View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by UPCommons. Portal del coneixement obert de la UPC

TRABAJO FINAL DE CARRERA

TÍTULO: Simulación de VANETS (Vehicular Ad-Hoc Networks)

AUTOR: Raul Santos Leiva

DIRECTOR: Mari Carmen Domingo Aladrén

FECHA: 21 de Noviembre del 2007

Título: Simulación de VANETS (Vehicular Ad-Hoc Networks)

Autor: Raul Santos Leiva

Director: Mari Carmen Domingo Aladrén

Fecha: 21 de Noviembre del 2007

Resumen

No podemos negar que hoy en día las redes inalámbricas espontáneas compuestas por terminales móviles sin dependencia de ninguna infraestructura están marcando el camino hacia una nueva generación de redes; así ha sido posible que hayan surgido nuevos servicios y prestaciones aplicables a diferentes campos o escenarios dónde hasta hace unos pocos años no era posible ofrecer tal conectividad. En este entorno es dónde encontramos las VANETS (Vehicular Ad-Hoc Networks), redes formadas entre los diferentes vehículos de un escenario determinado con la finalidad de intercambiar información para aumentar el confort y la seguridad de sus tripulantes.

Actualmente el tema de las VANETS está en pleno desarrollo e investigación, de hecho existen varios grupos de trabajo, tanto por parte de las universidades y los gobiernos, como de la industria, que investigan en este campo debido a la multitud de posibles aplicaciones que podría suponer su utilización.

El objetivo de este TFC ha sido el de simular comunicaciones VANETS en su propio entorno mediante las herramientas existentes, haciéndolo siempre lo más próximo a la realidad posible para obtener así resultados fiables.

Mediante la combinación de tres programas específicos, un simulador de red como es NS2 (Network Simulator 2), un simulador de tráfico para los movimientos de los vehículos como es SUMO (Simulation of Urban MObility) y un programa para comunicar los dos anteriores como es MOVE (MObility model generator for VEhicular networks) obtendremos toda una serie de interesantes resultados en función de varios parámetros y del entorno por dónde circulan los vehículos, como es una autopista o bien nuestra recreación del Eixample de Barcelona.

Finalmente, los valores recogidos en gráficos nos confirman la posibilidad de comunicación mediante VANETS, cumpliéndose así los objetivos propuestos y haciendo viable la existencia de este tipo de redes en un futuro próximo.

Title: Simulation of VANETS (Vehicular Ad-Hoc Networks)

Author: Raul Santos Leiva

Director: Mari Carmen Domingo Aladrén

Date: November 21, 2007

Overview

We can't deny that nowadays wireless and spontaneous networks of mobiles terminals without dependency of any infrastructure are the basis for a new generation of networks; thus new services have appeared for different fields or scenes where a few years ago connectivity was impossible. In this environment it's where we find VANETS (Vehicular Ad-Hoc Networks), networks formed by different vehicles of a certain scene with the purpose of exchanging information to increase the comfort and the safety of the crew members.

Nowadays VANETS are in full development and research; in fact, several working groups exist belonging to universities and governments or to the industry, investigating in this field due to the numerous possible applications that it might arise.

The objective of this work of end of career has been to simulate VANET communications in their own environment by means of the existing tools and trying to work as realistic as possible to obtain trustworthy results.

With the combination of three specific programs, a network simulator like NS2 (Network Simulator 2), a traffic simulator to generate the movements of the vehicles like SUMO (Simulation of Urban MObility) and a program to communicate SUMO with NS2 like MOVE (MObility model generator for VEhicular networks) we will obtain various interesting results depending on several parameters and depending on the environment where vehicles circulate, like a highway or our recreation of the Eixample of Barcelona.

Finally, the values gathered in graphs confirm the possibility of communication through VANETS, fulfilling our purposes and making them viable in a nearby future.

“Una de las cosas que siempre me ha dicho mi familia es que es de bien nacido ser agradecido, por eso en este TFC no podía faltar la mención a todas aquellas personas que directa o indirectamente han contribuido al mismo:

En primer lugar agradecer a la directora de este TFC Mari Carmen Aladrén por darme la oportunidad de hacer este proyecto y por su ayuda prestada en todo momento, ya que sin ella este TFC no hubiera sido posible.

También me gustaría agradecer la colaboración que en todo momento ha tenido conmigo mi compañero Josep Canales, quién también está realizando su respectivo TFC sobre VANETS.

Agradecer también a todos mis compañeros y amigos de la EPSC con los que he pasado tantos momentos durante estos años de universidad, y a mis amigos de siempre por estar ahí en los momentos de ocio necesarios también para tirar adelante en la carrera y en un proyecto como éste.

Agradecer el apoyo de mi familia y en especial el de mis padres, que sin su sacrificio y trabajo me hubiera sido imposible cursar esta carrera, y también gracias a mi novia Isa que siempre me ha estado apoyando y escuchando en todo momento.

A todos vosotros muchas gracias.”

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Actualmente las redes inalámbricas espontáneas sin dependencia de ninguna infraestructura como son las redes ad-hoc, se encuentran en plena investigación y desarrollo a causa de su infinidad de posibles nuevas aplicaciones.

Una VANET o Vehicular Ad-Hoc Network, como su propio nombre indica, se trata de una red ad-hoc donde sus nodos se corresponden con vehículos, concretamente estamos hablando de una red del tipo MANET (Mobile Ad-Hoc Network), es decir, una red ad-hoc móvil dónde sus nodos formarán dicha red en pleno movimiento con las dificultades que conllevará esto.

Así pues, nuestra principal motivación es la existencia de varios grupos de trabajo, tanto por parte de las universidades y los gobiernos, como de la industria, que investigan en este campo debido a la multitud de posibles aplicaciones que podría suponer su utilización a la hora de mejorar la seguridad y confort de los tripulantes.

En la primera parte de este TFC, en el capítulo 1 y 2, introduciremos los conceptos y aplicaciones mas importantes sobre las redes ad-hoc y VANETS, seguidamente detallaremos algunas de las herramientas mas importantes existentes para simular VANETS en el capítulo 3.

En el capítulo 4 escogeremos las mejores herramientas para llevar a cabo nuestro objetivo, simular un entorno VANET real para así obtener resultados fiables; en nuestro caso escogeremos el simulador de red NS2 (Network Simulator 2), el simulador de tráfico para los movimientos de los vehículos SUMO (Simulation of Urban MObility) y un programa para comunicar los dos anteriores como es MOVE (MObility model generator for VEhicular networks).

Seguidamente en el capítulo 5 explicaremos los protocolos a utilizar en nuestras comunicaciones; y en el capítulo 6 los parámetros empleados en éstas.

En el capítulo 7 mostraremos y comentaremos los resultados obtenidos; y sacaremos nuestras propias conclusiones a partir de los valores del tráfico que llega con éxito al receptor (goodput) por fuente, es decir, los paquetes o datagramas que llegan con éxito al receptor de entre los que ha enviado cada fuente existente, todo esto en función de varios parámetros y en función del entorno por dónde circulan los vehículos, como es una autopista o bien nuestra recreación del Eixample de Barcelona.

Finalmente, una vez evaluados los resultados mediante gráficas, podremos confirmar la comunicación mediante VANETS, cumpliéndose así los objetivos propuestos y haciendo viable la existencia de este tipo de redes en un futuro próximo.

CAPÍTULO 1. INTRODUCCIÓN A LAS REDES AD-HOC

1.1. ¿Qué son las redes ad-hoc?

Una posible definición conceptual que describa de forma general y resumida lo que es una red ad-hoc, sería la de una red compuesta por terminales fijos y móviles o bien sólo móviles, que no dependan de una infraestructura preexistente, desplegándose de una forma espontánea en un entorno inalámbrico [1]; sin excluir por tanto, que alguno de ellos posea conectividad con cable (véase la figura 1.1).

Lo más interesante de este tipo de redes es la posibilidad de establecer una red entre dispositivos aislados sin la necesidad de tener estaciones base, routers fijos… etc. o tener un administrador del propio sistema, ya que las redes ad-hoc son adaptativas y pueden autoconfigurarse [2].

En nuestro escenario ad-hoc, por tanto, algunos nodos también actuarán como routers, con la finalidad de comunicar dos dispositivos que no se encuentran directamente conectados mediante su propio radio de cobertura de enlace (dos de los protocolos de enlace mas comunes en redes ad-hoc serían el IEEE 802.11 o Bluetooth); lo que se denomina red inalámbrica multisalto, pudiendo existir centenares de nodos, donde cada uno sería capaz de cubrir distancias de 30 a 100 metros en interiores; y de 100 a 300 metros en exteriores [2].

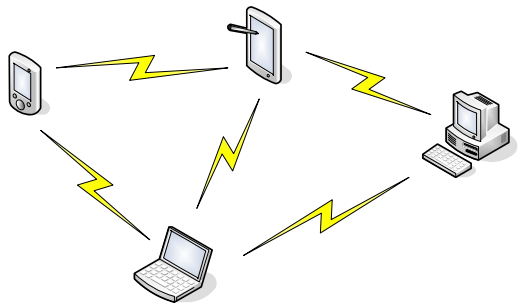

Figura 1.1 Red ad-hoc

Las redes ad-hoc presentan así topologías dinámicas donde los nodos se mueven en el espacio de forma arbitraria, teniendo como gran dificultad y reto la gran variabilidad y la imprecisión que ofrece el medio radio (fading, jitter, atenuación, multicamino, ruido…etc) obligándonos por tanto a este dinamismo frente a cambios repentinos e inesperados (enlaces bidireccionales o unidireccionales, restricción del throughput, caída y recuperación de enlaces…etc.).

Las limitaciones del medio radio serán un fundamento determinante en el funcionamiento de las redes ad-hoc y también en de los protocolos que intervienen (comunicación, encaminamiento, ofrecer QoS (Quality of Service,

calidad de servicio)… etc.), cosa que nos hará comprender su comportamiento y estudio.

En cualquier caso habría que decir que se trata de un campo que actualmente se encuentra en plena investigación y desarrollo, debido a la multitud de posibles aplicaciones que ofrecería esta tecnología (áreas remotas, comunicación en rescates o catástrofes, comunicaciones entre vehículos para prevenir de accidentes ocurridos…etc.) [1].

CAPÍTULO 2. INTRODUCCIÓN A LAS VANETS

2.1. ¿Qué es una VANET?

Una vez introducidos los conceptos mas relevantes sobre las redes ad-hoc, vamos a profundizar en un tipo en concreto, ya que éste será el objeto de estudio de este TFC.

Una VANET o Vehicular Ad-Hoc Network, como su propio nombre indica, se trata de una red ad-hoc donde sus nodos se corresponderían a vehículos (coches, camiones, autobuses …etc.); en este caso, cabría la posibilidad de que dichos nodos formaran la red en pleno movimiento (por ejemplo mientras se circula por una autopista), por tanto, nodos que se mueven de forma arbitraria y que se comunican entre ellos (vehicle-to-vehicle), pudiendo tener también un equipo fijo próximo que formara parte de la red y que también dotará a dicha red de una conexión hacia Internet por ejemplo (vehicle-to-roadside o vehicle-to-enviroment) [3], al igual que hoy acceden nuestros terminales móviles a Internet a través de GPRS o UMTS [4] (Véase figura 2.1).

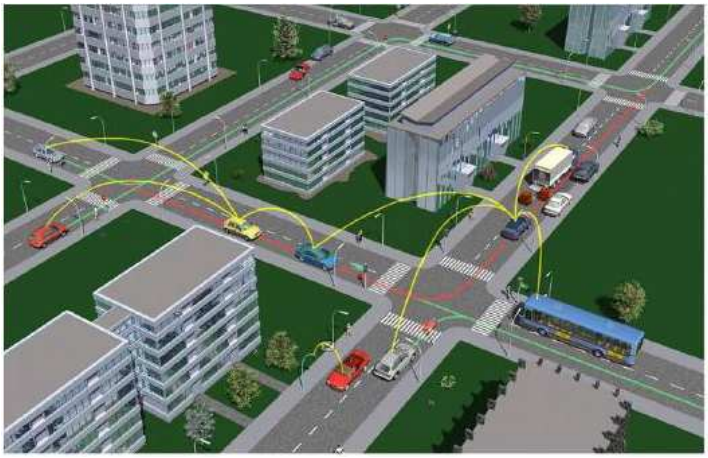

Figura 2.1 Escenario VANET

En cuanto a nomenclatura, cabe decir que una VANET es un tipo de MANET (Mobile Ad-Hoc Network), es decir, una red ad-hoc móvil, por las razones que hemos expuesto anteriormente; aunque cabe diferenciar que MANET describe sobre todo un campo de investigación académico, mientras que el término VANET está mas enfocado a una aplicación en concreto de éstas [5].

Actualmente todo lo que envuelve el tema de las VANETS está en pleno desarrollo e investigación, de hecho, existen varios grupos de trabajo, tanto por parte de las universidades y los gobiernos, como de la industria, que investigan en este campo debido a la multitud de posibles aplicaciones que podría suponer su utilización; algunos de los consorcios son por ejemplo el VSC (USA), C2CCC (Europa), Internet ITS (Japón), Sigmobile (USA) …etc [3] y el propio IEEE, que conjuntamente con el European Car-to-Car Communication

Consortium, está desarrollando el protocolo IEEE 802.11p, lo que se podría ver como una adaptación del propio IEEE 802.11, que actualmente conocemos, a estos escenarios; equivalente al estándar DSRC o Dedicated Short Range Communication, utilizado en Estados Unidos [6].

2.2. Aplicaciones de las redes VANET

Anteriormente hemos hablado a rasgos generales del concepto de VANET, ahora vamos a ver algunas de las aplicaciones más prometedoras de este tipo de redes, mencionando también los grupos de trabajo vinculados a la investigación y desarrollo de éstas.

2.2.1. Car-to-Car Services

Este tipo de servicio es el mas ligado a la seguridad, pensado fundamentalmente para el intercambio de información entre vehículos con la finalidad de prevenir de accidentes ocurridos para evitar colapsos, intercambio del estado del tráfico, de las condiciones de la carretera, del clima, enviar un mensaje de auxilio, mantenimiento de las distancias de seguridad…etc. [6]

Por otro lado, uno de los retos de esta tecnología y por tanto, estudio y desarrollo por parte del C2CCC (véase figura 2.2) es la de ofrecer un entorno escalable, permitir comunicaciones a altas velocidades, ofrecer una diseminación de la información en escenarios donde intervengan mas de 100 nodos, ofrecer seguridad y privacidad en las comunicaciones, QoS (Quality of Service o calidad de servicio) y servicios a tiempo real, tener un entorno de simulación realista…etc. [4].

Figura 2.2 Logotipo del consorcio C2CCC

2.2.2. Car-to-Infrastructure Services

Este servicio está pensado como refuerzo del anteriormente explicado, aparte de ofrecer un intercambio exclusivo de datos entre vehículos, la idea sería la de introducir una serie de estaciones base que podrían ofrecer por ejemplo la información climatológica de una fuente externa, información del trafico…etc. A parte de poder ampliar el radio de amplitud de la red ad-hoc formada, con la posibilidad por ejemplo, de poder enviar a un servicio de emergencia un mensaje de auxilio, ya que cabría la posibilidad que en esa parte de la carretera no hubiera más vehículos en ese momento… etc. Actualmente el grupo de trabajo que profundiza en su estudio y posibles aplicaciones es el CVIS o Cooperative Vehicle-Infrastructure Systems (véase figura 2.3)

Figura 2.3 Logotipo del CVIS

2.2.3. Portal Based Services

Éste quizá es el servicio con más expectativas de futuro dentro de las VANETS en el entorno de ocio y empresarial, aparte de poder ofrecer también parte de los servicios anteriormente descritos.

Este sistema se basa en ofrecer a los vehículos servicios basados en infraestructura (como en el apartado anterior 2.2.2) bajo una plataforma de servicios telemáticos (véase figura 2.4) , como podría ser por ejemplo conexión a Internet desde los vehículos, pagar por tiempo conducido por parte de las aseguradoras ofreciendo así otro tipo de tarificación en el mercado, creación de portales Web de las ciudades donde se podría consultar en tiempo real el estado de las plazas de aparcamiento, hoteles, restaurantes …etc. [6]

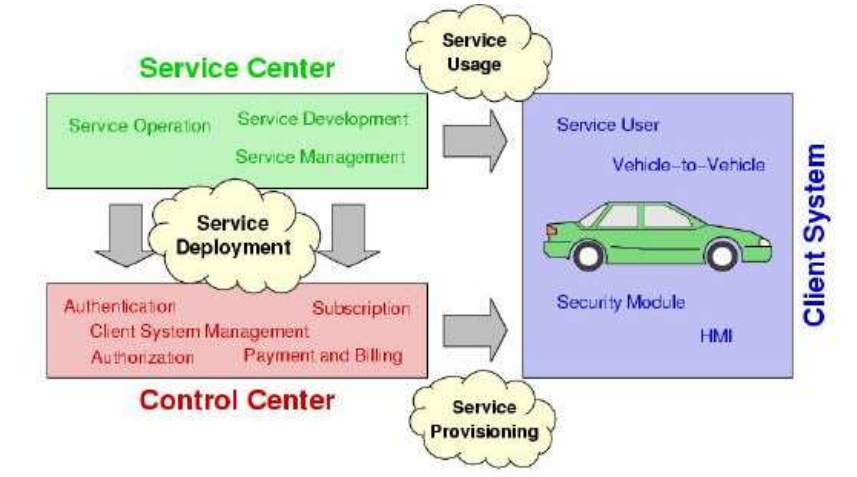

Figura 2.4 Esquema del servicio Portal Based Services

El principal grupo de trabajo que lidera el desarrollo y estudio de este tipo de servicio es el GST o Global System of Telematics (véase figura 2.5), un consorcio formado entre 49 compañías de automóviles (BMW, Crysler, Fiat, Ford, Renault, Volvo…) [7] que pretenden crear un estándar de facto [4] como ocurrió en el caso de Bluetooth.

Figura 2.5 Logotipo del consorcio GST

CAPÍTULO 3. HERRAMIENTAS PARA LA SIMULACIÓN DE VANETS

3.1. Concepto de modelo de movilidad

Como dijimos en el capítulo anterior, uno de los grandes retos de las VANETS es el campo de su simulación para su estudio y desarrollo debido a su propia naturaleza. Hemos de tener en cuenta que nos será muy difícil acercarnos a la generación de modelos de movilidad realistas, ya que nos encontramos en un escenario donde los nodos existentes se moverán de forma arbitraria y cada uno de ellos de forma distinta, sus enlaces irán cambiando de estado a lo largo del tiempo dependiendo del entorno…etc. En definitiva, existe un gran numero de variables a tener en cuenta a la hora de realizar estos modelos (véase figura 3.1) [8].

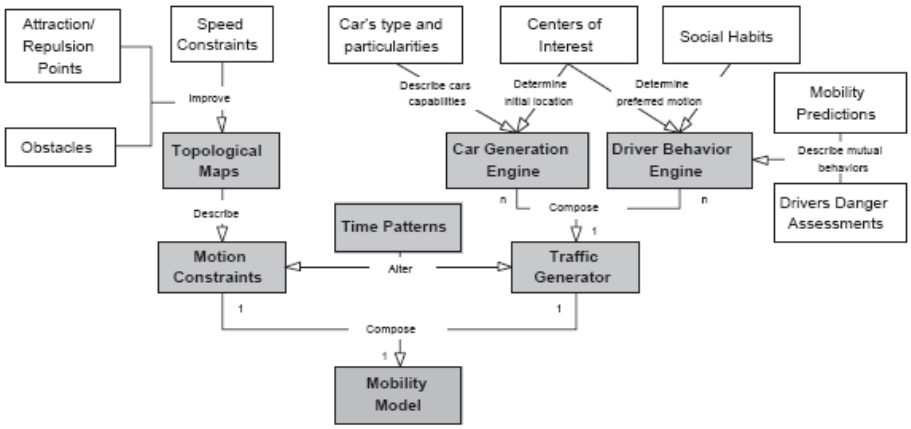

Figura 3.1 Propuesta de esquema donde se recogen posibles variables a tener en cuenta en un modelo de movilidad realista

Aunque para simplificar, las variables que verdaderamente tenemos que tener en cuenta y las que definirán nuestra simulación en mayor medida, serán la conjunción entre el comportamiento de los conductores en relación a la movilidad de cada vehiculo y el estado de la comunicación Wireless de cada enlace existente (véase figura 3.2) [10].

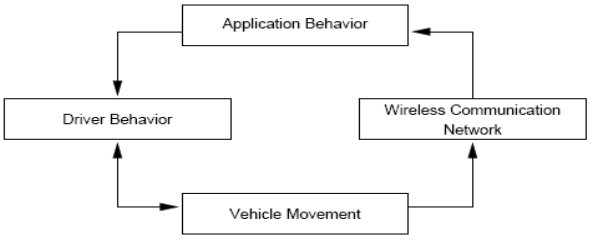

Figura 3.2 Variables más relevantes en nuestro modelo de movilidad

Toda esta interacción de variables y elementos que configuran y determinan nuestra simulación viene dada por la utilización de varias herramientas de forma paralela; en este caso lo más habitual es utilizar un simulador de tráfico, que nos definirá el comportamiento de los nodos, movimiento, condiciones…etc. con un simulador de redes, que nos servirá para evaluar los datos generados por el simulador de tráfico [10] y simular la comunicación Wireless que nos definirá las características de propagación del medio radio, acceso al medio, enrutamiento, errores, seguridad …etc. A este método conjunto de simulación se le denomina, simulación híbrida [4] (véase figura 3.3).

