DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

Delay Tolerant Networks for Efficient Information Harvesting and Distribution in Intelligent Transportation Systems

By

Sergio Martínez Tornell

Advisors:

Dr. Pietro Manzoni
Dr. Juan Carlos Cano Escribá

Universitat Politècnica de València
Valencia, Spain

May, 2016

To my mother and to Marisa.

The most important discoveries will provide answers to questions that we do not yet know how to ask and will concern objects we have not yet imagined.

John N. Bahcall

Acknowledgements

I want to sincerely thank my advisors, Dr. Pietro Manzoni and Dr. Juan Carlos Cano. This thesis would not have been possible without their support and guidance (and funding). I also want to thank Dr. Carlos Calafate for his meticulous corrections and the long discussions we had at lunch. I can not forget that Carlos gave me the opportunity to teach, which was one of the most rewarding experiences I will ever have.

Special thanks goes to Prof. Jörg Ott from TUM Technische Universität München and Prof. Joel Rodrigues from the Universidade da Beira Interior, who took me in during my time in Helsinki and in Covilhã. I will never forget their advices and their efforts to make me feel like home.

I also want to thank my colleges in the GRC: to Johann, for the long talks in the evenings; to Alvaro and Miguel, for the moments we shared and the moments we will share; to "the foreigners", Ali and Suhba; and to all the "Ecuadorians" that at some point decided to conquer the lab. Each and every one of them made every moment at the GRC worth it.

Apart from job and research, there were a lot of people that made Valencia my second home. Because of them, I want to thank to Rugby. Because during this time Rugby brought to me my best friends, made me meet an awesome group of people like the CRUBI in Covilhã and illuminated the darkest Finish days.

When I came to Valencia, there were people I left behind in Cartagena. I always remember my closest friends, Sergio & Jose, Pablo & Inma, David, and Dani. I also want to thank my mother Maria Jesús and my sister Miriam, because I know they will be always there and never will let me down.

Finally, I want to thank to my lover Marisa, who not only gave me a new vision of life and people, she also encouraged me to "do things", to research, to travel, to teach, to collaborate, to create new projects, no matter it brought us together or made us apart.

Thank you all.

Sergio Martínez Tornell
Valencia, May 12, 2016

Abstract

Intelligent Transportation Systems (ITS) can make transportation safer, more efficient, and more sustainable by applying various information and communication technologies. One of these technologies are Vehicular Networks (VNs). VNs combine different communication solutions such as cellular networks, Vehicular Ad-Hoc Networks (VANETs), or IEEE 802.11 technologies to provide connectivity among vehicles, and between vehicles and road infrastructure.

This thesis focuses on VNs, and considers that the high speed of the nodes and the presence of obstacles like buildings, produces a highly variable network topology, as well as more frequent partitions in the network. Therefore, classical Mobile Ad-hoc NETwork (MANET) protocols do not adapt well to VANETs. Under these conditions, Delay Tolerant Network (DTN) have been proposed as an alternative able to cope with these adverse characteristics. In DTN, when a message cannot be routed to its destination, it is not immediately dropped but it is instead stored and carried until a new route becomes available. The combination of VN and DTN is called Vehicular Delay Tolerant Networks (VDTNs).

In this thesis, we propose a new VDTN protocol designed to collect information from vehicular sensors. Our proposal, called Map-based Sensor-data Delivery Protocol (MSDP), combines information about the localization obtained from a GNSS system with the actual street/road layout obtained from a Navigation System (NS) to define a new routing metric. Both analytical and simulation results prove that MSDP outperforms previous proposals.

Concerning the deployment of VNs and VANET technologies, technology already left behind the innovation and the standardization phases, and it is about time it reach the first early adopters in the market. However, most car manufacturers have decided to implement VN devices in the form of On Board Units (OBUs), which are expensive, heavily manufacturer dependent, and difficult to upgrade. These facts are delaying the deployment of VN. To boost this process, we have developed the GRCBox architecture. This architecture is based on low-cost devices and enables the establishment of V2X, *i.e.* V2I and V2V, communications while integrating users by easing the use of general purpose devices like smartphones, tablets or laptops. To demonstrate the viability of the GRCBox architecture, we combined it with a DTN platform called Scampi to obtain actual results over a real VDTN scenario. We also present several GRCBox-aware applications that illustrate how developers can create applications that bring the potential of VN to user devices.

Resumen

Los sistemas de transporte inteligente (ITS) son el soporte para el establecimiento de un transporte más seguro, más eficiente y más sostenible mediante el uso de tecnologías de la información y las comunicaciones. Una de estas tecnologías son las redes vehiculares (VNs). Las VNs combinan diferentes tecnologías de comunicación como las redes celulares, las redes ad-hoc vehiculares (VANETs) o las redes 802.11p para proporcionar conectividad entre vehículos, y entre vehículos y la infraestructura de carreteras.

Esta tesis se centra en las VNs, en las cuales la alta velocidad de los nodos y la presencia de obstáculos como edificios producen una topología de red altamente variable, así como frecuentes particiones en la red. Debido a estas características, los protocolos para redes móviles ad-hoc (MANETs) no se adaptan bien a las VANETs. En estas condiciones, las redes tolerantes a retardos (DTNs) se han propuesto como una alternativa capaz de hacer frente a estos problemas. En DTN, cuando un mensaje no puede ser encaminado hacia su destino, no es inmediatamente descartado sino es almacenado hasta que una nueva ruta esta disponible. Cuando las VNs y las DTNs se combinan surgen las redes vehiculares tolerantes a retardos (VDTN).

En esta tesis proponemos un nuevo protocolo para VDTNs diseñado para recolectar la información generada por sensores vehiculares. Nuestra propuesta, llamada MSDP, combina la información obtenida del servicio de información geográfica (GIS) con el mapa real de las calles obtenido del sistema de navegación (NS) para definir una nueva métrica de encaminamiento. Resultados analíticos y mediante simulaciones prueban que MSDP mejora el rendimiento de propuestas anteriores.

En relación con el despliegue de las VNs y las tecnologías VANET, la tecnología ha dejado atrás las fases de innovación y estandarización, ahora es el momento de alcanzar a los primeros usuarios del mercado. Sin embargo, la mayoría de fabricantes han decidido implementar los dispositivos para VN como unidades de a bordo (OBU), las cuales son caras y difíciles de actualizar. Además, las OBUs son muy dependientes del fabricante original. Todo esto esta retrasando el despliegue de las VNs. Para acelerar la adopción de las VNs, hemos desarrollado la arquitectura GRCBox. La arquitectura GRCBox esta basada en un dispositivo de bajo coste que permite a los usuarios usar comunicaciones V2X (V2V y V2I) mientras utilizan dispositivos de propósito general como teléfonos inteligentes, tabletas o portátiles. Las pruebas incluidas en esta tesis demuestran la viabilidad de la arquitectura GRCBox. Mediante la combinación de nuestra GRCBox y una plataforma de DTN llamada Scampi hemos diseñado y probado un escenario VDTN real. También presentamos como los desarrolladores pueden crear nuevas aplicaciones GRCBox para llevar el potencial de las VN a los dispositivos de usuario.

Resum

Els sistemes de transport intel·ligent (ITS) poden crear un transport més segur, més eficient i més sostenible mitjançant l'ús de tecnologies de la informació i les comunicacions aplicades al transport. Una d'aquestes tecnologies són les xarxes vehiculars (VN). Les VN combinen diferents tecnologies de comunicació, com ara les xarxes cel·lulars, les xarxes ad-hoc vehiculars (VANET) o les xarxes 802.11p, per a proporcionar comunicació entre vehicles, i entre vehicles i la infraestructura de carreteres.

Aquesta tesi se centra en les VANET, en les quals l'alta velocitat dels nodes i la presència d'obstacles, com els edificis, produeixen una topologia de xarxa altament variable, i també freqüents particions en la xarxa. Per aquest motiu, els protocols per a xarxes mòbils ad-hoc (MANET) no s'adapten bé. En aquestes condicions, les xarxes tolerants a retards (DTN) s'han proposat com una alternativa capaç de fer front a aquests problemes. En DTN, quan un missatge no pot ser encaminat cap a la seua destinació, no és immediatament descartat sinó que és emmagatzemat fins que apareix una ruta nova. Quan les VN i les DTN es combinen sorgeixen les xarxes vehicular tolerants a retards (VDTN).

En aquesta tesi proposem un nou protocol per a VDTN dissenyat per a recol·lectar la informació generada per sensors vehiculars. La nostra proposta, anomenada MSDP, combina la informació obtinguda del servei d'informació geogràfica (GIS) amb el mapa real dels carrers obtingut del sistema de navegació (NS) per a definir una nova mètrica d'encaminament. Resultats analítics i mitjançant simulacions proven que MSDP millora el rendiment de propostes prèvies.

En relació amb el desplegament de les VN i les tecnologies VANET, la tecnologia ha deixat arrere les fases d'innovació i estandardització, ara és temps d'aconseguir als primers usuaris del mercat. No obstant això, la majoria de fabricants han decidit implementar els dispositius per a VN com a unitats de bord (OBU), les quals són cares i difícils d'actualitzar. A més, les OBU són molt dependents del fabricant original. Tot això està retardant el desplegament de les VN.

Per a accelerar l'adopció de les VN, hem desenvolupat l'arquitectura GRCBox. L'arquitectura GRCBox està basada en un dispositiu de baix cost que permet als usuaris usar comunicacions V2V mentre usen dispositius de propòsit general, com ara telèfons intel·ligents, tauletes o portàtils. Les proves incloses en aquesta tesi demostren la viabilitat de l'arquitectura GRCBox. Mitjançant la combinació de la nostra GRCBox i la plataforma de DTN Scampi, hem dissenyat i provat un escenari VDTN pràctic. També presentem com els desenvolupadors poden crear noves aplicacions GRCBox per a portar el potencial de les VN als dispositius d'usuari.

Contents

List of Figures

List of Tables

Listings

Part I

Background

Chapter 1

Introduction

Wireless networks have evolved at a very fast rate and are being used in several contexts to offer different communication solutions. In the automobile industry, many wireless solutions have been proposed to implement safety- related applications by implementing data communication among vehicles and between vehicles and infrastructure. These proposals contribute to the Intelligent Transport Systems (ITSs) field, which can make transport safer, more efficient, and more sustainable by applying various information and communication technologies to passengers and freight transport. One of these technologies are Vehicular Networks (VNs). VNs combine different communication technologies such as cellular networks, Vehicular Ad-Hoc Networks (VANETs)[50], or IEEE 802.11 networks to provide communication between vehicles (Vehicle to Vehicle (V2V)), and between vehicle and road infrastructure (Vehicle to Infrastructure (V2I)). The core of VNs is the IEEE 802.11p standard [54], which provides direct V2V communications. The IEEE 802.11p standard modifies the IEEE 802.11a standard to meet low delay requirements for safety applications.

This thesis focuses on VANETs to provide V2V communication. Sometimes, VANETs are considered as a subset of Mobile Ad-hoc NETworks (MANETs). However, the high speed of the nodes in a VANET, and the presence of obstacles like buildings, produce a highly variable network topology, as well as more frequent partitions in the network. Therefore, typical MANET protocols [69] do not adapt very well to VANETs since an instantaneous fully connected path between sender and receiver is usually unavailable. Under these conditions, Delay Tolerant Networks (DTNs) [16] are considered an alternative able to deal with VANET characteristics, and are also applicable to VN to provide ITS services.

DTNs were originally proposed for InterPlanetary Networks (IPNs) to provide communication between satellites and base stations. DTNs permit information to be shared between nodes even in the presence of high delays, which are typical in satellite communications. In DTNs, when a message cannot be routed to its destination, it is not immediately dropped but it is instead stored and carried until a new route becomes available. Messages are removed from the buffer when their lifetime expires or because buffer capacity reasons. This mechanism cannot only be applied to InterPlanetary Networks (IPNs) but also to VNs, taking advantage of their high degrees of mobility [34, 68]. DTNs have been standardized by the Delay Tolerant Network Research Group (DTNRG) [26] to ensure network interoperability.

