UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Trabajo Final de Máster

Evaluación mediante simulación de redes de sensores aplicadas a robots móviles y vehículos ligeros

Autor:
Navarro Alabarta, José

Dirigida por:
Dr. Capella Hernández, Juan Vicente
Dr. Valera Fernández, Ángel

Trabajo Final de Máster en Automática e Informática Industrial

Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas (ITACA)
Instituto de Automática e Informática Industrial (ai2)
Departamento de Ingeniería de Sistemas y Automática (DISA)
Departamento de Informática de Sistemas y Computadores (DISCA)

14 de diciembre de 2015

Resumen

En el siguiente Trabajo Final de Máster se realiza un estudio de los protocolos de enrutamiento de redes vehiculares. Estos protocolos se dividen en dos grandes grupos, los protocolos basados por el encaminamiento y los basados por localización geográfica. Este primer grupo a la vez se divide en dos grupos, los basados en protocolos reactivos y los proactivos. La diferencia entre estos dos subtipos de protocolos es que los reactivos obtienen la información de direccionamiento en el momento exacto que se necesita y en cambio los protocolos proactivos están constantemente enviando y solicitando la información. Los protocolos más representativos de estudio son AODV y DSR en cuanto a protocolos reactivos, y OLSR y DSDV en cuanto a proactivos. Respecto a los protocolos basados en geolocalización, estos se caracterizan porque necesitan de dispositivos GPS para poder trabajar correctamente. En los últimos años a este tipo de protocolos se les ha ido añadiendo técnicas de inteligencia artificial para optimizar y hacerlos más robustos y fiables.

Este tipo de protocolos se integran con sistemas de transporte inteligente, de esta forma, con este tipo de sistema se pretende reducir el número de accidentes y otros factores que afectan a la conducción.

En cuanto a la etapa de experimentación, el proceso se divide en diversas subetapas, la primera de ellas esta basada en el diseño y desarrollo de protocolos confiables y robustos. Una vez superada esta etapa hay que obtener tendencias de conducción con los simuladores de movilidad, para así, obtener la dicha información y suministrarla al simulador de redes inalámbricas para poder estudiar los diversos comportamientos que van ocurriendo, como es el caso del estudio de la densidad de los nodos, de como afectan las sobrecargas en la red, el retardo en la entrega de los paquetes, además de la tasa de paquetes perdidos. Esta etapa, permite iterar sobre ella, de forma que según los resultados obtenidos se puede redefinir las especificaciones, además permite la evaluación de las propuestas con un gran número de dispositivos sin constes adicionales. Cuando los resultados son los deseados se pasa a la última etapa del proceso, que consiste en implementar el protocolo en equipos físicos.

Este TFM cubre las dos primeras etapas descritas, constituyendo un primer paso necesario para el desarrollo de futuros trabajos en la línea de mejorar los sistemas basados en protocolos de enrutamiento basados en GPS aplicables tanto a vehículos como robots móviles. Además se ha propuesto un ecosistema completo para el diseño y evaluación de este tipo de protocolos. En esta línea, se ha desarrollado un modelo de protocolo de enrutamiento para VANETs con características novedosas, habiéndose validado y realizado

una completa evaluación del mismo mediante el ecosistema propuesto. Finalmente, también se han realizado ya las primeras implementaciones y pruebas sobre hardware real que permitirá la aplicación de los protocolos propuestos, con resultados muy positivos.

Resum

En aquest Treball Final de Màster és realitza un estudi dels protocols d'encaminament de xarxes vehiculars. Aquestos protocols es divideixen en dos grans grups, els protocols basats per l'encaminament i els basats per localització geogràfica. El primer grup a més a més, es divideix en altres dos subgrups, els basats amb protocols reactius i amb els proactius. La diferència amb ambdós subtipus de protocols es que al cas dels reactius adquireixen la informació d'encaminament al moment exacte que es necessita, i en canvi estan constantment transmitent-la i sol·licitant-la. Els protocols més representatius d'estudi son AODV i DSR en quant al reactius, i OLSR i DSDV en quant als proactius. Pel que fa als protocols basats per geolocalització es caracteritzen per emprar dispositius GPS per al seu correcte funcionament. Als últims anys a aquest tipus de protocols se'ls ha anat afegint tècniques d'intel·ligència artificial per optimitzar i fer-los més robustos i fiables.

Aquests tipus de protocols s'integren amb els sistemes de transport intel·ligent, d'aquesta manera els sistemes pretenen reduir el nombre d'accidents i altres factors que afecten a l'hora de conduir.

En quant a l'etapa d'experimentació el proces es divideix com es diverses subetapes, la primera consta del disseny i desenvolupament dels protocols confiables i robustos. Una vegada superada aquesta etapa hi ha que llançar simulacions dels models de mobilitat per a obtindre els resultats, seguidament aquesta informació es subministrada als simuladors de xarxes sense fils per a observar el que va passant, segons canvie la densitat dels nodes, com afecten les sobrecarregues a la xarxa i els temps d'entrega, a més a més de la taxa de paquets perduts. Aquesta etapa ofereix la possibilitat d'iterar sobre ella mateixa, de tal forma que segons es vagen obtenint els resultats es poden redefinir les especificacions a més, d'avaluar les propostes amb un gran nombre de dispositius sense fer despeses addicionals. En quan els resultats son els desitjats es passa a l'última etapa del proces, què consisteix en implementar el protocol en dispositius físics.

Aquest TFM cobreix les dos primeres etapes del procés mencionat, constituint una primera etapa necessària per al desenvolupament de futurs treballs que estiguen orientats en línia de millorar els sistemes basats en protocols d'encaminament basats amb dispositius GPS aplicables tant a vehicles com a robots mòbils. A més a més. s'ha proposat un ecosistema complet per al disseny i avaluació d'aquest tipus de protocols. En aquesta línia, s'ha desenvolupat un model de protocol d'encaminament per a xarxes VANETs amb noves característiques, havent-se validat i realitzat ja les primeres implementacions i probes sobre dispositius reals que permetran l'aplicació dels protocols proposats, amb resultats prou diferents.

Abstract

This master thesis presents a study of routing protocols for vehicular networks. These protocols are divided into two main groups: protocols based on routing and protocols based on geographical location. The first group is divided into two groups, proactive and reactive protocols. The difference between these two subtypes of protocols is that the necessary control information is obtained when needed in the reactive protocols while proactive protocols are constantly sending and requesting control information. The most representative reactive protocols are AODV and DSR, and the most representative proactive protocols are OLSR and DSDV. On the other hand, geolocation based protocols are characterized by the necessity of GPS devices to work properly. In recent years, protocols of this last type have been improved applying artificial intelligence techniques to optimize their functioning and increase their robustness and reliability.

All these protocols are integrated with intelligent transport systems. In this way, the goal with these systems is to reduce the number of accidents among other factors that affect driving.

Regarding the experimental phase, the process is divided into several sub-steps, the first involves the design and development of robust and reliable protocols. Once this step has been passed driving trends should be obtained with the mobility simulators in order to be provided to the network simulators to study and evaluate the behavior and performance. This stage allows iterate over it, so that according to the results you can refine the specification, also allows the evaluation of proposals with a large number of devices without additional costs. Finally, last step consists on the real implementation on physical devices.

This TFM covers the first two described stages, being a necessary work that will allow the development of future lines dedicated to improve based on GPS routing protocols that can be applied to both vehicles and mobile robots. Additionally, a complete ecosystem for the design and evaluation of such protocols has been proposed in this thesis. In this line, a model for VANETs routing protocol with new characteristics has been developed and validated. After that, an exhaustive experimentation and evaluation using the proposed ecosystem has been conducted. Finally, early implementations and testing on real hardware that will allow the application of the proposed protocols also have already been carried out, with very interesting results.

Índice general

Resumen II

Resum IV

Abstract VI

Contenido VIII

Lista de Figuras XI

Lista de Tablas XIII

1. Introducción 1
- 1.1. Objetivos del Trabajo Final de Máster . . . 2
- 1.2. Estructura del TFM . . . 2

2. Estado del arte 5
- 2.1. Tecnología inalámbrica . . . 5
- 2.2. Técnicas de IA para VANETs . . . 7
- 2.3. Estudio protocolos . . . 9
  - 2.3.1. Clasificación Protocolos . . . 9
  - 2.3.2. ERBA . . . 10
  - 2.3.3. Hybrid Location Ad-Hoc Routing (HLAR) . . . 13
  - 2.3.4. Intelligent OLSR . . . 15
  - 2.3.5. Fast energy-aware OLSR . . . 16
  - 2.3.6. Junction-based Adaptive Reactive Routing (JARR) . . . 18
  - 2.3.7. Reliable Geocast Routing Protocol . . . 20
  - 2.3.8. Routing algorithm for Dense Traffic Density Vehicular Networks (BTDAR) . . . 22
  - 2.3.9. Intersection-Based Routing Protocol (IBR) . . . 23
  - 2.3.10. Stability and Reliability aware Routing (SRR) . . . 24

3. Herramientas Empleadas 31
- 3.1. NS-3 . . . 31
  - 3.1.1. Arquitectura del simulador . . . 32
- 3.2. Simulator Urban MObility - SUMO . . . 32

3.3. Robot Operating System - ROS 33
3.4. BeagleBone Black 35
3.5. ROBOCape 36

4. Trabajo Desarrollado **39**
4.1. Ecosistema 39
4.2. Generación de escenarios en SUMO 40
4.3. Desarrollo de un modelo en NS-3 44
4.3.1. Modelo de propagación empleado 44
4.3.2. Implementación del protocolo de red 46
4.4. Nodos ROS 55
4.4.1. Descripción nodos de ROS 55
4.4.2. Integración librerías en ROS 57

5. Experimentación y resultados **59**
5.1. Evaluación del protocolo propuesto 59
5.1.1. Parámetros de la simulación y escenarios 59
5.1.2. Impacto del número de nodos 62
5.1.3. Impacto del *Channel Shadowing* 63
5.1.4. Impacto de la velocidad de los nodos 63
5.1.5. Comparación con el protocolo original 64
5.1.6. Otros análisis 66
5.2. Evaluación Beaglebone Black 74

6. Conclusiones, futuros trabajos y publicaciones **79**
6.1. Conclusiones 79
6.2. Publicaciones y otras contribuciones 80
6.3. Futuros trabajos 80

Bibliografía **83**

Índice de figuras

1.1. Aglomeraciones en las carreteras 2

2.1. Arquitectura Red Vehicular 6
2.2. Comunicaciones V2V y V2I 6
2.3. Esquema general EA 8
2.4. Taxonomía de los protocolos en redes VANETs 10
2.5. OLSR parámetros 16
2.6. Función proximidad en cruce 19
2.7. Ejemplo de envío de paquetes en la ruta 20
2.8. Nodo emisor envía un paquete seleccionado la ruta más óptima 25
2.9. Cambio de modo desconectividad a conectividad 25
2.10. Diagrama flujo protocolo SRR 26

3.1. Arquitectura NS3 32
3.2. Beaglebone Black 36
3.3. ROBOCape 37
3.4. Descripción de conectores, módulos, IC's y distribución 38

4.1. Diagrama de bloques del ecosistema propuesto 40
4.2. Área de la Universidad Politécnica de Valencia 41
4.3. Área simulada de la Universidad Politécnica de Valencia 42
4.4. Ejemplo simulación 42
4.5. Ejemplo .nod.xml 42
4.6. Ejemplo .edg.xml 42
4.7. Ejemplo .net.xml 43
4.8. Ejemplo .flow.xml 43
4.9. Ejemplo .rou.xml 43
4.10. Ejemplo .cfg 43
4.11. Ejemplo simulación 44
4.12. Fichero de traza 44
4.13. Diagrama de flujo de SUMO 45
4.14. Distancia entre Rx y Tx 45
4.15. Distribución LogNormal 47
4.16. Diagrama UML del protocolo implementado 48
4.17. Cabecera mensaje HELLO 48
4.18. Cabecera mensaje ACK 50
4.19. Cabecera mensaje STATUS 50
4.20. Diagrama flujo protocolo propuesto 53

4.21. Esquema comunicaciones con niveles inferiores de la pila IP 54
4.22. Jerarquía librería BlackLib 55
4.23. Nodo implementados en ROS 56

5.1. Impacto Densidad variando el número de nodos del escenario, calculando el retardo medio (arriba) y Packet Delivery Ratio (abajo) 63
5.2. Impacto Channel Shadowing variando σ, calculando el retardo medio (arriba) y Packet Delivery Ratio (abajo) 64
5.3. Impacto Velocidad variando la velocidad de movimiento de nodos del escenario, calculando el retardo medio (arriba) y Packet Delivery Ratio (abajo) 65
5.4. Comparación PDR protocolo propuesto (arriba) VS SRR (abajo) 66
5.5. Comparación Delay protocolo propuesto (arriba) VS SRR (abajo) 67
5.6. Comparación Overhead protocolo propuesto (arriba) VS SRR (abajo) . . 68
5.7. Comparación PDR paquetes de control habilitados (arriba) VS deshabilitados (abajo) 69
5.8. Comparación Throughput sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo) 70
5.9. Comparación PDR sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo) 71
5.10. Comparación Delay sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo) 72
5.11. Comparación Densidad sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo) 73
5.12. Comparación retardo sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo) 74
5.13. Comparación Velocidad sin paquetes de control con 1 destino (abajo) VS destino todos los nodos de la red (abajo) 75
5.14. Ruta 1 obtenida por el GPS 76
5.15. Ruta 2 obtenida por el GPS 77
5.16. Datos obtenidos del GPS 77
5.17. Datos publicados de la IMU 77
5.18. en la imagen de la izquierda se puede observar el robot Bioloid con la cámara y la BeagleBoneBlack, en la imagen de la derecha se puede observar una instantánea del robot compitiendo. 78

Índice de cuadros

2.1. Esquema protocolo ERBA 11
2.2. Tabla estimada de densidad 19
2.3. Reglas *Fuzzy logic* 27

4.1. Exponentes de pérdida de trayectoria para diferentes entornos 46

5.1. Tabla resumen experimentos 61

Capítulo 1

Introducción

En la última década las redes inalámbricas han experimentado un enorme avance en cuanto a prestaciones y nuevas tecnologías. Este avance a llevado a que este tipo de sistemas se estén incorporando en entornos que hace unos años la idea no era concebible, como por ejemplo dotar a los vehículos de sensores inalámbricos que sean capaces de comunicarse con el resto de vehículos del entorno. Esto es lo que se conoce como "Vehicular Ad-Hoc Networks" (VANETs). Este tipo de redes se caracterizan porque los nodos son vehículos, aunque pueda existir una infraestructura inalámbrica fija. Los vehículos dotados con esta tecnología forman una red en pleno movimiento y se mueven de forma arbitraria, así incrementando la seguridad en las carreteras y reduciendo el impacto medioambiental. La evolución que han experimentado los sistemas electrónicos en los últimos años tanto en sus prestaciones como reducción de precios, ha llevado a que tecnologías que era inconcebibles, hoy estén al alcance de nuestras manos. Y con los enormes avances que han sufrido los sistemas informáticos en los últimos en cuanto a capacidad de cómputo y nuevas tecnologías, como son los sistemas expertos e inteligentes.

Con la evolución de esta tecnología, tanto en el ámbito doméstico como profesional, se ha llegado a situaciones realmente algo desagradables, como es el caso de que las personas estén pendientes de sus dispositivos móviles y poco atentos a la conducción, y como consecuencia existe una alta tasa de siniestralidad. Asimismo, la elevada flota de automóviles que existen hoy en día ha llevado a que se formen aglomeraciones y esto es un peligro tanto para el medio ambiente como para el personal Figura 1.1. A raíz de esta serie de problemas y acompañados de otros existentes como los simples despistes, unos grupos de personas han decidido emplear sus energías en el desarrollo en nuevas tecnologías y técnicas que se puedan incorporar en los vehículos para reducir este problema. Para solventar este rompecabezas se están empleando técnicas y tecnologías están basadas en las redes de sensores inalámbricos, sistemas inteligentes y expertos

donde los vehículos han de ser capaces de tomar decisiones en caso de que los humanos no hayan sido capaces de tomarlas.

FIGURA 1.1: Aglomeraciones en las carreteras

1.1. Objetivos del Trabajo Final de Máster

Tras lo presentado anteriormente los objetivos de este trabajo consiste en:

- Aprendizaje de las herramientas empleadas para la simulación de redes vehiculares.
- Investigar en protocolos de redes vehiculares y técnicas de IA que sean aplicadas a vehículos o robots móviles. De esta forma que puedan tomar decisiones coordinadas de forma distribuida a través de la información suministrada por la WSN.
- Desarrollar un modelo de enrutamiento basado en redes vehiculares.
- Llevar a cabo la correspondiente experimentación mediante campañas de simulación.
- Analizar los resultados obtenidos tras el desarrollo y validarlo.

En primer lugar se realizará una evaluación de las propuestas mediante simulación y posteriormente sobre nodos reales que serán adecuados en función de los resultados obtenidos en las campañas de simulación, estudiándose finalmente su aplicación en la transferencia de información entre los vehículos ligeros y robots móviles.

1.2. Estructura del TFM

Este trabajo se estructura de la siguiente forma:
En el capítulo dos se realizará una descripción de una de las tecnologías que se emplea

en los *Sistemas de Transporte Inteligente (ITS)*, esta tecnología es *Wireless Access in Vehicular Environments (WAVE)*. Después de exponer esta primera parte se procederá a explicar una serie de técnicas de *Inteligencia Artificial* que se están aplicando en los últimos años al análisis de protocolos de enrutamiento en las redes vehiculares. Ya con esto presentado, se centrará el interés en analizar diferentes tipos de protocolos para este tipo de redes.

Seguidamente, en el capítulo tres se introducirán las herramientas empleadas en este trabajo, para ello se realizará una breve descripción del simulador de redes basado en eventos discretos NS-3 y se describirá brevemente su arquitectura. También se detallará el simulador SUMO, que es un simulador para generar escenarios de movilidad y definir comportamientos de vehículos. Se especificará el entorno donde se va a realizar la implementación en equipos reales, esta herramienta es ROS y finalmente se introducirá los dispositivos electrónicos que se emplearán, la tarjeta BeableBoneBlack y la cape ROBOCape.

A continuación en el capítulo cuatro, se expondrá el trabajo realizado con las tres herramientas introducidas anteriormente y el ecosistema de simulación que ha resultado con la incorporación de un nuevo protocolo.

En el capítulo cinco se presentará la experimentación y análisis de los resultados obtenidos, comparando los resultados obtenidos con el protocolo propuesto. Además, se describirá la aplicación realizada sobre la tarjeta BeableBoneBlack mostrando que puede ser aplicable a cualquier sistema robotizado móvil.

Finalmente, en el capítulo seis, se expondrán las conclusiones obtenidas de todo el trabajo, se enunciarán las publicaciones y colaboraciones que se han realizado durante la elaboración de este trabajo, y finalmente se expondrá los futuros trabajos que darán lugar al desarrollo de la tesis doctoral.

Capítulo 2

Estado del arte

2.1. Tecnología inalámbrica

Hoy en día se conocen aplicaciones para *Intelligent Transport Systems* (ITS) [1]. Estos sistemas se dividen en dos grupos:

- Aplicaciones que aportan seguridad.
- Aplicaciones sin seguridad.

La principal diferencia entre estos dos grupos es que la primera está orientada a que los vehículos se comuniquen entre sí para así poder intercambiarse mensajes frente a accidentes, atascos, estado de las carreteras, y el segundo está más orientado al ocio.

Esto hace que se elaboren una serie protocolos enunciados en 2.3.1 que están basados en el estándar 802.11. Dentro de este estándar existen diversas variantes como es el caso de los estándares mas comúnmente conocidos 802.11a/b/g, pero existe el estándar 802.11p que incorpora la arquitectura necesaria para **Wireless Access In Vehicular Environments (WAVE)**, que es para sistemas de comunicaciones vehiculares, este opera en la banda de 5.8-5.9 GHz con un ancho de banda de 75 MHz.

Dentro de este modo de comunicación inalámbrica se encuentran las conexiones **Vehicular-To-Vehicular (V2V)** [1, 2] el cual permite realizar envíos de información entre distintos vehículos y el otro tipo de conexiones es **Vehicular-To-Infraestructure (V2I)**, que permite a los nodos comunicarse con infraestructuras que tienen una localización fija y están dotados de Gateways (GW) Figura 2.1.

Los sistemas de comunicaciones V2V están libres de usar una infraestructura y tan solo se comunican con los nodos que están dentro de su alcance, para que esto se produzca

los vehículos han de estar dotados de sistemas **On Board Unit (OBUs)**, así creando redes ad-hoc, esto permite que exista una alta movilidad dinámica en la red y por tanto requiere un descubrimiento, mantenimiento y recuperación de la ruta.

En cuanto a los sistemas de comunicaciones V2I involucran los sistemas OBUs y **Road-side Unit (RSUs)**. Esto permite que las OBUs sean capaces de intercambiar mensajes con las RSUs y estas mediante **Access Points (APs)** sean capaces conectarse a internet a través de **Internet Gateways (IGWs)**, así adquiriendo un direccionamiento IP global. Esto permite que la gestión de la movilidad esté garantizada y los paquetes de los nodos sean alcanzables. En la Figura 2.2 se muestra el esquema de conexiones.

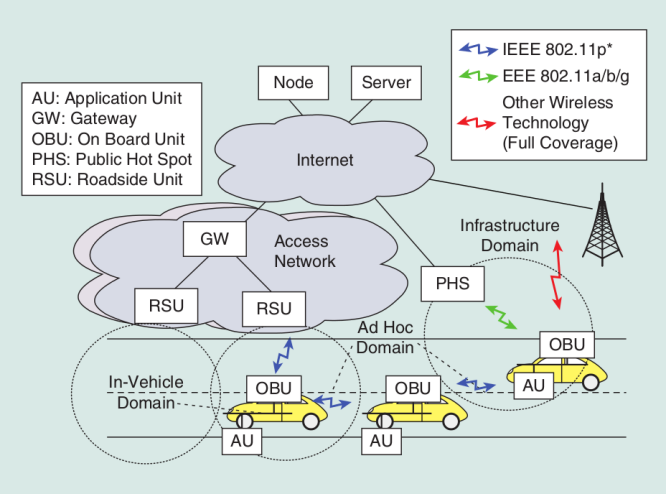


FIGURA 2.1: Arquitectura Red Vehicular

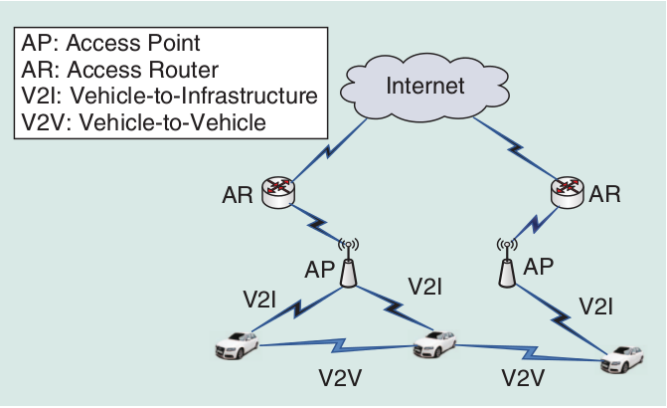


FIGURA 2.2: Comunicaciones V2V y V2I

En este sistema de comunicaciones el direccionamiento IP emplea una topología jerárquica plana debido al rápido cambio que se produce en la misma. Esta topología ha de ser compatible con la existente en modo cableada. Por ello se emplean direcciones IPv6 autoconfigurables. Debido a la elevada movilidad de los nodos se producen fallos de conexión, por ello surge la necesidad de emplear mallas de redes inalámbricas híbridas. Este tipo de redes han de presentar una serie de características, como por ejemplo; cada

nodo ha de ser capaz de conectarse a internet; todos los nodos de estas redes han de tener un prefijo de red para conectarse a internet; han de tener soporte para diferentes gateways para así moverse libremente; las conexiones en la capa de transporte han de estar constante abiertas. Y el intercambio de IP se puede dividir en tres fases [1]:

- **Handover initiation**: fase inicial reconoce la necesidad de la entrega debido a diversos factores.
- **Handover decision**: fase intermedia que permite o deniega la transmisión
- **Handover execution**: fase final que se da después de que se cumplan las dos fases anteriores y permite la conmutación real de una sesión activa de una estación base o punto de acceso a otros.

Además existen diversos tipos de intercambios de IP:

- **Horizontal**: en el cual los puntos de acceso se empleas la misma tecnología.
- **Vertical**: en el cual existen diferentes tecnologías para el acceso.

2.2. Técnicas de IA para VANETs

En los últimos años los grupos de investigación en VANETs han visto la necesidad de aplicar nuevas técnicas para el análisis de diversos parámetros que conforman las configuraciones de los protocolos de encaminamiento que están desarrollando. Con tal de obtener las configuraciones óptimas y automáticas de estos parámetros se están empleando diversas técnicas de inteligencia artificial. Los métodos que se están usando son **Computación Evolutiva (EC)** [3, 4], técnicas de **Fuzzy Logic** [5] y **redes neuronales**. Seguidamente se realizará una breve descripción de los técnicas enunciadas anteriormente.

La EC es una rama de la Computación y la Inteligencia Artificial que comprende métodos de búsqueda y aprendizaje automatizado inspirados en los mecanismos de la evolución natural. Se han propuesto diversos enfoques de la EC: **Estrategias Evolutivas**, **Algoritmos Genéticos (GA)**, **Programación Genética**, **Clasificadores Genéticos**, **Algoritmos Metaheurísticos** y **Algoritmos Evolutivos Paralelos (PEA)** entre otros. Esta serie de métodos se les denomina de manera colectiva como **Algoritmos Evolutivos (EA)** [6], entre los cuales los más conocidos son probablemente los GA. Estos algoritmos han sido aplicados exitosamente en la resolución de problemas en distintas ramas de la ingeniería, diseño, industria, economía y ciencias naturales. Debido

a que EA emulan a la evolución natural cabe destacar que comprenden una serie de características:

- Una representación o codificación de las soluciones potenciales al problema bajo estudio.
- Una población (conjunto de individuos) de estas soluciones potenciales.
- Mecanismos para generar nuevos individuos o soluciones potenciales al problema estudiado, a partir de los miembros de la población actual (los denominados operadores de mutación y recombinación).
- Una función de desempeño o evaluación (del inglés fitness function) que determina la calidad de los individuos en la población en su capacidad de resolver el problema bajo estudio.
- Un método de selección que otorgue mayores oportunidades de sobrevivir a las buenas soluciones.

En la Figura 2.3 se ilustra el esquema general de un algoritmo evolutivo.

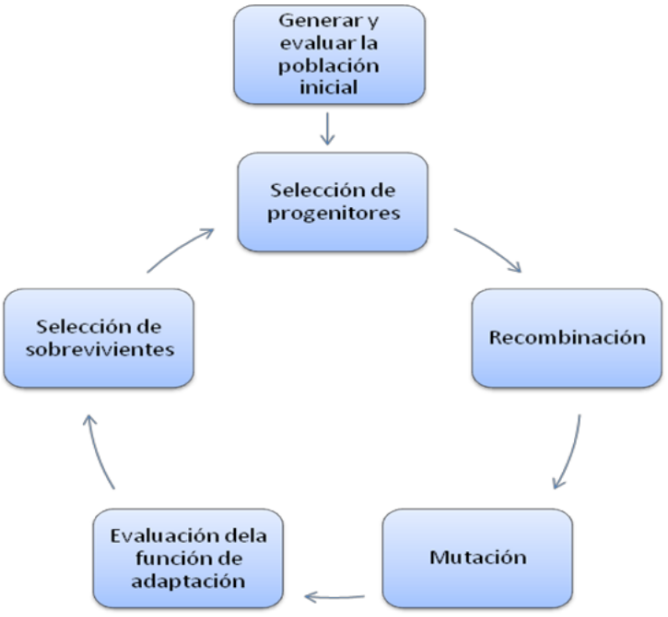


FIGURA 2.3: Esquema general EA

Otra técnica IA que están empleando los grupos de investigación es la *Fuzzy Logic*. Este tipo de sistema toma un número de entradas relativas a un estudio previo, y con las cuales se obtiene una salida. Así, adaptándose lo mejor posible al mundo real. Estos sistemas tienen un funcionamiento basado en reglas de la forma *SI (antecedentes) Entonces (Consecuente)* donde su salida es también un conjunto difuso.