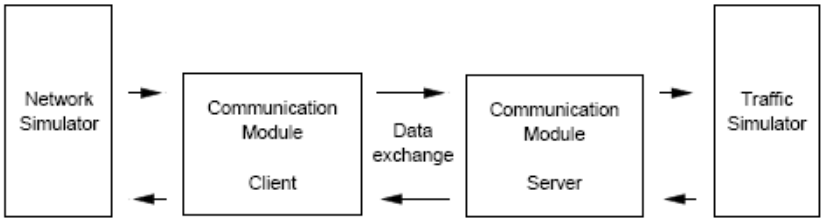

Figura 3.3 Modo de trabajo híbrido, simulador de tráfico trabajando conjuntamente con un simulador de redes externo

3.2. Simuladores de tráfico

En este apartado vamos a describir algunos de los simuladores de tráfico de entornos VANET mas utilizados y aceptados [4]; comentar que la razón por la que hemos escogido éstos frente a otras opciones viene dada por la inclinación por parte de la mayoría de investigadores que se han dedicado a evaluar los diferentes simuladores de tráfico para estos entornos [10]; estos simuladores de tráfico son GrooveSim, Carisma y SUMO [4].

3.2.1. GrooveSim

La finalidad de este simulador de tráfico híbrido es la de ofrecer un modelo de movilidad en un escenario topográfico real a través de mapas digitales (véase figura 3.4). Los desarrolladores del mismo afirman que Groove Simulator es capaz de simular comunicaciones vehicle-to-vehicle en movimiento donde intervengan miles de ellos, obteniendo logs para el registro de los datos de la simulación (pequeños informes donde se registran los datos referentes a la simulación realizada). Uno de los aspectos a comentar es que GrooveSim se trata de un programa propietario y no gratuito [9].

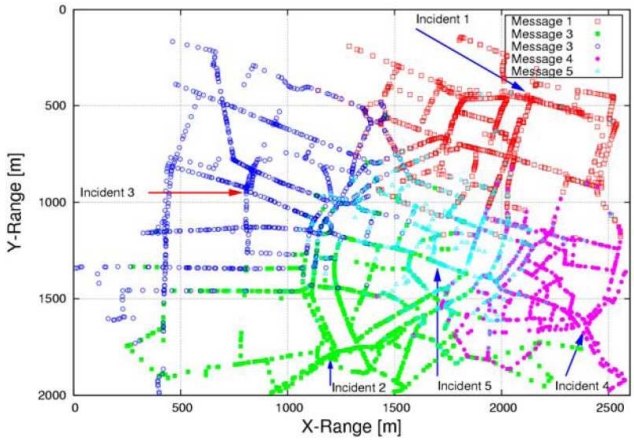

Figura 3.4 Captura del simulador GrooveSim

3.2.2. CARISMA

En este caso estamos ante un simulador, que como el anterior, realiza las simulaciones en un entorno topográfico real tratándose también de un simulador híbrido (véase figura 3.5) [10].

Las principales características de CARISMA o Context-Aware Reflective middleware System for Mobile Applications [11] son la posibilidad de emular escenarios que abarquen unos cientos o unos pocos miles de vehículos, solamente soporta dos carriles por carretera, cada calle posee la misma capacidad y prioridad de tráfico, supone que los edificios existentes están ubicados a lo largo de la calle, actualización de la posición de los vehículos cada segundo durante la simulación, evaluación y asignación de diferentes mecanismos de enrutamiento…etc. sin duda un herramienta muy potente a tener en cuenta. De igual manera que en el caso anterior, Carisma también es un programa propietario y no gratuito. [10].

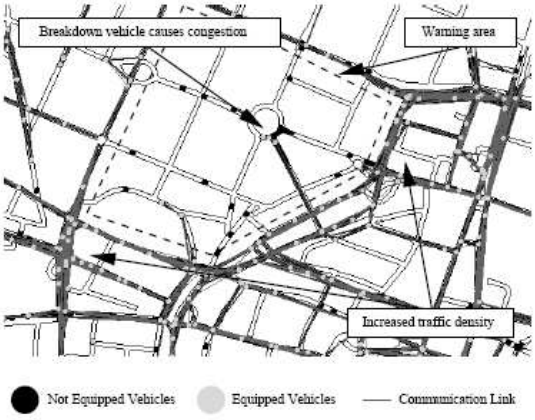

Figura 3.5 Captura de CARISMA

3.2.3. SUMO

Una de las grandes ventajas de SUMO o Simulation of Urban Mobility (véase figura 3.6) es que se trata de un simulador de VANETS de código abierto y gratuito desarrollado por la DLR o *Deutsche Gesellschaft für Luft und Raumfahrt*, es decir, el centro aeroespacial nacional alemán.

Se trata de un simulador que permite definir entornos de movilidad reales gracias a la utilización de mapas digitales, ofrece la posibilidad de utilizar diferentes tipos de vehículos, carreteras que cambian en cuanto a su composición (carriles, velocidad, prioridades… etc.) y un amplio abanico de posibilidades e integración con otros simuladores debido a que es código abierto (véase figura 3.7).

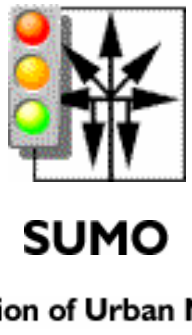

Figura 3.6 Logotipo de SUMO

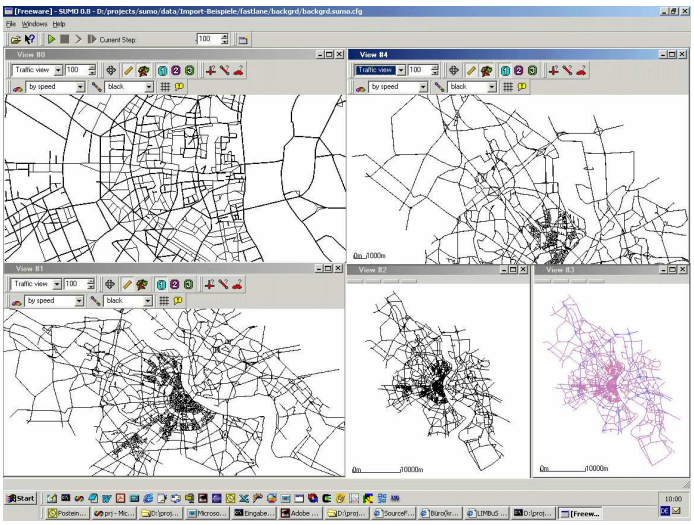

Figura 3.7 Captura de SUMO

Una de las limitaciones de SUMO frente a CARISMA, es el hecho de que éste calcula las rutas de los vehículos antes de realizar la simulación, cosa que dificulta la evaluación de la comunicación vehicle-to-vehicle afectada por variaciones en el comportamiento de los mismos durante la propia simulación,

es decir, que nos será más difícil evaluar a tiempo real la comunicación entre vehículos frente a una variación provocada de su comportamiento de una forma repentina (aceleración, cambio de ruta, frenada …etc.) ya que con SUMO se tiene que prever todo esto antes de hacer la propia simulación [10].

3.2. Simulador de redes

Como hemos comentado anteriormente, nuestro simulador de redes será el encargado de emular el entorno Wireless en nuestro modelo de movilidad, para ello hemos de exigir a éste una serie de prestaciones para su ejecución.

Todos los que se han aventurado a la simulación de VANETS coinciden que el simulador que cumple con las exigencias es NS2 o Network Simulator 2 [10].

3.3.1. NS2

Network Simulator 2 es uno de los simuladores mas ampliamente aceptados en la comunidad científica [10], además da la posibilidad de ser utilizado en conjunción con otros programas, como un simulador de tráfico (CARISMA o SUMO por ejemplo).

NS2 implementa diferentes escenarios Wireless, diferentes modelos de propagación, implementación del protocolo de acceso al medio IEEE 802.11, tiene en cuenta los efectos de rebotes de la señal, como le pasa a esta por ejemplo en la carretera en una comunicación entre dos vehículos…etc. En definitiva, estos aspectos hacen a este simulador más que apto para su utilización en este campo. (Véase figura 3.8)

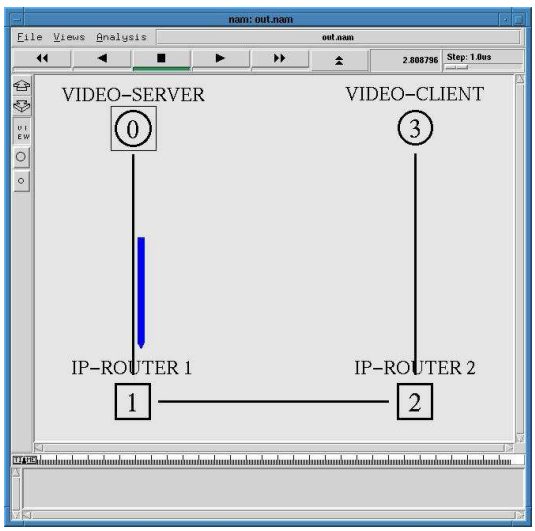

Figura 3.8 Captura de NS2 utilizando el animador de redes NAM

CAPÍTULO 4. ELECCIÓN DE HERRAMIENTAS PARA LA SIMULACIÓN

En este apartado vamos a describir con más profundidad las herramientas que hemos decidido escoger para la simulación del escenario VANET.

Las herramientas escogidas para ello son SUMO como simulador de tráfico, NS2 como simulador de red y MOVE (MObility model generator for VEhicular networks), que será el programa que nos servirá para realizar la comunicación entre NS2 y SUMO, haciendo así posible nuestra simulación híbrida; Y todo esto sobre el sistema operativo Linux, mas concretamente la distribución denominada Fedora Core 6 (véase figura 4.1), ya que en plataformas Red Hat han sido probados los programas que vamos a utilizar.

Figura 4.1 Captura de Fedora Core 6

4.1. SUMO

4.1.1. ¿Por qué SUMO?

Como hemos visto en el apartado anterior, existen varias posibilidades a la hora de escoger un simulador de tráfico. Aunque existen varias opiniones ante la calidad de cada una, la gran parte de los investigadores que han evaluado diferentes simuladores, coinciden en afirmar que SUMO es una herramienta que ofrece unos resultados óptimos y que además se trata de un programa de código abierto (mucha documentación, gratuito…etc.) frente a la mayoría de opciones existentes de pago. Como pasa en todo el software de licencia pública, cada uno tiene la posibilidad de poner su grano de arena para ir mejorando y desarrollando esta herramienta, que aunque ya se encuentra entre una de las principales, es una buena apuesta a tener en cuenta de cara al futuro debido a su utilización y continua mejora por parte del sector científico; por tanto, todas estas razones son las que han hecho que nos decidiéramos a utilizar SUMO.

4.1.2. Más sobre SUMO

En el capítulo anterior describimos de forma generalizada las características más importantes sobre SUMO y sus orígenes (véase figura 4.2), en este apartado vamos a profundizar más sobre este simulador desarrollado en lenguaje C++, más concretamente en su funcionamiento; la definición formal de SUMO sería la de un simulador de tráfico microscópico, que trabaja en espacio continuo y con tiempo discreto.

Figura 4.2 Logotipo de la agencia nacional aeroespacial alemana, la principal entidad desarrolladora de SUMO

Que SUMO es un simulador microscópico quiere decir que se basa en la emulación individual del movimiento de cada vehículo que participa en el escenario, es decir, que SUMO es capaz de adoptar un movimiento diferente a cada vehículo del escenario haciendo así un entorno más próximo a la realidad, dónde el modelo matemático que implementa este comportamiento microscópico está desarrollado por Stefan Krauss (véase figura 4.3).

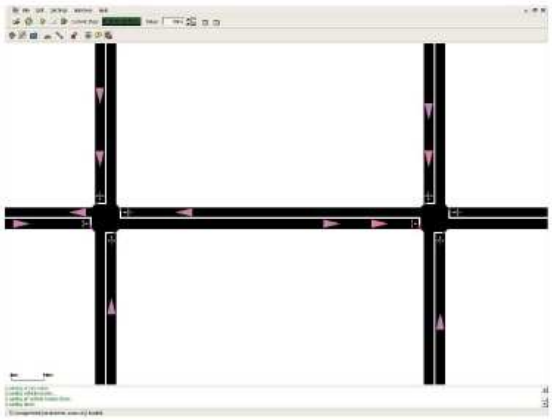

Figura 4.3 Simulación microscópica de SUMO

SUMO trabaja en espacio continuo en el sentido que en cada instante de tiempo discreto de la simulación, es decir solo en unos instantes de tiempo determinados, la posición de cada vehículo está perfectamente descrita, es

decir, que podemos saber la posición exacta de cada vehículo en cualquier instante de tiempo definido en nuestra simulación.

Como dijimos en el capítulo anterior SUMO computa las rutas de los vehículos antes de realizar la simulación con los posibles inconvenientes que comentamos, una vez en la simulación, los vehículos llegan a su destino definido (por el usuario o automáticamente) mediante el trayecto más corto; En un escenario donde actuase un número elevado de vehículos (unos pocos miles de vehículos, según especificaciones de los autores) esto podría causar congestión de tráfico en las calles arteriales del mapa digital utilizado; precisamente para evitar esto, se utiliza un algoritmo denominado “dynamic user assignment” desarrollado por Christian Gawron.

Como contraste comentar otra de las limitaciones de SUMO al tener preestablecidas las rutas de cada vehículo en la simulación, que será el gran uso de memoria de la máquina utilizada, sin la utilización tampoco de ningún tipo de compresión, cosa que se agrava bastante en escenarios donde interviene un gran número de vehículos; éste es, por tanto, uno de los aspectos que se están mejorando de este simulador.

Por ultimo, valdría la pena resaltar que SUMO se puede dividir en pequeños subprogramas que trabajan en conjunción y dónde cada uno tiene su función para llevar a cabo el resultado final, en este caso, nuestro entorno simulado (véase figura 4.4 y 4.5).

Application	Application Name (Windows)	Application Name (Linux/UNIX)	Description
NETCONVERT	netconvert.exe	sumo-netconvert	A network converter/importer
NETGEN	netgen.exe	sumo-netgen	A generator of abstract networks
DFROUTER	dfrouter.exe	sumo-dfrouter	A router using detector flows
DUAROUTER	duarouter.exe	sumo-durarouter	A router for dynamic user assignment
JTRROUTER	jtrrouter.exe	sumo-jtrrouter	A router using junction turning ratios
SUMO	sumo.exe	sumo	The microscopic simulation
GUISIM	guisim.exe	sumo-guisim	The gui-version of the microscopic simulation
POLYCONVERT	polyconvert.exe	sumo-polyconvert	A tool for importing polygons from other formats
other	---	---	---

Figura 4.4 Aplicaciones principales que conforman SUMO

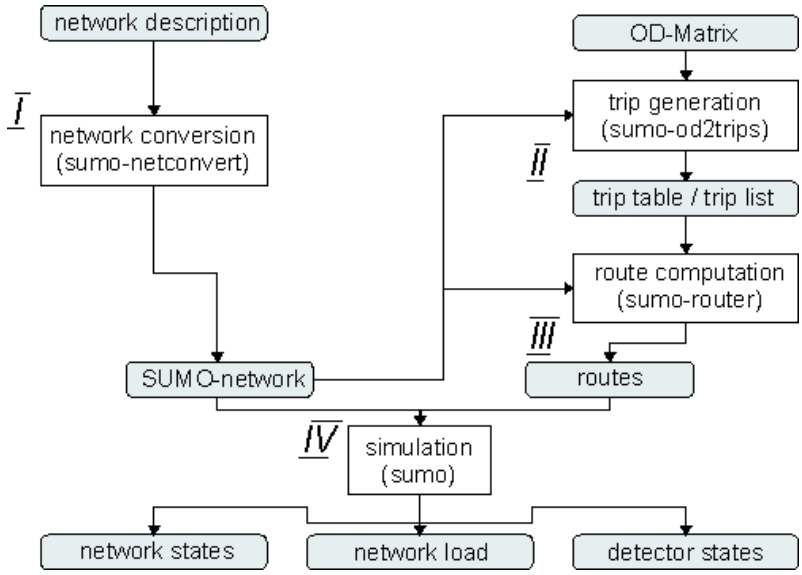

Figura 4.5 Proceso de simulación de SUMO, en gris los datos usados y en blanco las aplicaciones que intervienen

En este apartado hemos explicado los datos más relevantes sobre SUMO, ya que las posibilidades y características de este simulador son mucho más amplias, para su consulta detallada existe la página Web oficial del proyecto con todo tipo de documentación [12].

4.2. NS2

4.2.1. ¿Por qué NS2?

Las razones por las que hemos escogido este simulador de red son básicamente las mismas que en el caso anterior, ya que NS2 es uno de los simuladores de red más aceptados por la comunidad científica y de código abierto, con todos los beneficios que comporta, aparte de que por supuesto, cumple con nuestras exigencias en cuanto a protocolos de comunicación a la hora de simular nuestro escenario VANET.

4.2.2. Más sobre NS2

Network Simulator 2 está desarrollado en lenguaje C++, aceptando los comandos de configuración por parte de los usuarios en Otcl (Object Tool Command Language), un extensión del lenguaje TCL al que nos referiremos de nuevo posteriormente; NS2 es la versión mejorada de Network Simulator, siendo ésta una de las variantes de Real Network simulator, respaldado por DARPA (Defense Advanced Research Projects Agency) (véase figura 4.6), es decir, el departamento de desarrollo del ejército de los Estados Unidos con la colaboración del consorcio VINT (Virtual InterNetwork Testbed) (véase figura 4.6) formado por la Universidad del Sur de California, el centro de desarrollo de Palo Alto y el grupo de investigación de la universidad de Berkeley. Por tanto, se veía con claridad la necesidad de obtener un simulador de redes de altas

prestaciones con tal de simular escenarios de todo tipo y en cualquier entorno posible. Con el paso del tiempo NS fue adoptando una cara más académica y fue donado para que fuera un instrumento más para la investigación, cosa que sirvió para mejorar y desarrollar este simulador con la aportación de diferentes usuarios y entidades, como Sun Microsystems, a lo largo de su historia.

Figura 4.6 Logotipo de DARPA y el consorcio VINT respectivamente

Uno de sus mejores aspectos es su gran funcionalidad, pues resulta capaz de simular la inmensa mayoría de protocolos de comunicación existentes en diversos entornos (LAN o Local Area Network, entornos Wireless, comunicaciones satélite…etc.), además de poseer packs adicionales con nuevos protocolos que se incorporan gracias al desarrollo y soporte de la comunidad científica.

Una de las herramientas anexas a NS2 es NAM o Network Animator basado en el lenguaje Tcl/Tk (Tool Command Language) (véase figura 4.7) desarrollado por Sun Microsystems (véase figura 4.8); su función no es otra que la de dar un aspecto gráfico y animar los escenarios emulados mediante ns-2 (véase figura 4.9).

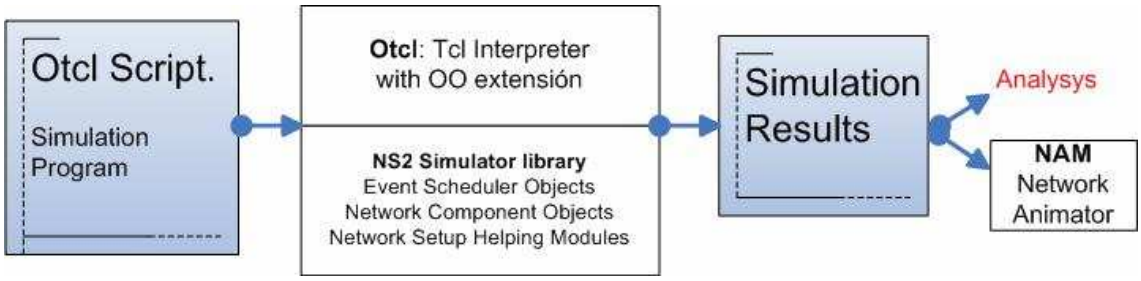

Figura 4.7 Vista simplificada del funcionamiento de NS2 y NAM

.

Figura 4.8 Logotipo de Sun Microsystems

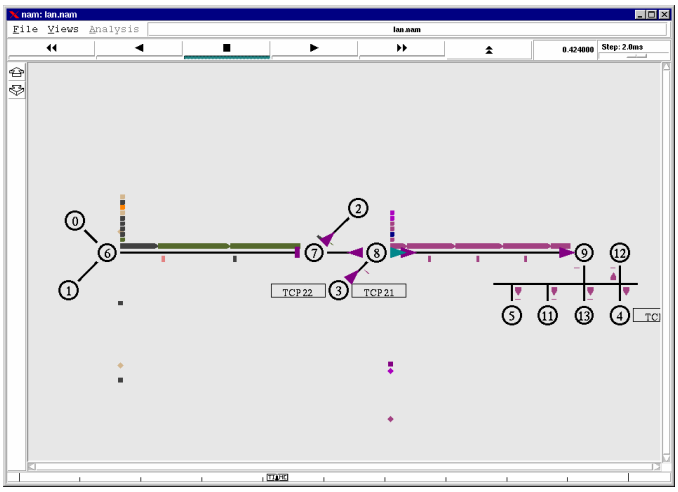

Figura 4.9 Captura de NS2 con el animador NAM

En todo caso, hemos expuesto algunas de las características generales de NS2; existe una amplia documentación del proyecto en su página Web oficial [13].

