
**Proyecto Final de Carrera**

**Ingeniería en Informática**

**Curso 2010/2011**

# Simulador de VANETs para evaluar técnicas de gestión de datos basadas en agentes móviles

**Eduardo López García**

Director: Sergio Ilarri Artigas
Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior
Universidad de Zaragoza

Diciembre de 2010

# Resumen

Hoy en día existe un gran interés en el desarrollo de sistemas de asistencia a conductores en carretera basados en VANETs (Vehicular Ad-hoc Networks) que usan comunicaciones de corto alcance, como IEEE 802.11, para el intercambio dinámico de información entre vehículos cercanos. Sin embargo, antes de la puesta en marcha de estos sistemas se necesita hacer las pruebas pertinentes sobre un simulador. Por lo tanto, el objetivo de este proyecto consiste en desarrollar una plataforma de simulación de escenarios donde distintos automóviles regidos por políticas de movilidad modificables se comunican entre sí para intercambiar información que consideren relevante usando distintas estrategias.

El proyecto desarrollado ha mejorado un prototipo de simulación de VANETs ya existente, desarrollado en el grupo SID (Sistemas de Información Distribuidos de la Universidad de Zaragoza), cuyo objetivo es la evaluación de técnicas basadas en agentes móviles para monitorizar áreas geográficas usando vehículos. El prototipo inicial se desarrolló para la evaluación de unas técnicas concretas y con una interfaz gráfica muy limitada, no siendo posible su utilización en distintos escenarios sin un importante esfuerzo por parte del usuario. Por tanto, era preciso extenderlo, generalizarlo y mejorarlo. El principal objetivo de la herramienta de simulación desarrollada es facilitar la evaluación de distintas técnicas de procesamiento de consultas y gestión de datos en redes de vehículos.

El simulador desarrollado permite asignar a los vehículos distintas estrategias de movilidad y de esta forma obtener un mayor abanico de diferentes simulaciones y una mejor representación del comportamiento real de los vehículos. Las políticas de movilidad implementadas son: una variación de Gauss-Markov, una variación de Manhattan y dos estrategias de movilidad basadas en rutas.

La plataforma de simulación cuenta con una interfaz de usuario amigable que permite definir escenarios de simulación y controlar las simulaciones de forma sencilla, característica que no poseía el prototipo implementado por el SID.

El primer prototipo diseñado por el SID sólo permitía realizar las simulaciones sobre un único mapa (ciudad de Valenciennes en Francia), por lo que se consideró importante desarrollar una herramienta que permitiera añadir nuevos mapas para realizar simulaciones. Esta herramienta permite descargar mapas del tamaño y localización que el usuario desee y de diversas fuentes (OpenStreetMaps, MicrosoftMaps y GoogleMaps) y aplicarles filtros geográficos.

El simulador cuenta con un sistema de carga y almacenamiento de los parámetros de simulación, de forma que se pueden repetir simulaciones con los mismos parámetros y configurar simulaciones de forma rápida a través de un fichero XML.

El sistema cuenta con una herramienta que permite la importación de trazas GPS reales. De esta forma, los vehículos pueden reproducir movimientos que otros vehículos han realizado en la realidad y así conseguir simulaciones más realistas que las conseguidas con las políticas de movilidad y dotar al simulador de la capacidad de utilizar información externa. Además, el usuario puede realizar filtros sobre las trazas.

El sistema cuenta también con una herramienta que permite realizar simulaciones con la existencia de elementos fijos de soporte a la VANET, como nodos de soporte para las comunicaciones, *relays*, en las carreteras.

Por todo ello, se puede concluir que se ha cumplido el objetivo de proporcionar a los integrantes del SID una herramienta útil para realizar pruebas de sus investigaciones en un entorno con una interfaz amigable, y que cuenta con las funcionalidades necesarias para disponer de un amplio abanico de configuraciones para las simulaciones.

# ÍNDICE

# ÍNDICE DE ANEXOS

# ÍNDICE DE FIGURAS

## Memoria



## Anexo A

**Anexo B**



**Anexo C**



**Anexo D**

**Anexo E**



**Anexo F**

# ÍNDICE DE TABLAS

**Memoria**

# 1. INTRODUCCIÓN

Esta memoria tiene como objetivo documentar el proyecto fin de carrera (PFC o proyecto) “Simulador de VANETs para evaluar técnicas de gestión de datos basadas en agentes móviles”. En esta primera sección se pretende describir el contexto adecuado para facilitar la comprensión del resto del documento. Esta sección está compuesta por diversos apartados: la motivación que da lugar al proyecto, los objetivos del mismo, el contexto en el que se realiza, las tecnologías utilizadas, las fases del proyecto y por último se explica la estructura de esta memoria.

## 1.1. Contexto y motivación

Hoy en día existe un gran interés en el desarrollo de protocolos y procesamiento de datos para redes de vehículos (VANETs [53]). En una VANET, los vehículos forman una red ad hoc en la cual diferentes tipos de información pueden ser intercambiados entre los vehículos siempre y cuando se encuentren lo suficientemente cerca como para poder establecer una conexión entre ellos. Esta forma de comunicación de corto alcance entre vehículos puede tener numerosos usos: aviso sobre la existencia de accidentes, información sobre la existencia de gasolineras, hoteles, lugares turísticos, información sobre la existencia de plazas libres para estacionar el vehículo, monitorización de una determinada zona, etc.

Debido al gran interés que despierta este tipo de redes existen numerosas investigaciones que tratan de sacar el mayor provecho de ellas [1, 2, 3], otras que investigan sobre protocolos de comunicación entre los vehículos [4, 5, 6], otras que plantean diversas estrategias de movilidad para los vehículos para que éstos se comporten de la forma más realista posible [7, 8, 9 ,10] y otras que analizan los diversos simuladores de redes de comunicaciones y redes de vehículos existentes [11].

En la actualidad existe una amplia cantidad de vehículos dotados de algún tipo de sensor para medir alguna característica del ambiente, como por ejemplo sensores de temperatura. Por lo tanto, para medir la temperatura de una determinada zona, con una VANET y unos vehículos dotados de sensores de temperatura se podría monitorizar la zona que se desease. Sin embargo, antes de la puesta en marcha de algún experimento de este tipo se necesitan hacer las pruebas pertinentes sobre un simulador.

El actual PFC surge para mejorar un prototipo de simulación de VANETs ya existente y dotarlo de nuevas funcionalidades que lo complementen. Dicho prototipo fue desarrollado por el grupo SID (Sistemas de Información Distribuidos de la Universidad de Zaragoza) con el objetivo de evaluar distintas técnicas basadas en agentes móviles para monitorizar áreas geográficas usando vehículos. El prototipo inicial se desarrolló para la evaluación de unas técnicas concretas y con una interfaz gráfica muy limitada, no siendo posible su utilización en distintos escenarios sin un importante esfuerzo por parte del usuario.

Este PFC ha sido desarrollado en el área de Lenguajes y Sistemas Informáticos del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza.

### 1.1.1. Prototipo anterior del SID

El grupo SID (Sistemas de Información Distribuidos de Universidad de Zaragoza) desarrolló un prototipo para evaluar diferentes técnicas basadas en agentes móviles (para más información sobre los agentes móviles consúltese el Anexo E "Material relacionado") para la monitorización de un área geográfica usando una VANET (para más información sobre la monitorización de áreas geográficas consúltese el Anexo E.3 "Monitorización de áreas geográficas"). El prototipo contaba con una interfaz muy limitada con la que sólo se podía arrancar la simulación, pararla y realizarla en modo *fast* (velocidad de ejecución mucho más rápida que la velocidad normal). Además de estas funciones, se podía realizar zoom sobre el mapa, mostrar las imágenes del mapa (*tiles*), dibujar el área de monitorización y modificar algún parámetro de la simulación.

Una de las principales limitaciones del prototipo era que no tenía ninguna herramienta para importar y trabajar con mapas externos. Por tanto, si se quería simular el comportamiento de un protocolo en un mapa con unas características diferentes al único existente, no se podía. Además de esta limitación, el prototipo tenía otras a nivel de interfaz o a nivel de funcionalidades. Se muestra la ventana de simulación del prototipo previo en la Figura 1:

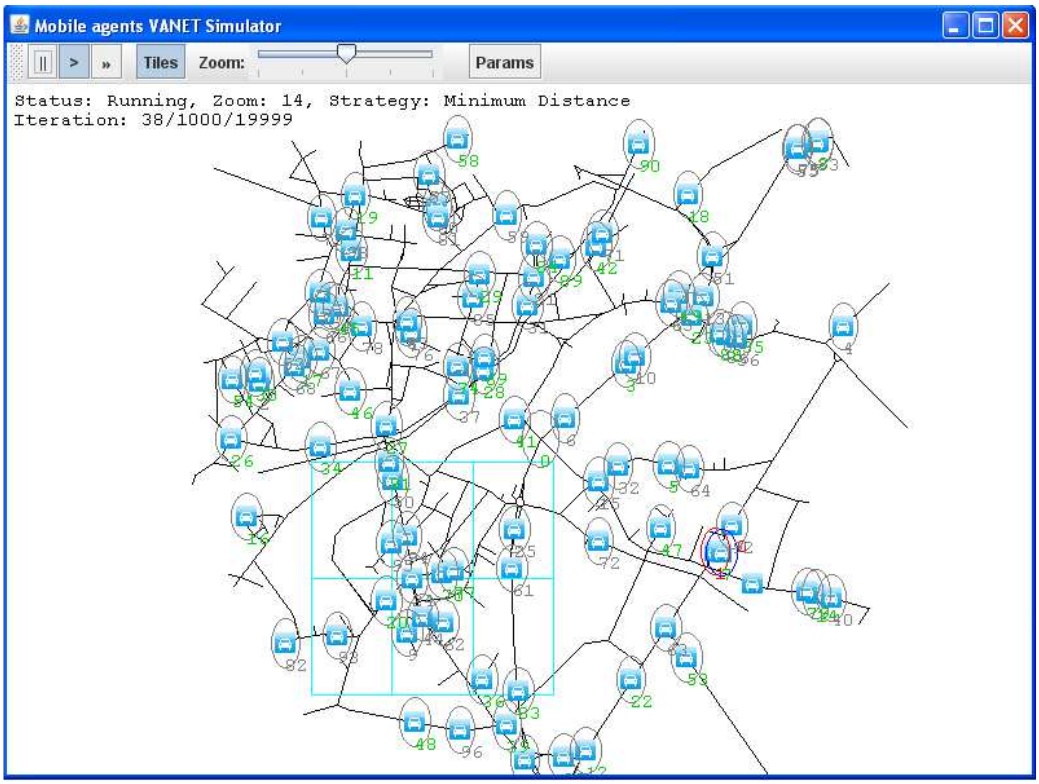


**Figura 1 – Imagen del prototipo previo desarrollado en el grupo SID**

Como se puede ver en la Figura 1 la interfaz sólo permitía:

- Pausar la simulación
- Ejecutar la simulación a velocidad normal
- Ejecutar la simulación a velocidad rápida
- Mostrar/ocultar las imágenes (*tiles*) del mapa
- Acercar/alejar el zoom
- Abrir un menú de configuración de la simulación.

Para ver otras limitaciones que poseía el prototipo previo del SID, consultar el Anexo E "Material relacionado".

## 1.2. Objetivos

El objetivo de este PFC es el de mejorar el prototipo de simulación de VANETs, anteriormente comentado, para que se pueda realizar un mayor número de simulaciones diferentes y más realistas, y de esta forma obtener una mayor cantidad de resultados para evaluar los distintos protocolos de intercambio de datos en el procesamiento de consultas. Y además, que esta herramienta desarrollada facilite la evaluación de diferentes técnicas de gestión de datos en redes de vehículos.
Dicho objetivo se ha dividido en distintos sub-objetivos:

- **Crear una interfaz de usuario amigable** que permita al usuario definir escenarios de simulación y controlar las situaciones de forma sencilla. Gracias a la interfaz, el usuario podrá realizar diversas tareas como por ejemplo: desplazarse por el mapa, hacer zoom, abrir un nuevo mapa, visualizar información sobre un vehículo, etc.
- **Desarrollar distintas estrategias de movilidad para los vehículos**. Dotar a los vehículos de diversas políticas de movilidad es importante, ya que los resultados obtenidos en la simulación de una red ad hoc cambian debido a la estrategia de movilidad de los vehículos [7].
- **Desarrollar un sistema de carga y almacenamiento de los parámetros de simulación**. Con este sistema se podrán repetir simulaciones con los mismos parámetros que simulaciones anteriores y de esta forma evitar el tener que apuntarlos para posteriormente introducirlos de nuevo en el simulador.
- **Dotar al simulador de una herramienta que incorpore nuevos mapas** en los que realizar las simulaciones. Los mapas podrán descargarse de diferentes fuentes (OpenStretMaps, GoogleMaps y MicrosoftMaps). Además, se podrán realizar filtros geográficos sobre el mapa descargado para cargar el área del mapa que el usuario desee.
- **Crear una herramienta que permita la importación de trazas GPS reales**. La herramienta de importación permitirá aplicar distintos filtros a la importación de trazas GPS, como por ejemplo filtros temporales, espaciales, o de velocidad.
- **Desarrollar una herramienta que permita simular la existencia de elementos fijos de soporte** a la VANET. Estos elementos realizarán la función de nodos de soporte para las comunicaciones en las carreteras, de

forma que es posible usar estos nodos de soporte además de los demás vehículos para conseguir el objetivo.

- **Integrar el sistema desarrollado** en el anterior simulador desarrollado por el grupo SID.

## 1.3. Tecnologías utilizadas

Para el desarrollo de este proyecto se han utilizado las siguientes tecnologías:

- El proyecto está desarrollado en Java [54, 55].
- La plataforma usada para el desarrollo del proyecto ha sido Eclipse [58, 59].
- Para varios ficheros de configuración e información se ha utilizado XML [57].
- Para obtener la situación geográfica de cualquier lugar para posteriormente proceder a la descarga de dicho mapa se ha utilizado una herramienta que suministra la siguiente página web [14].
- Se usa el JDK 6 update 11 [56] para realizar las pruebas necesarias del simulador en Linux, y el JDK 6 update 18 para realizar las pruebas en Windows a través de los scripts.
- Se han usado varios formatos de trazas GPS reales (plt, txt, log).