2.3. Estudio protocolos

2.3.1. Clasificación Protocolos

Existen dos grandes grupos donde se pueden englobar los protocolos de encaminamiento para las redes inalámbricas vehiculares [7]. Estos dos grupos son los basados en **Enrutamiento por Direccionamiento** y **Enrutamiento Geográfico** como se puede observar en la Figura 2.4.

El primer grupo nombrado se divide en dos categorías: proactivos y reactivos. El primero se caracteriza por mantener siempre actualizada la información de direccionamiento entre los nodos, aquí podemos encontrar los protocolos OLSR y DSDV entre otros. Este tipo de protocolos presenta un problema que es la elevada sobrecarga que existe cuando la topología de la red cambia y por tanto existe la probabilidad de que la entrega de paquetes no exista debido a la rotura de los enlaces. La segunda subclase se caracteriza por obtener la información de enrutamiento en el momento que se va a enviar un paquete y no antes de eso como ocurre en la primera subcategoría presentada. Esto hace que exista una sobrecarga menor, pero por contra existe una elevada latencia durante la entrega de paquetes. Los protocolos típicos de estudio en este subgrupo son AODV y DRS.

El segundo grupo basado en métodos de geolocalización parte de que el nodo o vehículo conoce su ubicación actual gracias a un dispositivo GPS y puede tomar decisiones para el reenvío de paquetes de forma directa, gracias a que conoce la ubicación de sus vecinos. A la hora describir este tipo de protocolos, se hace una diferencia entre dos tipos: los protocolos basados en **Packet buffering based GPS** y **Non-Packet buffering based GPS**. En el primer subconjunto introduce GeOpps, VADDS. En cuanto a los basados en Non-Packet buffering based GPS se realiza una subdivisión en dos subclases: **Connectionless routing** y **Connection-oriented routing protocols**. Dentro de esta primera subclase están los protocolos CBF, GDBF. La segunda subcategoría se divide en dos tipos de de protocolos: **Trajectories-based Geographical Routing Protocols** y **Source and Map based Routing**. En el primer tipo se pueden agrupar los protocolos; Trayectory-Based Fordwarding (TBF) y Motion Vector Schema (MoVe). En el segundo grupo están los protocolos GPSR, GSR, GPCRGpsrJ+, CAR, SADV, A-STAR, VCLCR, LOUVRE, RBVT, GyTAR, TO-GO, y Fuzzy Logic-based Route Selection in VANET. Cogiendo lo mejor de estos dos tipos de protocolos, hay autores que han propuesto protocolos híbridos como puede ser el protocolo Stability and Reliability aware Routing (SRR).

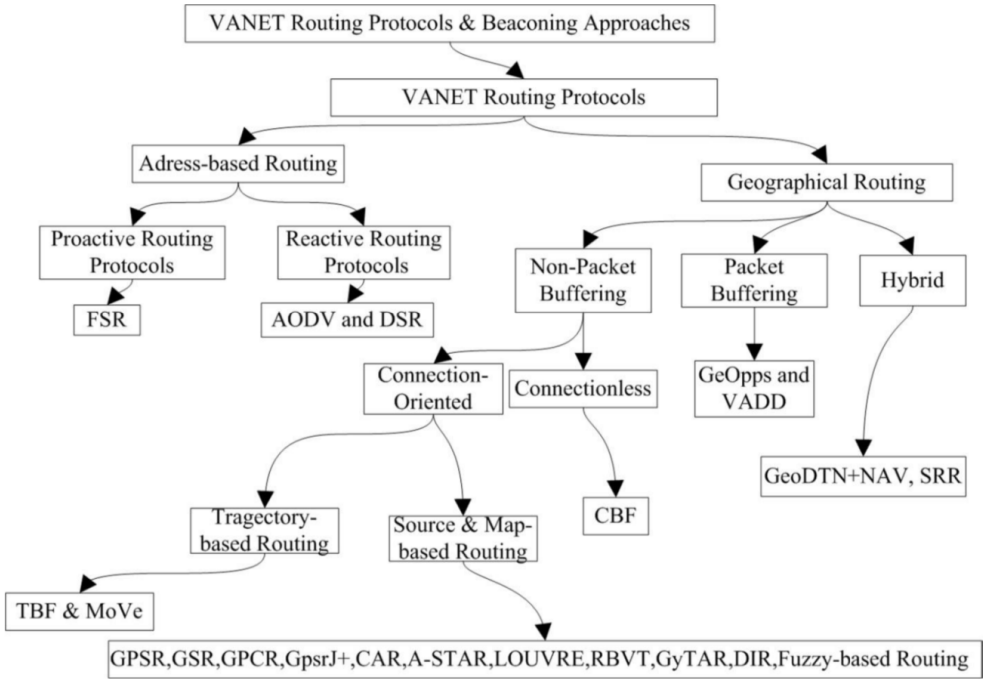


FIGURA 2.4: Taxonomía de los protocolos en redes VANETs

El tipo de conexiones presentado en 2.1 conlleva una serie de problemas cuando existe una elevada movilidad de los vehículos, puntos de acceso estáticos, retardos en el envío de paquetes, links que pueden fallar. De ahí que surjan protocolos de movilidad de redes como por ejemplo NEMO BS [1] y los analizados a lo largo de esta sección.*NEMO Basic Support* es un protocolo definido en (RFC 3963) que es capaz de gestionar la movilidad de una red entera basada en IPv6. Es capaz de asegurar la continuidad de una sesión para todos los nodos móviles que existen en la red, y la actualización del punto de acceso a internet. De esta forma garantiza la conectividad y accesibilidad de la red móvil conforme esta va evolucionando.

2.3.2. ERBA

En este apartado se presenta el protocolo de enrutamiento ERBA [8], el cual surge del proyecto Shangai Grid. Sus fuerte es que es eficiente con el consumo energético y emplea las tendencias de movimiento de los vehículos. La principal característica de este proyecto es adquirir las tendencias de una diferente variedad de conductores, y monitorizar el transporte público. Este protocolo es capaz de distinguir entre los conductores de autobuses y vehículos privados. Esta distinción se realiza porque los conductores de autobuses siempre realizan la misma ruta, por ejemplo, el autobús inicia su recorrido desde la estación y su recorrido siempre acaba en el mismo lugar. Por otro lado, la ruta de los conductores privados es aleatoria.

ERBA está probado solamente en entornos urbanos, y cada vehículo esta equipado con distintos dispositivos, GPS, senores, etc. y emplea la notación mostrada en la tabla 2.1 para su funcionamiento.

Notations	Explication
SV	the source vehicle
DV	the destination vehicle
CD	the current direction
ND	the next direction
VI	the distance between the vehicle location to the intersection
VT	the vehicle type
REQ	the route request
REP	the route replay
ERR	the route error message
TTL	the time-to-live of a REQ
LRS	the link reliable significance

Cuadro 2.1: Esquema protocolo ERBA

Este protocolo emplea técnicas de tablas de enrutamiento para localizar las rutas y autorizar a los vehículos y los nodos a intercambiar mensajes a través de los vecinos hacía los nodos que se alcanzan de forma directa intentando buscar la mejor ruta entre los vehículos. El intercambio de paquetes se realiza de la siguiente forma:

En primer lugar, el vehículo origen (SD) envía un mensaje al vehículo destino (DV) y DV devuelve un mensaje. Si DV es un nodo vecino es que existe una comunicación y SV realiza una inundación de mensajes de petición (REQ). Cuando realiza esta inundación, los nodos cercanos actualizan su tabla de vecinos. El mensaje REQ está compuesto de varios campos (bits): nodo de origen, nodo destino, TTL, y secuencia de números de ID, CD, ND, VI, VT y LRS. El último campo sirve como marca de tiempo, por lo que, los nodos pueden mantener actualizada su información con los otros nodos. Cuando un nodo envía un mensaje, este aumenta su número de secuencia.

En segundo lugar, cuando un nodo recibe un mensaje REQ pueden ocurrir dos cosas:

1. El nodo es el destino o conoce el nodo destino, en el primer caso casi envia un mensaje REP al nodo fuente, en otro caso estima el LRS y envía el mensaje REP.
2. El nodo realiza inundaciones con mensajes REQ.

En tercer lugar, el mensaje REP se envía a lo largo de los vehículos que son almacenados en el paquete REQ. Así, la trayectoria de la ruta más fiable está configurada.

Finalmente, con el fin de mejorar la estabilidad de ruta, incorpora un mecanismo preventivo para descubrir nuevas rutas antes de ser vencido en los enlaces de la ruta.

ERBA es capaz de acordarse de CD, ND, VI y VT del nodo que está cerca de la siguiente intersección para así actualizar la tabla de vecinos. De este modo predice el movimiento de tendencia y otra información de los conductores. Existe un algoritmo que determina la calidad de la ruta de los vehículos vecinos y evalúa el rango en una escala de cinco. Además propone un mecanismo de ruta preventivo con el fin de mejorar la estabilidad y fiabilidad del enrutamiento, esto ocurre cuando el vehículo está pasando por una intersección, e inmediatamente la tabla de los vecinos es actualizada con la nueva información. También incorpora un mensaje de error para ajustar el nodo de la ruta. Cuando el nodo recibe un mensaje de error este elimina de la tabla de enrutamiento la información errónea del nodo.

Los precursores de este protocolo ha realizado una serie de experimentos para validarlo. Estos tests están elaborados en la ciudad de Shangai y consisten en comparar ERBA con los protocolos AODV y ROMSGP. Para ello han seleccionado unos rangos horarios comprendidos entre las 10:00 - 11:00 y 15:00 - 16:00. Donde cada autobús está equipado con un GPS y sensores. Se han centrado en dos tipos de vehículos -autobuses y vehículos privados-. De cada bus se recogen sus rutas, estaciones y la hora de salida de internet. Los automóviles privados se ven afectados por la topología de las calles, semáforos, señales y otros factores. En ERBA se emplea el protocolo IEEE 802.11p y el modelo de propagación empleado es el Nakagami.

Las métricas escogidas son las siguientes:

1. Función de Probabilidad de densidad (PDF), esta función es una métrica básica de ERBA.
2. El retardo End-End denota la media del tiempo, es el tiempo que tarda en deliberar los paquetes de datos.
3. Existe un retardo entre las dos comunicaciones

Los resultados obtenidos por los autores son los siguientes. En el primer experimento el PDF de los nodos ha sido comparado con el número de total de nodos, y como conclusión se ha obtenido se ha obtenido que ERBA es mejor que los otros dos, esto es debido a que el rango de cobertura es mayor que en los otros protocolos. En el experimento que se estudia el retardo End-End, el retardo ofrecido por ERBA es mejor que el los otros dos escogidos, este resultado es debido a que la ruta seleccionada es confiable. En el último experimento que está basado en los enlaces fiables cercanos el protocolo ROMSGP obtiene un menor valor porcentual que ERBA y AODV.

Concluyendo, el protocolo ERBA obtiene mejores resultados que AODV y ROMSGP en los dos primeros experimentos, pero en el número de enlaces fiables es mejor ROMSGP.

Éste se ha presentado como un protocolo nuevo, en el cual aparecen las palabras protocolo de enrutamiento de energía eficiente, pero en los experimentos llevados a cabo no se ha analizado el consumo energético.

2.3.3. Hybrid Location Ad-Hoc Routing (HLAR)

Se analizan una serie protocolos y los diferentes problemas de escalabilidad que presentan los protocolos de enrutamiento tratados. Los autores han propuesto una solución basada en *Hybrid Location Ad-Hoc Routing (HLAR)* [9], este protocolo está diseñado para optimizar el rendimiento de escalabilidad.

Las principales características de HLAR es que combina con una serie de modificaciones con el protocolo AODV y con técnicas de protocolos de enrutamiento *greedy-forwarding geographical*. La modificación del protocolo AODV esta basada en contar métricas diseñadas para obtener la mejor calidad en la ruta [10]. Con esta modificación, un vehículo intermedio es capaz de reparar los links rotos con un bajo consumo energético y enlaces compartidos. Periódicamente, un *beacon* está continuamente enviando su ID. Con la ayuda de este ID, los nodos vecinos están identificados y les ayuda a construir una tabla de vecinos para coordinarse.

El protocolo se comporta de la siguiente forma: inicia la ruta haciendo un descubrimiento bajo demanda, en caso que el nodo origen no encuentra una ruta al nodo destino, el vehículo origen envía sus coordenadas en un paquete de solicitud de ruta y busca el destino más próximo en su tabla de vecino. Cuando hay coches cerca un mensaje RREQ se envía al nodo origen. En otro caso, se realiza una inundación a todos los nodos con paquetes RREQ. Este paquete incluye un campo de vida TTL. Este campo se decrementa cada vez que el mensaje da un salto de un nodo a otro mientras no encuentra el nodo objetivo. Cuando TTL llega a cero el paquete con el ID se elimina de la tabla de vecinos.

Se realizan una serie de estudios, estos están basados en la escalabilidad y la densidad de HLAR comparándolo con una versión extendida del protocolo AODV, conocida como AODV-EXT. Considerando que la escalabilidad es “la habilidad del protocolo de enrutamiento para mantener la sobrecarga del ratio de la ruta con un mínimo de de carga en el tráfico, como parámetro que describe el incremento de la red (MTL)”. MTL es el mínimo de ancho de banda requerido para reenviar paquetes a través de las cortas distancias que están disponibles. Se describen varios parámetros que aparen en las ecuaciones, por ejemplo λ_i, que representa el ratio de generación, movilidad, el ratio de sobrecarga O_N que indica el factor de escalabilidad del protocolo entre otros. Este factor se analiza como sigue:

1. *initiation overhead rate* O_i: se requiere para iniciar las rutas.
2. *maintenance overhead rate* O_m: se requiere para mantener las rutas.
3. *beacon overhead rate* O_b: se requiere para estimar la calidad de los enlaces y construir la tabla de vecinos.

En cuanto al análisis de la densidad, empleando HLAR el crecimiento según la sobrecarga en el ratio de enrutamiento es constante y por el contrario con AODV es exponencial. Se ha diseñado una métrica para cuando se desconozca la ubicación por GPS. Esta métrica está basada en triangularización. Para así, ayudar en la construcción de la tabla de vecinos:

- **Optimización de HLAR en presencia de error de localización**: Los vehículos están constantemente enviando paquetes con su localización. Se pretende emplear HLAR para enrutar los paquetes, de esta forma se pretende obtener una optimización. Al existir una imprecisión en la localización, el envío de los paquetes puede tener una ruta demasiado larga, así, con esto se consigue una sobrecarga mínima de enrutamiento. Aplicando esta optimización en términos generales en [11] se demuestra que con errores de ubicación en la transmisión el protocolo es óptimo.
- **HLAR y otras métricas de rendimiento**: Aunque se han centrado en la escalabilidad de HLAR sobre AODV. Son conscientes de que otros parámetros son importantes, el tiempo de entrega de los paquetes, y el retardo que existe en la entrega de paquetes en la autopista. Una creciente densidad puede provocar una sobrecarga en el enrutamiento.
- **Otros protocolos de enrutamiento**: Existe una multitud de protocolos para VANETs que sirven para obtener la información relativa a la posición, la diferencia que existe es que los protocolos usan unas métricas distintas para obtener la posición del coche más cercano. Una desventaja de estos protocolos es que no permiten la inclusión de errores de la posición durante la etapa de análisis. Se ha simulado con un protocolo de posición geográfica, Closer Mean Protocol (CMP) para así, comparar con HLAR el error existente en la localización.

Con el aumento de la densidad de nodos se observa un crecimiento lineal en cuanto al protocolo de enrutamiento HLAR. Y en cuanto al protocolo CMP conforme aumenta la densidad la sobrecarga de enrutamiento aumenta exponencialmente. Esto sucede porque las rutas que se crean con CMP son las largas que con HLAR, por tanto se puede concluir que CMP no es escalable.

En este artículo no se ha realizado un análisis de consumo energético por parte de los nodos que conforman el experimento. Como se ha comentado en dicho paper solo se han centrado en la sobrecarga de enrutamiento, y no se ha calculado el tiempo que tarda un paquete en enviarse y recibirse.

2.3.4. Intelligent OLSR

En este trabajo [3] se propone realizar una serie de estrategias de optimización, de las cuales se propone que un determinada cantidad de algoritmos metahurísticos este acoplados con el simulador ns-2 [12], para así ajustar los parámetros del protocolo OLSR o cualquiera de los existentes. Los resultados de los algoritmos metaheurísticos se calculan con MALLBA [13], para así después incorporar los resultados como parámetros de OLSR. Y como existen una serie de problemas de movilidad a la hora de generarla con el ns-2 se soluciona empleando el simulador SUMO. Estas simulaciones están basadas en diversos escenarios de Málaga.

Uno de los principales motivos de optimización de los parámetros del protocolo OLSR es que en su definición estándar no son "concisos". Uno de los objetivos de realizar esta optimización es que sirva para ayudar a los expertos a identificar posibles fallos en las comunicaciones. Para ellos se centra la atención en los tres términos mas usados para la QoS. Estos son; la entrega del ratio del paquete; la carga de nivel de enrutamiento; y el retardo que existe en la entrega de un paquete en conexiones extremo a extremos.

Para poder realizar las optimizaciones se emplean cuatro métodos que buscan la solución óptima. Primero se obtienen los parámetros ya optimizados y a continuación se lanza la simulación en el ns-2 para obtener la información global sobre el PDR, NRL, y el E2ED. El objetivo es maximizar el PDR y minimizar NRL y E2ED. Al realizar esto ha surgido un problema, y es que OLSR delibera un gran número de paquetes de gestión, y esto hace que el NRL se incremente y el PDR empeore. De ahí que en [3] se aconseje usar unos valores en la función de conste de comunicación para evitar ese problema.

Se realizan una serie de análisis los algoritmos metaheurísticos antes de utilizarlos para realizar las optimizaciones a OLSR. Con estos análisis lo que se ha realizado a continuación es una comparación, para así después con los resultados obtenidos, poderlos aplicar a los parámetros de OLSR. Para el indicador PDR, empleando los 4 algoritmos y comparándolo con el algoritmo RAND ha tenido una tasa completa de optimización. En cambio para las simulaciones hechas en [14] la tasa de optimización a sido inferior, esto es debido a los paquetes administrativos generan una congestión en la red. En cuanto al indicador NRL, los resultados obtenidos son similares entre sí a diferencia de los resultados obtenidos por el algoritmo GA, que son los peores, en cambio los resultados

obtenidos con DE son los mejores. La importancia de reducir la métrica NLR es porque con una baja tasa se puede reducir posibles fallos provocados por la congestión de paquetes. En cuanto a la métrica E2ED los valores obtenidos en la mayoría de los casos son peores que en la propuesta de [14] aunque la optimización realizada con el algoritmo GA para este caso obtiene el mejor resultado , estos resultados se deben a la baja carga de enrutamiento. Se realizan un total de nueve [3] simulaciones sobre la ciudad de Málaga, divididas en tres escenarios. Cada escenario tiene una determinada densidad de nodos. De esta forma se pueden evaluar las métricas propuestas en escenarios simulados. Dependiendo de la densidad de vehículos los resultados obtenidos son variables, y por tanto a grandes rasgos se podría decir que ningún algoritmo metaheurístico de los propuestos es el mejor. Aunque la configuración automática de las métricas propuestas para estudio, han obtenido buenos resultados en la QoS.

Es importante conocer la congestión de la red en VANETS y la QoS para evitar grandes consumos energéticos y pérdidas de paquetes. Con las métricas propuestas se ha evaluado la generación de paquetes y la congestión que producen, y por tanto los fallos que pueden provocar, pero hubiera sido interesante que con el estudio realizado se hubiera analizado el consumo energético que se necesita durante el envío y recepción de los paquetes.

2.3.5. Fast energy-aware OLSR

Siguiendo con el uso de algoritmos metaheurísticos para la optimización de protocolos de enrutamiento en [4] se aborda el problema del consumo energético que hay en el protocolo OLSR. Se presenta una solución basada en una rápida metodología para buscar configuraciones de energía eficiente de OLSR empleando algoritmos evolutivos paralelos. El trabajo introduce la tecnología que se integra con las tecnologías de redes ad-hoc. Y las limitaciones que ofrecen los protocolos VANETs basados en protocolos MANETs. Además, el artículo concluye que con el fin de reducir el consumo de energía de varios parámetros en el protocolo OLSR han ser modificados. Estos parámetros se modifican con un algoritmo evolutivo paralelo que sirve para aplicar una configuración automática de los parámetros de OLSR. Pero al aplicar estas técnicas multihilo con OLSR presenta una serie de desventajas y es que está gobernado por ocho parámetros Figura 2.5.

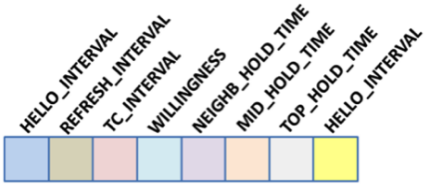


FIGURA 2.5: OLSR parámetros

Cuando se trata de definir la función de conveniencia es importante definir un mecanismo de optimización de GA y seleccionar la población a analizar. [E. Alba et al []] plantean una optimización de las comunicaciones consciente de la energía, donde el tema principal de la función de aptitud es el consumo de energía. Sin embargo, hay varios inconvenientes, por ejemplo, si se produce una reducción del exceso de energía de la energía puede causar pérdidas en la QoS las comunicaciones. La función de conveniencia que se describe en el documento se basa en la métrica PDR. Para esta propuesta se definen dos ecuaciones. La primera ecuación es válida cuando la degradación PDR es menor que 15 %, en el caso de degradación es superior a 15 % se aplica la función de conveniencia penalizada. Este articulo resuelve varios aspectos presentes en enfoques previos [15]. Otro inconveniente es que GA sufre una baja diversidad de la población y un estancamiento temprano, los autores proponen algunas variaciones basadas en la inicialización de la función canónica y operadores de mutación. Para inicializar, la población se necesita una distribución uniforme con el fin de obtener patrones uniformes. Con este enfoque es posible obtener el genero para estimar el valor aleatorio de la distribución, α^p. En el paso de la recombinación del PGA se emplea la recombinación de la aritmética clásica para recombinar los problemas con valores reales. En la última etapa, la mutación introduce nueva información genética para la diversificación de la población del PGA. Esta información consiste en añadir problema de la información relacionada, por lo que, la nueva información genética se genera aleatoriamente. Esta información afecta a los siguientes parámetros: HELLO_INTERVAL y NEIGHB_HOLD_TIME.

En la fase de análisis experimental primero se describen los escenarios y software empleados para la simulación. Después de esto, se han analizado una serie de parámetros para obtener los mejores resultados acerca de la probabilidad de cruce (p_C) y la probabilidad de mutación (p_M) en los PGA. Una vez empieza la simulación, se crean tres escenarios con nueve combinaciones para aplicar a los valores candidatos de los parámetros p_C y p_M. Después de esto, se obtienen una serie de valores, estos son la media, la desviación estándar relativa y los mejores valores de conveniencia, de esta forma es posible obtener la media de la energía, PDR y la diferencia media de la energía y PDR con la configuración establecida por la configuración estándar. Comparando estos valores se obtiene que los mejores resultados son con $p_C = 0.7$ y $p_M = 0.25$, estos valores son contrastados con el consumo energético y PDR de la configuración de RFC 3626.

Se han realizado 3 experimentos con el algoritmo PGA. Los mejor implementación con la cual se consigue una reducción de la energía ha resultado ser un test con 24 individuos. Así obteniendo una reducción del 30 % usando una computación maestro-esclavo. Las ventajas que presenta la configuración del experimento es: generación del control de tráfico mejor que con la configuración estándar de OLSR; el consumo del nodo respecto a la configuración estándar disminuye significativamente; los nodos tienen una actuación

superior funcionando como *Multipoing Relays* (MPR). Pero esta configuración presenta una desventaja: los tiempos de validación son muy altos, por tanto tarda más en detectar que un link ha fallado.

Aplicando algoritmos paralelos se consigue una mayor eficiencia y velocidad respecto al algoritmo secuencial, por este motivo se aplica esta técnica a los experimentos anteriormente realizados, obteniendo unos valores eficiencia superiores. Como los resultados obtenidos hay que validarlos, esta se realizará analizando la energía, principalmente: energía de transmisión y recepción, energía total y energía total por vehículo. Así, reduciendo significativamente el consumo respecto a la configuración con los parámetros estándar de OLSR. Además, se analiza la QoS, centrándose en el tiempo de envió, la sobrecarga de la red y el número de saltos hasta llegar al destino. Obteniéndose una considerable reducción en la entrega de paquetes y sobrecarga en la red respecto a la configuración estándar de OLSR.

No se ha tenido en cuenta el uso de un dispositivo de geolocalización a la hora de enviar los paquetes a los nodos vecinos.

2.3.6. Junction-based Adaptive Reactive Routing (JARR)

Se presenta el protocolo Junction-based Adaptive Reactive Routing (JARR) [16]. Este es un protocolo de enrutamiento reactivo adaptativo multisalto, que tiene en cuenta la dirección y la velocidad a la que viajan los nodos, además de conocer en qué condiciones se encuentra la red. Esta última característica es una ventaja frente a otros protocolos de enrutamiento, como por ejemplo GPSR. El protocolo no requiere de una infraestructura externa y por tanto soporta una alta escalabilidad en entornos de VANETs. En la implementación del protocolo, el paquete que sirve para establecer el enrutamiento lleva implementados diversos modos, así de esta forma puede adaptarse en redes dispersas y redes con alta densidad de nodos. Obtiene el camino más corto para así comparar con otros caminos y obtener la ruta más optima. Los vehículos tienen un alcance de unos 250 metros usando dispositivos wireless. En comparación con otros protocolos [17, 18], este modifica estrategias de transporte y reenvío para reducir los intervalos. Además de implementar un mecanismo adaptativo de beacons para ofrecer altos ratios en redes dispersas.

JARR funciona de la siguiente forma: el paquete se enruta desde un cruce, y se va retransmitiendo hasta llegar a otro cruce, en ese momento se toman las decisiones oportunas para obtener el camino óptimo a la ruta que seguirá el nodo. A la hora de calcular la ruta óptima se tiene en cuenta a la velocidad y la ruta que siguen los vehículos y su posición actual. De esta forma se obtiene los pesos para calcular dicha ruta.

La forma obtener la ruta óptima se realiza empleando grafos dirigidos, donde los vértices indican el número de cruces que hay en conectados en la carretera y las aristas representan el número el número de direcciones de las carreteras conectadas a los cruces. El peso de las aristas indica la distancia entre dos cruce y además puede indicar el posible retardo que haya entre dos cruces próximos. La forma de obtener la ruta óptima es mediante el algoritmo de Dijkstra. En [16] se ha definido la siguiente función para determinar la proximidad de un nodo a un cruce Figura 2.6:

Di = Distance from current node to the target junction
Dj = Distance from next hop to the target junction
Dk = Dj / Di
Vi = Velocity vector
α = Weight factor for position of node
β = Weight factor for direction of travel
Wp = Calculated weighted score for node on a path

$$Wp = \alpha\,(1 - Dk) + \beta\,(Vi)$$

FIGURA 2.6: Función proximidad en cruce

Gracias a la información ofrecida por el mecanismo de los *beacons* el protocolo JARR es un protocolo adaptativo. Ya que se puede conocer la posición, velocidad, dirección de trayecto de los nodos vecinos que se encuentran dentro del rango de transmisión.

Para conocer la densidad en la red se realiza una estimación mediante el radio de *beacons* y a la velocidad a la que viajan los nodos, ver tabla 2.2.

Velocity of Node (m/s)	**Beaconing (s)**	**Estimated Density of Path**
Slow (0 - 13)	Slow	Dense
Slow	Average	Average
Slow	Fast	Sparse
Average (¿13 - ¡20)	Slow	Average
Average	Average	Average
Average	Fast	Sparse
Fast (20 -25)	Slow	Average
Fast	Average	Average
Fast	Fast	Sparse

CUADRO 2.2: Tabla estimada de densidad

En carreteras donde existan carriles con las dos direcciones el vehículo que retransmite el paquete lo enviará a los vehículos que vayan en su misma dirección Figura 2.7.