4.3. MOVE

4.3.1. ¿Por qué MOVE?

La verdad es que aunque hemos encontrado otro programa denominado traNS, que tiene la misma finalidad que éste, hemos optado por MOVE ya que aparte de ofrecer comunicación entre NS2 y SUMO, se implica directamente en la simulación y automatiza mediante su interfaz gráfica algunos de los procesos necesarios para generar la simulación, como son la configuración de rutas, tipos de vehículos…etc. que ahorran al usuario realizar scripts para su configuración (pequeños fragmentos de código que se implementan en el simulador NS-2 para caracterizar la simulación).

4.3.2. Más sobre MOVE

Los creadores de MObility model generator for VEhicular networks han sido Feliz Karnadi, Zhi Hai Mo, pertenecientes al CSE (College of Computer Science and Engineering) y Kun Chan Lan, perteneciente a NICTA (National Information and Communications Technology Australia), quienes han desarrollado este simulador mediante lenguaje JAVA; Su finalidad era la desarrollar un software de código abierto que intercomunicara SUMO y NS2 (entre otros) haciendo de la simulación un tarea fácil, rápida e intuitiva por parte del usuario a partir de su interfaz gráfica (véase figura 4.10).

Figura 4.10 Captura de la interfaz gráfica de MOVE

Entre sus características destaca la posibilidad de importar mapas reales a través de Google Map, configurar los movimientos de los vehículos, velocidades, número de vehículos, capacidad de tráfico y prioridad del mismo para cada calle…etc. (véase figura 4.11).

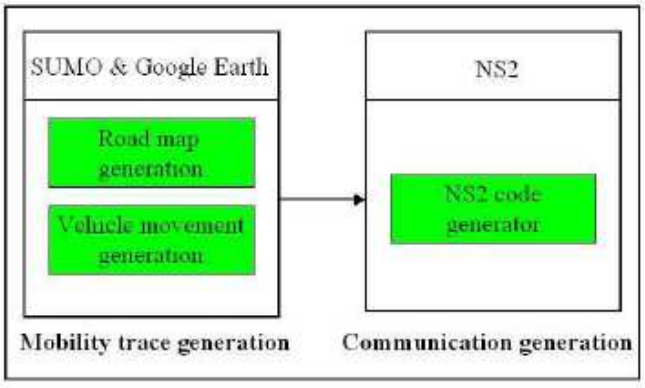

Figura 4.11 Figura que muestra la arquitectura de MOVE

Por otro lado, una de las limitaciones a tener en cuenta y que los autores resaltan de este software es el alto consumo de recursos de la máquina utilizada en escenarios donde intervenga un gran número de vehículos (unos pocos miles de vehículos) debido a que SUMO calcula las rutas de todos los vehículos antes de realizar la simulación, tal y como comentamos en el capítulo anterior [14].

CAPÍTULO 5. ELECCIÓN DE LOS PROTOCOLOS

En este capitulo vamos a describir con profundidad los protocolos escogidos para realizar nuestra simulación. El objetivo principal de este TFC es realizar simulaciones de VANETS aproximándonos lo mas posible a la realidad, lo ideal por tanto hubiera sido utilizar protocolos específicos para VANETS, como por ejemplo el IEEE 802.11p como tecnología de enlace, del que hablamos en el capítulo 2; conjuntamente con un protocolo de routing también específico. El principal inconveniente es que se tratan de tecnologías muy recientes que actualmente aún se encuentran en desarrollo y actualmente no existe ningún código implementado para realizar las simulaciones.

Muchos de los investigadores que han simulado entornos VANET han utilizado como protocolo de enlace el estándar IEEE 802.11b operando en modo DCF (Distributed Coordination Function) como protocolo de enlace (que trae por defecto NS2) y la mayoría coincide en la utilización del protocolo de encaminamiento AODV (Ad hoc On Demand Distance Vector) pensado para redes ad-hoc dónde los nodos se mueven de forma arbitraria, como pasa en las VANETS; nosotros hemos querido dar un paso más y por eso vamos a utilizar el protocolo DYMO (Dynamic On-demand MANET routing protocol), un protocolo que ha surgido recientemente; su base funcional es muy parecida a AODV, aunque añade algunas modificaciones comentadas en el anexo 1 de este TFC.

IEEE 802.11b DCF a nivel de la capa de MAC y DYMO como protocolo de encaminamiento, son los protocolos escogidos de los que hablaremos con mas profundidad a lo largo de este capítulo.

5.1. IEEE 802.11b

Seguidamente vamos a explicar el protocolo IEEE 802.11b, el estándar actualmente utilizado en la gran mayoría de comunicaciones Wireless.

5.1.1. Funcionamiento general

IEEE 802.11b es un protocolo de nivel de enlace que puede operar en modo PCF (Point Coordination Function) o DCF (Distributed Coordination Function) utilizando el protocolo CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance*)* como mecanismo de acceso al medio; alcanza velocidades teóricas de 1-2-5,5 y 11 Mbit/s ,y llega a una tasa máxima de transmisión de 5.9 Mbit/s sobre TCP y 7.1 Mbit/s sobre UDP aproximadamente; utiliza una modulación de la señal denominada DSSS (Direct Sequence Spread Spectrum) (véase figura 5.1) y opera en la banda de 2,4 Ghz (nivel físico) (véase figura 5.2) [15]. De todos estos aspectos trataremos con una mayor profundidad en los siguientes apartados.

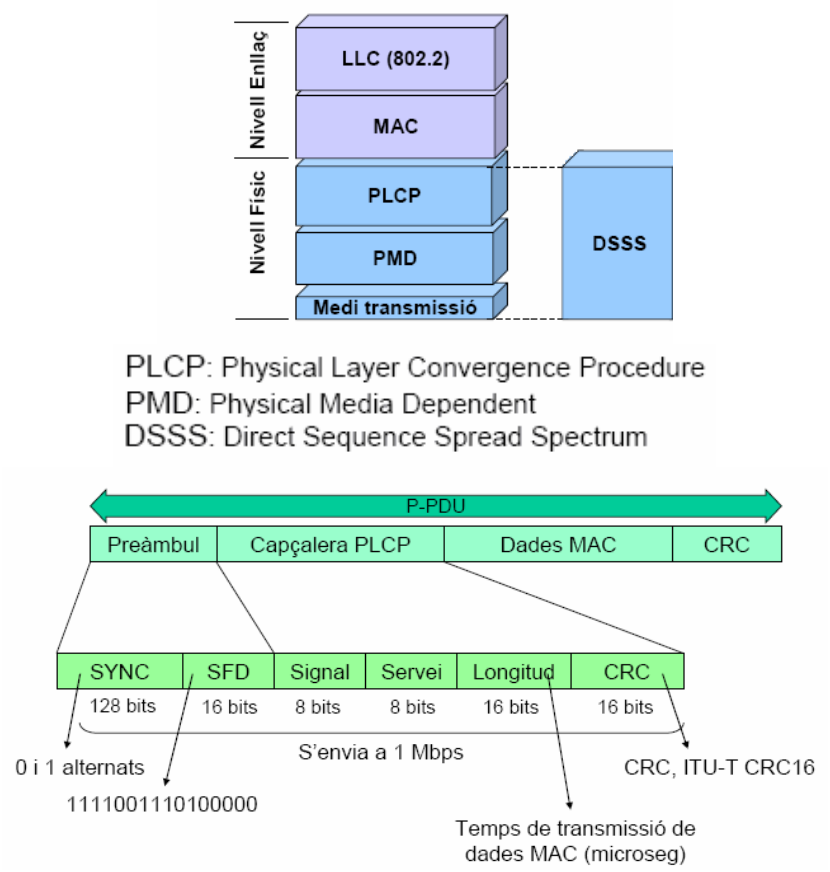

Figura 5.1 Arquitectura de protocolos y formato de la trama P-PDU (capa PLCP) de IEEE 802.11b

Identificador de Canal	Frecuencia en MHz	Dominios Reguladores				
		América (-A)	EMEA (-E)	Israel (-I)	China (-C)	Japón (-J)
1	2412	×	×	—	×	×
2	2417	×	×	—	×	×
3	2422	×	×	×	×	×
4	2427	×	×	×	×	×
5	2432	×	×	×	×	×
6	2437	×	×	×	×	×
7	2442	×	×	×	×	×
8	2447	×	×	×	×	×
9	2452	×	×	×	×	×
10	2457	×	×	—	×	×
11	2462	×	×	—	×	×
12	2467	—	×	—	—	×
13	2472	—	×	—	—	×
14	2484	—	—	—	—	×

Figura 5.2 Los identificadores de canales, frecuencias de canales centrales, y dominios reguladores de cada canal de IEEE 802.11b.

5.1.2. IEEE 802.11b DCF: CSMA/CA

Como hemos comentado anteriormente 802.11b puede trabajar en modo PCF o DCF; en nuestro caso, al tratarse de una red ad-hoc donde la topología es totalmente distribuida resultaría inadecuado y también ineficiente trabajar en modo PCF, el cual se basa en una coordinación de acceso al medio teniendo un nodo central que se encarga de administrar las peticiones basadas en encuesta por parte del resto de miembros de la red. Por este motivo utilizamos el modo distribuido DCF que funciona mediante el mecanismo CSMA/CA.

Para una mayor facilidad de compresión vamos a explicar el funcionamiento básico de CSMA/CA a partir de la siguiente figura (véase figura 5.3):

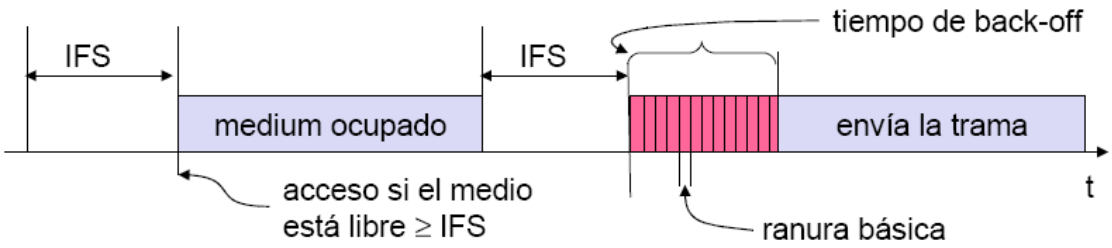

Figura 5.3 Funcionamiento básico de CSMA/CA

Partimos desde el momento en el cual una estación desea enviar una trama, entonces se pone a escuchar el medio; si está libre y continúa así durante un intervalo de tiempo IFS (Inter Frame Space), la estación comenzará a enviar la trama, por el contrario, si el medio ya estuviera ocupado o pasara a estarlo antes de transcurrir el IFS, la estación esperaría a que volviera a estar libre, volvería a esperar el IFS con un tiempo añadido de “back-off” o de “contienda” que será un múltiplo de la ranura elegido de forma uniforme entre 0 y un valor CW, dónde inicialmente CW será igual a CWmin y donde por cada intento de transmisión fallido (habiendo un número máximo de intentos por trama), el valor CW se iría incrementando en 2CW hasta llegar en este caso a CWmax, una vez esperado este tiempo, la estación volverá a escuchar el medio y si se cumple la condición, entonces transmitirá la trama.

En el caso de que otra estación ocupara el medio durante el tiempo de back-off, la estación almacenaría el tiempo que le queda por esperar y cuando haya transcurrido el IFS de nuevo, sin estar el medio ocupado, el tiempo de espera se contará a partir del valor guardado anteriormente, tal como vemos en la siguiente figura (véase figura 5.4):

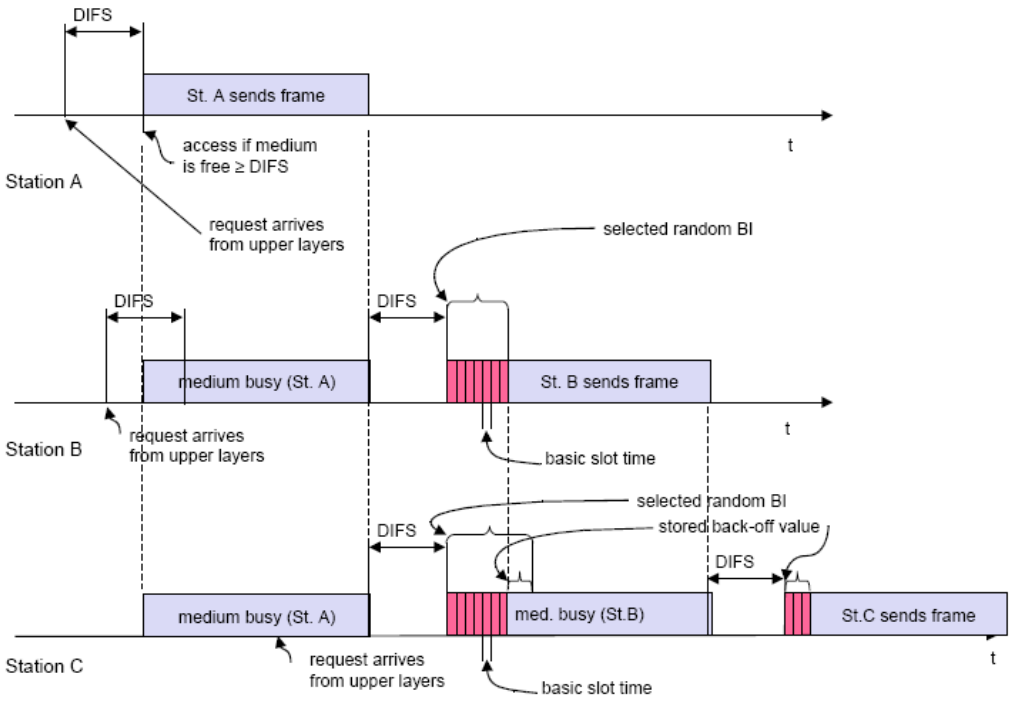

Figura 5.4 Tres estaciones accediendo al medio con CSMA/CA

Comentar que existen tres tamaños de IFS:

- DIFS (DCF Inter-Frame Space), (3 ranuras): para tramas con datos de usuario.
- PIFS (PCF Inter Frame Space) (2 ranuras): para tramas propias de IEEE 802.11 urgentes.
- SIFS Short IFS (1 ranura): para las tramas propias de IEEE 802.11 más urgentes.

Además de lo explicado anteriormente, si las tramas son unicast, la transmisión será comprobada por el receptor mediante un control de redundancia cíclica o CRC, si ésta es exitosa, el receptor enviará una confirmación (ACK) inmediatamente después de un SIFS (Short IFS); si el emisor no recibiera dicha confirmación éste se comportaría de la misma forma como si hubiera habido una colisión (véase figura 5.5).

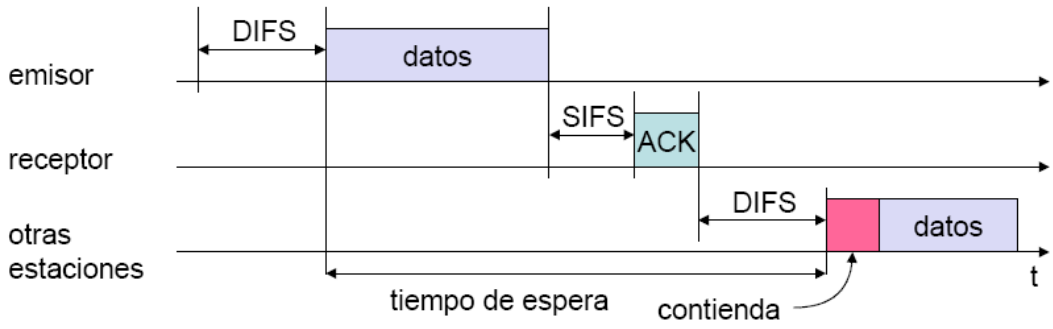

Figura 5.5 Comportamiento de estaciones enviando tramas unicast mediante CSMA/CA

Existe un mecanismo opcional en CSMA/CA solamente para tramas unicast denominado RTS/CTS (Request to Send / Clear to Send) (activado por defecto en NS2) que intenta combatir el problema del terminal escondido. Éste ocurre en el momento en que dos nodos A y C quieren enviar datos al mismo destino y en el mismo instante, es decir al nodo B, y este nodo se encuentra en el alcance radio de A y de C, pero A y C no se ven entre ellos ya que no se encuentran en sus respectivos alcances, en esta situación tanto A como C verán el medio libre, transmitirán la trama y se producirá colisión que tanto A como C no detectarán, es decir, diremos que A está escondido respecto de C y a la inversa (véase figura 5.6) [16]

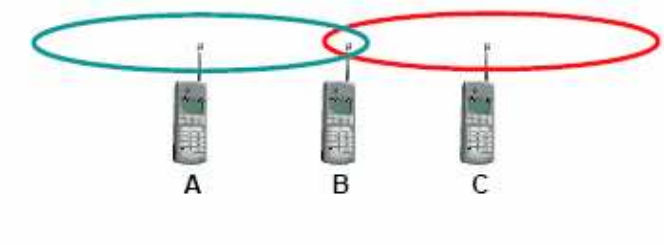

Figura 5.6 Problema del terminal escondido

El mecanismo RTS/CTS funciona entonces de la siguiente manera; la estación que quiere enviar la trama envía antes un RTS que contendrá las identidades del emisor y receptor y la duración de la trama de datos que se quiere enviar; una vez llegado al receptor, éste lo confirmará con un CTS después de un SIFS (en el caso de poder recibir datos). Llegados a este punto el emisor ya puede enviar datos mientras las otras estaciones guardan los parámetros de reserva escuchados en el RTS y el CTS mediante el NAV (Net Allocation Vector); así el resto de nodos no podrán transmitir durante el tiempo indicado por CTS y RTS y evitaremos las colisiones (véase figura 5.7) [17].

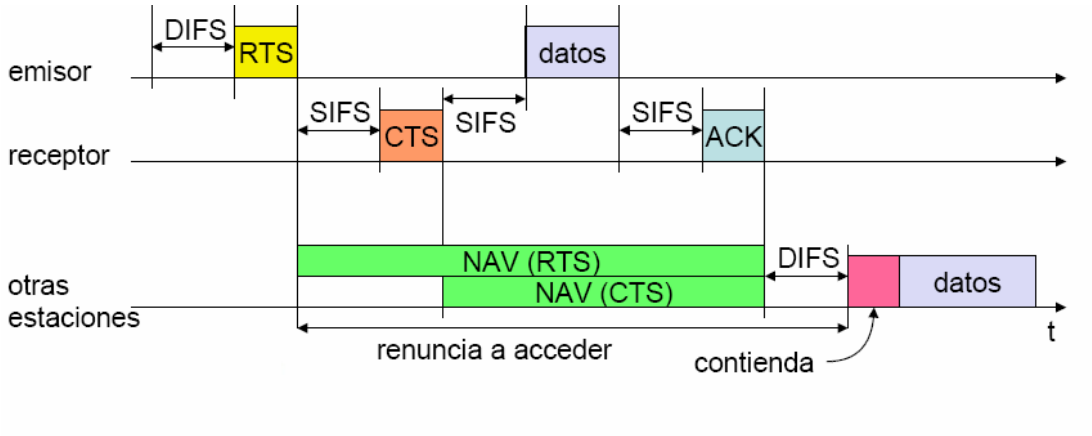

Figura 5.7 Mecanismo RTS/CTS en CSMA/CA

Teniendo en cuenta entonces el mecanismo RTS/CTS, el procedimiento esquemático de CSMA/CA quedaría de la siguiente forma (véase figura 5.8) [18]:

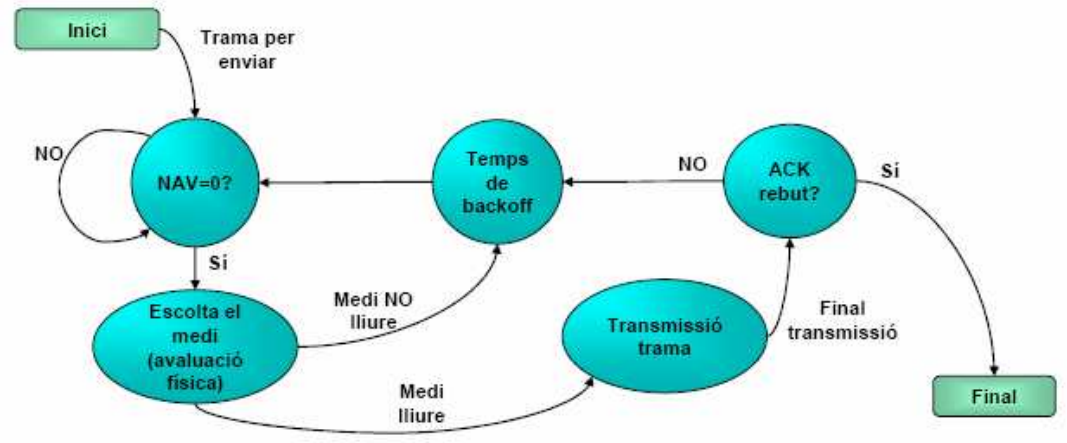

Figura 5.8 Funcionamiento esquemático de CSMA/CA con RTS/CTS

5.1.3. IEEE 802.11b: DSSS

En este apartado vamos a hacer mención de la tecnología que se utiliza en 802.11 b para modular la señal transmitida e intentar protegerla de factores como el ruido o el fading (desvanecimiento de la señal con la frecuencia); para ello se utiliza Direct Sequence Spread Spectrum (DSSS).