## 1.4. Fases del proyecto

Las fases seguidas a lo largo del proyecto han sido las siguientes:

- **Búsqueda de información:** investigar sobre las redes ad hoc, diferentes usos [1, 2, 3], monitorización de áreas [1], protocolos de diseminación de información [5] y simuladores existentes [11].
- **Familiarización con el proyecto existente:** debido a la existencia de un prototipo previo sobre el que había que añadir nuevas funcionalidades, fue necesaria una etapa de estudio y análisis del mismo.
- **Análisis de requisitos:** a partir de la información buscada y de diversas reuniones con el tutor se establecieron los puntos sobre los que trabajar en el proyecto.
- **Diseño de la aplicación:** se realizaron diversos diagramas para estructurar mejor el proceso de implementación.
- **Implementación del simulador:** se implementaron las funcionalidades del sistema.
- **Pruebas:** se realizaron pruebas unitarias, de integración y globales de las funcionalidades del simulador y se corrigieron los errores detectados.

## 1.5. Estructura de la memoria

En esta última sección se comenta la estructura de este documento. Principalmente, está formado por la memoria y los anexos.

La memoria está constituida por los siguientes capítulos:

- **Capítulo 1**: Introducción. En este capítulo se introduce la memoria.
- **Capítulo 2**: Planificación del proyecto. En este capítulo se explican las fases que se siguieron a lo largo del desarrollo de este PFC.
- **Capítulo 3**: Contexto tecnológico. En este capítulo se comentan los diferentes aspectos tecnológicos considerados en el desarrollo de la aplicación.
- **Capítulo 4**: Planteamiento y desarrollo del sistema. En este capítulo se comentan el análisis de requisitos y las diversas funcionalidades implementadas.
- **Capítulo 5**: Pruebas y problemas encontrados. En este capítulo se habla sobre las pruebas a las que fue sometida la aplicación y diversos problemas que surgieron en su implementación y cómo se solucionaron.
- **Capítulo 6**: Conclusiones. En este capítulo se comentan las conclusiones obtenidas una vez finalizado el proyecto y posibles futuras ampliaciones que se podrían desarrollar.

La parte dedicada a los anexos se divide en:

- **Anexo A**: Manual de usuario de la aplicación desarrollada.
- **Anexo B**: Formato de los ficheros. Se explican los formatos de los diversos ficheros utilizados en la aplicación.
- **Anexo C**: Análisis del sistema. Se detallan los requisitos y los casos de uso del mismo.
- **Anexo D**: Diseño e implementación del sistema. Se explica la estructura modular del sistema y las clases que forman la aplicación.
- **Anexo E**: Material relacionado. Se explican algunos temas relacionados con este proyecto.
- **Anexo F**: Estrategias de movilidad. Se explica de forma detallada cada estrategia de movilidad implementada en este PFC.

# 2. PLANIFICACIÓN DEL PROYECTO

La realización del proyecto está dividida en varias fases que se muestran en el siguiente diagrama de Gantt (Figura 2). Los días que aparecen indicados en cada fase muestran los días dedicados a la realización del proyecto.

El proyecto se realizó durante la segunda mitad del curso académico 2009/2010 y el principio del 2010/2011. La realización del proyecto fue compartida con varias asignaturas que se cursaron en la segunda mitad del curso 2009/2010, por lo que la finalización del mismo se demoró en el tiempo.

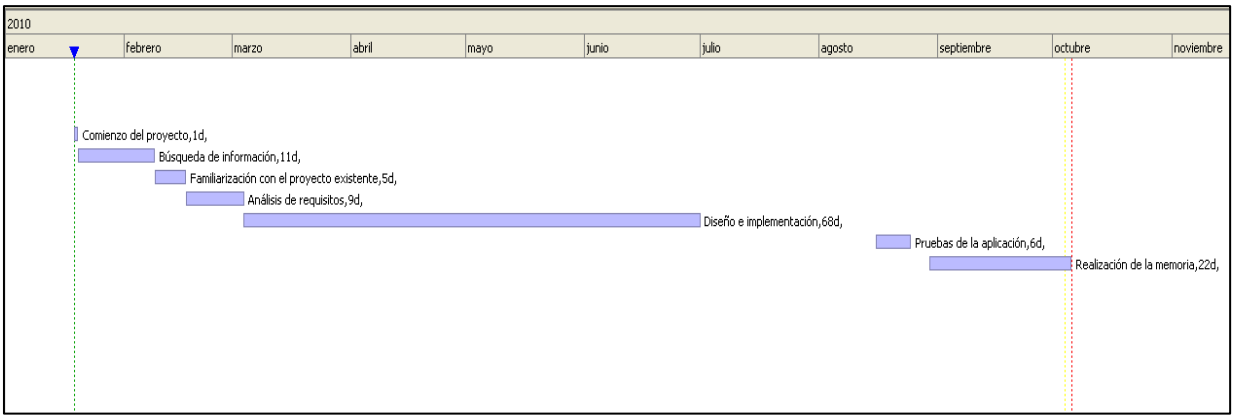


**Figura 2 – Fases del proyecto**

## 2.1. Listado de tareas

A continuación se explican las fases seguidas para el desarrollo del proyecto:

- **Comienzo del proyecto:** primera fase en la realización del proyecto, en la que tras una reunión con el tutor se trataron posibles funcionalidades a añadir para el prototipo ya existente. Además, el tutor proporcionó material necesario para las siguientes fases, es decir, código fuente del prototipo desarrollado por el SID y diversos documentos acerca de las VANETs.

- **Búsqueda de información (40 horas):** esta fase consistió tanto en la lectura del material proporcionado por el profesor como en la búsqueda de nueva información. Esta fase fue un periodo de aprendizaje sobre lo referente a las VANETs. Se obtuvo información acerca de otros proyectos basados en VANETs, como por ejemplo el proyecto VESPA [15], protocolos de diseminación de información, como por ejemplo el basado en EP (Encounter Probability) [5] y simuladores existentes de VANETs [11, 15, 16, 17].

- **Familiarización del prototipo existente (18 horas):** comprensión del código fuente del prototipo existente desarrollado por el SID ya que había que desarrollar nuevas funcionalidades para dicho prototipo. Además de esto, se obtuvo información acerca de apartados relacionados

con el prototipo: agentes móviles y protocolo de comunicación de los agentes móviles del prototipo [1].

- **Análisis de requisitos (26 horas):** a partir de las reuniones con el tutor y los conocimientos adquiridos en las fases previas se fueron concretando las funcionalidades que se querían integrar en el prototipo.

- **Diseño e implementación (256 horas):** se realizaron diversos diagramas para orientar la fase de implementación. Para afrontar la implementación de la interfaz de forma eficiente se realizaron bocetos de las ventanas y árboles de navegación. Tras el proceso de diseño se comenzó la fase de implementación. Algunos bocetos de las ventanas tuvieron que ser modificados debido a la introducción de nuevos elementos en las ventanas en la fase de implementación.

- **Pruebas del simulador (32 horas):** se realizaron pruebas unitarias de cada elemento implementado, pruebas de integración para comprobar que la nueva funcionalidad implementada funcionaba correctamente con el conjunto de la aplicación y pruebas de sistema para comprobar el buen funcionamiento de todo el simulador. Las pruebas se realizaron en un sistema operativo Windows XP Service pack 3 desde el propio entorno de programación Eclipse [18] y a través de scripts de "compilación" y ejecución y el JDK 1.6.0_18 [19], y en una distribución Debian del sistema operativo GNU/Linux (debian-live-503-i386-gnome-desktop) [20, 21].

- **Realización de la memoria (104 horas):** por último se realizó esta documentación.

- **Reuniones con el tutor (7 horas):** las reuniones se produjeron a lo largo de toda la duración del PFC, es decir, no componen una fase en concreto por sí solas.

**Total de horas invertidas: 483 horas**

## 2.2. Control de tiempos

Como se puede observar, se realizó un control de tiempos de todas las horas dedicadas a cada actividad del PFC. El PFC se realizó en un total de 483 horas. A continuación, se muestra un diagrama de tiempos (Figura 3) en el que se pueden ver tanto las horas dedicadas a cada fase como el porcentaje que constituye cada fase sobre el total de las horas invertidas.

- Comienzo del proyecto
- Búsqueda de información: 40 h
- Familiarización con el proyecto existente: 18 h
- Análisis de requisitos: 26 h

- Diseño e implementación: 256 h
- Pruebas de la aplicación: 32 h
- Realización de la memoria: 104 h
- Reuniones: 7 h

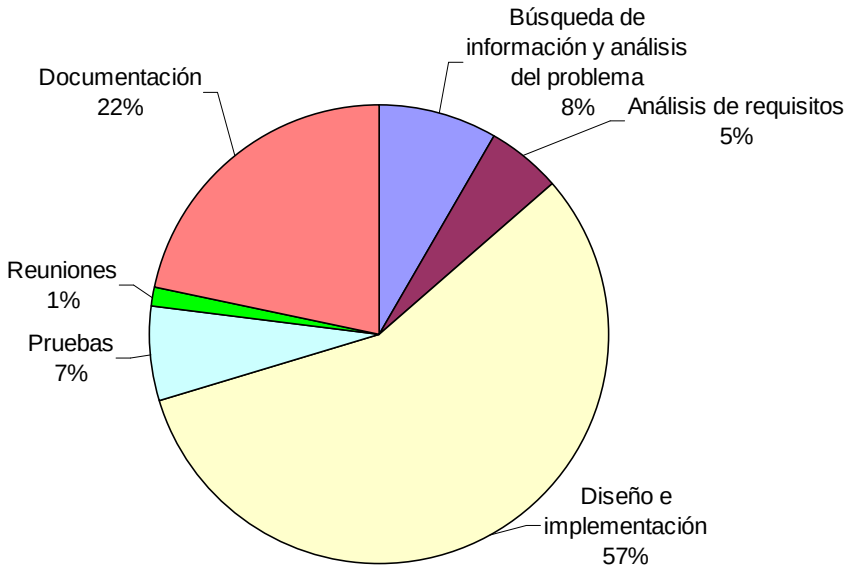


**Figura 3 – Diagrama de tiempos**

# 3. CONTEXTO TECNOLÓGICO

Las tecnologías empleadas para el desarrollo del proyecto fueron varias, ya que las funcionalidades que se han implementado abarcan diversos campos: descarga de mapas, trazas GPS reales, redes de vehículos.

Además de hablar sobre las tecnologías usadas se realizará una pequeña comparativa con otros simuladores de VANETs existentes.

## 3.1. Java y Eclipse

El lenguaje de programación utilizado para el desarrollo de este proyecto ha sido Java [54, 55]. La razón por la que se ha usado este lenguaje de programación es que el prototipo existente estaba implementado en Java ya que se buscaba una aplicación que fuera multiplataforma.

Para realizar la programación del simulador se ha usado el entorno de programación Eclipse [58, 29] por las ventajas que ofrece en la programación y depuración del código frente a un editor de texto convencional.

## 3.2. Redes de vehículos

Una red de vehículos ad hoc o VANET (Vehicular Ad hoc Network) [22] es una forma de red ad hoc móvil para permitir comunicaciones entre vehículos cercanos. Uno de los principales objetivos de una VANET es proveer de información a los conductores de los vehículos para que la conducción sea más segura y cómoda. Además de este uso de las VANETs se puede utilizar con otros fines, como por ejemplo la monitorización de áreas geográficas.

En una VANET cada vehículo está equipado con un dispositivo, lo que le convierte en un nodo de la red ad hoc, y con este dispositivo puede recibir y transmitir información a otros vehículos de la red si se encuentran lo suficientemente cerca (ver Figura 4).

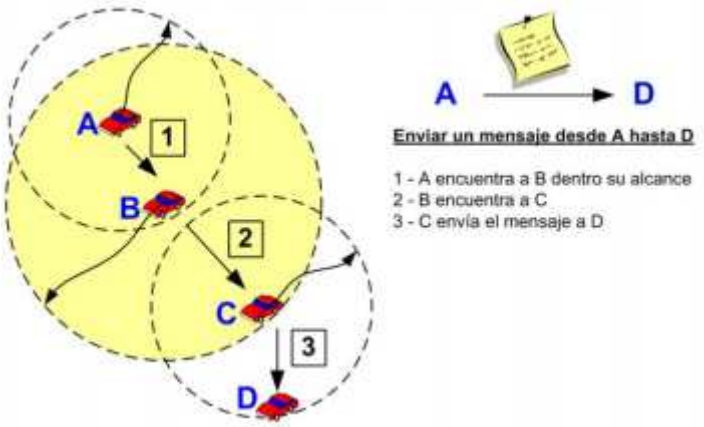


**Figura 4 – Ejemplo de comunicación multi-salto en una red ad-hoc** "Extraído de [23]"

## 3.3. OpenStreetMap

Ante la necesidad de incorporar una herramienta para la descarga de mapas, se buscó información y se encontró un PFC anterior que solucionaba dicho problema [24] con los mapas de OpenStreetMap [26]. Se dedica una sección a este tipo de mapas y no a los otros tipos (MicrosoftMaps y GoogleMaps) porque OpenStreetMaps provee tanto el mapa (imágenes) como la información geográfica de los mismos.

OpenStreetMap (también conocido como OSM) es un proyecto colaborativo para crear mapas libres y editables. Los mapas se crean utilizando información geográfica capturada con dispositivos GPS móviles, ortofotografías y otras fuentes libres. Esta cartografía, tanto las imágenes creadas como los datos vectoriales almacenados en su base de datos, se distribuye bajo licencia Creative Commons Attribution-Sahre Alike 2.0.

En enero de 2010 el proyecto superaba los 200.000 usuarios registrados, de los cuales cerca de 11.000 realizan alguna edición en la base de datos cada mes. El número de usuarios suele doblarse cada cinco meses. Los usuarios registrados pueden subir sus trazas desde el GPS, crear y corregir datos vectoriales mediante herramientas de edición creadas por la comunidad OpenStreetMap.

Cada día se añaden 25.000 km nuevos de carreteras y caminos con un total de casi 34.000.000 km de vías, eso sin añadir otros tipos de datos (puntos de interés, edificaciones, etc.). El tamaño total de la base de datos (llamada planet.osm) se sitúa por encima de los 160 GB (6,1 GB con compresión bzip2), incrementándose diariamente en unos 10 MB de datos comprimidos [25]. En la Figura 5 se muestra la imagen de la ciudad de Zaragoza de OpenStreetMap.