En la selección del modo de enrutamiento, inicialmente un nodo selecciona como ruta más óptima el camino a la primera intersección con la que se encuentra. A continuación debe de seleccionar mediante el algoritmo una ruta adecuada para retransmitir los datos. El camino se selecciona mediante la información estimada de la densidad. En caso de

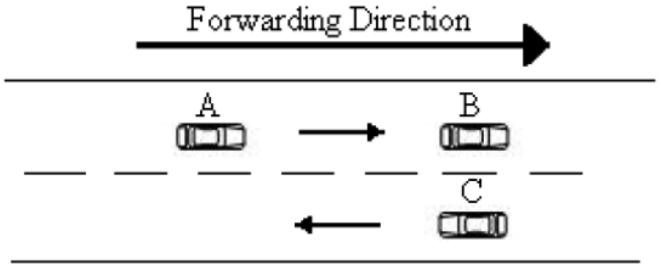


FIGURA 2.7: Ejemplo de envío de paquetes en la ruta

que no existan nodos en una ruta determinada esta tendrá prioridad baja, pero al mismo tiempo que se va descubriendo los cruces serán considerados en el algoritmo.

En las simulaciones realizadas y comparándose con el protocolo GPSR, conforme aumenta el número de conexiones el protocolo JARR obtiene mejores resultados que GPSR, este último los resultados obtenidos para el PDR desciende casi a la mitad, mientras que JARR obtiene valores próximos a 0.8. Evaluando la métrica de la sobrecarga de paquetes, mientras el número de nodos es muy disperso el número de beacons generados por GPSR es muy inferior a JARR, pero en cuanto el número de nodos va aumentado, y por tanto la densidad, la sobrecarga de JARR es mucho inferior a GPSR.

La cobertura de alcance que ofrece este protocolo para los nodos es reducido, aproximadamente de unos 250 metros. Por otro lado no se tiene en cuenta el consumo energético que se produce en los envíos de los paquetes.

2.3.7. Reliable Geocast Routing Protocol

Debido a la limitaciónes ofrecidas por el protocolo de enrutamiento AODV, se proponen dos mecanismos [19] como: eficiencia en la transmisión de datos al destinatario y usar la inundación dentro del perímetro. Para ello se realizan una serie de suposiciones para la mejora del protocolo AODV. Estas suposiciones son: Todos los vehículos están dotados de GPS, y por tanto conocen su posición. Además se realiza un broadcast a todos los vecinos; Se asume que los bloques de datos se generan para codificar los paquetes que son lo suficientemente grandes para identificar y codificar paquetes despreciables; Existe como mínimo un vehículo dentro del rango de recepción; Cuando se realiza un broadcast en una región si un coche lo recibe, todos los coches pueden recibir ese mensaje; El estándar *Dedicated Short Range Communication* (DSRC) se utiliza para establecer el rango de transmisión de vehículos; Todos los nodos se mueven a velocidad constante.

Cuando el reenvían mensajes, el origen realiza un broadcast usando el protocolo AODV, y cuando llega al destinatario correcto, este contesta al mensaje con un mensaje unicast informando al origen de cuál es su ruta. Asumiendo que los nodos conocen su ubicación

la zona de destino se conoce como *Zone of Relevance* (ZoR) y se asume que es una zona rectangular donde las esquinas que forman los cruces están definidas por coordenadas. El mensaje de cabecera incluye el ID la zona de destino. En cuanto a la región de reenvío se le llama *Zone of Forwarding* (ZoF) y se usa para el reenvío de paquetes.

Cuando existe una tasa baja en la entrega de mensajes y un pobre QoS se emplea un generador de números aleatorios para seleccionar un subconjunto de bloques y se "guardan" juntos para generar un paquete. Este proceso se repite para generar un número K de bloques que serán recuperados por el destinatario. En la propuesta realizada para la mejora de AODV el nodo origen envía paquetes codificados de baja tasa de codificación hasta que el destino le envía un mensaje de ACK, esto indica que el número de paquetes recibidos es suficiente. Una vez el origen recibe este mensaje es capaz de enviar nuevos paquetes de datos.

El mecanismo de recuperación rápida ante fallos consiste en que el nodo origen recibe paquetes desde el destino hasta que recibe un paquete de error. Entonces con la información recibida en este paquete se crea una nueva ruta mediante el proceso de descubrimiento.

En las suposiciones realizadas anteriormente el nodo origen solo acepta el primer mensaje recibido y transmite por la ruta creada los datos. Con tal de reducir la perdida de paquetes y QoS se permite la aceptación de mensajes desde diversas rutas. De esta forma también se reduce el retardo de los mensajes hacia el destino y los mensajes de error no se mantienen para conocer la ruta.

Para realizar las simulaciones en el entorno propuesto se emplean cuatro métricas para obtener los resultados, estas son, número de paquetes recibidos en la ZoR, la media de retardo de los mensajes, número de paquetes perdidos y la sobrecarga. Los valores obtenidos se compraran con el protocolo IVG. En las simulaciones se observa que con las dos mejoras propuestas al existir diversas rutas para recibir mensajes se reduce el tiempo de retardo en comparación con una única ruta. También se reduce la sobrecarga en la ruta, esto implica que la tasa de recuperación ante fallos sea menor. Los métodos propuestos con enrutamiento unicast han reducido la sobrecarga de la red en comparación con IVG y con la línea-base definidas. En cuanto al número de paquetes perdidos por la propuesta es casi despreciable por parte del IVG tiene un gran número de paquetes perdidos. E retardo por los métodos propuestos es menor y por tanto en un instante de tiempo habrá más paquetes recibidos.

2.3.8. Routing algorithm for Dense Traffic Density Vehicular Networks (BTDAR)

(BTDAR) [2] es un protocolo que tiene como objetivo, entregar paquetes de forma fiable y eficiente en entornos con poca y mucha densidad de vehículos. En el modelo inter-vehicular definido se obtiene el espacio medio que hay entre dos vehículos. Además de la velocidad entre dos vehículos, se definen una serie de ecuaciones para el cálculo de estas métricas y así que los vehículos sean capaces de transmitirse los datos e incluso encolar los datos recibidos en un buffer.

Se definen dos tipos de protocolos para la densidad y la escasez de nodos, en los cuales se tiene en cuenta la velocidad relativa, tiempo de vida de las comunicaciones y la probabilidad de espera ante un exceso de tiempo de espera en una cola. Se presenta el protocolo *BTDAR-R: Routing algorithm for Dense Traffic Density Vehicular Networks* el cual se emplea cuando exista tráfico denso en los entornos. Este protocolo es capaz de mantener la calidad del servicio en la entrega de paquetes en la ruta óptima seleccionada. Además, es capaz de mantener reenvíos con multisalto. Los nodos candidatos son capaces de reenviar la posición de los nodos buscando la ruta óptima, esta ruta se obtiene buscando el camino con menor tiempo de entrega entre dos nodos satisfaciendo la estabilidad y la calidad del servicio. Se presenta también el protocolo *BTDAR-P: Routing algorithm for Sparse Traffic Density Vehicular Networks* este se empleará cuando la densidad de tráfico sea escasa. La conectividad se verá afectada por intermitencia provocada por la escasez de vehículos. El protocolo está dotado de un buffer que será capaz de reenviar y recibir datos cuando exista conectividad con algún nodo vecino.

BTDAR se compara con otros protocolos basados en enrutamiento por geolocalización: GPSI, IBR, JARR. Simulaciones realizadas sobre la densidad de nodos existentes en un entorno muestran que conforme aumenta la densidad, todos tienden a decrementar el tiempo de entrega de los paquetes (PDD), esto se debe a que los protocolos rápidamente obtienen la mejor ruta para retransmitir. Cabe notar que BTDAR-R tiene los mejores resultados conforme aumenta el número de nodos. En cambio BTDAR-P cuando existen pocos nodos obtienen los mejores resultados. En cuanto a las simulaciones realizadas sobre el tiempo de entrega de los paquetes conforme aumenta la velocidad, se obtiene que todos los protocolos comparados tienden a incrementar el tiempo de retardo, esto es debido a los patrones de movilidad que se han considerado. Se puede observar que en BTDAR-R los tiempos de entrega son inferiores al resto de protocolos presentados. En cuanto al PDR analizado, BTDAR-P y BTDAR-R tienen una tasa muy elevada de paquetes entregados sobre el resto de protocolos analizados. En cuanto a la densidad de nodos es muy escasa BTDAR-P tiene los mejores resultados, seguidos de BTDAR-R, pero en cuanto la densidad aumenta el número de nodos recibidos por BTDAR-R va

aumentado, y BTDAR-P decrementa ligeramente el rendimiento. En cuanto a la simulación realizada para obtener el PDR según los patrones de movilidad establecidos, se obtiene como resultados que BTDAR-R y BTDAR-P tienen un alto porcentaje de entregas en comparación con el resto, aun aumentando la velocidad de los nodos. En cuanto al control de Bytes transmitidos por Bytes de datos entregados, según los resultados obtenidos para la densidad de nodos el protocolo JARR se mantiene constate porque en su definición no tiene un modo predictivo. En cuanto a GPSI tiene un aumento considerable en la sobrecarga de control, esto es debido al constante cambio de un modo "ávido.[a] un modo predictivo. En cuanto a BTAR conforme aumenta la densidad va mostrando una sobrecarga en la red, pero aun así es mucho menor que al resto de protocolos, por tanto los resultados son mejores. En los test realizados sobre el aumento de la velocidad BTDAR continúa teniendo los mejores resultados respecto a los otros protocolos.

Finalmente los análisis realizados sobre el número de paquetes transmitidos por paquete entregado BTDAR tiene una mejor eficiencia de acceso al canal que el resto de propuestas, esto se debe a que BTDAR entrega los datos a través de rutas estables.

2.3.9. Intersection-Based Routing Protocol (IBR)

Se presenta el protocolo IBR [20], el cual está basado en la estrategia de transporte y reenvío. Una de sus características es evitar que el paquete llegue a los coches que van en dirección opuesta al vehículo origen.

IBR se asume que los coches están dotados de un GPS. Cada vehículo ha de mantener su segmento de carretera en la tabla, este segmento incluye la siguiente información: ID del segmento; número de vehículos en el segmento de carretera; y ultima actualización de la tabla.

IBR sigue los principios básicos implicados en el modelo de carretera, además de añadir información de los cruces y carretera recta. Adopta el método .[a]vido.[en] las carreteras rectas. Cuando llega a una intersección la transmisión dependerá de la dirección de enrutamiento del paquete y dirección en que vayan los vehículos incluidos el siguiente reenvío y "libertador". Cuando se reenvía el paquete al coche que va en la misma dirección el tiempo de vida del paquete se decrementa. Si hay una alta densidad de nodos la tasa de paquetes perdidos y el retardo en la entrega aumenta.

Ya que el protocolo solo transmite en la dirección en la cual va el vehículo, se establecen cuatro condiciones de encaminamiento. Estas son:

1. Hay al menos un vehículo en el siguiente segmento de carretera del paquete. Es decir, cuando un vehículo se aproxima a una intersección, este intercambia su segmento de carretera con los vehículos que estén cerca.

2. No hay vecinos en la intersección. Cuando esta condición ocurre, el vehículo que ha pasado por la intersección lleva los paquetes hasta que encuentra un vehículo con el que intercambiarlos, y así poder replanificar la ruta.

3. La dirección de movimiento de un vehículo es diferente a la dirección de transferencia del paquete, pero no hay vehículos en el siguiente segmento de carretera, pero sí que hay vecinos en la intersección. Cuando ocurre esta condición es porque no se encuentran nodos que vayan en la misma dirección que el paquete, y el vehículo bloquea el paquete hasta que encuentra un vehículo que va e la misma dirección y a este se le asigna la mejor ruta.

4. Los paquetes llegan tarde a la intersección. Cuando esta condición sucede un paquete se reenvía al siguiente segmento de carretera y al segmento de carretera por el cual va. Por tanto se realiza un duplicado de paquetes.

2.3.10. Stability and Reliability aware Routing (SRR)

Se describe un nuevo protocolo de enrutamiento basado por geolocalización, además incluye en su implementación algoritmos de lógica difusa. Este protocolo se llama *Stability and Reliability aware Routing (SRR)* [5]. Debido a la incorporación de un sistema basado en *Fuzzy Logic* este protocolo es capaz de tomar decisiones más rápidamente que el resto de protocolos, estas decisiones se toman a partir de la distancia y la dirección de los vehículos, para así crear un paquete que será retransmitido a los nodos que están dentro del radio de cobertura del nodo origen. Por otra parte lleva un mecanismo de decisión local para el caso que la red sea dispersa y no se pueda conectar a ningún nodo vecino, de esta forma el protocolo puede cambiar de modo SRR a modo de encolado y al revés. En este último modo enunciado se utiliza el mecanismo de *carry-and-fordward* para el caso en que el nodo no esté conectado a ninguna red, y se usara como métrica de entrada la densidad de tráfico.

SRR está adoptado para sistemas de comunicación basados en V2V. La información se transmite desde un vehículo origen a otro destino mediante *multi-hop*. El vehículo al llevar un GPS incorporado da la posibilidad de proporcionar su posición al resto de vehículos que están dentro de la zona de alcance. Un vehículo puede conocer la dirección y la ubicación de los vehículos vecinos. Esto se puede conocer ya que el nodo constantemente está haciendo *boradcasting* con mensajes HELLO. Esto facilita la zona de conocimiento de un vehículo.

El protocolo tiene dos modos de funcionamiento Figura 2.10:

1. **Packet Routing in Connected Vehicular Sceneario** Figura 2.8. En este modo el vehículo origen está dentro de un convoy de coches y desea enviar datos a un destinatario. El origen consulta su tabla de vecinos y selecciona el vehículo que mejor ruta presente hacia el destino. Esta ruta se obtiene mediante técnicas Fuzzy. Y entonces envía el dato a ese vehículo. Y este proceso se realiza hasta llegar al vehículo destino.

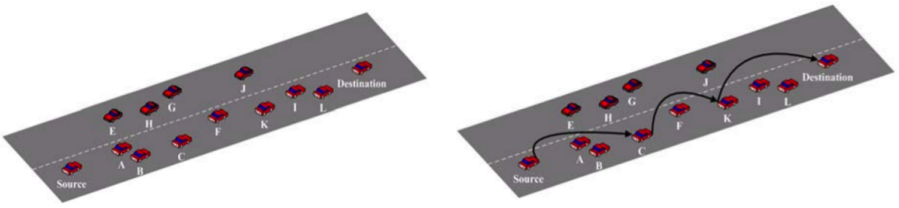


FIGURA 2.8: Nodo emisor envía un paquete seleccionado la ruta más óptima

2. **Tackling Packet Loss in Dis-connected Vehicular Sceneario** Figura 2.9. Cuando el vehículo entra en este modo es debido a que no hay ningún vehículo vecino cerca. Entonces empieza a guardarse en una cache la información que enviara a un vehículo. En el momento que este vehículo es alcanzado por un convoy de coches este cambiara de modo y se conectara y seleccionara la mejor ruta para enviar el paquete al destino.

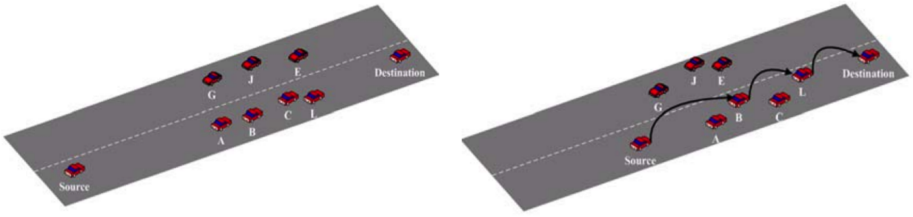


FIGURA 2.9: Cambio de modo desconectividad a conectividad

Las métricas que se emplean en este protocolo ya se han nombrado anteriormente. Pero faltaba comentar porque se seleccionan la distancia y la dirección: La primera de ellas, se conoce aplicando el teorema de Pitágoras, para conocer la distancia que hay entre dos nodos. Una vez conocida la distancia de los nodos vecinos con el origen se decide que los paquetes se han de enviar a los vehículos que estén a una distancia media. De esta forma se evita el fallo de transmisión con los nodos que está en el límite del rango de alcance, y en el caso de los nodos más cercanos se evita la sobrecarga de paquetes. En cuanto a la dirección, los vehículos solo se pueden mover en dos direcciones por limitaciones de

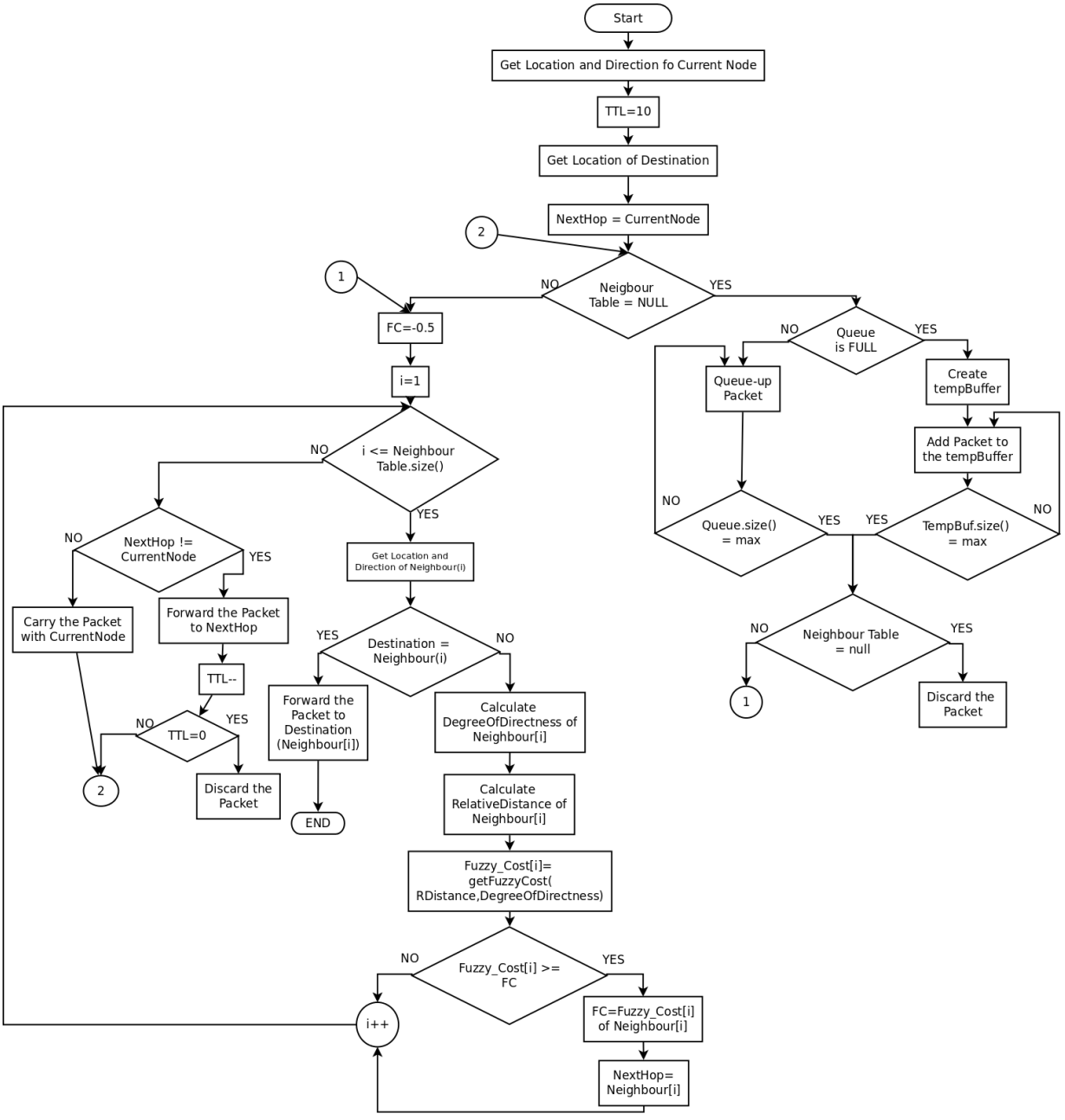


FIGURA 2.10: Diagrama flujo protocolo SRR

las carreteras. Pero las carreteras no son rectas, ya que tienen curvaturas, por tanto los vectores de dirección de los vehículos no son siempre paralelos unos a otros y se necesita conocer siempre que ángulo existe entre un nodo y otro de la misma dirección.

Para seleccionar las entradas y salidas para la fuzzificación se han declarado unas funciones llamadas, Less Directed, Mid Directed and More Directed. Estas funciones se emplearán dependiendo del ángulo de dirección que exista entre el nodo origen y el nodo vecino. El valor de este ángulo estará comprendido en [-1,1] ya que el coseno oscila entre estos valores. En cuanto a la distancia entre un nodo y otro depende del radio de cobertura y se crean tres funciones: Far, Intermediate, Close. Tras esto, surge la necesidad de normalizar los valores. Esta normalización resulta de dividir la distancia con el radio de alcance, de forma como se ilustra en [5]. Con eso se obtiene las reglas para la estructura de la lógica difusa del protocolo 2.3:

	IF		THEN
Rule	RDistance	Degree.Directness	Fuzzy-Cost
1	Far	Less Directed	Low
2	Far	Mid Directed	Medium
3	Far	More Directed	High
4	Intermediate	Less Directed	Medium
5	Intermediate	Mid Directed	High
6	Intermediate	More Directed	V. High
7	Close	Less Directed	V. Low
8	Close	Mid Directed	Low
9	Close	More Directed	Medium

CUADRO 2.3: Reglas *Fuzzy logic*

En cuanto a las simulaciones realizadas, se ha usado el simulador JiST/SWANS [21]. Dentro del simulador se integró la implementación del sistema de inferencia de lógica fuzzy. Y se han realizado diversas pruebas para finalmente compararse con el protocolo GPCR. En los primeros resultados analizados se estudia el impacto del "solapamiento" del canal. En la cual se demuestra el efecto sobre la variación del link de error sobre canales inalámbricos y trafico CBR comparado con el PDR, retardo medio de paquete y sobrecarga de paquetes. Conforme la velocidad de transferencia entre nodos ha ido aumentando hasta 72 Kbps, el valor del PDR ha ido descendiendo tal como se muestran en las gráficas. En cuanto a la media del retardo de entrega de los paquetes según se ha incrementado la velocidad de transferencia el retardo medio ha ido incrementándose, y lo mismo ha ocurrido con el control de sobrecarga de los paquetes. En esta simulación se ha comparado con diversos valores del canal de perdida inalámbrica entre vehículos.

En cuanto a la simulación realizada con el impacto de la densidad del tráfico de los vehículos. También se demuestra el efecto sobre la variación del link de error sobre

canales inalámbricos y trafico CBR comparado con el PDR, retardo medio de paquete y sobrecarga de paquetes. Según ha ido aumentando la velocidad de transferencia el PDR ha ido descendiendo, y en cuanto al retardo medio de entrega de los paquetes y el control de sobrecarga de los paquetes han ido aumentado, las pruebas se han realizado con diversas densidades de vehículos para comprobar cuando un escenario tiene nodos dispersos y existe una gran densidad de vehículos.

En la simulación realizada, para analizar el impacto de la velocidad de los vehículos, se vuelve a fijar en las mismas métricas que en los casos anteriores. En la gráfica sobre el PDR se puede observar que con el aumento de la velocidad de los vehículos, los valores analizados se producen una decadencia exitosa con la entrega de los paquetes. En cuanto a la media de retardo de entrega y control de sobrecarga las gráficas muestran que los valores se incrementan, esto se debe a que la actualización de las tablas de vecinos es incierta.

En el siguiente estudio realizado para este protocolo se ha estudiado el impacto de número de nodos emisores. En la gráfica en la cual se analiza el PDR se puede observar que cuando aumenta el rango de emisión aumenta, en el caso de que hayan 5 y 7 nodos los resultados para esta métrica se aumentan, pero sin embargo ocurre el efecto contrario cuando el número de nodos emisores es mayor que el valor de PDR es menor y con el aumento de la distancia de cobertura decae. En cuanto al tiempo medio de entrega conforme aumenta el rango de cobertura el tiempo aumenta, esto se debe a que se introducen retardos extras en el protocolo. Finalmente, en el control de carga de la red con un bajo número de nodos emisores se observa un débil incremento conforme aumenta el radio de alcance. Pero cuando el número de nodos emisores es superior conforme aumenta el radio de cobertura la tendencia de carga también aumenta, esto se debe a que conforme aumenta el radio de cobertura se produce una mayor contención en el nivel MAC.

Finalmente se realiza una comparación con los protocolos GPCR y DSR, para comparar el rendimiento de SRR respecto a estos dos. En cuanto a la tasa de paquetes entregados SRR tienen un mayor número de paquetes entregados, esto se debe a que cuando transmite paquetes el origen envía el paquete al nodo que ocupa la posición del medio respecto al destino y al origen, y en cambio DSR presenta unos valores muy bajos, esto se debe a que este protocolo no está preparado para las elevadas velocidades de este tipo de nodos. En cuanto al tiempo medio de entrega GPCR presenta muy buenos resultados de entrega respecto a SRR y a DSR, esto se debe a que a que SRR realiza reenvíos con pequeños saltos hacia el destino. En cuanto al control de carga SRR presenta una similitud con GPCR, esto se debe a que comparte una fórmula similar para calcular la sobrecarga de la red.

En cuanto a la comparación de SRR con los otros dos protocolos para el número de nodos en un escenario. Cuando el escenario presenta dispersión con los nodos que forman la red, SRR presenta una tasa mayor de entrega de paquetes respecto a GPCR y DSR. En cambio cuando existe una gran densidad de nodos a velocidades bajas de transferencia GPCR presenta mejores resultados que SRR, pero conforme aumenta la velocidad de transferencia GPCR decae fuertemente y sin embargo SRR tiene una leve caída en el número de paquetes que se entregan. En cuanto al tiempo medio de entrega, cuando los nodos están esparcidos GPCR presenta un menor retardo en la entrega, pero conforme aumenta la velocidad de transferencia, GPCR y SRR presentan resultados similares. Sin embargo cuando hay una densidad considerable de nodos a bajas velocidades la diferencia entre SRR y GPCR es mínima, pero a altas velocidades SRR tiene una tasa menor de tiempo de entrega. En cuanto al control de carga de la red, cuando existe una enorme densidad de nodos los tres protocolos presentan una enorme carga de la red, y aunque GPCR y SRR presentan resultados similares, SRR es ligeramente mejor, esto se debe a la estrategias basada en el reenvíos de múltiples paquetes de métrica para reducir el control en la capa MAC.

En la evaluación de este protocolo hubiera sido interesante que se realizara un estudio de la energía que se consume en la transmisión de paquetes de un nodo a otro. Y aunque los resultados en el control de la carga de la red cuando existe una gran densidad de nodos es mejor que la de los otros dos protocolos, podría ser interesante añadir un mecanismo balanceamiento y carga de la red.

Capítulo 3

Herramientas Empleadas

En este capítulo se describen las herramientas empleadas en este trabajo. Primero se describe el simulador NS-3, posteriormente se describe SUMO y seguidamente el middleware ROS y finalmente la tarjeta BeagleBoneBlack y la cape ROBOCape.

3.1. NS-3

El simulador ns-3 [22] es un un simulador de redes basado en eventos discretos el cual se emplea principalmente para investigación y educación. El proyecto ns-3 se inició en el año 2006, y es un proyecto de desarrollo de código abierto bajo licencia GNU GPLv2. Ns-3 proporciona una plataforma de simulación implementada principalmente en C++, aunque algunas estructuras del mismo se encuentran escritas en Python, además es compatible con Linux, Mac OS y FreeBSD.

Este simulador proporciona modelos para trabajar con paquetes de datos y redes, además de incorporar modelos de movilidad y propagación. Todo ello forma un motor de simulación para que los usuarios puedan realizar sus experimentos. De esta forma se pueden realizar estudios sobre el comportamiento de los sistemas cuando no se pueden disponer de todos los dispositivos reales necesarios para realizar experimentos o el comportamiento de las redes.

