña el escenario ha estado compuesto de 150 nodos a 25 m/s y el *datarate* o tasa de envío de paquetes de aplicación ha sido de 16 kbps. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

Consumo de red

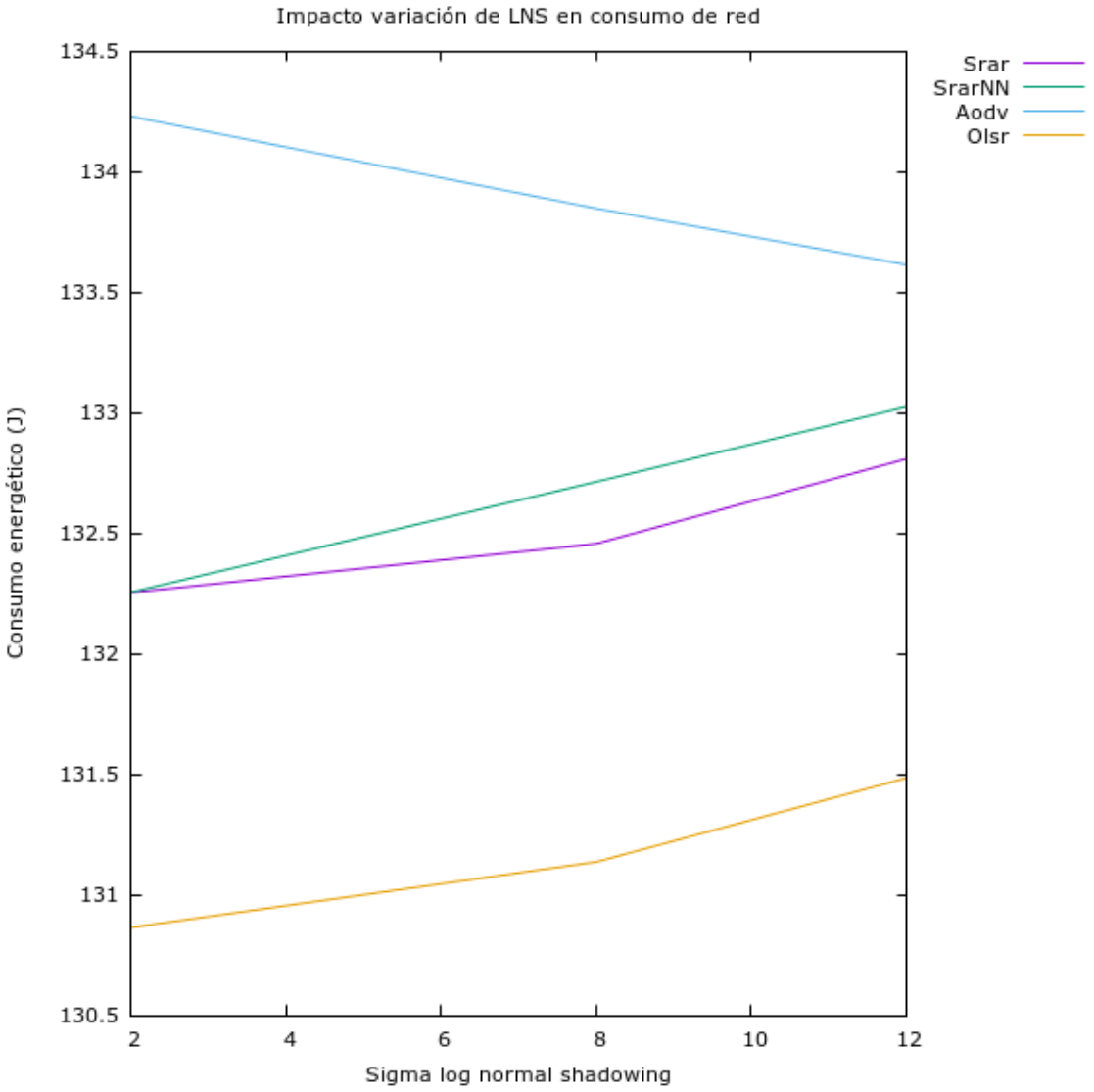


Ilustración 23. Gráfico comparativo del consumo de red variando sigma del modelo de propagación

Aumentar el parámetro del modelo de propagación provoca que cada nodo observe menos vecinos a su alrededor, al verse más limitado su radio inalámbrico.

Se observa que la tendencia en AODV es a disminuir el consumo energético a medida que empeoramos el canal de transmisión. Tener un menor número de vecinos por nodo implica realizar un menor número de *rebroadcast* asociados al *flooding*, lo cual supone un menor número de paquetes broadcast recorriendo la red, hecho que acaba por reducir el consumo asociado a sus paquetes de control, a costa de perder paquetes.

En el caso de SRAR y SRARNN, se observa que la tendencia es a incrementar los consumos. Limitar el radio inalámbrico supone que cada nodo enviará sus paquetes a nodos a menor distancia. De esta forma se necesitará un mayor número de saltos hasta llegar al destino a medida que empeoramos el medio de propagación, favoreciendo el aumento de consumo.

Para OLSR, la explicación es similar, se ve limitado el radio inalámbrico, favoreciendo un mayor número de saltos y por tanto un aumento en el consumo energético.

4.5.1.3 Consumo variando número de nodos

En este apartado se exponen los resultados obtenidos en las simulaciones relacionadas con el consumo de red variando el número de nodos entre 30 y 250. En dicha campaña el escenario ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación y el *datarate* o tasa de envío de paquetes de aplicación ha sido de 16 kbps. La velocidad de los nodos ha sido 25 m/s. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

Consumo de red

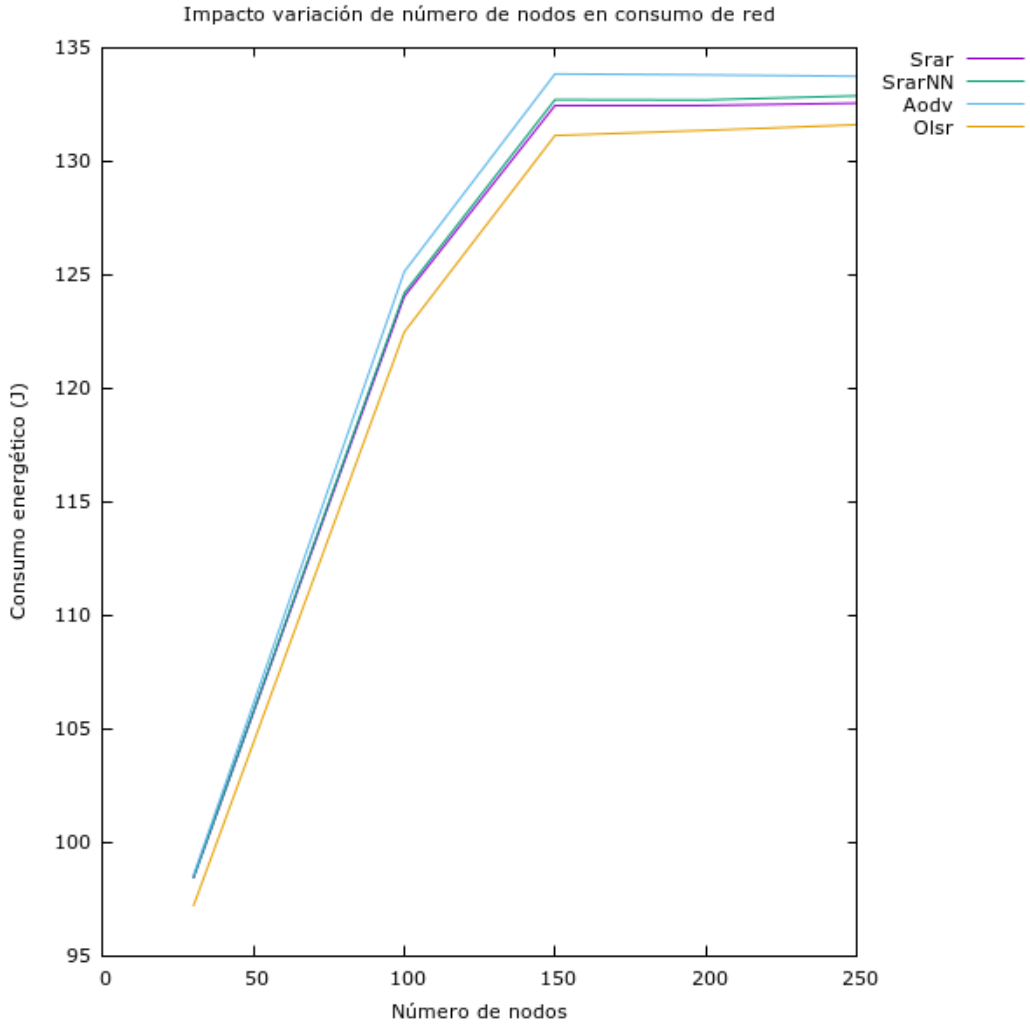


Ilustración 24. Gráfico comparativo del consumo de red variando el número de nodos

Se observa que a medida que se añaden nuevos nodos al escenario, la energía que se consume por nodo aumenta.

Dado un nodo, se tiene un radio de alcance inalámbrico de una cantidad determinada de metros. Dentro de ese espacio, cabrán como máximo una cantidad determinada de vehículos circulando en la carretera y guardando ciertas distancias de seguridad que sean visibles por el nodo.

Llega un punto en el que por limitaciones de espacio, un nodo no puede tener más vecinos a su alcance, y el consumo asociado al procesamiento de paquetes de control se verá reducido.

Por tanto, aumentar el número de nodos puede suponer dos cosas:

1. Aumentar la densidad de vecinos: Tener un mayor número de vecinos supone hacerse cargo de un mayor número de paquetes de control, algunos re-enviables, aumentando aún más, si cabe, el consumo.
2. Aumentar nodos en el escenario, pero no la densidad de vecinos: Desde el punto de vista de cada nodo, los consumos asociados a paquetería de control no se verían tan perjudicados como en el primer caso, siempre y cuando los nodos añadidos no inyecten tráfico de datos a la red.

Sobre los resultados de esta experimentación, hay que tener en cuenta que para 30 nodos, la simulación ha durado 40 segundos menos que el resto y para el caso de 100 nodos, ha durado 10 segundos menos que el resto. Además, estos dos escenarios tenían una menor densidad de vecinos. Si hacemos la media de consumo por segundo en cada escenario, se obtendrían resultados mucho más ajustados, también con tendencia a aumentar consumos a medida que se aumenta el número de nodos.

Orden de los escenarios por densidad de vecinos: 30 nodos (100 m) < 100 nodos (30 m) < 150 nodos (20 m) < 200 nodos (10 m) = 250 nodos (10 m).

Como ya se ha comentado en otros casos, AODV debido a su cantidad de paquetes de control se sitúa con los peores consumos. SRAR y SRARNN están en posición intermedia, los paquetes que han llevado a esos consumos son tanto de control como de datos. OLSR obtiene los consumos más bajos, pero principalmente asociados a paquetes de control, obteniendo también una tasa de entrega muy baja.