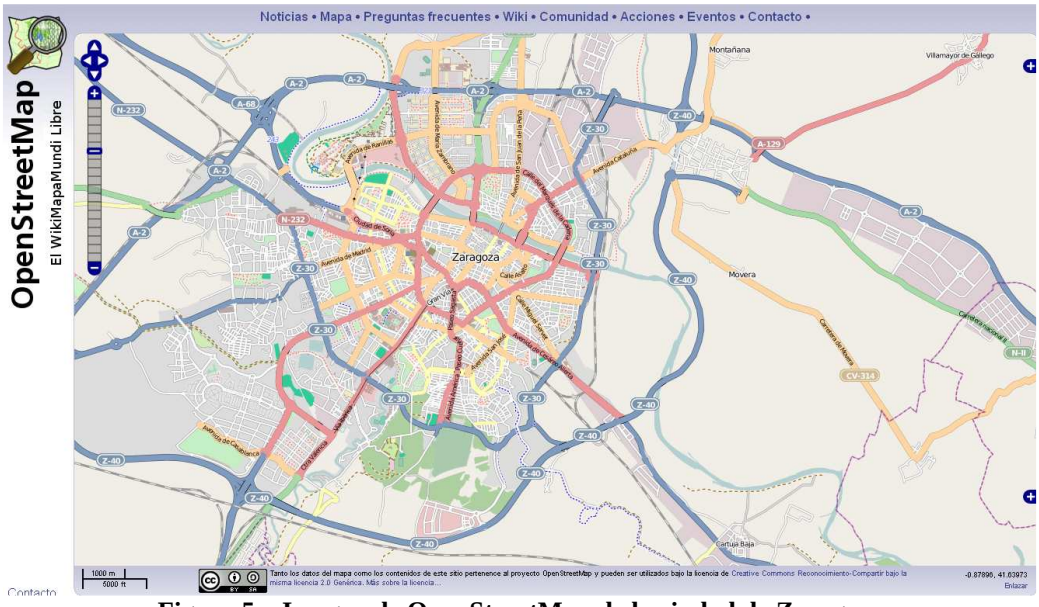


**Figura 5 – Imagen de OpenStreetMap de la ciudad de Zaragoza**

OpenStreetMap utiliza una estructura de datos topológica. Los datos se almacenan en el *datum* WGS84 lat/lon (EPSG:4326) de proyección de Mercator.

Los elementos básicos de la cartografía OSM son los nodos (*nodes*), las vías (*ways*), las relaciones (*realations*) y las etiquetas (*tags*).

Se puede exportar el mapa del área geográfica que se indique en un fichero en formato XML. Los elementos más importantes del fichero XML son:

- Los **nodos** (***nodes***). Son puntos que recogen una posición geográfica dada. Los atributos más importantes del nodo son el identificador del nodo y sus coordenadas geográficas (latitud y longitud).

```
<node id="30438349" lat="41.6579256" lon="-0.9085487" user="jynus" uid="7406"
visible="true" version="5" changeset="1296522" timestamp="2009-05-
23T17:24:47Z"/>
```

- Las **vías (*ways***). Son una lista ordenada de nodos que representa una polilínea o polígono. Sus atributos más importantes son el identificador de la vía y los identificadores de los nodos que componen dicha vía.

```
<way id="23133938" user="Jesus Gomez" uid="6389" visible="true" version="4"
changeset="622994" timestamp="2008-09-13T20:30:29Z">
<nd ref="30438473"/>
<nd ref="296483955"/>
<nd ref="296483948"/>
<tag k="created_by" v="JOSM"/>
<tag k="highway" v="secondary"/>
<tag k="name" v="Avenida de Madrid"/>
</way>
```

Para más información sobre cómo abrir y descargar un mapa, consultar el Anexo A "Manual de usuario". Para más información acerca de la descarga de los mapas consúltese el Anexo C.2.1 "Descarga de Mapas" y "Anexo E" de [24].

## 3.4. Trazas GPS

Una traza GPS es una sucesión de puntos registrados por un dispositivo el cual indica diversa información sobre cada punto registrado: coordenadas geográficas, fecha, hora, etc.

En la actualidad existen varios formatos de trazas GPS: gpx, log, txt, plt, kml, etc. Los formatos que se han tratado en este PFC han sido: log, plt y txt. Los ficheros de trazas GPS utilizados se han extraído del proyecto GeoLife [27] de Microsoft, ya que en dicho proyecto existe la capacidad de descargarse trazas GPS reales [28] obtenidas por un grupo de 165 personas.

El proyecto GeoLife de Microsoft es una red social basada en la localización, y es un servicio de Microsoft Virtual Earth [29]. Virtual Earth es una extensión para Internet Explorer que permite visualizar las principales ciudades de los Estados Unidos en 3D. Una de las funcionalidades que tiene GeoLife es *GeoLife GPS Trajectories* con la que se pueden compartir trazas GPS. *GeoLife GPS Trajectories* tiene una base de datos de trazas GPS reales recolectadas por Microsoft Research Asia. Durante 2 años (desde abril del 2007

a agosto del 2009) fueron registrados de forma voluntaria los movimientos en el exterior de 165 personas. Las trayectorias GPS fueron registradas por diferentes modelos de dispositivos GPS o teléfonos móviles con GPS con diversas frecuencias de muestreo. El 95 % de las trayectorias tiene sus nodos registrados en intervalos de 2 a 5 segundos o de 5 a 10 metros Estos movimientos pueden ser movimientos rutinarios de las personas como ir a casa o al trabajo y también incluyen movimientos fuera de la rutina como ir de compras, de excursión, senderismo o montar en bicicleta.

Las trayectorias GPS corresponden a lugares de todo el mundo: 30 ciudades de China, ciudades de Estados Unidos y ciudades de Europa. Pero la mayoría de ellas fueron registradas en Beijing (China).

## 3.5. Otros simuladores existentes

En la actualidad existen diversos simuladores de red, de tráfico de vehículos e híbridos (de red y de tráfico de vehículos).

Por una lado, los simuladores especializados en las redes dan la posibilidad de configurar bastantes parámetros de la comunicación entre dispositivos, como por ejemplo el tipo de tecnología para la comunicación, porcentaje de pérdida de datos, latencias, situar obstáculos para que dificulten la comunicación inalámbrica entre dos dispositivos, etc.

Por otro lado, los simuladores especializados en el tráfico de vehículos dan la posibilidad de generar trazas de movimiento de los vehículos, cuyo comportamiento puede ser configurable, en unos escenarios determinados.

A continuación, se van a comentar varios de los simuladores de redes, de tráfico e híbridos que existen y finalmente se hará una comparativa entre ellos y el simulador realizado en el PFC.

### 3.5.1. Simuladores de red

A continuación se van a comentar algunos de los simuladores de red existentes:

**Ns-2**

En 1989, Ns-2 [60] comenzó como una variante del simulador de red REAL [61]. Ns-2 soporta simulaciones de TCP, routing, multicast, redes cableadas y redes inalámbricas (locales o con satélite). Este simulador está programado en C++ y OTCL (300.000 líneas de código) [32]. Además, tiene una gran cantidad de librerías para simular redes inalámbricas. En simulaciones inalámbricas no permite la existencia de obstáculos que dificulten la comunicación y tampoco permite la atenuación de la señal. Los dispositivos de comunicación no son capaces de detectar la presencia de otros dispositivos a no ser que se indique expresamente. En [33], se realizó una simulación que necesitó 5,6 GB de memoria y 118,93 minutos para simular 500 nodos. En la Figura 6 se muestra una imagen del simulador Ns-2.

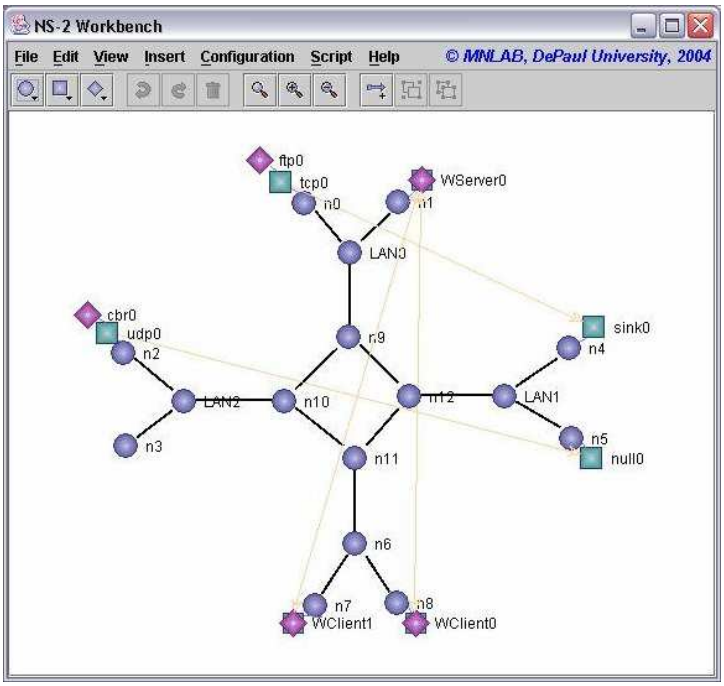


**Figura 6 – Imagen del simulador Ns-2**

## QualNet

QualNet (Quality Networking) es un software desarrollado en el año 2000 para evaluar redes y está programado totalmente en C++. Puede funcionar en los sistemas operativos UNIX, Windows, MAC OS y Linux. Posee una gran cantidad de librerías que le permiten realizar simulaciones de redes WiFi, WiMAX y redes de sensores. El simulador soporta hasta 20.000 nodos. Está implementado en capas similares al modelo OSI; tiene cuatro capas: aplicación, transporte, enlace y física. A diferencia de Ns-2, QualNet soporta alguna estrategia de movilidad. Tiene 2 versiones, una de pago y otra de evaluación. En la Figura 7 se muestra una imagen del simulador QualNet.

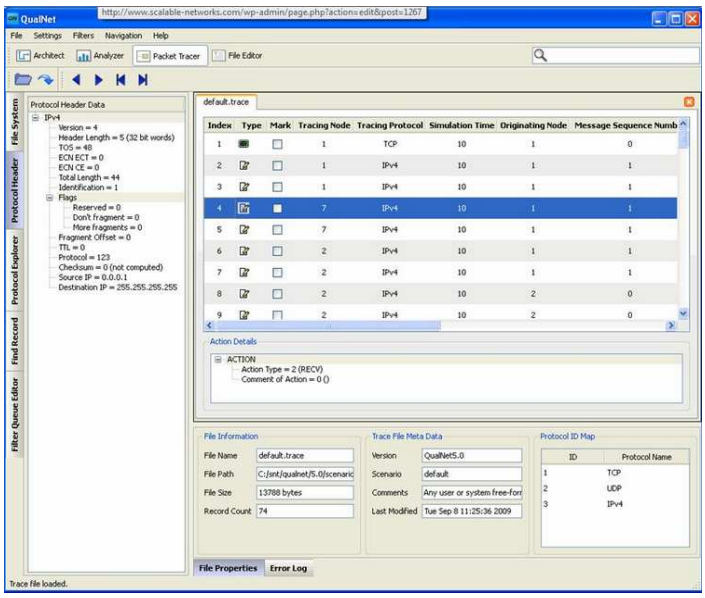


**Figura 7 – Imagen del simulador QualNet**

### 3.5.2. Simuladores de tráfico

A continuación se van a comentar algunos de los simuladores de tráfico existentes:

**VanetMobiSim**

VanetMobiSim [65] es una extensión de CANU Mobility Simulation Enviroment (CanuMobiSim [66]). Está implementado en Java. Permite a los coches interactuar entre ellos y con los elementos de la carretera (semáforos, señales de tráfico, etc.). Extrae la topología de las carreteras de TIGER [67] (Topologically Integrated Geographic Encoding and Referencing) y GDF [68] (Geographic Data File). OpenStreetMap contiene los datos de TIGER que sólo tiene datos de los Estados Unidos, Puerto Rico, islas Virginia, Samoa, Guam, islas Marianas y las islas Midway. Permite al usuario usar rutas eligiendo un punto inicial y otro final, y el simulador creará la ruta a través del algoritmo del camino más corto de Dijkstra. Puede exportar sus datos de movimientos de vehículos a un simulador de red como Ns-2, pero no puede realimentarse de los datos generados por Ns-2 a partir de los datos que le había enviado previamente. En la Figura 8 se muestra una imagen del simulador VanetMobiSim.

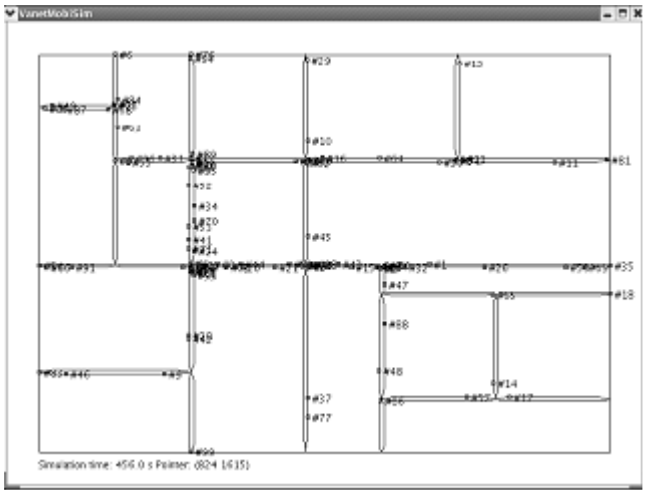

**Figura 8 – Imagen del simulador VanetMobiSim** "Extraído de [37]"

**TraNs Lite**

TraNs (Traffic and NetWork Simulator enviroment) está implementado en Java. Tiene una herramienta de visualización para integrar SUMO [64] y Ns-2. Después surgió TraNs Lite [70] con el objetivo de generar trazas de movilidad para un simulador de red como Ns-2. TraNs Lite es capaz de simular hasta 3.000 nodos. Extrae la topología de las carreteras de TIGER, ShapeFile [69] y OpenStreetMap. En la Figura 9 se muestra una imagen del simulador TraNs Lite.