Al estar organizado como una librería y escrito en C++ permite realizar enlaces estáticos y dinámicos. Por otra parte, permite la definición de nuevas topologías, escenarios y modelos de red, que pueden ser fácilmente integrados y depurados en el núcleo del simulador depurados. Todo ello se organiza en módulos, de esta forma facilita la independencia entre los diferentes niveles que conforma la arquitectura de ns-3 Figura 3.1.

Otras de las características que posee este simulador es que se pueden realizar simulaciones con varios equipos a la vez, y validar las simulaciones realizadas con dispositivos reales, ya que permite el uso de tecnología real.

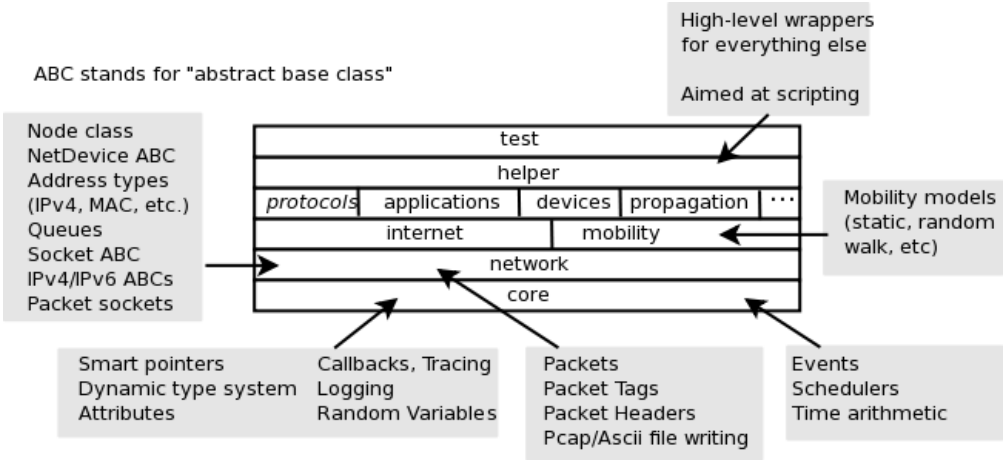


FIGURA 3.1: Arquitectura NS3

3.1.1. Arquitectura del simulador

Como se ha comentado ya, ns-3 está escrito principalmente en C++, y además, está organizado en módulos, de la misma forma que se puede apreciar en la Figura 3.1. El núcleo del simulador ofrece las facilidades al programador, el cual está formado por *Smart pointers*, *Callbackas*, un sistema de agregación de objetos y un sistema para la generación de trazas y atributos de los objetos en tiempo de ejecución, entre otros. Los paquetes son los objetos fundamentales en las redes y están implementados en el módulo *network*, y es donde se definen los dispositivos de red, colas y sockets entre otros. El módulo *internet* proporciona una API para poder emplearlo. En el módulo de *mobility* se describen diversos modelos que se emplean para el análisis del comportamiento de las redes en distintos entornos de movilidad. Finalmente otro módulo importante en ns-3 es *helper*, que es el que sirve para realizar las llamadas desde el fichero de script y con el cual se facilita el acceso al uso de los distintos modelos que existen en el simulador.

3.2. Simulator Urban MObility - SUMO

El simulador SUMO (Simulation of Urban MObility) [23] es un simulador de tráfico que facilita la evaluación de cambios en la infraestructura. Este simulador es una herramienta de simulación abierta y libre que está disponible desde el año 2001. Las principales características que ofrece esta herramienta son las siguientes:

- Simulación microscópica - vehículos, peatones y el transporte público se modelan explícitamente.
- Interacción Online - control de la simulación con TraCI.
- Simulación de tráfico multimodal.
- Se puede generar o importar el comportamiento de los semáforos.
- No existe limitación alguna en cuanto al tamaño de red y número de simulaciones a lanzar.
- Soporta los formatos: OpenStreetMap, VISUM, VISSIM, NavTeq.
- Está implementado en C++ y usa solo librerías portables

Este simulador incorpora una serie de paquetes, estos componentes son:

- SUMO: Simulación mediante línea de comandos.
- GUISIM: Simulación mediante interfaz gráfica.
- NETCONVERT: Importar la red de carreteras.
- NETGEN: Generador abstracto de redes.
- OD2TRIPS: Convertir de matrices O/D a viajes.
- JTRROUTER: Generador de rutas basado en intersecciones.
- DUAROUTER: Generador de rutas basado en una asignación dinámica por el usuario.
- DFROUTER: Generador de rutas con uso de detección de datos.
- MAROUTER: Asignación de usuario macroscópica basado en funciones de capacidad.

3.3. Robot Operating System - ROS

"Robot Operating System", o más comúnmente conocido como ROS [24]. Es un framework para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo en un clúster heterogéneo. ROS se desarrolló originalmente en 2007 bajo el nombre de **switchyard** por el Laboratorio de Inteligencia Artificial de Stanford para dar soporte al proyecto del Robot con Inteligencia Artificial de Stanford.

Aunque en su nombre aparezcan las palabras "Sistema Operativo" no es tal cual un OS. Ya que funciona sobre un sistema Linux. Es más bien una infraestructura de desarrollo, despliegue y ejecución de sistemas robotizados. En el cual existe una gran variedad de paquetes con software para que se pueda realizar un rápido desarrollo sobre un robot, además de solventar muchos problemas, que otros Frameworks de Desarrollo de Robots, RSF, no solventan. Las soluciones que solventa son las que se detallan a continuación:

- Abstracción de Hardware (HAL) y reutilización e integración de robots y dispositivos encapsulando estos tras interfaces estables y manteniendo las diferencias en archivos de configuración.
- Algoritmos de robótica con bajo acoplamiento. Es decir, desde algoritmos de bajo nivel de control, cinemática, SLAM, etc. Hasta algoritmos de alto nivel como puedan ser los de planificación o aprendizaje. Además de otros tantos más específicos como puede ser que los robots se conecten a una red eléctrica cuando lo necesiten o incluso coger y dejar objetos.
- Está provisto de un mecanismo de comunicaciones (middleware) distribuido entre los nodos del sistema robotizado. Un nodo es cualquier pieza de software del sistema (desde un algoritmo SLAM hasta un driver para el manejo de un motor). El objetivo de este sistema es doble: Encapsulado/abstracción/reutilización de software; Ubicuidad, es decir, independencia de donde este nodo está localizado (un sistema robotizado puede tener muchos procesadores). Estos nodos se comunican entre ellos mediante mecanismos de paso de mensajes RPC o Publish/Subscribe, Service Lookup, etc. Permite crear arquitecturas P2P de componentes robotizados distribuidos.
- Permite realizar simulaciones de sistemas robóticos con dinámicas de sólidos rígidos.
- Herramientas de desarrollo, despliegue y monitorización de sistemas robóticos.

La diferencia existente entre ROS y otros RSF es que ha logrado agrupar muchas de las mejores características de otros proyectos, como pueda ser Player/Stage/Gazebo, Yarp, Orocos, Carmen, Orca, OpenRave, Open-RTM, así dando una solución integral y muy uniforme al problema de desarrollo de sistemas robóticos. Además de ser un sistema multilenguaje, peer2peer, orientado a herramientas, ligero y OpenSource.

Desde que surgió hasta la fecha su popularidad ha ido en aumento y esto ha sido debido a que se está empleando en diferentes ámbitos profesionales: Científico, Investigación, Divulgación.

3.4. BeagleBone Black

La tarjeta BeagleBoard [25] es producida por la empresa Texas Instruments en asociación con Digi-Key y Newark element14. Surge a mediados de 2008 como el primer micro ordenador de bajo coste del mercado. Desde su aparición ha ido evolucionando mediante modelos nuevos que incorporan ciertas mejoras en rendimiento y conectividad. Hoy en día el modelo más reciente es el BeagleBone Black rev c, cuya revisión consta de mayo de 2014. Este modelo ofrece un rendimiento y una capacidad de cómputo que pocas tarjetas de precios similares son capaces de superar. La BeagleBone Black tiene un precio de menos de 50€, y posee un procesador ARM Cortex-A8 a 1000MHz que permite realizar 2000 MIPS, una GPU PowerVR SGX530 a 200MHz que es capaz de procesar 20MPoligonos/s, viene con 512MB de RAM de tipo DDR3 a 800Mhz, una conexión Ethernet, salida de vídeo y audio micro-HDMI, un puerto USB, una ranura para tarjeta microSD y dos filas de 46 pines de entrada o salida y dos núcleos programables de tiempo real. Hay que tener en cuenta que el voltaje de entrada de la tarjeta es de 5V y que sin conectar nada, la tarjeta sola consume entre 210 y 460mA.

Otra de las ventajas más destacables de la BeagleBone Black es la conectividad que ofrece. Dispone de un total de 92 pines reconfigurables que incluyen hasta cuatro puertos serie diferentes, la generación de ocho PWM independientes, cuatro temporizadores, un bus CAN, siete entradas analógicas en base a 1.8V con una resolución del conversor AD de 12 bits, varias salidas de alimentación a 5 y 3.3V, dos conexiones SPI, dos puertos I2C para su interconexión con cualquier tipo de hardware compatible, además de 65 entradas y salidas digitales reconfigurables.

Las extensiones para las tarjetas Beagleboard se denominan capes y hay más de 40 modelos diferentes. Del mismo modo que con el resto de tarjetas, su funcionalidad es otorgar mayor facilidad de interconexión y/o dotar de nuevas características a la tarjeta.

La BeagleBone Black soporta diversos sistemas operativos basados en la arquitectura ARM Linux como Fedora, Ubuntu u openSUSE, pero también tiene soporte para Android e incluso para Windows Embedded. Estos sistemas se ejecutan desde la microSD aunque cabe mencionar que la tarjeta dispone de un kernel de auto arranque con Linux Debian en su memoria eMMC de 2GB que se ejecuta cuando no se inserta ninguna microSD.

Las opciones de programación en la BeagleBoard son prácticamente las mismas que las de un sistema operativo basado en una arquitectura x86. Se pueden utilizar lenguajes como C, C++, Java, Python, JavaScript, Android e incluso también Matlab-Simulink.

Empleando la distribución de Linux Ubuntu, se permite la instalación y uso del framework Robot Operating System (ROS) y permite un funcionamiento idéntico a como si se estuviese trabajando con una arquitectura x86.

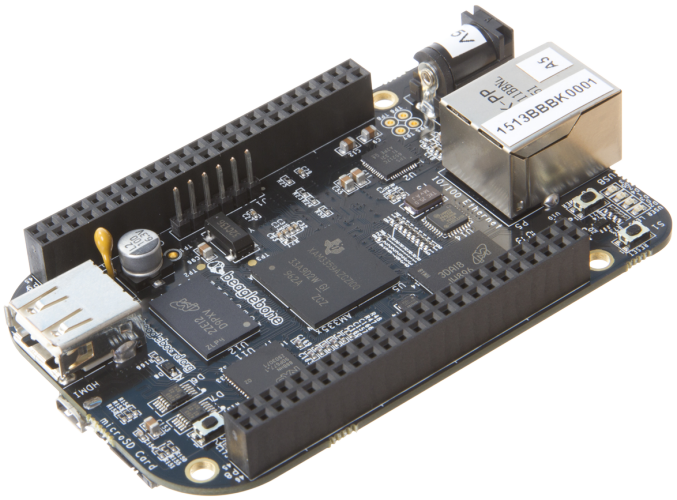


FIGURA 3.2: Beaglebone Black

3.5. ROBOCape

Como se ha mencionado en la sección anterior, las extensiones para la tarjeta BeagleBone, se llaman capes. En este apartado se va a dar a conocer la cape ROBOCape [26], la cual ha sido diseña por Miguel Albero del servicio de electrónica del Instituto Universitario de Automática e Informática Industrial.

ROBOCape es una placa electrónica que ha sido diseñada exclusivamente para usarla con la tarjeta BeagleBone Black. Así se permite un amplio abanico para añadirlo al conjunto a sistemas robotizados y poder hacer mas fácil la programación de estos sistemas. Durante el diseño de esta tarjeta se ha realizado un análisis exhaustivo sobre que dispositivos se usan más a menudo en este tipo de sistemas. Y gracias a esto permite un amplio abanico de aplicaciones de control para la gran mayoría de robots existentes.

Para que esta placa pueda funcionar con la BeagleBone, esta debe estar alimentada con alimentación externa de 5V y 1A o más para que el sistema de seguridad no apague la tarjeta. Además incorpora una entrada para conectar una batería externa de entre 6 y 12V de dos celdas. Las características técnicas que ofrece son las siguientes.

- Sistema de alimentación DC/DC independiente, es decir, la tensión de entrada tiene que ser tensión continua y como mínimo de 1A. La entrada máxima permitida

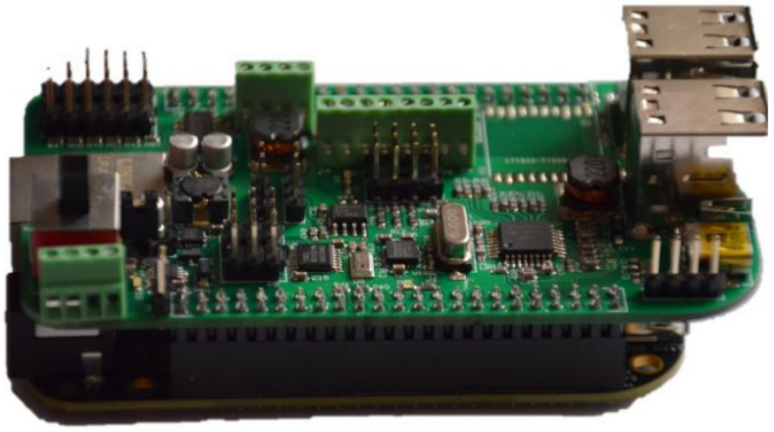


FIGURA 3.3: ROBOCape

es de 12V y capaz de ofrecer hasta 3A. Incorpora un sistema para proteger al sistema de cambio de polaridad y protección para evitar sobre-tensiones.

- El sistema de carga de baterías es para las baterías lipo de 2 células de 7.4V, además incorpora un jumper para seleccionar el modo de funcionamiento, si carga solo o carga y funciona con alimentación externa.

- El acelerómetro que incluye es el MPU-9150. Este, incluye un giróscopo que es capaz de devolver la velocidad angular en las coordenadas X, Y, Z con un rango escalable de entre $\pm 250°a \pm 2000°/sec$. El acelerómetro es capaz de devolver aceleraciones de entre ±2G hasta ±16G en los 3 ejes, X, Y, Z. Además incluye un magnetómetro, que obtiene el campo magnético en los 3 ejes, gracias a un sensor monolítico de efecto Hall. Este es capaz de devolver una salida con 13 bits de resolución, es decir 0.3T por LSB. El cual está conectado al bus I2C.

- Otro de los componentes que incluye es el altímetro MPL3115A2, y permite conocer la información a la BBB, referente a la presión y altitud, así como la temperatura del entorno en el cual este. También se encuentra conectado al bus I2C.

- El GPS que incorpora es el FGPMMOPA6E, el cual permite obtener una alta precisión en la posición y velocidad, y con su alta sensibilidad lo hace un buen candidato para funcionar en entornos urbanos. Es capaz de soportar hasta 66 canales. Tiene un rango de funcionamiento de entre 1 y 10 Hz. Y el formato de adquisición de datos esta compuesto por el protocolo NMEA. Este dispositivo se conecta a través de una UART.

- Debido a que la BBB lleva un puerto USB y esto es una limitación, la ROBOCape permite ampliar hasta 4 puertos USB, gracias al HUB TUSB2046B que incorpora.

- Otro de los componentes existentes en esta placa, son los puentes H con los cuales se pueden manejar los motores DC o paso a paso, el controlador (BA6845) integrado permite trabajar con corrientes inferiores a 1A en régimen nominal. El control se realiza mediante las señales PWM y los pines de la GPIO.

- Como cabe esperar, ROBOCape, incluye entradas analógicas, en concreto 4, estas sirven para conectar los sensores de distancia Sharp GP2xx, y admite tensiones de hasta 1.8V, e integra un adaptador de tensiones máximas mediante un divisor resistivo.

- Permite incorporar sensores de ultrasonido HC-SR04, en total se permiten conectar hasta 6 empleando una topología en anillo. Gracias a la lógica de diseño que lleva, permite utilizar un número no muy elevado de pines. El funcionamiento de este modulo sigue la lógica de calculo del tiempo de vuelo de la señal.

- Finalmente existen 4 líneas directas que vienen de los controladores PWM, estas adaptan las señales de la tensión típica en señales para el servo mediante buffers de protección. De esta forma se crea un sistema de protección para los problemas eléctricos provocados por circuitos externos.

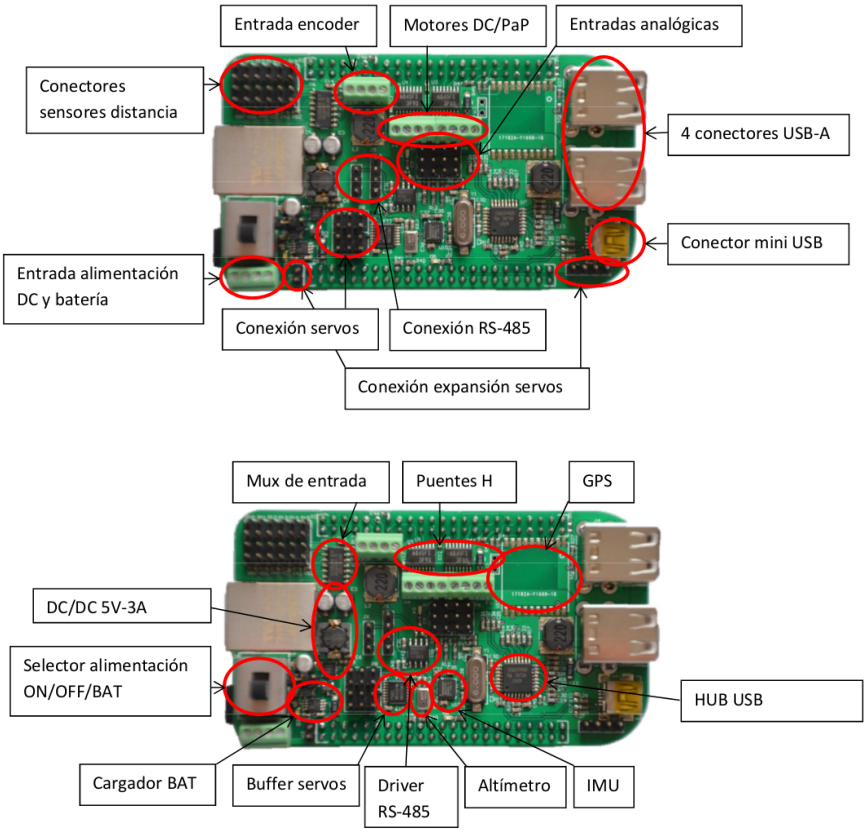


FIGURA 3.4: Descripción de conectores, módulos, IC's y distribución

Capítulo 4

Trabajo Desarrollado

En el capítulo actual se describe el trabajo realizado para la elaboración de este trabajo final de máster. Primero se demuestra resumidamente como se emplea el simulador de movilidad SUMO y se detalla el escenario creado para obtener el fichero de configuración para el simulador NS-3. En la siguiente sección se da a conocer el modelo de propagación empleado e implementado para dicho simulador. Y a continuación se describe el modelado e implementación de un protocolo de enrutamiento para redes vehiculares para el simulador NS-3. Finalmente, en la última sección de este capítulo se explica el desarrollo que realizado sobre la placa BeagleBone Black, empleando el middleware ROS, y un ejemplo de aplicación para robots humanoides, así mostrando su alta escalabilidad para diferentes plataformas.

4.1. Ecosistema

Con las herramientas expuestas anteriormente se ha obtenido un ecosistema que tiene la funcionalidad implementar, ejecutar y analizar una serie de simulaciones basadas en Redes de Sensores Inalámbricos y Movilidad.

Por un lado 3.1 se emplea para desarrollar nuevos protocolos y tecnologías de red donde poder comparar, analizar y validar todos los desarrollos. Esto permite proponer y evaluar nuevas mejoras y protocolos implementados.

Para completar el ecosistema se necesita una serie de patrones de movilidad, que previamente han tenido que ser generados y analizados, de ahí la necesidad de emplear SUMO 3.2.

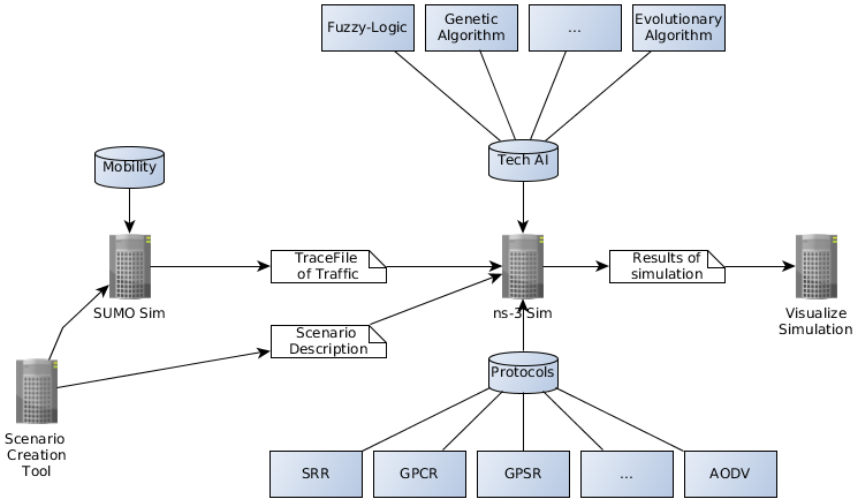


FIGURA 4.1: Diagrama de bloques del ecosistema propuesto

4.2. Generación de escenarios en SUMO

La principal necesidad de emplear el simuladores de movilidad es que permite generar comportamientos similares al mundo real. De esta forma se puede analizar la densidad de tráfico en una zona concreta. En este campo de investigación emplear un simulador de movilidad con las características que ofrece SUMO es de vital importancia, ya que se pueden generar todo tipo de tendencias de conducción, aumentar la densidad del tráfico, generar posibles accidentes, entre otros comportamientos reales. La obtención de este tipo información se puede reutilizar para la experimentación y análisis de las redes vehiculares, y así poder desarrollar nuevas técnicas y tecnologías que permitan a los conductores una sensación de seguridad cuando van al volante de su coche.

La elección de SUMO como simulador de movilidad ha sido debido a que es de de código libre y esta escrito bajo licencia *GNU General Public License version 3.0 (GPLv3)*, además de ser portable. Proporciona un paquete de simulación de tráfico rodado microscópico y continuo, está diseñado para manejar grandes redes de carreteras. Permite la simulación incluyendo peatones intermodales y viene con un gran conjunto de herramientas para la creación de escenarios. Por otra parte, permite un enorme acoplamiento con cualquier simulador de redes, ya que la información que se obtiene viene en un formato estandarizado, de fácil y rápida la conversión para los simuladores de redes, como por ejemplo a ns-2 y ns-3.

A continuación se describirán una serie de pasos básicos para el rápido manejo de SUMO.

La creación de escenarios en SUMO se puede realizar de dos formas, una mediante la descarga de mapas en formato XML o mediante la edición de ficheros XML. Si se

emplea la primera mencionada, accediendo previo registro a **OpenStreetMap** se puede descargar una determinada zona. Como por ejemplo Figura 4.2.

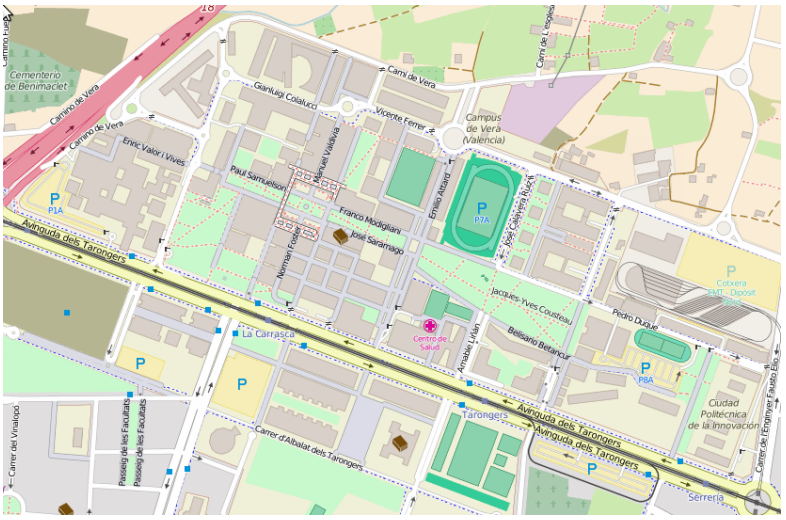


FIGURA 4.2: Área de la Universidad Politécnica de Valencia

Cuando el mapa esté descargado hay que convertirlo a formato XML, esto se realiza con la ayuda del comando **netconvert** de SUMO. Una vez completado este paso se obtiene el fichero XML que conforma la red de los nodos que forman las calles, carreteras, etc. Para crear un escenario de simulación que se aproxime a la realidad, es decir, añadir zonas verdes, construcciones, etc, hay que emplear el comando **polyconvert** el cual recibe como entrada el fichero obtenido anteriormente y un fichero que se obtiene de la web de la organización que lo ha desarrollado. Seguidamente se crean una serie de trayectorias aleatorias, esto se consigue con el comando **randomTrips.py** el cual recibe como argumento de entrada el fichero que contiene la red de carreteras y como fichero de salida genera un XML con las rutas. Una vez completado esto, se edita un fichero de configuración que sirve para ejecutar la la simulación. Para ejecutar la simulación se escribirá la orden **sumo-gui map.sumo.cfg** y se obtendrá un ventana similar a la mostrada en la Figura 4.3

Una vez cargado el simulador, si se realiza zoom y se pone en ejecución la simulación aparecerán vehículos en movimiento y semáforos cambiando sus luces, en la Figura 4.4 se muestra un ejemplo.

Por otra parte, SUMO permite crear escenarios personalizados, para ello se debe de seguir un flujo de trabajo tal como indica el diagrama de la Figura 4.13.

Siguiendo el diagrama de flujo de la Figura 4.13 se van creando los ficheros. Primero se crea el archivo que contendrá los nodos. En la Figura 4.5 se muestra un ejemplo.