La idea de DSSS es la de multiplicar la señal a transmitir por una secuencia pseudo-aleatoria, en este caso denominada chips, mediante una operación XOR; de esta manera aumentamos el ancho de banda de la señal resultante (se dice por ejemplo que hemos aumentado X chips por bit, siendo X por ejemplo 128) (véase figura 5.9). Posteriormente será demodulada por el receptor (véase figura 5.10).

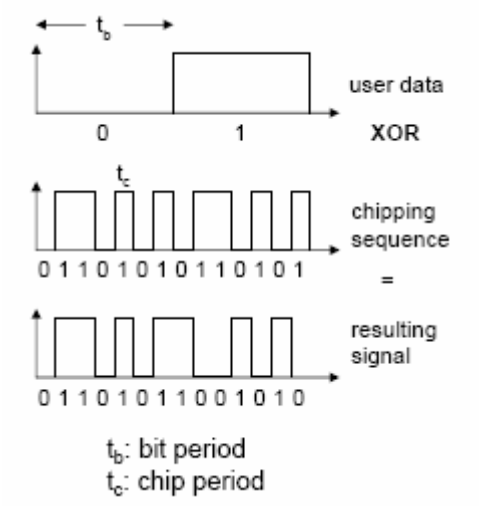

Figura 5.9 Operación XOR de la señal original en DSSS

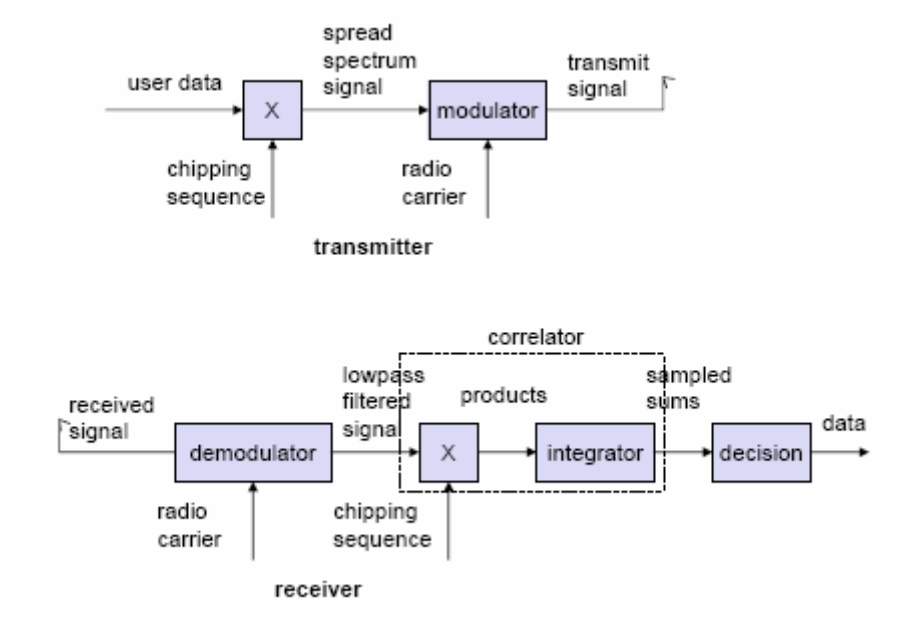

Figura 5.10 Esquema de modulación/demodulación mediante DSSS

Del resultado de estas operaciones tenemos la gran ventaja de proteger nuestra señal transmitida frente a interferencias, posibilitando al receptor discriminar perfectamente entre señal y ruido y permitiendo a éste eliminar dicho ruido con la aplicación de un filtro, por ejemplo, si fuera necesario (véase figura 5.11).

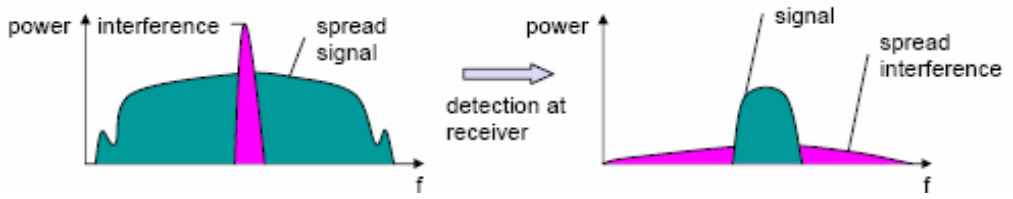

Figura 5.11 Efectos en la señal al aplicar DSSS

5.2. DYMO

En este apartado vamos a describir la base funcional de DYMO; como este protocolo actualmente se encuentra en desarrollo, nos hemos basado en el ultimo draft escrito por sus autores, dónde se encuentra el funcionamiento de DYMO con todo detalle [19]; comentar que explicaremos el funcionamiento de DYMO mediante el complemento de una explicación general del mismo al inicio del apartado, para finalmente hacerlo de una forma mas detallada. Tal como dijimos al inicio del capítulo, en el anexo 1 podremos ver las diferencias entre DYMO y AODV.

5.2.1. Funcionamiento general

Dynamic MANET On-demand (DYMO) es un protocolo de encaminamiento para redes ad-hoc que trabaja en forma reactiva, es decir, que las rutas solamente se construyen en el momento que se necesita (cuando dos o mas nodos quieren comunicarse).

Describiendo de forma generalizada el funcionamiento de DYMO, las dos operaciones básicas que describen su funcionamiento son el descubrimiento y mantenimiento de rutas.

Si un nodo quiere comunicarse con otro nodo y posee la ruta aún vigente para llegar a éste la utiliza sin más (temporizadores, véase apartado 5.2.2.1); en el caso de no conocerla o de que dicha comunicación sea interrumpida por la caída de un enlace, este nodo procederá a realizar un descubrimiento de ruta mediante un mensaje de petición denominado RREQ (Route Request) (véase apartado 5.2.3.1) difundido mediante “broadcast controlado” (véase apartado 5.2.3.1) donde el nodo fuente incluye su dirección y un número de secuencia; el paquete RREQ se extiende salto a salto por cada nodo que participa en la red en ese momento. Cuando un nodo recibe el RREQ obtiene la información del nodo generador del mismo y seguidamente compara la información contenida en el paquete con la entrada correspondiente en su tabla de encaminamiento:

- Si la entrada no existiera ésta se crearía poniendo como next hop (próximo salto) el nodo a través del cual hemos recibido este RREQ, teniendo en cuenta también la interfície por la que lo hemos recibido.
- Si la entrada sí existiera, se examinaría el número de secuencia del RREQ para comprobar la vigencia del mismo (se comprueba si el número de secuencia del RREQ es mayor o igual que el número de

- secuencia correspondiente a una posible ruta hacia ese destino que estuviera almacenada en la tabla de encaminamiento; esto indica que el RREQ tiene una ruta más nueva (si el número de secuencia es mayor); si los números de secuencia fueran iguales se debería actualizar la tabla de encaminamiento con aquella ruta que tuviera un menor número de saltos para alcanzar el destino).
- Una vez comprobada la vigencia del RREQ, si ésta fuera válida, la entrada correspondiente en la tabla de encaminamiento se actualizaría con la información del RREQ. Además de registrar la ruta hacia el nodo originario del RREQ y añadir su propia dirección y numero de secuencia al paquete RREQ.
- Por otro lado, si la información del RREQ estuviera caducada (tiene un número de secuencia menor al que figura en la tabla de encaminamiento en la entrada de la ruta para llegar a ese destino), el RREQ sería descartado y eliminado posteriormente por el resto de nodos [20].

Una vez el nodo destino recibe el RREQ, contesta con un RREP (Route Reply), que es enviado de forma unicast, de nuevo, salto a salto hacia el nodo origen gracias a la información que ha ido agregando cada nodo intermedio, ya que conoceremos el número de saltos intermedios y su correspondiente ruta, tal como hemos comentado anteriormente; es entonces cuando cada nodo intermedio vuelve a registrar su ruta, pero esta vez, hacia el nodo destino de la comunicación. En el momento en el que el nodo origen recibe el RREP, la ruta entre ambos a quedado establecida tanto en una dirección como en la otra (véase figura 5.12).

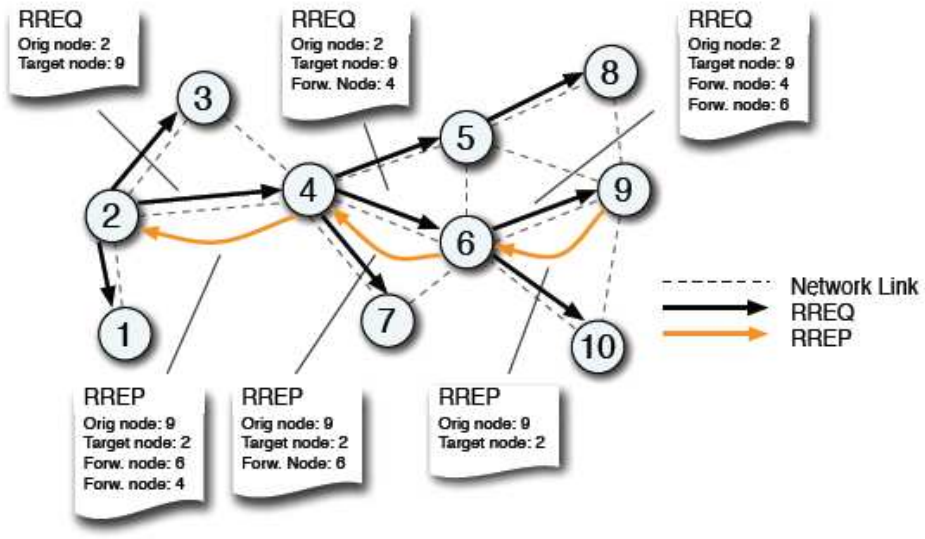

Figura 5.12 Descubrimiento de rutas en el protocolo DYMO

En el mantenimiento de ruta DYMO no se basa en los cambios topológicos en sí que puede sufrir a lo largo del tiempo la red formada, sino que los nodos mantienen y monitorizan sus rutas solamente cuando existe tráfico en ellas, es decir, se controlan los cambios topológicos de las rutas activas, por ejemplo, en

el caso de que un nodo vecino se mueva y deje de estar fuera del alcance radio de algún otro nodo que lo utilizaba para enviar paquetes a lo largo de una ruta. Un nodo debe de comprobar cuál es el estado de los enlaces que mantiene con sus vecinos para enviar tráfico a través de ciertas rutas. Para realizar esta comprobación cuenta con los siguientes mecanismos:

- Dejando la responsabilidad a la capa de enlace.
- Mediante un mecanismo de descubrimiento de vecinos (mensajes periódicos de estado del enlace).
- Por un temporizador de ruta que expira.
- O por otro mecanismo de monitorización (los autores de DYMO dejan este campo de momento abierto).

Los nodos continuamente monitorizan los enlaces activos y actualizan el campo valid Timeout (temporizador válido) de las entradas en su tabla de encaminamiento cuando envían y reciben paquetes con el fin de mantener aquellos caminos que son válidos. En cualquier caso, cuando se detecta que un enlace se ha roto, el valor de ROUTE_DELETE pasará a ROUTE_DELETE_TIMEOUT (temporizador fuera de tiempo, que ha expirado) (véase apartado 5.2.2.1).

El nodo que no ha sabido llegar al destino inicia entonces la difusión del mensaje RERR (Route Error) mediante un “broadcast controlado” (véase apartado 5.2.3.2). Cuando el nodo origen recibe esta notificación sabe entonces que debe iniciar de nuevo un descubrimiento de ruta para llegar al nodo deseado (véase figura 5.13). Para diferenciar por tanto los mensajes de descubrimiento de rutas del mismo origen en momentos diferentes o para que un nodo no reenvíe el mismo mensaje que le ha llegado por dos caminos distintos se utilizan números de secuencia que evitarán también la formación de bucles en nuestra red.

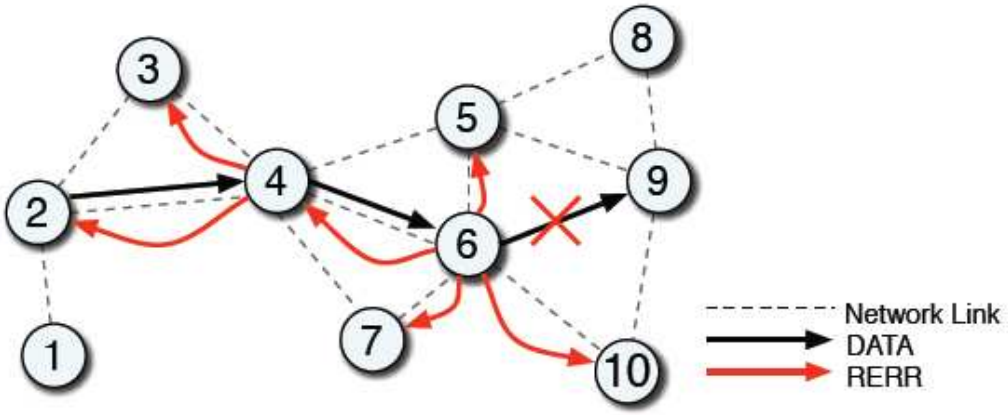

Figura 5.13 Generación y diseminación de los mensajes RERR en DYMO

Una vez entendido el modelo básico funcional de DYMO vamos a profundizar tanto en el formato de sus estructuras de datos como en su funcionamiento.

5.2.2. Estructuras de datos

El protocolo DYMO funciona sobre el protocolo de transporte UDP (User Datagram Protocol) y puede usarse con IPv4 o IPv6; en nuestro caso contemplaremos la opción de IPv4 por ser la que se utiliza en la actualidad.

5.2.2.1. Tabla de encaminamiento

Para empezar vamos a describir el formato de la tabla de encaminamiento que poseen los nodos; los campos que la conforman son los siguientes:

- Route.Address: La dirección IP del nodo destino asociada a la ruta especificada en la tabla.
- Route.SeqNum: El número de secuencia que utiliza DYMO para asociarlo a la información de esta ruta.
- Route.NextHopAddress: La dirección IP del próximo nodo en esta ruta.
- Route.NextHopInterface: La interfície utilizada para enviar los paquetes al nodo destino.
- Route.Broken: Si este flag está activado indica que la ruta está rota.
- Route.HopCnt (opcional): Indica el número de nodos intermedios atravesados antes de llegar al nodo destino.
- Route.Prefix (opcional): Indica el valor de la máscara de red en bits.

Los timeouts asociados son:

- ROUTE_AGE_MIN: Temporizador que da validez a la entrada de una ruta en la tabla; si éste expira, la entrada puede ser borrada con seguridad.
- ROUTE_AGE_MAX: Un temporizador mayor que el anterior, que existe por la posibilidad de pérdida de un número de secuencia durante la comunicación por parte del nodo; si esto ocurre y expira el temporizador, la entrada se borrará.
- ROUTE_NEW: Este temporizador limita el uso de una ruta que acaba de ser descubierta; si ésta no es usada durante este periodo de tiempo, al temporizador ROUTE_DELETE le es establecido el valor ROUTE_DELETE_TIMEOUT.
- ROUTE_USED: Si una ruta no ha vuelto a ser utilizada durante este temporizador, cuando expira, al temporizador ROUTE_DELETE le es establecido el valor ROUTE_DELETE_TIMEOUT.
- ROUTE_DELETE: Temporizador que da vigencia a la información de routing utilizada en ese momento, si éste expira (ROUTE_DELETE_TIMEOUT), la entrada en la tabla asociada debe ser borrada.

5.2.2.2. Routing Messages (RM) - RREQ & RREP

Como hemos comentado en el apartado 5.2.1 los mensajes RM (Routing Messages, mensajes de encaminamiento) son utilizados para el descubrimiento y mantenimiento de ruta; los campos que los conforman son los siguientes (véase figura 5.14):

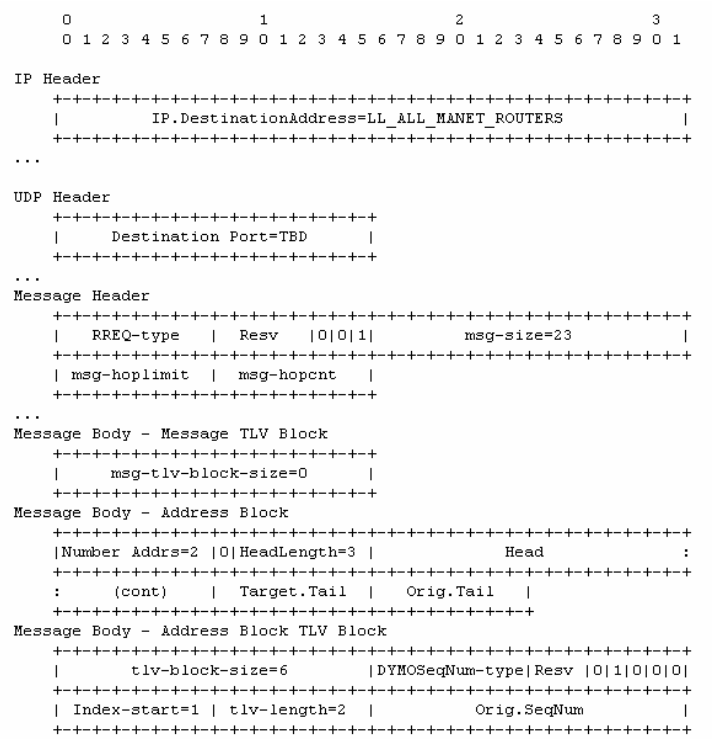

Figura 5.14 Visión general de un mensaje RREQ en DYMO

Los campos más relevantes los explicamos a continuación:

- IP.DestinationAddress: Dirección IP del nodo destino. Para RREQ ésta será establecida con el valor LL_ALL_MANET_ROUTERS, es decir broadcast. Para RREP tendrá el valor de la dirección IP del próximo nodo utilizado para llegar al nodo origen de la comunicación.
- UDP.DestinationPort: Puerto UDP destino.
- MsgHdr.HopLimit: Determina el número de nodos restante que se permite atravesar a este mensaje.
- AddBlk.TargetNode.Address: La dirección IP del TargetNode (denominada Target Tail en la figura 5.14); para un RREQ será el nodo para el cual una ruta determinada no existe mientras se está realizando el descubrimiento (es decir el nodo hacia el cual queremos descubrir la ruta), y para un RREP es el nodo origen de la comunicación. La dirección del TargetNode es la primera dirección en el mensaje RM (Routing Message, mensaje de encaminamiento) (véase figura 5.12).

- AddBlk.OrigNode.Address: La dirección IP del nodo origen de la comunicación. Es la segunda dirección en el mensaje RREQ (denominada Orig.Tail en la figura 5.14).
- OrigNode.AddTLV.SeqNum: El número de secuencia que utiliza el nodo origen de la comunicación.
- TargetNode.AddTLV.SeqNum (opcional): El último número de secuencia conocido del TargetNode.
- TargetNode.AddTLV.HopCnt (opcional): La dirección IP del último nodo atravesado conocido por el TargetNode.
- AddBlk.AdditionalNode.Address (opcional): La dirección IP de un nodo adicional al que se puede acceder a través de este nodo.
- AdditionalNode.AddTLV.SeqNum (opcional): El número de secuencia perteneciente a la información de un nodo adicional intermedio (AdditionalNode).
- Node.AddTLV.HopCnt (opcional): El número de nodos atravesados para llegar al nodo asociado. Este campo es incrementado por cada nodo intermedio.
- Node.AddTLV.Prefix (opcional): La máscara de red de la dirección IP del nodo, cuando ésta posee una longitud particular respecto a las más comunes.

5.2.2.3. Route Error (RERR)

Como hemos comentado en el apartado 5.2.1 RERR sirve para notificar al nodo origen que la ruta no puede alcanzarse; los campos que conforman RERR son los siguientes (véase figura 5.15):

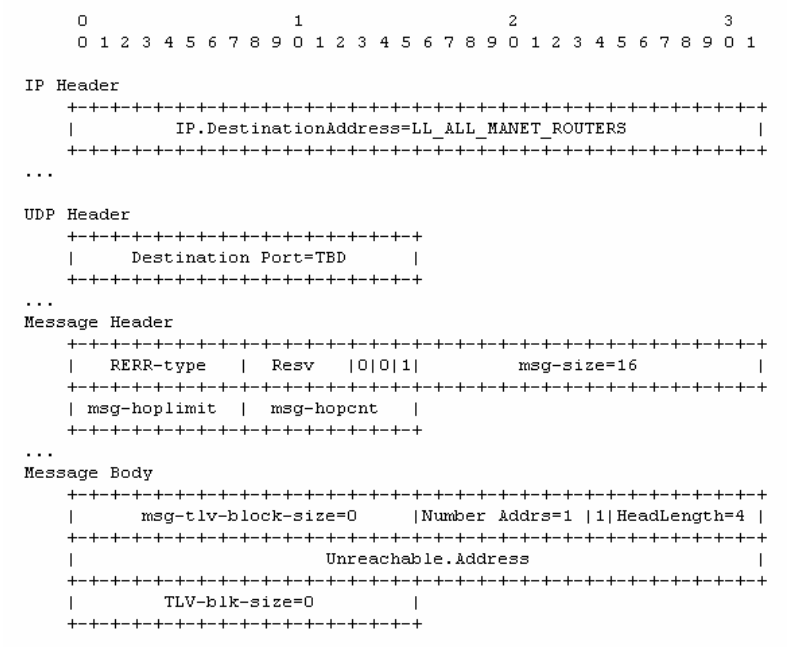

Figura 5.15 Visión general de un mensaje RERR en DYMO

Los campos más relevantes los explicamos a continuación:

- AddBlk.UnreachableNode.Address: La dirección IP del nodo no accesible, puede haber mas de una.