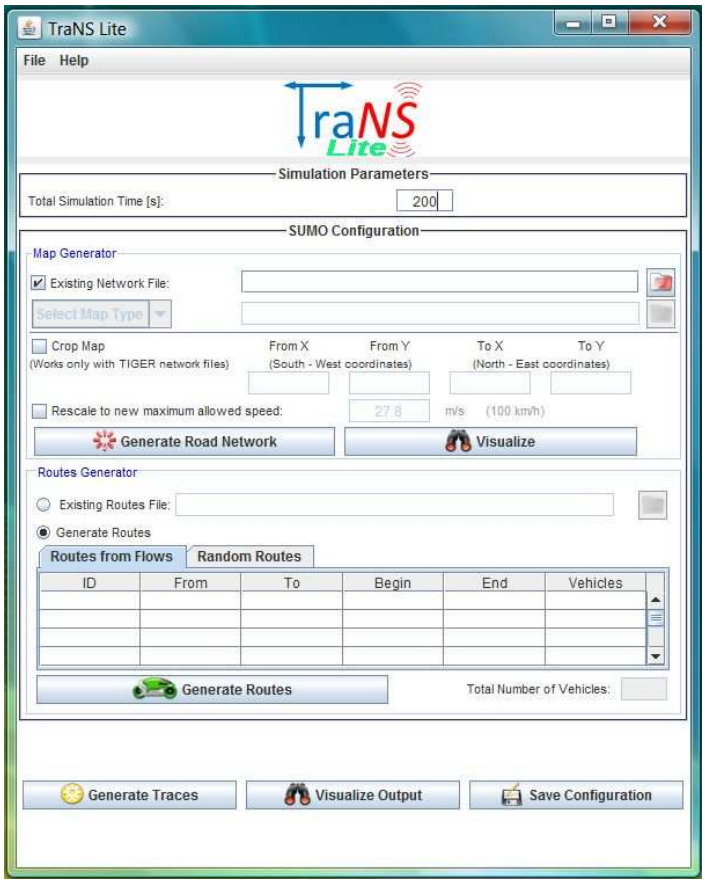


**Figura 9 – Imagen del simulador TraNSLite**

Además de estos simuladores de tráfico existen otros más completos con simulaciones en 3D como por ejemplo: AIMSUM [34], VISSIM [36], CORSIM [35] cuyas licencias de uso valen unos 9.000 $. En la Figura 10 se muestra una imagen del simulador AIMSUM.

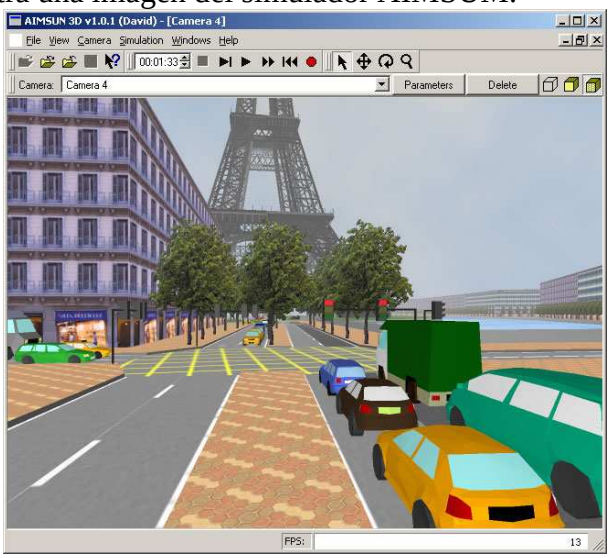


**Figura 10 – Imagen del simulador AIMSUM**

### 3.5.3. Simuladores híbridos

Un simulador híbrido es aquel que es simulador de red y simulador de tráfico de vehículos.

**NCTUns**

NCTUns (National Chiao Tung University Network Simulator) está basado en el simulador de Harvard propuesto por S.Y. Wang en 2002. NCTUns está programado en C++. Los vehículos de este simulador tienen la capacidad de interactuar con el medio en el que se encuentran realizando cambios de carril o respetando los semáforos. Es capaz de simular las tecnologías 80.211a, 802.11b, 802.11g y 802.11p. Puede introducir obstáculos que dificulten la comunicación inalámbrica y simular atenuación de la señal. NCTUns puede simular como máximo 4096 nodos. En la Figura 11 se puede ver una imagen de este simulador.

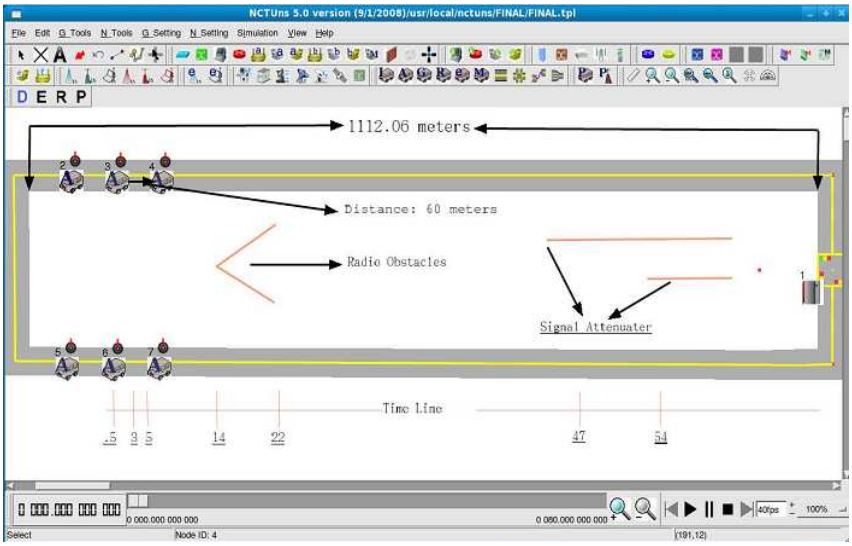


**Figura 11 – Imagen del simulador NCTuns** "Extraído de [11]"

A continuación se muestra una tabla comparativa (Tabla 1) entre todos los simuladores y el realizado en el PFC.

| | **REDES** | | **TRÁFICO DE VEHÍCULOS** | | **HÍBRIDOS** | |
|---|---|---|---|---|---|---|
| **Simulador** | **Ns-2** | **QualNet** | **VanetMobiSim** | **TraNs Lite** | **NCTUns** | **Simulador del PFC** |
| **S.O.** | Multiplataforma | Multiplataforma | Multiplataforma | Multiplata-forma | Linux | Multiplata-forma |
| **Precio** | Gratis | Pago | Gratis | Gratis | Depende * | Gratis |
| **Simulador de tráfico** | No | No | Sí | Sí | Sí | Sí |
| **Simulador de red** | Sí | Sí | No | No | Sí | Depende ** |
| **GUI** | Sí | Sí | Sí | Sí | Sí | Sí |
| **Permite simular monitorización de áreas** | No | No | No | No | No | Sí |
| **Disponibilidad del código fuente** | Sí | No | Sí | Sí | Sí | Sí |
| **Descarga mapas** | No | No | GDF | TIGER database | Bitmap Image | OpenStreet-Map, GoogleMaps y MicrosoftMaps |
| **Estrategia movilidad** | No | No | Random walk, Dijkstra, modelo inteligente de conductor, modelo inteligente capaz de cambiar de carril, modelo inteligente en intersecciones | Random walk, Dijkstra, Random Start-End, modelos de seguimient o de vehículos | Random walk, modelo inteligente capaz de seguir otro vehículo, Modelo inteligente capaz de cambiar de carril, Modelo inteligente en intersecciones | Gauss-Markov, Random walk, Manhattan adaptada a cualquier mapa, Dijkstra, rutas dinámicas con heurística, straight adjacent node, seguimiento de trazas GPS reales |
| **Configuración de la velocidad de los vehículos de la simulación** | No | No | Dependiente de la carretera, constante | Dependiente de la carretera, constante | Dependiente de la carretera, constante | Dependiente de las trazas GPS, constante, modificable en cualquier momento de la simulación |
| **Nodos fijos soporte comunica-ciones** | Sí | Sí | No | No | Sí | Sí |
| **Filtros geográficos a mapas** | No | No | No | No | No | Sí |
| **Filtros a trazas de movimientos vehículos** | No | No | No se ha encontrado la información | No se ha encontrado la información | No se ha encontrado la información | Sí (filtros, temporales, geográficos y de velocidad) |
| **Ejecución de comandos** | Sí | Sí | No se ha encontrado la información | Sí | No se ha encontrado la información | Sí |

**Tabla 1 – Comparativa de simuladores**

* : El precio del simulador depende del uso que se le dé. Si se usa para la enseñanza o para realizar evaluaciones, el coste del producto es cero. Mientras que si se usa para fines comerciales entonces es un producto de pago.

**: No es un simulador de red, pero a diferencia de los simuladores de tráfico de vehículos, éste permite realizar simulaciones en las que los vehículos se comunican entre sí.

**Conclusiones de la comparativa**

Como se puede ver en la tabla comparativa Tabla 1, los simuladores de red carecen de herramientas que proporcionen estrategias de movilidad a sus elementos, no permiten modificar la velocidad de sus elementos ni permiten descargar mapas. Por otro lado, los simuladores de tráfico carecen de herramientas que permitan añadir elementos de soporte para las comunicaciones entre vehículos. Todas estas diferencias se deben a que los simuladores de red están diseñados para realizar simulaciones de redes con diversos protocolos y configuraciones, y los simuladores de tráfico están diseñados para escenarios en los que lo importante es el comportamiento de los vehículos.

Por otro lado, se puede encontrar el simulador híbrido NCTUns que permite realizar simulaciones de tráfico en las que los vehículos se comunican entre sí (simulador de red). Éste sería el simulador más parecido al desarrollado en este PFC. Alguna de las ventajas que tiene el simulador desarrollado en este PFC frente a NCTUns es que es multiplataforma mientras que NCTUns sólo funciona sobre Linux; que es gratuito y que permite realizar simulaciones de monitorización de áreas geográficas, mientras que NCTUns no lo permite. La principal ventaja que tiene NCTUns es que sus vehículos pueden seguir estrategias de movilidad que les proporcionan a los vehículos un comportamiento algo más real ya que pueden realizar cambios de carril. De todas formas, el simulador desarrollado en este PFC, debido a su diseño extensible, permite la incorporación de nuevas estrategias de movilidad de una forma rápida y sencilla.

# 4. PLANTEAMIENTO Y DESARROLLO DEL SISTEMA

Una vez realizada la búsqueda de información y la familiarización con el prototipo, se procedió al análisis de requisitos.

A continuación, se van a comentar los aspectos más importantes que se han planteado en el análisis de requisitos y el desarrollo del sistema.

## 4.1. Análisis de requisitos

Tras diversas reuniones con el tutor se fueron tratando los diversos puntos que se consideraban interesantes para incluir como funcionalidades del prototipo.

El primer requisito que se consideró importante para añadir fue el referente a las estrategias de movilidad para los vehículos. El prototipo contaba con dos políticas de movilidad: “straight adjacent node” y “random walk”.
Estas estrategias de movilidad no eran suficientes ya que el comportamiento de los vehículos no parecía muy real y lo que se buscaba era un comportamiento lo más fidedigno posible a la realidad para que los resultados de las simulaciones fueran también más reales. Por lo tanto, se buscó información acerca de políticas de movilidad existentes y como resultado de esta búsqueda se consideraron las siguientes políticas de movilidad.

La primera política de movilidad que se estudió e implementó fue la basada en rutas que utiliza el algoritmo de Dijkstra para calcular el camino desde un punto origen hasta un punto destino. De la implementación de esta estrategia de movilidad surgió otra similar que usa una heurística para alcanzar un punto destino obtenido al inicio de la “ruta”. La principal diferencia con la primera es que ésta no usa el algoritmo de Dijkstra para obtener inicialmente una ruta que sigue hasta alcanzar su punto destino final, sino que cada vez que se mueve a un punto nuevo, calcula con la heurística la siguiente posición para alcanzar el punto destino final. Otras estrategias de movilidad que se estudiaron e implementaron fueron Gauss-Markov y Manhattan.

El siguiente requisito que se consideró importante fue incluir una herramienta que permitiera el almacenamiento y carga de parámetros de configuración para la simulación. Debido a que cada vez que se quería realizar una simulación con parámetros diferentes a la anterior había que realizar los cambios sobre el propio código fuente, se consideró que era necesario solucionar este problema.

El siguiente aspecto a considerar fue la interfaz. Inicialmente se contaba con una interfaz muy limitada que no permitía muchas opciones, como se puede ver en la Figura 1. Por lo tanto, se creyó que era necesario dotar al prototipo de una interfaz más amplia que le permitiera controlar las simulaciones de la forma más simple posible. Para ello se realizaron bocetos para las distintas ventanas y pequeños árboles de navegación que facilitarían la posterior implementación.

Otro punto importante que se trató para añadir fue el dotar al prototipo de una herramienta capaz de añadir mapas. Para ello había que descargar los mapas (fichero con información de las carreteras e imágenes del mapa) y dar la posibilidad al usuario de que pudiera elegir el mapa que quisiese para realizar

simulaciones sobre él. Más tarde se vio la necesidad de añadir la opción de realizar filtros geográficos a los mapas para centrar las simulaciones sobre un área determinada del mapa.

Debido a que el simulador era cerrado al exterior, es decir, no podía admitir datos obtenidos desde otras fuentes, se consideró añadir una herramienta que solucionara este problema. Para ello se pensó permitir importar trazas GPS reales de diversos formatos. De esta forma los vehículos realizarían movimientos 100% reales, por lo que se podría considerar como una política de movilidad muy realista, y además el simulador sería capaz de incluir datos obtenidos desde el exterior y usarlos para sus simulaciones. Más tarde se consideró importante poder filtrar estas trazas GPS para centrarse en las que interesaran en cada momento. Los filtros que se consideraron importantes fueron los temporales, geográficos y de velocidad.

El último aspecto que se trató fue el dar la posibilidad de incluir nodos de soporte para las comunicaciones entre los vehículos. Con estos nodos de soporte, los vehículos además de poder realizar comunicaciones vehículo-vehículo podrán realizar comunicaciones vehículo-nodo de soporte.

A continuación, se muestra un pequeño resumen de los requisitos que se han comentado (para ver con detalle el análisis de requisitos y los casos de uso, consúltese el Anexo C “Análisis del sistema”):

- Dotar al prototipo de nuevas estrategias de movilidad.
- Añadir una herramienta que permita la carga y guardado de los parámetros de configuración y permita la configuración de las simulaciones de una forma rápida.
- Implementar una interfaz de usuario amigable que permita al usuario controlar el simulador de forma rápida y fácil.
- Añadir una herramienta para descargar mapas y posteriormente usarlos en las simulaciones. Añadir la opción de realizar filtros geográficos a los mapas
- Implementar una herramienta para importar trazas GPS reales para que posteriormente los vehículos reproduzcan esos movimientos. Añadir la posibilidad de realizar filtros temporales, espaciales y de velocidad.
- Permitir añadir nodos de soporte en la comunicaciones de los vehículos

En las primeras reuniones se trataron bastantes de los requisitos comentados de una forma más general y según se iba avanzando en el proyecto las reuniones se centraban más sobre determinados requisitos o se iban añadiendo otros requisitos que surgían debido a la necesidad encontrada. Por ejemplo, el realizar filtros geográficos sobre la carga de un mapa surgió una vez que se importaron las trazas GPS reales y se descargó un mapa para visualizar las trazas GPS sobre el mapa. Esta necesidad surgió porque el mapa que abarcaba todas las trazas GPS que se importaban era excesivamente grande y era necesario centrarse en una determinada zona.