This thesis focuses on the combination of VNs and DTNs, called Vehicular Delay Tolerant Networks (VDTNs). VDTNs have been explored as a solution to overcome the problems of VANETs related with mobility, such as network partitioning and short duration routes. VDTNs can be used for applications like information harvesting and dissemination, cooperative downloads, or floating content.

1.1 Motivation

Due to the cost of deployment, VDTN proposals are usually evaluated through simulations. While dozens of researchers have already presented their VDTN protocol proposals, the variety of simulators and simulation models, as well as the extreme complexity of some protocols, make almost impossible to replicate other researcher results. This problem means that it is currently unfeasible to reach conclusive results when comparing several protocols through simulations.

To improve the reproducibility and the repeatability of simulation based studies, we need to develop a new mobility manager and a new VDTN protocol model that allow to easily and quickly implement and compare VDTN protocols.

Concerning the deployment of VNs and VANETs technologies, the technology has left behind the innovation and the standardization phases, now it is time to reach the first early adopters in the market. However, most car manufacturers have decided to implement VN devices in the form of On Board Units (OBUs). OBUs are integrated in the dashboard which increases their price and complicates their update. Therefore, early adopters are reluctant to invest money in VN devices and their deployment has been delayed for years.

Simultaneously, smartphones recently reached a 60% of penetration in developed countries, and this value is still growing. Smartphones are typically equipped with several network interfaces: WiFi, cellular network, and Bluetooth. From our point of view, smartphones offer an opportunity for developers and VANETs, which can evolve from a pure ad-hoc network, with its known limitations, to a heterogeneous and more versatile network taking advantage of the possibilities offered by their wireless network interfaces.

The integration of smartphones into VNs can increase their adoption. It would also allow developers to quickly implement and test new VN applications. However, smartphones connectivity is limited to infrastructure networks, which limits their direct adoption for VN.

1.2 Objectives

This thesis provides a wide overview of VN focused on VDTN. Besides its novelty, this thesis offers a solid background to novel researchers. The overall objective of this thesis is: designing and implementing a new VDTN protocol, which will be implemented in a novel platform able to accelerate the adoption of VN. In order to accomplish the main objective of this thesis the following specific objectives have been defined:

- We propose the Map-based Sensor-data Delivery Protocol (MSDP), a new VDTN protocol. MSDP combines information obtained from the Geographic Information Service (GIS) with the actual street/road layout obtained from the Navigation System (NS) to define a new routing metric. Most proposed VDTN protocols do not take into consideration the effects of common basic communication mechanisms, as stated in section 2.4.4. MSDP is based on our Generic One-Copy DTN Model (GOD) model, which enables the definition and quick implementation of a variety of VDTN protocols. The GOD model enables MSDP configuration to optimize both, its own routing metric and low level routing mechanisms.

- Although big efforts have big done to improve mobility models for VN, few works have focused on providing the appropriate tools for modeling road traffic demand. It is hard to define how the number of vehicles in a simulation varies in time and, in most previous works, it is barely reported. Our contribution, VACaMobil, simplifies the definition of road traffic demand, which improves reproducibility and repeatability.

- All previous VN implementations relay in self-designed OBUs, which usually are expensive devices. This fact is slowing the penetration rate of the VN technology. On the other side, possible alternatives like smartphones present software limitations due to their original purpose. A platform that enables users to use off the shelf devices such as smartphones or tablets into VNs is desirable. In this thesis we have developed the GRCBox, an architecture that enables the quick integration of smartphone applications into VNs.

1.3 Structure of the Thesis

This dissertation is organized in 5 parts. Below, we briefly describe the contents of each part:

I **Background:** Includes this introductory chapter as well as a chapter covering the state of the art of VN and VDTN. It introduces the basic concepts to understand this thesis and surveys the most relevant previous proposals for VDTN.

II **Contributions:** Presents the main contributions of this thesis. We introduce: The MSDP, a new VDTN protocol; VACaMobil, a mobility manager for VN simulations; and the GRCBox architecture, which join smartphones and VNs.

III **Results & Experiments:** Presents the methodology used to test, prove, validate and evaluate each one of our contributions. The evaluation methods range from simulations to real experiments.

IV **Conclusions & Publications:** Concludes this thesis and presents the publications related to this thesis as well as a list of future research lines.

V **Appendices and References:** The final part includes appendices like the list of acronyms and the bibliography.

Chapter 2

State of the Art

The contents of this chapter have been partially published in:

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "DTN Protocols for Vehicular Networks: An Application Oriented Overview". In: *IEEE Communications Surveys & Tutorials* 17.2 (2015), pp. 868–887. ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2375340. Impact Factor: 6.806(Q1).

- M. Báguena, S. M. Tornell, Á. Torres, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A Tool Offering Steady-State Simulations for VANETs". In: *Recent Advances in Communications and Networking Technology* 2.2 (2013), pp. 102–112. DOI: 10.2174/2211740711201660008.

This chapter has two main objectives: providing an introduction to Vehicular Networks (VNs), Delay Tolerant Networks (DTNs) and their combination in the form of VDTNs, and surveying previously proposed VDTN protocols. Section 2.1 introduces the reader to VNs, by presenting their technologies and standards. Section 2.2 presents the DTN paradigm, which has been proposed to face some of the specific challenges of VNs. Section 2.3 gives an insight on how VNs and DTNs can be combined to form VDTN. Then, we analyze the previous works done by other researchers. Section 2.4 surveys 40 previously proposed VDTN protocols by classifying them in different groups and exposing their strength and weaknesses. In section 2.5 we introduce some of the possible applications of VDTN and discuss the suitability of each protocol group for each application. Section 2.6 analyzes how those protocols were evaluated, we identified problems in the reproducibility and the repeatability of the conducted experiments. Finally, Section 2.7, summaries our findings.

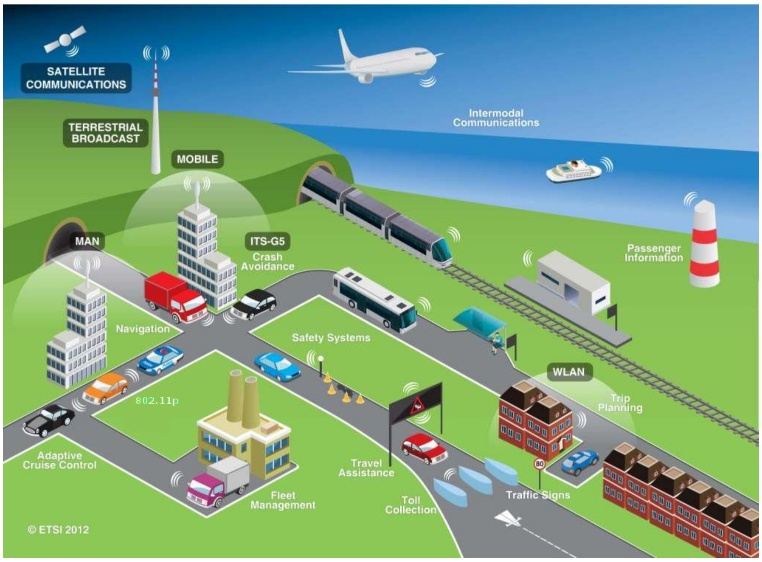


Figure 2.1: An example of several technologies interconnected to stablish an ITS scenario[32].

2.1 Vehicular Networks (VNs)

VNs are a core part of ITS. They will connect the vehicles and the infrastructure to make ITS possible. Depending on the application, two different technologies are mainly used: The ad-hoc communication technology 802.11p [53] and infrastructure networks such as WiFi, 3G, or LTE. The later depend on centralized network topologies that require a base station. On the other side, 802.11p relies on direct communication between mobile stations (also referred as *vehicles*) by creating and interconnecting Vehicular Ad-Hoc Networks (VANETs). Figure 2.1 illustrates how different technologies can be combined to create ITS scenarios.

VANETs can be seen as a specific form of MANETs [69]. However, while in MANETs nodes are considered to move slowly (usually walking speed is considered), in VANETs nodes are vehicles that move much faster. As a consequence, neither reactive [105] or proactive [59] previously designed MANET routing protocols are suitable for VANETs. VNs and specially VANETs suffer certain challenging issues that difficult the operation of previously designed routing protocols. We can summarize these issues in three main characteristics:

- High mobility: Since the nodes of a VANET are essentially vehicles that move at high speed, the network topology is highly variable, therefore a

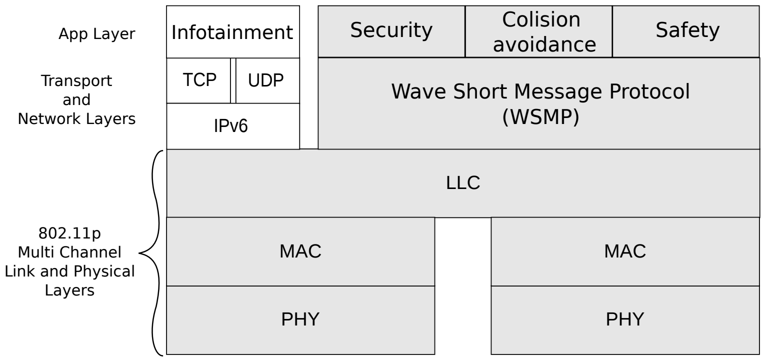

App Layer
Infotainment
Security
Colision avoidance
Safety
Transport and Network Layers
TCP
UDP
IPv6
Wave Short Message Protocol (WSMP)
LLC
802.11p Multi Channel Link and Physical Layers
MAC
MAC
PHY
PHY

Figure 2.2: WAVE Architecture.

discovered route for a message may expire even before the message has been sent to the network.

- Highly variable number of nodes: VANETs' number of nodes ranges from a few nodes in rural areas to thousands of nodes in urban areas. This high variability is a challenge for routing protocols originally designed for other wireless networks.

- Network partitioning: Due to mobility, VANETs tend to partition, thereby creating islands of connectivity. VANET routing protocols should carefully take this issue into consideration.

Some researches have seen in DTNs an alternative to face these issues. Contrary to the *store-and-forward* paradigm used in other networks, the *store-carry-and-forward* paradigm used in DTNs can cope with network partitioning and mobility. In section 2.2 we introduce the DTNs and its application to VNs.

2.1.1 VNs' Standardization

Due to the severe requirements of vehicular safety applications, dedicated protocol stacks have been defined in Europe and the USA for V2V communications. Both standards are similar, using the 5.9 GHz band and relying on the 802.11p protocol for medium access.

The currently approved standard in the USA for ITS is the Wireless Access for Vehicular Environment (WAVE) standard [60]. The Federal Communications Commission (FCC) allocated 75 MHz band at 5.850-5.925 GHz specially intended for ITS. Figure 2.2 shows the architecture of the WAVE standard. WAVE architecture includes two different transport/network layers: one compatible with IPv6 and its own network/transport layer based on the WAVE Short Message Protocol (WSMP), which reduces the overhead by simplifying transport and network layers.

In Europe, under the mandate M/453 of the European Commission (EC), the European Telecommunications Standards Institute (ETSI) has released several standards that regulate MAC and LLC layers, as well as network, transportation and application layers for ITS. This set of standards is known as ITS-G5 [22].

2.1.2 IEEE 802.11p

The IEEE 802.11p standard was standardized by the IEEE 802.11 task group. 802.11p can be seen as a modification of the 802.11a standard. Both protocols operate in the 5 Ghz band and use the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol to access the medium. The main difference between IEEE 802.11p and other IEEE 802.11 protocols is, in IEEE 802.11p, authentication, authorization, and security procedures between stations have been disabled to save time. IEEE 802.11p also disables the scanning of frequencies by the station in order to find available networks. Therefore, the deployment of IEEE 802.11p requires channel allocation.

2.1.3 VNs Deployment

Despite all the standardization efforts, little or none VN systems have been widely deployed. One of the main reasons is that manufacturers have implemented IEEE 802.11p and other VN technologies in the form of OBUs. Since OBUs are usually integrated in the dashboard of vehicles, they are rarely updated. Therefore new technologies are installed only in new vehicles. According to the European Automobile Manufacturers Association, the average age of the car fleet in Europe is 9.7 years, and 34.5% of the automobile fleet in the EU are older than 10 years [33]. Taking these statistics into account, if manufacturers started installing specialized OBUs in every car right now, in the best case, it would take more than 10 years to achieve a penetration rate of about 80%. In addition, experience demonstrates that only luxury cars tend to incorporate these hi-tech devices as standard equipment. Another reason because manufacturers are delaying the deployment of IEEE 802.11p enabled devices is that, since there will not be other vehicles to communicate with, early adopters will not benefit from technology until a considerable number of vehicles implement the technology.