4.5.1.4 Consumo variando velocidad

En este apartado se va a explicar los resultado correspondientes a las simulaciones relacionadas con el consumo de red variando la velocidad de los nodos entre los 15 y los 30 metros por segundo. En dicha campaña el escenario ha contado con 150 nodos, ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación y el *datarate* o tasa de envío de paquetes de aplicación ha sido de 16 kbps. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

Consumo de red

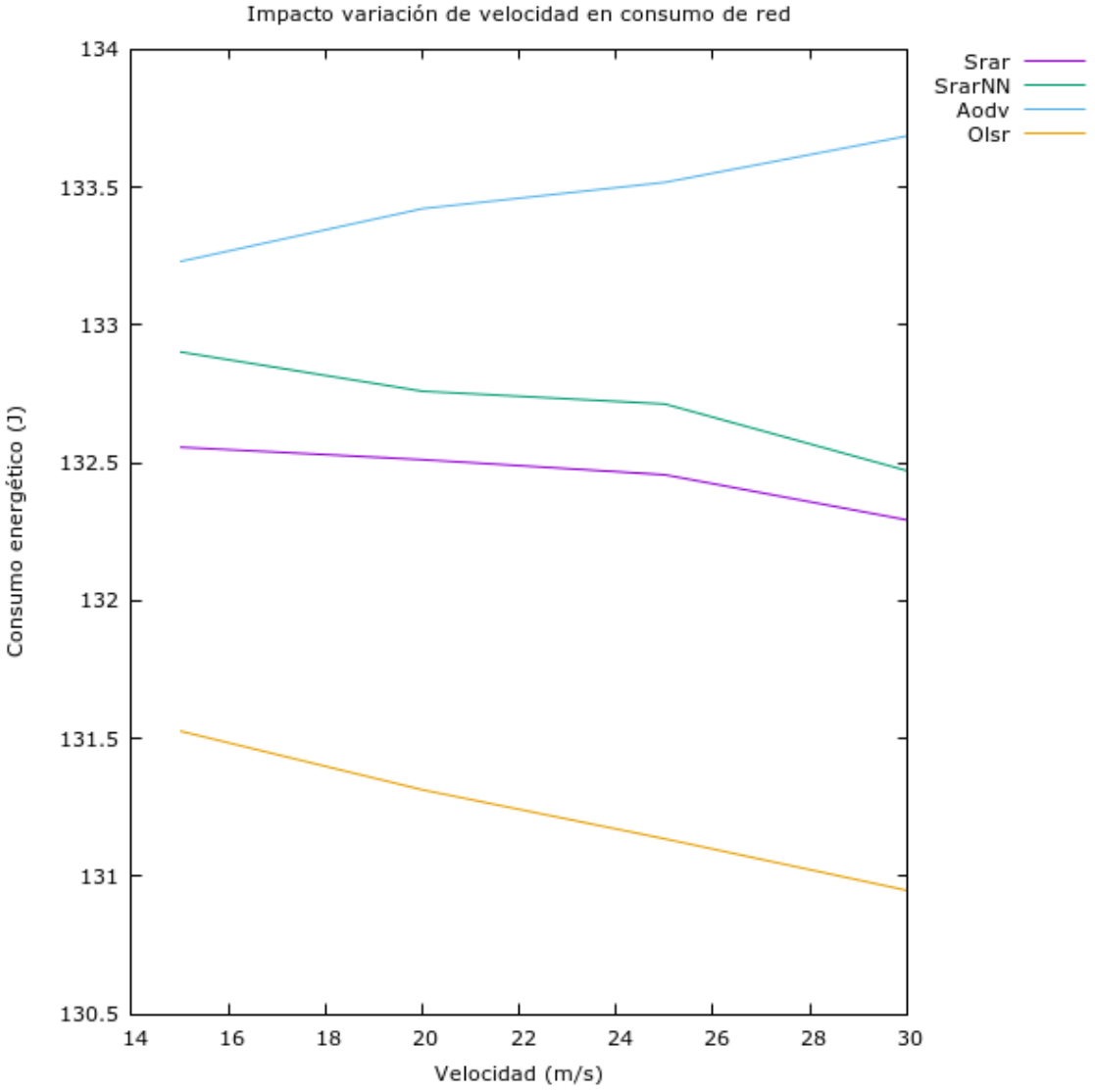


Ilustración 25. Gráfico comparativo del consumo de red variando la velocidad

Si analizamos los resultados de manera comparativa se observa lo mismo que en los experimentos anteriores. Los consumos más elevados se asocian con AODV y son principalmente fruto del trato de sus paquetes de control.

SRAR y SRARNN obtienen resultados ligeramente más reducidos, pero asociados principalmente al reenvío de paquetes de datos correctamente a destinos más lejanos que el resto de protocolos.

OLSR obtiene consumos ligeramente más reducidos que las propuestas de este trabajo, también asociados principalmente a paquetes de control, ya que, al igual que AODV, se obtiene un rendimiento muy bajo y sus paquetes de datos no tienen mucho recorrido en la red.

Mirando la tendencia del consumo al variar la velocidad, se puede observar como SRAR, SRARNN y OLSR tienden a reducir muy levemente sus consumos. Mayores dinamismos implican mayores dificultades para transmitir paquetes de datos. Acaban perdiéndose un mayor número de paquetes, reduciendo consumos.

En el caso de AODV, observamos que su consumo aumenta a medida que se eleva la velocidad. El dinamismo asociado al aumento de velocidad de los nodos perjudica a AODV, que tiene que formar rutas con mayor frecuencia y su mecanismo de mantenimiento de rutas acaba penalizándole. Se emiten un mayor número de paquetes de control que aumentan el consumo total.

4.5.2 Evaluación respecto al retardo extremo a extremo

Ahora se procede a estudiar cómo ha afectado la variación de los parámetros de simulación al tiempo que tardan en llegar los paquetes que son recibidos en su destino.

4.5.2.1 End-to-end delay variando datarate

En este apartado se va a explicar los resultados obtenidos en las simulaciones relacionadas con el *end-to-end delay* variando el *datarate* o tasa de envío de paquetes de aplicación entre los 16 y los 72 kbps. En dicha campaña el escenario ha contado con 150 nodos, ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación y la velocidad de los nodos ha sido de 25 m/s. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

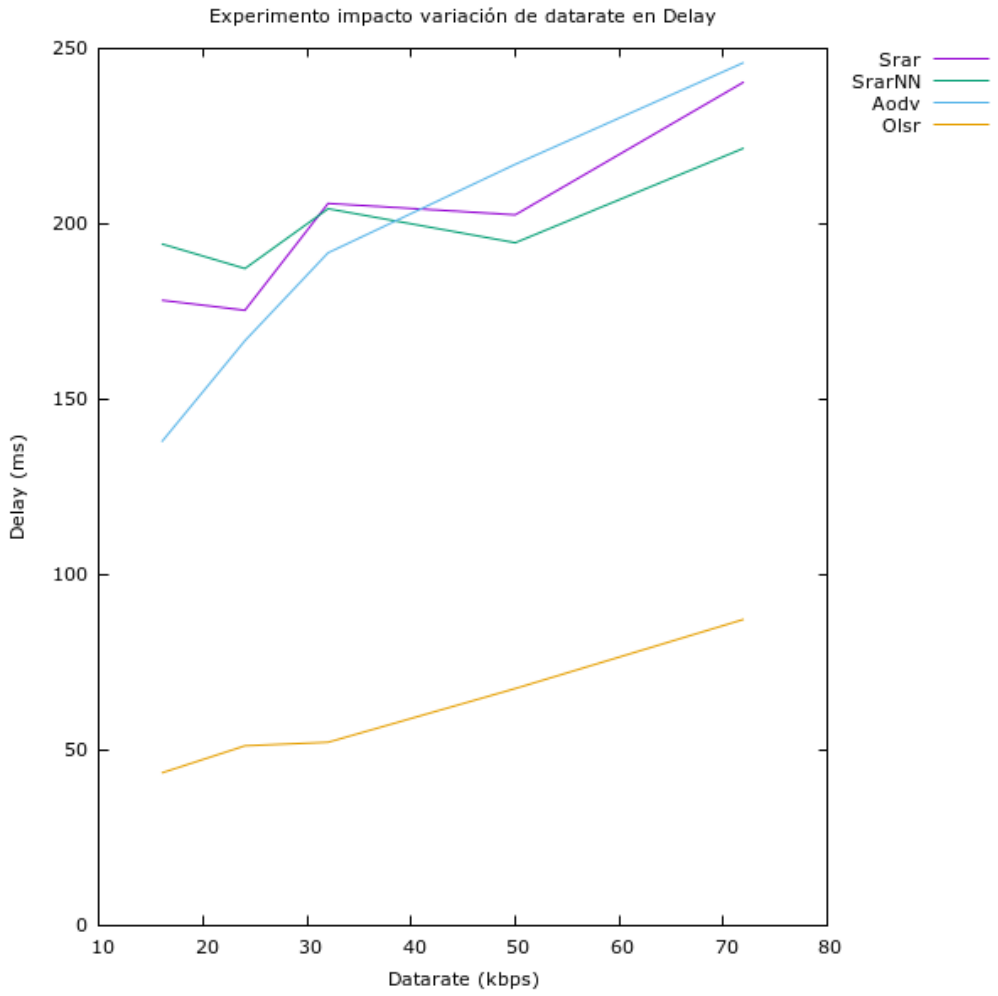


Ilustración 26. Gráfico comparativo del *delay* variando *datarate*

A medida que se aumenta la tasa de envío de paquetes, se añade más carga a la red, lo que supone un aumento en el número de colisiones e interferencias, fenómenos que penalizan al retardo a la hora de enviar datos. Por esto, la tendencia en todos los casos es a aumentar el retardo a medida que se aumenta la tasa de envío.

Si se comparan los protocolos respecto al *end to end delay* se puede ver como las propuestas estudiadas han obtenido resultados similares a los obtenidos por AODV. No obstante, el rendimiento de SRAR y SRARNN es superior, al conseguir con retardos similares entregar una mayor cantidad de paquetes.

El aumento del *delay* en el caso de AODV, proviene del problema que supone hacer *flooding* en una red del tipo *ad-hoc*. Se fomentan interferencias y colisiones que acaban por penalizar la calidad del enlace. A la hora de entregar paquetes, el tener un canal saturado propicia mayores retardos. Por ello, a pesar de haber entregado paquetes de datos a pocos saltos, se obtienen retardos del mismo orden que SRAR y SRARNN, cuyos paquetes sí que han tenido un mayor recorrido por la red.

OLSR no sobrecarga tanto el canal inalámbrico con paquetes de control, hecho que supone que sus enlaces no tengan tanta penalización en términos de retardo al enviar paquetes. No obstante, su rendimiento sigue siendo peor que el obtenido con SRAR y SRARNN, ya que OLSR entrega menos datos y los entrega a distancias más cortas.