## 4.2. Diseño e implementación del sistema

Una vez establecidos los requisitos se comenzó con el diseño y la implementación de cada uno ellos.

El simulador está diseñado para favorecer la extensibilidad y modularidad. Diversos paquetes conforman el simulador, y cada uno de ellos contiene unas clases según su funcionalidad. Por ejemplo, existe un paquete que gestiona todo lo referente a los gráficos, otro lo referente a las estrategias de movilidad, etc. De esta forma es más fácil el mantenimiento del código. Por otro lado, para favorecer la extensibilidad se han implementado algunas clases abstractas e interfaces que permiten la incorporación de nuevas funcionalidades de una forma más rápida y sencilla. Por ejemplo, si se quiere añadir una nueva estrategia de movilidad para los vehículos, se dispone de una clase abstracta llamada *Mobility* que cuenta con los métodos y atributos necesarios para todas las estrategias de movilidad, y de esta forma es más sencillo añadir nuevas políticas de movilidad. Existe también una clase abstracta para las clases encargadas de leer los formatos de ficheros de trazas GPS (*TrackReader*), por lo que se si se quiere incorporar un nuevo formato de fichero de trazas GPS, tan sólo habría que implementar la clase encargada de la lectura del fichero de nuevo formato. Además, existe una clase interfaz y una clase abstracta para facilitar la incorporación de nuevos experimentos a realizar en el simulador (*IAgent, Agent*). Para más información acerca del diseño modular y extensible consúltese el Anexo D “Diseño e implementación del sistema”.

A continuación, se describe la estructura general de la aplicación (ver Figura 12).

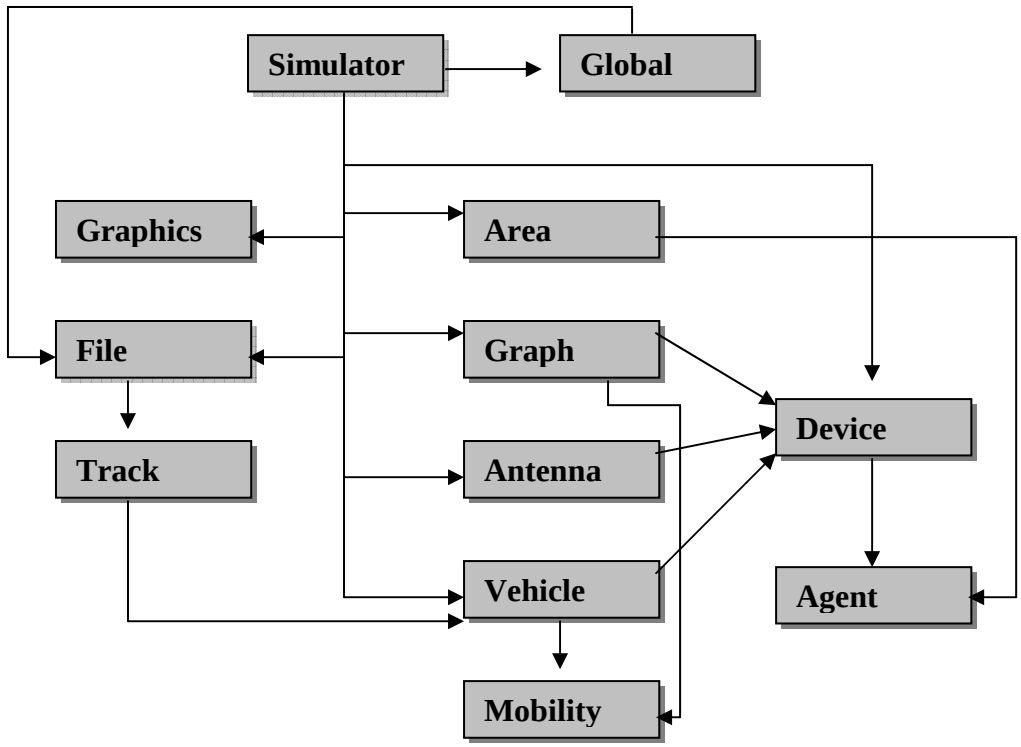


**Figura 12 – Esquema general de la aplicación**

### 4.2.1. Estrategias de movilidad

Las estrategias de movilidad implementadas en este PFC (basadas en rutas, basada en Manhattan y basada en Gauss-Markov) y las ya existentes se organizaron de forma que se facilitara la extensibilidad. Todas las clases referentes a las políticas de movilidad se encuentran en el paquete *Mobility*. Para más información sobre las estrategias de movilidad implementadas, consúltese el Anexo F "Estrategias de movilidad".

En la Figura 13 se muestra un esquema sobre la organización de las clases del paquete *Mobility*.

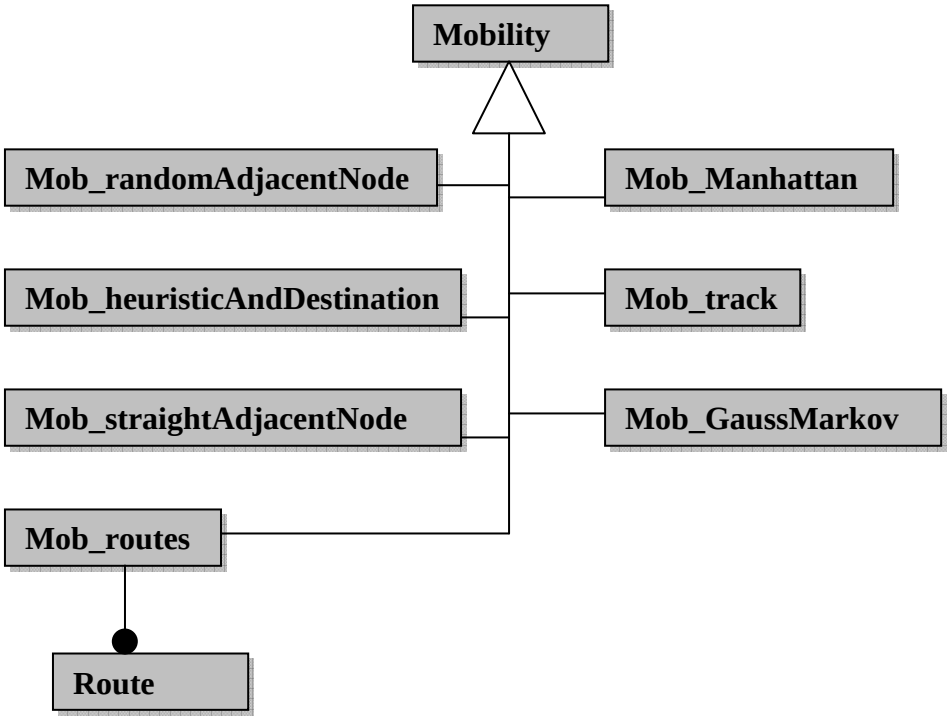


**Figura 13 – Diagrama de clases esquemático del paquete *Mobility***

### 4.2.2. Sistema de carga y almacenamiento de parámetros

Debido a que en el prototipo anterior, cada vez que se quería modificar un parámetro de simulación había que modificar el código fuente del mismo, se decidió desarrollar una herramienta que facilitara este escollo.