- UnreachableNode.AddTLV.SeqNum (opcional): El último número de secuencia conocido del nodo no accesible.

5.2.3. Funcionamiento detallado

Ahora vamos a ver con más detalle los procesos fundamentales de DYMO, como son el descubrimiento y mantenimiento de ruta (para una mayor comprensión mirar antes el apartado 5.2.2)

5.2.3.1. Descubrimiento de ruta

Cuando se crea un mensaje RREQ, en primer lugar se especifica la AddBlk.TargetNode.Address del nodo al que queremos llegar (dirección IP).

Posteriormente se introducirá el TargetNode.SeqNum (número de secuencia del destino) y el TargetNode.HopCnt (número de saltos intermedios), si éstos son conocidos por el nodo (entrada en la tabla de enrutamiento) se incluirán sin más; en caso contrario no serán incluidos y por tanto se asumirá que no son conocidos.

Entonces el nodo añade su AddBlk.OrigNode.Address (su dirección IP) y su OwnSeqNum (su número de secuencia), el cual es incrementado en una unidad con tal de diferenciarlo de otras posibles peticiones anteriores existentes en la red. Estos campos serán los utilizados por los nodos intermedios para crear una ruta hacia el nodo origen y poderle después hacer llegar el RREP.

También se establecerá un MsgHdr.HopLimit a un tamaño adecuado (según RFC3561) para que pueda llegar a todos los nodos de la red formada. El nodo origen inicialmente propaga el RREQ un determinado número de saltos (TTL) más allá; si después de esperar durante un cierto periodo de tiempo no ha recibido ningún RREP, enviaría un nuevo RREQ en modo broadcast aumentando el número de saltos donde este RREQ se propaga; si de nuevo no recibiera en un periodo de tiempo determinado ningún RREP repetiría el proceso explicado hasta llegar a permitir que el RREQ se propagara un determinado número de saltos TTL máximo (que suele coincidir con el diámetro de la red).

Finalmente al campo IP.DestinationAddress se le da el valor de LL_ALL_MANET_ROUTERS; En este caso el paquete llega a todos los nodos de la red, pero gracias a los números de secuencia un nodo puede discriminar el mismo RREQ que le ha llegado por dos caminos diferentes y no volverlo a transmitir puesto que ya lo ha hecho anteriormente. Por esta razón decimos que propagamos el RREQ en modo broadcast controlado,

Si el nodo origen no encuentra la ruta deseada irá haciendo intentos de descubrimiento mediante un tiempo de backoff RREQ_WAIT_TIME (2 seg. por defecto) que irá creciendo en un factor 2 exponencial en RREQ_TRIES intentos (3 por defecto). Los datos que se quieran enviar al nodo destino se almacenarán en BUFFER_SIZE_PACKETS o BUFFER_SIZE_BYTES (50x1500 Bytes por defecto). En caso de intento fallido los datos del buffer se borrarán y se considerará que el destino es inalcanzable.

En caso contrario, cuando el RREQ es recibido por el nodo destino (comprueba el campo AddBlk.TargetNode.Address de dicho mensaje que es el que tiene que coincidir con su dirección IP), verifica la vigencia de la petición de ruta mediante el número de secuencia, para finalmente enviar el mensaje RREP a la dirección IP origen de la petición (AddBlk.OrigNode.Address) en este caso en unicast, gracias a la información que han ido almacenando los nodos intermedios, teniendo como IP.DestinationAddress, la dirección IP del próximo nodo (next-hop) para llegar al nodo origen, e incluyendo también su número de secuencia incrementado en uno para evitar confusiones con información obsoleta (véase figura 5.12).

5.2.3.2. Mantenimiento de ruta

Como hemos comentado en el apartado 5.2.1, cuando existe tráfico y un nodo no puede acceder hacia un destino, genera un mensaje RERR con IP.DestinationAddress LL_ALL_MANET_ROUTERS y poniendo la dirección IP del nodo/s no accesible/s por la caída del enlace en AddBlk.UnreachableNode.Address y si es conocido su número de secuencia lo pondremos en UnreachableNode.AddTLV.SeqNum y el campo MsgHdr.HopLimit se establecerá de la misma forma que en el descubrimiento de ruta.

El mensaje RERR es enviado a todos los LL_ALL_MANET_ROUTERS que tengan dependencia del enlace caído. Una vez llega el RERR a los nodos dependientes, éstos verifican la vigencia del mensaje mediante el número de secuencia y borran todas las rutas que pasen por ese enlace y activan el flag de Route.Broken para aquellos nodos que no sean accesibles.

Finalmente cuando el mensaje RERR llegue al nodo que originó la comunicación sabrá que tendrá que volver a realizar un descubrimiento de ruta para encontrar de nuevo al nodo (véase figura 5.13).

CAPÍTULO 6. EMPEZANDO A SIMULAR

Seguidamente presentaremos los diferentes casos, escenarios, parámetros y ambientalización que adoptaremos a la hora de realizar nuestras simulaciones; en el anexo B de este TFC también encontraremos la explicación detallada sobre el procedimiento que hemos seguido para realizar éstas; cabe comentar que no existe una configuración referente a los anteriores de forma correcta o equivocada, sino que existen diferentes soluciones y posibilidades válidas dependientes de los rasgos a evaluar; así pues, los casos planteados y los valores que hemos escogido son los siguientes:

6.1. Casos a evaluar

En los casos siguientes lo que evaluaremos será el goodput por fuente, es decir, el tráfico que llega con éxito a los receptores por fuente existente. El goodput podría definirse de la forma siguiente:

$$\text{Goodput}=\frac{\textit{Máximo número de paquetes recibidos}\times\textit{Tamaño del paquete}}{\textit{Intervalo de medida}} \qquad \textbf{(6.1)}$$

En nuestro caso para realizar el cálculo de goodput por fuente lo hemos realizado de la siguiente manera:

Goodput por fuente= [(Nº de paquetes recibidos x Tamaño del paquete/ nº de fuentes) / Intervalo de medida] **(6.2)**

Los distintos casos a evaluar serán los siguientes:

- Caso 1: Tráfico TCP/UDP en función de la velocidad, el número de vehículos y el % de fuentes en una autopista.
    - Caso 1.1 Tráfico TCP dónde el 20% son fuentes
    - Caso 1.2 Tráfico TCP dónde el 80% son fuentes
    - Caso 1.3 Tráfico UDP dónde el 20% son fuentes
    - Caso 1.4 Tráfico UDP dónde el 80% son fuentes
- Caso 2: Tráfico TCP/UDP y el impacto de un accidente sobre la comunicación en una autopista en función del % de fuentes.
    - Caso 2.1 Tráfico TCP
    - Caso 2.2 Tráfico UDP
- Caso 3: Tráfico TCP/UDP y el impacto de un entorno urbano (Eixample de Barcelona) frente al caso de la autopista en función del % de fuentes.
    - Caso 3.1 Tráfico TCP

  - Caso 3.2 Tráfico UDP

- Caso 4: Tráfico TCP/UDP y el impacto de un accidente sobre la comunicación en un entorno urbano en función del % de fuentes.

  - Caso 4.1 Tráfico TCP
  - Caso 4.2 Tráfico UDP

6.2. Parámetros escogidos

- Ruido: Utilización de pseudoruido aleatorio (para simular ruido ambiente).

- Tiempo durante el cual los vehículos están en movimiento: 1050 segundos.

- Intervalo de tiempo en el cual se envía tráfico: se inicia el envío de tráfico en el segundo 50 de la simulación y finaliza en el segundo 1040. (se envía tráfico durante 990 segundos).

- Número de nodos (vehículos): de 20 a 200 (incrementos de 20 en 20) para el caso 1, y 20 vehículos para los casos 2, 3 y 4.

- Velocidad de nodos (vehículos): de 60 a 140 Km/h (en incrementos de 20 en 20) para el caso 1, y de 100 km/h para el caso 2.
  Para el caso 3 100 km/h para la autopista, y velocidades entre 10 y 50 km/h (velocidades repartidas entre los vehículos del escenario sufriendo aceleraciones y desaceleraciones) para el entorno urbano del mismo caso 3 y para el entorno urbano descrito en el caso 4.

- Número de nodos que actúan como fuentes: 20% y 80 % para el caso 1, de 20% a 80% (en incrementos de 10) para los casos 2, 3 y 4.

- Tipo de tráfico que envían las fuentes: TCP // CBR (Constant Bit Rate o tasa constante) UDP.

- Tipo de TCP: TCP Reno.

- Paquetes enviados: Datagramas UDP de 512 bytes a una tasa de envío constante de 32 Kbps y con un número máximo de paquetes enviados de 150.000 paquetes // Paquetes TCP de 512 bytes, con un tamaño de ventana de 32 paquetes.

- Tipo de LLC: LL (estándar implementado en NS2).

- Nº Máximo de paquetes en cola o buffer (IFQ): 50 (valor por defecto de NS2).

- Tipo de cola o buffer: Queue/DropTail/PriQueue, se da prioridad a los paquetes de encaminamiento, de forma que cuando llega un paquete de encaminamiento se coloca en la cabeza de la cola por delante de los paquetes de datos. Si se supera la capacidad del buffer porque llega un paquete de datos, este paquete se pierde; si se supera la capacidad del buffer porque llega un paquete de encaminamiento, entonces se pierde el último paquete de datos de la cola para dejar espacio al de encaminamiento.

- Protocolo MAC: 802.11b

- Ancho de Banda*:* 2 Mbps. La máxima velocidad que permite el estándar 802.11 b (ya que velocidades de 11 Mbps se dan solamente en un ambiente libre de interferencias y a muy corta distancia).

- Protocolo de encaminamiento (routing): DYMO (implementación denominada DYMOUM en NS2). Comentar que para adaptar la implementación DYMOUM a la última especificación de DYMO añadiremos al script TCL de las simulaciones la línea (Agent/DYMOUM set reissue_rreq_ true) para que el nodo origen de la comunicación haga reintentos de petición en el caso de no encontrar al nodo deseado siguiendo el procedimiento explicado en el apartado 5.2.3.1 (Descubrimiento de ruta) del capítulo 5.

- Modelo de propagación: Shadowing Model (modelo realístico de propagación radio implementado en NS2) ya que mediante diferentes pruebas se ha demostrado que este modelo da resultados más próximos a la realidad en simulaciones VANET respecto al resto [22] // exponente de perdida = 2,56 y desviación estándar de 4 dB (valores experimentales pertenecientes a medidas reales realizadas en comunicaciones vehicle-to-vehicle según los autores de MOVE [14]).

- Tipo de antena: Omnidireccional

- Rango de transmisión: 250 m para todos los casos (valor por defecto de NS2).

- Distancia entre vehículos: La distancia entre vehículos dependerá de la velocidad de los vehículos, para mantener la distancia de seguridad adecuada; dicha distancia es adoptada en todo momento por el simulador SUMO mediante un algoritmo desarrollado por Stefan krauss [21] una aproximación del mismo lo podemos encontrar en una fórmula común para calcular distancias de seguridad (en metros) como es:

$$\text{Distancia de seguridad mínima} = V^2/C \qquad \textbf{(6.3)}$$

Dónde V es la velocidad del vehículo en km/h y C es una constante igual a 170. (Es decir que según la velocidad de los vehículos del escenario tendremos unas distancias u otras).

- Longitud de cada vehículo: será configurada a 4 m (longitud media aproximada de un turismo).

- Nº de accidentes: ningún accidente para los casos 1 y 3; 1 accidente para los casos 2 y 4.

- Tipo de accidente: A mitad de la simulación 2 vehículos que actúan como nodos intermedios sufren un accidente y a causa de ello no siguen circulando con el resto; esto provocará caídas de enlaces y establecimiento de nuevas rutas para la comunicación entre los nodos del escenario.

6.3. Escenarios

6.3.1. Autopista

Los vehiculos se mueven en una autopista de 4 carriles en el mismo sentido de la circulación, donde la anchura media de un carril de autopista es de 2,7 metros (según la simulación de SUMO un vehículo ocupa la anchura de un carril sin especificar la anchura del mismo). Para establecer la longitud total de nuestro tramo de autopista hemos escogido un valor que nos sea valido para todas las simulaciones a realizar, para ello contemplaremos el caso extremo en el que tenemos el número mayor de vehículos en el escenario (200) y todos van a la máxima velocidad (140 km/h) y uno detrás de otro, es decir, en fila india, así entonces tenemos:

200 (vehículos) x 4 (metros/vehículo) + 140^2 (velocidad en km/h al cuadrado) / 170 (constante) * 199 (número de distancias de seguridad que generan los 200 vehículos) + 38,9 (140 km/h pasado a m/s) * 1050 (tiempo en segundos durante el cual los vehículos están en movimiento)= 64588 m ≈ 65 Km.

Los vehículos son repartidos entre los 4 carriles existentes por igual, excepto en el caso de 20 vehículos, donde son repartidos entre los 2 carriles centrales, es decir, 10 en cada uno; éstos viajan agrupados y separados entre ellos por la distancia mínima de seguridad; y sus velocidades son constantes para cada caso, ya que en este entorno y para trayectos largos y despejados, las velocidades tienden a ser constantes; nuestra intención es evaluar el goodput por fuente y el posible impacto sobre de la velocidad en éste, entre otros parámetros.

6.3.2. Entorno Urbano: Eixample de Barcelona

Para este caso hemos recreado 49 manzanas del Eixample de Barcelona, habiendo 2 sentidos por calle (dos carriles), ocupando así una extensión total

de 6.531 m² = 133,3 m² (extensión de una manzana) * 49 (manzanas), lo que es equivalente a 6 campos y medio de futbol (véase figura 6.1).

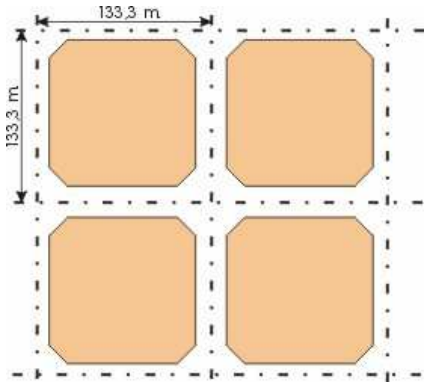

Figura 6.1 Tramo urbano básico del Eixample de Barcelona

Para recrear este escenario hemos cogido las medidas que constituyen las manzanas del Eixample de Barcelona y las hemos convertido en un mapa mediante las herramientas que ofrece SUMO, proceso que comentaremos con mas profundidad en el Anexo B*: Procedimientos para la simulación,* de este TFC.

Los vehículos que van circulando por las manzanas de este entorno lo hacen de una forma dinámica, es decir, adoptarán velocidades diferentes sufriendo también aceleraciones y desaceleraciones diferentes, tal como ya hemos comentado; así cada vehículo tendrá una velocidad diferente y un movimiento diferente, con tal de amoldarnos a este tipo de escenario de una forma real, tal como hemos hecho en el caso de la autopista.

6.4. Ambientalización

En las simulaciones evaluamos el comportamiento de tráfico TCP y UDP; una posible utilización de los mismos podría ser la siguiente:

- Tráfico TCP: Comunicaciones vehicle-to-vehicle en las que se intercambia información del estado del tráfico, climatológica, sobre las condiciones de la vía…etc. Con la finalidad de aumentar el confort y la seguridad del viaje aparte de poder evitar posibles colapsos o caravanas…etc. Podríamos Imaginar que toda esta información iría presentada a los tripulantes del vehículo mediante una pantalla o display central interpretada por una aplicación específica.

- Tráfico UDP: Comunicaciones vehicle-to-vehicle en la que se intercambia tráfico multimedia como por ejemplo clips de video pertenecientes a un tramo de la vía en malas condiciones, un obstáculo a esquivar…etc. con tal de complementar la información adoptada por la aplicación anteriormente comentada.

CAPÍTULO 7. RESULTADOS Y CONCLUSIONES

En este capítulo vamos a presentar y evaluar los resultados que hemos obtenido para después así sacar nuestras propias conclusiones al final del mismo; comentar que para todos los casos hemos realizado la misma gráfica en dos versiones diferentes, la primera compuesta por puntos que describen los valores obtenidos unidos por una línea; y la segunda mediante barras; en el anexo C de este TFC encontraremos los resultados numéricos.

7.1. Resultados

7.1.1. Caso 1: Tráfico TCP/UDP en función de la velocidad, el número de vehículos y el % de fuentes en una autopista

7.1.1.1. Caso 1.1 Tráfico TCP dónde el 20% son fuentes

Los resultados pertenecientes al caso 1.1 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.1):

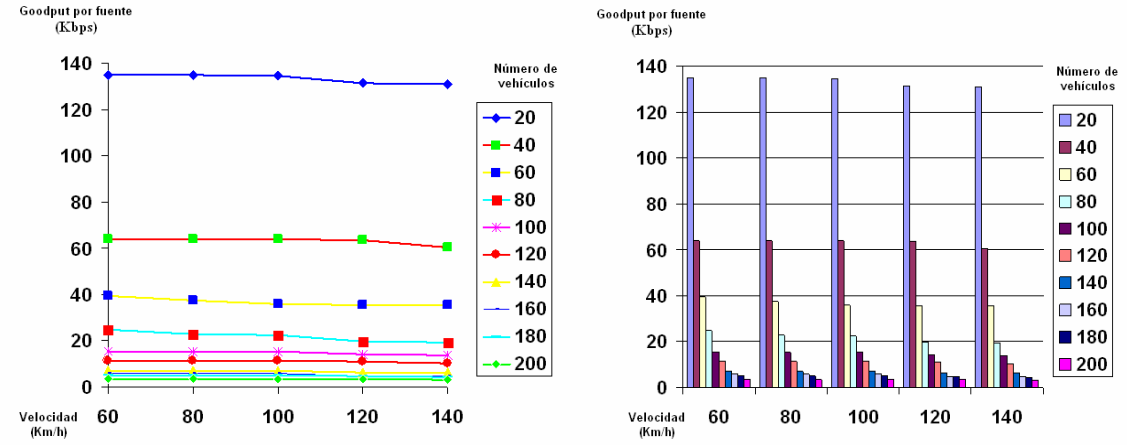

Figura 7.1 Gráficas del caso 1.1

Observando ambas gráficas vemos que en este caso hemos obtenido unos valores de goodput por fuente próximos a 135 Kbps para el mejor de los casos (20 vehículos) y de 3 Kbps para el peor (200 vehículos).

Observamos que el impacto de la velocidad en este caso particular es prácticamente nulo, ya que las líneas se mantienen prácticamente constantes con el aumento de la velocidad. Esto es debido a que todos los vehículos de este escenario van a la misma velocidad y éstas además son constantes; esto da lugar a que SUMO reparta todos los vehículos del escenario a la misma distancia entre ellos, en este caso la distancia de seguridad mínima, que en el caso mas extremo de nuestras simulaciones (cuando los vehículos circulan a 140 km/h) será igual a 115,3 metros (siguiendo la fórmula utilizada para

calcular la distancia de seguridad en función de la velocidad de una forma aproximada V^2/C dónde V^2 es la velocidad en Km/h al cuadrado y C es una constante igual a 170), es decir, que entre un vehículo y otro siempre será posible la comunicación ya que 115,3 metros es inferior al rango de transmisión configurado en estas simulaciones que es igual a 250 m.

Lo que sí impacta negativamente en el goodput por fuente es el número de vehículos que participan en el escenario, dónde su aumento hace decrecer exponencialmente el valor de goodput; ya que al haber un número mayor de nodos enviando paquetes o datagramas (ya sean nodos fuente o nodos intermedios), el medio físico tiene que ser compartido pues por más nodos, y por tanto existirán más colisiones y reintentos a la hora de acceder al medio físico, aparte de la necesidad de encaminar más paquetes o datagramas por los nodos intermedios en las comunicaciones, creación de un número mayor de rutas…etc.

7.1.1.2. Caso 1.2 Tráfico TCP dónde el 80% son fuentes

Los resultados pertenecientes al caso 1.2 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.2):

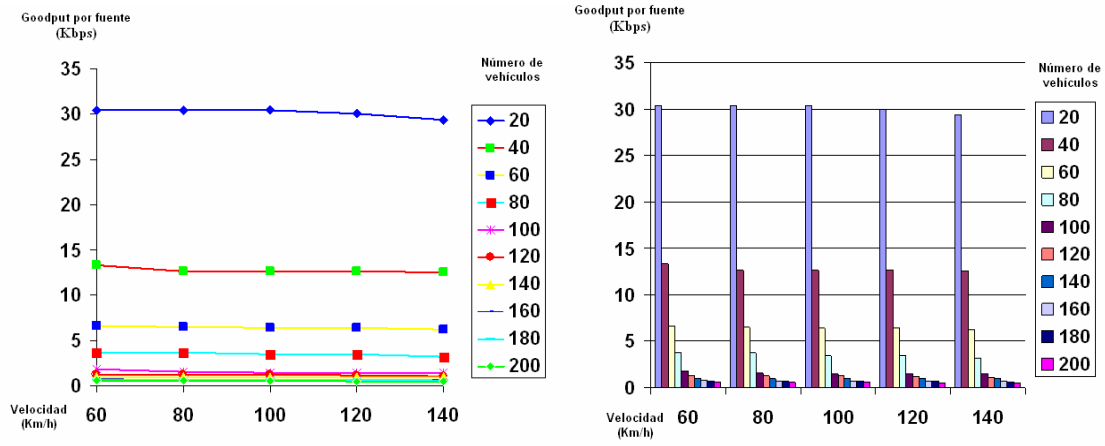

Figura 7.2 Gráficas del caso 1.2

Si observamos las gráficas del caso 1.2 podemos ver los mismos efectos que comentábamos anteriormente. En este caso vemos también que al aumentar el porcentaje de fuentes de un 20 a un 80 % tenemos valores muchos mas bajos de goodput por fuente; al haber muchos mas nodos transmitiendo habrán más colisiones entre paquetes en el medio y como consecuencia disminuirá el goodput, como es de esperar; tal y como comentábamos en el caso 1.1, con un 20 % de fuentes obteníamos valores entre 135 y 3 Kbps, mientras ahora con un 80 % de fuentes obtenemos valores entre 30 y 0,5 Kbps (los valores de goodput de la gráfica anterior han disminuido en torno a un 80%).