4.5.2.2 End-to-end delay variando efecto log normal shadowing

En esta sección se presentan los resultados obtenidos en las simulaciones relacionadas con el *end-to-end delay* variando el *sigma* del modelo de propagación entre 2 y 12. En dicha campaña el escenario ha contado con 150 nodos, ha empleado un *datarate* o tasa de envío de paquetes de aplicación de 16 kbps y la velocidad de los nodos ha sido de 25 m/s. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

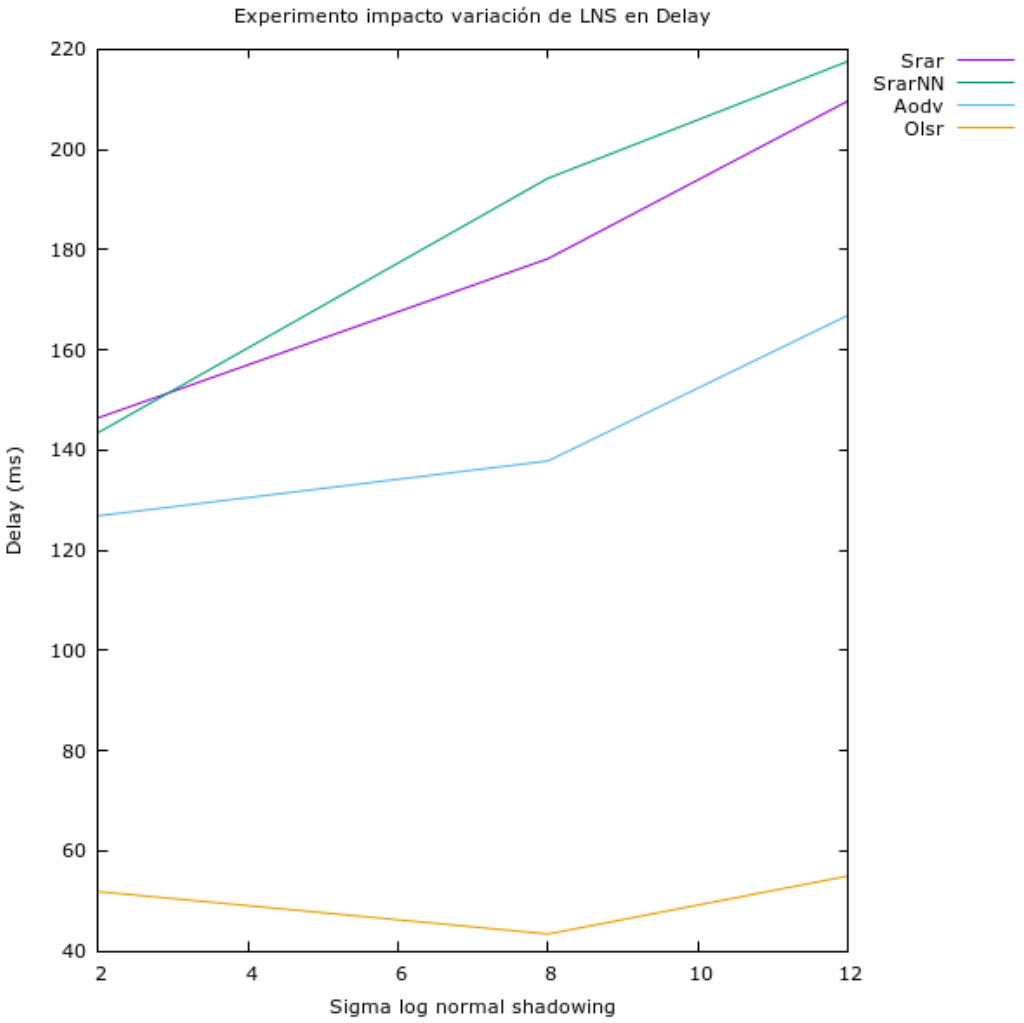


Ilustración 27. Gráfico comparativo del delay variando sigma del modelo de propagación

Como se puede ver en la gráfica, el impacto del medio de transmisión sobre el retardo es directo.

Peores medios de transmisión, donde la señal tiene más dificultades para propagarse, propician que los enlaces formados por los nodos de la red cada vez sean menos robustos y den un mayor número de colisiones y un aumento en el tiempo de propagación de la señal que acabe penalizando el retardo final en la entrega de paquetes.

En este caso, SRAR y SRARNN han presentado retardos más elevados, no obstante su rendimiento en comparación es mucho mayor a AODV y OLSR. Se entregan un mayor número de paquetes y a mayores distancias en cuanto a número de saltos, cosa que aumenta en cierta forma el retardo global, pero obteniendo a cambio un incremento en la fiabilidad.

Las dos propuestas estudiadas han obtenido retardos similares, un poco superiores en el caso de la red neuronal, ya que un pequeño porcentaje de las decisiones que toma son ligeramente imprecisas, favoreciendo un aumento reducido en los retardos de ruta.

En el caso de AODV los retardos vienen favorecidos por técnicas ya comentadas y poco deseables como el *flooding*. Sus paquetes sufren de un retardo por salto más elevado que el resto de protocolos.

OLSR obtiene buenos retardos, pero una vez más a costa de sacrificar significativamente el rendimiento. No entrega un buen número de paquetes, y los entregados están a un número reducido de saltos de red.

4.5.2.3 End-to-end delay variando número de nodos

En el presente apartado se presentan los resultados obtenidos en las simulaciones relacionadas con el *end-to-end delay* variando el número de nodos del escenario entre 30 y 250. En dicha campaña el escenario ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación, un *datarate* o tasa de envío de paquetes de aplicación de 16 kbps y la velocidad de los nodos ha sido de 25 m/s. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

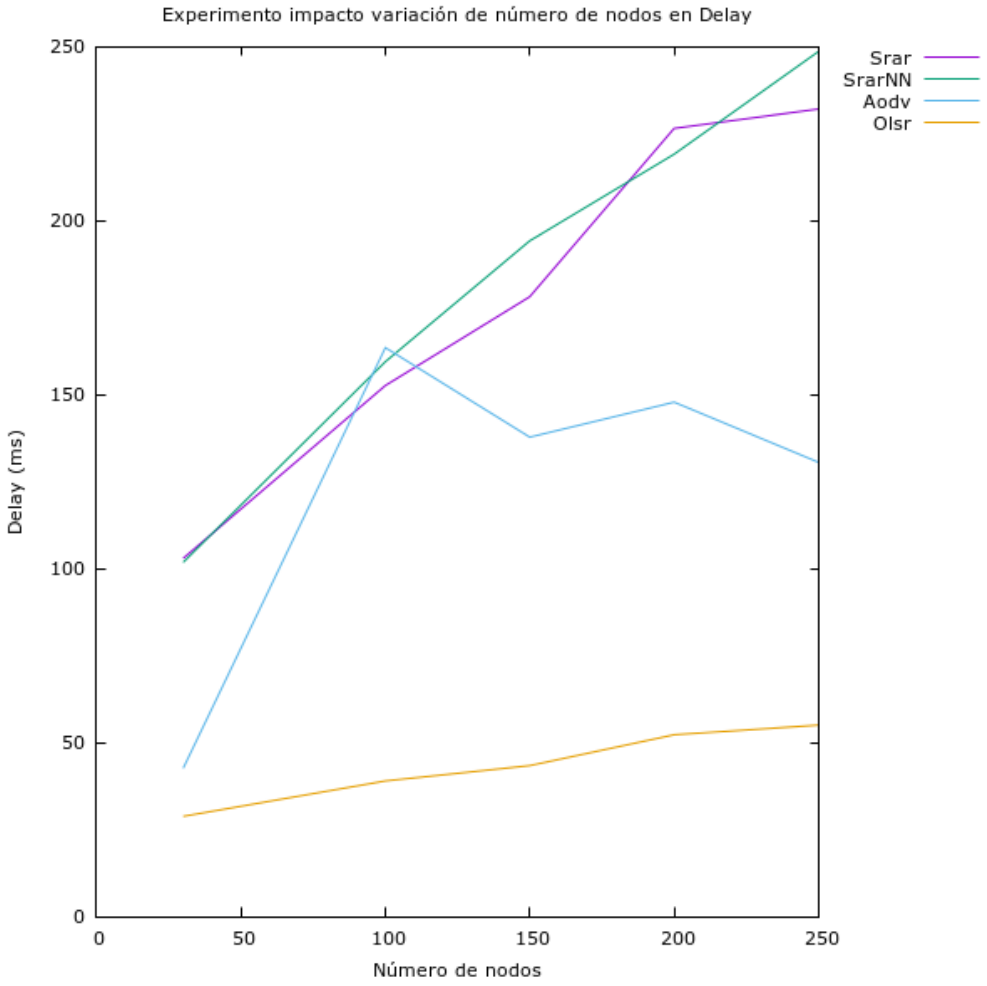


Ilustración 28. Gráfico comparativo del *delay* variando número de nodos

Las propuestas estudiadas, con resultados muy similares, obtienen de nuevo retardos superiores. Estos son nuevamente consecuencia de la acumulación de un mayor número de saltos en los paquetes entregados, en rutas más fiables, pero que suponen un cierto aumento en el retardo global.

En un punto intermedio se encuentra AODV, con retardos asociados principalmente al colapso del canal que se sufre como consecuencia del *flooding* que usa. Llega un punto en el que satura tanto la red que únicamente entrega paquetes a nodos que se encuentran a 1 o 2 saltos. Esto explica la subida inicial, cuando puede enviar a distancias un poco más largas. La bajada repentina corresponde al momento en el que por limitaciones de la red, no puede formar rutas largas y solo envía a nodos cercanos, donde el retardo no puede acumularse tanto.

En general, la tendencia es a aumentar el retardo a medida que se aumenta la densidad de nodos del escenario. Aumentar el número de nodos de la red eleva la utilización de la misma y el número de vecinos que tiene cada nodo. Por tanto, se fuerza a cada nodo a procesar una mayor cantidad de información proveniente de nodos vecinos.

Esto provoca que cada nodo esté más tiempo 'ocioso' recibiendo paquetes y favoreciendo el aumento del retardo. Además, el mayor número de nodos propicia un entorno inalámbrico de menor calidad, con mayor número de colisiones e interferencias, resultando en el aumento de los retardos finales.

OLSR nuevamente obtiene los retardos más reducidos, fruto de su bajo rendimiento al enviar paquetes únicamente a distancias cortas.

4.5.2.4 End-to-end delay variando velocidad

A continuación se presentan los resultados obtenidos en las simulaciones relacionadas con el *end-to-end delay* variando la velocidad de los nodos entre los 15 y los 30 metros por segundo. En dicha campaña el escenario ha contado con 150 nodos, ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación y el *datarate* o tasa de envío de paquetes de aplicación ha sido de 16 kbps. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

