PROJECTE FINAL DE CARRERA

PERFORMANCE EVALUATION OF MANHATTAN DOWNTOWN SCENARIOS FOR VEHICULAR AD HOC NETWORKS WITH CITYMOB AND NCTUms

Estudis: Enginyeria de Telecomunicació

Autor: Xavier Campos Ríos / Diego Pastor Pérez

Directora: Mónica Aguilar Igartua

Co-Directora: Carolina Tripp Barba

Any: Juny 2011
AGRADECIMIENTOS

A nuestras directoras, Mónica y Carolina, por motivarnos continuamente y ofrecernos tanta atención y dedicación durante el desarrollo de este proyecto.

A todos nuestros compañeros de carrera, por los buenos momentos que hemos pasado juntos, en clase, estudiando, o pasando las horas libres en el Polimenú.

Diego

A Rafael, María del Pilar, Julián y Patricia por apoyarme y exigirme cuando lo necesitaba.

A mis abuelos Diego, y a Prudencia, Julián y Julia que ya no me acompañan; estoy convencido de que disfrutan tanto o más que yo de este momento.

A mis amigos, por tener la paciencia necesaria para aguantarme y por hacerme reír siempre que lo he necesitado.

Xavi

Aos meus pais, a Ana, ás miñas avoas e na memoria dos meus avós, aos que tamén lles tivese gustado ver este proxecto rematado. Tamén aos meus padrínos, Anna e Pere.

A Alba, por apoyarme y ayudarme con todo lo que hago, y por estar siempre aquí, conmigo.

A Juan, porque aunque nos vemos menos de lo que me gustaría, es una de las personas más importantes para mí.

A los amigos, en general. Este proyecto ha demostrado que cuando las cosas se hacen en buena compañía todo resulta más sencillo.
INDEX

INDEX OF FIGURES AND TABLES ............................................................................................................. 7
GLOSSARY OF ACRONYMS .................................................................................................................. 9
RESUM.................................................................................................................................................. 12
RESUMEN ............................................................................................................................................. 13
ABSTRACT ............................................................................................................................................ 14
1. INTRODUCTION AND GOALS .................................................................................................. 15
2. INTRODUCTION TO MOBILE AND VEHICULAR NETWORKS .................................................... 17
   2.1. INTRODUCING MANETs (MOBILE AD HOC NETWORKS) ................................................... 19
       2.1.1. Challenges in MANETs ..................................................................................................... 20
             2.1.1.1. Scalability Weakness .................................................................................................. 20
             2.1.1.2. Routing ..................................................................................................................... 20
             2.1.1.3. Security ..................................................................................................................... 21
             2.1.1.4. Quality of Service (QoS) .......................................................................................... 21
             2.1.1.5. Energy-constrained operations .................................................................................. 21
             2.1.1.6. Interoperation ............................................................................................................ 22
       2.1.2. Applications of MANETs .................................................................................................. 22
             2.1.2.1. Community Networking .......................................................................................... 22
             2.1.2.2. Crisis-management applications .............................................................................. 22
             2.1.2.3. Personal Area Networking ......................................................................................... 23
             2.1.2.4. Military battlefield applications ................................................................................. 23
       2.1.3. Wireless Sensor Networks (WSN) ............................................................................... 23
             2.1.3.1. Properties .................................................................................................................. 24
             2.1.3.2. Current deployments and applications ...................................................................... 25
                 2.1.3.2.1. Remote Ecological Micro-Sensor Network .......................................................... 25
                 2.1.3.2.2. Environment Observation and Forecasting System ........................................... 25
                 2.1.3.2.3. Disaster Relief Management .............................................................................. 26
                 2.1.3.2.4. Health care monitoring ........................................................................................ 26
                 2.1.3.2.5. DARPA Efforts towards Wireless Sensor Networks ............................................ 26
       2.2. INTRODUCING VANETs (VEHICULAR AD HOC NETWORKS) ........................................... 27
             2.2.1. Properties ...................................................................................................................... 28
             2.2.2. State of the art in VANETs ........................................................................................... 29
                 2.2.2.1. VANETs Applications .............................................................................................. 30
INDEX

2.2.2.1. Safety-related Applications ........................................................................................................ 30
2.2.2.1.2. Comfort Applications ........................................................................................................... 31
2.2.2.1.3. Applications for Administration .......................................................................................... 31
2.2.2.2. Challenges in VANETs ............................................................................................................. 32
2.2.2.2.1. Routing .............................................................................................................................. 32
2.2.2.2.2. Security ............................................................................................................................. 32
2.2.2.2.3. Quality of Service (QoS) .................................................................................................... 33
2.2.2.2.4. Power Management ........................................................................................................... 33
2.2.2.3. Current projects and activities in VANETs ............................................................................. 33
2.2.2.3.1. Standardization process and research projects initiatives in VANETs ...................... 34
2.2.2.3.2. Vehicular Mobility Models ............................................................................................... 37
2.2.2.3.3. Other research areas in VANETs ..................................................................................... 39
2.3. VEHICULAR TRAFFIC MODELS IN VANETS ........................................................................... 40
2.3.1. Freeway model ............................................................................................................................. 40
2.3.2. Manhattan model ....................................................................................................................... 41
2.3.3. City Section Mobility (CSM) ..................................................................................................... 41
2.3.4. Stop Sign Model (SSM) .............................................................................................................. 41
2.3.5. Traffic Sign Model (TSM) .......................................................................................................... 42
2.3.6. STRAW (Street Random Waypoint) ......................................................................................... 42
2.3.7. MOVE (MObility model generator for VEHicular network) .................................................... 43
2.3.8. BonnMotion ............................................................................................................................... 44
2.3.9. VanetMobiSim ............................................................................................................................ 44
2.3.10. SUMO (Simulation of Urban Mobility) .................................................................................... 45
2.3.11. FreeSim .................................................................................................................................. 45
2.3.12. CityMob .................................................................................................................................. 46
3. TRANSLATOR PROGRAM. ADAPTING SIMULATORS’ OUTPUT TO NCTUNS. ......................... 48
3.1. DESCRIPTION OF THE TRANSLATOR PROGRAM ........................................................................ 48
3.1.1. Main ........................................................................................................................................ 48
3.1.2. Read ......................................................................................................................................... 51
3.1.3. Translate ................................................................................................................................... 53
3.1.4. Write ......................................................................................................................................... 55
3.2. EXECUTING TRANSLATOR PROGRAM ..................................................................................... 58
4. SIMULATION ENVIRONMENTS AND SCENARIO ........................................................................... 59
4.1. THE CITYMOB MOVEMENT SIMULATOR .................................................................................. 59
# INDEX OF FIGURES AND TABLES

## FIGURES

<table>
<thead>
<tr>
<th>FIG.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>ARCHITECTURE OF A WSN</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>MONITORING VOLCANIC ERUPTIONS WITH A WSN [42]</td>
<td>25</td>
</tr>
<tr>
<td>2.3</td>
<td>WIRELESS VEHICULAR NETWORKS</td>
<td>27</td>
</tr>
<tr>
<td>2.4</td>
<td>APPLICATION DOMAINS [47]</td>
<td>28</td>
</tr>
<tr>
<td>2.5</td>
<td>ARCHITECTURE ON VANETS COMMUNICATION [49]</td>
<td>36</td>
</tr>
<tr>
<td>2.6</td>
<td>SCREENSHOT OF STRAW WITH THE ENHANCED VISUALIZATION TOOL</td>
<td>43</td>
</tr>
<tr>
<td>2.7</td>
<td>SCREENSHOT OF MOVE</td>
<td>44</td>
</tr>
<tr>
<td>2.8</td>
<td>SCREENSHOT OF VANETMOBISIM</td>
<td>45</td>
</tr>
<tr>
<td>2.9</td>
<td>SCREENSHOT OF SUMO</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>FEDORA-NCTUNS SCREENSHOT</td>
<td>63</td>
</tr>
<tr>
<td>4.2</td>
<td>NCTUNS 6.0 SCREENSHOT</td>
<td>64</td>
</tr>
<tr>
<td>4.3</td>
<td>DRAW TOPOLOGY MODE</td>
<td>64</td>
</tr>
<tr>
<td>4.4</td>
<td>ITS ROAD SEGMENT</td>
<td>64</td>
</tr>
<tr>
<td>4.5</td>
<td>EXAMPLE OF A VALID SCENARIO IN NCTUNS</td>
<td>65</td>
</tr>
<tr>
<td>4.6</td>
<td>ITS CROSSROAD</td>
<td>65</td>
</tr>
<tr>
<td>4.7</td>
<td>802.11(B) MOBILE NODE (AD HOC MODE)</td>
<td>65</td>
</tr>
<tr>
<td>4.8</td>
<td>802.11(B) AD-HOC MODE INTERFACE</td>
<td>65</td>
</tr>
<tr>
<td>4.9</td>
<td>EDIT PROPERTY MODE</td>
<td>66</td>
</tr>
<tr>
<td>4.10</td>
<td>PROTOCOL STACK IN THE NODE EDITOR</td>
<td>66</td>
</tr>
<tr>
<td>4.11</td>
<td>C.P.A.N.S.T. BUTTON</td>
<td>66</td>
</tr>
<tr>
<td>4.12</td>
<td>MAKING ALL NODES RECEIVERS</td>
<td>67</td>
</tr>
<tr>
<td>4.13</td>
<td>PHYSICAL LAYER EDITOR BUTTON</td>
<td>67</td>
</tr>
<tr>
<td>4.14</td>
<td>PHYSICAL LAYER EDITOR</td>
<td>68</td>
</tr>
<tr>
<td>4.15</td>
<td>NODES TO BE SELECTED</td>
<td>68</td>
</tr>
<tr>
<td>4.16</td>
<td>SIMULATION MENU</td>
<td>69</td>
</tr>
<tr>
<td>4.17</td>
<td>RUN SIMULATION MODE</td>
<td>69</td>
</tr>
<tr>
<td>4.18</td>
<td>EXAMPLE OF SCENARIO</td>
<td>71</td>
</tr>
<tr>
<td>4.19</td>
<td>IMPORT MOBILE NODES</td>
<td>73</td>
</tr>
</tbody>
</table>
FIG. 4.20. SCENARIO LOADED ON NCTUNS ................................................................. 73
FIG. 4.21. COMMUNICATION PATTERN ........................................................................ 74
FIG. 4.22. MESSAGE “SIMULATION IS DONE” .............................................................. 74
FIG. 5.1. PACKET LOSSES AS A FUNCTION OF FADING MODEL ..................................... 79
FIG. 5.2. PACKET LOSSES AS A FUNCTION OF SPEED WITH RICEAN MODE .................. 79
FIG. 5.3. PACKET LOSSES AS A FUNCTION OF SPEED WITH RAYLEIGH MODE .............. 80
FIG. 5.4. END-TO-END DELAY AS A FUNCTION OF FADING MODEL ............................... 81
FIG. 5.5. END-TO-END DELAY AS A FUNCTION OF SPEED WITH RICEAN MODEL .......... 81
FIG. 5.6. END-TO-END DELAY AS A FUNCTION OF SPEED WITH RAYLEIGH MODE ........ 82
FIG. 5.7. THROUGHPUT AS A FUNCTION OF FADING MODELS ....................................... 83
FIG. 5.8. THROUGHPUT AS A FUNCTION OF SPEED WITH RICEAN MODEL ..................... 83
FIG. 5.9. THROUGHPUT AS A FUNCTION OF SPEED WITH RAYLEIGH MODEL ................. 84
FIG. A.1. EXAMPLE OF CONFIDENCE INTERVALS .......................................................... 97

TABLES

TABLE 2.1. COMPARISON OF VEHICULAR TRAFFIC MODELS ........................................ 46
TABLE 3.1. TITLE ............................................................................................................. 55
GLOSSARY OF ACRONYMS

ACK        ACKnowledgment
ACL        Audit Command Language
ADAS       Advanced Driver Assistance Services
AODV       Ad hoc On-demand Distance Vector
bps        bits per second
CALM       Communication Architecture for Land Mobile environment
CANU       Communication in Ad hoc Networks for Ubiquitous computing
CI         Confidence Interval
CICAS      Cooperative Intersection Collision Avoidance System
COOPERS    Co-operative Systems for Intelligent Road Safety
CSM        City Section Mobility
CTS        Clear To Send
CVIS       Co-operative Vehicle-Infrastructure Systems
DARPA      Defence Advanced Research Projects Agency
DM         Downtown Model
DNT        Delay Tolerant Network
DSR        Dynamic Source Routing
EOFS       Environment Observation and Forecasting System
GPRS       General Packet Radio Service
GPS        Global Positioning System
GRC        Networking Research Group
GSM        Global System for Mobile communications
GST        Global System of Telematics
GUI        Graphical User Interface
IEEE       Institute of Electrical and Electronic Engineers
IEEE       Institute of Electrical and Electronics Engineers
IMM        Integrated Mobility Model
IP         Internet Protocol
ISO  International Organization for Standardization
ITS  Intelligent transportation system
IVBSS  Integrated Vehicle-Based Safety Systems
JITEC  Joint Information Technology Experts Committee
LAGAD  Location-Aided Gateway Advertisement and Discovery
LAN  Local Area Network
MAC  Medium Access Control
MANET  Mobile Ad hoc NETwork
META  MEtropolitan TAxis
Mi-VANET  Mobile Infrastructure Based VANET
MIRT  Mobile Infrared Transmitter
MM  Manhattan Model
MOVE  MObility model generator for VEhicular networks
NCTUns  National Chiao Tung University Network Simulator
NOW  Network On Wheels
NS-2  Network Simulator
OBU  On Board Unit
OLSR  Optimized Link State Routing
OS  Operating System
PAN  Personal Area Network
PDA  Personal Digital Assistant
POI  Point Of Interest
QoS  Quality of Service
RAW  RAndom Waypoint
RSE  Road Side Equipment
RSU  RoadSide Unit
RTS  Requests To Send
RTX  Retransmission
RUM  Rice University Model
RWP  Random Waypoint
RX  Response
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEM</td>
<td>Standard Error of the Mean</td>
</tr>
<tr>
<td>SM</td>
<td>Simple Model</td>
</tr>
<tr>
<td>SSIM</td>
<td>Smart Sensors and Integrated Microsystems</td>
</tr>
<tr>
<td>SSM</td>
<td>Stop Sign Model</td>
</tr>
<tr>
<td>STRAW</td>
<td>Street Random Waypoint</td>
</tr>
<tr>
<td>SUMO</td>
<td>Simulation of Urban Mobility</td>
</tr>
<tr>
<td>TCP</td>
<td>Transfer Control Protocol</td>
</tr>
<tr>
<td>TSM</td>
<td>Traffic Sign Model</td>
</tr>
<tr>
<td>TX</td>
<td>Transmission</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunication System</td>
</tr>
<tr>
<td>UPC</td>
<td>Technical University of Catalonia</td>
</tr>
<tr>
<td>UPV</td>
<td>Technical University of Valencia</td>
</tr>
<tr>
<td>V2V</td>
<td>Vehicle to Vehicle</td>
</tr>
<tr>
<td>VANET</td>
<td>Vehicular Ad hoc NETwork</td>
</tr>
<tr>
<td>Willwarn</td>
<td>Wireless Local Danger Warning</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>
RESUM

El continu creixement del parc automobilístic a nivell mundial ha obert noves oportunitats de negoci i noves necessitats de seguretat viària. La mobilitat constant dels individus i la necessitat d’estar sempre connectats fa de l’estudi de les possibilitats de connectivitat i transmissió d’informació entre vehicles un dels principals objectius dels fabricants de cotxes.

Els costos socials que acompanyen aquest avanç – augment dels accidents, saturació de les ciutats, contaminació, etc. – Posen de manifest la necessitat d’augmentar la seguretat dels automòbils mitjançant la transmissió d’informació sobre l’estat del trànsit i carreteres fent també interessant l’estudi de les VANETs (xarxes ad hoc vehiculares) per part de les autoritats i l’Administració.

El projecte comença fent un resum de la situació actual dels estudis sobre VANETs, especialment en el camp de la simulació. A causa de l’elevat cost que suposa implementar els sistemes de transmissió necessaris per a realitzar estudis d’aquestes xarxes vehiculares, la recerca de situacions el més reals possibles per obtenir els resultats més fiables fan de les simulacions una eina bàsica per a l’estudi d’aquest tipus de xarxes de comunicació. Per a la implementació i estudi d’aquestes simulacions són necessàries dues eines bàsiques: un simulador de moviment que s’encarregui de simular de la forma més realista possible el moviment dels vehicles sobre un escenari preestablert, i un simulador de xarxa, amb el qual es puguin dissenyar els esquemes de comunicació entre els vehicles i les característiques físiques de les transmissions. En el nostre cas, el simulador de moviment escollit és el CityMob [29] i el simulador de xarxa el NCTUns [31].

La majoria dels simuladors de mobilitat existents estan adaptats al simulador de xarxes NS-2 [65]. El nostre simulador NCTUns presenta millors característiques que l’NS-2 en el camp de les VANETs. És per això que sorgeix la necessitat d’adaptar els simuladors de moviment al nostre simulador de xarxa. En el projecte s’ha desenvolupat un programa que fa precisament aquesta funció, adaptar les dades sobre la mobilitat dels nodes extrets de qualsevol simulador de moviment compatible amb el NS-2 al NCTUns.

Utilitzant el programa traductor hem pogut implementar escenaris de xarxes vehiculares realistes en el simulador de xarxa NCTUns i realitzar diferents simulacions mitjançant les quals hem fet una avaluació de prestacions, hem estudiant les pèrdues, el retard mitjà i el throughput en funció de la velocitat dels vehicles i de diferents models de esvaïments del canal de radio. Els resultats obtinguts s’aproximen als esperats de manera teòrica.
RESUMEN

El continuo crecimiento del parque automovilístico a nivel mundial ha abierto nuevas oportunidades de negocio así como nuevas necesidades de seguridad vial. La movilidad constante de los individuos y la necesidad de estar siempre conectados hace del estudio de las posibilidades de conectividad y transmisión de información entre vehículos uno de los principales objetivos de los fabricantes de coches. Los costes sociales que acompañan a este avance – aumento de los accidentes, saturación de las ciudades, contaminación, etc. – ponen de manifiesto la necesidad de aumentar la seguridad de los automóviles mediante la transmisión de información sobre el estado del tráfico y carreteras haciendo también interesante el estudio de las VANETs (redes ad hoc vehiculares) por parte de las autoridades y la Administración.

El proyecto empieza haciendo un resumen de la situación actual de los estudios sobre VANETs, especialmente en el campo de la simulación. Debido al elevado coste que supone implementar los sistemas de transmisión necesarios para realizar estudios de estas redes vehiculares, la búsqueda de situaciones lo más reales posibles para obtener los resultados más fiables hacen de las simulaciones una herramienta básica para el estudio de este tipo de redes de comunicación. Para la implementación y estudio de dichas simulaciones son necesarias dos herramientas básicas: un simulador de movimiento que se encargue de simular de la forma más realista posible el movimiento de los vehículos sobre un escenario preestablecido; y un simulador de red, con el que se puedan diseñar los esquemas de comunicación entre los vehículos y las características físicas de las transmisiones. En nuestro caso, el simulador de movimiento escogido es el CityMob [29] y el simulador de red el NCTUns [31].

La mayoría de los simuladores de movilidad existentes están adaptados al simulador de redes NS-2 [65]. Nuestro simulador NCTUns presenta mejores características que el NS-2 en el campo de las VANETs. Es por ello que surge la necesidad de adaptar los simuladores de movimiento a nuestro simulador de red. En el proyecto se ha desarrollado un programa que hace precisamente esa función, adaptar los datos sobre la movilidad de los nodos extraídos de cualquier simulador de movimiento compatible con el NS-2 al NCTUns.