La herramienta permite guardar de forma automática los parámetros de configuración de la simulación y su posterior uso en las siguientes simulaciones. Esto se lleva a cabo a través de un fichero XML que contiene todos los parámetros de simulación. El fichero de configuración se llama "config.xml" y tiene el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config>
   <numberOfVehicles>15</numberOfVehicles>
   <mobilityStrategy>Route</mobilityStrategy>
    …
   <MobileCommunicationRange>100.0</MobileCommunicationRange>
   <antennasEnabled>no</antennasEnabled>
</config>
```

Con esta herramienta, si se quiere configurar cualquier parámetro, como por ejemplo, el número máximo de iteraciones por simulación, sólo se tiene que configurar ese elemento (maxSimIter) en el fichero de configuración y al iniciar el simulador los cambios se harán efectivos.

Para más información sobre el formato del fichero de configuración consúltese el Anexo B.1 "Fichero de configuración".

### 4.2.3. Herramienta de descarga de mapas

Cuando se habla de la descarga de un mapa, realmente se descargan dos elementos, un fichero XML y las imágenes del mapa o "*tiles*". El fichero XML se descarga siempre de OpenStreetMap. Mientras que las imágenes del mapa se descargan de diversos servidores, como pueden ser: OpenStreetMap, GoogleMaps, MicrosoftMaps y YahooMaps.

El fichero XML contiene información acerca de las vías (calles y carreteras) de la zona seleccionada. Toda esta información se usa para componer el grafo de nodos con el que trabajan las políticas de movilidad. Este grafo se usa para representar las carreteras en el simulador.

Sobre las imágenes descargadas hay que decir que todas ellas, como si fueran baldosas, forman el mapa. Para entender este concepto mejor, a continuación, se va a explicar un poco más en profundidad.

Un mapa online, GoogleMaps está compuesto por imágenes de 256 x 256 píxeles llamadas *tiles*. Estas imágenes son las que se

descargarán junto con el fichero XML y juntas conformarán el mapa. Por lo tanto, no hace falta descargar todas las *tiles* del mapamundi sino aquellas que tengan que ver con el área que hemos seleccionado.

Los mapas se clasifican según su zoom. GoogleMaps tiene 18 niveles de zoom y por lo tanto habrá 18 mapas con sus *tiles* correspondientes que los conforman. Cada imagen de un mapa se compone de un número determinado de *tiles* de 256 x 256 píxeles, las cuales forman una matriz de *tiles*. Cada *tile* de esta matriz se identifica con un número de fila y columna. Y además hace falta indicar el nivel de zoom para identificar el mapa concreto. Por ejemplo, una *tile* designada con (4,5,6) indicaría la *tile* que ocupa la fila 4 y columna 5 de la matriz y pertenece al mapa de zoom 6. El número de filas y columnas que conforman la matriz de cada mapa viene determinado por el nivel de zoom y es concretamente $2^{\text{nivel de zoom}}$. Por lo tanto, el nivel de zoom 4 tendrá $2^4 = 16$ filas y 16 columnas. Véase la Figura 14 para entender mejor este concepto.

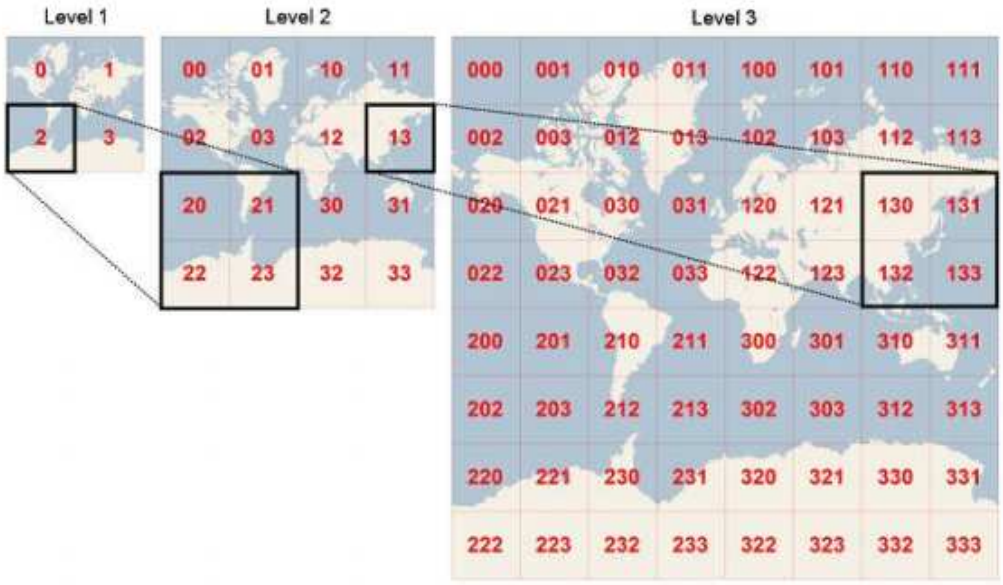


**Figura 14 – Mapa dividido en trozos con diferentes niveles de zoom** "Extraído de [24]"

Una vez que se conoce cómo se designa cada *tile* ya es posible proceder a la descarga de un mapa de una zona determinada.

Para descargar una zona determinada hace falta indicar los límites de dicha zona. Los límites se indicarán con dos puntos: uno situado en la posición superior izquierda (punto verde de la Figura 15) y otro en la posición inferior derecha (punto rojo de la Figura 15).


**Figura 15 – Definir mapa a descargar con 2 puntos**

A partir de las coordenadas geográficas de cada punto (latitud y longitud) y el nivel de zoom que se elige se puede extraer la *tile* a la que corresponde. Para conocer las coordenadas geográficas (latitud y longitud) de un determinado punto del mundo se ha usado la herramienta web [14]. Una vez que se tienen las *tiles* que limitan el área geográfica del mapa que se quiere descargar, la herramienta de descarga de mapas calculará el resto de *tiles* que es necesario descargar para componer la matriz que conforma el mapa. Todas estas *tiles* se almacenarán en el disco duro según se explica en el Anexo D.3. "Estructura del almacenamiento de las *tiles*".

Habíamos dicho que un mapa estaba formado por dos elementos: un fichero en formato XML y las imágenes. Así que sólo falta descargar el fichero XML que contiene la información de carreteras y calles. Para ello hay que hacer una petición al servidor de OpenStreetMap en el que se indican las coordenadas geográficas de los puntos anteriores que limitan el área que se desea descargar. A continuación, se obtendrá el fichero XML con toda la información de esa zona seleccionada y se podrá formar el grafo. Para más información acerca del formato de los ficheros OSM consultar el Anexo B.2. "Fichero de información OSM (OpenStreetMap)".

## Filtros geográficos

Cuando se descarga el fichero XML de una zona geográfica determinada no se conoce previamente la información acerca de las calles y carreteras que contendrá dicho fichero. Por lo tanto, puede darse el caso de que se descargue un fichero que contenga mucha información de una determinada zona y, por consiguiente, se demore más su lectura por parte del simulador, pese a que no se necesita tanta información de esa zona.

Debido a esto, se implementó la funcionalidad de realizar filtros geográficos a los mapas ya descargados. De esta forma, si se ha

descargado un mapa con excesiva información (muchos nodos), no es necesario volver a descargar el fichero XML y las imágenes que lo componen de un área geográfica más pequeña, sino que se puede aplicar un filtro geográfico que limite el área a analizar por el simulador.

Se puede guardar un filtro geográfico, se puede cargar otro filtro usado previamente y se puede determinar escribiendo la latitud y longitud de los límites o incluso indicando los límites de forma gráfica (Ver Anexo "A.12. Aplicar un filtro geográfico a un mapa").

### 4.2.4. Interfaz de usuario

Cada vez que se quería realizar un mejor control de la simulación y configuración de la misma, la interfaz no lo facilitaba. Por lo tanto, se consideró oportuno mejorarla de forma que diera la posibilidad de definir escenarios de simulación y controlar la simulación de una forma fácil y rápida y que además permitiera acceder a las demás funcionalidades implementadas en este PFC (ver Figura 16).

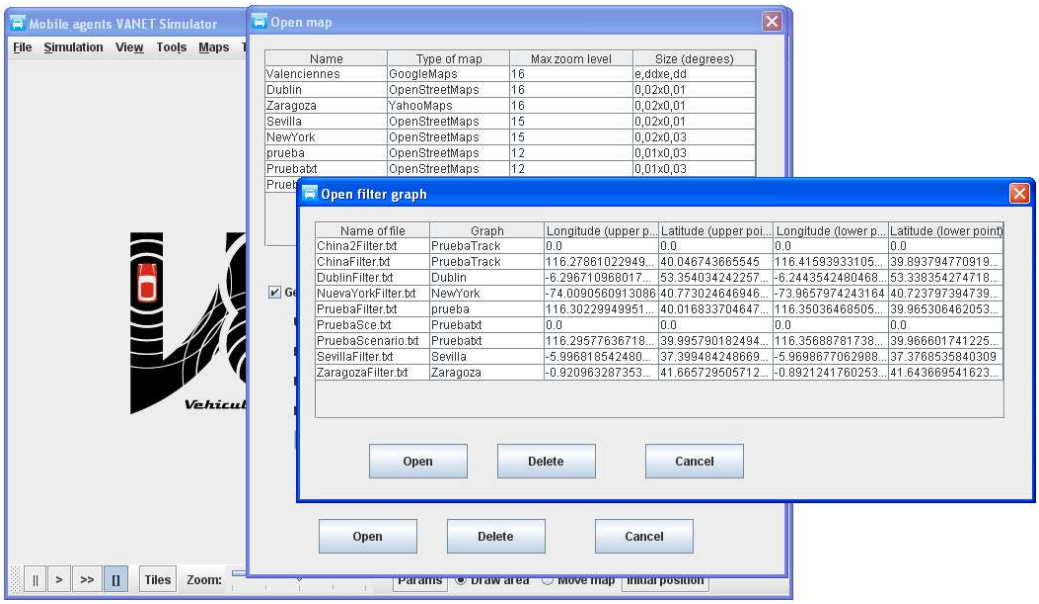


**Figura 16 – Ventanas del simulador**

La interfaz permite realizar las siguientes acciones:

- Controlar la simulación a partir de una barra que contiene diversos botones de control: parar la simulación actual, iniciar una simulación, pausar la simulación, simular de forma rápida, abrir una ventana en la que se pueden modificar parámetros de la simulación, mostrar las imágenes del mapa de la simulación (*tiles*), acercar el zoom, alejar el zoom, centrar la vista en el mapa y seleccionar si se quiere dibujar un área de monitorización o desplazarse por el mapa (como permiten otros mapas como GoogleMaps).

- Cambiar la zona de visualización del mapa a través de una barra de desplazamiento o través del cursor.
- Visualizar información (nombre del vehículo, distancia recorrida por el mismo, velocidad actual, número de agentes que han pasado por el vehículo, estrategia de movilidad, nombre del dispositivo del vehículo y número actual de agentes que tiene) sobre un vehículo determinado haciendo doble clic sobre el vehículo en cuestión.
- Elegir la información que se desea ver cuando se realice el doble clic sobre un vehículo.
- Modificar los siguientes parámetros de simulación: estrategia de salto de los agentes móviles, velocidad de los vehículos en km/h, estrategia de movilidad de los vehículos, mínima y máxima distancia a la que se tiene que encontrar el punto destino en una estrategia basada en rutas, número de vehículos, cobertura de los dispositivos de los vehículos, mostrar las rutas que siguen los vehículos o no, mostrar información acerca de los nodos o no, incluir nodos de soporte en la comunicaciones al realizar la simulación, cobertura de los dispositivos que se encuentran en los nodos de soporte y en los que se encuentran asociados a un nodo.
- A través de una barra de menú, además de controlar la simulación se permite acceder a diversas herramientas, como por ejemplo la descarga de mapas o la importación de trazas GPS reales.
- Mostrar información acerca del simulador y de la simulación.

Para obtener más información de la interfaz, consultar Anexo A. “Manual de usuario”.

### 4.2.5. Herramienta para importar trazas GPS

Una traza está compuesta por una sucesión de puntos en la que cada punto contiene diversa información. La información que nos interesa es: la localización geográfica de ese punto, que viene indicada con sus coordenadas geográficas (latitud, longitud y altitud), la fecha y hora en la que fue registrada esa posición y la velocidad a la que se movía el objeto que registró la posición. Según el formato del fichero de trazas puede que no aparezca la velocidad, y por tanto habría que calcularla a partir de la fecha y hora de dos puntos consecutivos.