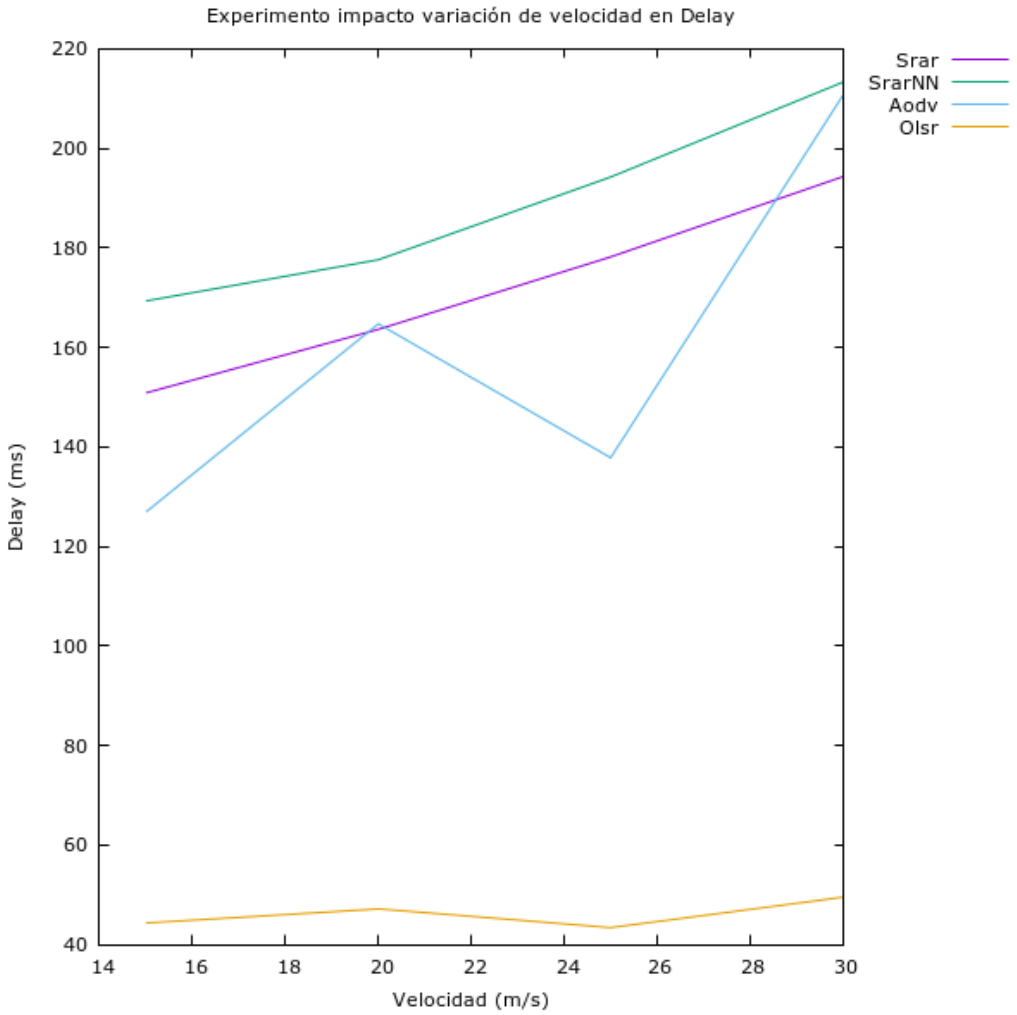


Ilustración 29. Gráfico comparativo del delay variando la velocidad

En general la velocidad de los nodos impacta negativamente sobre el retardo, aumentándolo a medida que se aumenta la velocidad.

Mayores velocidades propician un mayor dinamismo en la red, más conexiones y desconexiones que influyen negativamente en la validez de la información que cada nodo tiene sobre sus vecinos. La elección de a que vecino enviar el paquete se hace de forma más imprecisa, penalizando así el retardo en los protocolos usados.

AODV cuenta con técnicas para hacer frente a la movilidad, como técnicas de mantenimiento de rutas o varios intentos de re-entrega. El problema es que esos métodos están asociados a la emisión de un mayor número de paquetes de control y no tienen en cuenta escenarios donde puede no resultar interesante el mantenimiento de

rutas, ya que el dinamismo de las VANET provoca cambios muy bruscos en estas en cortos periodos de tiempo. Por tanto acaban penalizando en términos de retardos y entrega, disminuyendo su rendimiento.

SRAR y SRARNN obtienen retardos similares a AODV, pero una vez más lo superan en rendimiento y fiabilidad, entregan más paquetes de datos y a mayores distancias de red.

OLSR obtiene los retardos más reducidos, pero sigue manteniendo un bajo rendimiento en comparación a las dos propuestas, ya que obtiene bajos retardos a costa de enviar paquetes correctamente únicamente a distancias cortas, sin posibilidad de penalizar mucho el retardo.

4.5.3 Evaluación respecto al *packet delivery ratio*

4.5.3.1 Packet delivery ratio variando datarate

En el presente apartado se presentarán y analizarán los resultados obtenidos en las simulaciones relacionadas con el porcentaje de paquetes que se entregan correctamente variando el *datarate* o tasa de envío de paquetes de aplicación entre los 16 y los 72 kbps. En dicha campaña el escenario ha contado con 150 nodos, ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación y la velocidad de los nodos ha sido de 25 m/s. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

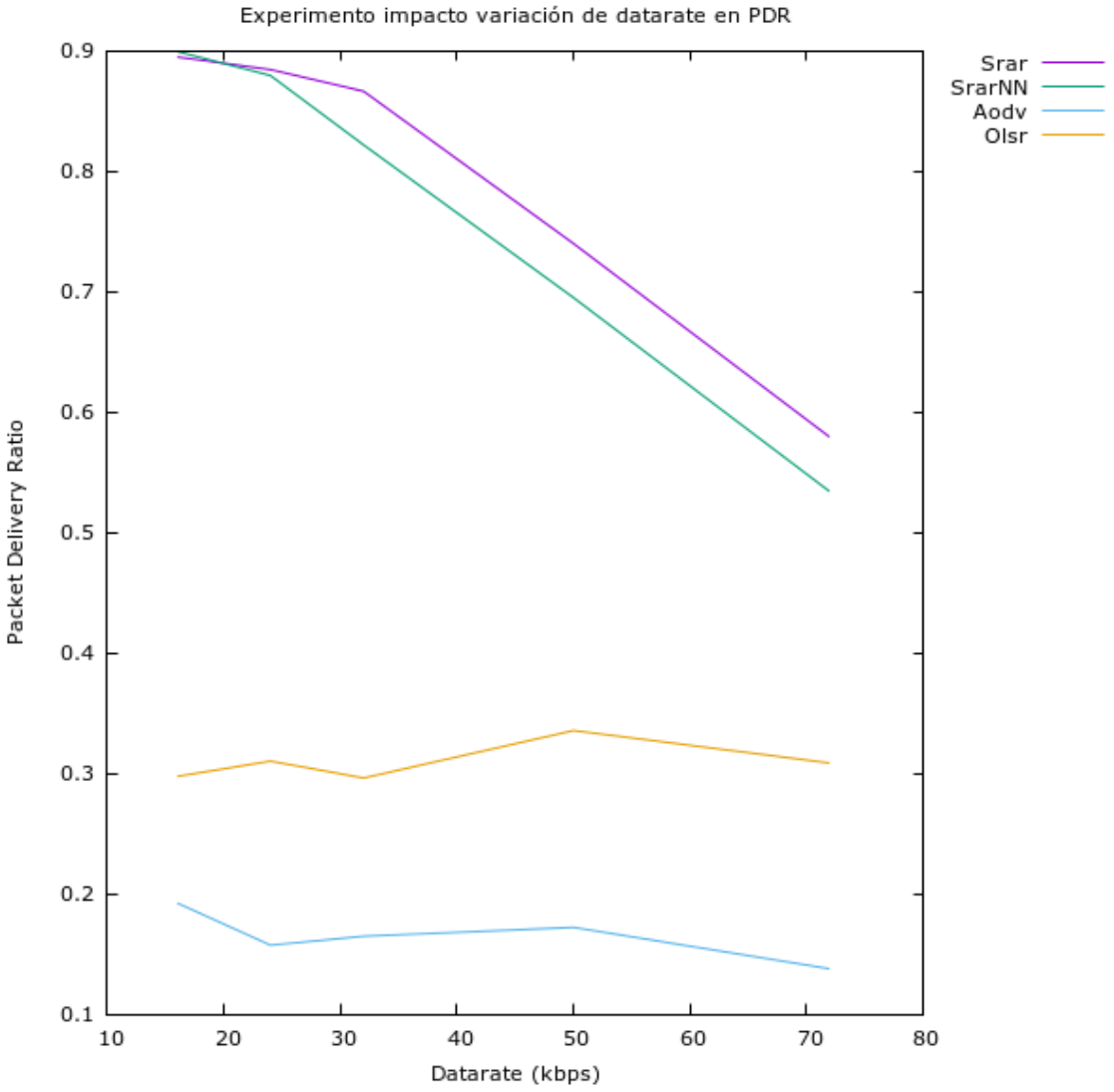


Ilustración 30. Gráfico comparativo del *packet delivery ratio* variando *datarate*

A medida que aumenta el *datarate* en los nodos emisores, la tasa de entrega correcta de paquetes en la capa de aplicación se ve afectada.

Incrementar el número de paquetes de los que se hace cargo la red fomenta las interferencias entre los nodos, así como las colisiones que sufren los paquetes a la hora de ser emitidos.

Los mejores resultados han sido obtenidos por SRAR y SRARNN, siendo este último el que se ha mantenido muy ligeramente por debajo.

Las peores tasas de entrega son las obtenidas por AODV y OLSR, aunque este fenómeno no tiene un impacto tan fuerte sobre ellos, ya que sus rendimientos se mantienen muy bajos en cualquiera de los casos

Además, los paquetes recibidos al usar estos protocolos tienen un recorrido muy corto (1 o 2 saltos hasta que se pierden) en comparación con SRAR y SRARNN.

4.5.3.2 Packet delivery ratio variando parámetro modelo de propagación

En este apartado se explican los resultados obtenidos en las simulaciones relacionadas con el porcentaje de paquetes que se entregan correctamente variando el *sigma* del modelo de propagación entre 2 y 12. En dicha campaña el escenario ha contado con 150 nodos, la velocidad de los nodos ha sido de 25 m/s y el *datarate* o tasa de envío utilizada ha sido 16kbps. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

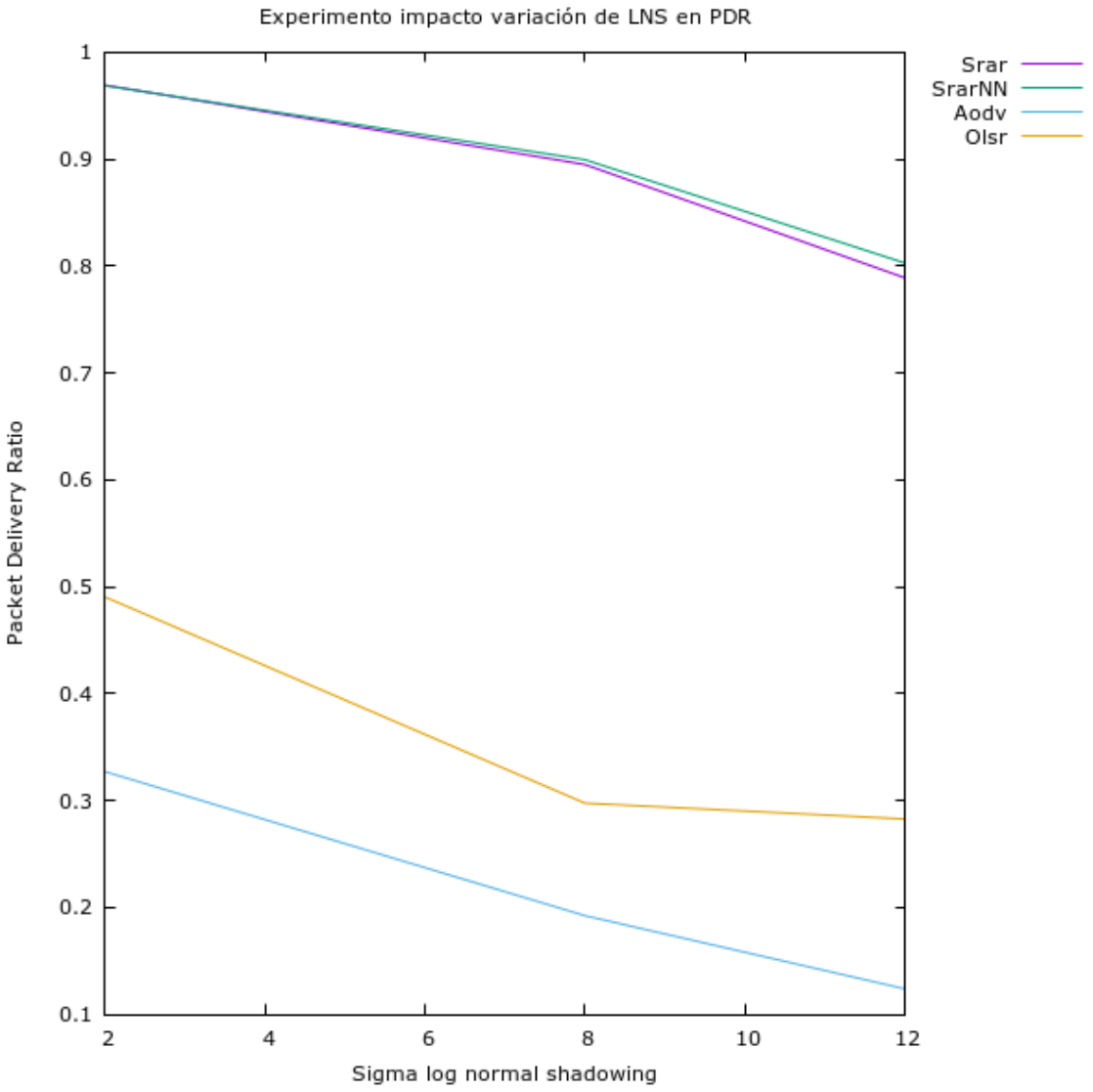


Ilustración 31. Gráfico comparativo del packet delivery ratio variando sigma del modelo de propagación