In the academia, some researchers have developed their own vehicular testbeds to test VN protocols: CarTel [52] uses nodes deployed in vehicles and sporadic connections to WiFi open access points for the purpose of monitoring and classifying road surface conditions. However, vehicles do not communicate between them. In [40] researchers from UCLA presented CVet, a VANET testbed deployed over vehicles belonging to the UCLA car fleet. As far as we know, these and other solutions presented for fast prototyping and testing in VANETs are built over dedicated hardware, which increases the cost of deployment and impede their general adoption.

In addition, some companies have already presented their own implementation of IEEE 802.11p based on OBUs. Commsignia [23], Arada [4], and Savari Networks [110], among others have presented their proposals. To allow interaction between their OBUs and external devices, such as smartphones or tablets, manufacturers have developed public Application Programming Interfaces (APIs). However, each manufacturer owns its unique API, therefore, applications designed for a specific OBU are incompatible with other OBUs.

2.2 Delay Tolerant Network (DTN)

The DTN paradigm was initially proposed to enable communication between satellites, surface rovers, and other devices within the InterPlanetary Network (IPN) [57] [12]. Space communication may suffer high delays and frequent disconnections. The DTN concept was also adapted for wildlife monitoring [62] and remote village communication [94], [112]. However, DTN solutions used their own protocols and were unable to intercommunicate. To enable intercommunication between different DTNs, regardless of the network technology, the Delay Tolerant Network Research Group (DTNRG) [26] started to work towards its standardization [34]. Figure 2.3 represents a heterogeneous DTN, which interconnects the IPN with terrestrial DTN nodes. As a result of these efforts, in 2007 two RFCs were published in 2007 that defined the DTN architecture [16] and an application level transport protocol, called the Bundle protocol [111]. The following subsections describes the DTN architecture, the Bundle protocol, and the Convergence Layer that allows the Bundle protocol to run on the top of different network and transport layers.

2.2.1 Architecture and Standards

To support the heterogeneity of different networks, the DTN architecture is designed to run as an overlay network over the network layer (IP in the case of the Internet). To do so, two new layers are added: The bundle layer, and the convergence layer [111]. The bundle layer encapsulates application data units into bundles, which are then forwarded by DTN nodes following the bundle protocol. The convergence layer abstracts the characteristics of lower layers to the bundle layer. The convergence layer does not need to run over the internet protocol stack, thus allowing for the implementation of DTNs over any type of network.

2.2.2 The Bundle Protocol

The Bundle Protocol stores and forwards bundles between DTN nodes. Instead of end-to-end forwarding, the Bundle Protocol performs hop-by-hop forwarding. To

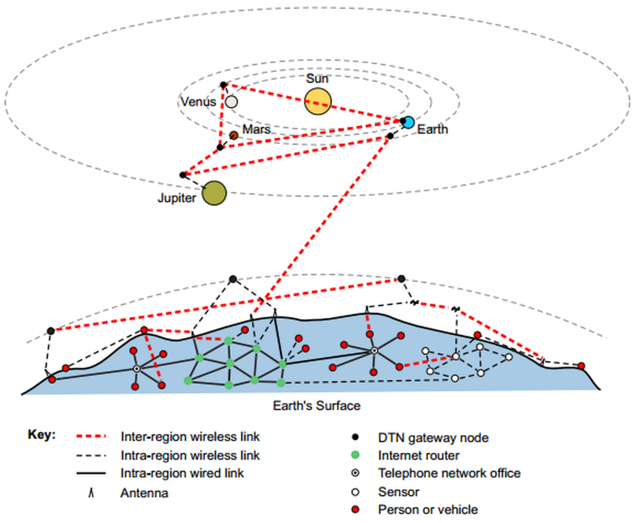


Figure 2.3: Heterogeneous Delay Tolerant Network Example [143].

deal with network disruption, the Bundle Protocol can store bundles in permanent storage devices until a new transmission opportunity appears. The concept of reliable custody transfer ensures that a DTN node will not remove a bundle from its buffer until another node has taken **custody** of it.

The Bundle Protocol operation depends on contacts. A **contact** occurs when a connection between two DTN nodes can be established. The contact type depends on the type of operating network: it may be deterministic, as in Interplanetary networks, opportunistic, as in VN, or persistent, as in the Internet.

When the size of a bundle exceeds the maximum transferred data of contacts, the bundle protocol must perform fragmentation. Fragmentation is supported in two different schemes: proactive, where a DTN node may fragment an application message into different bundles and forwards every bundle independently, and reactive, where bundles are fragmented during transmissions between nodes.

2.2.3 The Convergence Layer

The convergence layer abstracts the characteristics of lower layers to the bundle protocol and it is in charge of sending and receiving bundles on behalf of the bundle protocol. The convergence layer allows for any set of lower protocols to be used to reliably transfer a bundle between two DTN nodes. For example the

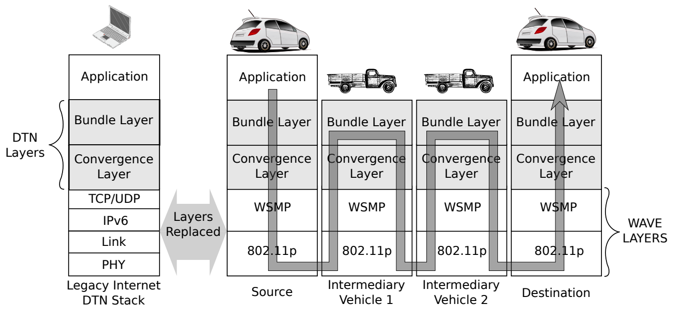


Figure 2.4: Comparison between Internet DTN stack and VDTN stack. Scheme of a message transmission in a VDTN.

TCP/IP convergence layer [27] uses a TCP connection between two DTN nodes to transfer bundles. That TCP connection can be established via the Internet. To implement a DTN over other technologies, new convergence layers are needed. Convergence layers must provide the bundle protocol with a reliable delivery and reception mechanism.

2.2.4 The Generic Opportunistic Routing Framework (GORF)

After the standardization of DTN architecture, the DTNRG focused on the routing protocols, releasing Generic Opportunistic Routing Framework (GORF) [78]. GORF architecture specifies all necessary basic functionalities common for utility-based routing protocols, and it provides a framework to easily define and implement any opportunistic routing protocol for DTNs. To date, only the Epidemic protocol [139] and the PRoPHET protocol [79] have been standardized [77].

The GORF assumes that nodes are able to detect their neighbors using a service running independently. When a neighbor has been detected the protocol sets up a link between the current carrying node, called **custodian**, and the detected neighbor, called **candidate**. Once a link is established, nodes exchange routing information on other nodes in the network. Afterwards, the custodian sends a *bundle offer* that contains a list of the bundles in its buffer. Then, the candidate responds with a list of requested bundles, that will be forwarded to it.

2.3 Combining DTNs and VNs

The characteristics of DTNs make them a feasible solution that can cope with the specific characteristics of VN. The store-carry-forward paradigm of DTN is resilient to quickly route expiration as well as to network partitioning. Because of that, many researchers have proposed to combine VNs and DTNs. The combination of VNs and DTNs, is called Vehicular Delay Tolerant Networks (VDTNs).

The standard DTN protocol stack can be used directly in VDTNs through the IPv6 compatible stack. To implement a pure VDTN directly over the WAVE Short Message Protocol (WSMP), which introduces less overhead and more flexibility, the only requirement is the implementation of a convergence layer between the bundle layer and the WSMP. Figure 2.4 compares the pure VDTN stack against the legacy Internet DTN stack. Few researchers have tried to adopt the standard DTN stack for VDTNs. Among the papers reviewed in the next section, only those proposals which were tested on the UMassDieselNet testbed [11] implemented the standard DTN stack.

With regards to the GORF architecture as it is proposed at present it may be applicable to all of the unicast protocols surveyed in the next section. However, due to its novelty, none of the protocols exactly match the functions and phases defined by the GORF. The main difference arises in the node that performs the routing decision process. Most proposals consider that the custodian node must decide whether or not to forward a bundle, according to its neighbors' characteristics; whereas GORF architecture assigns the routing decision process to the candidate node, which requests bundles stored in the custodian buffer. Since the candidate node may have a different local view of the network status, decisions may be different, and the routing information exchange phase should be appropriately adapted.

In the next three sections, we survey 40 previously proposed VDTN protocols. We not only survey them, but also analyze how they were evaluated, and their suitability to different VN applications.

2.4 VDTN Protocols

In this section, DTN protocols are classified according to different parameters. Firstly, they must be grouped together according to the objective of the protocol: *a)* protocols whose objective is to disseminate messages to all the nodes in the network (**Dissemination**) and *b)* protocols whose messages have a specific destination that can either be a vehicle or an Road Side Unit (RSU) (**Unicast**). Secondly, they are grouped together according to the amount of control information required by each protocol. Inside the dissemination protocols group, we distinguish between the epidemic approach and a group of protocols that uses geographic information to estimate connectivity of nodes (**geo-connectivity**). Inside the unicast group, we distinguish between **zero knowledge** protocols, those that do not require any knowledge about the vehicles status or the environment and **utility based** protocols. Utility based protocols try to estimate the benefit of each transmission (*i.e.* how a transmission improves the probability of reaching the destination) to determine the best forwarding node among neighbors. Each protocol estimates this utility using a pre-defined metric. We have divided these utility-based proposals into five different categories, according to the type of knowledge they need: *i)*

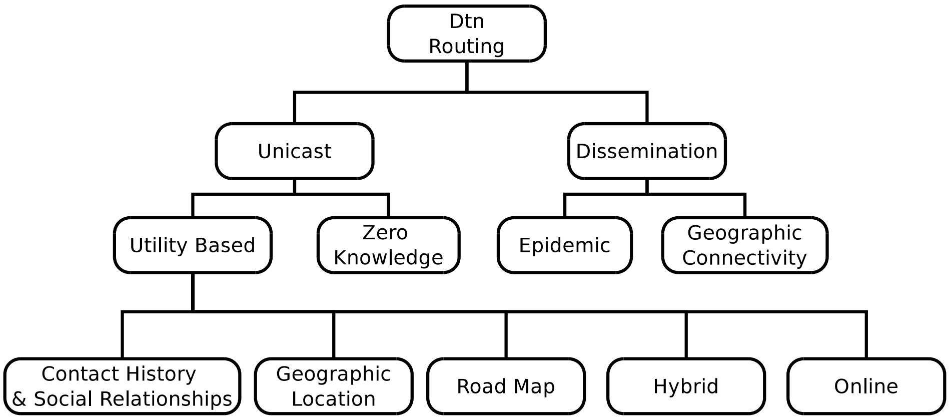


Figure 2.5: DTN Protocols Taxonomy.

contact history & social relationships, *ii)* **geographic location**, *iii)* **road map**, *iv)* **hybrid** protocols, and *v)* **online** protocols. The “online” subcategory includes protocols that, besides combining several simpler protocols, require information on the current state of the road network or use sophisticated metrics that do not fit into any other category. Figure 2.5 summarizes this classification, while Figure 2.6 orders and classifies the surveyed protocols chronologically. For each category, we first enumerate the different protocols and then we explain their advantages and disadvantages.