Utilizando el programa traductor hemos podido implementar escenarios de redes vehiculares realistas en el simulador de red NCTUns y realizar diferentes simulaciones mediante las cuales hemos llevado a cabo una evaluación de prestaciones, analizando las pérdidas, el retardo medio y el throughput en función de la velocidad de los vehículos y de diferentes modelos de desvanecimiento en el canal radio. Los resultados obtenidos se aproximan a los esperados de manera teórica.
ABSTRACT

The continued growth of the vehicular fleet has opened new business opportunities and new needs of road safety. The constant mobility of individuals and the necessity of staying connected makes studying the possibilities of connectivity and transmission of information between vehicles one of the main objectives of the cars manufacturers. The social costs that accompany this development – increase in accidents, congestion of cities, etc. – increases the needing of improving car safety by providing information on road traffic conditions and also making interesting the study of VANETs (vehicular ad hoc networks) by the authorities and the administration.

The project begins by summarizing the current status of research on VANETs, especially in the field of simulation. Due to the high cost of implementing the necessary transmission systems to study these vehicular networks, the search for situations as real as possible to obtain reliable results from simulations makes it a basic tool for the study of such networks communication. For the implementation and study of such simulations is needed two basic tools: a vehicular movement simulator to set up and to simulate in the most realistic way the movement of vehicles on a preset scene, and a network simulator, with which designs the communication patterns between the vehicles and the physical characteristics of transmissions. In our case, the vehicular simulator chosen is CityMob [29] and NCTUns [31] as a network simulator.

Most spread vehicular simulators are adapted to NS-2 [65] network simulator. Our network simulator NCTUns has better features than the NS-2 in the field of VANETs. This is why there is a needing to adapt our motion simulator to network simulator. During the project it has been developed a program that does t function, to adjust the data on the mobility of nodes extracted from any motion simulator compatible with the NS-2 to NCTUns.

Using translation program we can implement realistic vehicular network scenarios with the network simulator NCTUns and perform different simulations for making a performance evaluation, analyzing packet losses, end-to-end delay and the throughput as a function of the cars’ speed and different channel fading models. Obtained results are close to those expected theoretically.
1. INTRODUCTION AND GOALS

In recent years, the continuous advances in wireless communications have opened new research lines. Is in this situation where the concept of mobile networks was born. The constant mobility of individuals and the need of being always connected require focus research efforts in this field. Logically, connection and transmission of information from our cars is one of the main objectives. In addition to this, the exponential growth of vehicle number in cities has led that one of the main focuses in which to invest time and money is road safety. Besides, a better trip planning will decrease pollution, which currently is a problem of major importance.

VANETs (Vehicular Ad hoc NETworks) are a particular case of mobile networks, in which mobile nodes are vehicles. The concept—technology that uses moving cars as nodes in a network to create a temporary mobile network—initially seems simple, but it presents many challenges both theoretical and technical, such as routing protocols, transmission ranges and signal frequencies, quality of service, among others.

The study of VANETs is expensive and complicated because of the deployment that would be necessary to test it in real environments. Thus, this arises the need of performing simulation as realistic as possible in order to study, from a computer, settings and situations as similar as possible to those obtained in a real environment.

Firstly, a movement simulator that reproduces the movement of vehicles on maps and scenarios closer to reality is needed. Most of these VANET simulators are adapted to the NS-2 network simulator. However, in our case we are going to work with NCTUns network simulator since it includes some interesting features for VANETs. Because of this, one of our most important challenges in the present project was to develop a universal translator for NS-2
compatible-files for any mobility pattern simulator in order to make them compatible with NCTUns.

The choice of NCTUns as our network simulator is because this network simulator integrates some traffic simulation capabilities, such as designing maps and controlling vehicles mobility. Also, NCTUns includes a GUI to aid in the design process of the maps. But the main reasons of choosing this simulator is because NCTUns network simulator has many useful features: it directly uses the real-life Linux TCP/IP protocol stack to generate high-fidelity simulation results, it provides a highly-integrated and professional GUI environment and its simulation engine adopts an open-system architecture and is open source.

The rest of this project Memory is organized as follows:

The second chapter of the project introduces the concept of MANET, VANET and the most spread vehicular traffic models; mentioning their main applications and current projects and challenges in this areas. The third chapter presents the translator program developed by the team that translates any NS-2 movement pattern file in order to make it compatible with the NCTUns network simulator. The fourth chapter explains how the team developed the simulations employing CityMob [29] (vehicular movement generator) and NCTUns [31] (network simulator). In this chapter is also explained the scenario evaluated. The fifth chapter presents the results obtained. Here it is evaluated the influence of different fading models and car speed in the packet losses ratio, end-to-end delay and throughput. Finally, at the last chapter conclusions and future lines of work are outlined.
2. INTRODUCTION TO MOBILE AND VEHICULAR NETWORKS

Over recent years, the considerable mobile services sector growth around the world was certainly the major phenomenon in the telecommunications field. Wireless technology is capable of reaching virtually every location on the surface of the earth. With such success of mobile communication demand it is hardly surprising that wireless technology has led to the development of new multimedia services and to evolution in user requirements in terms of throughput and universal mobility throughout different systems. Mobile communications are already applied to the realm of personal and business computing, making that people living habits and working ways evolve [38]

Generally there are two distinct approaches for enabling wireless mobile units to communicate each other:

**Infrastructured or centralized networks:** Wireless mobile networks have traditionally been based on the cellular concept, where all the devices are connected to a central node which is the acting agent for all communications, and relied on good infrastructure support. Typical examples of this kind of wireless networks are GSM, UMTS, WLAN, etc.

**Infrastructureless:** As to infrastructureless approach, the mobile wireless network is commonly known as a mobile ad hoc networks or MANETs. A MANET is a collection of wireless nodes that can dynamically be set up anywhere and anytime without using a pre-existing network infrastructure. It is an autonomous system in which mobile host connected by wireless links are free to move randomly and often act as routers at the same time.
The design of network protocols for MANETs is a complex and wide area of research with many challenges. Hence, during the last few years, mobile ad hoc networks have become a very popular field of study within the research community.

In the next section a more detailed description of MANETs is presented, including its main features, challenges and an introduction of two well-known applications: WSNs and VANETs. Their properties and current deployments are also discussed.
2.1. **Introducing MANETs (Mobile Ad hoc Networks)**

A MANET is a collection of wireless devices than can dynamically form a network with a very simple deployment capability, paving the way for new applications which have not been able to emerge until now and offers solutions in multiple environments that have no infrastructure. These devices or nodes can move in a random way and are capable to self-organize themselves arbitrarily, collaborating in order to communications succeed. Examples of devices are laptops, PDAs, mobile phones, handhelds and wearable computers. The most outstanding features of MANETS are detailed below:

**Dynamic network topology**

The dynamically network topology is undoubtedly the element characterizing in MANETs. Since the nodes are mobile, the network topology may change rapidly and unpredictably and the connectivity among the terminal may vary with time. MANETs should adapt to the traffic and propagation conditions as well as the mobility patterns of the mobile networks nodes.

**Autonomous terminals and self-organization**

In MANETs, each mobile terminal is an autonomous node, which may function as both a host and a router and are responsible for dynamically discovering other nodes to communicate or handle the network configuration e.g. addressing, and position location issues.

**Distributed operation**

Since there is no background network for the central control of the network operations, the control management of the network is distributed among the devices. The nodes involved in a MANET should collaborate each other and act as a relay as needed, to implement routing and security functions.

**Multi-hop routing**

As delivering data packets from a source to its destination out of the direct wireless transmission range, the packets should be forwarded via one or more intermediate node.

**Fluctuating link-capacity**

The nature of high bit-error rates of wireless connections might be more critical in a MANET. The radio transmission rate is vulnerable to noise, fading, multiple access and interference conditions, and has less bandwidth than wired networks.

**Light-weight terminal**

In most cases, the MANET nodes are mobile devices with limited processing capability, small memory size and low power storage. Such devices need optimized algorithms to execute computing and communicating functions.
Scalability

Sometimes the number of devices which set up the network can increase until dozens or hundreds. Since there is no a central element which is in charge of network management, adding or rejecting nodes into the topology is a simple process.

2.1.1. Challenges in MANETs

Regardless of MANETs capabilities, possibilities and characteristics have risen a quickly spreading, that benefits lead to several challenges as well, which must be studied carefully. Researching in the area of mobile ad hoc networking is receiving more attention from academia, industry, and government during the last few years. Almost every aspect of the network has been explored in some level of detail although no ultimate resolution to any of the problems is found yet and there are already many open issues for research and significant contributions.

2.1.1.1. Scalability Weakness

The number of network nodes can be large and finding route to a destination also requires frequent exchange of routing control information among the nodes. Thus, the amount of update traffic can be substantial, and it is even higher when nodes with increased mobility are present.

2.1.1.2. Routing

Routing in ad hoc networks, which is quite different from traditional IP routing, is a particularly complex problem because of many factors including topology, selection of routers, locations of request initiator, resource limitations and unreliability of wireless links. A node at least needs to know the reachability information to its neighbors for determinate a packet route, while the network topology can change quite often in a MANET. Thus, routes may change while in use and become no longer valid in a very short time [39].

Since the arrival of ad hoc network concepts, many proposals have been studied, simulated and evaluated. The same proposals have led to variations, specializations to given environments and optimizations. Ad hoc routing proposals can be classified into two main categories: proactive and reactive routing. Proactive or table-driven protocols are directly inspired by routing protocols deployed in the Internet and consist of maintaining a routing table for sending data to any node in the network. Instead, ad hoc reactive routing algorithms research the vital information of a route between two nodes when a request for this route is expressed by the higher protocol layers. The protocol of this class attempt to keep the routes used and only those as up to date as possible in order to minimize the use of control messages to a minimum to save bandwidth. We can add other generally hybrid proposals to these two families, which includes both features.
2.1.1.3. Security

Research on security in addition to routing challenges has become a primary concern to mobile ad hoc networks. Historically, network security has adopted a centralized, largely protective paradigm to satisfy aforementioned requirements. This is effective because the privileges of every node in the network are managed by dedicated machines, e.g. authentication servers. Membership in such a network allows individual nodes to operate in an open fashion because it is simplicity guaranteed that any malicious user from outside world will not be allowed access. Although these solutions have been considered very early in the evolution of ad hoc networks, attempts to adapt similar client-server solutions to a decentralized environment have largely been ineffective.

Attempts to secure ad hoc networks must be ad hoc: they must establish security without reference to centralized. Instead, security paradigms should be carried out by the cooperation of all available nodes in the network.

An implementation of an inefficient authentication protocol in ad hoc networks may lead to a vulnerability increase and network will be compromised. Attacks from malicious nodes could range from message replay, passive eavesdropping to injecting erroneous messages or liable information into routing tables in order to make network congestion and denials of service by forwarding traffic to a black hole. Hence, security solutions need to consider malicious attacks not only from outside but also from within the network. So, key management and authentication procedure are issues that must be carefully considered.

2.1.1.4. Quality of Service (QoS)

As a wired network, the flows generated by applications supported by mobile ad hoc networks have diverse characteristics such as type and the volume of exchanged information, duration of interaction to name a few examples. These flows also have different QoS requirements: bandwidth requirements for video on demand or end-to-end requirements for voice over IP services. That is why uniform packet processing is not appropriate, and QoS support which considers the different QoS requirements is vital.

In MANETs, the dynamic networks environment with continuous topology changes and the limited resources raise that problem of QoS support at different levels [38].

2.1.1.5. Energy-constrained operations

Some or all of the nodes in an ad hoc network may rely on batteries or other exhaustible means for their energy. Therefore, energy conservative networks are becoming extremely popular within the ad hoc networking research.

The goals can be achieved either by developing better batteries, or by focusing on the devices’ networks interface, which is often the single largest consumer of power.
Energy efficiency at the network interface can be improved by developing transmission/reception technologies on the physical layer, but especially with specific networking algorithms. Nevertheless, energy conservation is currently being addressed in every layer of the protocol stack.

Much research has been carried out yet, however, there are still much more work to be done [38].

### 2.1.1.6. Interoperation

The self-organization of ad hoc networks is a challenge when two independently formed networks come physically close to each other. This is an unexplored research topic that has implications on all levels on the system design. The issue of joining two networks is not trivial: the networks may be using different synchronization, or even different MAC, routing or security protocols.

Another important issue comes into picture when we talk about all wireless networks. One of the most important aims of recent research on all wireless networks is to provide seamless integration of all types of networks. The issue raises questions on how the ad hoc network could be designed so that they are compatible with, for instance, wireless LANs, 3G and 4G cellular networks [39].

### 2.1.2. Applications of MANETs

With the increase of portable devices as well as progress in wireless communication, ad hoc networking is gaining importance with the number of widespread applications. Ad hoc networking can be applied anywhere where there is little or no communication infrastructure or the existing infrastructure is expensive or inconvenient to use. Ad hoc networking allows the devices to maintain connections to the network as well as easily adding and removing devices to and from the network. The set of applications for MANETs is diverse, some well-known are [39]:

#### 2.1.2.1. Community Networking

For some business scenarios, the need for collaborative computing might be more important outside office environments than inside a building. After all, it is the case where people do need to have outside meetings to cooperate and exchange information on a given project. It is also an interesting solution for neighborhood scenarios, stadiums, museums or airports.

#### 2.1.2.2. Crisis-management applications

That includes emergency or rescue operations, as a result of natural disasters where the entire communications infrastructure is in disarray or inoperative (Tsunamis, hurricanes). Restoring communications quickly is essential. By using ad hoc networks, an infrastructure could be set up in hours instead of days/week required for wire-line communications.
2.1.2.3. Personal Area Networking

A personal area network (PAN) is a short-range, localized network where nodes are usually associated with a given person. Bluetooth is an example of a technology aimed at supporting PANs by eliminating the need of wires between devices such as printers, cell phones and PDAs or laptop computers so on.

2.1.2.4. Military battlefield applications

MANETs networking was created for military purposes. Ad hoc networking would allow the military to take advantage of commonplace network technology to maintain an information network between the soldiers, vehicles, and military information head quarters.

Besides of these applications, two fields of study have become very interesting within the research community: wireless sensor networks and vehicular ad hoc networks. The special features and advantages of both of them are detailed below.

2.1.3. Wireless Sensor Networks (WSN)

In recent years, advances in wireless networking, micro-fabrication and integration and embedded microprocessors have enabled a new technological vision possible: wireless sensor networks. WSN consist of a large number of sensor nodes which collect data and interoperate each other to carry out functions involving some kind of tracking, monitoring or controlling. A sensor node is basically a device that converts a sensed attributed (such as temperature or vibrations) into a form understandable by the users. It consists of a transducer to sense a given physical quantity with a predefined precision, an embedded processor for local processing, small memory unit for storage and a wireless transceiver to transmit or receive data.

WSNs, which are considered as a special case of a MANET with reduced or no mobility, are expected to find increasing deployment in coming years, as they enable reliable monitoring and analysis of unknown and untested environments.

Development of wireless sensor networks was motivated by military applications such as battlefield surveillance and is now used in a variety of physical phenomena of interest: monitoring pedestrian or vehicular traffic in human-aware environments, report wildlife habitat conditions for environmental conservation, detect forest fires to aid rapid emergency response and monitoring industrial process or healthcare applications [40].

Fig. 2.1. Architecture of a WSN
2.1.3.1. Properties

The advancement in technology has made it possible to have a network of hundreds or even thousands of extremely small devices equipped with programmable computing and sensing and wireless communications capability, enhancing the reliability and the coverage area. Some of the features and challenges of WSN are as follows.

Easy of deployment

These wireless sensors can be deployed (dropped from a plane or placed in a factory) at the site of interest without any prior organization, thus reducing the installation cost and time, and also increasing the flexibility of deployment.

Fault tolerant

With macro-sensors, the failure of one node makes that area completely unmonitored until it is replaced. With wireless sensors, failure of one node does not affect the networks operation substantially as there are other adjacent nodes collecting similar data. At most, the accuracy of data collected may be somewhat reduced.

Networking and security implementation

Sensor nodes have limited computing power and therefore may not be able to run sophisticated network protocols or authentications and encryption algorithms contrary to ad hoc networks, leading to light weighted and simple versions of routing protocols and security implementations. Besides, two operational modes or states are defined for each node, awakened mode and slept mode, if the node must be active or not active, so the protocols and algorithm implementations must take into account this limitation too.

Data centric

In traditional networks, data is requested from a specific node. WSN are data centric, data is requested based on certain attributes. An attribute-based address is composed of a set of attribute-value pair query. For instance, if the query is something like temperature >35º, then only those device sensing temperature >35º need to respond and report their readings and other sensor can remain in the slept state. Once an event of interest is detected, the system should be able to configure itself so as to obtain high quality results [39].

Mobility

Since these wireless sensors are equipped with battery, they can possess limited mobility. Thus, if a region becomes unmonitored we can have the nodes rearrange themselves.

Energy conservation

Sensor nodes can use up their limited energy supply carrying out computations and transmitting information in a wireless environment. As such, energy-conserving forms of
communication and computation are crucial as the node lifetime shows a strong dependence on the battery lifetime.

### 2.1.3.2. Current deployments and applications

Judging by the interest shown by military, academia, and the media, dozen applications do exist for sensor networks such as weather monitoring, security and tactical surveillance, detecting ambient conditions or domotic and healthy applications. A brief description of some of them is presented below:

#### 2.1.3.2.1. Remote Ecological Micro-Sensor Network

PODS [41] is a research project undertaken at the University of Hawaii that has built a wireless network of environmental sensors to investigate species of plants will grow in one area. They deployed camouflaged sensor nodes in the Hawaii Volcanoes National Park, where two types of sensor data are collected: weather data are collected every ten minutes and image data are collected once per hour. Users employ the Internet to access the data from a server in University of Hawaii at Manoa.

![Fig. 2.2. Monitoring volcanic eruptions with a WSN [42]](image)

#### 2.1.3.2.2. Environment Observation and Forecasting System

The Environment Observation and Forecasting System (EOFS) is a distributed system that spans large geographic areas and monitors, models and forecast physical processes such as environmental pollution or flooding. CORIE [43] is a prototype of EOFS for the Columbia River (Oregon, USA) which integrates a real-time sensor network, a data management system and advanced numerical models. Approximately thirteen stationary sensor nodes fixed to a pier are deployed across the Columbia River estuary, while one mobile sensor station drifts off-shore. The stationary are powered by a power grid, while the mobile station uses solar panel to harness solar energy. Sensor data are transmitted via wireless links towards on-shore master stations which, in turns, forward the data to a centralized server where it serves as input to a computationally physical environment model used to guide forecasting.
2.1.3.2.3. Disaster Relief Management

Novel sensor network architecture has been proposed in [44] that could be useful for major disasters including earthquakes, storms, floods, fires and terrorist attacks. The sensor nodes are deployed randomly at homes, offices, and other places prior to the disaster and data collecting nodes communicate with database server for a given sub area which are linked to a central database for continuous update. Based on the statistical data form Izmit earthquake in 1999, various performance curves are obtained to indicate required average number of active sensor nodes to detect a disaster, probability of the disaster to be within the sensing range, total number of transmitted packets and the number of sensor nodes failed due to energy depletion.