A medida que aumenta el valor de sigma (indicando al simulador que se tiene mayores dificultades para transmitir datos por el canal) se ve como disminuye el PDR efectivo por la pérdida de paquetes en el medio inalámbrico para todos los protocolos usados. Aparece el efecto de atenuación de la señal y se favorecen las interferencias y colisiones al emitir a distancias más cortas.

SRAR y SRARNN se ven más afectados por los cambios en el modelo de propagación. Los otros dos protocolos ya presentaban un rendimiento muy significativamente inferior.

Como se observa, la calidad del entorno inalámbrico también juega un papel importante en el desempeño de los protocolos.

AODV sufre las consecuencias de utilizar *flooding* en un entorno inalámbrico complicado y acaba obteniendo tasas de entrega ínfimas en comparación con las propuestas realizadas en el presente trabajo que muestran una robustez significativamente superior.

4.5.3.3 Packet delivery ratio variando número de nodos

En el presente apartado se presentan los resultados obtenidos en las simulaciones relacionadas con el porcentaje de paquetes que se entregan correctamente variando el número de nodos del escenario entre 30 y 250. En dicha campaña el escenario ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación, la velocidad de los nodos ha sido de 25 m/s y el *datarate* o tasa de envío ha sido de 16 kbps. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

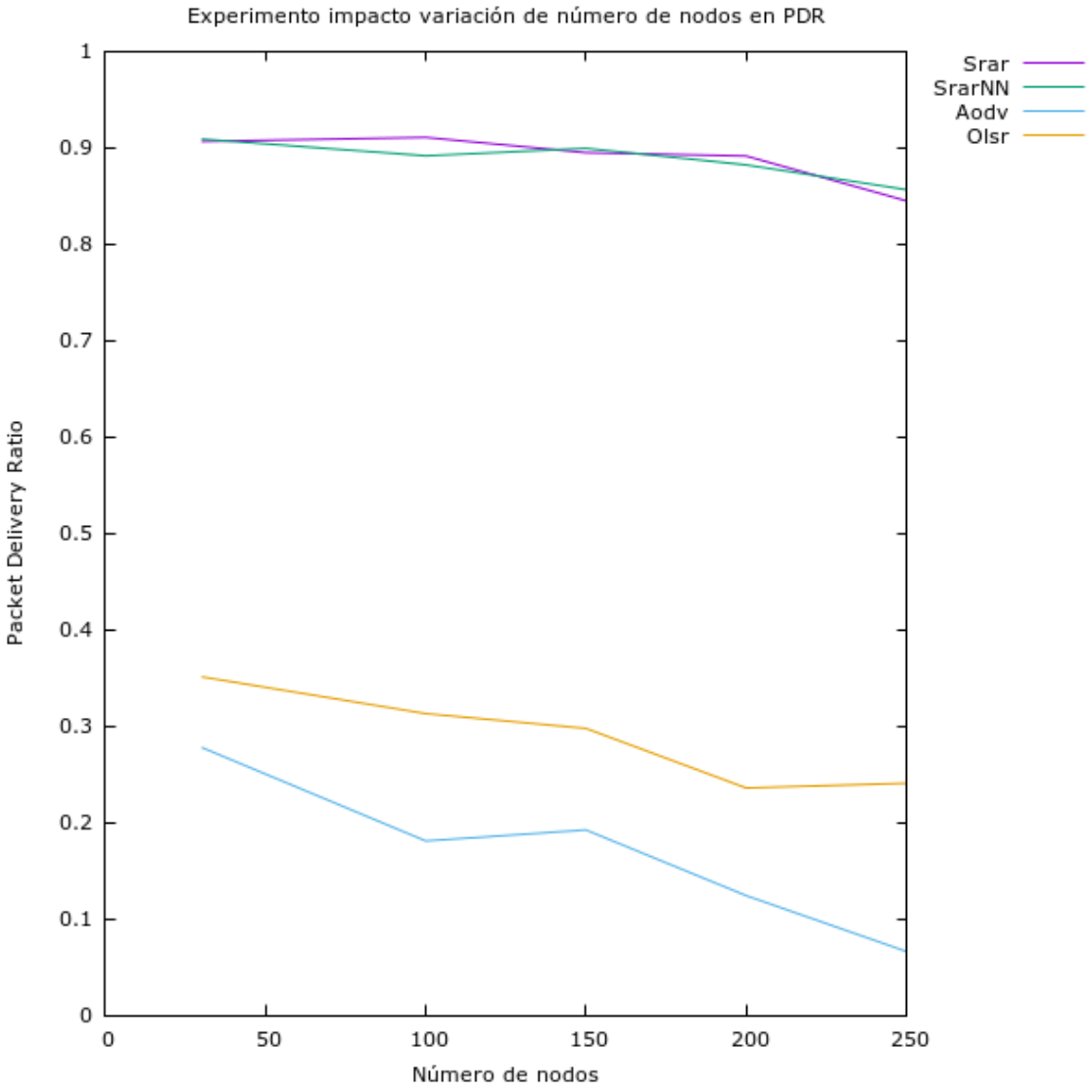


Ilustración 32. Gráfico comparativo del *packet delivery ratio* variando el número de nodos

Añadir un mayor número de nodos a la red usando SRAR o SRARNN, sin aumentar el número de nodos que actúan como emisor no provoca que se añada demasiada sobrecarga a la red.

Esto es así porque haber optado por el uso de un único paquete de control, enviado de forma periódica e incondicional cada segundo, permite escalar correctamente el protocolo a un mayor número de nodos sin sufrir problemas relacionados con el *overhead* relacionado con la trata de paquetes de control de los nodos vecinos.

Algo distinto hubiese sido que todos esos nodos añadidos actuasen como nodos fuente, inyectando tráfico en la red, ahí sí que podría haber sufrido los efectos negativos en el rendimiento.

No obstante, debido al número de saltos que pueden haberse dado con el protocolo, aumentar el número de nodos puede provocar que el retardo acumulado en la ruta para los paquetes de los protocolos implementados se haya visto ligeramente perjudicado.

Esto es debido a que cada nodo tendrá más vecinos a su alrededor, y por tanto tendrá que procesar más información al recibir datos de más vecinos. También tendrá que elegirse el mejor vecino entre un conjunto más amplio de nodos y la posibilidad de interferencias será mayor al tener un mayor número de nodos emitiendo paquetes de control o enrutando paquetes de la capa de aplicación.

En el caso de AODV y OLSR se obtienen resultados muy reducidos desde el primer momento, al no responder bien ante entornos tan dinámicos. El impacto negativo del aumento de número de nodos es mucho mayor en AODV, por ser el único protocolo que hace uso de *flooding*.

4.5.3.4 Packet delivery ratio variando velocidad

A continuación se presentan los resultados obtenidos en las simulaciones relacionadas con el porcentaje de paquetes que se entregan correctamente variando la velocidad de los nodos entre los 15 y los 30 metros por segundo. En dicha campaña el escenario ha contado con 150 nodos, ha empleado un sigma de 8 como valor de la desviación típica para el modelo de propagación y el *datarate* o tasa de envío ha sido de 16 kbps. 5 Nodos han emitido paquetes de aplicación y sólo ha habido un nodo sumidero.

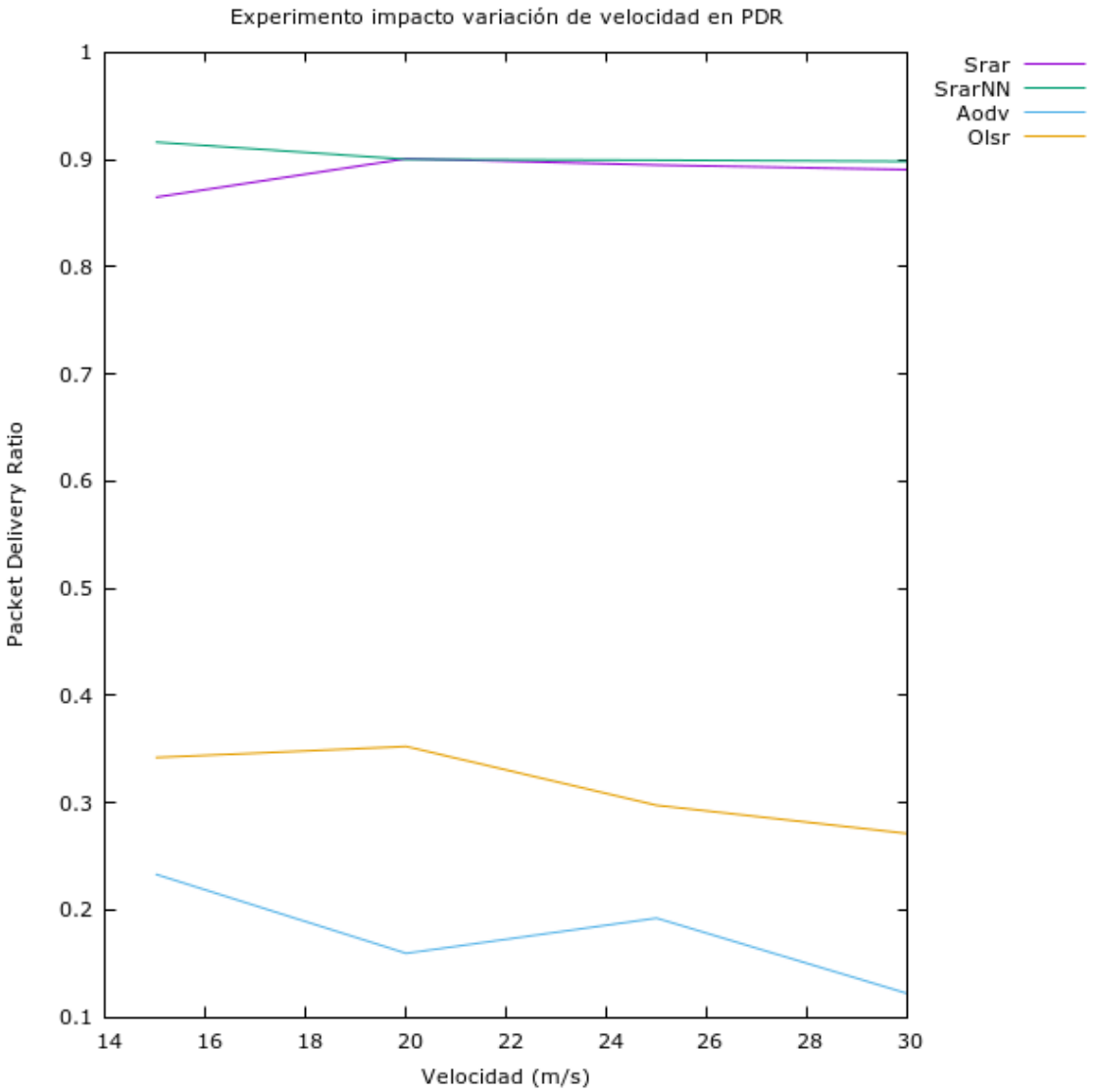


Ilustración 33. Gráfico comparativo del packet delivery ratio variando la velocidad

Mayores velocidades implican un mayor dinamismo en la red. Esto provoca elecciones de mejor vecino más imprecisas que favorecen el aumento del retardo, aunque el número de paquetes correctos entregados no se vea tan afectado.