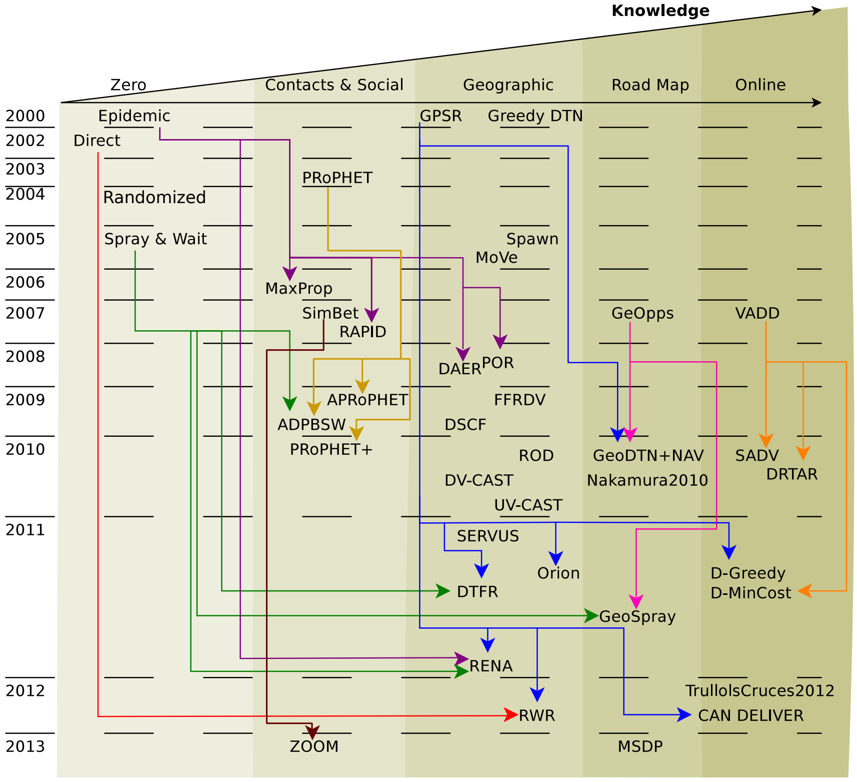


Figure 2.6: Protocols ordered chronologically, grouped by knowledge required. Protocols at the end of the arrows are an evolution of the protocol at the beginning of the arrow.

Table 2.1: Characteristics of different protocols.

Protocol	Year	VN specific	Application	Group	Routing Metric	Optimizations		
						Reliability	Redundancy	Messages Priority
Epidemic [139]	2000	No	Dissemination	Zero Knowledge	–	–	Multicopy	No
ProPHET [79]	2003	No	P2P	Contact History	Contact Rate	No	Open	No
MoVe [71]	2005	Yes	Collect	Geographic Loc.	Direction	No	No	No
Spawn [96]	2005	Yes	Cooperative Download	Geographic Loc.	Distance	No	Fragment Multicoy	Neighbor Dist
Spray&Wait [122]	2005	No	P2P	Zero Knowledge	–	No	Multicopy	No
MaxProp [11]	2006	Yes	P2P	Zero Knowledge	Conctac Rate	End-End ACK	Multicopy	PRoPHET
RAPID [8]	2007	No	P2P	Zero Knowledge	Contact Rate	No	Multicopy	Contact Rate
SimBet [24]	2007	No	P2P	Social	Social Graph	No	No	No
GeOpps [74]	2007	Yes	P2P/V2I	Road Map	Nearest Point, ETA	No	No	No
[41]	2008	Yes	Collect	Zero Knowledge	Direct	No	No	No
POR [75]	2008	No	P2P	Zero Knowledge	Distance	No	Multicopy	Distance
DAER [84]	2008	Yes	P2P/V2I	Zero Knowledge	Distance	No	Multicopy	Distance
VADD [151]	2008	Yes	P2P/V2I	Online	Loc+Density+Speed	No	No	No
DSCF [70]	2009	Yes	Dissemination	Geographic Loc.	Loc+Connectivity	No	No	No
FFRDV [148]	2009	Yes	Dissemination	Geographic Loc.	Speed	Hop ACK	No	No
Infocast [109]	2009	Yes	Dissemination	Zero Knowledge	–	–	Rateless Coding	No
ADPBSW [146]	2009	No	P2P	Contact History	Contact Rate	No	No	No
Adv. ProPHET [145]	2009	No	P2P	Contact History	Contact Rate	No	No	No
Extended GeOpps [73]	2010	Yes	Cooperative Download	Road Map	GeOpps+Estimated Route	No	No	No
C-DTN [19]	2010	Yes	Dissemination	No Data	No Data	Open	Open	–
DvCast [126]	2010	Yes	Dissemination	Connectivity	Loc+Connectivity	No	No	No
ROD [21]	2010	Yes	Dissemination	Connectivity	Loc+Connectivity	No	No	No
Uv-Cast [140]	2010	Yes	Dissemination	Connectivity	Loc+Connectivity	No	No	No
ProPHET+ [51]	2010	No	P2P	Contact History	Buffer+Power+Contact Rate	No	No	No
DRTAR [142]	2010	Yes	P2P/V2I	Road Map	Loc+Density+Speed	No	No	No
GeoDTN+NAV [20]	2010	Yes	P2P/V2I	Geographic Loc.	GPCR+GeOpps	No	No	No
[95]	2010	Yes	P2P/V2I	Road Map	Nearest Point	No	No	No
SADV [28]	2010	Yes	P2P/V2I	Online	Loc+Density+Speed	No	No	No
D-Greedy D-MinCost [114]	2011	Yes	Collect	Online	Nearest Point/VADD like	Hop ACK	No	No
SERVUS [43]	2011	Yes	Dissemination	Geographic Loc.	Loc+Connectivity	Hop ACK	Multicopy	No
DTFR [113]	2011	Yes	P2P	Hybrid	S&W+Location	No	Multicopy	No
Orion [88]	2011	No	P2P	Hybrid	Distance+Contact Rate	No	No	No
RENA [144]	2011	Yes	P2P	Hybrid	Contact rate+S&W	No	No	No
GeoSpray [116]	2011	Yes	P2P/V2I	Hybrid	S&W+GeOpps	No	Multicopy	No
DSRelay [80]	2012	Yes	Cooperative Download	Distance	Direction on Highway	No	No	No
[137]	2012	Yes	Cooperative Download	Online	Predicted Contacts	No	Fragment Multicoy	No
CAN DELIVER [90]	2012	Yes	P2P	Online	Nearest Point	End-End ACK	Multicopy	No
RWR [154]	2012	No	P2P	Hybrid	Distance+RSU	No	No	No
CSM [141]	2013	Yes	Collect	Data Agregation	Geographic Location	No	Yes	No
MSDP [128]	2013	Yes	Collect	Road Map	Nearest Point+Buffer	Hop ACK	Fragment Redundancy	No
ZOOM [152]	2013	Yes	P2P	Social	Social Graph + Contact Rate	No	No	No

Table 2.1 summarizes the characteristics of the different proposals. The second column indicates whether the protocols were originally proposed for VN or not. The third column contains the objective application of each protocol as it is stated in its original publication. The fourth column classifies protocols according to the classification explained previously. The fifth column offers a quick and simple description of the routing metric used by each proposal. Finally, the columns under the "Optimizations" label indicate whether the mechanisms described in Section 2.4.4 were considered in the design of the protocol.

2.4.1 Dissemination Protocols

The objective of the dissemination protocols is to inform as many nodes as possible of an event. The most obvious solution is the simple flooding scheme, where nodes rebroadcast every message received [99]. However, this scheme generates some well-known problems, such as the broadcast storm [97], or infinite rebroadcasting loops that waste resources. To limit the impact of these problems, some modifications to the simple flooding scheme have been proposed [138]. Simple flooding and its modifications are limited by the connectivity of the network: they will only propagate messages as long as the network is connected. In this section we present proposals that add DTN support to dissemination protocols. Since DTN dissemination protocols are not limited by the connectivity of the network, the dissemination process must be limited in time or space to avoid collapsing the network.

Epidemic Protocol

The simplest DTN dissemination protocol is the Epidemic protocol [139], which consists of sharing all the messages in the nodes' buffers every time a contact occurs. The Epidemic protocol needs a *negotiation phase* to determine which messages to share, increasing the delay and generating more overhead than the non-DTN proposals. In dense networks this *negotiation* traffic may be even bigger than data traffic. Moreover, the Epidemic protocol neglects the opportunity of a node *overhearing* a message from broadcast transmissions between neighbors. The Infocast protocol [109] extends the Epidemic protocol with fragmentation and coding, to provide better performance.

Geographic & Connectivity Protocols

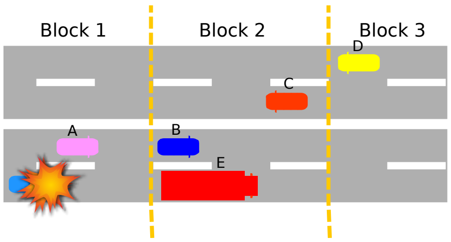
Within this category we include DTN dissemination protocols that need information on node location. This information can be used to limit the number of messages exchanged by nodes and to estimate the connectivity of the network in order to choose the best possible candidate as the new carrier. This carrier will bring the message to the next group of nodes. The protocols matching this definition are: Directional Store-Carry-Forward (DSCF) [70], Fastest Ferry Routing in DTN-enabled Vehicular Ad-Hoc (FFRDV) [148], Road Oriented Dissemination (ROD) [21], Urban Vehicular BroadCast (UV-CAST) [140], Distributed Vehicular BroadCast (DV-CAST) [126], and SERVUS [43].

- The **DSCF** protocol [70] requires every node to have 2 different antennas. It works by following three simple rules; *i)* messages received from one direction are transmitted in the opposite direction, *ii)* if there are no vehicles in the propagation direction, the message is stored in the buffer until a new neighbor appears, and *iii)* any duplicated message is ignored. Apart from the requirement of having two interfaces, which is not considered in the WAVE standard, when several nodes rebroadcast a message they will probably collide when accessing the channel. Moreover, it is limited to highways, where the propagation direction is clearly defined.

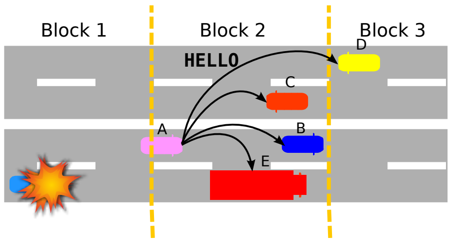
- The **FFRDV** protocol [148] assumes that vehicles are moving on a highway. It divides the road into small blocks, and, when an event occurs, the first vehicle passing by generates a message and becomes its carrier. The carrier broadcasts a beacon message every time it enters a new block. Neighbors inside the same block answer the beacon message with information on their speed and moving direction. Then, the fastest vehicle moving towards the propagation direction is chosen to become the new carrier, while the remaining nodes overhear the message. If no neighbor answers to the beacon, the carrier keeps it in its buffer until the next block. It is clear that, besides the connectivity of the network, the propagation delay depends on the size of the blocks. Moreover, since the FFRDV is invalid for city environments, it must be complemented by other dissemination protocols. Figure 2.7 depicts the behavior of this protocol.

As long as the network is connected, multi-hop forwarding protocols disseminate information faster than store and carry protocols. To take advantage of this characteristic, several protocols use the multi-hop forwarding scheme until they detect a disconnected network. Then, they use geographic information to choose several carriers that will carry the message further.

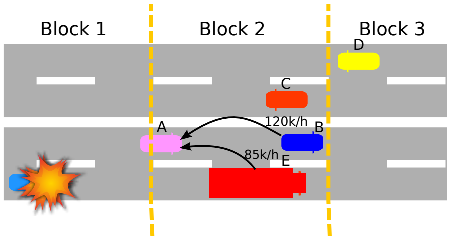
- The **ROD** protocol [21] does not need nodes to periodically send beacon messages. When a node receives a message from another node, it decides whether to retransmit it according to its relative position with respect to the sender. This phase of the protocol is similar to the Distance Defer Transmission (DDT) protocol [123]. If a node detects that none of its neighbors rebroadcasted a message, it switches to *store-carry and forward* mode. In this mode, the node periodically rebroadcasts the message until it detects that another node has also received and rebroadcasted the message.



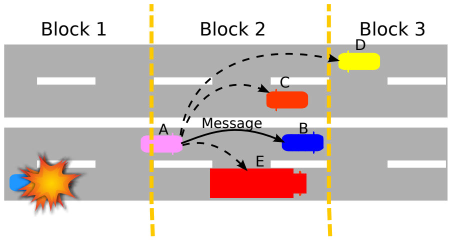
(a) T1: An event is detected and node A becomes a carrier.