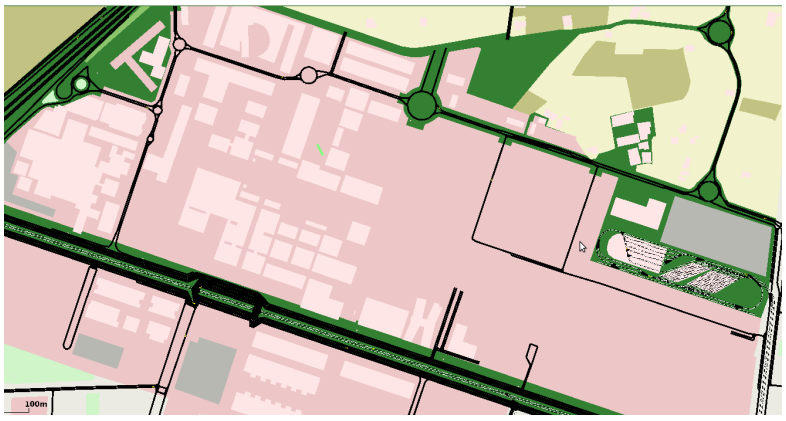


FIGURA 4.3: Área simulada de la Universidad Politécnica de Valencia

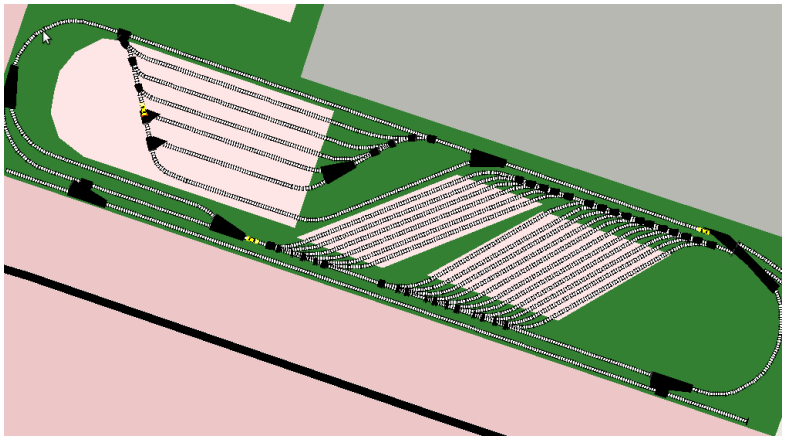


FIGURA 4.4: Ejemplo simulación

```
<nodes>
        <!--<node id="id" x="x-coordinate" y="y-coordinate" />-->
        <node id="1" x="+250.0" y="0.0" />
        <node id="2" x="-250.0" y="0.0" />
        <node id="3" x="-295.0" y="-45" />
        <node id="4" x="-295.0" y="-195" />
</nodes>
```

FIGURA 4.5: Ejemplo .nod.xml

A continuación se editará el fichero que contendrá las aristas de conexión entre los nodos. En la Figura 4.6 se muestra un ejemplo.

```
<edges>
    <edge from="1" id="1to2" to="2" numLanes="2"/>
    <edge from="2" id="2to3" to="3" numLanes="2"/>
    <edge from="3" id="out" to="4" numLanes="2"/>
</edges>
```

FIGURA 4.6: Ejemplo .edg.xml

Tras la edición de estos dos ficheros y la ejecución de la aplicación *netconvertert* se obtiene el fichero que contiene la red de nodos que conformará el escenario, en la Figura 4.7.

```
<net version="0.13" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/net_file.xsd">
    <location netOffset="0.00,0.00" convBoundary="0.00,0.00,4000.00,500.00"
    origBoundary="0.00,0.00,4000.00,500.00" projParameter="!"/>
    <edge id=":1_0" function="internal">
        <lane id=":1_0_0" index="0" speed="13.90" length="4.82"
        shape="-1.65,0.00 -0.83,-1.24 -0.00,-1.65 0.82,-1.24 1.65,0.00"/>
    </edge>
    <edge id=":2_0" function="internal">
```

FIGURA 4.7: Ejemplo .net.xml

El siguiente paso consiste en generar una serie de flujos de movilidad, ello se consigue con la edición del fichero de flujo. Figura 4.8

```
<flows>
        <interval begin="0" end="160">
                <flow id="0" from="1to2" to="out"  number="600"/>
                <flow id="1" from="4to3" to="in"  number="600"/>
        </interval>
</flows>
```

FIGURA 4.8: Ejemplo .flow.xml

Ejecutando al aplicación *duarouter*, y indicando el fichero de entrada como el .flow.xml y una serie de parámetros se obtiene el fichero .rou.xml que contendrá las rutas a seguir por los vehículos en el escenario diseñado. En la Figura 4.9 se muestra en fragmento de código.

```
<vType id="CAR" length="5.00" maxSpeed="90.00" probability="1.00">
    <carFollowing-Krauss accel="0.50"/>
</vType>
<vehicle id="V0.0" type="CAR" depart="0.00" color="green">
    <route edges="1to2 2to3 out"/>
</vehicle>
```

FIGURA 4.9: Ejemplo .rou.xml

Una vez llegados hasta este punto, se procede a la edición del fichero de configuración de la simulación, para ello, hay que crear un fichero con extensión .cfg similar al mostrado en la Figura 4.10, dónde se indicarán los ficheros que conforman la red y las rutas, además se puede indicar una serie de parámetros, como el inicio y finalización de la simulación, el tamaño de incremento entre cada instante, entre otros.

```
<input>
    <net-file value="srr_1.net.xml"/>
    <route-files value="srr_1.rou.xml"/>
</input>
<time>
    <begin value="0"/>
    <end value="160"/>
<step-length value="0.1"/>
</time>
```

FIGURA 4.10: Ejemplo .cfg

Tras terminar de crear este fichero se ejecutará el simulador, indicando un fichero de salida para almacenar los resultados obtenidos. En la Figura 4.11 se muestra un escenario de testeo, y en la Figura 4.12 se puede observar el formato que sigue el fichero de traza.

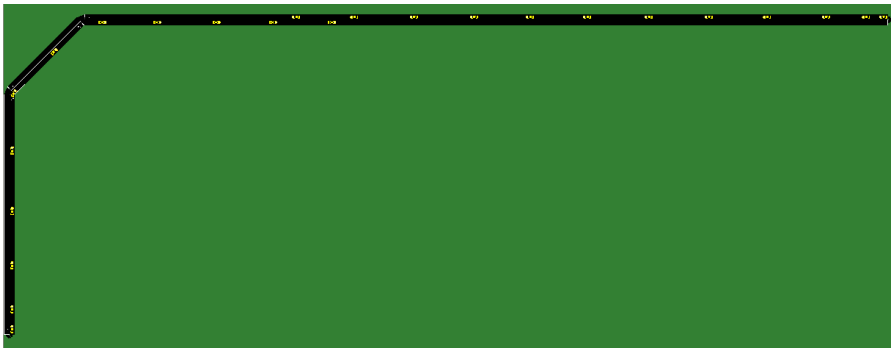


FIGURA 4.11: Ejemplo simulación

```
<timestep time="0.00">
    <vehicle id="V0.0" x="4.95" y="5.10" angle="180.00"
    type="CAR" speed="0.00" pos="5.10" lane="1to2_0" slope="0.00"/>
    <vehicle id="V1.0" x="3994.90" y="504.95" angle="-90.00" type="CAR"
    speed="0.00" pos="5.10" lane="4to3_0" slope="0.00"/>
</timestep>
```

FIGURA 4.12: Fichero de traza

Una vez terminado esta parte del proceso, con ayuda del script *traceExporter.py* que incorpora SUMO, se pueden generar una serie de ficheros que serán empleados por el simulador de redes NS-3.

4.3. Desarrollo de un modelo en NS-3

4.3.1. Modelo de propagación empleado

El modelo de propagación en espacio libre se emplea en las comunicaciones inalámbricas para predecir el nivel de potencia de la señal de recepción de cada paquete. Existe una línea de visión o más comúnmente conocido como *line-of-sigth (LOS)* entre un nodo emisor (Tx) y un nodo receptor (Rx) Figura 4.14 en la cual por diversos fenómenos puede aparecer una zona de sombreado y dificultar la recepción de paquetes. En el simulador NS-3 existen una serie de modelos de propagación que son más frecuentemente usados para las comunicaciones inalámbricas, estos son: *Friss propagation loss model*, *Log Distance propagation model* entre otros. El primer modelo enunciado se caracteriza por asumir que solo existe una única ruta en la línea de visión entre el transmisor y el receptor, y suele emplearse para calcular la señal de recepción en espacios abiertos a una distancia d desde el transmisor. Este modelo está definido por la ecuación 4.1

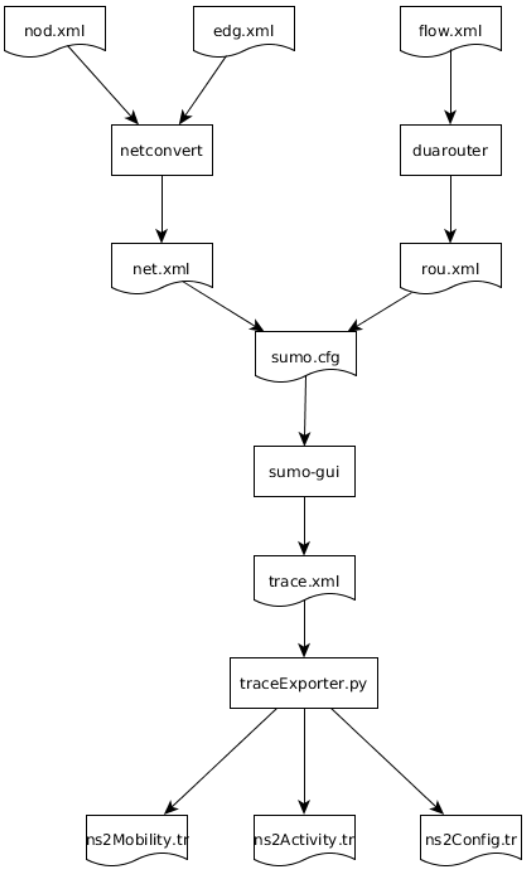


FIGURA 4.13: Diagrama de flujo de SUMO

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{4.1}$$

Donde P_r es la potencia de la señal transmitida, G_t y G_r es la ganancia de la antena del transmisor y del receptor. $L(L \geq 1)$ es la pérdida del sistema y λ es la longitud de onda.

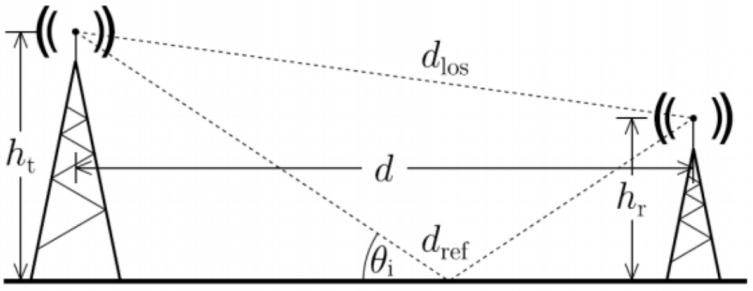


FIGURA 4.14: Distancia entre Rx y Tx

EL siguiente modelo enunciado ha sido *Log Distance propagation model* el cual es una extensión del modelo de propagación de *Friss*. Este tipo de modelos se les conoce como *Shadowing Model* y está formado por dos partes: una parte en la cual se modelan

Entorno	Exponente pérdida trayectoria
Espacio Libre	2
Área urbana de radio celular	2.7 a 3.5
Radio Sombreado urbano	3 a 5
Dentro de edificio - Línea de visión	1.6 a 1.8
Obstrucción por edificios	4 a 6
Obstrucción por fabrica	2 a 3

CUADRO 4.1: Exponentes de pérdida de trayectoria para diferentes entornos

las perdidas en la ruta de comunicación y otra que predice la potencia media a una distancia d denotado por $P_r(d)$ y n es el exponente de perdida de la ruta, este modelo está representado por la ecuación 4.2

$$PL(d)[dB] = \overline{PL}(d_0) + 10nlog(\frac{d}{d_0}) \tag{4.2}$$

Este modelo presenta un inconveniente, y es que no considera las condiciones de los entornos. Es decir, estos pueden ser diferentes para una misma separación entre un nodo Tx y Rx, por tanto, a esas mediciones se cree oportuno aplicar un valor de la *media* diferente. Esto da lugar al modelo *Log Normal Shadowing*, aplicando el campo X_σ que es una variable aleatoria Gaussiana con media cero y desviación estándar σ [27]. Quedando el modelo como 4.3

$$PL(d)[dB] = \overline{PL}(d_0) + 10nlog(\frac{d}{d_0}) + X_\sigma \tag{4.3}$$

donde $PL(d)$ es la pérdida de trayectoria en una distancia d entre un Tx y un Rx. $PL(d_0)$ es la media de la perdida de la trayectoria de la referencia d_0, n es el exponente de la perdida de la trayectoria 4.1. En Figura 4.15 se puede observar una representación del modelo.

En la definición del último modelo ilustrado se han empleado las características de radiofrecuencia de un dispositivo real, concretamente las características de la antena Wifi WF121 [28]. En el datasheet [28] se describen sus características.

4.3.2. Implementación del protocolo de red

Se ha desarrollado un protocolo de enrutamiento basado en el descrito en *Stability and Reliabilty Aware Routing (SRR)*[5] que ha sido presentado en 2.3. En la implementación del protocolo se han modificado determinados aspectos y este ha sido desarrollado en el

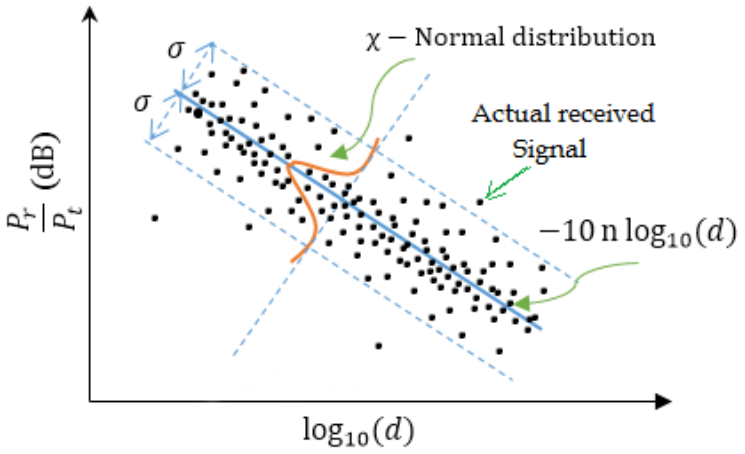


FIGURA 4.15: Distribución LogNormal

simulador ns-3. Como referencia se ha tomado el diagrama de flujo del protocolo SRR mostrado en Figura 2.10. Este protocolo tiene dos modos de funcionamiento: un modo conectividad, en el cual existe una red ad-hoc que permite que los nodos se comuniquen entre ellos sin la necesidad de que exista un infraestructura V2I y que estén siempre dentro de un radio de cobertura. Cuando se envían los paquetes hacia el nodo destino se toman una serie de decisiones. Para el cálculo de estas decisiones se calcula la distancia y dirección, además del ángulo que existe entre el nodo fuente y el nodo destino y la orientación. Esto hace que el envío de paquetes sea más directo para alcanzar el destino. Los parámetros de la distancia, y orientación se calculan para que la entrega de los paquetes sea mas óptima y robusta, originalmente el autor ha empleado un sistema de basado en lógica difusa. Con este sistema obtiene una función de coste que sirve para indicar que nodo es el mejor candidato para alcanzar el destino. El segundo modo de funcionamiento, se activa cuando el nodo transmisor ha perdido la conectividad con el resto de nodos, esto se debe a que no existe ningún nodo dentro del radio de alcance, es lo mismo que decir que su tabla de vecinos está vacía. En este modo los paquetes se van guardando en una cola y periódicamente ésta está intentando enviar los paquetes hacía el destino. En el caso de no tener conectividad durante un tiempo si la cola se ha llenado se emplea un buffer temporal. Cuando ambas colas están llenas el protocolo descarta ese paquete y vuelve a intentar entrar en conectividad.

Para el desarrollo de este protocolo de enrutamiento se ha propuesto una pseudo-implementación debido a que no existe información al 100 % del protocolo. Esta pseudo-implementación queda refleja en el siguiente diagrama de clases que es mostrado en la Figura 4.16.

Seguidamente se detalla la función que tiene cada clases que forma forma el diagrama 4.16. *HelloHeader* es una clase que hereda las características de la clase *Header* que

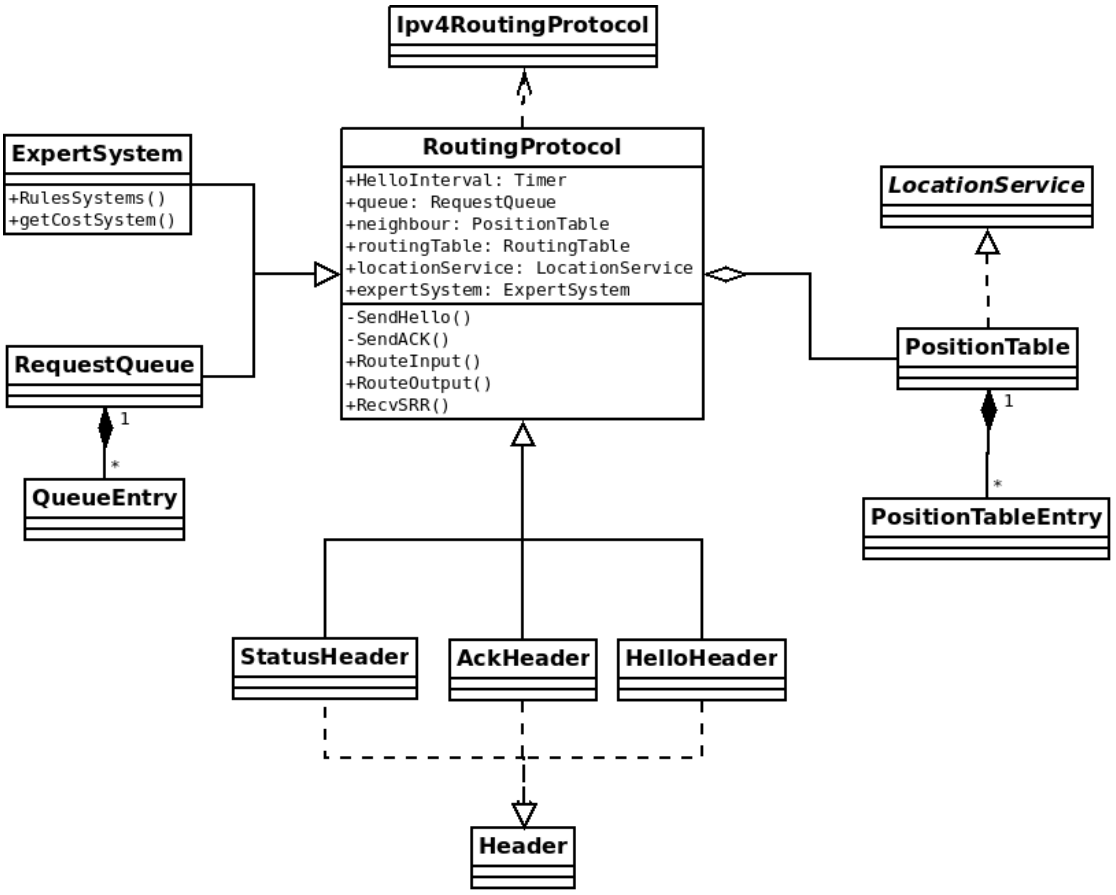


FIGURA 4.16: Diagrama UML del protocolo implementado

es propia del simulador ns-3. Esta clase, la función que tiene es crear la cabecera del mensaje **HELLO** que se emplea para enviar la localización actual del nodo origen al resto de nodos que forman la red. Está compuesto por los siguientes campos Figura 4.17:

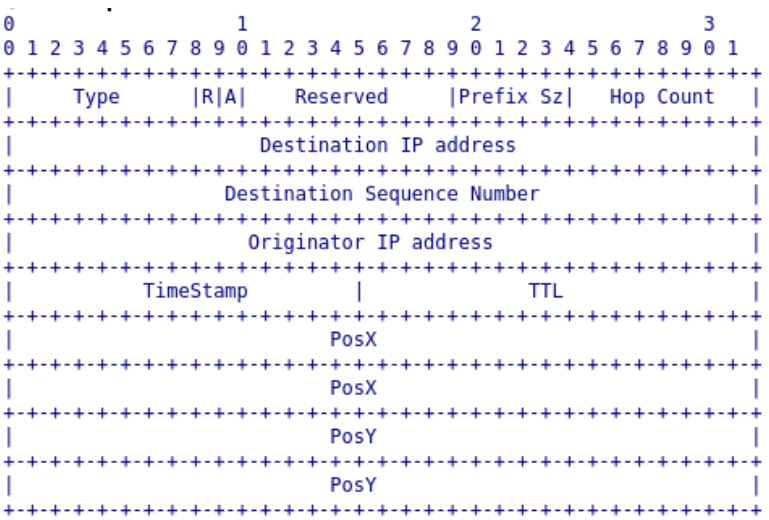


FIGURA 4.17: Cabecera mensaje HELLO

- **Hop Count**: este campo tiene un tamaño de 8 bits, y sirve para llevar la cuenta de los saltos que ha realizado un determinado paquete.
- **Destination IP Address**: es un campo de 32 bits y contiene la IP destino donde se enviará el mensaje, como este paquete sirve para realizar un descubrimiento de red albergará la dirección de broadcast propia de la red.
- **Destination Sequence Number**: este campo tiene un tamaño de 32 bits y lleva la cuenta de números de paquetes enviados por un determinado nodo.
- **Originator IP Address**: este campo tiene un tamaño de 32 bits y contiene la IP del nodo que origina dicho mensaje.
- **TimeStamp**: este campo también tiene una capacidad de almacenamiento de 16 bits, y sirve para conocer el instante en que ha sido creado el paquete.
- **TTL**: TimeToLive, este campo se emplea para delimitar la vida de un paquete, así evitando que este indefinidamente viajando por la red, y por tanto sobrecargándola.
- **PosX** y **PosY**: son dos campos de 64 bits del tipo double, se ha optado que tenga esta capacidad porque la información que adjuntará hace uso de la coma flotante del computador en que este, e interesa que el dato sea lo más próximo a la realidad, y sin que exista la certeza de que el tamaño llegue a desbordar y por tanto transmitir un mensaje erróneo.

AckHeader es otra clase que hereda la funcionalidad de *Header*, este tipo de paquete sirve como respuesta a los mensajes HELLO. En este paquete se envía la ubicación del nodo destino, además de encapsular en la cabecera la dirección y el sentido de la marcha con respecto al nodo que ha enviado el mensaje HELLO. En la Figura 4.18 se muestra el formato de la cabecera que es detallada a continuación:

- **Hop Count**: este campo tiene un tamaño de 8 bits, y sirve para llevar la cuenta de los saltos que ha realizado un determinado paquete.
- **Destination IP Address**: es un campo de 32 bits y contiene la IP destino donde se enviará el mensaje de respuesta al nodo que haya enviado previamente el mensaje HELLO.
- **Originator IP Address**: este campo tiene un tamaño de 32 bits y contiene la IP del nodo que origina dicho mensaje.

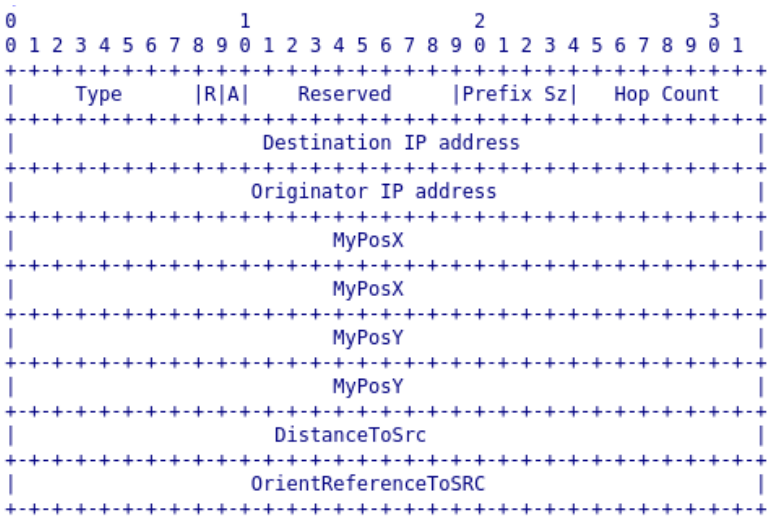


FIGURA 4.18: Cabecera mensaje ACK

- **MyPosX** y **MyPosY**: son dos campos de 64 bits del tipo double, se ha optado que tenga esta capacidad porque la información que adjuntará hace uso de la coma flotante del computador en que este, e interesa que el dato sea lo más próximo a la realidad, y sin que exista la certeza de que el tamaño llegue a desbordar y por tanto transmitir un mensaje erróneo.
- **DistanceToSrc**: es un campo de 32 bits que permite informar al nodo que envió el mensaje HELLO cual es la distancia que existe entre el nodo origen y destino, además, gracias al signo se puede saber si va en la misma dirección o en la dirección opuesta.
- **OrientReferenceToSRC**: Permite obtener el ángulo que existe entre el nodo origen y destino, este campo tiene un tamaño de 32 bits.

La clase *StatusHeader* implementa un paquete de datos, que sirve para informar a un nodo destino que esté dentro de de la propia red ad-hoc creada. Los campos que conforman este paquete son los mostrados en la Figura 4.19:

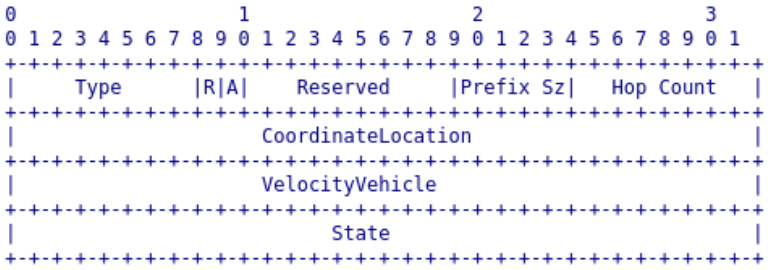


FIGURA 4.19: Cabecera mensaje STATUS

- **Hop Count**: este campo tiene un tamaño de 8 bits, y sirve para llevar la cuenta de los saltos que ha realizado un determinado paquete.
- **CoordinateLocation**: es un campo de 32 bits que contiene las coordenadas de localización del nodo emisor.
- **VelocityVehicle**: es un campo de 32 bits que contiene la velocidad a la que circula el vehículo en un instante de tiempo determinado.
- **State**: tiene una capacidad de 32 bits, y sirve para indicar a los vehículos vecinos del estado actual de calzada.

LocationService es una clase abstracta que sirve para localizar un nodo geográficamente, en la versión oficial del simulador ns-3 no se encuentra aún. Este servicio fue creado para protocolos basados en localización geográfica [29] y así poder emular el comportamiento de un GPS.

RequestQueue sirve para encolar los mensajes que no han podido ser enviados debidos a una desconexión temporal de los nodos.

La clase *ExpertSystem* implementa las reglas descritas en [5] en un sistema experto basado en reglas. Con ella, se permite obtener las funciones de coste, para ayudar a decidir a que vecino hay que enviar los mensajes hacia el destino. De esta forma se intenta obtener un comportamiento aproximado al propuesto en Figura 2.10.

Finalmente, en la clase *RoutingProtocol* se encuentra la lógica del protocolo de enrutamiento propuesto en la figura 2.10. Aquí es donde se realizan las funciones de establecer la IP a los nodos que forman la red ad-hoc enviar y recibir los distintos paquetes, además de tomar las decisiones para enviar a un nodo u otro los distintos paquetes. A continuación se va a detallar el funcionamiento de las funciones principales que se emplean en la implementación de este objeto. En la figura 4.20 se muestra el diagrama de flujo del protocolo implementado, las modificaciones realizadas ha sido suprimir el modo desconectividad, y cambiar el sistema basado en *Fuzzy Logic* por un sistema basado en reglas. Este sistema emplea la misma salida que el propuesto originalmente.