El haber mantenido una red conexa todo el tiempo favorece que el PDR no se vea tan penalizado. Los paquetes tienen siempre una ruta posible desde origen hasta destino; aumentar la velocidad no tiene efectos como por ejemplo colisiones o interferencias (algo que si ocurre al aumentar *datarate*, número de nodos o al empeorar el medio inalámbrico).

Una vez más, los protocolos clásicos AODV y OLSR se mantienen muy por debajo de lo obtenido con la implementación realizada y difícilmente pueden degradar aún más su rendimiento.

En este caso, las dos técnicas propuestas han obtenido unos muy buenos resultados, similares en ambos casos.

4.6 Experimentaciones físicas con el protocolo funcionando sobre sistemas empotrados (*Raspberry Pi*)

Previo al desarrollo de este trabajo, se implementó sobre nodos reales la aproximación al protocolo SRAR con un sistema experto basado en reglas.

Por limitaciones presupuestarias, la experimentación asociada al mismo tuvo en cuenta el dinamismo de forma simbólica, se simularon conexiones y desconexiones entre nodos a distancias relativamente cortas. Se observó, por tanto, como reaccionaba el protocolo implementado a los cambios de ruta, obteniendo un buen comportamiento asociado a la entrega de casi la totalidad de los paquetes.

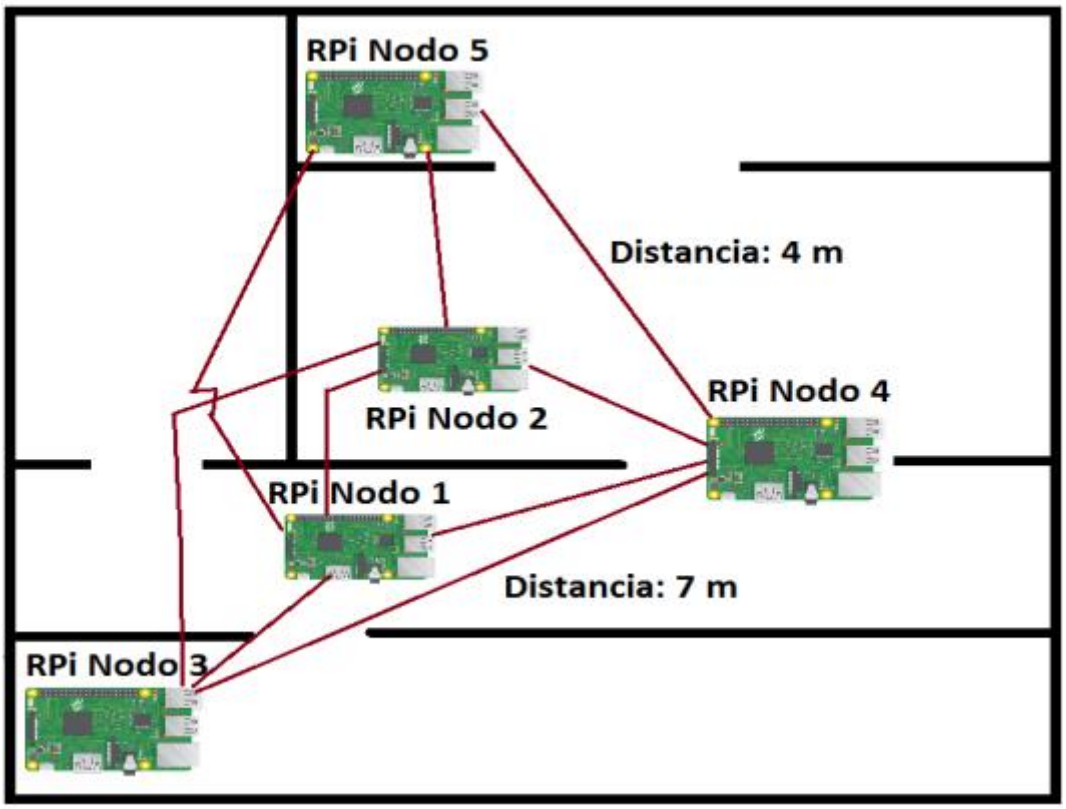


Ilustración 34. Escenario utilizado en la implementación física

La experimentación consistió ir variando los nodos emisores en las distintas campañas de experimentos. Se configuraron los nodos de forma que únicamente comunicasen con los vecinos marcados en el escenario descrito en la ilustración 22. Durante el proceso de envío de paquetes, los nodos intermedios se desconectaban y se estudiaba si el protocolo era capaz de hacer frente a las desconexiones correctamente.

ENLACES	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
Nodo 1	**********	%H=97,72 %A=100 %S=100 %D=100 TH=103ms TS=226ms TD=206ms TA=148ms	%H=93,65 %A=100 %S=100 %D=100 TH=29,9ms TS=65,5ms TD=69,6ms TA=38,4ms	%H=100 %A=100 %S=100 %D=100 TH=24ms TS=40,7ms TD=64,8ms TA=44,1ms	%H=85,71 %A=73,68 %S=67,85 %D=73,33 TH=1200ms TS=953ms TD=1087ms TA=1353ms
Nodo 2	%H=100 %A=100 %S=100 %D=100 TH=27,5ms TS=16,6ms TD=8,3ms TA=5ms	**********	%H=100 %A=100 %S=100 %D=100 TH=34,2ms TS=132ms TD=19ms TA=22ms	%H=95,65 %A=100 %S=100 %D=100 TH=17,5ms TS=25,2ms TD=7,3ms TA=5,9ms	%H=90,9 %A=92,85 %S=100 %D=100 TH=1670ms TS=1750ms TD=1764ms TA=1520ms
Nodo 3	%H=95,16 %A=100 %S=100 %D=100 TH=8ms TS=4,4ms TD=3,3ms TA=12,7ms	%H=93,18 %A=100 %S=100 %D=100 TH=70,8ms TS=110ms TD=121ms TA=125ms	**********	%H=87,5 %A=100 %S=100 %D=100 TH=22,2ms TS=12,2ms TD=14,5ms TA=9,8ms	%H=73,17 %A=77,19 %D=80 TH=1024ms TD=1349ms TA=1373ms
Nodo 4	%H=100 %A=100 %S=90,9 %D=80 TH=85,3ms TS=79,6ms TD=149ms TA=120ms	%H=100 %A=100 %S=90,9 %D=80 TH=102ms TS=151ms TD=303ms TA=175ms	%H=96 %A=95,23 %S=81,81 %D=80 TH=85,3ms TS=124ms TD=251ms TA=121ms	**********	%H=92 %A=69,56 %S=72,72 %D=80 TH=1976ms TS=2038ms TD=2300ms TA=1873ms
Nodo 5	%H=91,8 %A=93,181 %S=100 %D=100 TH=46,9ms TS=31,8ms TD=109ms TA=123ms	%H=97,67 %A=100 %S=100 %D=100 TH=148ms TS=186ms TD=196ms TA=156ms	%H=70,37 %A=90,47 %D=87,5 TH=75,8ms TD=294ms TA=250ms	%H=95,65 %A=100 %S=100 %D=100 TH=35,1ms TS=31,2ms TD=105ms TA=51ms	**********

Ilustración 35. Experimento con nodos reales. Tamaño paquete datos 128B, periodo de envío cada 4 segundos. Todos enviaban a todos.

En aquella experimentación, la implementación tenía dos tipos de paquetes más:

- ACK para los mensajes HELLO, que se enviaban mediante *flooding*
- STATUS, enviados a nodos a un salto para refrescar la entrada en la tabla de vecinos

La nomenclatura mostrada en las tablas corresponde a lo siguiente:

- %H,%A,%D,%S: PDR de paquetes HELLO, ACK,DATA y STATUS
- TH,TA,TD,TS: *Delay end to end* de paquetes HELLO, ACK,DATA y STATUS

Los resultados fueron muy positivos y se entregaron casi la totalidad de los paquetes de la experimentación para los que se tenía una ruta origen-destino posible. Estos resultados sobre nodos reales siguen la línea de los obtenidos con la simulación, mostrando en ambos casos un buen funcionamiento para el protocolo.

5. Conclusiones

El principal objetivo de este trabajo ha consistido en realizar una aportación a nivel experimental al campo de las redes vehiculares.

Para poder llevarlo a cabo se realizó un estudio exhaustivo de distintas tecnologías y protocolos de red usadas en dicho campo, además de diversas técnicas de IA, para concluir las más adecuadas para su inclusión en los protocolos y aprovechar sus características con el objetivo de diseñar un protocolo de enrutamiento robusto y eficiente.

Los entornos de simulación a utilizar fueron elegidos en base al buen soporte que estos reciben, la flexibilidad que otorgan y al gran uso que hacen de ellos a lo largo de la comunidad de internet. De esta forma, se aseguró que las herramientas con las que se realizó la experimentación diesen resultados válidos.

Teniendo los resultados de la experimentación, se puede concluir con lo siguiente:

- Se ha presentado el estado del arte en cuanto al desarrollo de aplicaciones y protocolos usados en redes vehiculares, así como lo relativo a las redes neuronales. Esta información ha servido para guiar en la implementación de un protocolo con características diferentes que basase sus decisiones en parámetros directamente relacionados con el dinamismo de las redes.
- Se han mejorado las capacidades de dos simuladores con amplias funcionalidades y nuevos modelos, para poder llevar a cabo la experimentación que analizase de qué forma operaba lo implementado y como de positivos eran sus resultados. Las herramientas utilizadas han permitido compararlos con otro tipo de protocolos más clásicos como son AODV y OLSR.
- Utilizar un *framework* para implementación de redes neuronales ha permitido simplificar la tarea de diseñar una estructura de red neuronal adecuada para este trabajo.
- La implementación ha sido realizada con éxito, y esto se ha avalado con los resultados de la experimentación y su comparación con experimentos sobre nodos reales. La explicación general de lo implementado, así como aspectos más técnicos, han sido expuestos lo más claramente posible, para que el lector pudiese comprender el trabajo con mayor facilidad.
- Experimentar con lo implementado analizando diferentes parámetros indicadores de su funcionamiento, ha permitido observar los puntos fuertes y débiles de cada protocolo, a partir de estos se sientan las bases para futuras propuestas de mejora.
- Se han utilizado técnicas de *scripting* y *parsing* de información para ahorrar trabajo repetitivo en el desarrollo del mismo, y, en la medida de lo posible conseguir agilizar el proceso de desarrollo.
- Se realizó previamente una experimentación con el protocolo implementado sobre sistemas empotrados (*raspberry pi*) consistente en observar como el protocolo hacia frente a las conexiones y desconexiones entre sus nodos. Sus resultados siguen la línea de lo obtenido con las simulaciones.