(b) T2: Node A enters a new block and broadcasts a beacon.



(c) T3: Only nodes moving away from the event answer the beacon.



(d) T4: Node B is chosen as the new carrier, it will broadcast a beacon as soon as it enters block 3.

Figure 2.7: Example of FFRDV operation.

- The **UV-CAST** protocol [140] defines a Region of Interest (ROI) where the message must be disseminated. The main difference between UV-CAST and ROD lies in how they choose the carrier nodes. While in ROD the selection is only based on overhearing messages from neighbors, UV-CAST nodes use their geographic location information to determine wheter they are boundary nodes for the source node's connected region. To determine if a node must switch to *store-carry and forward* mode UV-CAST follows this process. Suppose node A receives a message from the source (S), with N neighbors (N_i) (Fig. 2.8): *i)* it calculates the angle θ_i between $\overrightarrow{AS}$ and $\overrightarrow{AN_i}$, *ii)* if the sum of the smallest and largest angles is less than π, A must switch to *store-carry and forward* mode. Once in *store-carry and forward* mode, the node will rebroadcast the message and switch to normal mode as soon as a beacon from a new neighbor is received.

- The **DV-CAST** protocol [126] is another example of a highway-limited protocol. As in ROD, nodes are grouped into clusters, and they switch between *normal* and *store-carry and forward* modes according to the estimated connectivity. DV-CAST defines three different operation modes, *well connected neighborhood*, *sparsely connected neighborhood*, and *totally discon-*

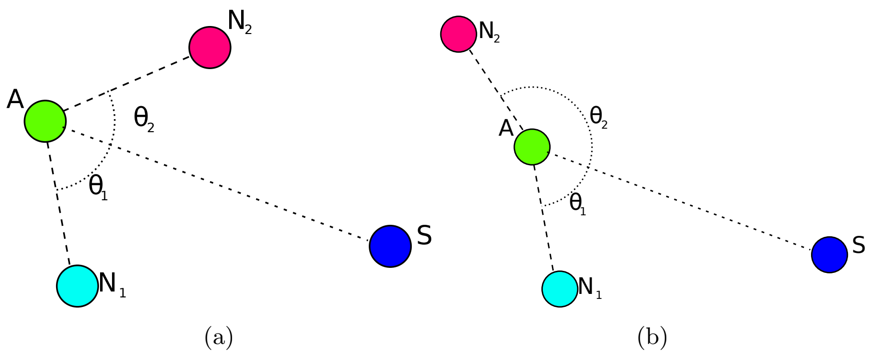


Figure 2.8: UV-CAST example: in a) node *A* switch to SCF mode while in b) does not.

nected neighborhood. In the first mode, nodes work in *normal* mode; in the second mode, nodes switch to *store-carry and forward* mode, when they move contrarily to the message source and, finally, in the third mode, nodes always switch to *store-carry and forward* mode.

- The **SERVUS** protocol [43] follows a similar approach, where nodes modify their behavior according to the location of their neighbors. In SERVUS, nodes detect whether they are the last node of a group of connected nodes, called a cluster, and then they rebroadcast previous messages when they contact a new node from outside the cluster. In SERVUS, cluster detection is only based on the geographic location of neighbors, which is obtained from periodic beacons.

To conclude, in all these protocols, to choose the next carrier node the algorithms assume that all the nodes in the neighborhood have the same information and, therefore, they depend greatly on the correctness of the neighbors list, which can be easily compromised by a high loaded channel and high mobility. Moreover, the calculation of angles and relative locations may be affected by the variability of heterogeneous Global Positioning System (GPS) devices.

2.4.2 Unicast Protocols

Besides pure unicast protocols, we have included in this category those anycast protocols where the destination is any of the RSUs present in the VN, since they are reduced to unicast by choosing the closest RSU as the destination.

The first subgroup inside the unicast protocols category includes those protocols that do not need any external source of information; we call these **Zero Knowledge** protocols. A much larger group includes protocols that estimate the utility of each transmission, *i.e.* how a transmission improves the probability of reaching the destination to determine the best forwarding node among neighbors. For the sake of clarity, we will discuss the **Utility Based Protocols** in a separate subsection (2.4.3).

Zero Knowledge Protocols

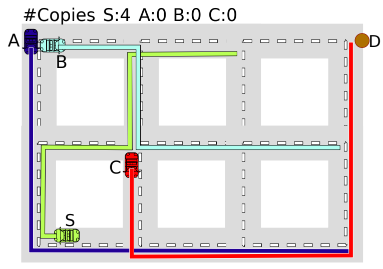
Under the Zero Knowledge category we have included protocols that do not need any external source of information, or to collect information while they are running. As a result of this limitation, their performance is usually surpassed by utility based protocols. Most of them were designed for intermittently connected MANETs [69], but are usually used as a reference for comparison with VDTN protocols. The protocols included in this category are: *Direct* [48], *Randomized Routing* [121], Epidemic [139], and Spray&Wait [122].

- The **Direct** is the simplest possible protocol [48]. It works as follows: a node A forwards a message to a node B only if B is the destination. This case presents an unbounded delay but it has the advantage of performing only a single transmission per message. It represents an upper bound for delay and a lower bound for delivery ratio.

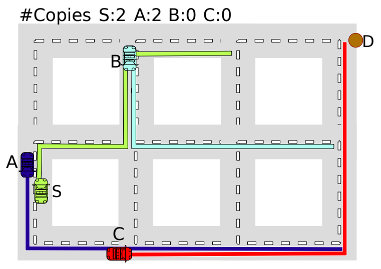
- The **Randomized Routing** protocol was presented in [121]. It works as follows; node A forwards a message to another node B, which A finds with a given probability p. In its work, authors showed that random routing behaves better than direct routing.

- The **Epidemic** protocol [139] has also been applied to the unicast problem. As long as enough resources are available, the Epidemic protocol guarantees that messages will eventually arrive at their destination along the shortest path. Therefore, under ideal conditions, the Epidemic protocol provides a lower bound for delay and an upper bound for delivery probability. The main problem of the epidemic protocol is that it wastes resources by propagating copies of messages that have already been delivered, and along paths that will never reach the destination. In order to limit this resource wastage, researchers have proposed several modifications to the original Epidemic protocol. In [115], authors presented four different mechanisms to block the propagation of already-delivered messages. In [150], nodes exchange a copy of the messages with a probability smaller than 1, which reduces the number of copies in the network. Protocols such as MaxProp [11], RAPID [8], POR [75], and DAER [84] add message priority management techniques to make the most of every contact. We will go into detail about these techniques in Section 2.4.4.

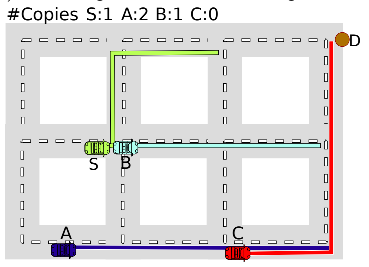
- The **Spray&Wait** protocol [122] divides the propagation of messages into two different phases. Initially it disseminates a certain number of copies of a



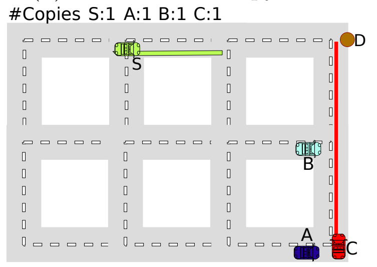
(a) T1: S generates a message for D.



(b) T2: S sends a copy to A.



(c) T3: S sends a copy to B.



(d) T4: A sends a copy to C, which will finally deliver it to D.

Figure 2.9: Binary Spray and Wait example.

message to neighbors and then it waits until any of the carrier nodes moves and reaches the destination of the message. Several spraying mechanisms were presented and studied in [122], where the Binary Spray & Wait (BS&W) protocol offered the best results. In the BS&W protocol, the source of a message initially starts with L copies. Any node A that has $n > 1$ message copies (source or carrier) and encounters another node B (with no copies) hands $\lfloor n/2 \rfloor$ copies over to B and keeps $\lceil n/2 \rceil$ for itself. When only one copy is left, it switches to direct transmission. Figure 2.9 shows this behavior.

In the following subsubsections we will go on to detail as to how some authors adapted these *zero knowledge* protocols to turn them into utility based protocols, as seen for example in [146] and [116].

Since the protocols included in this category do not consider any type of external information, they are suitable for environments where we cannot make any assumption about mobility models, road maps, or social relationships. However, in VDTNs we typically find better alternatives because mobility is restricted to the road network, vehicles are driven following certain rules and people usually live in communities.

2.4.3 Utility Based Protocols

We define the utility function as a function that combines several parameters to obtain an index that estimates how a transmission would increase the probability of reaching the destination of a message (hereafter called the Utility Index). In some protocols the utility function can be as simple as the distance to the destination, while in others it may combine several parameters from different sources of information. In this section we classify utility-based protocols into five different categories according to the type of knowledge they need to obtain the required parameters to calculate the utility index: *i)* **contact history & social relationships**, *ii)* **geographic location**, *iii)* **road map**, *iv)* **hybrid** protocols and *v)* **online** protocols.

Contact History & Social Relationship Protocols

The protocols included in this category work under the assumption that the probability of a node meeting the destination node of a message can be estimated based on the history of previous contacts. Although most of them were developed for MANETs, and are mainly applicable to wildlife tracking systems [62] or pedestrian communities [51] (where the *frequent contacts* paradigm seems to clearly apply). These protocols have been extensively used for comparison with VDTN protocols. In this category, we find the following protocols: PRoPHET [79], APRoPHET [145], PRoPHET+ [51], ZOOM [152], and SimBet [24].

- **PRoPHET**, which was the first *contact history based* protocol, was presented in [79]. This protocol relies on a self-defined *delivery predictability metric*, $P \in [0, 1]$, which is updated according to Equation 2.1, where $P_{(a,b)}$ is the *delivery predictability* that node a has for node b, and P_{init} is an initialization constant. Note that, nodes experiencing frequent encounters have a higher *delivery predictability.*

$$P_{(a,b)} = P_{(a,b)_{old}} + (1 - P_{(a,b)_{old}}) \times P_{init} \tag{2.1}$$

  The defined *delivery predictability* ages (decreases its value) when two vehicles do not meet for a while. PRoPHET also defined the transitivity property for the *delivery predictability*, *i.e.* if node a frequently encounters node b, and node b often encounters node c, node a is a *good* node to forward messages to c. To grasp this behavior, the *delivery predictability* metric is updated in line with Equation 2.2, where β is a constant that quantifies the impact of the transitivity on the *delivery predictability* metric.

$$P_{(a,c)} = P_{(a,c)_{old}} + (1 - P_{(a,b)_{old}}) \times P_{(a,b)} \times P_{(b,c)} \times \beta \tag{2.2}$$

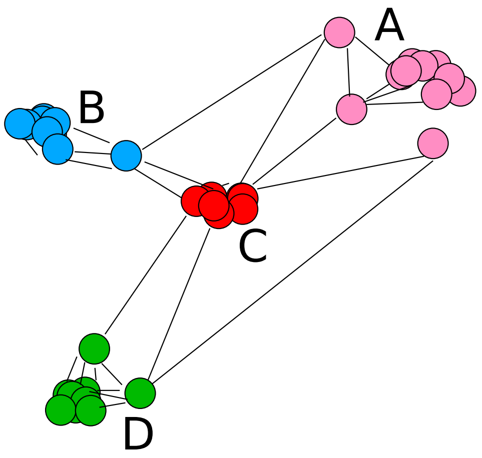


Figure 2.10: Social graph: nodes inside cluster C are connected better than nodes in others clusters.

- The **Advanced PRoPHET** protocol was presented in [145]. It modifies the PRoPHET protocol's metric to smooth its variability. The main effect of the smoothed variability is that the protocol needs more time to react to changes in the network.