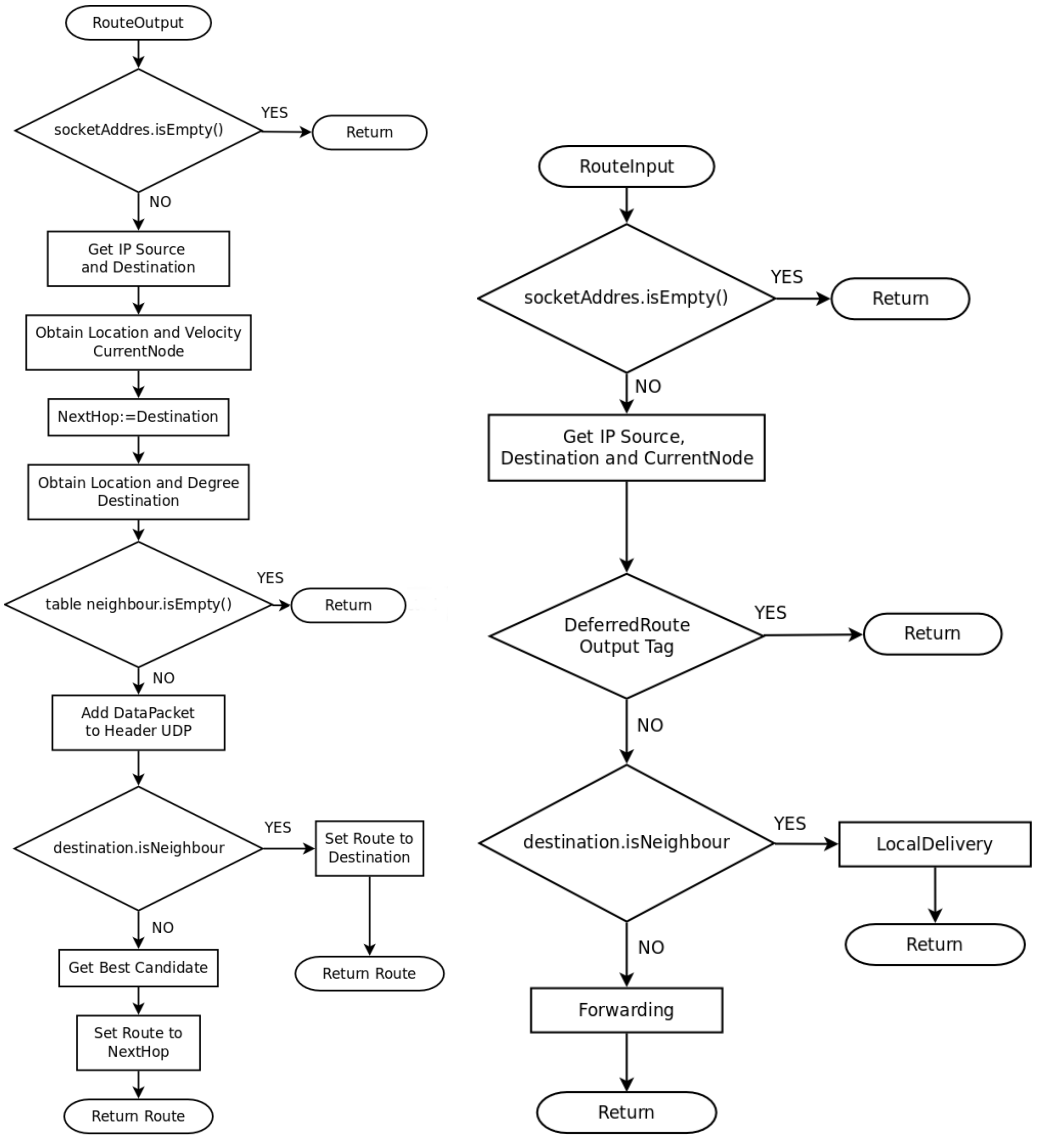
Seguidamente se describirá el funcionamiento del protocolo implementado. Cuando desde el fichero de configuración del escenario se invoca al protocolo, ha este se le pueden pasar una serie de atributos para su configuración. En este conjunto de atributos se encuentra *HelloInterval*, la cual es una variable del tipo *Time* que por defecto está inicializara a un segundo, con este tiempo se configura un *JITTER* el cual sirve para la inicialización de la tasa de envío de paquetes HELLO. Este tipo de paquetes se envían periódicamente para la construcción de la tabla de vecinos de los nodos. Este tipo de mensaje hace una

inundación a todos los nodos que están dentro de su alcance. En el momento que un nodo que está dentro del rango de cobertura y ha recibido este paquete analiza dicha cabecera y crea una entrada en la tabla de vecinos, en caso de que ya existiera la entrada actualiza sus valores. Una vez completado este proceso el nodo envía un paquete ACK al nodo que inició la comunicación, trasmitiendo los campos descritos anteriormente. El nodo receptor de este paquete analiza el mensaje y actualiza su entrada de la tabla de vecinos.

Una vez, se ha vencido el tiempo de espera para enviar los mensajes de datos entra en funcionamiento las funciones *RouteOutput* y *RouteInput*, que están mostradas en la figura 4.20. La llamada a RouteOutput se realiza en el momento que el nodo fuente está listo para retransmitir el paquete de datos y por tanto a configurar la ruta de salida para que después RouteInput a través de los nodos intermedios sea capaz de alcanzar el destino del paquete, ya sea realizando una entrega directa o retransmisiones, *Forwarding*. Como se puede observar en la figura 4.20 la función RouteOutput y RouteInput siguen el flujo de datos establecido por el diagrama original 2.10.

Cuando RouteOutput entra en ejecución se comprueba que el socket de la dirección no esté vacío, en caso de estarlo devuelve una ruta establecida por defecto en el simulador. El siguiente paso es obtener los distintos atributos del nodo fuente, es decir, su IP, posición, y a través del Header obtenemos la dirección IP destino.Este destino se establece como siguiente salto a alcanzar dentro de la ruta. Entonces se comprueba si la tabla de vecinos del nodo fuente esta vacía, en caso de estarlo el paquete se descarta. En caso contrario se añade la información al Header y seguidamente se comprueba si el destino es vecino o no, en caso serlo se configura la ruta de salida para alcanzar el destino. En caso de no ser un nodo vecino el destino entra en funcionamiento el sistema basado en reglas. En este sistema se obtiene el nodo vecino con una mejor función de coste para configurar la ruta.

Una vez configurada la ruta por el nodo origen, entra en funcionamiento la función RouteInput, en esta función se peude realizar la llamada a una serie de callbacks para alcanzar el destino, aquí se vuelve a comprobar si la IP destino y su interfaz son destinos, en caso afirmativo se realiza una llamada al callback *LocalDeliverCallback* y de esta forma el paquete ha alcanzado su destino. En caso contrario se realiza la llamada a la función *Forwarding*. Esta función tiene un comportamiento similar a RouteOutput. Se obtiene la IP destino, fuente e IP de nodo intermedio, seguidamente se comprueba si la IP destino del paquete es la del nodo destino. En caso afirmativo se configura la ruta para la entrega directa. En el caso contrario se obtiene cual será el mejor nodo candidato para alcanzar



(a) RouteOutput (b) RouteInput

FIGURA 4.20: Diagrama flujo protocolo propuesto

el destino, esto se realiza de forma similar que en RouteOutput, para poder retransmitir los paquetes se emplea el callback *UnicastForwardCallback*.

En la figura 4.21 se puede apreciar el esquema de comunicaciones que sigue el simulador ns-3 para transmitir el paquete a través del protocolo de transporte y del protocolo IP, y las llamadas que se realizan a la hora de intercambiar el paquete con las capas inferiores de la pila IP.

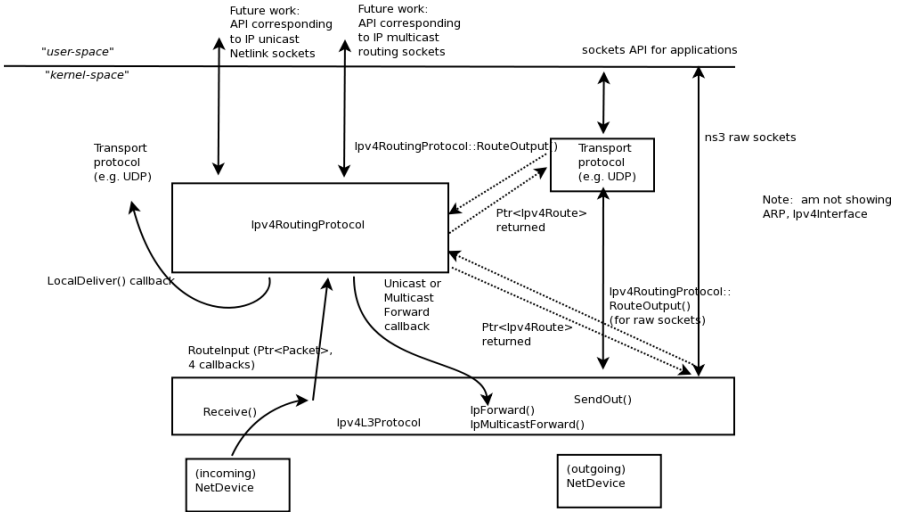


FIGURA 4.21: Esquema comunicaciones con niveles inferiores de la pila IP

4.4. Nodos ROS

En paralelo se ha desarrollado parte de la tesis doctoral, esto es la implementación de nodos en ROS para la BeagleBone Black, necesaria para la implementación sobre nodos reales de las propuestas realizadas de protocolos para redes vehiculares previamente simuladas y evaluadas, dado que estos protocolos necesitan de diferentes dispositivos y periféricos como GPS, etc.

4.4.1. Descripción nodos de ROS

En el trabajo realizado se ha integrado la librería BlackLib [30] dentro de un nodo ROS. Esta librería ha sido escrita para que el acceso a bajo nivel de los componentes HW de la BBB sea transparente. El lenguajes empleado para su desarrollo es C++, y permite la lectura de las entradas analógicas, generar señales PWM, poder emplear los pines GPIO. Además permite realizar la comunicación con otros dispositivos que empleen la UART, I2C y SPI. Esta integración se ha podido realizar ya que ROS y BlackLib están escrito en C++. A la librería original se le ha han añadido dos objetos como se muestra en la Figura 4.22. Estos dos objetos añadidos permiten leer los valores devueltos por la IMU y por el Altímetro, son los que están de color verde. Se ha optado por realizar estas dos clases para facilitar en el futuro el uso de estos dispositivos ya que había que realizar la lectura una serie de registros y direcciones de memoria. Para ello se han empleado los datasheets [31, 32] de los fabricantes, es donde se describe el mapa de direcciones de los dispositivos.

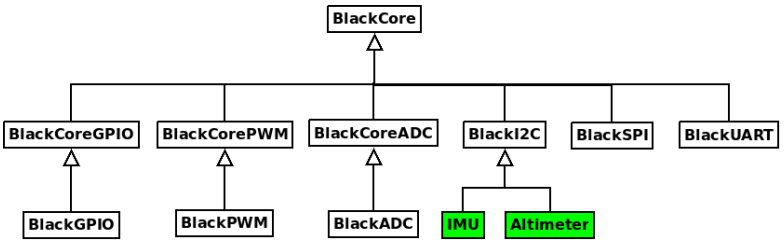


FIGURA 4.22: Jerarquía librería BlackLib

Para la implementación en ROS, se ha desarrollado el nodo mostrado en la Figura 4.23, que en el cual existen tres programas que permiten poner en funcionamiento la IMU, el altímetro y el GPS[33].

El primer componente desarrollado ha sido el acceso a la unidad de medida inercial el cual está funcionando a una frecuencia de 1 Hz. En este se puede obtener los valores en los 9 ejes, es decir, se puede obtener la aceleración lineal en las coordenadas X Y Z, además de las coordenadas respectivas tanto para la velocidad angular y el campo

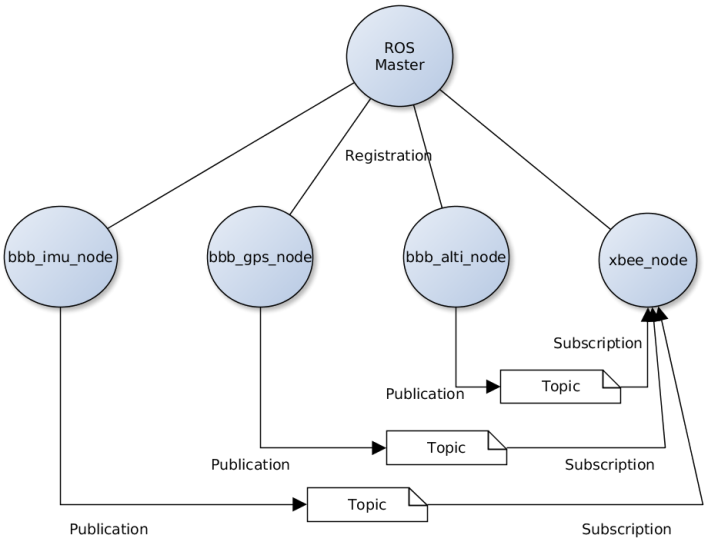


FIGURA 4.23: Nodo implementados en ROS

magnético al cual está sometida la BeagleBone Black. Esta información se publica en los mensajes estándares de ROS.

El segundo componente que se describe es el altímetro, con el cual se pueden obtener a que altitud sobre el nivel del mar a la que se encuentra la placa, además de conocer la temperatura que tiene la placa en un instante de tiempo y cual es la presión atmosférica a la cual está sometida. Como en el caso anterior este nodo también se puede conocer mediante los mensajes típicos de ROS y funciona a la misma frecuencia.

Otro nodo que ha sido implementado es el acceso al GPS, este está funcionado a una frecuencia de 0.5 Hz, el mensaje obtenido por el GPS sigue el estándar del protocolo NMEA y por tanto se ha tenido que realizar una conversión de formato y permite conocer la longitud, latitud y la altura sobre el nivel del mar a la que se encuentra la tarjeta en un momento determinado. Y al igual que en los casos anteriores también se publica a través de un mensaje de ROS.

Y por último, se ha creado otro nodo para la lectura de los sensores de tipo Sharp comentados en la sección 3.5. Este nodo permite la lectura de los 4 conversores ADC que permite el acceso la cape. Este nodo también funciona a 1 Hz y publica los datos a través de mensajes de ROS.

4.4.2. Integración librerías en ROS

Para poder emplear la librería Blacklib desde un nodo ROS, primero de todo debe de existir una área de trabajo para el usuario en ROS, esto se puede realizar de la siguiente forma:

```
mkdir -p ros_workspace/catkin_ws/src
cd ros_workspace/catkin_ws/src
catkin_init\_workspace
catkin_make
```

Listing 4.1: Comando para configurar área de trabajo en ROS

Seguidamente, hay que crear se tiene que crear un nodo. Este nodo ha de contener los ficheros necesarios para que pueda ponerse en funcionamiento desde el nodo *master*. Para crear el nodo hay que ejecutando la orden:

```
catkin_create_pkg 'nombre del nodo'
```

Listing 4.2: Crear un nodo en ROS

El siguiente a seguir consiste en ubicar los ficheros de cabecera dentro del directorio *include/nombre_nodo*, con los ficheros fuente se realiza una tarea similar pero esta vez en el directorio *src*. A continuación se ha de editar un fichero fuente el cual deberá de tener la función *main* y la funcionalidad que se desee que implemente el nodo. Para completar el proceso de integración hay que editar el fichero *CMakeList.txt*. En este fichero hay que localizar una serie de funciones que están comentadas. Estas funciones son:

- catkin_package: en su interior hay que añadir el directorio *include* y el nombre de la librería a emplear, hay que anteponer la palabra *lib*.
- add_library: hay que añadir el el nombre de la librería que se ha incorporado al paquete, además de la ruta a los ficheros fuente que se van a emplear.
- add_executable: en esta función se debe de escribir el nombre del ejecutable que a implementar y el nombre del fichero donde esté su implementación.
- target_link_libraries: esta función sirve para vincular el ejecutable con las librerías, por tanto hay que añadir el nombre del ejecutable que se ha dado en la función *add_executable* y el nombre de la librería dado en *catkin_package y add_library*.

A la hora de implementar el protocolo de enrutamiento propuesto en este Trabajo Final de Master, se seguirá una secuencia de trabajo similar a la mencionada en 4.4.2.

Capítulo 5

Experimentación y resultados

En el presente capítulo se describirá la metodología seguida para la validación del protocolo modelado en el capítulo 4. Para ello, primero se definirá un escenario similar al propuesto en [5] y se realizarán una serie de experimentos con los mismos parámetros de configuración. Seguidamente los resultados obtenidos serán analizados para validar dicha implementación.
Además se variarán algunos parámetros de simulación para analizar los cambios que provocan en la red y evaluar las prestaciones del protocolo. Finalmente, se mostrarán los resultados obtenidos en la implementación del equipo físico y sus distintas aplicaciones.

5.1. Evaluación del protocolo propuesto

Tal como se ha presentado en la sección 4.3 se ha desarrollado una aproximación del protocolo de enrutamiento SRR para entornos vehiculares. Para poder validar esta implementación se ha empleado un escenario generado con el simulador SUMO siguiendo los pasos mostrados en 4.2.

A continuación se detallan los parámetros que se han empleado para validar la implementación y los escenarios empleados en el simulador ns-3 para validar dicho protocolo.

5.1.1. Parámetros de la simulación y escenarios

En la realización del estudio en el impacto del *Channel Shadowing* se han empleado los siguientes parámetros:

- El número de nodos fuente han sido 5

- El número total de nodos del escenario han sido 150.
- La velocidad a la que se estaban moviendo los nodos ha sido de 25 metros por segundo.
- La velocidad de transmisión empleada ha ido variando de 16 kbps hasta 72 kbps.
- El valor desviación estándar σ para la variable aleatoria Gaussiana X_σ ha sido 2, 8 y 12.

En este experimento se analizarán el *Packet Delivery Ratio* (PDR), retardo medio y el control del *Overhead* con el ratio de envío de paquetes variando la desviación estándar X_σ.

En el estudio en el impacto de la Densidad se han utilizado los siguientes parámetros:

- El número de nodos fuente han sido 5.
- El número total de nodos del escenario han ido variando entre 30, 150 y 250.
- La velocidad a la que se estaban moviendo los nodos ha sido de 25 metros por segundo.
- La velocidad de transmisión empleada ha ido variando de 16 kbps hasta 72 kbps.
- El área de cobertura ha sido de 200 metros.

En este experimento se estudiará el PDR, retardo medio y el control del Overhead con el ratio de envío de paquetes variando la densidad de nodos.

En el análisis del impacto de la velocidad sobre el escenario se han empleado los siguientes parámetros:

- El número total de nodos del escenario han sido 150.
- La velocidad ha ido variando entre 25 y 30 metros por segundo.
- La velocidad de transmisión empleada ha ido variando de 16 kbps hasta 72 kbps.
- El área de cobertura ha sido de 200 metros.
- El valor desviación estándar σ para la variable aleatoria Gaussiana X_σ ha sido 2.

En este experimento se medirá el PDR, retardo medio y el control del Overhead con el ratio de envío de paquetes variando la velocidad a la circulan los nodos.

En todos los escenarios el número de nodos receptores ha sido siempre uno y el tamaño de paquete de datos se ha establecido en 1000 Bytes. A continuación se muestra una tabla 5.1 resumen de los parámetros empleados en cada escenario.

Exp.	Fuentes	Núm. Nodos	Vel (m/s)	CBR (kbps)	X_σ
ch. Shadowing	5	150	25	16 - 72	2, 8, 12
Densidad	5	30,150,250	25	16 - 72	8
Velocidad	5	150	25, 30	16 - 72	2

CUADRO 5.1: Tabla resumen experimentos

El escenario sobre el que se realizarán los experimentos es una aproximación al escenario propuesto por [5]. El escenario empleado ha sido ligeramente modificado, aún así simula una autovía. Este escenario tiene unas dimensiones de 4000x5 metros, lo que vienen a ser dos carriles de una carretera con gran afluencia de tráfico. Los motivos de escoger estos parámetros es debido a que cuando no existen aglomeraciones los vehículos están muy dispersos y se quiere estudiar el comportamiento de cuando hay un gran riesgo de que los nodos estén desconectados, de esta forma se puede conocer el rango de alcance de los dispositivos inalámbricos. En caso contrario, cuando existe una gran densidad de vehículos también es importante conocer como se comportará la red creada por los vehículos, de ahí que se tenga que analizar la sobrecarga que se genera en la red. La importancia del estudio del comportamiento de la red según distintas velocidades se debe a que cuando los vehículos van a elevadas velocidades es más difícil que pueda se pueda establecer una conectividad con el resto de vehículos ya que el espectro formado por la antena inalámbrica está cambiando constante y está influenciado por las irregularidades del entorno. Finalmente, se analiza también el *Channel Shadowing*, por que como bien se sabe, no siempre llega la señal en perfectas condiciones, esto se debe a que sí el área en que se encuentre el vehículo en un cierto momento está libre de elevaciones u otros fenómenos naturales adversos que provoquen una mala propagación de la señal. De ahí que se haya empleado el modelo de propagación mencionado en 4.3.1. En la realización de la simulación cada escenario se ha simulado diez veces y se ha obtenido el valor medio del total de las simulaciones realizadas. Por otra parte, el tiempo de simulación son 160 segundos. Se ha establecido un tiempo de espera de 30 segundos de estabilización de la red, para que los vehículos puedan ir construyendo sus tablas de vecinos, a partir de este instante de tiempo, los nodos emisores son capaces de enviar sus paquetes de de datos hacía el nodo destino. Para garantizar que los paquetes de datos puedan llegar al destino, los nodos emisores dejan de transmitir datos 30 segundos antes de que finalice el tiempo de simulación total. La tecnología inalámbrica empleada para

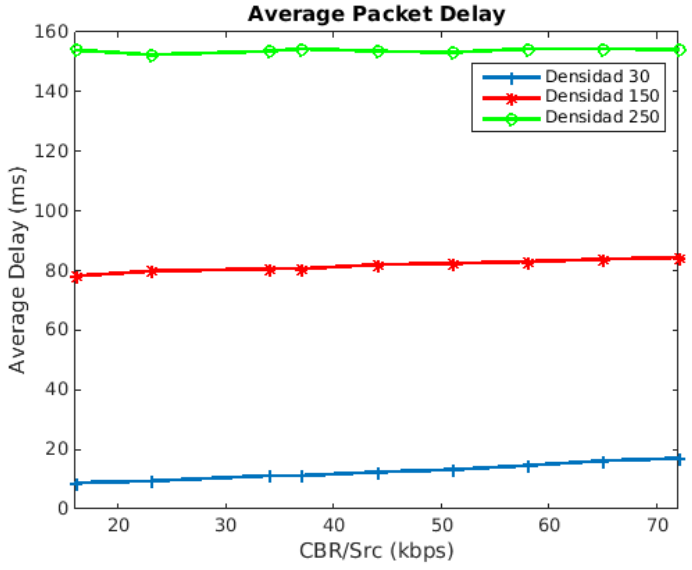
la simulación ha sido dispositivos de red basados en el estándar IEEE 802.11b con un ancho de canal de 2 Mbps y una frecuencia de 2.412 GHz.

Los experimentos de simulación se han realizado en un computador con las siguientes características:

- **Versión ns3**: ns-3-dev
- **Sistema Operativo**: Linux Mint 17.2 (64 bits)
- **Procesador**: Intel(R) Core(TM) i7-4771 a 3.5 Ghz (8 cores)
- **Memora RAM**: 32GB
- **Compilador**: gcc 4.8.4
- **Versión SUMO**: sumo-0.23

5.1.2. Impacto del número de nodos

En la figura 5.1 se muestran los resultados obtenidos cuando se va variando el número de nodos que conforman el escenario. Se puede observar que según la densidad de nodos que formen la red el retardo medio de los paquetes aumenta con una densidad elevada de nodos y cuando la densidad es baja, el retardo también lo es. Esto se debe a la influencia de la cantidad de paquetes que van circulando por la red y las colisiones que existen por los paquetes que navegan por la red. Este fenómeno queda reflejado en el PDR cuando se incrementa la densidad de nodos este parámetro sufre un decremento.



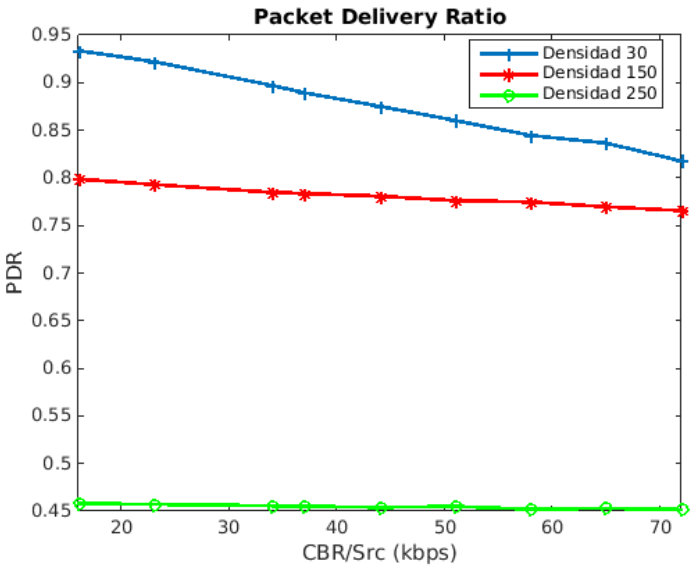


FIGURA 5.1: Impacto Densidad variando el número de nodos del escenario, calculando el retardo medio (arriba) y Packet Delivery Ratio (abajo)

5.1.3. Impacto del *Channel Shadowing*

En la figura 5.2 se estudia el comportamiento de la red según se varía la desviación estándar (σ). Al ser una distribución Gaussiana, cuando mayor σ esto implica una mayor ganancia entre Rx y Tx, por tanto esto limita el radio de cobertura de los nodos. Cuando menor σ la potencia de transmisión es menor, esto conlleva que exista un retardo menor en la entrega de los paquetes, porque van a llegar a menos nodos y la consecuencia es que los exista una gran tasa de entrega. En caso contrario al establecer σ a 12 la línea de regresión que se calcula en el modelo abarca una mayor área y por tanto, dentro de área de cobertura del nodo hay más nodos y esto supone que haya un mayor número de mensajes y que la red se sature.

5.1.4. Impacto de la velocidad de los nodos

En la figura 5.3 se estudia el comportamiento de los nodos moviéndose a altas velocidades. En este análisis se pueden observar dos factores importantes. El primero es que cuando los vehículos viajan a elevadas velocidades el protocolo sufre un declive en la métrica del PDR, esto se debe a que se se dificultan la conectividad. El segundo es que según se va incrementando la frecuencia de transmisión se va reduciendo el número de paquetes entregados debido a saturaciones en la red.