- Se deja abierta la posibilidad aplicar los protocolos a otro tipo de entornos, como por ejemplo el medio subacuático, donde se pueden obtener considerables mejoras respecto a los protocolos propuestos hasta el momento.

Observando los resultados de la implementación, se puede afirmar que los protocolos implementados han conseguido adaptar su funcionalidad a las características dinámicas de las redes VANET. Esto se ha traducido en un aumento muy notable en el número de paquetes entregados correctamente, que ya no se pierden en el trayecto, como sucede al usar otros protocolos de referencia.

En términos de *packet delivery ratio* es donde principalmente destaca este protocolo, proporcionando una elevada fiabilidad con un reducido overhead de control, consistente en un solo paquete. Esto es posible gracias al buen recorrido que pueden llevar los paquetes dentro de la red, ya que cada *next hop* es elegido cuidadosamente para asegurar en la medida de lo posible su entrega final, consiguiendo paso a paso para cada paquete la ruta que tenga el mejor balance de fiabilidad y eficiencia.

Ha sido un acierto tener en cuenta aspectos ligados al dinamismo para que los sistemas de decisión, ya sea el sistema experto basado en reglas como el de red neuronal, puedan tenerlos en cuenta y paliar los problemas que aparecían en otros protocolos debidos a la alta movilidad de sus nodos. Se ha analizado cual era la debilidad y a partir de ella se han estudiado parámetros que ayuden a combatirla.

Viendo los resultados en lo que respecta al *end to end delay*, podría pensarse que las propuestas implementadas no han obtenido tan buenos resultados en comparación con AODV u OLSR, sin embargo, una vez que tenemos en cuenta el número de *hops* que se han podido realizar para cada paquete utilizando cada protocolo, podemos observar como la razón por la que se obtienen valores más altos es por que únicamente se tiene en cuenta el *delay* de los paquetes que llegan a ser recibidos. En el caso de AODV y OLSR los pocos paquetes entregados suelen haber dado de 1 a 3 *hops*, mientras que para SRAR y SRARNN, se entregan casi todos los paquetes y este valor puede llegar hasta más de 10 *hops* antes de alcanzar el destino del paquete, hecho que aumenta por acumulación el *delay* total.

Si se observa cómo afronta cada protocolo los cambios en parámetros significativos de la red se aprecia lo siguiente:

- En cuanto al número de nodos, las dos propuestas de protocolo escalarían mejor que AODV u OLSR. Se puede aumentar el número de nodos sin que el rendimiento se vea reducido a mínimos, como en el caso de los dos últimos.
- Al ver cómo afecta el medio de transmisión a cada protocolo, se resalta que también se defienden mejor que el resto. Si los nodos comunican por un buen canal obtienen muy buenos resultados, entregando casi la totalidad de los datos. Si los llevamos a un escenario con un medio más duro y proclive a fallos se ve una ligera bajada en sus rendimientos. Aun así no son comparables con los que se obtienen en AODV u OLSR, ya que estos cuentan desde el principio con un rendimiento muy bajo, que difícilmente puede verse más reducido.
- La velocidad de los nodos no tiene prácticamente efecto en los resultados del PDR de cada uno de los protocolos propuestos, se ve una gráfica prácticamente recta, cosa que para AODV y OLSR se puede observar como a mayores

velocidades la entrega correcta disminuye al no ser protocolos adaptados a un dinamismo tan elevado.

- El efecto del *datarate* es el más sufrido por las propuestas implementadas, las gráficas de PDR caen de manera más pronunciada a medida que subimos este valor. Esto puede deberse a que el *datarate* de las redes *ad-hoc* viene limitado por naturaleza, y aumentar la tasa de envío satura los enlaces un poco más dando pie a una mayor probabilidad de perder paquetes, no hay que olvidar que estos pueden haber dado en el experimento hasta más de 10 saltos hasta llegar al destino. AODV y OLSR no han sufrido su efecto al contar desde el inicio con un rendimiento difícilmente empeorable.

5.1 Trabajos futuros

Los resultados obtenidos en este trabajo, pueden servir de base para futuras líneas de investigación.

Viendo los resultados obtenidos con el protocolo basado en red neuronal y sabiendo que este tipo de tecnologías tienen un amplio potencial, la propuesta aquí estudiada puede servir como primera aproximación para realizar implementaciones de protocolos similares, pero en otro tipo de entornos, que requieran tomar decisiones más exigentes contemplando un mayor número de parámetros.

En entornos de aplicación como las redes subacuáticas es necesario. Será muy interesante aplicar como motor de decisión una red neuronal como se ha propuesto en este trabajo, dado que tiene la capacidad de procesar la gran cantidad de parámetros a considerar en dicho contexto (posición, vecinos, distancia, profundidad, presión, salinidad, calidad de señal, nivel de batería (propio y de nodos intermedios), etc.) y que pueden ser determinantes para mejorar las prestaciones de los protocolos de redes subacuáticas actuales y su fiabilidad. De esta forma, partiendo de la base de este trabajo, se está considerando como parte de mi futura tesis doctoral el diseño de un protocolo de enrutamiento subacuático que aplicando las técnicas aquí contempladas mejorara las prestaciones y fiabilidad en las comunicaciones móviles submarinas, lo que resulta actualmente de gran interés al incluirse de forma considerable UAVs o drones subacuáticos en las mismas.

5.2 Aportaciones

Fruto de los resultados obtenidos en este trabajo se ha realizado la publicación que se detalla a continuación, estando en preparación otro artículo que incluirá los resultados preliminares de las líneas futuras de trabajo:

Javier Moraga, José Navarro, Alberto Bonastre, Rafael Ors, Juan José Serrano, Juan V. Capella (2018). DESARROLLO DE UN NUEVO MODELO DE PROTOCOLO GEOGRÁFICO PARA REDES DE SENSORES MÓVILES. XII International Workshop on Sensors and Molecular Recognition, Valencia, España.

6. Bibliografía

[1] Bijan Paul,Md. Ibrahim and Md. Abu Naser Bikas, "*VANET Routing Protocols: Pros and cons"* ,International journal of Computer Applications (0975-8887) Volume 20-No.3,April 2011

[2] Mir, Z. and Filali, F. (2014) "*LTE and IEEE 802.11p for Vehicular Networking: A Performance Evaluation*" EURASIP Journal on Wireless Communications and Networking, 2014, 89. http://dx.doi.org/10.1186/1687-1499-2014-89

[3] Sandhaya Kohli, Bandanjot Kaur, Sabina Bindra. "*A comparative study of Routing Protocols in VANET"*

[4] Fan Li, Yu Wang. "*Routing in Vehicular Ad Hoc networks: A survey"*, IEEE Vehicular Technology Magazine, June 2007

[5]. G. Liu, B.-S. Lee, B.-C. Seet, C.H. Foh, K.J. Wong, and K.-K. Lee, *"A routing strategy for metropolis vehicular communications,"* in International Conference on Information Networking (ICOIN), pp. 134–143, 2004.

[5.1]. H. Füßler, M. Mauve, H. Hartenstein, M. Kasemann, and D. Vollmer, *"Locationbased routing for vehicular ad-hoc networks,"* ACM SIGMOBILE Mobile Computing and Communications Review (MC2R), vol. 7, no. 1, pp. 47–49, Jan. 2003.

[6] M. Durresi, A. Durresi, and L. Barolli, *"Emergency broadcast protocol for intervehicle communications,"* in ICPADS '05: Proceedings of the 11th International Conference on Parallel and Distributed Systems—Workshops (ICPADS'05), 2005.

[7] M. Sun, W. Feng, T.-H. Lai, K. Yamada, H. Okada, and K. Fujimura, *"GPS-based message broadcasting for inter-vehicle communication,"* in ICPP '00: Proceedings of the 2000 International Conference on Parallel Processing, 2000

[8] Marie-Ange Lèbre, Frédéric Le Mouël, Eric Menard, Julien Dillschneider, Richard Denis. "*VANET Applications: Hot Use Cases"*, Center of Innovation in Telecommunications and Integration of Services, University of Lyon,1 Jul 2014.

[9] Rakesh Kumar, Mayank Dave, *"A Comparative Study of Various Routing Protocols in VANET"* IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011

[10] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode, *"Trafficview: traffic data dissemination using car-to-car communication,"* Mobile Computing and Communications Review, vol. 8, no. 3, pp. 6–19, 2004

[11] Kayhan Zrar Ghafoor, Kamalrulnizam Abu Bakar, Shaharuddin Salleh, Kevin C. Lee, Mohd Murtadha Mohamad, Maznah Kamat, and Marina Md Arshad. "Fuzzy logic-assisted geographical routing over vehicular ad hoc networks". International Journal of Innovative Computing, Information and Control, 8(7(B)):5095–5120, July 2012. ISSN 1349-4198.

[12] Navarro Alabarta, José *"Evaluación mediante simulación de redes de sensores aplicatas a robots móviles y vehículos ligeros"*, 14 Diciembre de 2015

[13] Genbetadev.com. (2011). *¿Cómo calcular la distancia entre dos puntos geográficos en C#? (Fórmula de Haversine)*. [online] Disponible en: http://www.genbetadev.com/cnet/como-calcular-la-distancia-entre-dos-puntos-geograficos-en-c-formula-de-haversine [Accedida 3 Jul. 2018].

[14] Seattlerobotics.org. (2016). *Fuzzy Logic Tutorial - An Introduction*. [activa] Disponible en: http://www.seattlerobotics.org/encoder/mar98/fuz/flindex.html [Accedida 3 Jul. 2018].

[15] Kashif Naseer Qureshia, Abdul Hanan Abdullaha, Raja Waseem Anwara, Muhammad Anwara, Khalid Mahmood Awan "Aegrp: An enhanced geographical routing protocol for VANET", 9 de February 2016

[16] R. Brendha and V. S. J. Prakash, "A survey on routing protocols for vehicular Ad Hoc networks," 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, 2017, pp. 1-7.

[17] Nagaraj, Uma & Dhamal, Poonam. "Broadcasting Routing Protocols in VANET". 2018

[18] Brad Karp and H. T. Kung. "GPSR: greedy perimeter stateless routing for wireless networks" In Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom '00). ACM, New York, NY, USA, 243-254. Year 2000

[19] M. Nekovee and B. B. Bogason, "Reliable and Effcient Information Dissemination in Intermittently Connected Vehicular Adhoc Networks," 2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring, Dublin, 2007, pp. 2486-2490.