Con esta herramienta se importa el fichero o ficheros de trazas que el usuario elige y se le asigna a cada vehículo una de las trazas. De esta forma cada vehículo se mueve independientemente (posiciones y velocidades). Existen diversas restricciones a la hora de importar las trazas. Pueden suceder los siguientes casos:

- Número de vehículos (V) > número de trazas obtenidas (T).
- V < T.

Dependiendo de la configuración que haya elegido el usuario se actuará de una forma u otra. Si sucede el primer caso (V > T), se avisará al usuario de ello. Si sucede el segundo caso (V < T), se avisará al usuario de ello y si el usuario ha decidido ignorar el número de vehículos se crearán tantos vehículos con su respectiva traza como trazas obtenidas (T). Si el usuario no ha decidido ignorar el número de vehículos, se usarán V vehículos y V trazas (las V primeras trazas leídas).

Además de estos casos especiales, existen varios filtros que criban las trazas GPS leídas. Los filtros que se pueden aplicar a las trazas GPS son los siguientes:

- **Filtro temporal**: se puede elegir el filtro temporal de duración o el filtro temporal de intervalo de tiempo. El filtro de duración admite puntos de la traza comprendidos entre la fecha inicial (f1) del primer punto y la fecha final determinada por la suma de la fecha inicial y la duración elegida (f2 = f1 + duración). La duración la indica el usuario en horas, minutos y segundos. El filtro de intervalo de tiempo admite sólo puntos de la traza comprendidos entre la fecha inicial y la fecha final determinada por el usuario.
- **Filtro de velocidad**: con este filtro se indica la velocidad mínima que debe tener un punto de la traza para ser aceptado e importado. De esta forma, si sólo se está interesado en trazas de vehículos por carretera, con este filtro se omitirían las trazas registradas por viandantes, ya que su velocidad será baja.
- **Filtro geográfico**: este filtro es similar al filtro geográfico aplicado en la lectura de los ficheros XML. Para aplicar este filtro hay que indicar el área geográfica que se desea, que vendrá determinada por dos puntos (uno en la posición superior izquierda y otro en la inferior derecha). Los puntos de la traza que estén situados dentro de esa área será aceptados e importados (ver Figura 17).

Al igual que con los filtros geográficos aplicados a los mapas, estos filtros se pueden guardar para posteriormente usarlos en otras importaciones de trazas GPS (ver Anexo A.14 “Cómo importar trazas GPS” y Anexo B.7 “Fichero de trazas GPS reales”).

En el Anexo B.7 se explican todos los formatos de los ficheros de trazas GPS reconocidos por el simulador. En el Anexo A.14 se explica cómo importar trazas de otros formatos o de otro simulador.

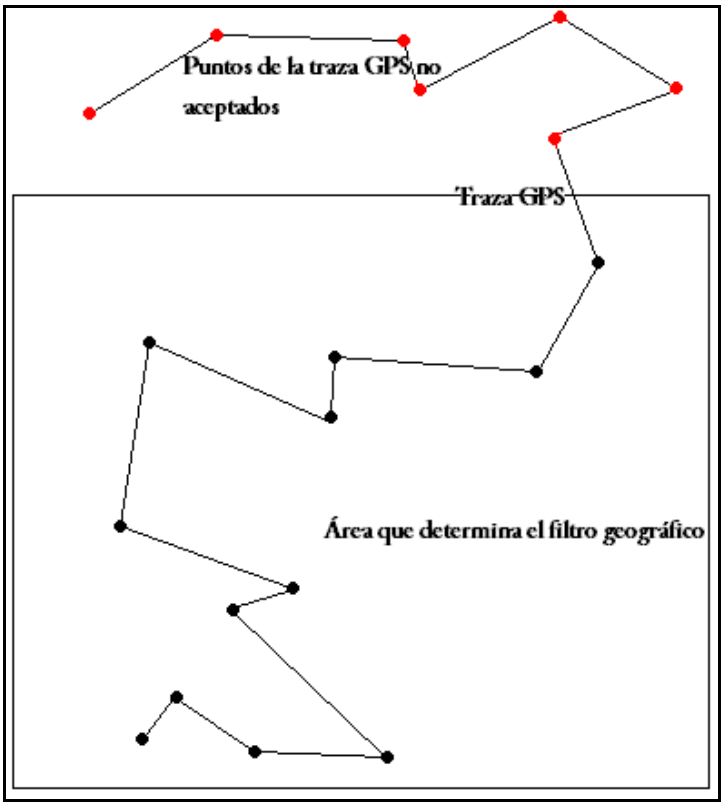


**Figura 17 – Ejemplo filtro geográfico de una traza GPS**

En la Figura 18 se muestra un diagrama de clases esquemático donde se puede ver el diseño de las clases encargadas de leer los diferentes formatos de fichero de trazas GPS. Este diseño está pensado para facilitar la extensibilidad. Si se desea más información acerca de cómo incorporar nuevos formatos de trazas GPS al simulador consúltese el Anexo A.14.2 “Importar trazas GPS de formato desconocido”.

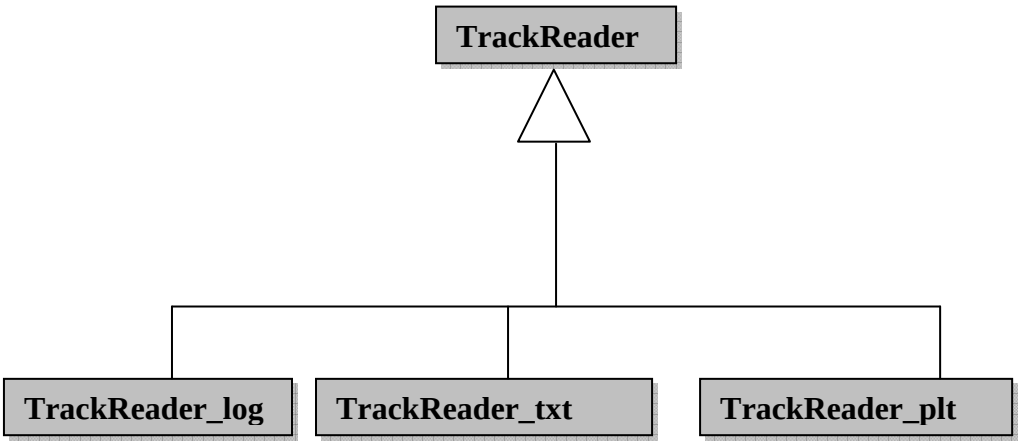


**Figura 18 – Diagrama de clases esquemático de las clases encargadas de leer los ficheros de trazas GPS**

### 4.2.6. Nodos de soporte para las comunicaciones

Como último aspecto a comentar sobre las herramientas implementadas está el permitir añadir nodos de soporte en las comunicaciones entre los vehículos.

En una VANET, los vehículos se comunican entre sí. Pero ¿qué pasaría en las comunicaciones si se añadieran dispositivos fijos de soporte que ayudaran en las comunicaciones a los vehículos?

Debido a esta pregunta, se consideró interesante añadir una herramienta que proporcionara la posibilidad de incluir dispositivos que ayudaran en las comunicaciones a los vehículos. De esta forma, si se está realizando una simulación con agentes móviles, el agente móvil puede saltar a un dispositivo fijo de soporte si lo considera oportuno según su estrategia de salto y cuando pase un vehículo saltar sobre él. O si se está realizando una simulación en la que los vehículos se transfieren información sobre cualquier evento pueden utilizar estos dispositivos fijos de soporte, situados en los viales, para facilitar que la información llegue hasta los destinatarios.

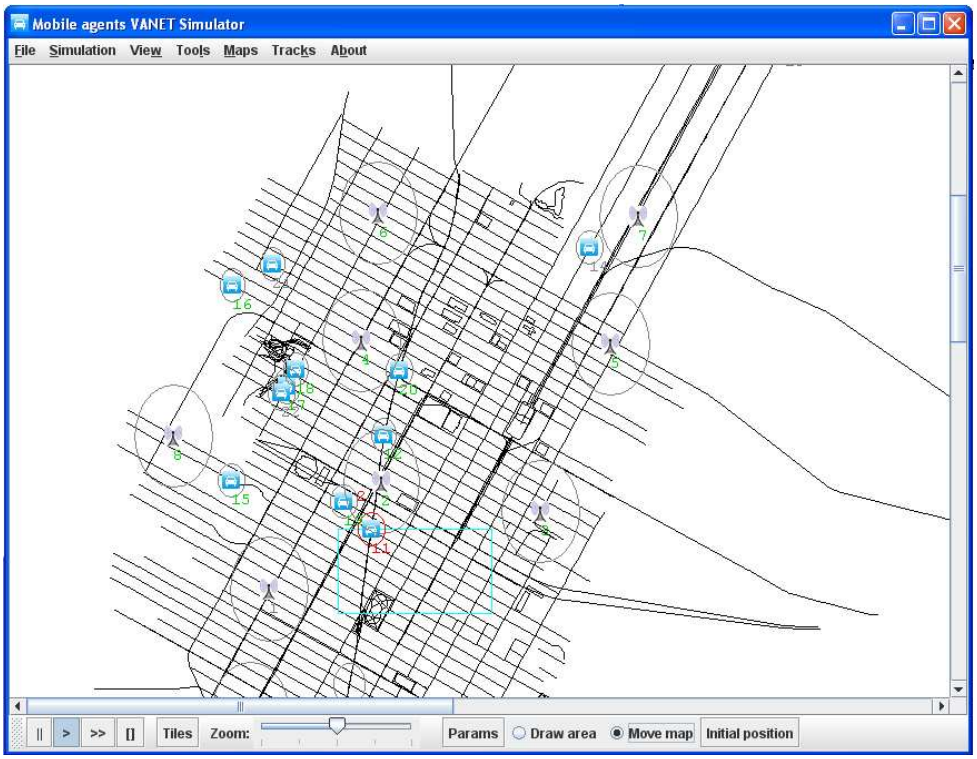


**Figura 19 – Imagen del simulador del PFC con nodos de soporte para las comunicaciones**

En la Figura 19 se muestra una simulación en la que aparecen varios dispositivos fijos de soporte para las comunicaciones de los vehículos en un fragmento del mapa de Manhattan. Para añadir los elementos fijos de soporte a un mapa determinado hace falta tener un fichero de texto en el que se indica la posición de cada dispositivo fijo de

soporte y su cobertura en metros (ver Anexo B.4 “Fichero de los elementos fijos de soporte” y Anexo A.16 “Cómo añadir nodos fijos de soporte de comunicaciones”). Cada mapa tiene uno o ningún fichero de dispositivos fijos de soporte. De esta forma es muy fácil añadir dispositivos fijos de soporte a las comunicaciones entre los vehículos.

### 4.2.7. Sistema de ficheros

El sistema gestiona 8 tipos de ficheros (ver el Anexo B “Formatos de los ficheros”), de los que 2 son en formato XML.

- **XML**: Fichero de configuración, fichero de información OSM.
- **Otro formato**: fichero de área de monitorización, fichero de dispositivos fijos de soporte, fichero de información de los mapas descargados, fichero de trazas GPS, fichero de registro y fichero para guardar los filtros.

### 4.2.8. Visualización de información de un vehículo

El simulador permite visualizar en cualquier momento de una simulación información acerca del vehículo que se desee. Para ello, basta con hacer doble clic sobre el vehículo que se quiera y aparecerá una pequeña ventana con información relativa al vehículo. La información que se muestra depende de la configuración del usuario ya que existe un menú donde se puede elegir qué información mostrar. La información que se puede mostrar es la siguiente:

- **Nombre del vehículo (Name)**: es el identificador que el simulador asigna a cada vehículo. El identificador consta de la palabra “Car” seguida de un identificador numérico. Por ejemplo, Car7.
- **Distancia recorrida (Distance covered)**: se indica la distancia recorrida en metros por el vehículo en su movimiento por el mapa.
- **Velocidad (Speed)**: velocidad actual del vehículo. Normalmente la velocidad es constante para todos los vehículos exceptuando cuando siguen una traza de movimiento.
- **Número de agentes (Number of agents)**: indica el número de agentes móviles que han pasado por el vehículo. Cuenta el número de veces que un agente ha estado en el vehículo, por lo que el agente móvil puede ser el mismo.
- **Estrategia de movilidad (Mobility strategy)**: estrategia de movilidad que rige el movimiento del vehículo.
- **Nombre del dispositivo del vehículo (Device name):** indica el identificador del dispositivo integrado en cada

vehículo. El identificador es un número entero asignado por el simulador.

- **Número actual de agentes (Current number of agents):** indica el número actual de agentes móviles que se encuentran en el vehículo.

Para más información consúltese el Anexo A, secciones A.8 “Menú de configuración de la información de un vehículo” y A.9 “Cómo visualizar información de un vehículo”.

### 4.2.9. Conversión de coordenadas

Debido a que se trabaja con coordenadas geográficas (latitud y longitud) extraídas de los ficheros XML de OpenStreetMap o de los ficheros de trazas GPS y después hay que trabajar sobre las coordenadas de la pantalla, es necesario realizar conversiones en ambos sentidos. Para ello, antes es necesario poder obtener el identificador de *tile* 256 x 256 píxeles a partir de la latitud y longitud y viceversa [40].

Lo que se va a explicar a continuación ya estaba implementado en el prototipo anterior, pero fue imprescindible comprenderlo para implementar diversas funcionalidades, como la descarga de mapas o el movimiento por un mapa con el cursor.

En la Sección 4.2.3 “Herramienta de descarga de mapas” se explicaba que las *tiles* están organizadas como una matriz o cuadrícula para formar un mapa. Por lo tanto, necesitamos saber qué posición x e y de esa matriz ocupa una *tile* para poder acceder a ella. Para obtener esas posiciones, se aplica la formula del recuadro de abajo. Para cada zoom hay que volver a recalcular lo que se va a explicar a continuación.

Queremos obtener qué *tile* es la que se encuentra en la parte superior izquierda (xtile0, ytile0). Ésta se obtiene a partir del valor mínimo de longitud del fichero .osm y del valor máximo de latitud del fichero .osm.

Se calcula de la siguiente forma (proyección de Mercator [72]):