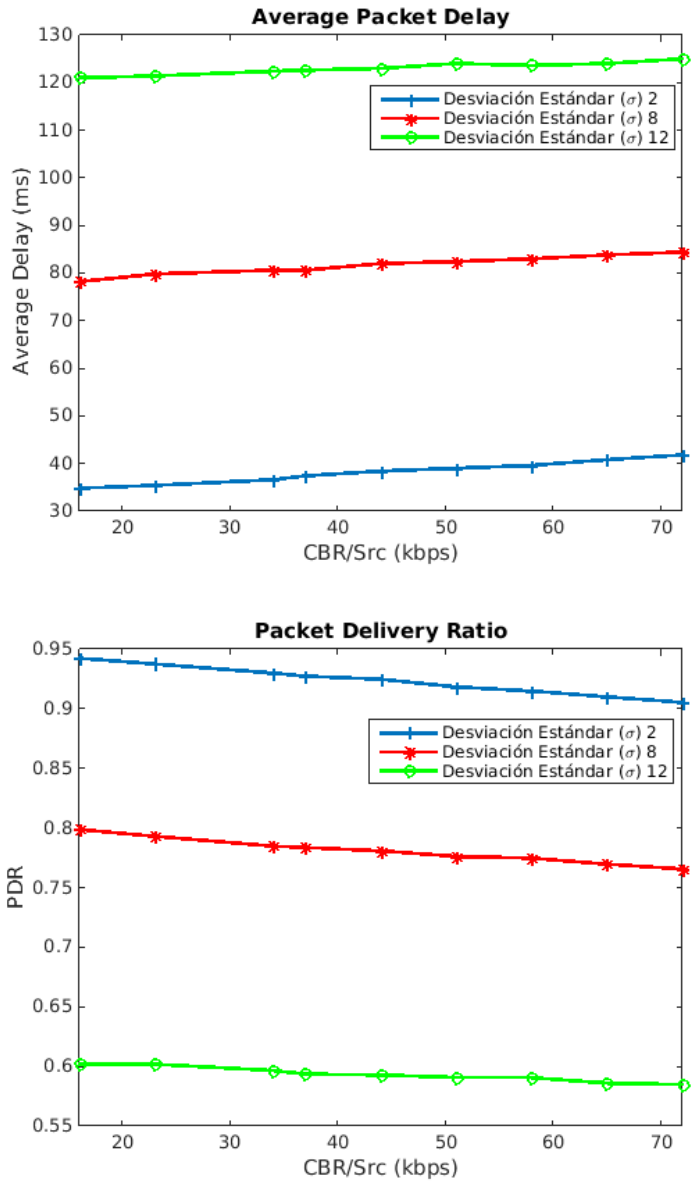


FIGURA 5.2: Impacto Channel Shadowing variando σ, calculando el retardo medio (arriba) y Packet Delivery Ratio (abajo)

5.1.5. Comparación con el protocolo original

El protocolo desarrollado se ha comparado con los resultados obtenidos por el descrito en [5], la tendencia del PDR sigue una tendencia similar 5.4. En cuanto al retardo medio en los paquetes 5.5, en el protocolo desarrollado el tiempo de espera para cada valor de σ es casi constante, esto se debe a que en el protocolo desarrollado no existe una cola para almacenar los paquetes que no se han podido enviar cuando los nodos entran en modo desconectividad. Esta cola no se ha implementado porque en estos escenarios de movilidad y donde los mensajes se están enviando constantemente a cada segundo, no

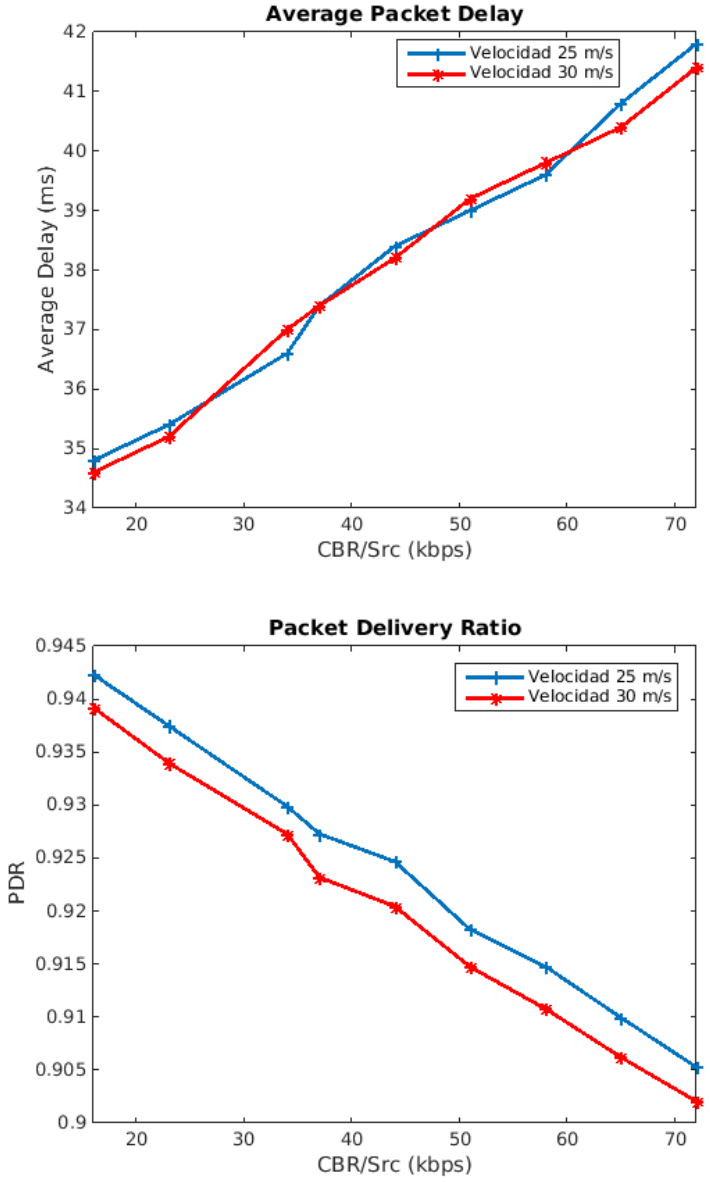


FIGURA 5.3: Impacto Velocidad variando la velocidad de movimiento de nodos del escenario, calculando el retardo medio (arriba) y Packet Delivery Ratio (abajo)

se ha considerado necesaria, incluso pueden provocar una sobrecarga innecesaria en el sistema.

Analizando los paquetes de control de Overhead 5.6 en el protocolo propuesto y comparándolo con el original se observa que hay un decremento en el valor de este parámetro, el principal motivo de esta diferencia es la forma de gestionar los mensajes de control transmitidos por el número de mensajes de datos transmitidos en ambas implementaciones.

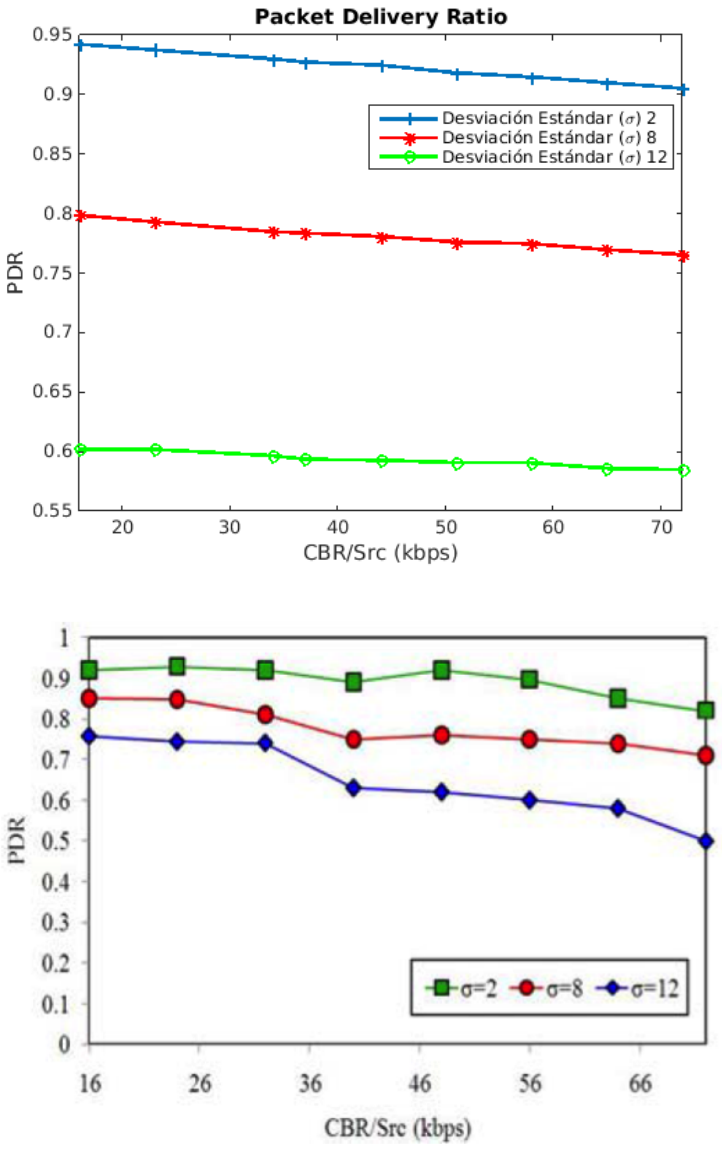


FIGURA 5.4: Comparación PDR protocolo propuesto (arriba) VS SRR (abajo)

5.1.6. Otros análisis

Debido a que el simulador ofrece gran libertad para realizar simulaciones realistas, se ha aprovechado y se han cambiado algunos parámetros. En las siguientes experimentaciones se analizan ha deshabilitado los paquetes de control *Clear To Send* (CTS) y *Request To Send* (RTS), cuando su tamaño sea inferior a 2200 Bytes. Como se puede apreciar en las gráficas 5.7, las tendencias son similares, pero debido a deshabilitar el envío de los paquetes de control el retardo medio ha disminuido, y en cambio el PDR ha sufrido un ligero aumento.

Otro análisis que se han realizado han sido cambiar el número de receptores de la red ad-hoc, hasta el momento en los análisis solo había un nodo receptor, en este experimento

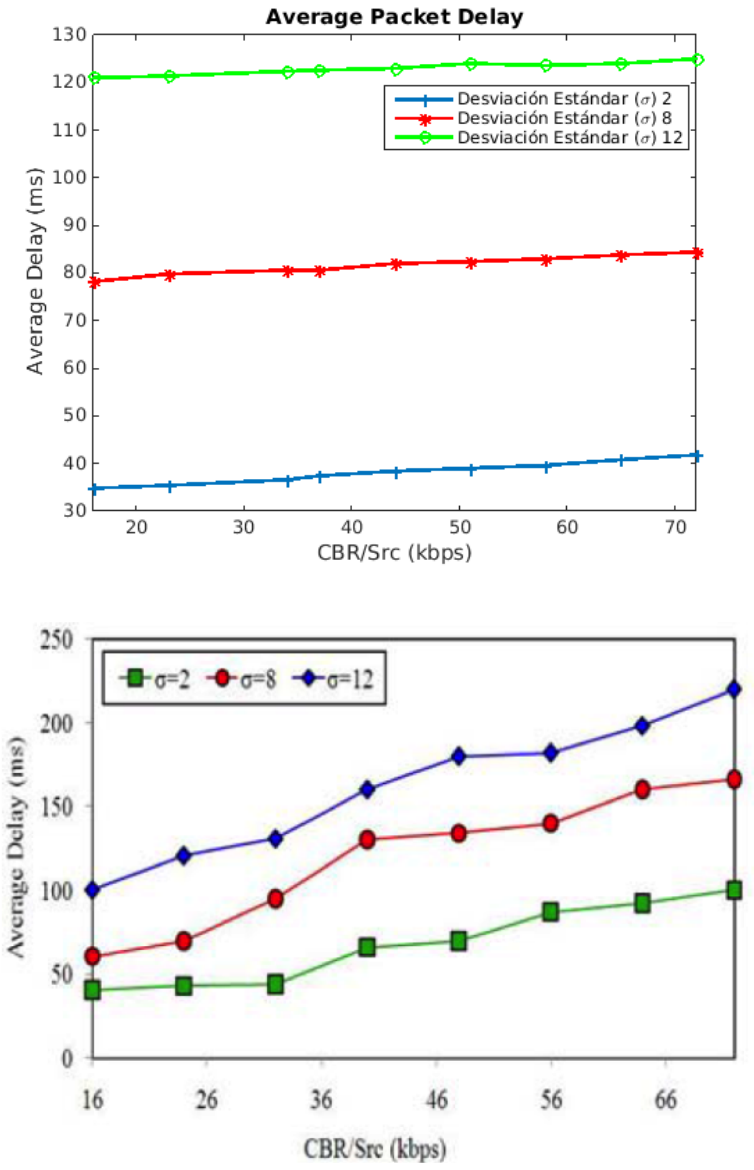


FIGURA 5.5: Comparación Delay protocolo propuesto (arriba) VS SRR (abajo)

los nodos fuente son 5 y transmiten al resto de nodos y se han deshabilitado los paquetes de control que tengan un tamaño inferior a 2200 Bytes. En cuanto a la productividad al incrementar el número de receptores cuando mayor valor de σ esta tasa es mayor. Como se puede observar en la figura 5.8 del impacto de según la variación del *Channel Shadowing*, cuando mayor valor tiene X_σ el retardo de entrega de los paquetes se incrementa, esto se debe a que mayor X_σ, el rango de nodos a los que se puede alcanzar es mayor. En cambio cuando menor valor de X_σ la tasa de ratio de entrega de paquetes es ligeramente superior en el caso de entregar a un destino que a todos los nodos restantes 5.9. Por último, al haber un mayor número de nodos receptores el retardo *End-To-End* ha sufrido una ligera mejora decrementado el retardo medio, figura 5.10. En cuanto a las otras métricas comparadas tienen una tendencia similar.

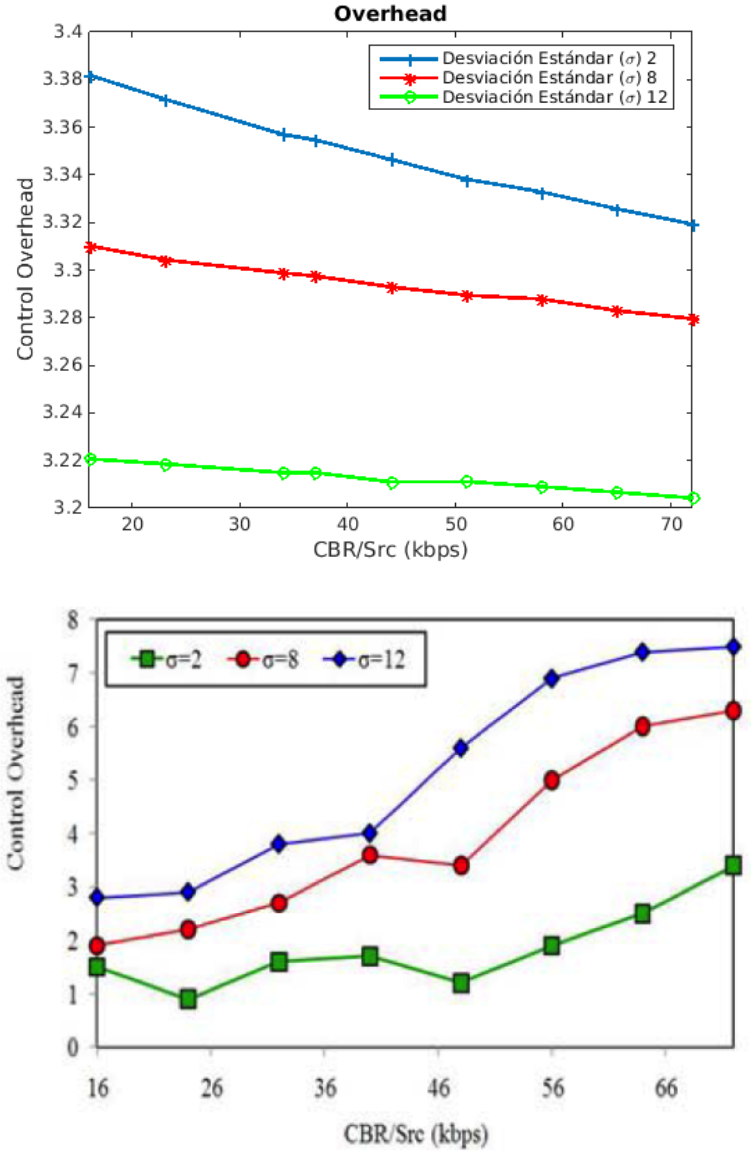


FIGURA 5.6: Comparación Overhead protocolo propuesto (arriba) VS SRR (abajo)

En la figura 5.11, se muestran los resultados de la simulación dependiendo de la densidad de vehículos que haya en la red. Cuando mayor densidad de nodos, el retardo se incrementa, esto es debido a la gran cantidad de paquetes que están viajando a lo largo de la red para alcanzar su destino. El principal factor que influye en este experimento son las colisiones entre paquetes. En cuanto a la entrega de paquetes cuando la densidad de los nodos disminuye la entrega de los paquetes es bastante elevada, se puede observar que conforme la velocidad de transmisión va aumentando, la tendencia de entrega va disminuyendo, esto se debe a que conforme aumenta la velocidad el ratio de alcance va disminuyendo paulatinamente. Finalmente, en este experimento, se puede observar que la mejor tasa de productividad existe cuando hay una gran densidad de paquetes, esto se debe a que hay mucho mas intercambio de paquetes en altas densidades que en bajas.

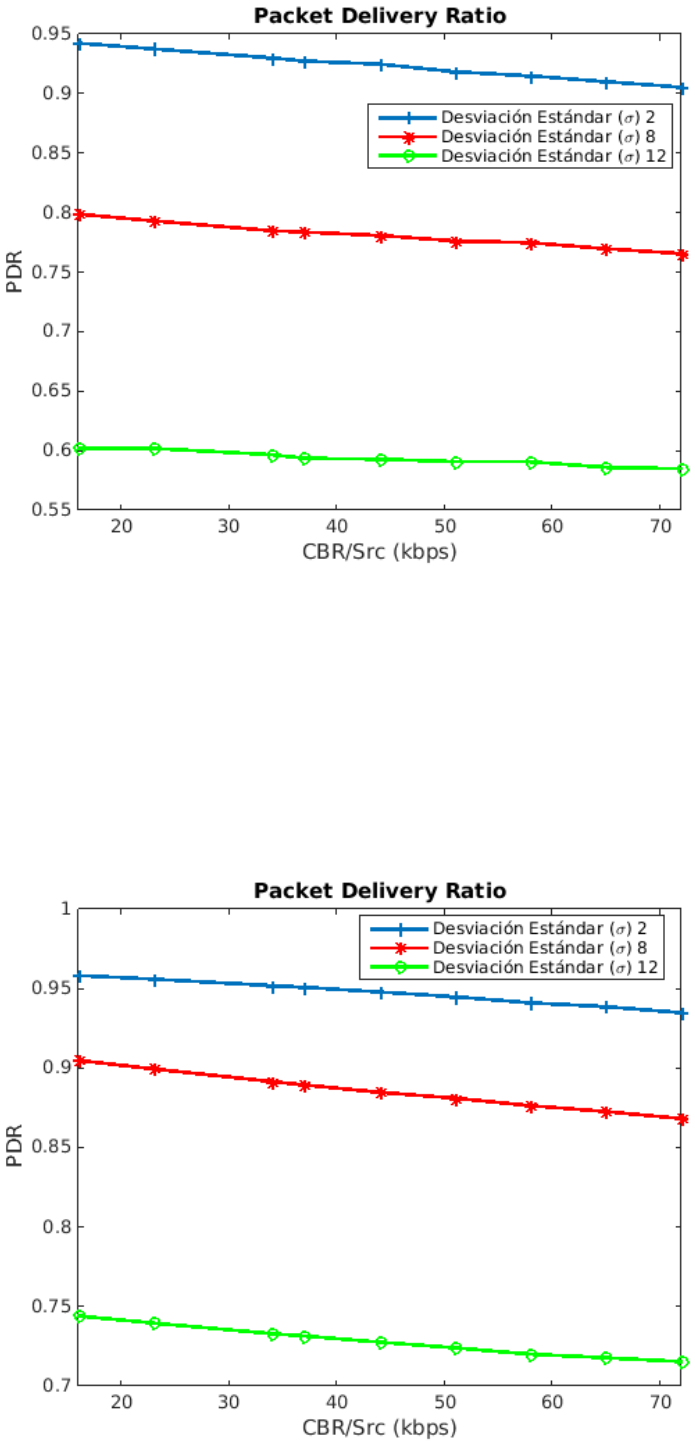


FIGURA 5.7: Comparación PDR paquetes de control habilitados (arriba) VS deshabilitados (abajo)

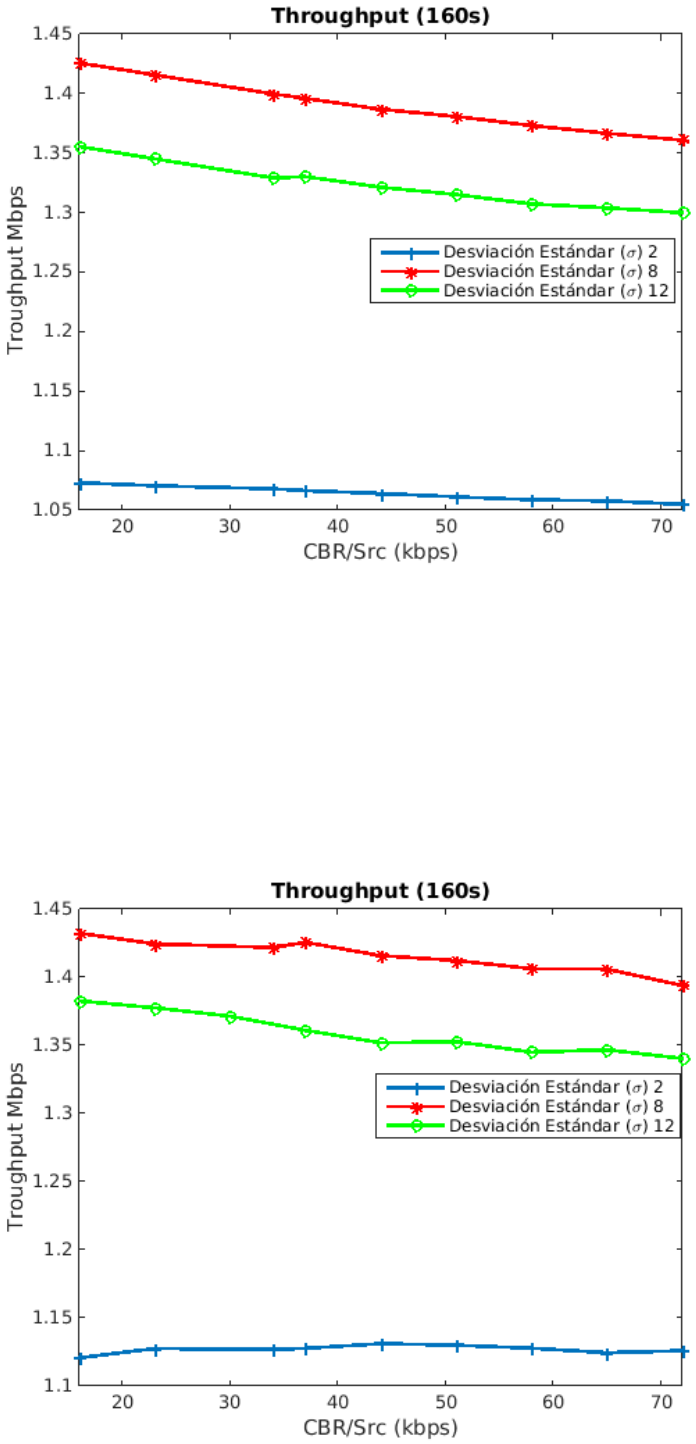


FIGURA 5.8: Comparación Throughput sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo)

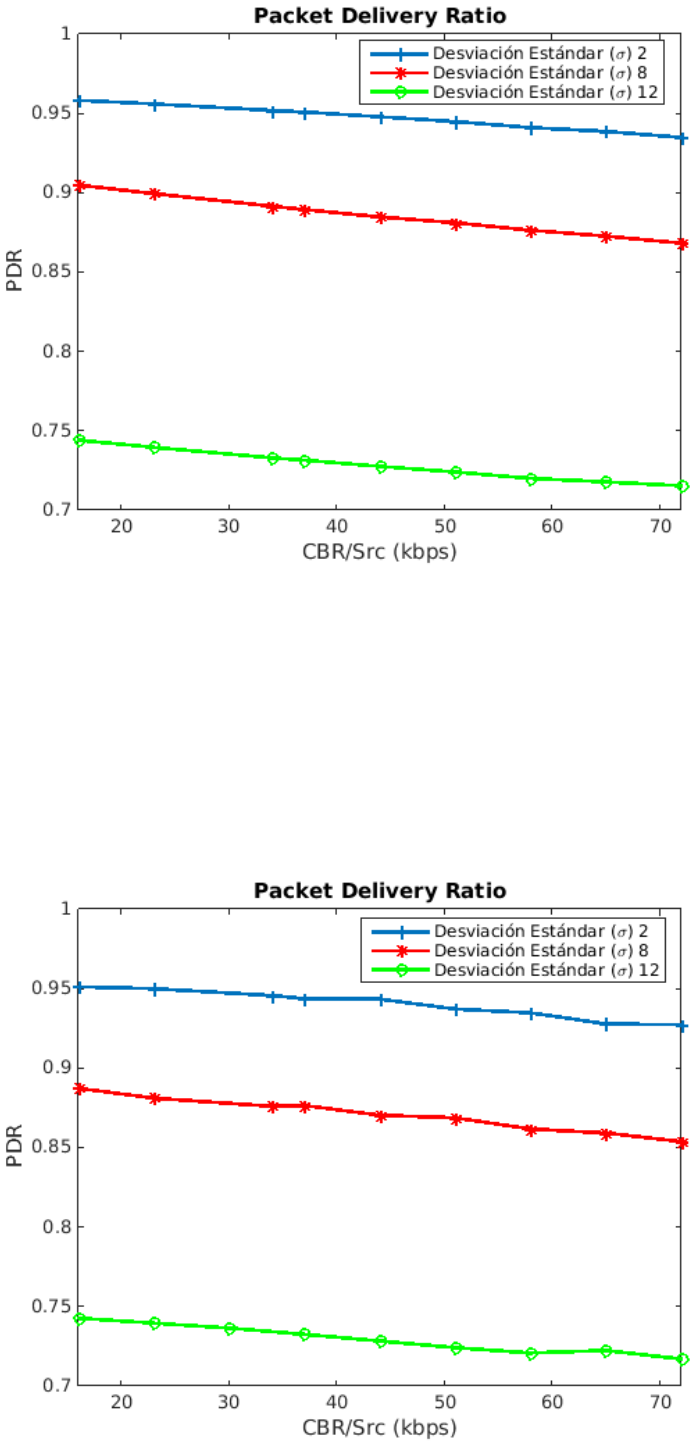


FIGURA 5.9: Comparación PDR sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo)

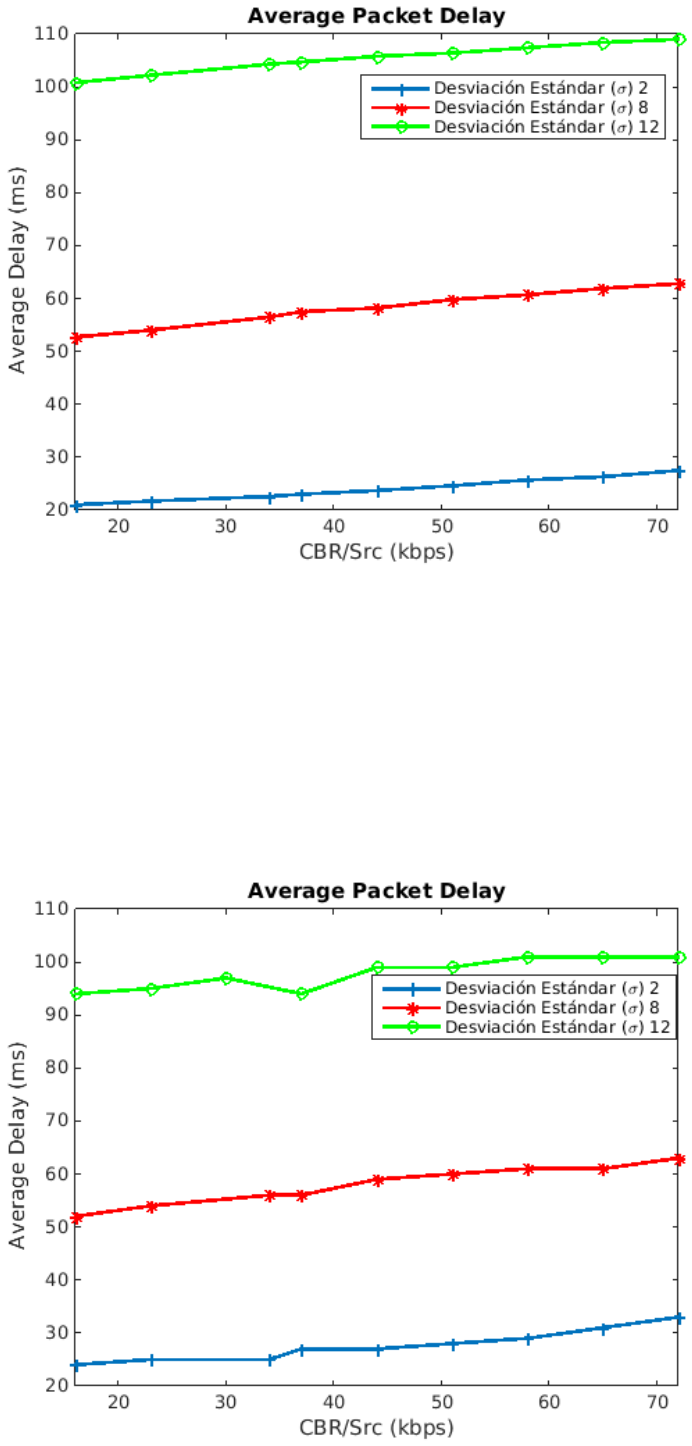


FIGURA 5.10: Comparación Delay sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo)

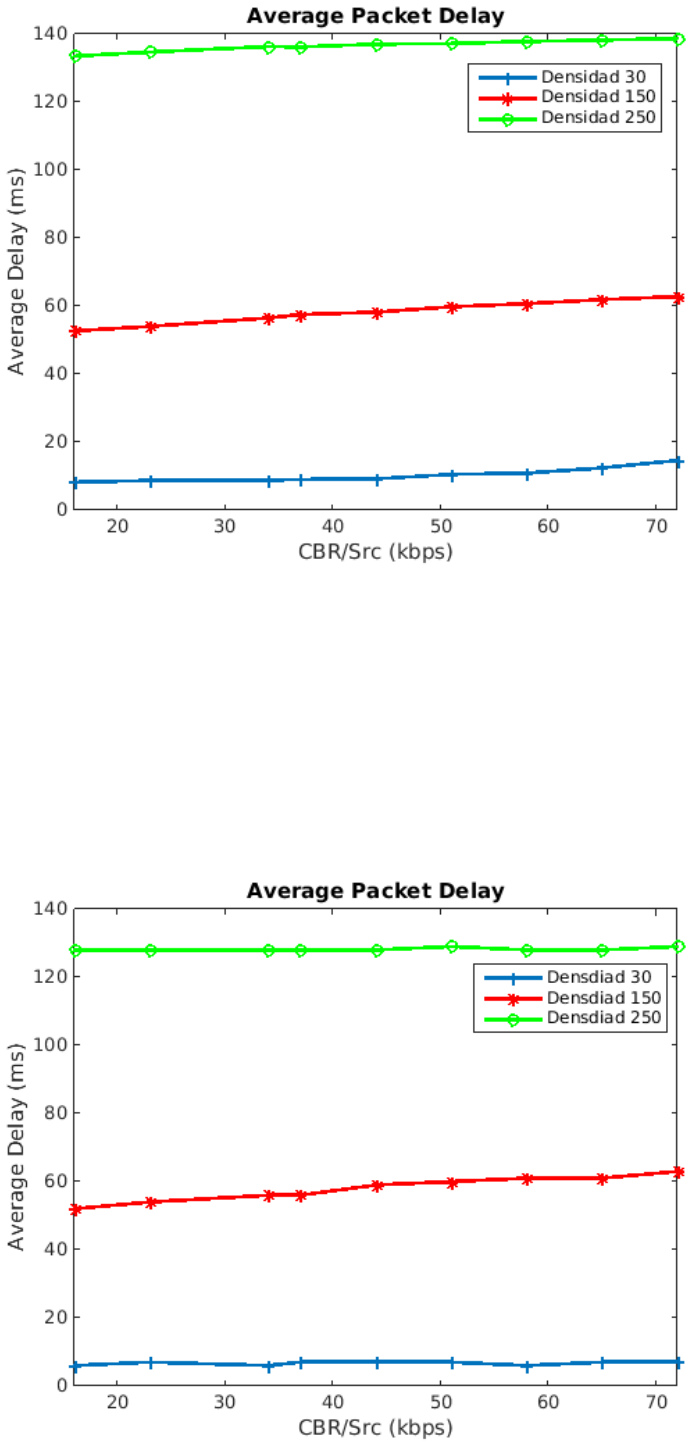


FIGURA 5.11: Comparación Densidad sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo)

En el análisis del impacto de la velocidad a la que se mueven los vehículos, se ha variado de 25 m/s a 30 m/s. Se puede observar que según aumenta la velocidad de transmisión, el retardo de los paquetes es mucho mayor 5.12. En caso contrario, en la entrega de paquetes se puede observar una reducción, estos fenómenos se deben a que según va aumentando la velocidad de transmisión el espectro es menor, y además añadiendo la velocidad de los nodos esto provoca que los mensajes no lleguen a su destino por posibles desconexiones 5.13.