- In [51], authors presented **PRoPHET+**, another improved version of the PRoPHET protocol that adds four new parameters related to *i)* buffer (V_B), *ii)* power (V_P), *iii)* popularity (V_O), and *iv)* bandwidth (V_A). Using Simple Additive Weighting, the utility function is defined as follows:

$$V_d = W_B(V_B) + W_P(V_P) + W_A(V_A) + W_O(V_O) + W_{PRoPHET}(V_{PRoPHET}) \tag{2.3}$$

  Where W_i refers to weight factors that must be configured by the users and whose optimal value depends on the scenario. Their results showed that, by considering more variables and not only the contacts history, the performance of PRoPHET is improved. They also proved that a misconfiguration of weight factors may degrade the performance of the protocol.

- In [152] and [24], authors presented **ZOOM** and **SimBet**, which use *social metrics*, such as the node's number of links in the social graph or their centrality, to choose the next forwarding node. They complement the *delivery predictability* by estimating the centrality of the node within the social graph formed by the nodes inside the network. Figure 2.10 shows an example of the relationships inside a community. Nodes from cluster C are better connected than nodes from other clusters, meaning that, those nodes are better carriers.

These routing schemes require a nearly-closed community to be effective: new nodes, which do not have previous contacts, seem to be isolated, and nodes that left the network, which had a long contacts-history, seem to still belong to it for a long time after leaving. Since a high mobility and a highly changing membership are common characteristics of VNs, the protocols studied in this section may tend to select old routes. Moreover, several authors have shown that the average inter-contact time, when applied to VNs, is in the order of several hours or even days [152, 149]. Since the inter-contact time is closely correlated to the *expected delay to destination*, the applications running on top of one of these protocols should expect an end-to-end delay in the order of hours. Finally, when using *social metrics*, the relationships between the nodes need to be carefully analyzed before full deployment, which presents scalability and privacy issues.

Geographic Location Protocols

Protocols included in this section assume that each node is aware of its location and its moving direction. Although we found only two examples of protocols related to VNs that match this exact definition, we decided to create a new category since these can be considered the ancestors of more advanced protocols that, beside location and direction, use other sources of information. Those protocols are Greedy-DTN and MoVe [71].

- The **Greedy-DTN** protocol is closely related to the most referenced geo-assisted routing protocols in literature, GPSR [64] and GPCR [82], which are not delay-tolerant protocols. In GPCR/GPSR messages are forwarded greedily towards the destination, *i.e.*, the best forwarding neighbor is the one closest to the destination. When a message reaches a local minimum, where no neighbor is closer to the destination, it is routed in *perimeter mode* in an attempt to find a new route. GPSR is generally adapted to DTN omitting the *perimeter mode* and carrying the message inside the buffer until a better forwarding node to forward the messages appears. From now on we will refer to this adapted version of GPSR as **Greedy-DTN**. Greedy-DTN has been widely used as a reference for comparison with more sophisticated DTN protocols [74][128].

- **MoVe** [71] is a protocol that estimates the future location of the nodes using their current direction of movement. In MoVe, the node whose estimated trajectory is the closest to destination becomes the best forwarding node. A modification of MoVe, called *MoVe-Lookahead*, uses the location of the next *waypoint* (authors assumed the random waypoint mobility model) to predict

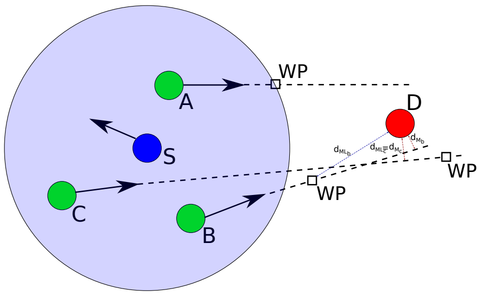


Figure 2.11: In MoVe, only the direction is taken into account to choose the next forwarding node, while in MoVe-Lookahead the way-points are also considered. Therefore, when using MoVe, node S will choose node B to forward a message to D, while it will choose node C when using MoVe-Lookahead.

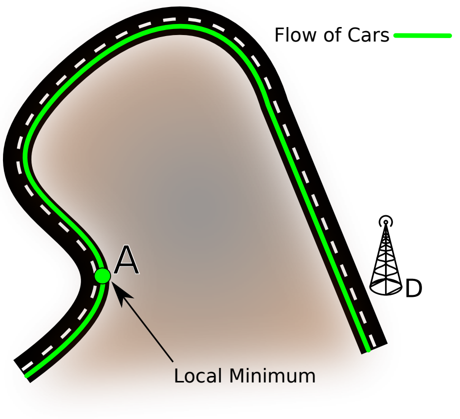


Figure 2.12: When using geo-routing, if a constant flow of vehicles exists, messages for D could get stuck in A because there is a permanent local minimum.

the mobility of the nodes and avoid forwarding messages to nodes that will change their direction before arriving at the closest point to destination. Figure 2.11 shows an example where node B is the best forwarding node when using MoVe, while C would be the best forwarding node when using MoVe-Lookahead.

These approaches are suitable for unrestricted mobility models, but they ignore the fact that mobility in vehicular networks, despite its high variability, is constrained to roads. Therefore, these proposals are prone to inducing suboptimal routing decisions. For example, the Greedy-DTN protocol may get blocked when

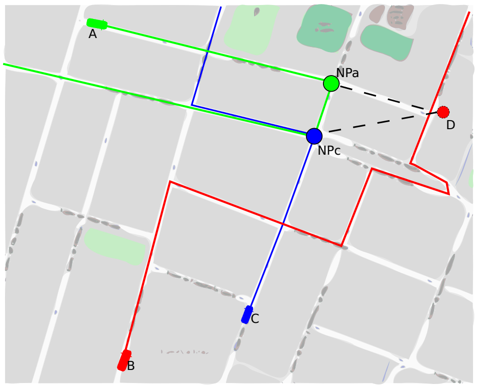


Figure 2.13: Calculation of the NP in GeOpps. Although NP_B is closer to the destination than NP_A and NP_C, A and C nodes are probably better forwarding nodes, since they will reach their NPs faster than B.

a constant flow of vehicles generates a permanent local minimum, as illustrated in Fig. 2.12. Besides, loops occur when two vehicles moving in opposite directions meet. In the case of MoVe, it assumes a random-waypoint model for node movements, ignoring the fact that the current direction of vehicles, especially in downtown or rough rural areas, may change frequently and that it may not match the *long term direction of movement*.

Another problem of geo-assisted protocols is that they require a location service to obtain the destination's location. However, authors usually ignore this requirement. Without a location service, protocols are limited to V2I communication. This problem also affects Road-Map protocols, which are covered in the next subsubsection. The design of a location service is far from trivial and is outside the scope of this thesis.

Road Map Protocols

Since vehicular mobility is always restricted to roads, the use of pure geographic protocols, such as Greedy-DTN or MoVe, can lead to messages being forwarded to vehicles whose *long term destination* is far from the destination of the message. The *long term destination* is important in the case of sparse networks, where vehicles rarely meet. Protocols included in this section assume that vehicles have a Navigation System (NS) that provides information on the road layout and the vehicle's future route, besides an accurate geographic location. The protocols included in this category are: GeOpps [74], and its extension [73] and the protocol presented in [95].

- The **GeOpps** protocol chooses the next-forwarding nodes based on the

Minimum Estimated Time of Delivery (METD) metric, which is the sum of: *i)* the estimated time that a vehicle would need to reach the nearest point (NP) of its route to the destination, plus *ii)* the time a vehicle would need to travel from the NP to the final destination. If the latter factor cannot be calculated, an estimation based on the straight distance can be used [74]. Fig. 2.13 shows an example of the NP calculation.

GeOpps was extended to support traffic from RSU to vehicles [73] by routing the reply message to a point inside the vehicle's route, and then backwards on the vehicle's path until the destination is reached.

- The protocol presented in [95] simplifies GeOpps by ignoring the speed of the vehicle, and selecting the vehicle whose route passes closest to the destination as next carrier.

These protocols emphasize the *store-carry and forward* phase, missing multi-hop communication opportunities, which increases the delay. Moreover, they heavily depend on the reliability of programmed routes, which may present vulnerability.

Hybrid Protocols

In this category we include protocols that combine the behavior of several protocols from those previously presented. The protocols we found were: Average Delivery Probability Binary Spray and Wait (ADPBSW) [146], GeoDTN+Nav [20], Orion [88], GeoSpray [116], Delay Tolerant Firework Routing (DTFR) [113], REgioN-bAsed (RENA) [144], and RWR [154].

- **ADPBSW** [146] combines the PRoPHET protocol with the BS&W protocol. It was originally designed for MANETs and it is the first proposed hybrid protocol for DTNs. It complements the Spray & Wait protocol by using the *delivery probability* calculated by PRoPHET to propagate copies only to vehicles experiencing a *delivery probability* higher than the current carrier.

- **GeoDTN+Nav** is a protocol that divides the process of delivering a message into two different phases [20]. During the first phase it uses GPCR to forward the message near to the destination. Once a local maximum is reached, the protocol switches to perimeter mode. Contrarily to GPCR, in GeoDTN+Nav, after a certain number of hops in perimeter mode, the protocol switches to DTN mode, and the message is delivered using the GeOpps protocol. The vehicle switchs back to GPCR phase if it finds a neighbor closer to the destination than the previous local maximum that triggered the switch to DTN mode.

- Similar to GeoDTN+Nav, the **Orion** routing protocol [88] combines the Greedy-DTN protocol with a contact history based protocol. Therefore,

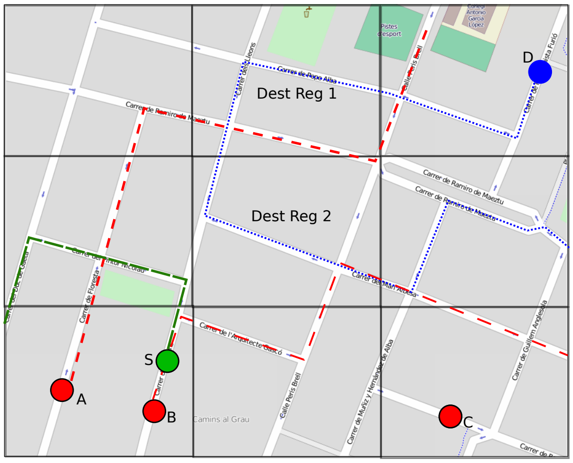


Figure 2.14: RENA: To send a message from S to D, two copies will be sent to A and B, which will later distribute them in the destination regions.

messages are forwarded greedily until a local maximum is reached. Afterwards, the message is scheduled to be forwarded to the vehicle with the highest delivery probability.

- The **GeoSpray** routing protocol [116] combines the S&W multi-copy scheme with the GeOpps protocol. Similarly to S&W, L copies of every message are distributed through the network. Then, instead of waiting until carriers arrive to the destination, the copies are propagated using GeOpps.

- The **DTFR** protocol [113] forwards a message to the destination greedily. However, the *target destination* differs from the actual destination of the message. The *target destination* depends on the phase of the protocol and changes step-by-step, combining phases similar to S&W with pure Greedy-DTN phases. If, at any time during any of the phases, a vehicle finds a path to the destination, it uses that path to deliver the message.

- The **RENA** protocol [144] combines the Spray & Wait and the Epidemic protocols. It divides the map into regions and calculates the probability of moving between them for every vehicle. Additionally, it estimates the probability of being inside a given region for every node. Then, the routing process is divided in four phases. When a message is generated, *i)* it distributes n copies to vehicles that will probably travel to regions where the destination vehicle is likely to be located, *ii)* those copies are forwarded to vehicles that have a better probability of reaching the destination region than the current carrier, *iii)* once the message has arrived at the destination regions, m copies are distributed to vehicles with a low probability of leaving the region, and *iv)* these copies are forwarded to vehicles with a smaller return time to the destination region until the destination is found. The main advantage of RENA, when compared with other replication mechanisms, is that RENA limits the replication of the messages to the destination regions. Figure 2.14 illustrates this behavior.

- Finally, **RWR** [154] combines pure multi-hop geo-routing with an alternative technique where messages are delegated to an RSU. It estimates the expected delay of a message using GPCR and using RSU delegation to choose between these two alternatives.