2.1.3.2.4. Health care monitoring

An example of such application is the artificial retina developed within the Smart Sensors and Integrated Microsystems (SSIM) project [45], where a retina prosthesis chip consisting of one hundred microsensors are built and implanted within the human eye allowing patients with no vision or limited vision to see at an acceptable level. Wireless communication is required to suit the need for feedback control, image identification and validation.

2.1.3.2.5. DARPA Efforts towards Wireless Sensor Networks

The Defense Advanced Research Projects Agency (DARPA) has identified networked microsensors technology as a key application for the future. There are many interesting projects and experiments going under the DARPA SensIT (Sensor Information Technology) program which aims to develop the software for distributed micro-sensors.

Vehicle type identification is important for defense applications and an experiment was performed for two weeks by placing sensor boards in the Marine Corps Air Ground Combat Center in Twenty-nine Palms (California) for collecting acoustic data [46]. To detect presence of a vehicle, the sensor board is equipped with acoustic, seismic and passive Infra-Red sensors under program control and local processing is done to do local classification and storage.
2.2. **Introducing VANETs (Vehicular Ad-hoc Networks)**

Traditional traffic management systems are based on centralized infrastructures where cameras and sensors implemented along the road collect information on density and traffic state and transmit this data to a central unit to process it and make appropriate decisions. This type of system is very costly in terms of deployment and is characterized by a long reaction time for processing and information transfer in a context where information transmission delay is vital and is extremely important this type of system. However, with the rapid development of wireless communication technologies a new decentralized architecture based on vehicle-to-vehicle communications (V2V) has created a very real interest these last few years for car manufacturers, the R&D community and telecom operators. Thus, a new concept was born: a vehicular ad hoc network (VANET), which is no more than a specific application of traditional mobile ad hoc networks (MANET).

Vehicular Ad hoc NETworks (VANETs) have recently emerged as a platform to support intelligent inter-vehicle communication to improve traffic safety. The road-constrained characteristics of these networks and the high mobility of the vehicles, their unbounded power source, and the emergence of roadside wireless infrastructures make VANETs a challenging and promising research topic.

![Fig. 2.3. Wireless Vehicular Networks](image)

VANET’s aim to provide our cars and roads with capabilities to make road more secure and to make our time on the road more enjoyable, enabling communications among nearby vehicles (car to car communication) as well as between vehicles and nearby fixed equipment (car to...
The following variety of applications is a typical example of an intelligent transportation system (ITS):

- **Safety**, in which a warning message will be broadcasted from a vehicle to its neighborhood notifying about some event such as car collision or road surface conditions in order to decrease traffic accidents rate and enhance traffic flow control. It refers to applications or systems that increase the protection of the people in the vehicle as well as the vehicle itself.

- **Resource efficiency**, referring to increase traffic fluency with data such as enhanced route guidance or parking spot locator services. Better efficiency results in less congestion and lower fuel consumption, helping to minimize environmental and economic impact.

- **Infotainment and Advanced Driver Assistance Services (ADAS)**, combining information and entertainment and offering multimedia and Internet connectivity facilities for passengers.

The huge potential of car-to-car connectivity is fundamentally due to the constant growth of automotive market and the increasing demand for the car safety. Some issues relating to architecture, routing, security, performance or QoS should be investigated. It is necessary to put special attention to ensure interoperability through the standardization of protocols and interfaces in order to allow the communication between vehicles from different manufacturers.

### 2.2.1. Properties

As previously mentioned, a VANET represents a specific aspect of MANETs. Nevertheless, research works studied and carried out in the field of MANETs cannot be applied directly in the context of vehicular networks because of the characteristics of VANET making the application of MANET protocols and architectures inappropriate. In the following, the main properties and constraints related to the environment of vehicular ad hoc networks are presented below.
Processing, energy and communications capacity

Contrary to the context of mobile ad hoc networks where energy constraint represents one of the main challenges, vehicles in a VANET have no limit in terms of energy, have large processing capability and allow supporting several communications interfaces.

Environment and mobility model

Environments considered in ad hoc networks are often limited to open spaces or indoors. Vehicle movements are connected to road infrastructures, on highways, or within a metropolitan area. The constraints imposed by this type of environments, such as radio obstacles because of buildings, and multipath and fading effects, considerably affect the mobility model and radio transmission quality.

Type of information and diffusion

Since one of the key VANETs applications is prevention and road safety, the types of communications will focus on message broadcast from a source to several receivers. Nevertheless, the vehicles concerned by such diffusion depend on their location and their degree of implication in the event. In such situations, communications are mainly unidirectional.

Network topology and connectivity

Contrary to ad hoc networks, VANETs are characterized by very high mobility because of car speed. Thus, an element can quickly join or leave the network in a very short time, which makes topology changes frequently. Solutions must then consider this constraint where connectivity is one of the key parameter. In addition, properties inherent to VANETs, especially in terms of size, raise scaling problems and a complete revision of existing solutions is required.

Security

Data sensitivity transmitted over a VANET demonstrates a high need for security. In fact, the importance of security in this context is vital because of the critical consequences resulting from a violation or attack. In addition, with a highly dynamic environment characterized by almost instant arrivals and departures of cars, the deployment of a security solution must cope with specific configurations and constraints.

2.2.2. State of the art in VANETs

In this section is going to be explained the applications, challenges and current projects in vehicular ad hoc networks.
2.2.2.1. VANETs Applications

A VANET communication platform allows an enormous variety of applications aimed at administration, companies, drivers and people in the vehicle. These services will help and support topics as important as security driving, safety-related, traffic and fleet control as well as entertainment applications. Generally, from the connectivity point of view they could be divided into four groups: car-to-car traffic, car-to home, car-to-infrastructure and routing based applications.

2.2.2.1.1. Safety-related Applications

Safety-related applications are the most important kind of applications for VANETs due to its main objective: decrease of injuries and deaths due to vehicle accidents. In this context, the European Commission is making an important effort to investigate, develop and implement these services in order to come into effect as soon as possible.

Cooperative collision avoidance

This service is about helping driving by detecting possible obstacles in the road. One such application would be emergency notifications. In case of an accident or sudden hard breaking, a notification is sent to the following cars. This information could also be propagated by cars driving in the opposite direction and, thereby, conveyed to the vehicles that might run into the accident.

For a correct display of this service, it would be necessary a little installation in users’ equipment which sends information about his/her position, trajectory or speed to the neighbours. Also, another system in the vehicle permanently listens to rely information from the rest of vehicles and infrastructure.

Cooperative driver assistance system

This service exploits the exchange of sensor data or other status information among cars. The basic idea is to broaden the range of perception of the driver beyond his/her field of vision and further to assist the driver with autonomous assistance applications. By transmitting this data to cars following on the same road, drivers get information about hazards, obstacles or traffic flow ahead, resulting in more efficient and safe driving. Sensors could detect a danger and warn drivers with a brief description or even the driver could detect it and, through a vocal interface, describe the danger of the situation to the rest of users.

Information could be sent to every user in the network to inform, for example, that a traffic jam has started in a certain point of the road where we are driving. On the other side, it could be necessary a geocast message if, for example, we detect an oil puddle in one exit road; it is necessary to send the information only for those vehicles that will take that exit.
INTRODUCTION TO MOBILE AND VEHICULAR NETWORKS

**eCall**

*eCall* [12] is a project of the European Commission intended to bring rapid assistance to drivers involved in a collision anywhere in the European Union. In case of crash, an eCall-equipped vehicle automatically calls the nearest emergency centre. Even if no passenger is able to speak, e.g. due to injuries, a “Minimum Set of Data” is sent, which includes the exact location of the crash site. Shortly after the accident, emergency services therefore know that there has been an accident, and where exactly. eCall cuts emergency services’ response time. It goes down to 50% in the countryside and 60% in built-up areas. Annually, the quicker response will save around 2,500 lives in the European Union. The severity of injuries could be considerably reduced in 15% of cases. Drivers could also make an eCall by pushing a button in the vehicle. Witness of an accident can report it and automatically give the precise location.

### 2.2.2.1.2. Comfort Applications

The general aim of these applications is to improve passenger comfort and traffic efficiency. That could include nearest POI (Points Of Interest) localization, current traffic or weather information and interactive communication. All kinds of applications, which may run on top of TCP/IP stack might be applied here (online games or instant messaging). Another application is reception of data from commercial vehicles and roadside infrastructure about their businesses ('wireless advertising'), information about gas stations or enterprises which can set up stationary gateways to transmit marketing data to potential customers passing by, online help in case of breakdown, etc.

Furthermore, these services could be integrated with electronic payments, toll paying systems, etc. An important feature of comfort or commercial applications is that they should not interfere with safety applications. In this context traffic prioritizing and use of separate physical channels is a viable solution.

**Optimum route calculation with real-time traffic data**

This service could be used from the vehicle or from another point with an Internet connexion. The fact that, in the long term, all vehicles will be equipped with this system will make data-taking and data-publishing easier. This data, conveniently analyzed, will inform about the state of the road, prevent traffic jams, etc.

### 2.2.2.1.3. Applications for Administration

**Vehicle identification**

This service will provide a safe and fast way of information provision from vehicles without the need of stopping them. It will be necessary an appropriate legislation to allow that each vehicle stores the necessary information in an electronic format and its automatic transmission if it is required by an authorized device.
Vehicle identification service will help police in different ways: it would be possible to check if a vehicle and his/her driver have the necessary documentation or, if an infraction is detected, its report would be automatically processed, etc.

2.2.2.2. Challenges in VANETs

When deploying of a vehicular network system, several issues have to be resolved. VANET characteristics – rapid topology changes, frequent fragmentation, variable and highly dynamic scale and network density, etc. – are opening some brand new lines of investigation and challenges for the scientific community.

In what follows, we briefly mention some issues related to VANETs that need to be addressed.

2.2.2.2.1. Routing

In vehicular networks, mobility is constant. This fact causes extremely fast changes in network topology and involves the need to reconfigure the routing tables of each node. Frequent network partitioning in VANETs requires a different approach, e.g. the ‘carry and forward’ idea [1], where, if there is no a direct route, a packet is carried by a node until it could be forwarded to a node being closer to the destination, or the Delay Tolerant Networking.

**Delay Tolerant Networking** (DTN) is an approach to computer network architecture that wants to address the technical questions in heterogeneous networks, such as mobile networks, that could lack continuous network connectivity. The Delay Tolerant Network Research Group (DTNRG) [32] has defined an architecture based on a store and forward paradigm to interconnect networks, even without end-to-end connectivity. Each DTN node may store packets and, when appropriate, forward them towards the destination through intermediate nodes.

2.2.2.2.2. Security

Security is an issue that needs to be carefully addressed and assessed in the design of the vehicular communication system. In a wired network, user has to access to physical wire if s/he wants to access the network’s information. However, wireless communications are weak from this point of view, because they use air as the transmission medium. This problem gets worse in vehicular networks due to the non-existence of infrastructure that provides security services centralization like user authentication or packet ciphering. The issue to be addressed includes trust – vehicles must be able to trust the messages they receive—, resiliency and efficiency —e.g. real-time message authentication—.

Privacy is also considered a major issue. Anonymity must be preserved making impossible tracking a vehicle for non-trusted parties. Not taking into account privacy could result in a multiple lawsuits after the network is deployed.

IEEE 802.11p dynamically assigns MAC addresses, along with a mechanism to duplicate MAC address discovery, thus vehicles and drivers would not be traceable for the MAC address.
As we can read in the online magazine DailyTech [6], Daimler-Chrysler is putting the finishing touches on a new system, **Wireless Local Danger Warning (Willwarn)**, that uses on-board ABS (Antilock Brake System), ESP (Electronic Stability Control), EBD (Electronic Brake Distribution) and GPS (Global Positioning System) systems to monitor hazardous road conditions or broken down vehicles. The information collected is then displayed to the driver so that proper precautions can be taken to avoid or safely navigate problem areas on the road. The Willwarn system is based on IEEE 802.11a/p and made use of the communication platform developed in the German **Network on Wheels (NOW)** project [20].

### 2.2.2.2.3. Quality of Service (QoS)

Quality of Service in wired networks is provided by different types of resources reservation mechanisms. However, executing these mechanisms is very complex due to the special features of VANET, such as high mobility nodes and large-scale node population. Nowadays, there exist some proposals, nevertheless, the majority of them are theoretical, simulated or implemented with fewer nodes.

Tarng et al. [7] proposed a method based on the stability from the radio propagation: signal strength and path loss; Sun et al. [8] proposed a grid based protocol. The digital map is pre-set with a grid. A routing path is selected based on the traffic features, such as the intersection, the number of vehicles, and roads, in a grid. Recently, Yan et al. [9] proposed a routing selection and maintenance based on the mobility of vehicles. The main ideas are to select and maintain one routing path and one backup routing paths based on the mobility of vehicles (relative speed and direction). They reduce delay and response time in QoS terms because the retransmission caused by routing breakage is reduced and they improve the throughput in QoS terms.

### 2.2.2.2.4. Power Management

Power management in VANET is not concerned about energy efficiency, but rather about the transmission power - when too high, the on-going transmission could disrupt another transmission at a distant node due to interferences -. Thus the denser the network is, the lesser transmission power should be used. Several algorithms could be employed here, e.g. in [10] the power is adjusted to keep the number of neighbours within the maximum and minimum threshold. On the other hand, [11] concentrates on improving the 1-hop broadcast coverage by transmission power adjustments.

### 2.2.2.3. Current projects and activities in VANETs

Nowadays, several projects are currently being developed in the field of vehicular networks. In this section, it is going to be exposed those that we consider more relevant.
2.2.2.3.1. Standardization process and research projects initiatives in VANETs

This section briefly explains the main progress and purposes of the standardization process and research projects initiatives. It is foreseen that these solutions will finally converge, leading to a common, worldwide VANET platform.

In Europe, several projects are held, joining partners from the industry, governmental agencies and academia. Within the Framework Programme 6 of the European Union, four integrated projects were started in areas that touch the field of VANET: COOPERS, CVIS, SAFESPOT and PReVEnT:

- The project **Co-operative Systems for Intelligent Road Safety (COOPERS, 2006–10)** [21] focuses on innovative telematics applications for cooperative traffic management. From a communication perspective, it therefore primarily addresses vehicle-to-roadside communications and makes use of CALM (Communication Architecture for Land Mobile environment) standards like the CALM infrared communication interface. CALM is the ISO TC 204 (ITS) Working Group 16 (Communication) on ‘Continuous Air interface for Long and Medium distance’. CALM aims to support continuous communications for vehicles by making use of various media and communication interfaces.

- The project **Co-operative Vehicle-Infrastructure Systems (CVIS, 2006–10)** [22] is a major European research and development project financed by the European Commission aiming to design, develop and test the technologies needed to allow cars to communicate with each other and with the nearby roadside infrastructure. Based on such real-time road and traffic information and although there are no ultimate results because of the project is still in progress, many novel applications have been produced as well as four new services: COMM, allowing car-2-X communications, POMA, providing positioning system, COMO, providing monitoring services and FOAM, linking vehicles to infrastructure. The consequence will be increased road safety and efficiency, and reduced environmental impact. The CVIS main challenges are:
  - To create a unified technical solution allowing all vehicles and infrastructure elements to communicate with each other.
  - To enable a wide range of potential cooperative services to run on an open application framework in the vehicle and roadside equipment.
  - To define and validate an open architecture and system concept for a number of cooperative system applications, and develop common core components to support cooperation models in real-life applications and services for drivers, operators, industry and other key stakeholders.

- The project **SAFESPOT (2006–10)** [23] aims to design cooperative systems for road safety based on vehicle-to-vehicle and vehicle-to-infrastructure communication.
The project PReVENT (2006–08) [24] addressed to development of preventive safety applications and technologies. Within the PReVENT Integrated Project, the subproject Willwarn, which we have mentioned before, focused on the topic of vehicle-to-vehicle and vehicle-to-infrastructure communication.

The Car2Car Communication Consortium [19] is a non-profit organization initiated by European six vehicle manufacturers (General Motors, Ford Motor, Honda, Toyota, BMW and Mercedes-Benz) in 2004, pushing for further increase of road traffic safety. Its mission it to create an open European industry standard for Car2Car communication systems based on wireless LAN components and guarantee European-wide inter-vehicle operability. That includes proposing of realistic deployment strategies and business models to speed-up the market penetration.

The Car2Car consortium has established its objective of improving road safety and efficiently managing traffic using inter-vehicular systems. The Car2Car consortium together with IEEE have developed IEEE 802.11p standard, which defines enhancements to 802.11 required to support Intelligent Transportations Systems (ITS) applications. Car2Car main missions were as follows:

- The creation of an open European standard for V2V communications based on wireless LAN components.
- Developing V2V system prototype demonstrator for road safety applications.
- Promoting tree allocations of a free exclusive band for Car2Car applications in Europe.
- Developing deployment strategies and economic models for market penetration.

The FleetNet (Internet on the road) project [48] is a German project introduced by a consortium of six manufactures (DaimlerChrysler AG, FHI FOKUS, NEC Europe Ltd., Robert Bosch GmbH, Siemens AG, and Temic Speech Dialog Systems GmbH) and the universities of Braunschweig, Hamburg-Harburg and Hannover. FleetNet’s objective was to develop a communications platform for vehicle networks, to implement a demonstrator, and to standardize the proposed solutions in order to ensure better security and comfort for driver and passengers. The FleetNet architecture is based on a routing mechanism based on a system of location and navigation, and also considers vehicle to infrastructure communications in order to provide Internet access service.
The **NOW** (Network on Wheels) project [20] is a German project from the Federal education and research government, founded by automobile manufacturers, telecommunications operators and academia. NOW is the successor of FleetNet Project and supports and strongly cooperates with the Car2Car consortium. One of NOW’s main objectives is the implementation of communication protocols and data security algorithms in vehicular network. Considering the wireless 802.11 technologies and location-based routing in a V2V or vehicle to infrastructure communication context, the goal is to implement a system of reference and to contribute to the standardization of such a solution in Europe in collaboration with the Car2Car consortium.

**GST** (Global System of Telematics) [50] is an EU-funded Integrated Project that is creating an open and standardized end-to-end architecture for automotive telematics services. Participants were major car manufacturer and major Telecom players, altogether around 60 companies. The purpose of GST is to create an environment in which innovative telematics services can be developed and delivered cost-effectively and hence to increase the range of economic telematics services available to manufacturers and consumers. GST has introduced a seven service-oriented sub-projects that seek to contribute to achieving the main targets: rescue, safety, payment, safety channel or extended floating car data.

In **Japan**, a standard for vehicle-to-infrastructure communication was published in 2001 and denoted as ‘Dedicated Short-Range Communication System’ (ARIB 2001). The specified system operates in the 5.8 GHz frequency band, is based on time division multiple access and targets a range of about 30 m. The primary use of the system was seen in electronic toll collection but the system was generalized to support various other services. In 2008, more than 20 million on-board units for electronic toll collection were deployed in Japan. Based on the success on this 5.8 GHz DSRC system and on infrared-based vehicle-to-infrastructure communication, various ITS projects and activities are currently joining forces to demonstrate and enhance vehicle-to-infrastructure and vehicle-to-vehicle communication under the umbrella of Japan’s national ITS Safety 2010 initiative.
In the USA, there are several industry/government projects on-going. Vehicle Infrastructure Integration (VII), which has been rebranded as IntelliDrive [25], has recently completed a large proof of concept demonstration. The majority of this testing environment was implemented in Detroit. This system comprised 55 Road Side Equipment (RSE) stations within 45 square miles and employed 27 vehicles. Seven applications were developed and tested:

- In-Vehicle Signage: RSEs trigger displays of advisory messages within the vehicle.
- Probe Data Collection: Vehicles provide historical data on their location/state and share with the RSE, which is then centrally compiled and analyzed.
- Electronic Payments – Tolling.
- Electronic Payments – Parking.
- Heartbeat: RSEs collect periodic status messages from vehicles including vehicle speed and location.
- Traffic Signal Indication: Broadcasts traffic light state.