[20] Tseng, YC., Ni, SY., Chen, YS. et al. Wireless Networks. "The Broadcast Storm Problem in a Mobile Ad Hoc Network" (2002)

[21] Felipe Cunha, Leandro Villas, Azzedine Boukerche, Guilherme Maia, Aline Viana, Raquel A. F. Mini, Antonio A. F. Loureiro, "Data communication in VANETs: Protocols, applications and challenges", Ad Hoc Networks,Volume 44,2016,Pages 90-103,ISSN 1570-8705.

[22] Eze, E.C., Zhang, SJ., Liu, EJ. et al. "Advances in vehicular ad-hoc networks (VANETs): Challenges and road-map for future development". Int. J. Autom. Comput. (2016) 13: 1.

[23] Saleh Yousefi & Mahmood Fathy "Metrics for performance evaluation of safety applications in vehicular ad hoc networks", Transport, 23:4, 291-298, Year 2008

[24] Alexander, Paul D., David Haley and Alex J. Grant. "Cooperative Intelligent Transport Systems: 5.9-GHz Field Trials." Proceedings of the IEEE 99 (2011): 1213-1235.

[25] United States Department of transportation. "DSRC: The Future of Safer Driving" [activa] Disponible en: https://www.its.dot.gov/factsheets/dsrc_factsheet.htm [Accedida 3 Jul. 2018].

[26] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, S. Tsugawa. "Overview of platooning systems". Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)

[27] Anna Maria Vegni, Mauro Biagi and Roberto Cusani (February 13th 2013). "Smart Vehicles, Technologies and Main Applications in Vehicular Ad hoc Networks", Vehicular Technologies Lorenzo Galati Giordano, IntechOpen.

[28] Amit Kumar Saha and David B. Johnson. "Modeling mobility for vehicular ad-hoc networks". In Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks (VANET '04). ACM, New York, NY, USA, 91-92. Year 2004

[29] A. R. Khan, S. M. Bilal and M. Othman, "A performance comparison of open source network simulators for wireless networks," 2012 IEEE International Conference on Control System, Computing and Engineering, Penang, 2012, pp. 34-38.

[30] Riley G.F., Henderson T.R. “The ns-3 Network Simulator”. In: Wehrle K., Güneş M., Gross J. (eds) Modeling and Tools for Network Simulation. Springer, Berlin, Heidelberg. 2010

[31] Ns-3 a discrete event network simulator. “Tutorial”. [Activa] Accesible en: https://www.nsnam.org/docs/tutorial/html/introduction.html [Accedida 3 Jul. 2018].

[32] Simulation of urban mobility. “Wiki”. [Activa] Accesible en: http://sumo.dlr.de/wiki/Simulation_of_Urban_MObility_-_Wiki. [Accedida 3 Jul. 2018].

[33] Institute of transportation systems. “SUMO – Simulation of Urban MObility”. [Activa] Accesible en: https://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/. [Accedida 3 Jul. 2018].

[34] A. K. Jain, Jianchang Mao and K. M. Mohiuddin, "Artificial neural networks: a tutorial," in Computer, vol. 29, no. 3, pp. 31-44, Mar 1996.

[35] Departamento de lenguajes y ciencias de la computación de la universidad de Málaga. “El perceptrón simple”. [Activo] Accesible en: http://www.lcc.uma.es/~munozp/documentos/modelos_computacionales/temas/Tema4MC-05.pdf [Accedida 3 Jul. 2018].

[36] Maind, S.B. & Wankar, P. (2014). “Research paper on basic of Artificial Neural Network”. International Journal on Recent and Innovation Trends in Computing and Communication. 2. 96-100.

[37] Douglas M. Hawkins. “The Problem of Overfitting”. Journal of Chemical Information and Computer Sciences 2004 44 (1), 1-12

[38] OpenNN neural networks. “OpenNN”. [Activa] Accesible en: http://www.opennn.net/ [Accedida 3 Jul. 2018].

[39] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. Neural Networks, Volume 61, 2015 ,Pages 85-117, ISSN 0893-6080.

[40] KDNuggets (2015). “Preventing overfitting in Neural Networks”. [Activa] Accesible en: https://www.kdnuggets.com/2015/04/preventing-overfitting-neural-networks.html/2 [Accedida 3 Jul. 2018].

[41] Robert Kieser, Pall Reynisson, Timothy J. Mulligan; Definition of signal-to-noise ratio and its critical role in split-beam measurements, ICES Journal of Marine Science, Volume 62, Issue 1, 1 January 2005, Pages 123–130.

[42] Fei Qin; Xuewu Dai; John E. Mitchell. “Effective-SNR estimation for wireless sensor network using Kalman filter”. Ad Hoc Networks, ISSN: 1570-8705, Vol: 11, Issue: 3, Page: 944-958. Year 2013

[43] Ns-3 a discrete event network simulator. “WifiRadioEnergyModel class reference”. [Activa] Accesible en: https://www.nsnam.org/doxygen/classns3_1_1_wifi_radio_energy_model.html. [Accedida 3 Jul. 2018].

[44] Michael Tsai (2011). “Path loss and shadowing (Large-scale fading)”. [Activa] Accesible en: https://www.csie.ntu.edu.tw/~hsinmu/courses/_media/wn_11fall/path_loss_and_shadowing.pdf. [Accedida 3 Jul. 2018].

[45] S. Srinivasa and M. Haenggi, "Path loss exponent estimation in large wireless networks," 2009 Information Theory and Applications Workshop, San Diego, CA, 2009, pp. 124-129.

[46] Thomas Schwengler(2016) “Chapter 4: Fading, shadowing and link budgets” [Activa] Accesible en: http://morse.colorado.edu/~tlen5510/text/classwebch4.html. [Accedida 3 Jul. 2018].

[47] Telecom ABC. "DSSS - Direct Sequence Spread Spectrum". [Activa] Accesible en: http://www.telecomabc.com/d/dsss.html. [Accedida 3 Jul. 2018].

[48] Oklahoma State University. "Ajith Abraham (2004). [Activa] Accesible en: http://03.softcomputing.net/fuzzy_chapter.pdf. [Accedida 3 Jul. 2018].

[49] Grosan C., Abraham A. (2011). "Rule-Based Expert Systems. In: Intelligent Systems". Intelligent Systems Reference Library, vol 17. Springer, Berlin, Heidelberg

7. Anexo

7.1 Escenarios utilizados en la simulación

Para cada experimento realizado mediante simulación, se ha necesitado la creación de un espacio que emule un lugar real dentro del simulador. Es aquí donde se posicionan todos los nodos que participan en la simulación, que se moverán en su interior con el comportamiento que se necesite, dependiendo del fenómeno a evaluar.

En este primer apartado del anexo, se exponen los escenarios generados para ser usados en las simulaciones.

7.1.1 Aspectos de código del script generador de rutas

Utilizando las herramientas proporcionadas por el propio simulador para generar flujos de movilidad, no conseguía generar flujos con vehículos que no sobrepasasen en ningún momento los 200 metros de distancia con los nodos vecinos. Por ello, creé un *script* en C++ que, aprovechando el conocimiento que tenía sobre el formato de los ficheros de rutas, fuese capaz de generar uno a partir de parámetros como la velocidad, la distancia intervehicular y el tamaño del escenario, para no superar el máximo permitido.

7.1.1.1 Llamada al script

```
int main()
{
    int numN;
    double velMS;
    double distVehicles;
    cout << "Introduce el numero de nodos de la simulacion" << endl;
    cin >> numN;
    cout << "Introduce la velocidad en m/s" << endl;
    cin >> velMS;
    cout << "Introduce la distancia entre vehiculos" << endl;
    cin >> distVehicles;
    creaXMLRutas(numN,450,sqrt(50*50+50*50),3950,velMS,distVehicles);
    return 0;
}
```

Ilustración 36. Flujo principal del *script* para generar rutas: Escenario 4000*500 m

Después de ser compilado, el script se ejecuta, pidiendo por línea de comandos el número de nodos, la velocidad y la distancia de los mismos. La función creaXMLRutas es la encargada de generar el fichero de rutas.

7.1.1.2 creaXMLRutas

```
void creaXMLRutas(int numNodos, double lenL1, double lenL2, double lenL3, int velMS, double distVeh);
```

Ilustración 37. *Header* de la función encargada de crear el fichero de rutas.

Mediante esta función, se realiza la distribución regular de nodos en los 3 tramos del escenario. Los parámetros que se pasan corresponden a los pedidos por línea de comandos como son el número de nodos, la velocidad de los mismos o la distancia que han de reservar.

Las distancias de los tramos del escenario también se pasan, de esta forma controlamos que no se metan nodos en localizaciones inválidas.

7.1.2 Escenarios generados

30 nodos a 25 m/s

15 vehículos en cada dirección, que parten a 100 metros de distancia unos de otros. La simulación en este escenario dura 120 segundos, teniendo el segundo 30 como inicio de la aplicación generadora de tráfico en los emisores y el segundo 90 como el segundo para el que se detiene la misma. El escenario ha tenido unas dimensiones de 6000*500 m.

100 nodos a 25 m/s

50 vehículos en cada dirección, que parten a 30 metros de distancia unos de otros. La simulación en este escenario dura 150 segundos, teniendo el segundo 30 como inicio de la aplicación generadora de tráfico en los emisores y el segundo 120 como el segundo para el que se detiene la misma. El escenario ha tenido unas dimensiones de 4000*500 m.

150 nodos a 25 m/s

75 vehículos en cada dirección, que parten a 20 metros de distancia unos de otros. La simulación en este escenario dura 160 segundos, teniendo el segundo 30 como inicio de la aplicación generadora de tráfico en los emisores y el segundo 160 como el segundo para el que se detiene la misma. El escenario ha tenido unas dimensiones de 4000*500 m.

200 nodos a 25 m/s

100 vehículos en cada dirección, que parten a 10 metros de distancia unos de otros. La simulación en este escenario dura 160 segundos, teniendo el segundo 30 como inicio de la aplicación generadora de tráfico en los emisores y el segundo 130 como el segundo para el que se detiene la misma. El escenario ha tenido unas dimensiones de 4000*500 m.

250 nodos a 25 m/s

125 vehículos en cada dirección, que parten a 10 metros de distancia unos de otros. La simulación en este escenario dura 160 segundos, teniendo el segundo 30 como inicio de la aplicación generadora de tráfico en los emisores y el segundo 130 como el segundo para el que se detiene la misma. El escenario ha tenido unas dimensiones de 4000*500 m.

150 nodos a 15 m/s

75 vehículos en cada dirección, que parten a 20 metros de distancia unos de otros. La simulación en este escenario dura 160 segundos, teniendo el segundo 30 como inicio de la aplicación generadora de tráfico en los emisores y el segundo 130 como el segundo para el que se detiene la misma. El escenario ha tenido unas dimensiones de 4000*500 m.

150 nodos a 20 m/s

75 vehículos en cada dirección, que parten a 20 metros de distancia unos de otros. La simulación en este escenario dura 160 segundos, teniendo el segundo 30 como inicio de la aplicación generadora de tráfico en los emisores y el segundo 130 como el segundo para el que se detiene la misma. El escenario ha tenido unas dimensiones de 4000*500 m.

150 nodos a 30 m/s

75 vehículos en cada dirección, que parten a 20 metros de distancia unos de otros. La simulación en este escenario dura 160 segundos, teniendo el segundo 30 como inicio de la aplicación generadora de tráfico en los emisores y el segundo 130 como el segundo para el que se detiene la misma. El escenario ha tenido unas dimensiones de 5250*500 m.