7.1.1.3. Caso 1.3 Tráfico UDP dónde el 20% son fuentes

Los resultados pertenecientes al caso 1.3 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.3):

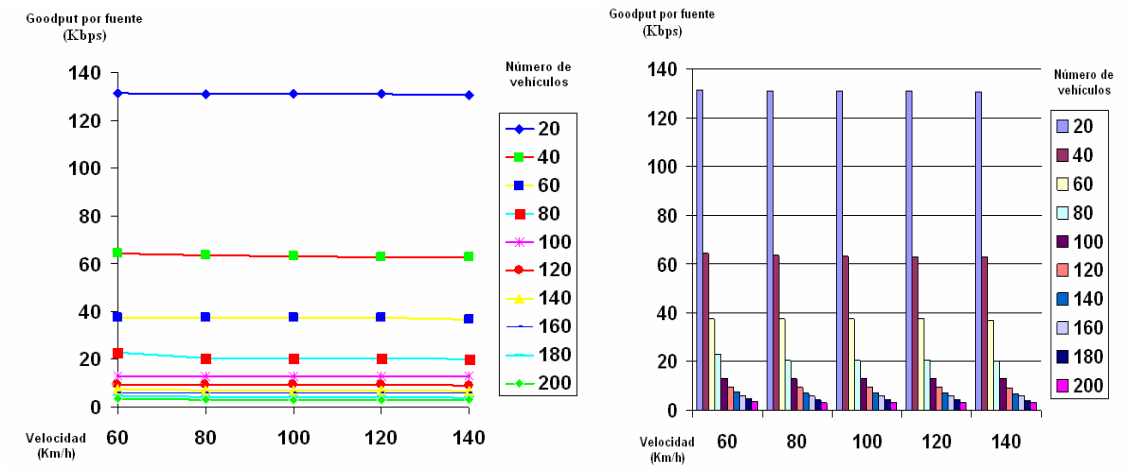

Figura 7.3 Gráficas del caso 1.3

Observando las gráficas del caso 1.3 podemos ver los mismo efectos que los comentados anteriormente en el caso 1.1; si hacemos una comparativa entre TCP y UDP, observamos que tenemos unos valores muy próximos de goodput por fuente entre ambos escenarios, dando unos valores ligeramente inferiores el de UDP, entre 131 y 3 kbps frente los 135 y 3 Kbps de TCP, ya que UDP al ser tráfico a tasa constante y no orientado a conexión (las fuentes envían a dicha tasa ocurra lo que ocurra) provoca que un gran número de datagramas se pierdan a causa del desbordamiento de los buffers que es exactamente lo que nos ocurría en nuestras simulaciones, dando por tanto, una tasa de paquetes recibidos muy similar a TCP, ya que en ambos casos los buffers eran igual a 50 paquetes o datagramas, siendo en ambos casos también igual el tamaño de paquete. Por el otro lado, TCP al ser orientado a conexión posee una tasa de pérdidas menor, haciendo retransmisiones de paquetes si fuera necesario dando en nuestras simulaciones valores más altos de goodput por fuente frente a UDP. Por tanto se deduce que como UDP envía muchos más paquetes congestiona más el medio y consecuentemente los receptores acaban recibiendo menos paquetes que en el caso de TCP, aunque se hubiera enviado inicialmente un número muy superior de paquetes.

7.1.1.4. Caso 1.4 Tráfico UDP dónde el 80% son fuentes

Los resultados pertenecientes al caso 1.4 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.4):

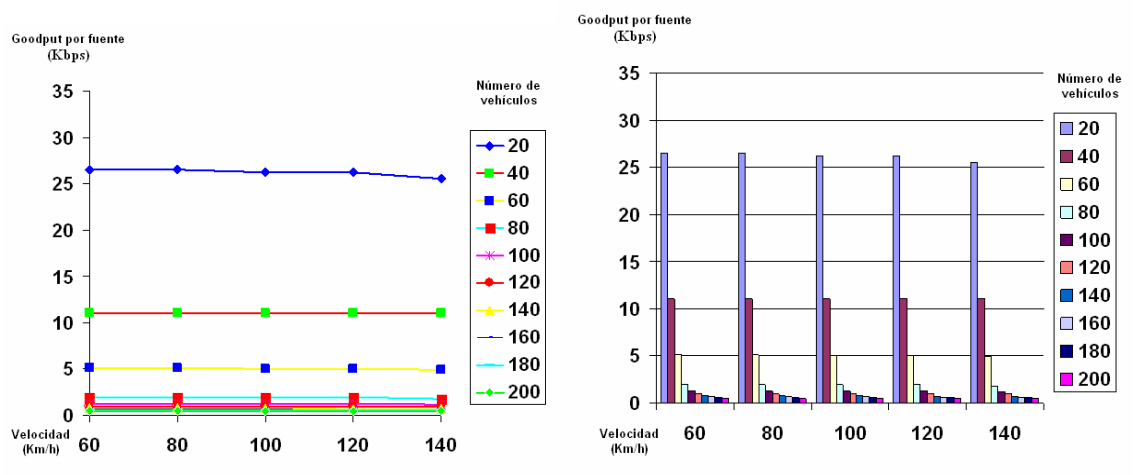

Figura 7.4 Gráficas del caso 1.4

En el caso 1.4 vuelve a quedar de manifiesto el impacto negativo del aumento del tanto por ciento de fuentes en el goodput por fuente, pasando de valores de entre 135 y 3 Kbps con el 20% de fuentes, a 27 y 0,5 Kbps para el de 80%; si volvemos a comparar el caso UDP con el de TCP, siguen manteniéndose los valores aproximados, ligeramente menores los de UDP frente a los de TCP, entre 27 y 0,5 Kbps frente a los 30 y 0,5 Kbps respectivamente; llegados a este punto también vemos que con independencia del tipo de tráfico (TCP o UDP) el goodput no se ve afectado por la velocidad de los vehículos; en cambio sí que disminuye al aumentar el porcentaje de fuentes, tal como hemos ido observando a lo largo de los casos ya presentados.

7.1.2. Caso 2: Tráfico TCP/UDP y el impacto de un accidente sobre la comunicación en una autopista en función del % de fuentes

Para este caso recordamos que el tipo de accidente es el siguiente:

A mitad de la simulación 2 vehículos que actúan como nodos intermedios sufren un accidente y a causa de ello no siguen circulando con el resto, esto provocará caídas de enlaces y establecimiento de nuevas rutas para la comunicación entre los nodos del escenario.

7.1.2.1. Caso 2.1 Tráfico TCP

Los resultados pertenecientes al caso 2.1 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.5):

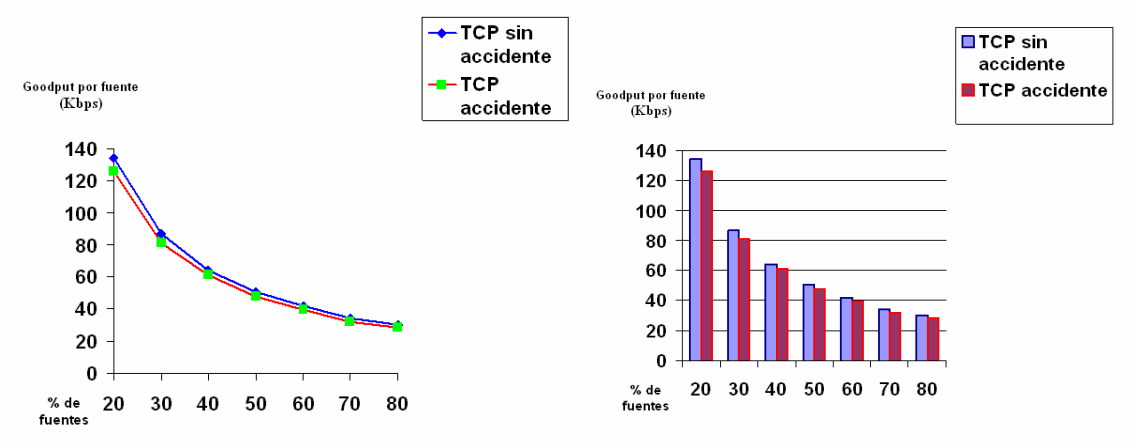

Figura 7.5 Gráficas del caso 2.1

En este caso podemos ver dos aspectos influyentes en la comunicación, en primer lugar vemos que el goodput por fuente va decreciendo de forma claramente exponencial con el aumento progresivo del porcentaje de fuentes en el escenario; en segundo lugar vemos el impacto de un accidente en la comunicación de los nodos, describiendo así una línea por debajo de la anterior, es decir, con valores mas bajos de goodput provocados por el mismo, concretamente entre 134 y 30 Kbps para el caso sin accidente y de 126 y 28 Kbps para el caso con accidente. Así vemos que el impacto de un accidente afecta al goodput, pues al romperse las rutas entre ciertos nodos que han sufrido el accidente se pierden paquetes y será necesario volver a reestablecer nuevas rutas para enviar la información.

7.1.2.2. Caso 2.2 Tráfico UDP

Los resultados pertenecientes al caso 2.2 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.6):

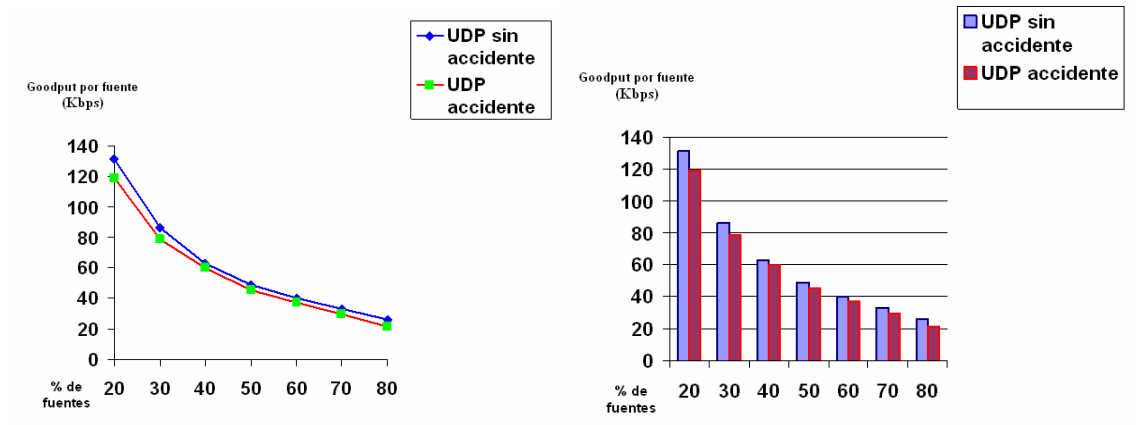

Figura 7.6 Gráficas del caso 2.2

Para el caso 2.2 podemos observar los mismos efectos comentados en el caso 2.1 sobre el impacto del accidente en la comunicación; obtenemos valores de 131 y 26 Kbps para el caso de UDP sin accidente y de 119 y 22 Kbps para el caso de UDP con accidente; si comparamos el caso TCP con el UDP nos continúan saliendo valores muy próximos, siendo los de UDP ligeramente más bajos, a causa de lo que ya hemos comentado en el caso 1.3 (134 y 30 Kbps para TCP sin accidente y de 126 y 28 Kbps para TCP con accidente).

7.1.3. Caso 3: Tráfico TCP/UDP y el impacto de un entorno urbano (Eixample de Barcelona) frente a la autopista en función del % de fuentes

Para este caso recordamos que los vehículos de la autopista viajan a una velocidad de 100 km/h y los del Eixample de Barcelona lo hacen a velocidades de entre 10 y 50 km/h (velocidades repartidas entre los vehículos del escenario sufriendo aceleraciones y desaceleraciones), como ya comentamos en el capítulo 6; como en el caso específico de la autopista visto anteriormente, la velocidad no influía en los valores de goodput por fuente, podemos tener distintos vehículos circulando a diferentes velocidades en el entorno urbano sin que suponga ningún problema de cara a comparar este entorno con el de la autopista.

7.1.3.1. Caso 3.1 Tráfico TCP

Los resultados pertenecientes al caso 3.1 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.7):

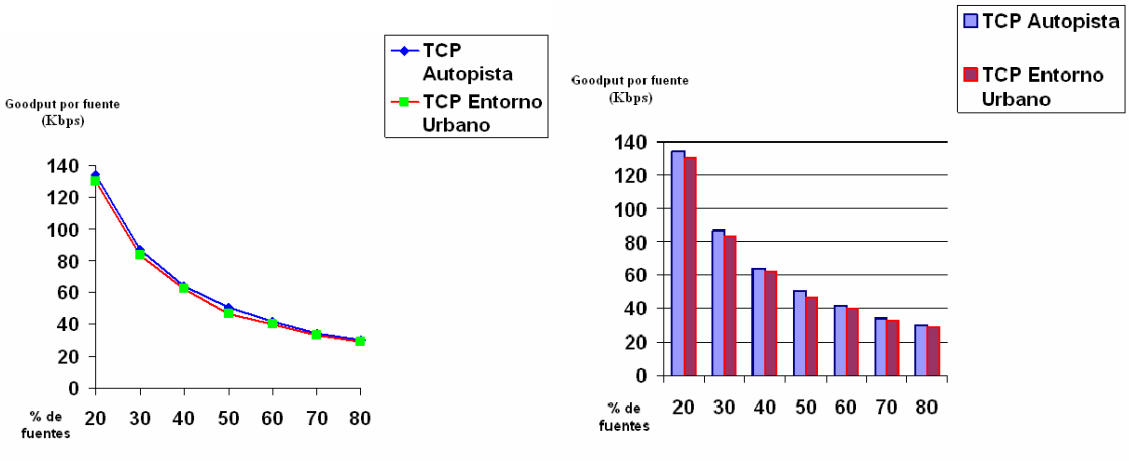

Figura 7.7 Gráficas del caso 3.1

En el caso 3.1 podemos observar el impacto del entorno por donde transitan los vehículos en la comunicación; en nuestros 2 casos particulares obtenemos que en el entorno urbano los valores son ligeramente inferiores a los de la autopista, siendo también ambos bastante próximos, entre 134 y 30 Kbps para

la autopista y de 130 y 29 Kbps de goodput por fuente para el Eixample de Barcelona; esto es debido a que en el entorno urbano las velocidades de los vehículos que circulan son diferentes para cada uno (entre 10 y 50 Km/h) sufriendo aceleraciones y desaceleraciones diferentes también para cada vehículo, mientras que en la autopista todos los vehículos circulan a velocidad constante (a 100 km/h), esto provoca que las distancias en el entorno urbano sean mucho menores que en la autopista, donde en este caso (100 Km/h) siempre serán igual a 58,8 metros, mientras que en el entorno urbano cabe la posibilidad de que varios vehículos se encuentren prácticamente pegados mientras van circulando por el Eixample de Barcelona; esto provoca que hayan mas nodos compitiendo por el mismo medio físico y en consecuencia a la hora de querer comunicarse con otro nodo, como hemos comentado anteriormente, existirán mas colisiones y por tanto reintentos a la hora de acceder al medio, cosa que provoca una disminución del goodput.

7.1.3.2. Caso 3.2 Tráfico UDP

Los resultados pertenecientes al caso 3.2 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.8)

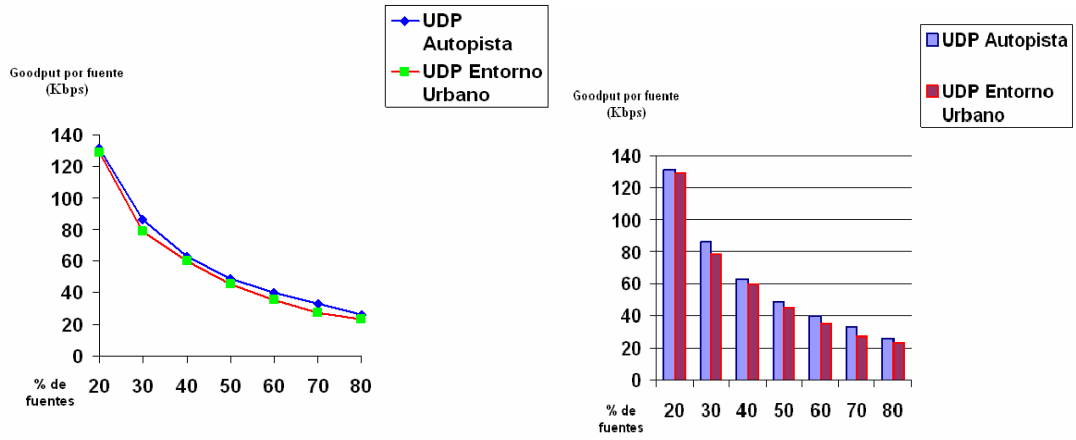

Figura 7.8 Gráficas del caso 3.2

En este caso podemos observar los mismos efectos que en el caso 3.1, obtenemos valores de goodput por fuente entre 131 y 26 Kbps para el caso de la autopista y de 129 y 24 Kbps para el Eixample de Barcelona. En este caso volvemos a tener valores muy próximos entre UDP y TCP, siendo los de UDP un poco menores por las razones que ya hemos explicado (134 y 30 Kbps para TCP en autopista y de 130 y 29 Kbps para TCP en entorno urbano).

.1.4. Caso 4: Tráfico TCP/UDP y el impacto de un accidente sobre la comunicación en un entorno urbano en función del % de fuentes

7.1.4.1. Caso 4.1 Tráfico TCP

Los resultados pertenecientes al caso 4.1 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.9)

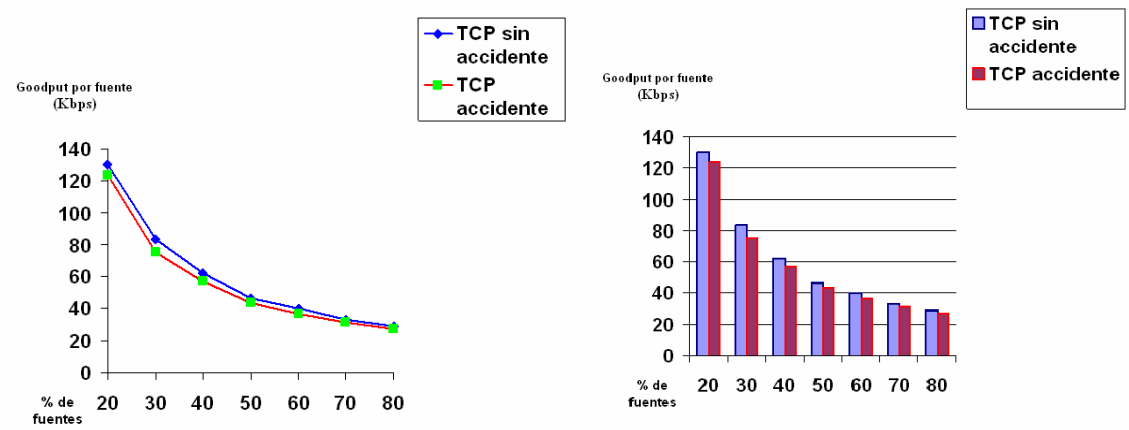

Figura 7.9 Gráficas del caso 4.1

Para este caso vemos de nuevo el impacto de un accidente en el entorno urbano, podemos observar los mismos efectos que en el caso 2, es decir, una disminución del goodput por fuente a causa del mismo pues al romperse las rutas entre ciertos nodos que han sufrido el accidente se pierden paquetes y será necesario volver a reestablecer nuevas rutas para enviar la información, concretamente valores entre 130 y 29 Kbps para el caso de TCP sin accidente, y de 124 y 27 Kbps en el caso de TCP con accidente.