Integrated Vehicle-Based Safety Systems (IVBSS) project [26] explores human-machine interface issues when several safety applications, with potentially overlapping or contradictory advisories, are operated simultaneously. The Cooperative Intersection Collision Avoidance System (CICAS) project [27], had three components: a Violation Warning project (demonstrated in Michigan), a Stop Sign Assist project (demonstrated in Minnesota), and a Signalized Left Turn Assist project (demonstrated in California).

### 2.2.2.3.2. Vehicular Mobility Models

Due to the prohibitive cost of deploying and implementing such a system in real world, most research in VANET relies on simulations for evaluation. A key component for VANET simulations is a realistic vehicular mobility model. Mobility models represent real world scenarios for vehicular ad hoc networks and play a vital role in the performance evaluation of routing protocols. More research focus is now on the development of realistic mobility models for VANETs. A number of mobility models have been presented and their impact on the performance on the routing protocols has been tested. To get accurate results, the model should be as realistic as possible, and involve road maps with all the constraints and facilities related to the vehicular movement. Below, we present some new mobility models specifically proposed for VANETS:

The Integrated Mobility Model (IMM) [3] is an integration of Manhattan mobility model, freeway mobility model, stop sign model, traffic signs model and some other characteristics like stationary nodes. The advantage of IMM is that it provides a more detailed scenario for the simulation of VANETs by representing both the rural and urban area which is clear from the simulation results. After simulate with three different routing protocols (AODV, DSR and OLSR) and compare results, they obtained that OLSR and AODV performs better than DSR in a
more stressed urban scenario. The future dimensions of this work are that they will add more realistic parameters to IMM and will enhance it for VANETs simulations for more comprehensive results. The code, available at [3], has been developed by M. Alam, M. Sher and S. A. Husain at the University of Islamabad, Pakistan.

**MEtropolitan TAxis (META)** [4] is a mobility model, proposed in the Shanghai Jiao Tong University, in collaboration with the State University of New York at Buffalo, that can be used to generate synthetics trace for the movement of taxis in an urban area. In order to characterize the regularity of taxi movement, the authors designed three model parameters: turn probability, road section speed and travel pattern. Through different validation results, they show that META has a good approximation to a real scenario which in turn shows the effectiveness of these parameters. Based on the validation, synthetic traces can be generated; they are similar to the reality using such parameters. Since these parameters are easier to be obtained than the real trace, the META model can be used to replace the high cost real trace on some extent in other VANET researches.

**MObility model generator for VEhicular networks (MOVE)** [5] is a tool that allows users to rapidly generate realistic mobility models for VANET simulations. The output of MOVE is a realistic mobility model and can be immediately used by popular network simulators. The authors warn that if simple mobility models are used for evaluation of VANET, the results might not be as close to reality as expected, so they show that the details of a mobility model such as the existence of traffic lights, driver route choice and car overtaking behaviour can have a significant impact on the simulation results.

**Bonnmotion** is Java software, available at [28], which creates and analyses mobility scenarios. It is developed within the Communication Systems group at the Institute of Computer Science 4 of the University of Bonn, Germany, where it serves as a tool for the investigation of mobile ad hoc network characteristics. The scenarios can also be exported for the network simulators NS-2, GloMoSim/QualNet, COOJA, MiXiM, ONE and NCTUns, using a conversion script developed in the UPC (Polytechnic University of Catalonia) by Guillermo Diaz. Several mobility models are supported (Random Waypoint, Random Walk, Gauss-Markov, Manhattan Grid, etc.).

**CityMob** [29] is a mobility pattern generator for VANETs, designed to be used with the NS-2 simulator developed in the Polytechnic University of Valencia. CityMob can generate traces for VANETs scenarios using three different mobility models: Simple, Manhattan, and Downtown. CityMob is the main mobility generator in this project. For further information, refer to Chapter 4.
2.2.2.3.3. Other research areas in VANETs

Apart from mobility models, VANET research covers other areas, as transportation systems, routing protocols and new infrastructures.

Intelligent transportation systems (ITSs) [15] covers different technologies as wireless communications, sensor networks, voice and data communication, real-time driving-assistant systems, etc. The interconnection of different networks, even in the case of the Internet, is one of the main difficulties that is delaying the wide spread of vehicular networks. Location-Aided Gateway Advertisement and Discovery (LAGAD) [15] is a scalable hybrid adaptive protocol that aims to reduce congestion on single channels through a channel diversity mechanism that uses multiple channels and multiple interfaces for the propagation of gateway requests and replies.

Bypass-AODV (Bypass Ad hoc On-demand Distance Vector) [16] is a new optimization of the AODV routing protocol for mobile ad hoc networks proposed as a local recovery mechanism to enhance its performance. It uses a specific strategy of cross-layer MAC-notification to identify mobility-related packet loss, and then setup up a bypass between the node at which the route failure occurred and this node’s previous successor via an alternative node. Simulation results show that Bypass-AODV is insensitive to any random mobility model used and has a clear performance improvement compared to AODV. It has a comparable performance under group mobility model compared to AODV. Currently, Bypass-AODV is not suitable for VANET applications because the movement of vehicles is constrained by the layouts of the roads. As a future work, Bypass-AODV needs more improvement in order to handle VANET applications.

With the help of VANETs, vehicles on the road can form wireless ad hoc mesh networks (VMeshs) [17]. These meshes can help to retain certain transient information in a specific region for a period of time, by cooperatively passing the information among them without any infrastructure help. Nevertheless, there is a VANET storage problem. Studies show that the transmission range has high impact on the storage lifetime for one-way highway traffic, and the size of the region in which we want the information stored has high impact for two-way highway traffic.

A new two-tier architecture called Mobile Infrastructure Based VANET (MI-VANET) [18] has been recently proposed. In this architecture, the buses constitute a mobile backbone for data delivery while the low tier is composed of ordinary cars and passengers. MI-VANET will not only bring the benefit that ordinary cars do not have to forward packets for other nodes, but also improve the network connectivity. They are currently working and studying to demonstrate that MI-VANET with MIRT performs much better than GPSR and VANET MIRT in terms of packet delivery ratio and throughput.
2.3. Vehicular Traffic Models in VANETs

Transportation and traffic research area classifies traffic models according to the granularity with which traffic flows are examined. Macroscopic models model traffic at a large scale, treating traffic like a liquid applying hydrodynamic flow theory to vehicle behavior. The simulation in a macroscopic model takes place on a section-by-section basis rather than by tracking individual vehicles. However, they do not have the ability to analyze transportation improvements in as much details as the microscopic models.

Mesoscopic models combine the properties of both microscopic and macroscopic simulation models. As in microscopic models, the mesosocopic models unit of traffic flow is the individual vehicle. However, their movements follow the approach of the macroscopic model and are governed by the average speed on the travel link, so movements do not consider individual dynamic vehicle speed and volume relationships.

Microscopic simulations, which model the behavior of single vehicles and the interactions between them, are the most appropriate mobility models for simulating VANETs. Transportation and traffic science has developed a number of microsimulation models, each taking a dedicated approach ranging from coarse to fine grain.

When dealing with vehicular mobility modeling, some authors have distinguished between macro-mobility and micro-mobility. For macro-mobility they refer to all the macroscopic aspects which influence vehicular traffic, for example: the road topology, constrained car movements, the per-road speed limits. Micro-mobility refers instead to the drivers' individual behavior when interacting with other drivers or with the road infrastructure, for instance, traveling speed under different traffic conditions, acceleration, deceleration and overtaking criteria. For a trustworthy VANET simulation that both macro-mobility and micro-mobility descriptions are jointly considered when modeling vehicular movements.

There are some models proposed in the general MANET context usable in VANET. The criteria of applicability considered are the employment of road maps, and limiting the nodes movements into the roads instead of moving in a wide area. The considered parameters differ from a model to another. For instance, some models use traffic control mechanisms at intersections, and some others just assume continuous movement at these points. Some assume roads to be a single-lane, but some others support multi-lanes roads. Some define the security distance, while others just ignore this parameter. In the following is going to be described the main features of the most spread vehicular traffic models.

In the following, the main features of the most spread vehicular traffic models are described.

2.3.1. Freeway model

Freeway [51] is a generated-map-based model, defined in the simulation area, represented by a generated map, includes many freeways, each side of which is composed of many lanes. No urban roads, thus no intersections are considered in this model. At the beginning of the
simulation the nodes are randomly placed on the lanes, and they move using history-based speeds, where the speed of each vehicle smoothly changes following a random acceleration. In addition to the realism related to the acceleration and the history-based speed, the model defines a security distance that should be maintained between two subsequent vehicles in a lane. If the distance between two vehicles is less than this required distance, the second one decelerates to enable the forward vehicle moving away. The change of lanes is not allowed in this model. The vehicle moves on the lane it is placed in until reaching the simulation area limit, and then it is placed again randomly in another position and repeats the process.

2.3.2. Manhattan model

Manhattan model [51] is also a generated-map-based model introduced to simulate an urban environment.

Before starting a simulation a map with vertical and horizontal roads are generated. Each road latter includes two lanes, allowing the movement in the two directions (north/south for the vertical roads and east/west for the horizontal ones) at the beginning of a simulation; vehicles are randomly put on the roads. Afterwards, they move continuously according to history-based speeds (exactly like Freeway). When reaching a crossroads, the vehicle randomly chooses a direction to follow. That is, continuing straight forward, turning left or turning right. The probability of each decision is set by the authors respectively to 0.5, 0.25 and 0.25. The security distance is also used in this model and nodes follow the same strategy as in the freeway model to maintain this distance. But contrary to the previous model, a vehicle can change a lane at a crossroads. Nonetheless, there is no control mechanism at these points (crossroads), where nodes continue their movements without stopping.

2.3.3. City Section Mobility (CSM)

CSM [52] can be viewed as a hybrid model between Random Waypoint Model (RWP), in which mobile nodes move randomly and freely without restrictions, and Manhattan as it introduces the principle of RWP, especially the pause-time and random selection destination, within a generated-map-based urban area. At each step of the vehicle’s movement a random point is selected from the generated road map, towards which it moves following the shortest path. After reaching that destination, it remains there for a pause-time, and then repeats the process. The speed of nodes is constrained by the security distance, along with the maximum speed limit of the road.

2.3.4. Stop Sign Model (SSM)

Contrary to the previous models, SSM [53] integrates a traffic control mechanism. In every crossroads, a stop signal is set, which obliges vehicles to slow down and make a pause there. This model is based on real maps of the TIGER/Lines database [54], but all roads are assigned a single lane in each direction. A vehicle should never overtake its successor (like in all the models presented before) and should tune its speed to keep the security distance. If many vehicles arrive at an intersection at the same time, they make a queue, and each one waits for
its successor to traverse the crossroads. This results in gathering of nodes, and hugely affects the network connectivity as well as the mobility (average speeds). According to the authors, the problem with this model is the unrealistic disposition of the spot signals, since it is impossible to find a region with spot signals at each intersection, therefore, they improved SSM and they proposed TSM.

2.3.5. Traffic Sign Model (TSM)

In TSM model [53], stop signals are replaced by traffic lights. A vehicle stops at crossroads if it encounters a red stoplight; otherwise it continues its movement. When the first vehicle reaches the intersection, the light is randomly turned red with probability $p$ (thus turned green with probability $1-p$). If it turns red, then it remains so for a random delay (pause-time) forcing the vehicle to stop, as well as the ones behind it. After the delay, it turns red, and then the nodes traverse the crossroads one after the other until the queue is empty. When the next vehicle arrives at the crossroads the process is repeated.

2.3.6. STRAW (Street Random Waypoint)

STRAW [55] is also a model using real maps of TIGER/Line [54]. Like the other models, except freeway, roads include one lane in each direction and it is divided into segments. The model is basically composed of three modules: intra-segment mobility manager, inter-segment mobility manager, and finally the rout management and execution module. At the beginning of the simulation the nodes are placed randomly one behind the other, they move using the car following and try to accelerate until reaching the maximum speed of the segment. The first module manages this movement until reaching an intersection. The security distance is maintained, but the overtaking is not allowed. At crossroads the vehicles always slow down, even when they change a segment and turn without a full stop, which is realistic. The second module defines the traffic control mechanism including both stop signals and traffic lights, which are put on crossroads according to the class of the intersected roads. In addition to this usual control form, the module makes sure that the next segment to take contains enough available space before moving the vehicle towards it. If it is fully busy, the vehicle waits at the crossroads (at the end of the first segment). The last module selects the routes to be taken by each vehicle during the simulation. In the first one the direction is randomly selected at each intersection, for example, when reaching an intersection, the vehicle randomly decides whether to continue straight forward or to turn and change the road. On the other hand in the second approach a destination is selected toward which the vehicle moves using the shortest path.
2.3.7. MOVE (MObility model generator for VEhicular network)

MOVE [56] is a VANETs mobility model that uses the compiler SUMO [57], which is a realistic vehicular traffics simulation model. SUMO is an open source application implemented with java that integrates many realistic parameters such as realistic accelerations; the usage of real maps reflecting several types of roads (with multiple lanes), and traffic lights defining priorities between vehicles. Basically, MOVE is composed of two components; the road map editor and the vehicle movement editor. The former serves to manually and randomly generate a road map, either from TIGER/Line [54] files or Google earth files, whereas the latter allows specifying the properties of each vehicle like the maximum speed, the acceleration, the probability of turning at crossroads and the path to take. The information collected by the two editors is sent to the SUMO compiler.
2.3.8. BonnMotion

BonnMotion tool [28] implement several random mobility models, plus the Manhattan model. While the important tool includes the Car Following Model which is a basic car-to-car inter-distance control schema, the BonnMotion does not consider any micro-mobility. When related to the framework, we can easily see that the structure of both tools is definitely too simple to represent realistic motions, as they only model basic motion constraints and hardly no micro-mobility.

2.3.9. VanetMobiSim

VanetMobiSim [58] is an extension of the CANU Mobility Simulation Environment (CanuMobiSim) [59], which focuses on vehicular mobility, and features realistic automotive motion models at both macroscopic and microscopic levels. At the macroscopic level, VanetMobiSim can import maps from TIGER/Line database, or randomly generate them. VanetMobiSim adds support for multi-lane roads, separate directional flows, differentiated speed constraints and traffic signs at intersections. At the microscopic level, it manages lane changes and vehicle accelerations and decelerations, providing realistic car-to-car and car-to-infrastructure interactions.
2.3.10. SUMO (Simulation of Urban Mobility)

SUMO [60] is an open source, highly portable, microscopic road traffic simulation package designed to handle large road networks. Its main features include collision free vehicle movement, different vehicle types, single-vehicle routing, multi-lane streets with lane changing, junction-based right-of-way rules, hierarchy of junction types, an openGL graphical user interface (GUI), and dynamic routing. SUMO can manage large environments, i.e., 10 000 streets. SUMO can simulate traffic in different locations of the globe. However, since SUMO is a pure traffic generator, its generated traces cannot be directly used by the available network simulators, which is a serious shortcoming.

2.3.11. FreeSim

FreeSim [61] is a fully customizable macroscopic and microscopic free-flow traffic simulator that allows for multiple freeway systems to be easily represented and loaded into the simulator as a graph data structure with edge weights determined by the current speeds.
Traffic and graph algorithms can be created and executed for the entire network or for individual vehicles or nodes, and the traffic data used by the simulator can be user generated or be converted from real-time data gathered by a transportation organization. Vehicles in FreeSim can communicate with the system monitoring the traffic on the freeways, which makes FreeSim ideal for Intelligent Transportation System (ITS) simulation. FreeSim is licensed under the GNU General Public License, and the source code is available freely for download.

### 2.3.12. CityMob

CityMob [62] is a NS-2 compatible mobility model generator proposed for use in VANETs. CityMob implements three different mobility models: (a) Simple Model (SM), (b) Manhattan Model (MM), and (c) realistic Downtown Model (DM).

For further information about CityMob, refer to Chapter 4.

Last models are programs which includes the models explained at the beginning of the chapter. To summarize all the information there is a table which shows the main characteristics of the last seven models, which are the most used.

<table>
<thead>
<tr>
<th></th>
<th>VanetMobiSim</th>
<th>SUMO</th>
<th>MOVE</th>
<th>STRAW</th>
<th>Freesim</th>
<th>CityMob</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maps</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>User-defined</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manhattan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>Mobility</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random Waypoint</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>Traffic Models</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multilane roads</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Lane changing</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Overtaking criteria</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collision free movement</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Different vehicles type</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>Traces</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-2 trace support</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NCTUns trace support</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Import different formats</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1. Comparison of Vehicular Traffic Models
As it can be seen none of these seven models have NCTUns compatibility but some have NS-2 trace support, which it is the most usual compatibility with all VANET simulators.
3. TRANSLATOR PROGRAM. ADAPTING SIMULATORS’ OUTPUT TO NCTUNS.

3.1. DESCRIPTION OF THE TRANSLATOR PROGRAM.

Translator_NS2_NCTUNS is a program that translates an NS-2 file into an NCTUns file. It is programmed in C over Linux, which means it is an open source program. This program has 3 main parts: Read, Translate and Write. Before running the program is needed to have in the same directory the NS-2 file is going to be translated with the name “prueba.txt”. After using the program you will have an NCTUns file called “salida.mdt” which could be directly loaded at NCTUns.

The complete source code of the translator could be consulted on the Annex 1.

3.1.1. Main

Before describing the three main parts it is important to know how works the main function to have a global vision of the program.

Firstly the main function creates the structures necessary to do the translation; those are the sNS2 structure and NCTUns structure.

```c
int main(int argc, char** argv) {
    int fd,fd2,n,x,y;
    NS2 sNS2;
    NCTUns sNCTUns;
```

The sNS2 structure and sNCTUns structure have these properties:
TRANSLATOR PROGRAM: ADAPTING SIMULATORS’ OUTPUT TO NCTUns.