The protocols included in this section have the advantages and some of the disadvantages of the combined protocols. For example, GeoDTN+Nav benefits from the typical low delay of GPCR for multi-hop routing and the low message loss ratio of DTN, but it consumes more resources than GeOpps; GeoSpray, probably performs better than its predecessors, the original GeOpps protocol and BS&W, at the cost of consuming more resources; and RENA is clearly expected to waste fewer resources than epidemic routing. However, their implementation is complicated and depends on many user-defined parameters, which may lead to incompatibilities.

Online Protocols

Under the name of *Online Protocols* we have included protocols that need information on the current state of the road network, for example, number of nodes, average speed of the nodes, and congestion of every road. Some of them are also hybrid protocols that combine these new metrics with modifications of protocols we have previously reviewed. The list of protocols included in this section is: Vehicle-Assisted Data Delivery (VADD) [151], Static-Node-Assisted Adaptive Data Dissemination in Vehicular Networks (SADV) [28], Distributed Real-time Data Traffic Statistics Assisted Routing (DRTAR) [142], D-Greedy and D-MinCost [114], the protocol presented in [137], and CAN DELIVER [90].

- In [151], authors presented **VADD**, which allows vehicles to send messages to an RSU. The routing process in VADD is divided in four steps; *i)* it estimates the travel time of a message for each road taking into account the vehicles density of the road, its length and the duration of traffic lights. Then *ii)* it calculates the shortest path to the destination using Dijkstra, *iii)* it routes messages between road intersections using the Greedy DTN protocol and finally *iv)*, when a threshold distance to the destination is reached, it routes messages using GPSR [64]. Every node traversed by the message recalculates steps *i* and *ii*. To obtain information on road density, duration of traffic lights, maximum speed of roads, etc, required in step *i*, a database containing this information is preloaded. The authors presented three variations of VADD that differ on how they route messages inside crossroads. L-VADD routes messages based on location, while D-VADD uses the direction of the vehicles. H-VADD combines L-VADD and D-VADD, switching from the first to the second when a loop is detected. The main problem of VADD is that it clearly tends to use the most heavily populated roads, which may congest the network.

- The **SADV** protocol [28] complements the VADD protocol by installing static nodes at intersections. It routes packets like VADD, although inside intersections, when no vehicle in the shortest path is found, the message is stored in static nodes until a vehicle in the shortest path appears. A more general, but similar, architecture, where the routing protocol between static nodes is not specified, was proposed in [67]. From our point of view, inside cities, the increase in cost from backbone-disconnected static nodes to fully-connected RSUs is negligible compared with the deployment cost. Therefore, we believe that once static nodes are deployed, it is a better option to connect them to the backbone than to simply use them as static relays.

- The **DRTAR** protocol [142] is similar to VADD, but it uses a distributed data traffic statistics service to obtain information on road status. In addition, in DRTAR, the shortest path is only calculated by the first node, which attaches it to the message. The shortest path is then only recalculated when the current carrier cannot find a neighbor inside the attached shortest path. Other authors have also proposed different distributed data traffic statistics services [10], which show the feasibility of this approach.

- In [114], authors presented **D-Greedy** and **D-MinCost**, two DTN protocols for traffic-monitoring in vehicular networks. As far as we know, this is the first paper to introduce a routing protocol that does not try to minimize the delay from source to destination, but minimizes the consumed resources while ensuring that the collected information meets certain maximum delay requirements. Authors defined two operation modes, multi-hop forwarding (MF) mode and the DTN mode (DM). During MF mode messages are forwarded using Greedy-DTN through the shortest path to destination, while in DM mode messages are only forwarded at intersections to keep them inside the shortest path when the current carrier moves away. The only difference between D-Greedy and D-MinCost is that, in the former, only local and map layout information is available, while in the latter the current road status information is also available. Therefore, D-Greedy calculates the shortest path to destination based solely on road lengths, while D-MinCost also takes into account the road's vehicle density, like VADD. Once the shortest path is calculated, both protocols estimate the delay of the message using MF, as well as DM. Afterwards, it uses the DM as long as its estimated delay is less than the Time To Live (TTL), switching to MF in all other cases. Since MF mode is much faster than DM, both modes tend to alternate, thereby minimizing the number of hops to the destination.

- In [137], authors presented a routing protocol for delivering data from RSUs to vehicles. In their protocol vehicles make requests while they are connected to an RSU. The answers are usually larger than the requests and, therefore, cannot be downloaded during the period while they are connected. Authors proposed the use of other vehicles to deliver the answers to the destination vehicles. They assumed that all RSUs are connected via a backbone network, and that, based on empirical data, contacts between vehicles can be predicted. With this information, their protocol uses other vehicles as carriers for answers.
- Finally, the **CAN DELIVER** [90] protocol allows routing of messages from vehicles to RSUs and vice-versa. In the former case, the vehicle calculates the shortest road-path to the RSU and attaches it, together with information on its own route and speed to the message. Then, the message is forwarded between the intersections using the Greedy DTN protocol. In the latter case, RSUs try to estimates the vehicle's location using the information from the vehicle previously attached to the message. Once the future location is estimated, an area around it is defined, and the reply message is forwarded to it using a scheme that combines the S&W multi-copy scheme with the Greedy-DTN forwarding metric. When the message reaches the estimated area, vehicles switch to a limited epidemic mode and broadcast the message inside that area. To avoid broadcast storms, vehicles only broadcast each message once. If a vehicle outside the estimated area receives a message from inside it, it must be dropped.

Online protocols require a complex platform formed by RSUs, information servers, databases, etc., increasing the implementation and deployment cost. Moreover, they depend on real-time information, which is easily available in simulations but can be difficult to obtain in real implementations.

2.4.4 Common Basic Mechanisms

Our DTN protocol taxonomy is based on the criteria used to select the next forwarding node, also called the routing metric. However, this is not the only element that can make a difference in the performance of different protocols. A set of mechanisms that define the hop-by-hop and the end-to-end communication schemes can heavily influence the delivery ratio, the delay or other important performance metrics. Generally, these mechanisms can be applied to any utility based protocol. In this section we cover the most representative mechanisms available in the bibliography addressing: reliability, redundancy, path diversity and message priority. We introduce these concepts, provide some examples of protocols that use them and measure their impact on performance.

Reliability

The reliability of a protocol is the degree of guarantee that the protocol provides to the sender with respect to the delivery of messages. The typical mechanism used to provide *end-to-end* reliability in non-DTN networks is the use of ACK messages to confirm that messages are correctly received. VDTN protocols use *hop-by-hop* reliable mechanisms. By using *hop-by-hop* ACK messages, the protocols ensure that a message will be kept in the buffer of the vehicle until another vehicle confirms its reception. This mechanism does not explicitly ensure the reception of the message by its destination, but it does ensure that a message will eventually reach its destination if no node failure occurs (node shutdown or buffer overflow). Most of the protocols covered in this chapter simply ignore the impact of reliability. Those that consider and use it are: DTFR [113], D-Greedy and D-MinCost [114] and CAN DELIVER [90].

The impact of *hop-by-hop* ACKs increases with the number of hops. For example, if a message traverses 6 hops and the Packet Error Rate (PER) is 10^{-1} (quite optimistic in wireless communications [89]), the *end-to-end* PER would be $1 - (1 - 10^{-1})^6 = 0.46$, which is an unacceptable value. Since the PER increases with the distance between transmitters, protocols that tend to select the furthest node as the forwarding node, face higher transmission losses and can heavily benefit from the use of *hop-by-hop* ACKs. Obviously, the use of ACKs increases both the load of the channel and the delay experienced by the messages but, from our point of view, it is a small price to pay compared with its advantages.

Since there is not a specific *destination* in dissemination protocols, the concept of *reliability* changes. In dissemination protocols we consider reliability as the capability of the protocol to guarantee that at least one of the nodes inside the Region of Interest (ROI) will disseminate the message until it expires. This feature is usually implemented as follows: *i)* the current carrier broadcasts the message, *ii)* after broadcasting the carrier keeps sniffing the channel to check if a neighbor has rebroadcasted it, *iii)* the transmission confirmation is implicit when a neighbor has rebroadcasted the message. All of the dissemination protocols included in this section implement this mechanism.

Fragmentation and Redundancy

The objective of fragmentation is to provide flexibility to routing. In VDTN, the duration of the contacts limits the amount of data that two nodes can exchange. When a connection between two nodes breaks, the message being transmitted has to be discarded by the receiver and enqueued again by the transmitter, thus wasting the resources used for that transmission to date. In the case of messages of a large size, the amount of wasted resources can be high.

The use of fragmentation allows for redundancy fragments to be added using Forward Error Correction (FEC) techniques. This means that, if a message needs N fragments, $N * \alpha$ fragments will be sent, where the redundancy factor alpha is greater than 1 and depends on the configuration. At the destination, only N fragments are needed to reassemble the original message. This type of redundancy is usually called *coding*, and it reduces the impact of possible losses. The cost of coding depends on the amount of extra fragments sent. Fragmentation and coding only appear in one of the protocols we have reviewed, which is CAN DELIVER [90].

A more aggressive type of redundancy consists of sending multiple copies of the same message. This mechanism is much simpler than coding, but it also consumes more resources. Moreover, it does not solve the problems arising from large-sized messages. This redundancy mechanism is much more common and is used in the following protocols: Epidemic [139], PRoPHET [79], Spray&Wait [122], MaxProp [11], RAPID [8], DAER [84], POR [75], ADPBSW [146], DTFR [113], GeoSpray [116], RENA [144] and CAN DELIVER [90].

Message Priority

By message priority we refer to the order in which messages are forwarded to another node when a contact occurs. This is important, as the duration of contacts is limited. In the bibliography, some protocols have extended the Epidemic protocol to consider message priority: MaxProp [11] and RAPID [8] prioritize those messages with a better transmission delivery probability according to PRoPHET, while POR [75] and DAER [84] prioritize those messages that will get closer to their destination. Although we were unable to find more examples of this mechanism, it may be implemented to complement and improve the performance of any protocol.

2.5 DTN Based Applications for VN

In this section we introduce applications proposed by the research community that depend on the use of DTNs. We describe them presenting some of the problems and challenges they must face. We start this classification with the most frequent application in the reviewed articles, **Peer-to-Peer (P2P)** communication. Secondly we present what we call **environment-sensing applications**, which consider the use of DTN protocols in order to collect information using vehicles as sensors. The third group includes **dissemination applications**; beside broadcast dissemination, we also consider context-based dissemination. Finally, we explore **collaborative content-downloading applications** and new proposals such as cellular offloading. Table 2.1 classifies each of the protocols analyzed previously in

Table 2.2: Grade of Suitability of Protocols to Different Aplications.

Application \ Group	Zero Knowledge	Contacts History & Social	Geographic Location	Road Map	Online
P2P & I2V	1	5	2	3	4
V2I & Sensors Collecting	1	2	3	4	5
Cooperative Downloads	1	5	4	3	5
Dissemination	3	5	4	5	3
					less suitable 1-5 more suitable

each of these categories. For each application described, we provide some examples of its utilization and discuss which group of protocols best adapts to it.

Table 2.2 quantifies the suitability of each group of protocols for each application according to the criteria explained in this section.

2.5.1 P2P Applications

The most obvious application of any communications system involves allowing users to exchange messages and information between them. Hence, it is not surprising that the majority of the analyzed articles focus on “P2P” communication.

As stated in previous sections, when using geographic protocols for P2P communication we need a Location Service to obtain the location of the destination of a message. Table 2.1 shows that 22 out of 41 works are labeled as “P2P” or “P2P/V2I”. The second label includes protocols that are presented as a “P2P” protocol, but obviate the complexity of the required location service, which makes the communication between vehicles impossible, thereby reducing them to V2I communication protocols. We have grouped V2I applications with environment-sensing applications, due to their similarities.

The typical example of a P2P application is a kind of e-mail system, where users can exchange personal messages. Obviously, this scenario application assumes that the sender and the receiver have met previously. We can also assume that the number of users of this application is relatively small (dozens of individuals), compared to the number of vehicles that typically form a VN (thousands of nodes). Given these assumptions, we believe that contact rate and social relationship-based protocols are the best alternative for this application.