```
n = 2 ^ zoom
xtile = ((lon_deg + 180) / 360) * n
ytile = (1 - (log(tan(lat_rad) + sec(lat_rad)) / π)) / 2 * n
```

Una vez que se tiene la *tile* inicial (xtile0, ytile0), hay que obtener la latitud y longitud a partir de esta *tile* y así conseguiremos el punto de referencia (latitud0 y longitud0).

```
n = 2 ^ zoom
longitud0 = xtile0 / n * 360.0 - 180.0
lat_rad = arctan(sinh(π * (1 - 2 * ytile0 / n)))
latitud0 = lat_rad * 180.0 / π
```

A continuación, queremos obtener la anchura en grados de una *tile*. Por lo tanto, obtendremos la coordenada longitud de la siguiente *tile* en el eje x, es decir, xtile0 +1. Esto lo conseguiremos con las ecuaciones anteriores y obtendremos “longitud1”. Obtenemos la diferencia entre

"longitud1" y "longitud0" y después se divide para 256 píxeles que tiene cada tile de anchura ((longitud1 – longitud0) / 256). De esta forma obtendremos la relación grados/píxel para las coordenadas x.

Si se realiza este mismo proceso en lo referente a la latitud se obtendrá la relación grados/píxel para las coordenadas y.

Así que cuando se quiere pasar de coordenadas GPS a coordenadas de la pantalla, hay que calcular la relación y después aplicar lo siguiente:

Coord x = (longitud a convertir – longitud0) / relación grado-píxel de coordenada x

Coord y = (latitud a convertir – latitud0) / relación grado-píxel de coordenada y

Y para pasar de coordenadas de pantalla a coordenadas GPS hay que aplicar lo siguiente:

longitud = longitud0 + (coord x de la pantalla * relación grado-píxel de coordenada x)

latitud = latitud0 + (coord y de la pantalla * relación grado-píxel de coordenada y)

### 4.2.10. Mejora en la lectura del mapa y organización del grafo

Se realizó una mejora en la lectura del fichero con la información sobre el mapa (fichero XML descargado de OpenStreetMap) y la formación del grafo. Para formar el grafo de un mapa se usaban los identificadores (números enteros) de los nodos que aparecían en el fichero. Estos identificadores, como son únicos, se usan para indicar la posición de cada nodo dentro del vector donde se guardan todos los nodos. Por lo tanto, se necesita un vector cuyo tamaño depende del identificador máximo. Y además puede darse el caso de que el vector se encuentre casi vacío y sus componentes estén dispersas.

Por ejemplo, podemos tener 3 nodos con identificadores (0, 498 y 9.999), y según lo explicado anteriormente, tendríamos un vector de 10.000 componentes en el que sólo habría 3 posiciones ocupadas.

Debido a que este sistema es ineficiente, y a que debido a la descarga de nuevos mapas aparecen identificadores todavía más grandes (por lo que el vector todavía estaría más vacío) se optó por otro sistema de organización.

En este nuevo sistema se almacenan los nodos según se leen en posiciones consecutivas y se les asigna un nuevo identificador según la nueva posición que ocupan en el vector de nodos. Para esto, en el proceso de lectura del fichero del mapa es necesaria la existencia de una tabla de traducción entre los identificadores iniciales del fichero y los identificadores asignados con este nuevo proceso.

Con este nuevo método de organización se puede tener un vector con todas sus componentes consecutivas, con el tamaño adaptado al número de nodos y con el tamaño del vector independiente del identificador máximo de un nodo.

# 5. PRUEBAS Y PROBLEMAS ENCONTRADOS

En esta sección se van a tratar las pruebas realizadas y los problemas que se encontraron.

## 5.1. Pruebas realizadas

Las pruebas a las que fue sometido el simulador fueron las siguientes:

- **Pruebas unitarias**: estas pruebas se realizaron para comprobar el buen funcionamiento de las clases y las funcionalidades implementadas de forma individual.

- **Pruebas de integración**: se realizaron pruebas para ver la comunicación entre las clases. Por ejemplo, la clase encargada de leer el fichero de configuración debe cargar los parámetros que lee para que la simulación se lleve a cabo con esos parámetros de simulación.

- **Pruebas de sistema**: este tipo de pruebas se realizaron una vez que se finalizó la implementación del simulador y se probó todo el simulador en general, haciendo especial hincapié en que el simulador tuviera un correcto funcionamiento en sistemas operativos Linux y Windows.

- **Pruebas de aceptación**: cada vez que se finalizaba la implementación de una funcionalidad, se enviaba el simulador al tutor del PFC y éste se lo enviaba también al desarrollador del prototipo anterior (Óscar Urra Cuairán) para que comprobaran el correcto funcionamiento y otros aspectos del simulador. Una vez finalizado el proyecto, también se realizó este proceso de pruebas para realizar la aceptación final.

Las pruebas se realizaron en un sistema operativo Windows XP Service pack 3 desde el propio entorno de programación Eclipse [58, 59] y a través de scripts de “compilación” y ejecución y el JDK 1.6.0_18 [19] y en una distribución Debian del sistema operativo GNU/Linux (debian-live-503-i386-gnome-desktop) [20, 21].

## 5.2. Problemas encontrados

Los problemas encontrados a lo largo de la fase de desarrollo y de la fase de pruebas que cabe destacar, son los siguientes:

- Problema en el cálculo de rutas debido a su ineficiencia. Debido a que puede haber una gran cantidad de vehículos queriendo crear su ruta, en el modo gráfico se veía que la simulación no era todo lo fluida que se quería. Para solucionar este problema se aplicaron las técnicas de “heurística”, “árbol” y “priority queue”.

- Cuando se paraba una simulación y se iniciaba otra sin cerrar el simulador, y esta operación se realizaba varias veces, el simulador se ralentizaba mucho debido a que no todos los *threads* acababan su ejecución. Para solucionar esto se tuvo que hacer una sincronización de forma que cuando se paraba la ejecución, no se dejaba iniciar otra hasta que no se hubieran matado todos los *threads*.

- Cuando se estaba cargando un nuevo grafo y se insistía sobre el botón de "play" a veces se iniciaba otra simulación sobre el grafo que no era y daba errores. Se solucionó sincronizando los *threads* de forma que hasta que no se carga el grafo es imposible comenzar otra simulación.

- Creación infinita de rutas. A veces cuando un vehículo intentaba crear una ruta hasta el punto destino se quedaba bloqueado en la creación de la ruta. Esto se debía a que el nodo origen y el nodo destino pertenecían a grafos diferentes. Esto se solucionó de forma que cuando en la creación de la ruta se detecta que no se puede llegar hasta el otro nodo, se vuelve a generar otro nodo destino. Si este problema sucede cuando se generan los vehículos, entonces también se genera otro nodo origen para el vehículo.

- Cuando se están cargando las *tiles* de un mapa y el usuario mueve el mapa con el cursor (función "Move map") a veces las *tiles* se descolocaban. Se solucionó sincronizando los *threads*, es decir, hasta que no se cargan todas las *tiles* no se permite mover el mapa con el cursor.

# 6. CONCLUSIONES

En este último capítulo se van a comentar los resultados obtenidos de este PFC, las posibles futuras ampliaciones de este simulador y por último una valoración personal de todo el trabajo.

## 6.1. Resultados obtenidos

Se puede concluir que se han alcanzado todos los objetivos que se establecieron al comienzo del PFC, incluyendo el último punto que trataba sobre los nodos fijos de soporte de las comunicaciones entre los vehículos, en el que se dijo que se consideraría el interés de añadir una herramienta que permitiera añadir estos elementos en las simulaciones.

Se ha desarrollado un simulador que permite seleccionar diferentes estrategias de movilidad y otras opciones de la simulación como, por ejemplo la cobertura del dispositivo integrado de un vehículo. Además, se puede descargar el mapa que se quiera y añadirlo en una simulación, importar trazas GPS y aplicarles diferentes filtros. También se pueden añadir a las simulaciones elementos fijos de soporte para las comunicaciones entre los vehículos. Y todo esto se puede controlar de una forma fácil a través de una interfaz amigable y de un sistema de ficheros de configuración.

Este proyecto ha consistido en la mejora de un prototipo ya existente, desarrollado por el SID, cuya finalidad principal era la evaluación de técnicas basadas en agentes móviles. Las mejoras han consistido en añadir diversas herramientas que le proporcionaran funcionalidades que en su momento se consideraron importantes. Por lo tanto, se extendió, generalizó y mejoró. El nuevo simulador lo explotará el grupo SID en sus investigaciones para realizar simulaciones, de manera flexible y cómoda, que permitan evaluar diferentes estrategias para la gestión de datos.

## 6.2. Futuras ampliaciones

Existen bastantes ampliaciones que se pueden añadir a este simulador, a continuación se comentan varias:

- **Exportación de trazas de movimiento**: el simulador sería capaz de exportar las trazas de movimiento de los vehículos de una simulación a través de un fichero de texto o lo que se considerara oportuno, y de esta forma se podrían alimentar otros simuladores.

- **Herramienta de simulación para mapas especiales de aparcamientos**: añadir algún tipo de herramienta que permitiera incluir mapas especiales de aparcamientos para realizar simulaciones sobre ellos.

- **Herramienta de dibujo de mapas personalizados**: añadir una herramienta que permitiera dibujar un mapa o escenario que no existiera en la realidad y por tanto no se pueda descargar.

- **Situar elementos de forma dinámica**: dar la posibilidad de situar elementos de la simulación (vehículos, antenas, etc.) haciendo clic sobre la posición en la que se quiere situar un elemento determinado.

- **Control de los vehículos con el teclado**: dar la posibilidad de controlar el movimiento de un vehículo con el teclado.

- **Insertar el proyecto VESPA (Vehicular Event Sharing with a mobile P2P Architecture) [15] en este simulador**: añadir una herramienta que permita añadir el "Asistente de navegación GPS con gestión de eventos P2P" [24] al simulador, de forma que se pudiera visualizar esta aplicación en un vehículo del simulador. VESPA es un sistema diseñado para compartir información en redes ad hoc de vehículos. El proyecto VESPA propone un sistema de comunicación ad hoc entre redes de vehículos, describe el intercambio de información en forma de estructuras denominadas eventos (aviso de accidentes, aviso de plazas libres de aparcamiento, etc.) y ofrece un método para estimar la relevancia que estos tienen para un vehículo receptor (Encounter Probability).

- **Nuevas estrategias de movilidad**: añadir nuevas estrategias de movilidad basadas en la interacción con el medio, es decir, con los semáforos, límites de velocidad, cambios de carril, etc.

- **Capacidad de envío de eventos por parte de los vehículos**: se le daría la posibilidad al simulador de que los vehículos se pudieran enviar información acerca de la existencia de aparcamientos, accidentes o lugares interesantes, información que podría llevarles a alterar sus rutas de forma dinámica.

- **Sincronización de trazas GPS**: cuando se importen trazas GPS dar la posibilidad de sincronizar los movimientos importados de las trazas entre sí. Por ejemplo, supongamos que se importan 2 trazas GPS (una para cada vehículo). Una traza contiene puntos comprendidos entre el siguiente intervalo temporal de las 12:00 hasta las 15:00 y la otra desde las 13:00 hasta las 17:00. El simulador podría sincronizar el movimiento de los vehículos de forma que el segundo vehículo se empezaría a mover una vez que el primero llegara hasta puntos situados a partir de las 13:00.

Como se puede observar existen numerosas posibles futuras ampliaciones que mejorarían el simulador de este PFC y le conferirían un mayor abanico de posibilidades en las simulaciones.

## 6.3. Valoración personal

Primero tengo que decir que estoy contento con el trabajo realizado, ya que se han cumplido los objetivos que nos propusimos. Además, estoy contento porque cuando comencé el proyecto cada punto que había que afrontar lo veía bastante complejo, pero cuando fui avanzado en el mismo me di cuenta que poco

a poco iba solucionando los diversos problemas que surgían. Todo esto reconforta una vez que observas todo el trabajo finalizado y ves hasta donde has llegado.

La realización de este proyecto ha sido muy gratificante, tanto por la inmersión en un proyecto ya existente que engloba diversas tecnologías, como por todo lo aprendido, como por haberme dado cuenta de que ante los problemas que surgen he podido salir adelante gracias a los conocimientos obtenidos a lo largo de la carrera.

En el apartado técnico, he aprendido mucho. Por una parte, he continuado aprendiendo sobre el lenguaje de programación Java, con el que ya había trabajado en varias asignaturas de la carrera. Por otra, he aprendido conceptos sobre estrategias de movilidad de vehículos, sobre la descarga de mapas y su utilización, sobre distintos formatos de trazas GPS, sobre tratamiento de ficheros XML, conversión de coordenadas GPS a coordenadas de pantalla y viceversa, cálculo de rutas, agentes móviles, redes VANET y sus diversos usos, la monitorización de áreas geográficas, proyecto VESPA.

En definitiva, una buena experiencia.

# BIBLIOGRAFÍA

[1] Oscar Urra, Sergio Ilarri, Eduardo Mena and Thierry Delot.Using Hitchhiker Mobile Agents forEnvironment Monitoring. 7TH International Conference on practical applications of agents and multi-agent systems (PAAMS 2009), Advances in Soft Computing, volume 55/2009, pp 557-566, 2009

[2] Thierry Delot, Nicolas Cenerario, Sergio Ilarri and Sylvain Lecomte. A Coperative Reservation Protocol for Parking Spaces in Vehicular Ad Hoc Networks. Mobility Conference 2009 Proceedings of the 6th International Conference on Mobile Technology, Application & Systems, ISBN: 978-1-60558-536-9, 2009

[3] Bruno Defude, Thierry Delot, Sergio Ilarri, José-Luis Zechinelli-Martini and Nicolas Cenerario. Data aggregation in VANETs: the VESPA approach. First International Workshop on Computational Transportation Science (IWCTS'08), in conjunction with the Fifth Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS'08), Dublin (Ireland), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ISBN 978-963-9799-27-1, pp. 6, July 2008.

[4] Thierry Delot, Nicolas Cenerario and Sergio Ilarri. Estimating the Relevance of Information in Inter-Vehicle Ad Hoc Networks. International Workshop on Sensor Network Technologies for Information Explosion Era (SeNTIE'08), in conjunction with the Ninth International Conference on Mobile Data Management (MDM'08), Beijing (China), IEEE Computer Society, ISBN 978-1-4244-4484-7, pp. 151-158, April 2008.

[5] Nicolas Cenerario, Thierry Delot and Sergio Ilarri. Dissemination of information in inter-vehicle ad hoc networks. 2008 IEEE Intelligent Vehicles Symposium (IV'08), Eindhoven (The Netherlands), IEEE Computer Society, ISBN 978-1-4244-2569-3, pp. 763-768, June 2008.

[6] Korkmaz Gökhan., Ekici Ekici., Ozgüner Füsun. and Ozgüner Ümit. Urban multi-hop broadcast protocol for inter-vehicle communication systems. In Proc. of ACM VANET (2004), Proceedingsof the 1st ACM International workshop on Vehicular ad hoc networks, ISBN 1-58113-922-5, 2004

[7] Tracy Camp, Jeff Boleng and Vanessa Davies. A survey of Mobility Models for Ad Hoc Network Research. Dept. of Math. And Computer Sciences, Colorado School of Mines, Golden, CO, Wireless Communications and Mobile Computing, volume 2, issue 5, pp. 486-502, 10 September 2002

[8] Viswanath Tolety. Load Reduction in Ad Hoc Networks Using Mobile Servers. 1999

[9] Natarajan Meghanathan. Impact of the Gauss-Markov Mobility Model on Network Connectivity, Liftime and Hop Count of Routes for Mobile Ad hoc Networks. Journal of Networks, volume 5, No 5 (2010), pp.509-516, May 2010

[10] Fan Bai, Narayanan Sadagopan, Ahmed Helmy. A framework to systematically analyze the Impact of Mobility on Performance of Routing protocols for Ad hoc Networks.Dept. of Electr. Eng., California Univ., Los Angeles, CA, USA, INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, ISBN 0-7803-7752-4, volume 2, pp. 825-835, July 2003

[11] Aamir Hassan. VANET Simulation (Master's Thesis in Electrical Engineering). School of Information Science, Computer and Electrical Engineering, Halmstad University, May 2009.

[12] C. Spyrou, G. Samaras, E. Pitoura, P. Evripidou. Mobile agents for wireless computing: the convergence of wireless computational models with mobile-agent technologies. Mobile Net-works and Applications 9 (5), pp. 517-528, 2004

[13] Wikipedia: Mobile agent. http://en.wikipedia.org/wiki/Mobile_agent. Última consulta: 06/09/2010

[14] Obtener latitud y longitud de un lugar. http://itouchmap.com/latlong.html. Última consulta: 06/09/2010

[15] Proyecto VESPA. http://www.univ-valenciennes.fr/ROI/SID/tdelot/vespa/. Última consulta: 06/09/2010

[16] GrooveNet: Vehicular Network Virtualization Platform. http://www.seas.upenn.edu/~rahulm/Research/GrooveNet/. Última consulta: 06/09/2010

[17] BonnMotion; BoMoNet Suite. http://web.informatik.uni-bonn.de/IV/BoMoNet/BonnMotion.htm. Última consulta: 06/09/2010

[18] Eclipse. http://www.eclipse.org/downloads/. Última consulta: 06/09/2010

[19] JDK. http://www.oracle.com/technetwork/java/javase/downloads/jdk6-jsp-136632.html Última consulta: 06/09/2010

[20] Debian Live. http://www.k-nabora.com/index.php/blog/InstalaciA-n-de-Debian-Lenny-en-un-pendrive-con-persistencia.html. Última consulta: 06/09/2010

[21] Descarga Debian Live. http://cdimage.debian.org/cdimage/release/current-live/. Última consulta: 06/09/2010

[22] VANET. http://www.vanet.info/. Última consulta: 07/09/2010

[23] Francisco Javier Ubalde Carramiñana. Interfaz adaptable para un sistema de información de eventos para redes de vehículos. Proyecto de Fin de Carrera de la titulación de Ingeniería Informática, Universidad de Zaragoza, Director Sergio Ilarri, Marzo 2009.

[24] Juan Barberán Sebastián. Asistente de navegación GPS con gestión de eventos P2P. Proyecto de Fin de Carrera de la titulación de Ingeniería Informática, Universidad de Zaragoza de, Director Sergio Ilarri, Septiembre 2009.

[25] Wikipedia: OpenStreetMap. http://es.wikipedia.org/wiki/OpenStreetMap. Última consulta: 07/09/2010

[26] OpenStreetMap. http://www.openstreetmap.es/. Última consulta: 07/09/2010

[27] GeoLife. http://research.microsoft.com/en-us/projects/geolife/. Última consulta: 07/09/2010

[28] GeoLife GPS tracks. http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/default.aspx. Última consulta: 07/09/2010

[29] Virtual Herat. http://microsoft-virtual-earth-3d.uptodown.com/. Última consulta: 07/09/2010

[30] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie. Understanding Mobility Based on GPS Data. In Proceedings of ACM conference on Ubiquitous Computing (UbiComp 2008), Seoul, Korea. ACM Press: 312–321, 2008

[31] Yu Zheng, Lizhu Zhang, Xing Xie, Wei-Ying Ma. Mining interesting locations and travel sequences from GPS trajectories. In Proceedings of International conference on World Wild Web (WWW 2009), Madrid Spain. ACM Press: 791-800, 2009

[32] Wikipedia: OTcl. http://en.wikipedia.org/wiki/OTcl. Última consulta: 09/09/2010

[33] Wireless Ad Hoc Network Simulation Partitioning Tool for NS-2 with Connection Pruning Emre Atsan{eatsan@ku.edu.tr}

[34] AIMSUM. http://www.aimsun.com/site/. Última consulta: 09/09/2010

[35] CORSIM. http://mctrans.ce.ufl.edu/featured/tsis/. Última consulta: 09/09/2010

[36] Wikipedia: VISSIM. http://es.wikipedia.org/wiki/VISSIM. Última consulta: 09/09/2010

[37] Jérôme Härri, Marco Fiore, Fethi Filali and Christian Bonnet. DEMO: Simulating Realistic Mobility Patterns for Vehicular Networks with VanetMobiSim

[38] QualNet Simulator Version 3.1 User's Manual. 2000, 2001 by Scalable Network Technologies, Inc,

[39] Jérôme Härri and Marco Fiore. VanetMobiSim – Vehicular Ad hoc NetWork mobility extension to the CanuMobiSim framework. 2005-2006 Institut Eurécom/Politecnico di Torino

[40] Conversiones coordenadas. http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames. Última consulta: 20/09/2010

[41] Formatos ficheros trazas GPS. http://www.fermoll.de/ozi_ce/formate.htm. Última visita: 23/09/2010

[42] GPX. http://www.topografix.com/gpx.asp. Última consulta: 13/09/2010

[43] Wikipedia: GPX. http://en.wikipedia.org/wiki/GPS_eXchange_Format. Última consulta: 23/09/2010

[44] Formato GPX.
http://www.rigacci.org/wiki/doku.php/tecnica/gps_cartografia_gis/gpx Última consulta: 23/09/2010

[45] Formato tcl. http://www.isi.edu/nsnam/ns/tutorial/nsscript5.html Última consulta: 24/09/2010

[46] Formato tcl.
http://ce.sharif.edu/~moshref/myworks/documentpages/mobisim/tracefiles.html Última consulta: 24/09/2010

[47] Ejemplo de fichero tcl.
http://gmsf.svn.sourceforge.net/viewvc/gmsf/src/output/NS2Formatter.java?view=log
Última consulta: 24/09/2010

[48] Formato fichero trazas GPS, plt.
http://www.elgps.com/ozi/formatos_archivos/formato_plt.html
Última consulta: 27/09/2010

[49] Formato fichero trazas GPS, txt.
http://www.nevasport.com/phorum/read.php?10,1346116,1346333
Última consulta: 27/09/2010

[50] Formato fichero trazas GPS, log. User Guide (incluida en la carpeta con las trazas del proyecto GeoLife GPS Trajectories).

[51] Wikipedia: WGS 84. http://es.wikipedia.org/wiki/WGS84 Última consulta: 27/09/2010

[52] OziExplorer. http://www.oziexplorer.com/ Última consulta: 27/09/2010

[53] Wikipedia VANET. http://en.wikipedia.org/wiki/Vehicular_ad-hoc_network. Última consulta: 02/11/2010

[54] Wikipedia: Java.
http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n) Última consulta: 02/11/2010

[55] Web oficial de Java http://www.java.com/es/download/ Última consulta: 02/11/2010

[56] Wikipedia: JDK. http://es.wikipedia.org/wiki/Java_Development_Kit Última consulta: 02/11/2010

[57] Wikipedia: XML. http://es.wikipedia.org/wiki/Xml Ultima consulta: 02/11/2010

[58] Wikipedia: Eclipse. http://es.wikipedia.org/wiki/Eclipse_(software) Última consulta: 02/11/2010

[59] Web de Eclipse. http://www.eclipse.org/ Última consulta: 02/11/2010

[60] Web de Ns-2. http://www.isi.edu/nsnam/ns/ Última consulta: 03/11/2010

[61] Web de REAL. http://www.cs.cornell.edu/skeshav/real/overview.html Última consulta: 03/11/2010

[62] Web de SeaWind. http://www.cs.helsinki.fi/research/iwtcp/seawind/ Última consulta: 03/11/2010

[63] Web de MobiCom. http://www.sigmobile.org/mobicom/ Última consulta: 03/11/2010

[64] Web de SUMO. http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Main_Page Última consulta: 03/11/2010

[65] Web de VanetMobiSim. http://vanet.eurecom.fr/ Última consulta: 03/11/2010

[66] Web de CanuMobiSim http://canu.informatik.uni-stuttgart.de/mobisim/ Última consulta: 03/11/2010

[67] Wikipedia: TIGER. http://en.wikipedia.org/wiki/Topologically_Integrated_Geographic_Encoding_and_Referencing Última consulta: 03/11/2010

[68] Wikipedia: GDF. http://en.wikipedia.org/wiki/Geographic_Data_Files Última consulta: 03/11/2010

[69] Wikipedia: ShapeFile. http://en.wikipedia.org/wiki/Shapefile Última consulta: 03/11/2010

[70] Web de TraNs Lite http://trans.epfl.ch/ Última consulta: 03/11/2010

[71] Wikipedia: algoritmo del camino más corto de Dijkstra http://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra Última consulta: 03/11/2010

[72] Wikipedia: Proyección de Mercator. http://es.wikipedia.org/wiki/Proyecci%C3%B3n_de_Mercator Última consulta: 03/11/2010