- **NCTUns structure**

  ```c
  typedef struct{
    int x;
    int y;
  } Tsize;

  typedef struct{
    int x;
    int y;
    float arrivalTime;
    float pausedTime;
    float speed;
  } Location;

  typedef struct{
    int numberTurnings;
    Location locationNC[200];
  } Tnode;

  typedef struct{
    Tsize size;
    int numberNodes;
    Tnode node[200];
  } NCTUns;
  ```

- **NS2 structure**

  ```c
  typedef struct{
    float x;
    float y;
    float time;
    float speed;
  } NLocation;

  typedef struct{
    int numberLocations;
    NLocation locationNS[200];
  } Nnode;

  typedef struct{
    int numberNodes;
    Nnode node[200];
  } NS2;
  ```

After creating the structures the program opens the NS-2 file, it must be called “prueba.txt”, and it creates, or opens if it is already created, a file called “salida.mdt” were is going to write the NCTUns information.

```c
fd=open("prueba.txt", O_RDONLY);
if (fd==-1){
```

49

UPC

Telecom Bcn
printf("Error charging NS2 file. Remember it must be called prueba.txt\n");
return;
}

fd2=fopen("salida.mdt", "w");
if (fd2==-1){
    printf("Error creating or charging NCTUns file. Try to create a file calling it salida.mdt");
    return;
}

After that step it calls the function Read with two parameters: file descriptor were "prueba.txt" had been opened and a pointer to the sNS2 struct. This function reads the information in an NS-2 file and it stores this information in a NS-2 struct. It will be more specifically described later.

read(fd, &sNS2);

Then the program closes the file descriptor related to "prueba.txt" and it sets some parameters that are needed to do the translation and they are not obtained in the NS-2 file. The first information needed is the number of nodes and it is obtained doing and easy calculation. It is the number of nodes detected in the function "lee" plus one. Secondly it is settled the dimension of the simulation area. Although the area probably is defined in the NS-2 file it is impossible to be collected so the program set the maxim that allows NCTUns. And after these lines the function Translate is called.

close(fd);
n=sNS2.numberNodes + 1;
x=3000;
y=3000;

The function Translate needs both structures, sNS2 and sNCTUns, pointers and also the number of nodes and the area described previously. This function does the calculations to pass the information stored in the NS-2 structure to the NCTUns structure. It will be more explicitly described later.

translate(&sNS2,&sNCTUns,n,x,y);

The last function called is Write. This function needs the file descriptor related to the file "salida.mdt" and the pointer to the struct sNCTUns. This function writes the information from the structure NCTUns in the NCTUns file. It will be more described later.

write(fd2,&sNCTUns);

And finally the program closes the file descriptor related to the NCTUns file and finishes the program.
close(fd2);
    return (EXIT_SUCCESS);
}

3.1.2. Read

The function Read has to read the NS-2 file and store the information in a NS-2 struct. The NS-2 file has been previously opened. This function needs as parameters the file descriptor related to the NS-2 file and the pointer to the NS-2 struct.

An NS-2 file has three kind of lines:

$node_.(149) set X_ 0.000000000000
#Node 0 had an ACCIDENT:
$ns_ at 0.000000000000 "$node_.(0) setdest 900.000000000000 800.000000000000 0.000000000000"

The first line is a set line where it is described the coordinate X, Y or Z position. The second line is a typical information line it is not important for the function translation but in a NS2 file provides some information about the next line. In this case is an advice that node 0 has an accident and the next line describes this accident. The third line is the usual NS2 line, the first parameter is the time where the action starts, then sets a destination, which coordinates are the two next numbers and finally it sets the speed that the node will have to reach the destination. On this line the speed is 0 because it is describing an accident and the new destination is the same as the previous position. This function is based in scenarios which only has X and Y position in the third type of line.

The function Read is helped by another function called ReadNumber. This function reads from a file in a specified position until the end of the word and translates the information into a float to be used by the Read function.

```c
float ReadNumber(int fd, char *s){
    int i,b;
    char a;
    float n;
    i=0;
    b=1;
    while(b==1){
        read(fd,&s[i],1);
        a=s[i];
        if(a==')') b=0;
        if(a== ' ') b=0;
        if(a==10) b=0;
        if(a=='\') b=0;
        else i++;
    }
    s[i]=\'\0\';
    n=strtof(s);
    return n;
}
```
The first thing that Read does it is read the first character from a line. If this character is “#” the line is not important as it has been show previously so the function continues reading until find a end of line character, “\n”.

If the function reads “$” it began to analyze the line. The difference between a set line and a common line is the third character. If the program reads an “o” it is a set line, if it reads an “s” it is a common line.

In case of reading and “o” character the program looks the node number.

```c
if(l=='o'){
    lseek(fichNS2,4,SEEK_CUR);
    n=ReadNumber(fichNS2,num);
}
```

After finding the node number the function enters in a loop for reading character by character until finding the coordinate X or Y, Z coordinate is not saved, and store the information in the right place on the NS2 struct.

```c
while(1){
    read(fichNS2,&l,1);
    if(l=='X'){
        lseek(fichNS2,2,SEEK_CUR);
        sNS2->node[n].locationNS[0].x=leeNumero(fichNS2,info);
        break;
    }
    else if(l=='Y'){
        lseek(fichNS2,2,SEEK_CUR);
        sNS2->node[n].locationNS[0].y=leeNumero(fichNS2,info);
        break;
    }
    else if(l==10){ break;}
}
```

And finally it checks that the number of node read at beginning is superior to the highest node number stored. This is the best way to know the number of nodes that there are on the NS-2 file.

```c
if(n>sNS2->numberNodes) sNS2->numberNodes=n;
```

Otherwise if the function reads “s”, which means that it is a common line, it begins to read the time, which is the first important value in the line.

```c
else if(l=='s'){
    lseek(fichNS2,5,SEEK_CUR);
    time=ReadNumber(fichNS2,info);
}
```
And as it has done before the program enter in a loop to read next information. Firstly read the node number. After that it increases the number of turnings in the NS2 struct file and copy the time read previously in the right place. Then the program reads the coordinates and the speed and store the information.

```c
n=ReadNumber(fichNS2,num);
sNS2->node[n].numberLocations++;

sNS2->node[n].locationNS[sNS2->node[n].numberLocations].time=time;
lseek(fichNS2,9,SEEK_CUR);

sNS2->node[n].locationNS[sNS2->node[n].numberLocations].x=leeNumero(fichNS2,info);

sNS2->node[n].locationNS[sNS2->node[n].numberLocations].y=leeNumero(fichNS2,info);

sNS2->node[n].locationNS[sNS2->node[n].numberLocations].speed=leeNumero(fichNS2,info);
```

The function Read is a simple function that can be improved reading the Z coordinate but for our use it is not important because NCTUns only works with X and Y coordinates. Also can be improved changing loops for lseek instructions but the distance between different information is not defined and we decided to read the information in a more secure way.

### 3.1.3. Translate

This function makes the calculations for the translation of the NS2 structure to NCTUns. It needs two pointers one for a structure NS2 and another for a structure NCTUns. Also needs the number of nodes and the size of the working area.

```c
void translate(NS2 *sNS2, NCTUns *sNCTUns, int n, int x, int y){

    for(i=0;i<sNCTUns->numberNodes;i++){
        sNCTUns->node[i].numberTurnings=sNS2->node[i].numberLocations-1;
    }

    The new loop is created, this deals with the number of turnings. Firstly it sets the coordinates X and Y and then it looks if this is the initial turning which has a different treatment. In this case the arrival time is 0 and the paused time is the time of the next NS2 node.

    for(j=0;j<=sNCTUns->node[i].numberTurnings;j++){
```
sNCTUns->node[i].locationNC[j].x=sNS2->node[i].locationNS[j].x;
sNCTUns->node[i].locationNC[j].y=sNS2->node[i].locationNS[j].y;
if(j==0){
    sNCTUns->node[i].locationNC[j].arrivalTime=0;
    sNCTUns->node[i].locationNC[j].pausedTime=sNS2-
    node[i].locationNS[j+1].time;
}

If it is not the initial turning it starts calculating the distance between the previously turning
and the actual. This function is based in a Manhattan grid which means that only has
movement in one of the coordinates, X or Y, never in both. This simplifies the calculus of
possible accidents and lights signals. The existence of a possible accident is calculate with a
comparative between the arrival time in the next turning in the NS2 file and a time previously
calculated with the distance and the speed settled in the previous turning. It is added a margin
to avoid mistakes in case of irrelevant differences between both times.

dist=sNS2->node[i].locationNS[j].x-sNS2->node[i].locationNS[j-1].x;
    if(dist==0){
        dist=sNS2->node[i].locationNS[j].y-sNS2->node[i].locationNS[j-
        1].y;
    }
    if(dist<0) dist=-dist;
    if(((dist/sNS2->node[i].locationNS[j].speed)+0.5)<((sNS2-
        >node[i].locationNS[j+1].time)-(sNS2-
        >node[i].locationNS[j].time)))
        semaforo=1;
    else
        semaforo=0;

If the program detects a light signal the arrival time will be calculated. The arrival time will be
the arrival time in the previous turning plus a time calculated by the distance divided by the
speed settled in the previous turning. The paused time is the time of the next turning
calculated by subtracting the earlier arrival time.

sNCTUns->node[i].locationNC[j].arrivalTime=(dist/sNS2-
    >node[i].locationNS[j+1].speed)+sNS2->node[i].locationNS[j].time;
sNCTUns->node[i].locationNC[j].pausedTime=sNS2->node[i].locationNS[j+1].time-
sNCTUns->node[i].locationNC[j].arrivalTime;

In case of no light signal the program sets the arrival time as the time of the next turning and
the paused time is 0.

sNCTUns->node[i].locationNC[j].arrivalTime=sNS2->node[i].locationNS[j+1].time;
sNCTUns->node[i].locationNC[j].pausedTime=0;

Finally the last part of the turning loop the program detects possible mistakes in the paused
time. Sometimes due a difference of calculus this value is negative. This negative value makes
the NCTUns file useless and NCTUns simulator does not charge the file. For not having this problem the function looks for this possible value and changes it to 0. Most of the time this value is very low so change it to 0 does not affect the final result.

```c
if(sNCTUns->node[i].locationNC[j].pausedTime<0){
    sNCTUns->node[i].locationNC[j].pausedTime=0;
}
```

And the final line sets the speed.

```c
sNCTUns->node[i].locationNC[j].speed=sNS2 >node[i].locationNS[j+1].speed;
```

This function does not deal with the final turning. To translate an NS-2 file into a NCTUns file the information of the next turning is necessary but when it is in the final turning this information does not exist. So the final turning in an NS-2 file does not have a translation into a NCTUns file.

To simplify the comprehension of the process done by this function the next table explains what is done.

<table>
<thead>
<tr>
<th>NCTUns struct</th>
<th>Initial location</th>
<th>Rest of locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate X</td>
<td>sNS2-&gt;node[i].locationNS[j].x</td>
<td>sNS2-&gt;node[i].locationNS[j].x</td>
</tr>
<tr>
<td>Coordinate Y</td>
<td>sNS2-&gt;node[i].locationNS[j].y</td>
<td>sNS2-&gt;node[i].locationNS[j].y</td>
</tr>
<tr>
<td>Arrival Time</td>
<td>0</td>
<td>(dist/sNS2-&gt;node[i].locationNS[j+1].speed)+sNS2-&gt;node[i].locationNS[j].time</td>
</tr>
<tr>
<td>Paused Time</td>
<td>sNS2-&gt;node[i].locationNS[j+1].time</td>
<td>sNS2-&gt;node[i].locationNS[j+1].time</td>
</tr>
<tr>
<td>Speed</td>
<td>sNS2-&gt;node[i].locationNS[j+1].speed</td>
<td>sNS2-&gt;node[i].locationNS[j+1].speed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Light Signal</th>
<th>No light signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>sNS2-&gt;node[i].locationNS[j].x</td>
<td>sNS2-&gt;node[i].locationNS[j+1].speed</td>
</tr>
<tr>
<td>sNS2-&gt;node[i].locationNS[j].y</td>
<td>sNS2-&gt;node[i].locationNS[j+1].speed</td>
</tr>
<tr>
<td>(dist/sNS2-&gt;node[i].locationNS[j+1].speed)+sNS2-&gt;node[i].locationNS[j].time</td>
<td>sNS2-&gt;node[i].locationNS[j+1].time</td>
</tr>
<tr>
<td>sNS2-&gt;node[i].locationNS[j+1].time</td>
<td>sNS2-&gt;node[i].locationNS[j+1].time</td>
</tr>
<tr>
<td>sNS2-&gt;node[i].locationNS[j+1].speed</td>
<td>sNS2-&gt;node[i].locationNS[j+1].speed</td>
</tr>
</tbody>
</table>

Table 3.1. Title

### 3.1.4. Write

The function write writes the information stored in the NCTUns struct in a file. This function needs a pointer to an NCTUns struct and the file descriptor related to “salida.mdt” which is going to be the NCTUns file.

```c
void write(int fichNCTUns, NCTUns *sNCTUns)
```
The first line to write is a commentary line and the second is the size of the working area. Coordinates X and Y are stored in the NCTUns struct. Then the program writes the number of nodes preceded by another commentary line.

```c
fprintf(fichNCTUns,"# The size of the working area.\n");
fprintf(fichNCTUns,"%d %d 1000\n",sNCTUns->size.x,sNCTUns->size.y);
fprintf(fichNCTUns,"# Number of mobile nodes\n");
fprintf(fichNCTUns,"%d\n",sNCTUns->numberNodes);
```

After the initial lines the function creates a loop controlled by the number of nodes. Inside the loop the first line is the line that describes the type of node. By definition the program sets a 802.11b ADHOC node. Then it writes two commentary lines. The first one defines the number of node and the second explains the information it is going to be written below.

```c
fprintf(fichNCTUns,"NODE_BEGIN ID_NODE_80211B_ADHOC\n");
fprintf(fichNCTUns,"# Node %d\n# initial location x(m) y(m) arrival_time(s) pause_time(s) speed(m/s)\n",i);
```

The next line to write is the line that has the information about the initial position. It has the coordinates X and Y, arrival time which is always 0, paused time and speed. All the numbers, except coordinates which are integer, are floats and they are showed with nine decimals.

```c
fprintf(fichNCTUns,"%d %d %.9f %.9f %.9f
",sNCTUns-
>node[i].locationNC[0].x,sNCTUns->node[i].locationNC[0].y,sNCTUns-
>node[i].locationNC[0].arrivalTime,sNCTUns-
>node[i].locationNC[0].pausedTime,sNCTUns->node[i].locationNC[0].speed);
```

Before it is written the turnings points the program writes a commentary line, a line which contains the number of turning points. The function checks if there is any turning point apart from the initial, this is the cause because there is a comparison of the number of turning points minus one with zero. And finally it writes the same commentary line written before the initial point.

```c
fprintf(fichNCTUns,"#       PATH_BEGIN number_of_turning_points\n");
if(sNCTUns->node[i].numberTurnings-1>0)
    fprintf(fichNCTUns," PATH_BEGIN %d\n",sNCTUns-
>node[i].numberTurnings-1);
else
    fprintf(fichNCTUns," PATH_BEGIN %d\n",0);
    fprintf(fichNCTUns,"#           x(m) y(m) arrival_time(s) pause_time(s) speed(m/s)\n");
```
Then the function creates a loop to write all turning points. The line written with any turning point is exactly the same that has been written with the initial point.

```c
for(j=1;j<sNCTUns->node[i].numberTurnings;j++){
    fprintf(fichNCTUns,"%d %d %.9f %.9f %.9f\n",sNCTUns->node[i].locationNC[j].x,sNCTUns->node[i].locationNC[j].y,sNCTUns->node[i].locationNC[j].arrivalTime,sNCTUns->node[i].locationNC[j].pausedTime,sNCTUns->node[i].locationNC[j].speed);
}
```

And finally after the turning loop the final node line is written and the final turning points line.

```c
fprintf(fichNCTUns,"PATH_END\n");
fprintf(fichNCTUns,"NODE_END\n");
```
3.2. **Executing Translator Program**

To translate an input file with an NS-2 format to NCTUns using the translator must proceed as follows:

In the first place, the NS-2 file should be renamed as `prueba.txt` and stored in the same directory where the translation program is saved. After that, the program can be executed typing in a terminal the following command:

```bash
./translator
```

Once the program has been executed, in the same directory will appear a new file called `salida.mdt`. This file is the translated version, compatible with NCTUns, of the original format of NS-2 and can be loaded into the network simulator.
4. SIMULATION ENVIRONMENTS AND SCENARIO

4.1. THE CITYMOB MOVEMENT SIMULATOR

In the following, we describe the CityMob movement simulator, selected by the team to generate the vehicular movement traces for the scenario.

4.1.1. Introducing CityMob

CityMob [29] is a mobility pattern generator for VANETs developed in the Polytechnic University of Valencia which allows researchers to easily create urban mobility scenarios implementing three different mobility models: Simple Model, Manhattan Model and Downtown Model. All streets are two-way, with lanes in both directions and node moves with random speed, within a user defined range of values.

It is designed for working with the NS-2 simulator, but, in our case, we developed a translator script (see Chapter 3) that makes compatible the output of this generator with the NCTUns simulator. CityMob makes it possible to model car accidents and to use a flooding-based protocol to alert or announce events, it has been especially designed to investigate different mobility models and their impact on vehicle communication performance.

In the process of representing realistic scenarios, CityMob proposes three different mobility models that with different level of randomness:

- **Simple Model**: Vertical and horizontal mobility patterns without direction changes or traffic lights.
- **Manhattan Model**: In this mobility model, the mobile nodes move in horizontal or vertical direction on a Manhattan style map. All streets are two-way with one lane in
each direction and car movements are constrained by these lanes. Nodes direction is randomly selected in every moment and they cannot be repeated in two consecutive movements. Moreover, this model simulates traffic lights at random positions (not only in crossroads) and with different delays.

- Downtown model: This model improves the Manhattan Model with traffic density not uniformly distributed; there are zones with a higher density acting as a downtown. In these areas vehicles must move more slowly than in the periphery. The downtown area can be selected and can never cover more than 90% of the total map area. There is a parameter, $p$, which can be used to establish the probability of a node being initially located inside the downtown area and also the probability that nodes on the outskirts move into the downtown. The remaining features are the same as for the Manhattan mobility model.

The distance between streets is configurable, limited by the map size, and there must be a minimum number of crossings to allow nodes to change their direction. User can configure the number of simulated nodes and the number of damaged nodes. Every node will take a random initial position, although in the Downtown model the probability of starting inside the downtown area is greater because of its higher density. Speed can vary according the map area, changing throughout the simulation, nodes will move with a random speed lower than the maximum one defined by the user.

### 4.1.2. Installation of the simulator

The CityMob simulator can be downloaded from the website [30] of the Networking Research Group (GRC) of the Technical University of Valencia (UPV).

The program, implemented in C language and distributed under a GNU/GPL license, can be used once downloaded and unzipped without more installation steps.

### 4.1.3. Executing a simulation

For generating a simulation of movement with CityMob, in the first place we have to execute the application followed by the required parameters in order to generate the desired trace file.

In a console or terminal, in the directory where CityMob simulator is placed, execute the following command:

```
./citymob -m M -n N -t T -s S -w W -h H -d D -a A -x xl -y yl -X Xl -Y Yl -p P
```

Where,

- **M**: Model
  - M=1: Simple Simulation Traffic Model
  - M=2: Manhattan Simulation Traffic Model
  - M=3: Downtown Simulation Traffic Model

- **N**: Nodes number
T: Simulation time
S: Max speed
W: Map width
H: Map Height
D: Streets distance
A: Accidents Number
x1: min X downtown (only for M=3)
y1: min Y downtown (only for M=3)
X1: max X downtown (only for M=3)
Y1: max Y downtown (only for M=3)
P: Probability of a node being initially located inside the downtown (only for M=3)
4.2. THE NCTUNS NETWORK SIMULATOR

In this chapter, it is going to be described the NCTUns simulator that has been the network simulator selected to study the performance of our scenario.

4.2.1. Introducing NCTUns. Computing environment used.

NCTUns (National Chiao Tung University Network Simulator) [31] is a network simulator and emulator. It is an open-source software running on Linux, Fedora 11, with an integrated GUI environment.

NCTUns provide important advantages over traditional networks simulators, like the using of the real-life TCP/IP protocol to conduct simulations providing more realistic and high-fidelity simulation results. The simulator integrates some simulation capabilities, such as designing maps and controlling vehicles mobility but, as we have seen before, we implement this mobility using external movement simulators.

NCTUns provide several advantages in front of other network simulators:

- Realistic network traffic can be generated by realistic life applications to generate more realistic simulation results.
- The performance of any real-life application can be easily evaluated on NCTUns under various simulated network conditions.
- Any network application program developed for NCTUns device can be directly run up on a real-life Linux device without any modification.

In its 5.0 release, the simulator provides a complete implementation of the emergent IEEE 802.11(p) and 1069 standards for wireless vehicular networks.