7.1.4.2. Caso 4.2 Tráfico UDP

Los resultados pertenecientes al caso 4.2 los podemos ver en las siguientes gráficas presentadas a continuación (véase figura 7.10)

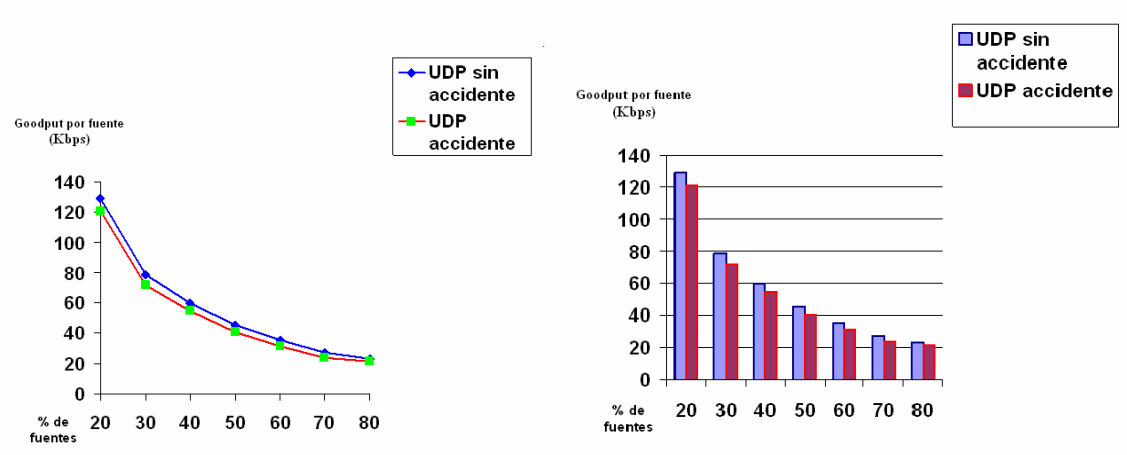

Figura 7.10 Gráficas del caso 4.2

Observamos los mismos efectos que en el caso 4.1, en este caso obtenemos valores de goodput por fuente entre 129 y 24 Kbps para el caso de UDP sin accidente y de 121 y 22 Kbps para el de UDP con accidente; continuamos teniendo valores próximos entre TCP y UDP, siendo inferiores los de UDP, por las razones ya explicadas (130 y 29 Kbps para el caso de TCP sin accidente, y de 124 y 27 Kbps en el caso de TCP con accidente).

7.2. Conclusiones y estudio de ambientalización

7.2.1. Conclusiones

Antes de empezar con las conclusiones referentes a los resultados obtenidos en el TFC, vamos a mencionar en modo de resumen lo que hemos estudiado para llegar a estos.

En primer lugar nos hemos introducido en las redes ad-hoc y VANET para conocer sus características y posibles aplicaciones, hemos realizado un visionado a las herramientas para simular VANETS mas utilizadas por los investigadores para poder escoger las mejores, hemos elegido y estudiado los protocolos con código disponible para simular, hemos meditado todos los parámetros, escenarios y casos a evaluar en nuestras simulaciones, hemos aprendido a utilizar las herramientas para obtener los resultados deseados y finalmente después de obtenerlos, presentamos las conclusiones siguientes:

Una de las primeras conclusiones, y quizá la mas inmediata, es poder afirmar que sí es posible la comunicación vehicle-to-vehicle en VANETS alcanzando velocidades en los casos más favorables planteados de hasta 135 Kbps con TCP y 131 Kbps con UDP, en este caso de goodput por fuente, válidas para desarrollar un intercambio de la información posible en este tipo de escenario según la ambientalización propuesta en el capítulo 6 (parámetros del estado del tráfico, estado de la carretera, posibles colapsos…etc. y tráfico multimedia).

Uno de los primeros aspectos que hemos comentado es el prácticamente impacto nulo de velocidad en el goodput por fuente en la autopista; esto es

debido a que todos los vehículos de este escenario van a la misma velocidad y éstas además son constantes, esto da lugar a que SUMO reparta todos los vehículos del escenario a la misma distancia entre ellos, en este caso la distancia de seguridad mínima, que en el caso mas extremo de nuestras simulaciones (cuando los vehículos circulan a 140 km/h) será igual a 115,3 metros (siguiendo la fórmula utilizada para calcular la distancia de seguridad en función de la velocidad de una forma aproximada V^2/C dónde V^2 es la velocidad en Km/h al cuadrado y C es una constante igual a 170), es decir, que entre un vehículo y otro siempre será posible la comunicación ya que 115,3 metros es inferior al rango de transmisión configurado en estas simulaciones que es igual a 250 m.

Observando los datos numéricos del caso de la autopista (véase anexo C), ya que a simple vista observando las gráficas no se puede apreciar, se puede observar como el goodput va disminuyendo de una forma muy tenue entre los vehículos que van a 60 km/h y los que van a 140 Km/h, cosa que nos hace ver que si continuásemos con velocidades mas altas, en este caso, con mayores distancias entre vehículos, el goodput iría disminuyendo, cosa lógica, ya que si la distancia entre vehículos se aproxima o supera ligeramente al rango de transmisión, habrán paquetes o datagramas que serían descartados por el receptor por no pasar el umbral de potencia de detección mínima asociada al rango de transmisión determinado, ya que la atenuación asociada a la distancia hace disminuir la potencia con la que los paquetes o datagramas llegan a su destino, por tanto, al descartarse de un número mayor de éstos, tendríamos una disminución del goodput.

Otros de los factores que hemos visto que tienen impacto en el goodput por fuentes, haciéndolo disminuir exponencialmente son:

- Aumento del número de fuentes en el escenario manteniendo el número de vehículos constante.

- Aumento del número de vehículos, manteniendo constante el porcentaje de fuentes.

En ambos casos se produce una disminución del goodput ya que al haber un número mayor de nodos enviando paquetes o datagramas (ya sean nodos fuente o nodos intermedios), el medio físico tiene que ser compartido pues por más nodos, y por tanto existirán más colisiones y reintentos a la hora de acceder al medio físico, aparte de la necesidad de encaminar más paquetes o datagramas por los nodos intermedios en las comunicaciones, creación de un número mayor de rutas…etc. cosa que añadirá un tiempo en el cual los nodos receptores no estarán recibiendo información respecto al escenario con un número menor de nodos debido a los reintentos en el envío de paquetes que se han perdido, pérdidas definitivas de paquetes, etc. traducido así en una disminución del goodput.

Otros de los efectos que hemos podido observar es que tanto con tráfico TCP como con UDP hemos tenido unos valores muy próximos entre ambos, siendo los de UDP ligeramente inferiores; en este caso si hubiéramos evaluado el Throughput en vez del goodput, hubiéramos obtenido un valor mucho más amplio en el caso de UDP, ya que al ser tráfico a tasa constante y no orientado a conexión, las fuentes envían a dicha tasa ocurra lo que ocurra (sin esperar recibir paquetes de reconocimiento o acks conforme los paquetes enviados han sido correctamente recibidos), siendo éste un tráfico adecuado para intercambio de contenido multimedia, dónde son permitidas una serie de pérdidas, con tal de recibir dicho contenido de una forma inteligible. Esto provoca que un gran número de datagramas se pierdan a causa del desbordamiento de los buffers, que es exactamente lo que nos ocurría en nuestras simulaciones, dando por tanto, una tasa de paquetes recibidos muy similar a TCP, ya que en ambos casos, los buffers eran igual a 50 paquetes o datagramas, siendo en ambos casos también igual el tamaño de paquete. Por el otro lado, TCP al ser orientado a conexión posee una tasa de perdidas menor, haciendo retransmisiones de paquetes si fuera necesario, siendo más adecuado para intercambiar información dónde la llegada ordenada de los paquetes con una tasa de pérdida pequeña sea obligada para su compresión, dando en nuestras simulaciones valores mas altos de goodput por fuente frente a UDP. Por tanto se deduce que como UDP envía muchos más paquetes congestiona más el medio y consecuentemente los receptores acaban recibiendo menos paquetes que en el caso de TCP, aunque se hubiera enviado inicialmente un número muy superior de paquetes.

Teniendo en cuenta todo lo descrito, también podemos afirmar que en nuestro caso de la autopista, independientemente del tipo de tráfico (TCP o UDP) el goodput no se ve afectado por la velocidad de los vehículos, en cambio sí que disminuye al aumentar el porcentaje de fuentes.

En nuestras simulaciones hemos obtenido valores de goodput por fuente ligeramente más bajos en el entorno urbano frente a la autopista; recordemos que en el entorno urbano las velocidades de los vehículos que circulan son diferentes para cada uno (entre 10 y 50 Km/h) sufriendo aceleraciones y desaceleraciones diferentes también para cada vehículo, mientras que en la autopista todos los vehículos circulan a velocidad constante (a 100 km/h); esto provoca que las distancias en el entorno urbano sean mucho menores que en la autopista, donde en este caso (100 Km/h) siempre serán igual a 58,8 metros, mientras que en el entorno urbano cabe la posibilidad de que varios vehículos se encuentren prácticamente pegados mientras van circulando por el Eixample de Barcelona; esto provoca que hayan mas nodos compitiendo por el mismo medio físico a la hora de querer comunicarse con otro nodo, como hemos comentado anteriormente, existirán mas colisiones y por tanto reintentos a la hora de acceder al medio, cosa que añadirá un tiempo en el cual no se estará enviando información respecto a la autopista (debido a las colisiones habrá más reintentos, pérdidas definitivas de paquetes, etc.), lo que será traducido en una disminución de goodput por fuente, como hemos visto. Así pues vemos que uno de los parámetros verdaderamente decisivos es la densidad de los nodos, ya que al aumentar la densidad de vehículos (disminuyendo la distancia entre ellos) tendremos más nodos compitiendo por el medio.

Hemos de tener en cuenta que en nuestro escenario de entorno urbano, solamente hemos descrito la tipología de las calles, es decir, que en nuestro mapa generado no existen todos los elementos urbanos que existirían en la realidad como por ejemplo los edificios existentes, ya que actualmente no existe ningún simulador que pueda generar simulaciones VANET en un entorno o mapa digital teniendo en cuenta también esto, excepto el simulador CARISMA que hace una aproximación considerando la existencia de edificios a lo largo de las calles. Esto provocaría que al haber más obstáculos, hubiera mas dificultades en el medio radio a la hora de comunicarse (rebotes, desvanecimientos…etc.), traducido a unos valores reales mas bajos que los que hemos obtenido; por este motivo, como primera opción hemos realizado nuestras simulaciones en una autopista, donde no existen prácticamente obstáculos de este estilo. De todas maneras, vimos interesante realizar simulaciones en un entorno urbano, ya que lo que sí hemos podido plasmar es el tipo de movilidad y el tipo de calles que se darían, y constatar así un efecto directo de estos parámetros en el goodput con respecto al caso de la autopista.

Uno de los objetivos principales que nos planteábamos inicialmente en este TFC era el de poder realizar simulaciones lo más próximas a la realidad posible, y teniendo en cuenta los medios y herramientas existentes para esto; creemos firmemente que hemos utilizado todas las necesarias para hacerlo, así pues, después de tener en cuenta todas las conclusiones expuestas en este apartado, finalizamos éstas reafirmando lo que decíamos al inicio de las mismas, la posibilidad de comunicación VANET confirma nuestros objetivos propuestos y hace viable su existencia en un futuro próximo.

7.2.2. Estudio de ambientalización

En este apartado vamos a intentar desgranar el posible impacto ambiental del tema que envuelve a este proyecto. Cabe considerar que las VANETS están dirigidas al intercambio de información entre vehículos tal como hemos ido comentando a lo largo del TFC. Quizá a simple vista pueda parecer que esta tecnología incentive el uso del vehículo privado para moverse, pero hemos de decir, que dicha tecnología también puede aplicarse al transporte público a la hora de dar mas calidad también a una movilidad limpia y sostenible.

Otro de los posibles impactos de las comunicaciones VANETS, concretamente en las comunicaciones vehicle-to-infraestructure, será la necesidad de nuevas infraestructuras, o estructuras de refuerzo, en las cercanías de las vías a transitar por donde se quiera dar este tipo de servicio, lo que provocará un impacto directo sobre el entorno, cosa que requerirá de un uso responsable y adecuado a la hora de su colocación.

Es verdad que nuestro mundo se sigue rigiendo por intereses puramente económicos, y el petróleo y la gran utilización, a veces por fuerza, del transporte privado siguen mermando nuestro planeta por el efecto invernadero entre otras cosas. Por eso esperamos y deseamos, que por que no, para cuando entre en funcionamiento la tecnología VANET se apueste por un combustible mas limpio y por el transporte público, quizá una utopía para algunos, pero sí un sueño para los que creemos que otro mundo es posible.

BIBLIOGRAFÍA

[1] Remondo, D., "*Redes Ad Hoc*", Transparencias Intensificació en Xarxes Telemàtiques, Universitat Politècnica de Catalunya, Castelldefels, 2006.

[2] Domingo Aladrén, M.C, "*Diferenciación de servicios y mejora de la supervivencia en redes ad-hoc conectadas a redes fijas*", Tesis Doctoral, Universitat Politècnica de Catalunya, Castelldefels, 2005.

[3] VANET 2006, The Third ACM International Sponsored by ACM SIGMOBILE , "*Workshop on Vehicular Ad Hoc Networks*", In conjunction with ACM MobiCom, Los Angeles, 2006.

[4] Eichler, S., Schroth, C., Eberspächer, J., "*Car-to-Car Communication*", VDE Congress "*Innovations for Europe*", Poster session, Institute of Communication Networks, Technische Universität München, München, 2006.

[5] Vehicular Ad-Hoc Network (VANET), http://en.wikipedia.org/wiki/VANET

[6] Eichler, S., Schroth, C., Eberspächer, J., "*Car-to-Car Communication*" Institute of Communication Networks, Technische Universität München and Institute of Media and Communication Management, SAP Research CEC, University of St. Gallen, München, 2006.

[7] GST (Global System of Telematics),http://www.gstforum.org/

[8] Harri, J., Filali, F., Bonnet, C., *"Mobility Models for Vehicular Ad Hoc Networks: A Survey and Taxonomy",* Research Report RR-06-168, Institut Eurecom, Department of Mobile Communications, France, 2006.

[9] Mangharam, R., Weller, D.S, Stancil, D.D, Rajkumar, R., Parikh, J.S, " *GrooveSim: A Topography-Accurate Simulator for Geographic Routing in Vehicular Networks* " Dept. of Electrical & Computer Engineering and General Motors Corporation, Carnegie Mellon University, U.S.A Warren, MI. USA, 2005.

[10] Eichler, S., Ostermaier, B., Schroth, C., Kosch, T., "*Simulation of Car-to-Car Messaging: Analyzing the Impact on Road Traffic*", Institute of Communication Networks, Munich University of Technology and BMW Research and Technology, Munich, 2005.

[11] Capra, L., Emmerich, W., Mascolo, C., "*CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications*", IEEE Computer Society, 2003.

[12] SUMO, http://sumo.sourceforge.net

[13] NS2, Network Simulator, http://nsnam.isi.edu/nsnam/index.php/Main_Page

[14] Karnadi, F.K.; Mo, Z.H, Lan, K., "*Rapid Generation of Realistic Simulation for VANET*", Appeared in ACM MOBICOMM 2005 poster session, Germany, 2005.

[15] Estándar 802.11, http://es.wikipedia.org/wiki/IEEE_802.11#802.11b

[16] Remondo, D., "*Transmisión por radio y acceso al medio inalámbrico*" transparencias de Intensificació en Xarxes Telemàtiques, EPSC, Universitat Politécnica de Catalunya, Castelldefels, 2006.

[17] Intensificació en Xarxes Telemàtiques, transparencias "*Evolución de IEEE 802.11*", Universitat Politécnica de Catalunya, Castelldefels, 2006.

[18] Arquitecturas Telemátiques, transparencias "*Xarxes d'area local sense fills*", Universitat Politécnica de Catalunya, Castellldefels, 2005.

[19] Chakeres, I.D., Perkins, C.E., "*Dynamic MANET On-demand (DYMO) Routing Protocol*". Internet-Draft Version 8, IETF, 2007.

[20] Thorup, R.E., "*Implementing and Evaluating the DYMO Routing Protocol*". Master's Thesis, Department of Computer Science, University of Aarhus, Denmark, 2007.

[21] Krauss, S., "*Investigation of Collision Free Vehicle Dynamics*" , Microscopic Modeling of Traffic Flow, Deutsches Zentrum fur Luft– und Raumfahrt, Germany, 1998.

[22] Dhoutaut, D., Régis, A., Spies, F., "*Impact of Radio Propagation Models in Vehicular Ad Hoc Networks Simulations*", Laboratoire d'Informatique de l'Université de Franche Comté, France, 2006.

[23] D.B. Johnson and D.A. Maltz, "DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks," in Ad Hoc Networking, C.E. Perkins, ed. Addison Wesley, 2001.

ANEXO A: DIFERENCIAS ENTRE DYMO Y AODV

Al protocolo DYMO se le puede ver como una evolución de AODV, manteniendo la base de éste y añadiendo algunas variaciones. En este apartado vamos a ver concretamente cuales son estas diferencias entre ambos.

Una de las primeras modificaciones que encontramos frente a AODV es la inserción por parte de cada nodo intermedio de sus respectivas direcciones IP en los paquetes RREQ y RREP en el descubrimiento de ruta a la hora de establecer la ruta en ambos sentidos entre el nodo origen de la comunicación y el nodo destino. Es decir, añade esta característica de Path Accumulation (acumulación de ruta o camino) característica del protocolo DSR [23] y también de una variante del propio AODV, anterior a DYMO, denominada AODV-PA (AODV with Path Accumulation). Esta modificación ha sido realizada ya que en numerosos estudios y simulaciones se ha demostrado que mediante este mecanismo el protocolo AODV adoptaba una mayor escalabilidad, en redes de grandes dimensiones; y por tanto, mejoraba el comportamiento del mismo.

Otra de las modificaciones respecto a AODV es la no utilización por defecto de mensajes periódicos para monitorizar el estado de los enlaces; mientras AODV utiliza un mecanismo de HELLO MESSAGES, DYMO por defecto deja la responsabilidad de esto a la capa MAC, con la finalidad de no introducir más overhead o tráfico añadido en la red; por otro lado, en el ultimo draft de DYMO dan a escoger el tipo de mecanismo a utilizar para esta finalidad, dejándolo abierto (pudiendo utilizar también HELLO MESSAGES si el administrador lo requiere).

La ultima modificación relevante respecto a AODV es la eliminación de la “Precursor List”, ésta es una lista que tenía cada nodo y que estaba compuesta por una lista de nodos a través de los cuales el nodo perteneciente de la lista puede llegar a un nodo destino determinado (como una tabla de encaminamiento), se incluía un nodo a dicha lista cuando por ejemplo se recibía un RREP a través del mismo o se borraba cuando se recibía un RERR perteneciente a dicho nodo. Se vio que si se eliminaban estas listas, el protocolo AODV obtenía la misma funcionalidad pero en este caso reducía su tiempo de difusión debido a una carga menor por parte de los nodos, ya que en un entorno totalmente variable e impredecible, la información que pueden aportar las “Precursors Lists” puede quedarse obsoleta en muy poco tiempo, dejando esta responsabilidad DYMO directamente a los temporizadores del propio protocolo (véase apartado *5.2.2.1 para mas detalle).*

Tanto la no utilización de HELLO MESSAGES como la de las “Precursors Lists” son adoptadas también en este caso de una variante de AODV anterior a DYMO, denominada AODVjr [20].

ANEXO B: PROCEDIMIENTOS PARA LA SIMULACIÓN

Seguidamente vamos a describir la metodología que hemos utilizado para realizar nuestras simulaciones, describiendo así todos los procesos realizados para obtener cada valor de goodput representado.

B.1. Generación de mapas

B.1.1. Mapa de autopista

Para generar el mapa de la autopista hemos utilizado MOVE y su opción de generar el mapa manualmente; para ello hemos tenido que configurar 2 scripts, primero uno con extensión .nod.xml, en el que mediante coordenadas cartesianas (X/Y) definimos los puntos que describirán el inicio y final de las diferentes vías o carreteras de nuestro escenario, vías que serán definidas en el archivo con extensión .edg.xml.

Una vez configurados los 2 scripts anteriores, se los pasamos a SUMO, mediante una interfaz gráfica específica de MOVE, y generamos nuestro mapa definitivo con extensión .net.xml, pasándole antes al programa los parámetros que definirán la longitud, el número de carriles, la velocidad máxima permitida, la capacidad máxima de vehículos permitidos y la identidad de cada una de las vías o carreteras que componen nuestro mapa (véase figura B.1).

Comentar que por ejemplo hemos tenido que realizar nuestra autopista dividida en diferentes segmentos, dándole al primero de ellos una longitud menor respecto al resto, ya que SUMO genera los vehículos en subgrupos de una forma uniforme a lo largo de la carretera inicial (de donde salen los vehículos), cosa que en carreteras iniciales con una longitud grande respecto al número de coches que van a circular en ella, provoca que hayan grandes distancias entre estos subgrupos y haciendo así imposible la comunicación entre todos los nodos del escenario, cosa que evitamos con dicho procedimiento. Aparte de esto también hemos dispuesto cada carril de nuestra autopista como si de carreteras independientes se tratara con tal de repartir manualmente los vehículos entre los 4 carriles existentes y disminuir el tiempo necesario para generar los vehículos del escenario *(véase apartado B.2 Generación de rutas para más detalle)*.