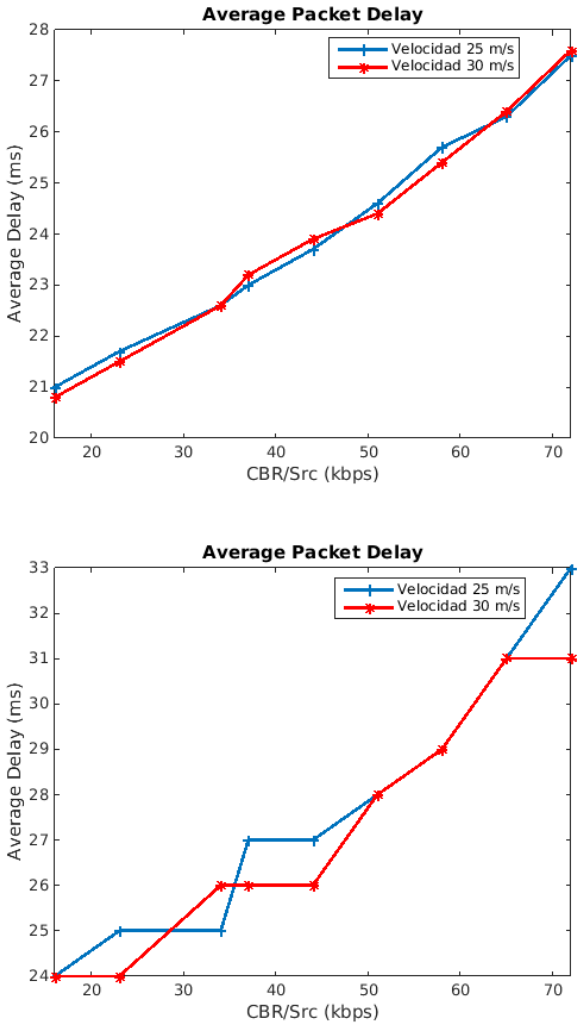


FIGURA 5.12: Comparación retardo sin paquetes de control con 1 destino (arriba) VS destino todos los nodos de la red (abajo)

5.2. Evaluación Beaglebone Black

A continuación se presentan los experimentos realizados con la BeagleBoneBlack y la ROBOCape empleando los nodos de ROS descritos en el apartado 4.4 que permitirán evaluar la implementación sobre hardware real de las partes del sistema necesarias para

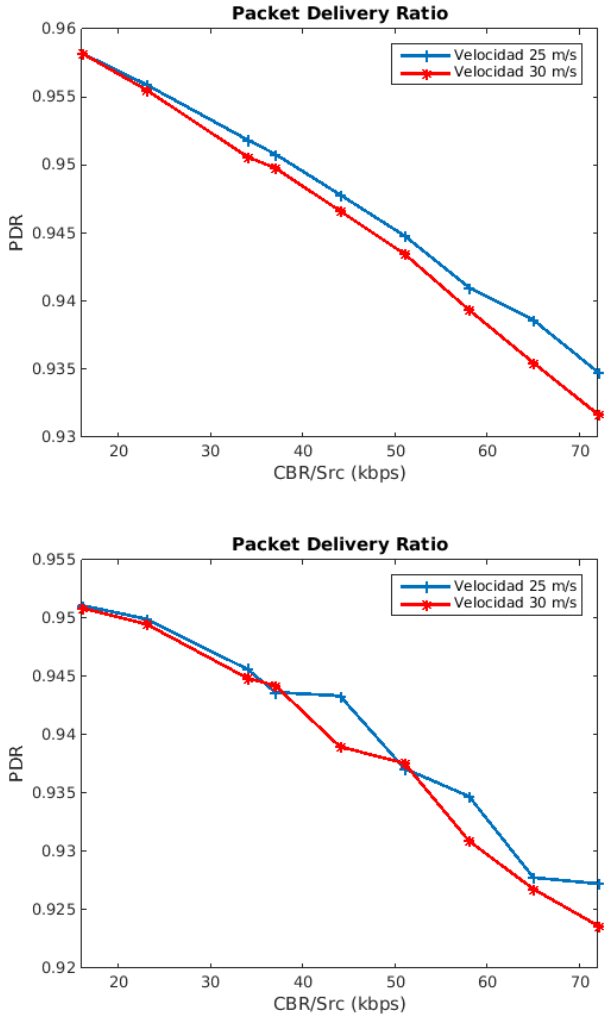


FIGURA 5.13: Comparación Velocidad sin paquetes de control con 1 destino (abajo) VS destino todos los nodos de la red (abajo)

complementar los protocolos de comunicación estudiados previamente basados en GPS. Estos experimentos han consistido en ubicar la BeagleBoneBlack en el vehículo eléctrico e ir por dentro del campus universitario dando vueltas para obtener los resultados del GPS. Durante estos recorridos se han realizado diversas acciones, como por ejemplo acelerar y desacelerar, además de realizar giros bruscos a derechas e izquierdas para tomar las mediciones de la IMU, la velocidad máxima alcanzada ha sido de 30 Km/h. El GPS estaba funcionando a una frecuencia de 0.5 Hz y la IMU a 1 Hz. En las figuras 5.14 y 5.14 se muestran algunas de las rutas obtenidas con el GPS. Para poder representar las coordenadas en Google Earth se han convertido las coordenadas del GPS de grados sexadecimales a grados decimales.

Como se ha podido observar el rendimiento del GPS es bastante bueno aun funcionando a un bajo rendimiento. Como ocurre con todo dispositivo GPS existe un rango de error. Este error se observa cuando se trazan las trayectorias sobre el mapa real.

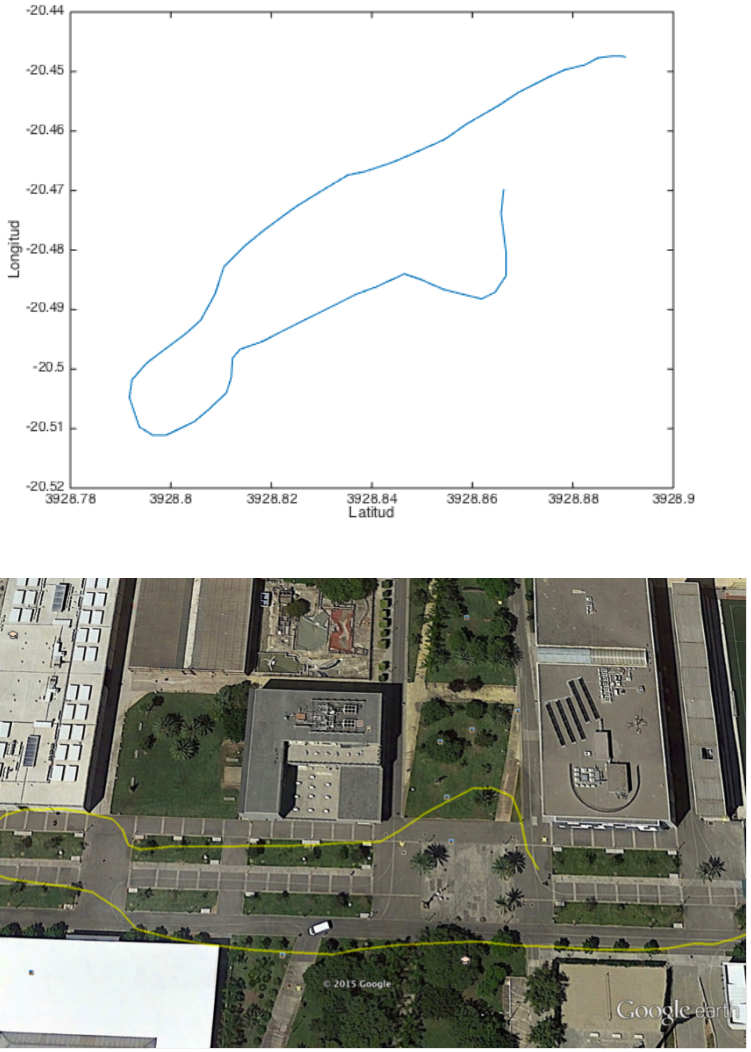


FIGURA 5.14: Ruta 1 obtenida por el GPS

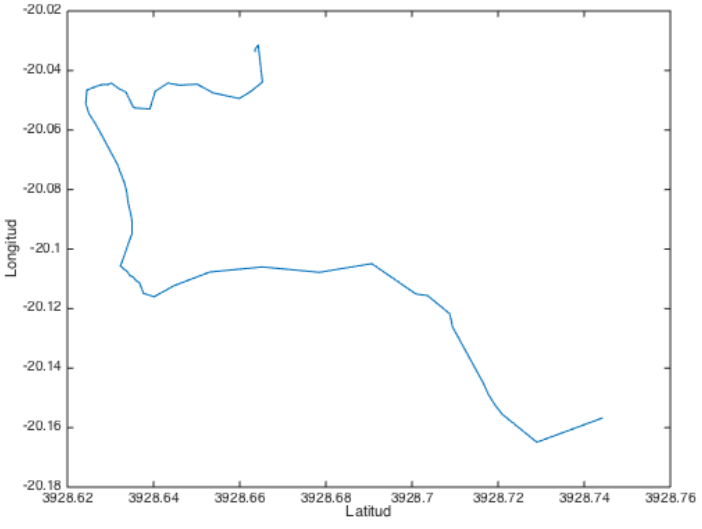


FIGURA 5.15: Ruta 2 obtenida por el GPS

En la figura 5.16 se muestra la transmisión de las coordenadas del GPS de la Beaglebone a través de un dispositivo inalámbrico Zigbee.

```
[ INFO] [1437945672.736088011]: MessageID = [GPGGA]
[ INFO] [1437945672.745787928]: Longitud = [20.729900]
[ INFO] [1437945672.752440011]: Latitud = [3928.957764]
[ INFO] [1437945672.758368386]: Alt_Mar = [97.400002]
```

```
GPGGA:
20.7317:3928.96:96.3
GPGGA:
20.7316:3928.96:96.2
```

FIGURA 5.16: Datos obtenidos del GPS

En la figura 5.17 se muestran los datos publicados en ROS para en referencia a la IMU.

```
[ INFO] [1437947118.445187354]: Acc_X = [3.988770]
[ INFO] [1437947118.450523771]: Acc_Y = [0.009521]
[ INFO] [1437947118.455666812]: Acc_Z = [0.891846]
[ INFO] [1437947118.460428312]: Gir_X = [0.320435]
[ INFO] [1437947118.465022896]: Gir_Y = [1.052856]
[ INFO] [1437947118.469635104]: Gir_Z = [0.892639]
```

FIGURA 5.17: Datos publicados de la IMU

Gracias al trabajo realizado con ROS y la tarjeta Beaglebone, se vio la oportunidad de colaborar en el desarrollo de la aplicación que funcionó en las competición de la CEABOT que organiza la CEA en las Jornadas de Automática, realizadas en Bilbao. El robot empleado fue un Bioloid Figura 5.18 el cual era capaz de comunicarse con la BeagleBone a través de su puerto serie. Las principales contribuciones realizadas fueron el desarrollo de la comunicación a través de la UART, para que cuando el robot solicitase información ésta se la pudiese proporcionar. La información que recibía el robot era obtenida a través de la ROBOCape, que incorporaba la sensorización oportuna. La función de la BeagleBone no era otra que ayudar al robot a que se mantuviese estable y poder realizar la computación que por limitaciones de hardware el robot no podia realizar. También se realizó una aplicación que fuese capaz de reconocer una serie de

códigos QR. Estos códigos servirian para guiar al robot a través de una serie de pruebas. Figura 5.18

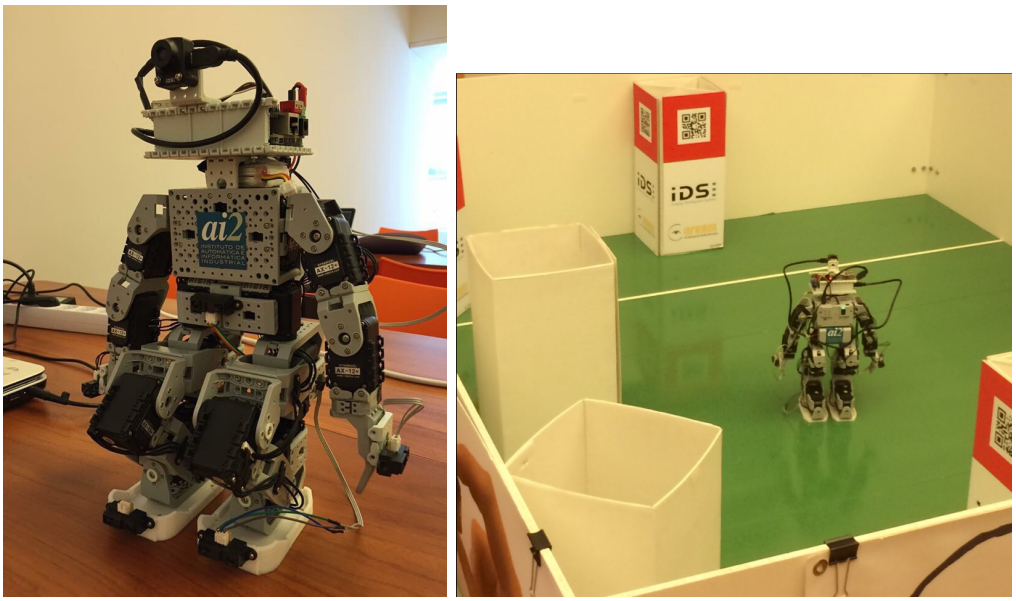


FIGURA 5.18: en la imagen de la izquierda se puede observar el robot Bioloid con la cámara y la BeagleBoneBlack, en la imagen de la derecha se puede observar una instantánea del robot compitiendo.

Cabe destacar la buena actuación que realizó el robot quedando en cuarta posición, a muy pocos puntos de los ganadores.

Capítulo 6

Conclusiones, futuros trabajos y publicaciones

6.1. Conclusiones

El principal de este Trabajo Final de Máster ha consistido en desarrollar un modelo de protocolo de encaminamiento para redes ad-hoc vehiculares. Para ello, se ha realizado un exhaustivo estudio sobre los diferentes protocolos de enrutamiento existentes para redes VANETs. Después de este estudio y elegir una propuesta para implementar y con que herramientas trabajar, se ha modelado y desarrollado un protocolo de red en el simulador ns-3. Se ha visto la necesidad de emplear otros tipos de simuladores de movilidad, el elegido ha sido SUMO. La elección de este simulador ha sido porque permite un acoplamiento con otros sistemas. En esta línea se ha propuesto un ecosistema completo para el desarrollo y evaluación de este tipo de protocolos, y con el cual se ha desarrollado el modelo y llevado a cabo las campañas de simulación oportunas, validándose los modelos desarrollados, y obteniendo resultados que servirán para la propuesta de futuras ampliaciones. De esta forma se ha podido constatar como el protocolo mantiene los retardos extremo a extremo en función del tráfico de la red, lo cual indica que escala correctamente. Por otro lado, tambien se ha advertido que el parámetro X_σ del modelo de propagación físico influye significativamente en el retardo y tasa PDR por lo que será un factor a tener en muy cuenta en el diseño de estos sistemas.

Para poder implementar los protocolos basados en GPS se han desarrollado diversos nodos con el middleware ROS, los cuales están funcionando en un dispositivo BeagleBone Black. Estos nodos desarrollados en esta plataforma permiten a otros dispositivos que funcionen con ROS suscribirse y obtener la información suministrada y trabajar de manera distribuida para no sobrecargar la CPU de los dispositivos. De esta forma queda

demostrado que se pueden tener dispositivos con altos rendimientos y con muy buena precisión a bajo coste.

Con las herramientas empleadas se ha obtenido un ecosistema hardware/software escalable y con el que poder realizar una serie de desarrollos y posteriormente poder validar los resultados en simulación y verificar su comportamiento en equipos reales, de esta forma se obtiene un feedback entre implementación en simulador y desarrollo en equipos reales.

6.2. Publicaciones y otras contribuciones

En la realización de este trabajo ha dado lugar a dos publicaciones en congresos; *Workshop on Innovation on Information and Communication Technologies (ITACA-WIICT 2015)* y en las *XXXVI Jornadas de Automática - 2015*:

- J. Navarro, P. Cardós, J.V. Capella, A. Bonastre, R. Ors and A. Valera. Towards a new WSN/VANET development ecosystem focused on reliability and intelligent techniques based on simulation and real HW implementation. *Workshop on Innovation on Information and Communication Technologies (WIICT'15)*, Valencia 2015.
- J. Navarro, J.V. Capella, M. Bosch, A. Soriano, M. Albero and A. Valera. Desarrollo de una plataforma HW/SW para el control de vehículos automáticos. *XXXVI Jornadas de Automática - 2015* CEA-IFAC, Bilbao 2015.

Además, durante el desarrollo de este trabajo se ha realizado una colaboración para el desarrollo de software para la competición de las XXXVI Jornadas de Automática CEABOT-15, para el equipo que participaba representando a la Universidad Politécnica de Valencia, desarrollando las aplicaciones para la lectura de la Inertial Measurement Unit que incorporaba la placa BeagleBone Black y la decodificación de códigos QR con la cámara IDS UEye XS.

6.3. Futuros trabajos

Este Trabajo Final de Máster da lugar al proyecto de tesis doctoral: "Contribución al Diseño de Redes Inalámbricas de sensores de alta fiabilidad para el control de robots móviles y vehículos ligeros". En esta línea, los futuros trabajos a realizar en esta tesis serán:

Se ampliará el modelo con la incorporación de nuevas características como por ejemplo fiabilidad, tolerancia de fallos. Además de aplicar técnicas robustas para el encaminamiento de las rutas, ampliando los parámetros a considerar (calidad de la señal y consumo energético) y estudiando nuevas técnicas. Así pudiendo evaluarlo con el ecosistema propuesto para su posterior implementación en hardware real.

Con los resultados obtenidos con el dispositivo GPS se ajustará la frecuencia de lectura para así poder obtener el máximo rendimiento sin llegar a saturar la plataforma hardware.

Posteriormente, se trabajará con la navegación automática de los vehículos ligeros, usando las propuestas desarrolladas para integrarlas con dicho sistema de navegación autónoma que ha sido desarrollo por el grupo. A este sistema se le dotará de mayores prestaciones y características que ayudarán a comunicarse con los vehículos. Así, adecuando los protocolos de comunicaciones a empleados en función de los requisitos de la aplicación que se indique. Y por tanto se estudiará su fiabilidad una vez estén adaptados.

Bibliografía

[1] L. Banda, M. Mzyece, and G. Noel. Ip mobility support: Solutions for vehicular networks. *Vehicular Technology Magazine, IEEE*, 7(4):77–87, Dec 2012. ISSN 1556-6072. doi: 10.1109/MVT.2012.2203881.

[2] David Chunhu Li, Li-Der Chou, Li-Ming Tseng, Yi-Ming Chen, and Kai-Wei. Kuo. A bipolar traffic density awareness routing protocol for vehicular ad hoc networks. *Mobile Information Systems*, 2015(2015):1–12, 2015. doi: 10.1155/2015/401518. URL http://dx.doi.org/10.1155/2015/401518.

[3] J. Toutouh, J. Garcia-Nieto, and E. Alba. Intelligent olsr routing protocol optimization for vanets. *Vehicular Technology, IEEE Transactions on*, 61(4):1884–1894, May 2012. ISSN 0018-9545. doi: 10.1109/TVT.2012.2188552.

[4] Jamal Toutouh, Sergio Nesmachnow, and Enrique Alba. Fast energy-aware olsr routing in vanets by means of a parallel evolutionary algorithm. *Cluster Computing*, 16(3):435–450, 2013. ISSN 1386-7857. doi: 10.1007/s10586-012-0208-9. URL http://dx.doi.org/10.1007/s10586-012-0208-9.

[5] 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.

[6] J. E. Smith A. E. Eiben. *Introduction to Evolutionary Computing*. Springer, 2003.

[7] Jaime Lloret Kamalrulnizam Abu Bakar Zaitul M. Zainuddin Kayhan Zrar Ghafoor, Marwan Aziz Mohammed. Routing protocols in vehicular ad hoc networks: Survey and research challenges. *Network Protocols and Algorithms*, 5(4):39–83, Dec 2013. ISSN 1943-3581. doi: 10.5296/npa.v5i4.4134. URL http://dx.doi.org/10.5296/npa.v5i4.4134.

[8] Daqiang Zhang, Zhijun Yang, Vaskar Raychoudhury, Zhe Chen, and Jaime Lloret. An energy-efficient routing protocol using movement trends in vehicular ad

hoc networks. *The Computer Journal*, pages 1–9, 2013. doi: 10.1093/comjnl/bxt028. URL http://comjnl.oxfordjournals.org/content/early/2013/03/21/comjnl.bxt028.abstract.

[9] M. Al-Rabayah and R. Malaney. A new scalable hybrid routing protocol for vanets. *Vehicular Technology, IEEE Transactions on*, 61(6):2625–2635, July 2012. ISSN 0018-9545. doi: 10.1109/TVT.2012.2198837.

[10] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. pages 134–146, 2003. doi: 10.1145/938985.939000. URL http://doi.acm.org/10.1145/938985.939000.

[11] M. Al-Rabayah and R. Malaney. Scalable hybrid location-based routing in vehicular ad hoc networks. pages 1–5, Sept 2011. ISSN 1090-3038. doi: 10.1109/VETECF.2011.6093116.

[12] ns-2. http://www.isi.edu/nsnam/ns/. [Online; accedido Septiembre-2015].

[13] Mallba. http://neo.lcc.uma.es/mallba/easy-mallba/html/mallba.html. [Online; accedido Septiembre-2015].

[14] D. García C. Gómez and J. Paradells. Improving performance of a real ad hoc network by tuning olsr parameters. *Proc. 10th IEEE ISCC*, pages 16–21, 2005.

[15] J. García-Nieto, J. Toutouh, and E. Alba. Automatic tuning of communication protocols for vehicular ad hoc networks using metaheuristics. *Engineering Applications of Artificial Intelligence*, 23(5):795–805, 2010. ISSN 0952-1976. doi: 10.1016/j.engappai.2010.01.012. URL http://www.sciencedirect.com/science/article/pii/S0952197610000308. Advances in metaheuristics for hard optimization: new trends and case studies.

[16] C.A.T.H. Tee and A. Lee. Adaptive reactive routing for vanet in city environments. pages 610–614, Dec 2009. doi: 10.1109/I-SPAN.2009.39.

[17] J. Zhao and G. Cao. Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks. *The 25th Conference on Computer Communications (IEEE INFOCOM'06)*, Apr 2006.

[18] A. Fagg J. Davis and B Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. *International Symposium on Wearable Computing*, Oct 2001.

[19] Zeinab Rezaeifar, Faramarz Hendessi, BehrouzShahgholi Ghahfarokhi, and T.Aaron Gulliver. A reliable geocast routing protocol for vehicular ad hoc networks. *Wireless*

Personal Communications, 83(1):281–295, 2015. ISSN 0929-6212. doi: 10.1007/s11277-015-2393-3. URL http://dx.doi.org/10.1007/s11277-015-2393-3.

[20] Li-Der Chou, Jyun-Yan Yang, Ying-Cheng Hsieh, Der-Chyn Chang, and Chi-Feng Tung. Intersection-based routing protocol for vanets. *Wireless Personal Communications*, 60(1):105–124, 2011. ISSN 0929-6212. doi: 10.1007/s11277-011-0257-z. URL http://dx.doi.org/10.1007/s11277-011-0257-z.

[21] Jist/swans. http://jist.ece.cornell.edu/. [Online; accedido Septiembre-2015].

[22] Ns-3 team. https://www.nsnam.org/, 2015. [Online; accedido Septiembre-2015].

[23] Institute of Transportation Systems. Sumo - simulation of urban mobility. http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/, 2015. [Online; accedido Septiembre-2015].

[24] Ros - robot operating system. http://www.ros.org/, 2015. [Online; accedido Septiembre-2015].

[25] Beagleboard. http://www.beagleboard.org/, 2015. [Online; accedido Septiembre-2015].

[26] Miguel Albero. Robocape for beagleboneblack. Reporte interno AI2, 2014.

[27] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, New Jersey, 1996.

[28] Bluegiga. https://www.bluegiga.com/en-US/products/wf111-wifi-module/, 2015. [Online; accedido Septiembre-2015].

[29] António Fonseca, André Camões, and Teresa Vazão. Geographical routing implementation in ns3. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, SIMUTOOLS '12, pages 353–358, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-1-4503-1510-4. URL http://dl.acm.org/citation.cfm?id=2263019.2263075.

[30] Blacklib. http://blacklib.yigityuce.com/, 2015. [Online; accedido Septiembre-2015].

[31] InvenSense Inc. Mpu-6000 and mpu-6050 register map and descriptions. Datasheet, 2012.

[32] Freescale Semiconductor. Xtrinsic mpl3115a2 i2c precision altimeter. Datasheet, 2013.

[33] GlobalTop Technology Inc. Fgpmmopa6h gps standalone module data sheet. Datasheet, 2011.