With two modes of operation, pre-specified and autopilot, parameters such as moving path and speed for each vehicle can be specified by the scenario designer in the first case and, in the second case, initial speed, maximum speed, initial acceleration, maximum deceleration, etc. can be specified. The autopilot mode selects the best route to navigate and is capable of performing lane changing, car following, turning, etc.

The commercial version of NCTUns, called EstiNet Network Simulator and Emulator, is “a software tool for network planning, testing, education, protocol development, and applications performance prediction. It is both a network simulator and emulator with worldwide customers and global impact.” [32]

In this project, we used the NCTUns 6.0 version in a virtual machine running Fedora 11. The employ of virtual machines allows us to avoid possible driver incompatibilities, work in a cleaning operating systems dedicated only for this purpose, run multiple parallel simulations, etc. Also, with virtual machines the assigned resources can be increased at need. We used the VMware tool for Mac OS X and Ubuntu Linux for the execution of the virtual machine developed by Daniel Navarro Giménez and Xavier Llàrio Frau in their Final Project [34].
4.2.2. Installation and execution of the simulator.

Due to the different machines used in the simulations (laptops from different brands, with different operating systems...) and in order to avoid possible incompatibilities with drivers, the team chose to work with virtual machines instead of working with a simulator installed in the native operating system.

The program chosen to load the virtual machine is the VMware Player, a free product developed by VMware that let us run virtual machines. It can be installed on Windows, Linux or Mac OS X.

In the first place, we have to download this software from its official website [35] and install it in our operating system. Once we downloaded and run the virtual machine in the VMware Player (we have to select the OS called NCTUns) we will be in front of a Fedora 11 operating system with everything needed to simulate scenarios and to run and use the network simulator, NCTUns.

Fig. 4.1. Fedora-NCTUns Screenshot

Now we are ready to start using NCTUns. The simulator comprises two components apart from the client, the coordinator and the dispatcher. These components have to be executed as root, and although it is not necessary to execute them at the same machine as the client, it is recommended. Once the dispatcher and coordinator are launched, the client can be started. So, in a console, as a root, we have to execute the following commands:

```
$ dispatcher &
$ coordinator &
$ nctunsclient &
```

The NCTUns is now ready to start working:
4.2.3. Design of an example VANET scenario in NCTUns 6.0.

NCTUns distinguished by the simplicity that offers when entering scenarios, through a file generated by an external program or through its graphical interface, which allows the introduction of roads with a variable number of lanes, intersections, vehicles, fixed antennas, walls...

As an example, we will design a simple scenario with the graphic editor in order to show how to set up a simulation. In later chapters we will see how, in this project, the nodes, their movements and the allocation of the streets are designed using an external movement generator, the CityMob. The steps to follow are described below:

1. "Draw Topology" mode:

   ![Fig. 4.3. Draw Topology Mode]

   In this mode we can, in the first place, draw the streets of our scenario. We select the icon "ITS road segment" (fig. 4.4.) and we will be able to draw the road to our liking on the blank area.

   ![Fig. 4.4. ITS Road Segment]
It is important to note that the road should be closed, or the vehicles will not move. The following figure shows an example of a valid scenario:

Crossroads can be drawn selecting the icon "ITS Crossroad":

Once we have drawn the streets we can begin to add vehicles, called RSUs (RoadSide Units) and OBUs (On Board Units). In this project, the RSUs are of the type “802.11(b) mobile node (ad hoc mode)” (Fig. 4.6.) and the OBUs are of the type “802.11(b) ad-hoc mode interface” (Fig. 4.7.).

Once the different nodes are placed, switching to the “Edit property” mode (Fig. 4.8.) we will be able to set the network parameters of each node.
2. “Edit property” mode.

![DERP](image)

Fig. 4.9. Edit Property Mode

Double clicking in a RSU or an OBU opens a menu called “Mobile station”. The “Path” tab allows adding movement to the node, in our case, the mobility model is generated by the CityMob simulator (Section 4.5.). In the same tab we can modify the protocol stack of the network interface by clicking on the “Node editor” button. We have to delete, using the X button the GOD block and replace it with the AODV block, which is located in the MROUTED tab, and reconnect the blocks with the arrow button. The protocol stack must stay as below:

![Protocol Stack](image)

Fig. 4.10. Protocol Stack in the Node Editor

If we want to copy this protocol stack to all the nodes of the simulation we have to accept and to push the C.P.A.N.S.T. button:

![Copy to 149 node(s)](image)

Fig. 4.11. C.P.A.N.S.T. button
In the “Application” tab we can configure the programs that will generate the traffic and the mobility model of the OBUs. NCTUns features different built-in applications, which use can be consulted by clicking in the “App. Usage” button. It is also possible to develop an application and communicate it with NCTUns; this process is described in [36] and [37]. In the simulations of this project the network traffic is generated by the stg command, which generates a constant bit-rate flux, the receiver uses the rtg applications.

For example,

```
stg -u 1000 100 1.0.1.24
```

It generates a constant bit-rate flux of 1000 bytes to address 1.0.1.24 during 100 s.

For the receiver (node 24 in this example) we have to use the following command:

```
rtg -u
```

We recommend, when we are modifying the protocol stack, to add to the first node this command and, using the C.P.A.N.S.T. button make that every node in the scenario be a possible receiver.

For setting parameters related to the physical layer we can use the physical layer editor:

For setting parameters related to the physical layer we can use the physical layer editor:

We can change the Fading Model from the “Propagation Channel Model” box. In this project we will study scenarios without and with Ricean and Rayleigh fading model. The “Path loss model” is “Two Ray Ground”. The “Node Connectivity Display” uses the receiving node perspective and the “Node Connectivity Determination” is determined by distance. For all the tests in this project, DTR is set to 100m and DIR to 180m for every OBUs. The rest of the parameters are set as below:
“Recalculate” button has to be pushed before Accept all this changes. As we have seen before, if we want to copy this parameters to all the nodes present in the simulation we have to push the C.P.A.N.S.T. button and choose the “Copy to all nodes of the same network type” option:

Opening “G_Setting” > “Simulation” menu allow editing the simulation parameters such as simulation time. This time is set to 100s for all the simulations:
The simulator measures time in tics. In the "Speed" tab we can configure the duration of one tic. Smaller durations will increase the simulation time.

3. "Run simulation" mode.

Once in this mode, the topology is saved and several files are generated. The simulation could be started by selecting the “Simulation” > “Run” option.
4.3. **Description of the Scenario Under Evaluation**

The considered scenario consists of a VANET deployed over an area of a city with square streets. In this way we are modeling a scenario similar to the Eixample district of Barcelona, with a central area with differs from the rest by the highest density of vehicles and an average speed lower than the outer areas. Nodes can be out of range when transmitting so the VANET has to rely on multi-hop to deliver the packages.

There are no base stations because only nodes (vehicles) will exchange information among themselves.

The scenario chosen for the simulations of this project has the following characteristics:

- Manhattan Downtown model
- 150 nodes (vehicles)
- Duration 100s
- Average speed: 15 m/s – 30m/s (50km/h – 100km/h)
- 1500x1500m with distance between streets of 100m
- Downtown: 500x500m centered on the scenario
- 5 accidents (an accident is modelled keeping a node static along the simulation):

The default probability of a node being initially located inside the downtown is set to 0.7 by default. The routing protocol used is AODV. The road features 2 lanes in each way.

This configuration of the scenario results in the following CityMob command for the “slow” simulations:

```
./citymob -m 3 -n 150 -t 100 -s 15 -w 1500 -h 1500 -d 100 -a 5 -x 500 -y 500 -X 1100 -Y 1100
```

And for “fast” simulations:

```
./citymob -m 3 -n 150 -t 100 -s 30 -w 1500 -h 1500 -d 100 -a 5 -x 500 -y 500 -X 1100 -Y 1100
```

Packet size, transmission range and the rest of the physical parameters will be set as described in the example of the previous chapter. It is important to mention that walls are added to the scenario, using the GUI of NCTUns, with an attenuation of 5dB according to the values studied by other two students, Leticia Lemus and Luis Felipe Urquiza, of our working group [63].

This example of scenario could be summarized with the following picture:
We could see later on, in the chapter 4.5., with the help of the NCTUns Network simulator, how this example is reflected.
4.4. **Generation and Translation of an NS-2 file to NCTUns**

This section explains how to generate a CityMob file containing the movement patterns of the 150 nodes that are present in the simulation and its subsequent translation by a program designed by the project team (chapter 3) and how to loading it in the NCTUns network simulator.

To generate the patterns of movement we have to use the command indicated in previous sections, unless otherwise indicated, the result is displayed on screen; as we are interested in storing the patterns in a file we must enter the following command in a terminal:

```
./citymob [PARAMETERS] > file
```

In our case the file must be named `prueba.txt` in order to be compatible with the translator program, so we will enter the following command with the parameters indicated in the previous chapter:

```
./citymob -m 3 -n 150 -t 100 -s 15 -w 1500 -h 1500 -d 100 -a 5 -x 500 -y 500 - X 1100 -Y 1100 > prueba.txt
```

Once generated the file, it should be copied to the folder where the translator is and entered the following command:

```
./traductor
```

It will generate a file called `salida.mdt` that contains the movement of the 150 nodes and can be loaded into the NCTUns proceeding as described in the following chapter.
4.5. **Simulating and Filtering the Traces**

Once we opened the NCTUns (see Section 4.2.2.), from the menu “G_Tools” > “Import Mobile Nodes and Their Paths From File” we can select the file *salida.mdt* obtained with the translator.

![Fig. 4.19. Import Mobile Nodes](image)

Once these steps have been completed we loaded into the simulator the 150 nodes and its movements through the streets of our scenario:

![Fig. 4.20. Scenario loaded on NCTUns](image)

Now we proceed as described in section 4.2.3. from the point 2. **“Edit property” mode** by choosing the type of fading (None, Rayleigh or Ricean) and which will be the sending and receiving nodes.
In our case, for the simulation of each scenario we have designed the following pattern of communication between nodes:

- 2 transmitting nodes from outside to outside through the downtown.
- 2 transmitting nodes within the downtown.
- 2 transmitting nodes outside-outside.
- 4 transmitting nodes outside-downtown.

The communication pattern can be summarized in the following picture:

![Communication Pattern Diagram](image)

After a variable period of time (6-9 hours approx.) our simulation will be finished presenting on the screen the following message:

![Simulation is done Message](image)

We will have to accept two dialog boxes, several files will be copied and, in the end, our simulation is finished.

After finishing a simulation, NCTUns generates a binary trace file, `Simulation_Name.ptr`, which the GUI client reads to do the replays. With the help of the printPtr application we can decode the binary file and translate it into a plain text file that can be treated more easily:

```
./printPtr trace_file.ptr > trace_text_file.txt
```
Traces have the next scheme:

802.11 RX 1503414258 2030 ACK <0 0> <133 149 149> 13248917 14 0 NONE 3

- The first column deals with the type of protocol of the message sent. In our simulations it is always 802.11.
- The second column is about the type of event. This can be RX (response), TX (transmission), RTX (retransmission) and DROP.
- The third column is the time of the event. In this case it is shown by number of ticks. Each tick represents 100ns, this value can be changed before the simulation in NCTUns.
- The fourth column is the duration of the event. It is calculated by the same way as the previous field.
- The fifth column is about the type of message:
  - DATA (802.3/802.11 Data packet)
  - RTS (802.11 RequestToSend packet)
  - CTS (802.11 ClearToSend packet)
  - ACK (802.11 Acknowledgement packet)
  - BCON (802.11 Beacon packet)
  ...
- The sixth column is the node IDs based on the IP address. In our case the value was always 0 for source and destination.
- The seventh column tells the node IDs based on the MAC address. According to the NCTUns manual the first value is the transmission node, the second value is the reception value and the third value is the final reception node, i.e., a packet sent by the node 27 which has to arrive to the node 42 has to pass before through the node 35, in this case the field will be <27 35 42>. Unfortunately we did not see any trace like this, except DROP packets, in that cases the program make 2 traces written like <27 35 35> and <35 42 42>.
- The eighth column is the ID of the packet.
- The ninth column is the size of the package. In our case we only use the traces with a package of 1070 bytes.
- The tenth column is about the number of RTX of this package has been done successive.
- The eleventh column is the cause of DROP
  - COLL (collision)
  - CAP (capture)
  - DUPX (duplicate)
  - BER (bit error)
  - RXERR (receiving a packet when transmitting another one)
  - NONE (in case it is not a DROP)
- The twelfth and last column is the frequency channel. In our chase it is always 3.
Performance Evaluation of Realistic Scenarios for Vehicular Ad hoc Networks with CityMob and NCTUns Simulator

All these information is treated and filtered with the help of the ACL. ACL is a program for massive treatment of data. Loading the text file with the traces obtained with the help of the `printPtr` program into de ACL and programming a few simple scripts (see Annex 2) we could export to an excel worksheet those traces of interest for our study (traces with a size of 1070 bytes).

Simulations were made with a MacBook Pro 2.66 GHz Intel Core 2 Duo with 4GB 1067MHz DDR of RAM memory.
5. SIMULATION RESULTS

In this chapter we are going to analyze the results that we have obtained from a performance evaluation carried out using the scenario and the simulators that we have described in the previous chapter.

5.1. VARIABLES ANALYZED

In the simulations, the effect of different types of fading and speed are analyzed. We study the effect of these variables in losses, delay end-to-end and throughput.

In wireless communications, fading is a deviation of the attenuation that a carrier-modulated telecommunication signal experiences over certain propagation media. The fading may vary with time, geographical position and/or radio frequency, and is often modelled as a random process. In the case of NCTUns we have 3 different options to choice: **no fading, Rayleigh model and Ricean model**.

The vehicle-to-vehicle radio link can be modelled statistically as a Ricean fading channel. The dominant component in the Ricean fading channel is likely to be relatively strong compared to the reflected signal, and the delay spread is likely to be relatively small because reflections occur in the immediate vicinity of the transmitter and receiver antenna. The propagation channel is modelled as a dominant component consisting of a direct-line-of-sight wave and a ground reflected wave, a set of early reflected waves, and intersymbol interference caused by excessively delayed waves. Besides without fading, the best results are expected with this fading model.
Rayleigh fading is viewed as a reasonable model for the effect of heavily built-up urban environments on radio signals. Rayleigh fading is most applicable when there is no dominant propagation along a line of sight between the transmitter and receiver.

The requirement that there be many scatters present means that Rayleigh fading can be a useful model in heavily built-up city centres where there is no line of sight between the transmitter and receiver and many buildings and other objects attenuate, reflect, refract and diffract the signal.

Speed is the other variable analyzed. We made two kinds of simulations. The first one, cars have an average speed of 50 km/h. And the other speed selected is 100 km/h as average speed. We expect to have better results in end-to-end delay with slower speed because there are less routing problems.

The metrics that we have analyzed are the following: packet losses, end-to-end delay and throughput. In the following, we describe how we computed every metric and we analyze the results obtained from the simulations. We made 15 simulations for each column in the case of channel fading models and, for each column, 10 simulations in order to study car’s speed influence. Each value in the following graphs is represented with a confidence interval of 95%, for further information about confidence intervals look Annex 3. Shorter confidence intervals are signal of similar results whereas larger confidence interval means higher dispersion of values.

### 5.1.1. Packet losses

Packet losses ratio has been calculated by this way:

\[
Packet \ Losses \ Ratio = \frac{tx + rtx - rx}{tx + rtx}
\]

In figure 5.1, the influence of fading models is showed. Each column represents a different fading model. Figures 5.2. and 5.3. show the influence of speed for each fading model.

As we preview the best results are for no fading model and worst result are for Rayleigh model. It is because of the influence of reflected waves. Without fading there are not reflected waves and in the case of Rayleigh model the direct line of sight is not stronger than waves reflected. Ricean model performs between both cases.

If we look to the speed figures we can see that, in both figures 5.2. and 5.3., when speed increases losses decrease. This is because of the scenario, cars that are outside downtown get faster to downtown and it means that they will find a better link to cars inside downtown. In both cases cars in downtown move slowly because of traffic jams. These results are similar to the obtained in the study made by another student of our working group, Roger Calzada [64], with low car density and slow speeds. In this case, he used the NS2 simulator. Both simulators show similar results in most of the scenarios we have analyzed.
SIMULATION RESULTS

Fig. 5.1. Packet losses as a function of fading model

Fig. 5.2. Packet losses as a function of speed with Ricean model
Fig. 5.3. Packet losses as a function of speed with Rayleigh model

5.1.2. End-to-end delay

The end-to-end delay formula is:

\[ \text{Delay} = \frac{\text{sum of delays}}{\text{rx}} \]

Sum of delays means the sum of the time between a tx or rtx message is send until a rx message is received.

Like in the previous result the figure 5.4. presents this variable as a function of the different fading models and in the next two figures, 5.5. and 5.6., results depicted as a function of speed. Same as before, each value is represented with a confidence interval of 95%.

The end-to-end delay is shown in the case of no fading model. Less loss usually means faster connection. According to this idea results in the Rayleigh model are much poorer than what we find in the Losses study. This is because after a packet lost a retransmission is sent, and usually it is needed more than one transmission to success. Usually links are broken and it is necessary to find a new routing path. Also received messages could be lost and it makes the end-to-end delay higher.

When this variable, end-to-end delay, is analyzed as a function of speed the result that we obtained is that when speed increases, end-to-end delay increases. This happens because links break faster and new routing path have to be found by the AODV routing protocol.
**SIMULATION RESULTS**

**Fig. 5.4.** End-to-end delay as a function of fading model

![End-to-end delay as a function of fading model](image)

**Fig. 5.5.** End-to-end delay as a function of speed with Ricean model

![End-to-end delay as a function of speed with Ricean model](image)
5.1.3. Throughput

The throughput has been calculated by this way:

$$\text{Throughput} = \frac{rx \text{ (bits)}}{\text{time (s)}}$$

Attending to the results obtained previously, especially packet losses study, the result of this variable is predictable.

Throughput decreases as a function of the influence of reflecting waves. In other words, without fading we obtain the best results and the worst results are obtained with the Rayleigh fading model (see Figure 5.7.).

Although throughput should decrease as a function of speed, in our scenario throughput increases for higher car’s speed. This happens because in this scenario, outside cars get faster to downtown and they will find easier a better link to inside cars. It is the same case that happens with the analysis of the packet losses ratio (see Figures 5.8. and 5.9.)
**Simulation Results**

**Fig. 5.7. Throughput as a function of fading models**

**Fig. 5.8. Throughput as a function of speed with Ricean model**
Fig. 5.9. Throughput as a function of speed with Rayleigh model
5.2. PROBLEMS FOUND DURING SIMULATION

NCTUns is a simulator under continuous development, and therefore, it still has some bugs and their performance is not optimal. Some of the problems found in the simulation process are described below.

The most important setback is that 802.11p protocol implementation is not yet complete. There are a lot of stub functions (e.g., transmit power cannot be set because some of the routines are not implemented). Therefore, to work with this protocol it is necessary to write your own code to integrate the original one. For instance, NCTUns simulator does not support multihop function for the 802.11p network.

Another problem is related only to the graphical interface. An error happens when configuring a scenario. The problem consists in that transmission and interference range values for both cars and base stations are not saved from one simulation to another. This causes that each simulation has to be configured from the beginning and makes it difficult to launch simulations in batches.