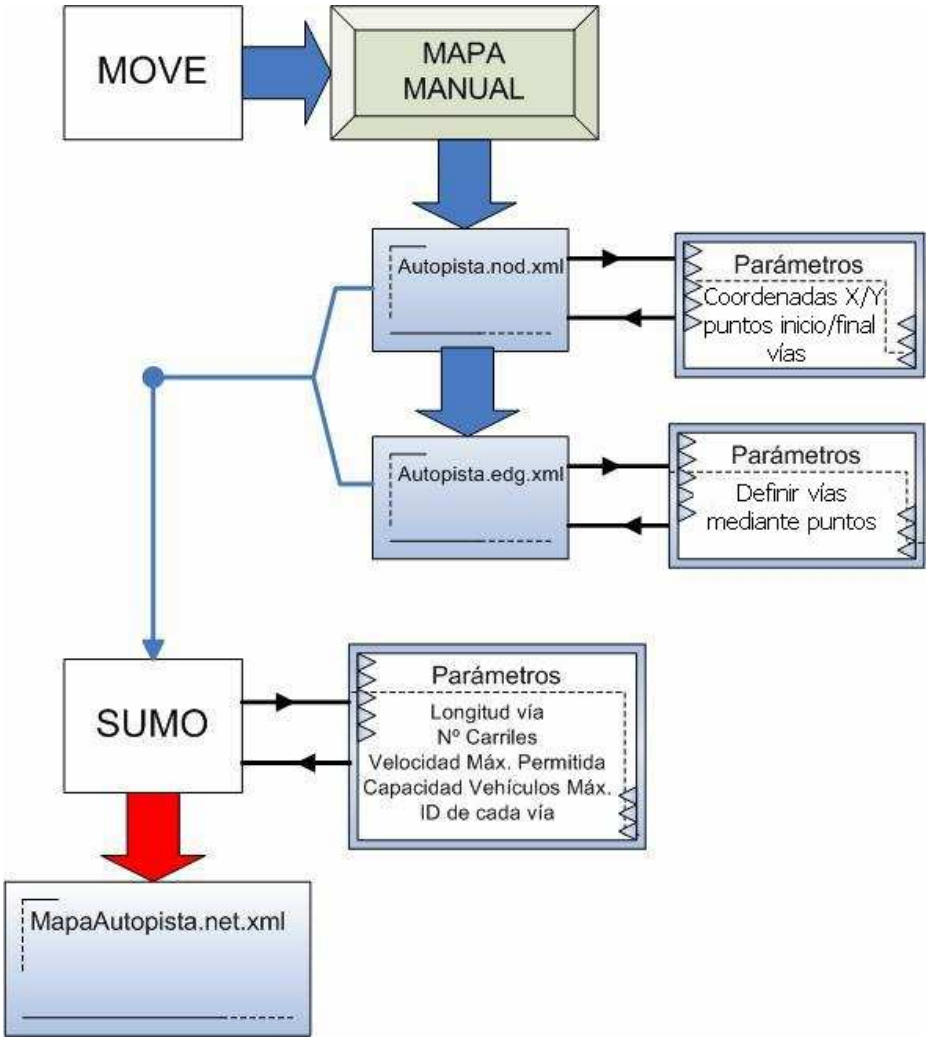

Figura B.1 Procedimiento para generar el mapa de la autopista

B.1.2. Mapa Urbano: Eixample de Barcelona

Para generar el mapa del Eixample de Barcelona, también lo hemos hecho mediante MOVE y la opción de generar un mapa automáticamente de tipo “grid” o cuadriculado, topología que sigue el trazo urbanístico del Eixample Barcelonés.

Después de pasarle los parámetros referentes a la longitud de vía (extensión de cada cuadrado o manzana) y el número de cuadrados a generar (número de manzanas) se nos genera mediante SUMO un archivo con la topología escogida de extensión .net.xml; para después abrirlo con un editor de textos para poder modificar los parámetros necesarios a la hora de adaptar nuestro escenario al Eixample, como el número de carriles, la velocidad máxima permitida, la capacidad de vehículos máxima permitida y la identidad de cada una de las vías que componen nuestro mapa, que servirá para identificar a estas mediante un nombre o número único; después de modificar los valores necesarios, ya tendremos nuestro mapa definitivo creado (véase figura B.2).

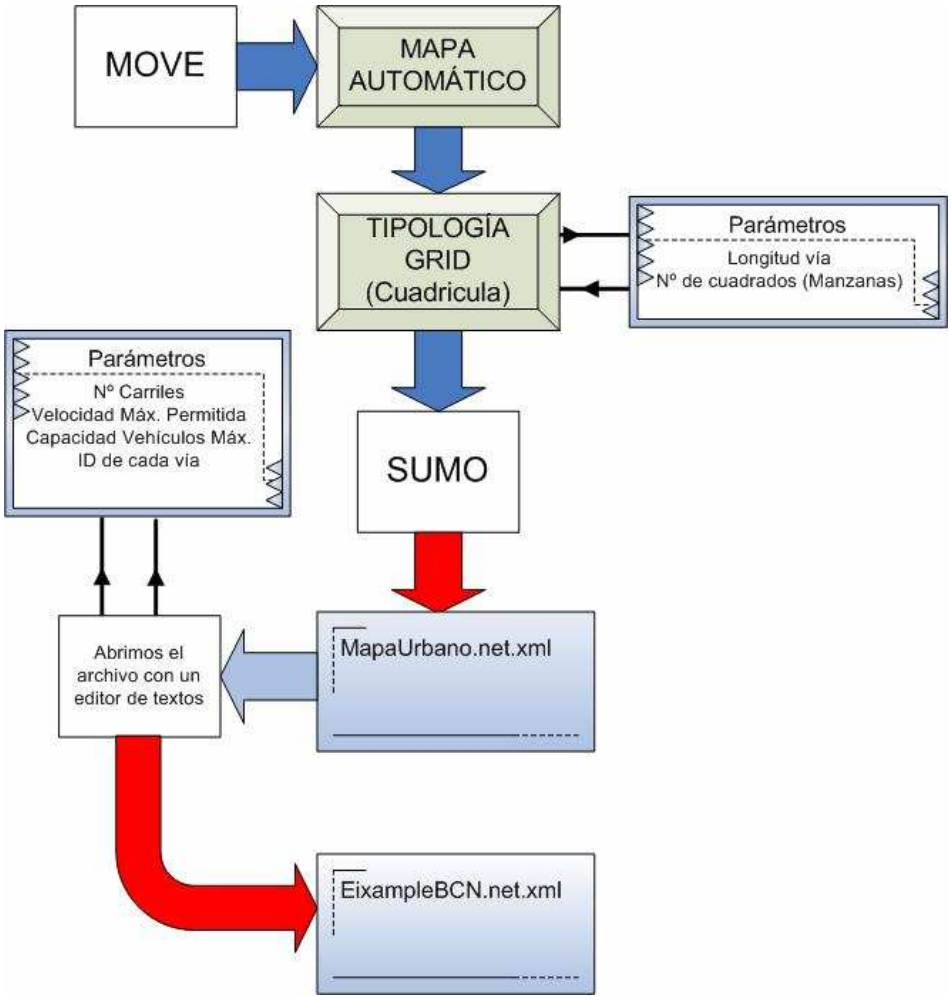

Figura B.2 Procedimiento para generar el mapa del Eixample de Barcelona

B.2. Generación de rutas

Para generar las rutas hemos escogido la opción de generar las rutas manualmente dentro de la interfaz de MOVE destinada para esto, pasando por parámetro el número de vehículos que habrá en la simulación, las velocidades, aceleraciones y desaceleraciones, la longitud y las vías que irán recorriendo cada vehículo o cada subgrupo de vehículos; otro parámetro a configurar es el intervalo de tiempo en el cual se generará cada vehículo en su vía o carretera correspondiente, (siendo el valor mínimo de 1 coche por segundo y por carretera o vía) ya que SUMO va generando los vehículos durante los primeros segundos de la simulación hasta llegar al número deseado, por esta razón en nuestras simulaciones los nodos comienzan a enviarse tráfico en el segundo 50 (caso extremo: 200 vehículos / 4 carriles de autopista= 50 vehículos por carril y 50 segundos necesarios para generarlos) (véase figura B.3).

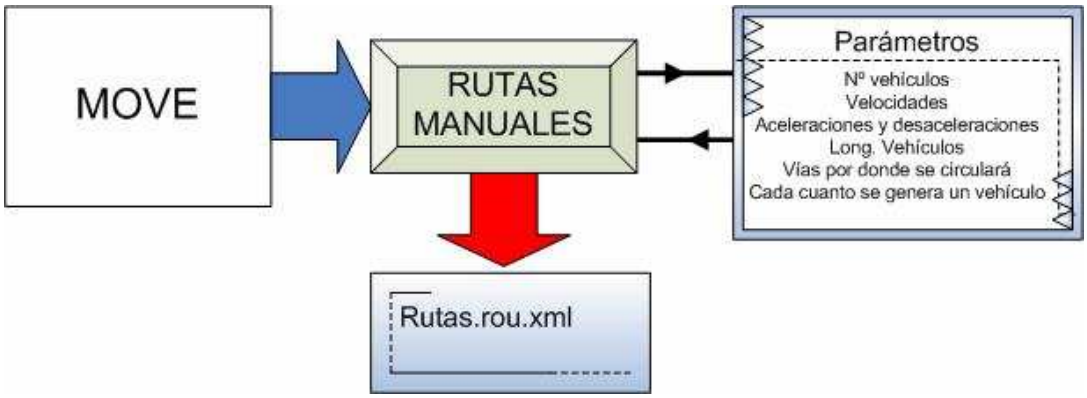

Figura B.3 Procedimiento para generar las rutas

B.3. Generación del archivo .tcl y obtención de resultados

Finalmente, procederemos a crear nuestro archivo con extensión .tcl que será el archivo que pasaremos al simulador NS2 para realizar la simulación y obtener los resultados deseados.

En primer lugar recuperamos el archivo de mapa y el de rutas, que hemos creado previamente y se los pasamos a MOVE para que cree un archivo con extensión .sumo.cfg, donde vendrá recogido todo lo necesario para que SUMO realice la simulación, pasándole antes por parámetro, durante cuanto tiempo deseamos que los vehiculos estén en movimiento. Una vez creado dicho archivo se lo pasamos a SUMO para que genere la simulación, dándo como resultado el archivo con extensión .move.trace donde tendremos recogido todo lo referente a dicha simulación que NS2 necesitará para crear los movimientos de los nodos; así pues abrimos el editor de NS2 de MOVE y adjuntamos el archivo con extensión .move.trace y el archivo de mapa, indicándole los parámetros referentes a NS2 (número de fuentes, intervalo de tiempo en el que se envía tráfico, protocolos, tamaño de los paquetes y datagramas …etc). De esta manera obtendremos nuestro archivo con extensión .tcl que será recogido por un script configurado por nosotros mismos con nombre ejecuto; y que se encarga de realizar 5 simulaciones consecutivas y filtrar la información que nos interesa (los paquetes o datagramas recibidos) y dejar cada archivo de información, denominados trazas (extensión .ns.tr) en una carpeta independiente. Finalmente calcularemos el goodput de cada simulación, haremos la media entre las 5 simulaciones realizadas y nos dispondremos a plasmar los resultados en una gráfica (véase figura B.4).

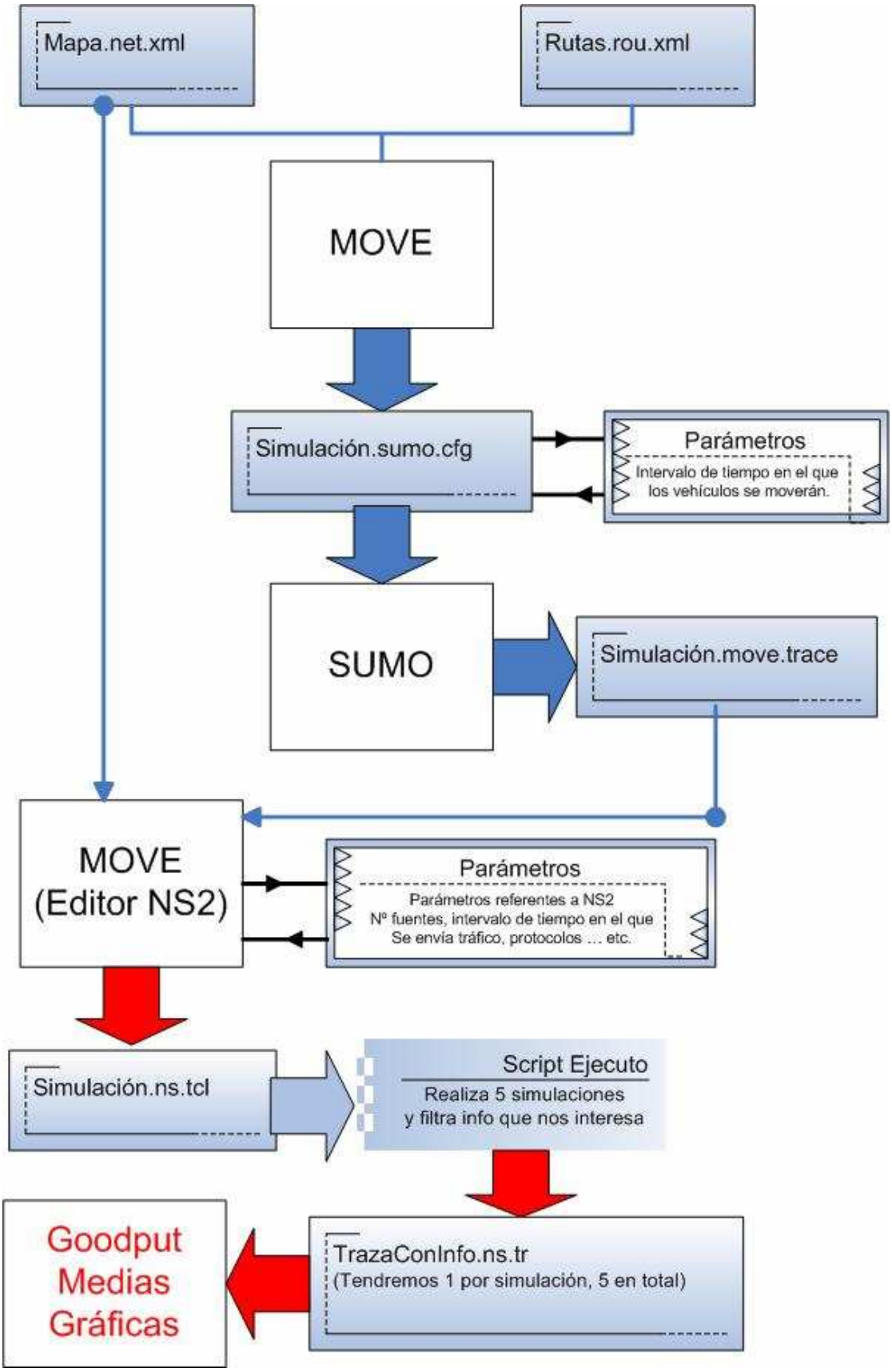

Figura B.4 Procedimiento para generar el archivo .tcl y la obtención de los resultados

ANEXO C: RESULTADOS NUMÉRICOS

C.1. Caso 1: Tráfico TCP/UDP en función de la velocidad, el número de vehículos y el % de fuentes en una autopista

C.1.1. Caso 1.1 Tráfico TCP dónde el 20% son fuentes

Tabla C.1 Caso 1.1

Nº de Nodos	Goodput por fuente en Kbps a 60 Km/h	Goodput por fuente en Kbps a 80 Km/h	Goodput por fuente en Kbps a 100 Km/h	Goodput por fuente en Kbps a 120 Km/h	Goodput por fuente en Kbps a 140 Km/h
20	135	134,7	134,4	131,4	131
40	63,8	63,7	63,7	63,4	60,2
60	39,4	37,6	35,7	35,5	35,3
80	24,8	23	22,6	19,6	19,4
100	15,3	15,3	15,4	14,2	13,8
120	11,6	11,5	11,5	11,2	10,3
140	7	7	7	6,5	6,4
160	5,9	5,9	5,9	4,8	4,9
180	5,2	5,2	5,2	4,7	4,5
200	3,5	3,5	3,5	3,5	3,2

C.1.2. Caso 1.2 Tráfico TCP dónde el 80% son fuentes

Tabla C.2. Caso 1.2

Nº de Nodos	Goodput por fuente en Kbps a 60 Km/h	Goodput por fuente en Kbps a 80 Km/h	Goodput por fuente en Kbps a 100 Km/h	Goodput por fuente en Kbps a 120 Km/h	Goodput por fuente en Kbps a 140 Km/h
20	30,4	30,4	30,4	30	29,4
40	13,3	12,6	12,6	12,6	12,5
60	6,6	6,5	6,4	6,4	6,2
80	3,7	3,7	3,5	3,5	3,2
100	1,8	1,6	1,5	1,5	1,5
120	1,3	1,3	1,3	1,2	1,1
140	1	1	1	1	1
160	0,8	0,7	0,7	0,7	0,7
180	0,7	0,7	0,7	0,7	0,6
200	0,6	0,6	0,6	0,5	0,5

C.1.3. Caso 1.3 Tráfico UDP dónde el 20% son fuentes

Tabla C.3 Caso 1.3

Nº de Nodos	Goodput por fuente en Kbps a 60 Km/h	Goodput por fuente en Kbps a 80 Km/h	Goodput por fuente en Kbps a 100 Km/h	Goodput por fuente en Kbps a 120 Km/h	Goodput por fuente en Kbps a 140 Km/h
20	131,4	131,1	131,1	131,1	130,7
40	64,3	63,4	63,2	62,7	62,6
60	37,6	37,6	37,6	37,5	36,5
80	23	20,7	20,4	20,4	20,2
100	13,2	13,2	13,2	13,1	13
120	9,5	9,3	9,3	9,3	9,1
140	7,4	7	7	7	6,7
160	6	6	6	6	6
180	4,6	4,4	4,4	4,4	4
200	3,5	3,1	3,1	3,1	3

C.1.4. Caso 1.4 Tráfico UDP dónde el 80% son fuentes

Tabla C.4 Caso 1.4

Nº de Nodos	Goodput por fuente en Kbps a 60 Km/h	Goodput por fuente en Kbps a 80 Km/h	Goodput por fuente en Kbps a 100 Km/h	Goodput por fuente en Kbps a 120 Km/h	Goodput por fuente en Kbps a 140 Km/h
20	26,5	26,5	26,2	26,2	25,5
40	11	11	11	11	11
60	5,1	5,1	5	5	4,9
80	2	2	2	2	1,8
100	1,3	1,3	1,3	1,3	1,2
120	1	1	1	1	1
140	0,8	0,8	0,8	0,7	0,7
160	0,7	0,7	0,7	0,6	0,6
180	0,6	0,6	0,6	0,6	0,6
200	0,5	0,5	0,5	0,5	0,5

C.2. Caso 2: Tráfico TCP/UDP y el impacto de un accidente sobre la comunicación en una autopista en función del % de fuentes

- 20 vehículos que circulan a 100 km/h.
- Tipo de accidente: A mitad de la simulación 2 vehículos que actúan como nodos intermedios sufren un accidente y a causa de ello no siguen circulando con el resto, esto provocará caídas de enlaces y establecimiento de nuevas rutas para la comunicación entre los nodos del escenario.

C.2.1. Caso 2.1 Tráfico TCP

Tabla C.5 Caso 2.1

	Goodput por fuente en Kbps						
Fuentes	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**
TCP sin accidente	134,4	87,2	64	51	42	34,5	30,4
TCP accidente	126	81,3	61	48	39,8	32	28,5

C.2.2. Caso 2.2 Tráfico UDP

Tabla C.6 Caso 2.2

	Goodput por fuente en Kbps						
Fuentes	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**
UDP sin accidente	131,1	86,5	63	49	40	33,2	26,2
UDP accidente	119	79	60	45,4	37,2	30	21,6

C.3. Caso 3: Tráfico TCP/UDP y el impacto de un entorno urbano (Eixample de Barcelona) frente al caso de la autopista en función del % de fuentes

- Entorno Urbano: Velocidades entre 10 y 50 km/h repartidas entre los 20 vehículos del escenario sufriendo aceleraciones y desaceleraciones.

- Autopista: *20 vehículos a 100 km/h.*

C.3.1. Caso 3.1 Tráfico TCP

Tabla C.7 Caso 3.1

	Goodput por fuente en Kbps						
Fuentes	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**
TCP Autopista	134,4	87,2	64	51	42	34,5	30,4
TCP Entorno Urbano	130	83,6	62,3	46,8	40	33,4	29

C.3.2. Caso 3.2 Tráfico UDP

Tabla C.8 Caso 3.2

	Goodput por fuente en Kbps						
Fuentes	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**
UDP Autopista	131,1	86,5	63	49	40	33,2	26,2
UDP Entorno Urbano	128,7	79	60	45,7	35,5	27,5	23,6

C.4. Caso 4: Tráfico TCP/UDP y el impacto de un accidente sobre la comunicación en un entorno urbano en función del % de fuentes

- Velocidades entre 10 y 50 km/h repartidas entre los 20 vehículos del escenario sufriendo aceleraciones y desaceleraciones.

- Tipo de accidente: A mitad de la simulación 2 vehículos que actúan como nodos intermedios sufren un accidente y a causa de ello no siguen circulando con el resto, esto provocará caídas de enlaces y establecimiento de nuevas rutas para la comunicación entre los nodos del escenario.

C.4.1. Caso 4.1 Tráfico TCP

Tabla C.9 Caso 4.1

	Goodput por fuente en Kbps						
Fuentes	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**
TCP Autopista	130	83,6	62,3	46,8	40	33,4	29
TCP Entorno Urbano	123,6	75,4	57	44	37	31,3	27,5

C.4.2. Caso 4.2 Tráfico UDP

Tabla C.10 Caso 4.2

	Goodput por fuente en Kbps						
Fuentes	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**
UDP Autopista	128,7	79	60	45,7	35,5	27,5	23,6
UDP Entorno Urbano	121	71,8	55	41	31,4	24	21,8