In the translator program that we have designed, we found a problem with a canary value (see Annex 4), which is a random value to control the data on the program stack to monitor possible buffer overflows. This value disrupted the normal operation of the translator; the instructions for fixing it are described in the same Annex 4.
6. CONCLUSIONS AND FUTURE WORK

The main motivation of this project was to study and simulate realistic scenarios from a VANET (Vehicular Ad hoc NETwork) simulator adapting it to the NCTUns [31] simulator. In Chapter 2, the concept of MANET (Mobile Ad hoc NETwork) was introduced as well as VANET, which is a particularization of MANET. Regarding both concepts we also explained the main challenges and their applications.

6.1. CONCLUSIONS

The second chapter finished describing the different mobility models in VANET and some of the most important simulators for mobility models. The majority of these programs are adapted to the NS-2 network simulator but not to NCTUns. This was the goal of Chapter 3, in which we described the development of a program that solves this problem by converting any NS-2 file into an NCTUns compatible file.

The translator program we designed makes it easier the work of generating realistic vehicular scenarios, which are very complex to design in NCTUns. At present, this program translates successfully any Manhattan grid scenario. We have chosen the Manhattan Downtown scenario because of its simplicity and also for being the most realistic scenario that represents a city with grid roads like Barcelona.

The translator program we designed helped us to set several nodes in movement and to take advantage of the specific characteristics of VANET simulators. That is, car accidents, traffic lights, downtown traffic model, multilane roads, line changing...
In our case we chose CityMob [29] as the mobility simulator. This simulator allowed us to create a more realistic city scenario, the Manhattan Downtown model, which is an improved type of a Manhattan grid that includes a downtown with higher density of cars and an exterior area with less density of cars but faster speeds. Like in a real city, most of the traffic is in downtown and outside downtown there are fewer cars.

Finally, at the end of our work, we simulated several Manhattan Downtown scenarios with 150 cars. We studied the influence of fading channel models and cars’ speed. NCTUns allowed us to select three different fading models: Ricean, Rayleigh and no fading. With the CityMob simulator we created scenarios with an average speed of 50km/h and 100km/h.

With all the simulations we obtained the expected results: fading cause poor performance in a communication system. The best result that we obtained in every variable is without fading, and the worst is with Rayleigh fading model. These results are obtained because of the effect of reflected signals. Without fading there are no signal reflected and there is only one signal in the direct-line-of-sight. In the case of Ricean model the direct-line-of sight wave is stronger that signals reflected and finally in Rayleigh model all signals have the same power. Packet losses, end-to-end delay and throughput get worse in terms of the influence of the reflected signal.

The increase in speed of the cars increased only the end-to-end delay but did not worsen the throughput or the packet losses ratio. The increase in the end-to-end delay is because with speed increase links break faster and AODV routing protocol has to recalculate routes more often. On the other hand, the improvement in packet losses ratio and throughput is because cars that are outside downtown get closer to the center faster and have more opportunities to communicate with others cars.

These results were obtained with the help of the Translator program mentioned above. The value of these results is the fact that they were achieved in a scenario with 150 cars, until now it has been worked with scenarios of 10 or 20 cars at most.
6.2. **Future Work**

All these results are obtained with AODV routing protocol, which is not the best routing protocol for this kind of vehicular communication system. One possible future work could be the design of an appropriated routing protocol based in the location of cars, density of traffic and destination.

In our simulations, data traffic is generated using a greedy source. Developing and running real multimedia applications over the vehicular network, would be another possible interesting future research direction.

And finally the last future work is the preparation of an article with all these results for the JITEL (*Jornadas de Ingeniería Telemática*) conference in 2012 (See Annex 5)
ANNEX 1. TRANSLATOR SOURCE CODE

Translator is a program, written in C, which translates a NS-2 file into an NCTUns file. For more information refer to Chapter 3. The code of the program is shown below:

/* # Translator Program #
* Xavier Campos Ríos
* Diego Pastor Pérez
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include "structs.h"

float ReadNumber(int fd, char *s){
    int i,b;
    char a;
    float n;
    i=0;
    b=1;
    while(b==1){
        read(fd,&s[i],1);
        a=s[i];
        if(a==')') b=0;
        if(a==' ') b=0;
        if(a==10) b=0;
        if(a=='\') b=0;
        else i++;
    }
    s[i]= '\0';
    n=atof(s);
    return n;
}

/* # Read Function #
* The function Read has to read the NS2 file and store the information in a NS2 struct.
*/

void read(int fichNS2, NS2 *sNS2){
char l;
char info[17], num[4];
int a,n;
float time;

lseek(fichNS2,0,SEEK_SET);

while((read(fichNS2,&l,1))>0){
a=1;
if(l=='$'){
lseek(fichNS2,1,SEEK_CUR);
read(fichNS2,&l,1);
if(l=='o'){
    lseek(fichNS2,4,SEEK_CUR); //move the pointer for reading the number
    n=ReadNumber(fichNS2,num);  //read the number
    while(1){
        read(fichNS2,&l,1);  //read character by character searching for the coordinates, I'm not sure it is a fixed distance to lseek
        if(l=='X'){
            lseek(fichNS2,2,SEEK_CUR); //for setting at the beginning of the number
            sNS2->node[n].locationNS[0].x=ReadNumber(fichNS2,info); //read the number
            break;
        }
        else if(l=='Y'){
            lseek(fichNS2,2,SEEK_CUR);
            sNS2->node[n].locationNS[0].y=ReadNumber(fichNS2,info);
            break;
        }
        else if(l==10){
            break;
        } //if it is not a X or Y, sometimes is Z and we do not care.
    }
    if(n>sNS2->numberNodes) sNS2->numberNodes=n; //check the number of nodes
}
else if(l=='s'){//Node movement line
    lseek(fichNS2,5,SEEK_CUR);
    time=ReadNumber(fichNS2,info);
    while(1){
        read(fichNS2,&l,1);
        if(l=='(') break;
    }
    n=ReadNumber(fichNS2,num);//get node number
    sNS2->node[n].numberLocations++;
    sNS2->node[n].locationNS[sNS2->node[n].numberLocations].time=time;//copy time
    lseek(fichNS2,9,SEEK_CUR);
    sNS2->node[n].locationNS[sNS2->node[n].numberLocations].x=ReadNumber(fichNS2,info);//copy coordinate X
ANNEX

```c
sNS2->node[n].locationNS[sNS2->node[n].numberLocations].y=ReadNumber(fichNS2,info);//copy coordinate Y
sNS2->node[n].locationNS[sNS2->node[n].numberLocations].speed=ReadNumber(fichNS2,info);//copy speed
}
else{
    while(a==1){
        read(fichNS2,&l,1);
        if(l==10) a=0;
    }
}
}
/* ##########################
* # Translate Function #
* ##########################
* This function makes the calculations for the translation of the NS2 structure to NCTUns structure
*/
void translate(NS2 *sNS2, NCTUns *sNCTUns, int n, int x, int y){
    int i=0,j=0,dist,semaforo;
    sNCTUns->size.x=x;
    sNCTUns->size.y=y;
    sNCTUns->numberNodes=n;
    for(i=0;i<sNCTUns->numberNodes;i++){
        sNCTUns->node[i].numberTurnings=sNS2->node[i].numberLocations-1;
        for(j=0;j<=sNCTUns->node[i].numberTurnings;j++){
            sNCTUns->node[i].locationNC[j].x=sNS2->node[i].locationNS[j].x;
            sNCTUns->node[i].locationNC[j].y=sNS2->node[i].locationNS[j].y;
            if(j==0){ // Node initial position
                sNCTUns->node[i].locationNC[j].arrivalTime=0;
                sNCTUns->node[i].locationNC[j].pausedTime=sNS2->node[i].locationNS[j+1].time;
            }
            else { // distance is calculated only for Manhattan case
                dist=sNS2->node[i].locationNS[j].x-sNS2->node[i].locationNS[j-1].x;
                if(dist==0){
                    dist=sNS2->node[i].locationNS[j].y-sNS2->node[i].locationNS[j-1].y;
                }
            }
        }
    }
}
```

/* # Translate Function #
* This function makes the calculations for the translation of the NS2 structure to NCTUns structure */
void translate(NS2 *sNS2, NCTUns *sNCTUns, int n, int x, int y){
    int i=0,j=0,dist,semaforo;
    sNCTUns->size.x=x;
    sNCTUns->size.y=y;
    sNCTUns->numberNodes=n;
    for(i=0;i<sNCTUns->numberNodes;i++){
        sNCTUns->node[i].numberTurnings=sNS2->node[i].numberLocations-1;
        for(j=0;j<=sNCTUns->node[i].numberTurnings;j++){
            sNCTUns->node[i].locationNC[j].x=sNS2->node[i].locationNS[j].x;
            sNCTUns->node[i].locationNC[j].y=sNS2->node[i].locationNS[j].y;
            if(j==0){ // Node initial position
                sNCTUns->node[i].locationNC[j].arrivalTime=0;
                sNCTUns->node[i].locationNC[j].pausedTime=sNS2->node[i].locationNS[j+1].time;
            }
            else { // distance is calculated only for Manhattan case
                dist=sNS2->node[i].locationNS[j].x-sNS2->node[i].locationNS[j-1].x;
                if(dist==0){
                    dist=sNS2->node[i].locationNS[j].y-sNS2->node[i].locationNS[j-1].y;
                }
            }
        }
    }
}
if(dist<0) dist=-dist;

if(((dist/sNS2->node[i].locationNS[j].speed)+0.5)<((sNS2->node[i].locationNS[j+1].time)-(sNS2->node[i].locationNS[j].time)))
    semaforo=1;
else
    semaforo=0;

if(semaforo==1) { // If there is a light signal
    sNCTUns->node[i].locationNC[j].arrivalTime=(dist/sNS2->node[i].locationNS[j].speed)+sNS2->node[i].locationNS[j].time;
    sNCTUns->node[i].locationNC[j].pausedTime=sNS2->node[i].locationNS[j+1].time-sNCTUns->node[i].locationNC[j].arrivalTime;
} else {
    sNCTUns->node[i].locationNC[j].arrivalTime=sNS2->node[i].locationNS[j+1].time;
    sNCTUns->node[i].locationNC[j].pausedTime=0;
}

//Possible negatives values in pausedTime
if(sNCTUns->node[i].locationNC[j].pausedTime<0){
    sNCTUns->node[i].locationNC[j].pausedTime=0;
}

sNCTUns->node[i].locationNC[j].speed=sNS2->node[i].locationNS[j+1].speed;
}

/* ##################
* # Write Function #
* ##################
* The function write writes the information stored in the NCTUns struct in a file.
*/

void write(int fichNCTUns, NCTUns *sNCTUns){
    int i=0, j=0;

    fprintf(fichNCTUns,"# The size of the working area.
";
    fprintf(fichNCTUns,"%d %d 1000\n",sNCTUns->size.x,sNCTUns->size.y);
    fprintf(fichNCTUns,"# Number of mobile nodes\n");
    fprintf(fichNCTUns,"%d\n",sNCTUns->numberNodes);
    for(i=0;i<sNCTUns->numberNodes;i++){
        if(i!=0) fprintf(fichNCTUns,"\n");
fprintf(fichNCTUns,"# Node %d\n initial location x(m) y(m) arrival_time(s) pause_time(s) speed(m/s)\n",i);

// Node intial position:
fprintf(fichNCTUns,"%d %.9f %.9f %.9f\n",sNCTUns->node[i].locationNC[0].x,sNCTUns->node[i].locationNC[0].y,sNCTUns->node[i].locationNC[0].arrivalTime,sNCTUns->node[i].locationNC[0].pausedTime,sNCTUns->node[i].locationNC[0].speed);

fprintf(fichNCTUns,"#       PATH_BEGIN number_of_turning_points\n");

if(sNCTUns->node[i].numberTurnings-1>0)
    fprintf(fichNCTUns," PATH_BEGIN %d\n",sNCTUns->node[i].numberTurnings-1);
else
    fprintf(fichNCTUns," PATH_BEGIN %d\n",0);

for(j=1;j<sNCTUns->node[i].numberTurnings;j++){
    fprintf(fichNCTUns," %d %d %.9f %.9f %.9f\n",sNCTUns->node[i].locationNC[j].x,sNCTUns->node[i].locationNC[j].y,sNCTUns->node[i].locationNC[j].arrivalTime,sNCTUns->node[i].locationNC[j].pausedTime,sNCTUns->node[i].locationNC[j].speed);
}

fprintf(fichNCTUns," PATH_END\n");
}
Performance Evaluation of Realistic Scenarios for Vehicular Ad hoc Networks with CityMob and NCTUns Simulator

```c
return;
}
read(fd, &sNS2);
close(fd);
n=sNS2.numberNodes + 1;
x=3000;
y=3000;
translate(&sNS2, &sNCTUns, n, x, y);
write(fd2, &sNCTUns);
close(fd2);
return (EXIT_SUCCESS);
```
ANNEX 2. ACL SCRIPTS

This script was used in the ACL program in order to extract those useful traces (those which packet size is 1070) for study the results of the simulations. In a first place, we load the text file generated by the *printPtr* program into the ACL (SimMuros001 in this case), and then, we executed the script. The result of the execution is a excel table with the following columns:

- **PROTOCOLO** (Protocol, in our case always 802.11)
- **TIPO_EVENTO** (Event type: RX, TX, RTX, DROP)
- **TIEMPO_INICIO** (Initial time)
- **DURACION_EVENTO** (Duration of the event)
- **TIPO_PAQUETE** (Packet Type: DATA, RTS, CTS, ACK...)
- **ID_NODO_ORIGEN** (source ID node based on the IP address)
- **ID_NODO_DESTINO** (destination ID node based on the IP address)
- **ID_MAC_ORIGEN** (source ID node based on the MAC address)
- **ID_DESTINO** (reception ID node based on the MAC address)
- **ID_MAC_DESTINO** (final reception ID node based on the MAC address)
- **ID_PAQUETE** (ID of the packet)
- **TAM_PQT** (Packet size)
- **NUM_RETX** (Number of retransmissions)
- **RAZON_PERDIDA** (Cause of drop: COLL, CAP, DUPX, BER, RXERR, NONE)
- **FREC_CANAL** (Channel frequency)

Next, the ACL script:

```
SET SAFETY OFF
SET VERIFY OFF
OPEN SimMuros001
EXTRACT FIELDS ALL TO tmp OPEN

DEFINE FIELD PROTOCOLO COMPUTED ALLTRIM(STRING(Field_1;15))
DEFINE FIELD TIPO_EVENTO COMPUTED Field_2
DEFINE FIELD TIEMPO_INICIO COMPUTED ALLTRIM(STRING(Field_4;15))
DEFINE FIELD DURACION_EVENTO COMPUTED ALLTRIM(STRING(Field_5;15))
DEFINE FIELD TIPO_PAQUETE COMPUTED Field_6
DEFINE FIELD ID_NODO_ORIGEN COMPUTED Field_8
DEFINE FIELD ID_NODO_DESTINO COMPUTED Field_9
DEFINE FIELD ID_MAC_ORIGEN COMPUTED Field_10
DEFINE FIELD ID_DESTINO COMPUTED ALLTRIM(STRING(Field_11;15))
DEFINE FIELD ID_MAC_DESTINO COMPUTED Field_12
DEFINE FIELD ID_PAQUETE COMPUTED ALLTRIM(STRING(Field_13;15))
DEFINE FIELD TAM_PQT COMPUTED ALLTRIM(STRING(Field_14;15))
DEFINE FIELD NUM_RETX COMPUTED ALLTRIM(STRING(Field_15;15))
DEFINE FIELD RAZON_PERDIDA COMPUTED Field_16
DEFINE FIELD FREC_CANAL COMPUTED ALLTRIM(STRING(Field_18;15))
```

SET FOLDER /Extracciones
EXTRACT FIELDS PROTOCOLO TIPO_EVENTO TIEMPO_INICIO DURACION_EVENTO
TIPO_PAQUETE ID_NODO_ORIGEN ID_NODO_DESTINO ID_MAC_ORIGEN ID_DESTINO
ID_MAC_DESTINO ID_PAQUETE TAM_PQT NUM_RETX RAZON_PERDIDA FREC_CANAL IF
TAM_PQT="1070" TO SimMuros001_ext
OPEN SimMuros001_ext
EXPORT FIELDS PROTOCOLO TIPO_EVENTO TIEMPO_INICIO DURACION_EVENTO TIPO_PAQUETE
ID_NODO_ORIGEN ID_NODO_DESTINO ID_MAC_ORIGEN ID_DESTINO ID_MAC_DESTINO
ID_PAQUETE TAM_PQT NUM_RETX RAZON_PERDIDA FREC_CANAL ASCII TO "/Extracciones/
SimMuros001_ext.TXT" KEEPTITLE
CLOSE PRI SEC
DELETE FORMAT tmp OK
DELETE tmp OK

SET SAFETY ON
SET VERIFY ON
SET FOLDER /
ANNEX 3. CONFIDENCE INTERVALS

In statistics, a confidence interval (CI) is a particular kind of interval estimate of a population parameter and is used to indicate the reliability of an estimate. It is an observed interval (i.e. it is calculated from the observations), in principle different from sample to sample, that frequently includes the parameter of interest, if the experiment is repeated. How frequently the observed interval contains the parameter is determined by the confidence level or confidence coefficient.

A confidence interval with a particular confidence level is intended to give the assurance that, if the statistical model is correct, then taken over all the data that might have been obtained, the procedure for constructing the interval would deliver a confidence interval that included the true value of the parameter the proportion of the time set by the confidence level. More specifically, the meaning of the term "confidence level" is that, if confidence intervals are constructed across many separate data analyses of repeated (and possibly different) experiments, the proportion of such intervals that contain the true value of the parameter will approximately match the confidence level; this is guaranteed by the reasoning underlying the construction of confidence intervals.

A confidence interval does not predict that the true value of the parameter has a particular probability of being in the confidence interval given the data actually obtained. (An interval intended to have such a property, called a credible interval, can be estimated using Bayesian methods; but such methods bring with them their own distinct strengths and weaknesses).

Interval estimates can be contrasted with point estimates. A point estimate is a single value given as the estimate of a population parameter that is of interest, for example the mean of some quantity. An interval estimate specifies instead a range within which the parameter is estimated to lie. Confidence intervals are commonly reported in tables or graphs along with point estimates of the same parameters, to show the reliability of the estimates.

Fig. A.1. Example of confidence intervals
For example, a confidence interval can be used to describe how reliable survey results are. In a poll of election voting-intentions, the result might be that 40% of respondents intend to vote for a certain party. A 90% confidence interval for the proportion in the whole population having the same intention on the survey date might be 38% to 42%. From the same data one may calculate a 95% confidence interval, which might in this case be 36% to 44%. A major factor determining the length of a confidence interval is the size of the sample used in the estimation procedure, for example the number of people taking part in a survey.

In the bar chart, the tops ends of the bars indicate observation means and the red line segments represent the confidence intervals surrounding them.

Confidence is defined as $1 - \alpha$ (1 minus the significance level). Thus, when we construct a 95% confidence interval, we are saying that we are 95% certain that the true population mean is covered by the interval - consequently, of course, we have a 5% chance of being wrong. Any statistic that can be evaluated in a test of significance ("hypothesis testing") can be used in constructing a confidence interval. Always, when constructing a confidence interval, two limits, "Upper" and "Lower", are computed. For each limit, the information needed is the computed statistic (e.g., $\bar{X}$), the two-tailed critical values (e.g., $t_{\alpha/2}$), and the standard error for the statistic (e.g., $SEM$).

The upper and lower boundaries for the confidence interval for the t-statistic given above are:

- Lower Limit = $\bar{X} - t \frac{\alpha}{2} / \sqrt{n}$
- Upper Limit = $\bar{X} + t \frac{\alpha}{2} / \sqrt{n}$
ANNEX 4. CANARY VALUE PROBLEM IN TRANSLATOR PROGRAM

Program Translator may have a problem of execution. If the number of nodes to translate is higher than 10 it has a stack error. It is solved following the next steps before compiling the program:

1. Open the console and go to the directory where the file traductor.c is.

2. Write
   
   "gcc -o 1.s -fomit-frame-pointer -S traductor.c"
   
   This line will create a file in assembly code called 1.s in the same directory of work.

3- Open the file 1.s and search the instruction
   
   “read+969: xor %gs:0x14,%eax”

   (or similar) The number 969 may be different if there is any change in the function read. This line makes a comparison in order to know the canary value of the stack and is the line that makes the file does not work. Erase the line.

4. Save the file.

5. Write in console

   “gcc -o traductor 1.s”

   This instruction will compile the file and the program is ready to work.

Canaries or canary words are known values that are placed between a buffer and control data on the stack to monitor buffer overflows. When the buffer overflows, the first data to be corrupted will be the canary, and a failed verification of the canary data is therefore an alert of an overflow, which can then be handled, for example, by invalidating the corrupted data. But in our case the data are the NS-2 files known by us so it means that there is no danger in erasing this line.

We find this solution after consulting several books and programming teachers that did not find the problem in the file traductor.c. Apparently everything works in the way it should and during several weeks we did not have any problem. We hope you do not have this problem.
ANNEX 5. ARTICLE FOR THE JITEL CONFERENCE

Performance evaluation of realistic scenarios for vehicular Ad Hoc networks with CityMob ad NCTUs

Xavier Campos Ron, Diego Pastor Pérez, Carolina Tripy Barbosa, Ahmad Melehi, Mónica Aguilar Iglesia
Departament d'Enginyeria Telsmònics, Universitat Politècnica de Catalunya (UPC)
C/Jordi Girona 1-3, Mol. C3, Campus Nord, 08034 Barcelona
E-mail: xcampoo@gmail.com, d.pastorperez@gmail.com, ctrippp@atctel.upc.edu, ahmad.melehi@atctel.upc.edu,
monica.aguilar@atctel.upc.edu

Abstract.

1 Introduction

In recent years, the continuous advances in wireless communications have opened new research lines. In this situation where the concept of mobile networks was born, the constant mobility of individuals and the need of being always connected require new research efforts in this field. Logically, connection and transmission of information from one car to the other is one of the main objectives. In addition, it’s the exponential growth of vehicle number in cities has led to one of the main focuses, in which to invest time and money is road safety.

VANETs (Vehicular Ad hoc NETWORKS) are a particular case of the mobile networks, in which mobile nodes are vehicles. The concept – technology that uses moving cars as nodes in a network to create a temporary mobile network – initially seems simple, but it presents many challenges both theoretical and technical (routing protocols, transmission ranges and frequencies, quality of service...) The study of VANETs is expensive and complicated because of the deployment that would be necessary to test it in real environments. This raises the need of performing simulation as realistic as possible in order to study, from a computer, settings and situations as similar as possible to those obtained in a real environment.

Firstly, a movement simulator that reproduces the movement of vehicles on maps and scenarios closer to reality is needed. Most of these VANET simulators are adopted to veh-2-vehicle network simulator but in our case we are going to work with NCTUs network simulator. Because of that, one of our greatest challenges in the present project was to develop a universal translator for veh-2-vehicle simulator in order to make them compatible with NCTUs.

The choice of NCTUs as our network simulator is because this network simulator integrates some traffic simulation capabilities, such as designing maps and controlling vehicles mobility. Also, NCTUs includes a GUI tool in the design process of the maps. But the main reason of choosing this simulator is because NCTUs network simulator has many useful features: it directly uses the real-life Linux TCP/EP protocol stack to generate high-fidelity simulation results, it provides a highly integrated and professional GUI environment and its simulation engine adopts an open-system architecture and is open-source.

2 Mobile networks

Vehicular Ad hoc NETWORKS (VANETs) have recently emerged as a platform to support intelligent inter-vehicle communication to improve traffic safety. The road-constrained characteristics of these networks and the high mobility of the vehicles, their unattended power source, and the emergence of roadside wireless infrastructures make VANETs a challenging and promising research topic.

VANETs aim to provide our cars and roads with capabilities to make road more secure and to make our time on the road more enjoyable, enabling communications among nearby vehicles (car to car communication) as well as between vehicles and nearby fixed equipment (car to infrastructure communication). The following variety of applications is a typical example of an intelligent transportation system (ITS):

- Safety, which a warning message will be broadcasted from a vehicle to its neighborhood notifying about some event such as a collision or road surface conditions in order to decrease traffic accidents rate and enhance traffic flow control. It refers to applications or systems that increase the protection of the people in the vehicles as well as the vehicles itself.
- Resource efficiency, referring to increase traffic fluidity with data such as advanced route guidance or parking spot location services. Better efficiency results in less congestion and lower fuel consumption, helping to minimize environmental and economic impact.
- Information and Advanced Driver Assistance Services (ADAS), combining
information and entertainment and offering multimedia and Internet connectivity facilities for passengers.

The huge potential of car-to-car connectivity is fundamentally due to the constant growth of automotive market and the increasing demand for car safety. Some issues relating to architecture, routing, security, performance or QoS should be investigated. It is necessary to put special attention to ensure interoperability through the standardization of protocols and interfaces in order to allow the communication between vehicles from different manufacturers.

3 VANET simulators
There are some models proposed in the general MANET context usable in VANET. The criteria of applicability considered are the employment of road maps, and limiting the nodes movements into the roads instead of moving in a wide area. The considered parameters differ from a model to another. For instance, some models use traffic control mechanisms at intersections, and some others just assume continuous movement at these points. Some assume roads to be a single-lane, but some others support multi-lane roads. Some define the security distance, while others just ignore this parameter. In the following, it is going to be described the main features of the most spread vehicular traffic models:

3.1 STRAW
3.2 SUMO
3.3 MOVE
3.4 VanetMobiSim
3.5 FreeSim
3.6 CityMob

4 Translator program
Translator_Ns2_NCTUs is a program that translates a ns-2 file into an NCTUs file. It is programmed in C over Linux, which means it is an open source program. This program has 3 main parts: Read, Translate and Write. Before running the program is needed to have in the same directory the ns-2 file is going to be translated with the name "game.txt". After using the program you will have an NCTUs file called "valida.mtx" which could be directly loaded at NCTUs.

This function makes the calculations for the translation of the NS2 structure to NCTUs. It needs two pointers one for a structure NS2 and another for a structure NCTUs. Also it needs the number of nodes and the size of the working area.

```c
void translateNS2_toNCTUs(int n, int m, int s, int y)
```

The function starts with a loop defined by the number of nodes, and inside the loop the first action is determine the number of turnings per node. The number of turnings is decreased because a problem in the way the different files are written. An NS2 file sets a future position and the node speed in a determinate instant and is a NCTUs file the node reach a determinate position with a defined speed in a certain instant.

```c
for (i=0; i<e; i++) {
    nCTUs->
        node[1].numberTurnings = NS2->
        node[1].numberLocations + 1;

    The new loop is created, this deals with the number of
    turnings. Firstly it sets the coordinates X and Y and
    then it looks if this is the initial turning which has a
    different treatment. In this case the arrival time is 0 and
    the paused time is the time of the next NS2 node.

    for (j=0; j<e; j++) {
        nCTUs->
            node[1].numberTurnings = j + 1;

        if (j==0) {
            nCTUs->
                node[1].locationNC[j].x = NS2->
                node[1].locationNS2[j].x;

            nCTUs->
                node[1].locationNC[j].y = NS2->
                node[1].locationNS2[j].y;

            if (j==0) {
                nCTUs->
                    node[1].locationNC[j].arrivalTime = 0;

            nCTUs->
                node[1].locationNC[j].pausedTime = NS2->
                node[1].locationNS2[j+1].time;
        }
    }

    If it is not the initial turning it starts calculating the
    distance between the previous turning and the actual
    This function is based in a Manhattan grid which
    means that only has movement in one of the coordinates, X or Y, never in both. This simplifies the
    calculus of possible accidents and lights signals. The
    existence of a possible accident is calculate with a
    comparative between the arrival time in the next
    turning in the NS2 file and a time previously calculated
    with the distance and the speed settled in the previous
    turning. It is added a margin to avoid mistakes in case
    of irrelevant differences between both times.

    if (dist >= 0) {
        dist = NS2->
            node[1].locationNS2[j+1].x - NS2->
            node[1].locationNS2[j].x;

        if (dist >= 0) {
            dist = NS2->
                node[1].locationNS2[j+1].y - NS2->
                node[1].locationNS2[j].y;
```
Performance Evaluation of Realistic Scenarios for Vehicular Ad Hoc Networks with CityMob and NCTUns Simulator

if (dist >= 0) dist = dist;
if (((dist/Ns2 - node[i].locationNS[i].speed) + 0.5) <= (Ns2 - node[i].locationNS[i].time) - (Ns2 - node[i].locationNS[i].time))
    sfilter = 1;
else
    sfilter = 0;

If the program detects a light signal the arrival time will be calculated. The arrival time will be the arrival time in the previous running plus a time calculated by the distance divided by the speed settled in the previous turning. The paused time is the time of the next running calculated by subtracting the earlier arrival time from the NCTUns - node[i].locationNS[i].arrivalTime = (dist / Ns2 - node[i].locationNS[i].time);

In case of no light signal the program sets the arrival time as the time of the next running and the paused time is 0.

Finally the last part of the turning loop the program detects possible mistakes in the paused time. Sometimes due a difference of calculation this value is negative. This negative value makes the NCTUns file useless and NCTUns simulator does not change the file. For not having this problem the function looks for this possible value and changes it to 0. Most of the time this value is very low so change it to 0 does not affect the final result.

And the final time sets the speed.

This function does not deal with the final running. To translate an .m-2 file into a NCTUns file the information of the next running is necessary but when it is in the final running this information does not exist. So the final running in a .m-2 file does not have a translation into a NCTUns file.

To simplify the comprehension of the process done by this function the next table explains what is done.

5 Description of scenario

The considered scenario consists of a VANET deployed over an area of a city with square streets. In this way we are modeling a scenario similar to the Example district of Barcelona, with a central area with differs from the rest by the highest density of vehicles and an average speed lower than the other areas. Nodes can be out of range when transmitting so the VANET has to rely on multi-hop to deliver the packages.

There are no base stations because only nodes (vehicles) will exchange information among themselves.

The scenario chosen for the simulations of this project has the following characteristics:

- Manhattan Downtown model
- 150 nodes (vehicles)
- Duration 100s
- Average speed: 15 m/s – 30 m/s (50km/h – 100km/h)
- 1500x1500m with distance between streets of 100m
- Downtown: 500x500m centered on the scenario
- 5 accidents (an accident is modelled keeping a node static along the simulation):

The death probability of a node being initially located inside the downtown is set to 0.7 by default. The
routing protocol used is AODV. The road features 2 lanes in each way.

This configuration of the scenario results in the following CityMob command for the “slow” simulations:
```
./citymob -m 2 -m 180 -d 100 -a 16 -w 1000 -h 1000 -d 100 -a 5 -x 500 -y 500 -x 1100 -y 1100
```
and for “fast” simulations:
```
./citymob -m 2 -m 180 -d 100 -a 20 -w 1000 -h 1000 -d 100 -a 5 -x 500 -y 500 -x 1100 -y 1100
```

Packet size, transmission range and the rest of the physical parameters will be set as described in the example of the previous chapter. It is important to mention that walls are added to the scenario using the GUI of NCTUs, with an attenuation of 5dB according to the values stated by another group of work [63]. This example of scenario could be summarized with the following picture:

![Scenario Diagram]

We could see later on, in the following chapter, with the help of the NCTUs Network simulator, how this example is reflected.

6 Simulation results

In the simulations, the effect of different types of fading and speed are analyzed. We look the effect of these variables in losses, delay and end-to-end throughput.

In wireless communications, fading is deviation of the attenuation that a carrier-modulated telecommunication signal experiences over certain propagation media. The fading may vary with time, geographical position and/or radio frequency, and is often modelled as a random process. In the case of NCTUs, we have 3 different options to choose: no fading, Rayleigh model and Ricean model.

The vehicle-to-vehicle radio link can be modelled statistically as a Ricean fading channel. The dominant component in the Ricean fading channel is likely to be relatively strong compared to the reflected signal, and the delay spread is likely to be relatively small because reflections occur in the immediate vicinity of the transmitter and receiver antennas. The propagation channel is modelled as a dominant component consisting of a direct line-of-sight wave and a ground reflected wave, a set of early reflected waves, and intersymbol interference caused by excessively delayed waves. Besides without fading, the best results are expected with this fading model.

Rayleigh fading is viewed as a reasonable model for the effect of heavily built-up urban environments on radio signals. Rayleigh fading is most applicable when there is no dominant propagation along a line of sight between the transmitter and receiver.

The requirement that there be many scatters present means that Rayleigh fading can be a useful model in heavily built-up city centers where there is no line of sight between the transmitter and receiver and many buildings and other objects attenuate, reflect, reflect and disturb the signal.

Speed is the other variable analyzed. We made two kinds of simulations. The first one, cars have an average speed of 50 km/h. A car with an average speed of 100 km/h as average speed. Downward results in end to end delay value moving speed because there are less routing problems.

6.1 Packet losses ratio

Packet losses ratio has been calculated by this way:

\[
\text{Packet Losses Rate} = \frac{\text{Lost Packets}}{\text{Total Packets Transmitted}} \times 100
\]

In the figure 6.1.1, the influence of fading models is showed. Each column represents a different fading model. Figures 6.1.2 and 6.1.3 shows the influence of speed for each fading model. Each value is represented with a confidence interval of 95%.

As we can see the best result Car model and worst result are for Rayleigh model, 

Because of the influence of reflected waves. Without fading there are no reflected waves and in the case of Rayleigh model the direct line of sight is not stronger than waves reflected. Ricean model is between both cases.

If we look to the speed figures we can use that in both figures when speed is increased losses decreases. This is because of the scenario, cars that are outside of the downtown approaches faster to the downtown and it means that they will have a better link to cars inside the downtown. In both cases cars in downtown move more slowly because of traffic jams. These results are similar to the obtained in the study of another group of work [1] with low density of cars and slow speeds.
6.2 End to end delay

End to end delay formula is:

\[
\text{Delay} = \frac{\text{sum of delays}}{\text{no of packets}}
\]

Like in the previous result the first figure presents this variable in function of the different fading models and in the next two figures, results are in function of speed. Since as before, each value is represented with a confidence interval of 95%.

In end-to-end delay is in the case of no fading model. Less loss usually means faster connection. According to this idea results in the Rayleigh model are much poorer than what we find in the Losses study. This is because after a packet lost a retransmission is sent and usually is needed more than one to have success. Usually in this time links are broken a it is necessary to create a new routing table. Also received messages could be lost and it makes the end to end delay bigger.

6.3 Throughput

Throughput has been calculated by this way:

\[
\text{Throughput} = \frac{\text{txp (bits)}}{\text{time (s)}}
\]

Attending to the results obtained previously, especially packet losses study, the result of this variable is predictable. Throughput decreases in function of the influence of reflecting waves. In other words, without fading we obtain the best results and the worst results are with Rayleigh fading model.

Although when speed increase end to end delay increase the fact that there are less packet losses helps to increase the throughput despite of the increase of speed.
7 Conclusion and future work

The main motivation of this project was to study and simulate realistic scenarios from a VANET simulator adapting it to NCTUs. In Chapter 2, the concept of MANET was introduced as well as VANET, which is a particularization of MANET. In both concepts we present challenges and applications.

The second chapter is finished describing the different mobility models in VANET and some of the most important simulators. The majority of these programs are adapted to the ns-2 network simulator and not to NCTUs. This is the goal of the Chapter 3, in which we describe the development of a program that solves this problem by converting any ns-2 file into an NCTUs compatible file.

The translator program makes easier the work of generation of realistic scenarios, which are very complex to create in NCTUs. At present, this program translates successfully any Manhattan grid scenario.

The translator helps us to place several nodes in movement and take advantage of the specific characteristics of VANET simulators. In our case we use CityMob as the VANET simulator. CityMob allows us to create the most realistic city scenario, the Manhattan Downton scenario, which is a improved type of a Manhattan grid that includes a downtown with higher density of cars and an exterior area with less density of cars but faster speeds. Like in a real city most of the traffic is in the downtown and outside of this area there are less cars.

Finally, at the end of our work, we simulate different Manhattan Downton scenarios with 159 cars. We study the influence of fading models and speed. NCTUs allows us to select two different fading models: Ricean and Rayleigh, or no fading. With the CityMob simulator we create scenarios with an average speed of 30 km/h and 100 km/h.

With all the simulations we obtained the results expected: fading causes poor performance in a communication system. The best result that we obtained in every variable is without fading, and the worst is with Rayleigh fading model. These results are obtained because of the effect of reflected signals. Without fading there are no signal reflected and there is only one signal in the direct line-of-sight. In the case of Ricean model, the direct line of sight wave is stronger than the signal reflected and finally in Rayleigh model all signals have the same power. Packet losses, end-to-end delay and throughput get worse in terms of the influence of the reflected signal.

Instead of making the performance much poorer, speed increase only affects the end to end delay. Packet losses and throughput improve with the increase of speed. The first effect, more end to end delay, is because with speed increase links break faster and AODV routing protocol has to calculate routes more often. On the other hand the fact that improving packet losses ratio at throughput is because cars that are outside of the downtown get closer to the center faster and have more facilities to communicate.

All this results are obtained with AODV routing protocol, which is not the best routing protocol for this kind of communication system. One possible future work could be the creation of an appropriated routing protocol based in the location of cars, density of traffic and destination.

In our simulation the traffic is generated using a flooding algorithm. Developing and running real applications over the network, would constitute another possible future research direction.

8 References

“Citymob: a mobility model pattern generator for VANETs.”

“Performance evaluation of a Vehicular Ad Hoc network in a realistic double Manhattan scenario”.
Proyecto final de carrera, UPC, 2011.
REFERENCES


[4] Hongyu Huang; Yanmin Zhu; Xu Li; Minglu Li; Min-You Wu; , "META: A Mobility Model of MEtropolitan TAxis Extracted from GPS Traces," Wireless Communications and Networking Conference (WCNC), 2010 IEEE , vol., no., pp.1-6, 18-21 April 2010


[18] Jie Luo; XinXing Gu; Tong Zhao; Wei Yan; , "MI-VANET: A New Mobile Infrastructure Based VANET Architecture for Urban Environment," Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd , vol., no., pp.1-5, 6-9 Sept. 2010


[22] CVIS (Cooperative Vehicle-Infrastructure Systems), official webpage: http://www.cvisproject.org/


[25] IntelliDrive, official webpage: http://www.intellidriveusa.org/


[27] Cooperative Intersection Collision Avoidance System (CICAS), official webpage: http://www.its.dot.gov/cicas/

[28] BonnMotion official webpage: http://net.cs.uni-bonn.de/wg/cs/applications/bonnmotion


[37] Shie-Yuan Wang, Chih-Liang Chou, Chih-Che Lin, Chih-Hua Huang, The Protocol Developer Manual for the NCTUns 6.0 Network Simulator and Emulator, National Chiao Tung University, Tajwan 2010


[51] F. Bai, N. Sadagopan, and A. Helmy, "Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for ad hoc networks"
[52] V. Davies, "Evaluating mobility models within an ad hoc network"