If the cost of infrastructure deployment is affordable, it is probably a better option to deploy a set of RSUs connected by a backbone network and then use them to slice the source-to-destination routing problem into two smaller problems: routing from source to an RSU, and routing from another RSU to the destination. This scheme is similar to the one described in [90].

2.5.2 V2I and Sensing Applications

In V2I applications the objective is to send information from a vehicle to an RSU. In environment-sensing applications, the main objective is the same, but it can be assumed that the information is typically correlated to the geographic location of the source.

An example of a V2I application is the scenario where a user wants to order a large number of goods in a shop. Using VDTNs, the user can send a message to the shop, which will be able to prepare the order in advance. In the second case, we envision a scenario where traffic management and road security authorities collect information on speed, road status or weather from vehicles. This information can be used to optimize emergency vehicle routes, monitor pollution inside cities, plan taxi routes, etc.

Since RSUs have a fixed location that can be stored in a quasi-static database, geographic, road map, and online protocols do not require a location service to route messages to its destination. This feature is used in protocols such as GeOpps [74], GeoDTN+Nav [20] or MSDP [128]. In [141], authors introduce a new scheme where messages from different nearby sources are combined to compress their information and reduce the channel load. As stated before, one of the key issues of zero knowledge protocols is that node mobility increases the probability of reaching the destination of a message. Since RSUs are static, zero knowledge protocols are not suitable for these applications. A similar problem applies to contact history and social based protocols. Since they require nearly-closed communities, they tend to ignore nodes that pass by a given region.

2.5.3 Dissemination

Dissemination applications aim to quickly deliver information to as many nodes as possible. In this scope, the adoption of delay-tolerant protocols may seem counter-intuitive since the expectable delay is rather high. Nevertheless, in sparse networks where the degree of node connectivity is low, the *store-carry and forward* paradigm may be the only method capable of guaranteeing a high message delivery ratio.

Accidents occurring on highways represent a typical scenario where quick message dissemination may be useful, for example by notifying drivers approaching the accident area and thereby avoiding cascading car crashes.

When disseminating information, an ROI where a message must be disseminated is typically defined. The ROI is usually related to geographic or road network restrictions, being mostly useful to vehicles moving towards an accident, vehicles moving on streets adjacent to a traffic jam or vehicles ahead of an ambulance route, for example. The strong relationship between the ROI and the actual characteristics of the road environment makes geographic and map based protocols the most suitable alternatives for this application.

Other cases, for example when disseminating non-geographically correlated information (e.g. advertisements), the best socially-connected nodes would probably be the best carriers.

2.5.4 Cooperative Download

In cooperative download applications, the main data flow occurs from RSUs to vehicles. Typically, a user requests data that is too large to be transferred during a single contact with an RSU. To solve this problem RSUs which are connected to a backbone inject fragments of the responses into the network. Once fragments are injected, there are two main alternatives: to distribute these fragments between every interested node [96] or to deliver them only to its specific destination [137], [73], [80].

When distributing fragments to every interested node, it is usually easy to identify social relationships between interested nodes and this information can be used to maximize the protocol performance. On the other hand, geographic information can also be useful to select the best contact, as in [96].

When delivering a message to its specific destination, it can be seen as a P2P communication between an RSU and a mobile node and, therefore, we can apply the same methods as for P2P applications.

2.6 Evaluation of VDTN Protocols

Since developing and conducting real implementation and tests for VNs is an expensive task in terms of time, personnel, and money, researchers have focused on simulations to evaluate and compare the performance of different protocols. However, on analyzing the reviewed articles, we have found a balanced mix of different simulation models that complicates the comparison of results. Moreover, very rarely do works evaluate the same metrics under the same scenarios, which totally invalidates any comparison among results from different works.

Table 2.3 summarizes how researchers evaluated their proposals in their works. The first column includes the name of the proposed protocol. The second column shows the different metrics measured during the evaluation of each proposal. The third column specifies the simulator used for this evaluation. The fourth and fifth columns contain the MAC and radio channel models they used. The sixth column briefly describes the simulated scenario. Finally, the last column shows the number of DTN protocols compared to justify every new proposal. As stated in Table 2.3, we found that most researchers did not compare their proposal against any other DTN protocol (14 out of 41 papers) and that a large group of researchers compared their proposal against only one previously proposed protocol (12 out of 41 papers). This unfortunate situation is a consequence of the mix of available simulation models, as well as the commonly vague description on low-level protocol details, as it is explained in Subsection 2.4.4. Moreover, researchers do not usually offer the source code of their proposals, which complicates the replication and validation of their experiments.

In this section, we first list the metrics evaluated by researchers discussing their relevance. Second, we provide an overview of the models and tools used by the research community to evaluate VDTN protocols and identify the most advanced solutions.

2.6.1 Evaluated Metrics

When introducing a new proposal, researchers need to justify the performance improvement by comparing metrics among different protocols. We have found that the most commonly evaluated metrics are:

- The Delivery Ratio (DR), which is given by the ratio of the number of successfully received messages and the number of sent messages. Since delivering messages to their destination is the task of a routing protocol, the DR is the most important metric when evaluating such a protocol. However, researchers must find a trade-off between resource consumption and effectiveness.
- The Average Delay (AD), which is given by the average time needed to deliver a message. In DTNs, this metric may be heavily influenced by a small number of high delay measurements and, therefore, its value is not representative of the general behavior of a protocol.
- The Delay Cumulative Distribution Function (DC), which illustrates the distribution of the delay experienced by messages. Since the average delay is heavily influenced by messages experiencing long delays, this measurement provides a better idea of the performance of a protocol.
- The Overhead (O), which measures the amount of extra bytes needed per delivered byte. This is a very important metric when evaluating VDTN protocols because part of the network may become easily saturated.
- The Average Number of Hops (H) traversed by a message. This measurement provides an idea of resource consumption. As a general rule, more hops means more consumed resources. However, fewer hops usually implies longer carrying phases, increasing the average delay of the messages.

Table 2.3 includes in its first column the different metrics evaluated in each paper.

Table 2.3: Evaluation of Different Protocols.

	Measurements	Simulator	Medium Access	Radio & Channel	Simulation Scenario	#Compared
Epidemic [139]	DR AD DC	Ns2	No, only contacts	Fix Distance	Random	0
ProPHET [79]	No evaluation					
MoVe [71]	DR AC DC O	Custom	No Data	No Data	Limited random in city map and traces	1
Spawn [96]	O H	Nab	CSMA/CA Model	No Data	One Direction Highway	0
Spray&Wait [122]	DR AD O	Custom	Sloted Collision Detection	No Data	Random Way Point	2
MaxProp [11]	DR AC H	Custom	No, only contacts	No Data	Real & Synthetic Traces	0
RAPID [8]	DR AD O	Custom	No, only contacts	Fix Distance	Real Traces, Contact Model	1
SimBet [24]	DR DC O H	Custom	No, only contacts	Contacts Traces	Real Traces	2
GeOpps [74]	DR AD DC O H	Omnet++, MF	CSMA/CA Model	Nakagami, No Obst	Synthetic Realistic Traces	2
Direct [41]	DR AD DC	Custom	No Data	Fix Distance	RandomWay Point in Grid & Implemented	0
POR [75]	DR AD DC	Custom	No, only contacts	Fix Distance	Real Traces	4
DAER [84]	DR DC H	Custom	No, only contacts	Fix Distance	Real Traces, SUMO	0
VADD [151]	DR AD O	Ns2	CSMA/CA Model	Fix Distance, Interferences	Limited Random in city map	1
DSCF [70]	DC	Custom	No, only contacts	Fix Distance	Grid Map, Random mobility	0
FFRDV [148]	DC	Ns2	No, Only contacts	Fix Distance	Highway, Car Following Model	1
Infocast [109]	DR	Ns2	No Data	Fix Distance	Only 1 road	0
ADPBSW [146]	DR AD	ONE	No, Only contacts	Fix Distance	Limited random in city map	1
Adv. ProPHET [145]	DR AD	ONE	No, Only contacts	Fix Distance	Limited random in city map	1
C-DTN [19]	DR AD	QualNet	CSMA/CA Model	Fix Distance, Interferences	Car Following Model in a Grid	0
DvCast [126]	DR DC O	Ns2	No Data	Ricean Fading	Circular High Way	0
ROD [21]	DR	Airplug-ns	CSMA/CA Model	Fix Distance, Interferences	VehicleMobiGen	2
Uv-Cast [140]	DR O	Ns2	CSMA/CA Model	Fix Distance, Interferences	Real City, SUMO	0

DR=Delivery Ratio; AD=Average Delay; DC=DelayCDF; O=Overhead; H=Number of Hops

Continued on next page

Table 2.3 – *Continued from previous page*

	Measurements	Simulator	Medium Access	Radio & Channel	Simulation Scenario	#Compared
ProPHET+ [51]	DR DC	ONE	No, Only contacts	Contacts Traces	Real Traces	1
DRTAR [142]	AD	Custom	CSMA/CA Model	Fix Distance, Interferences	Limited random in city map	0
GeoDTN+NAV [20]	DR AD H	QualNet	CSMA/CA Model	Fix Distance, Interferences	VanetMobiSim	0
[95]	DR O	Custom	CSMA/CA Model	Nakagami, No Obst, Interferences	NETSTREAM	1
SADV [28]	AD O	MatLab	No, Only contacts	Fix Distance	Limited random in city map	1
D-Greedy D-MinCost [114]	DR AD DC O	Custom	No, Only contacts	Fix Distance	Synthetic Realistic Traces	2
SERVUS [43]	DR O	Ns2	CSMA/CA Model	Fix Distance, Interferences	Grid, Random	0
DTFR [113]	DR AD	Custom	Slotted mac	Fix Distance, Interferences	Limited random in city map and traces	3
Orion [88]	DR AD H	Omnet++	No Data	No Data	Random	1
RENA [144]	DR AD	ONE	No, Only contacts	Fix Distance	Limited random in city map and traces	4
GeoSpray [116]	DR AD O	ONE*	No, Only contacts	Fix Distance	Limited random in city map	4
DSRelay [80]	DR	Custom	No, Only contacts	Fix Distance	Highway, Random Speed	1
[137]	DR OM	Custom	Slotted mac	Fix Distance	Synthetic Realistic Traces	0
CAN DELIVER [90]	DR AD DC O	Ns2	CSMA/CA Model	Nakagami, No Obst	Real Map, SUMO	3
RWR [154]	DR AD DC	Custom	No, Only contacts	Fix Distance	Real Traces	2
CSM [141]	Error Estimation	Custom	No, Only contacts	Fix Distance	Real Traces	1
MSDP [128]	DR AD DC O	Omnet++,INET	CSMA/CA Model	Nakagami, W. Obst	Real Map, SUMO	2
ZOOM [152]	DR AD O	Custom	No, Only contacts	Contacts Traces	Real Traces	3

DR=Delivery Ratio; AD=Average Delay; DC=DelayCDF; O=Overhead; H=Number of Hops

2.6.2 Simulators and Models

The choice of a certain simulator should not influence the results of simulation studies, but it commonly implies the use of a certain set of models and default values. Through the reviewed papers, we clearly identify a worrying trend: 18 works out of 41 used a custom simulator. The use of a custom simulator complicates or almost prevents proper comparison among different proposals. Moreover, it also complicates the peer reviewing system and code reutilization, slowing the developing pace. On the other hand, we have found four different event-driven simulators that have been previously validated and are long-established among the networking community: Ns2 (8 times), The ONE (5 times), OMNeT++ (3 times) and Qualnet (2 times). Below, we briefly describe the characteristics of different simulators.

- The Ns2 simulator integrates advanced propagation and channel models (Nakagami fading and shared channel), medium access (CSMA/CA) and mobility models (traces generated using SUMO) [98]. However, only one of the reviewed proposals used the most advanced features of Ns2 [90]. Three of the articles that used Ns2 neglect the effects of propagation and interferences, while remaining articles used a deterministic propagation model combined with an interference model.

- *The ONE* is a contact-oriented simulator[65]. As far as we know, it is the only simulator specifically designed for DTN, speeding up the development and implementation of new protocols. At present, it does not support propagation or channel models and the mobility model is limited to map-constrained random mobility or real traces, although it is easily extendable. Due to its simplicity, *The ONE* is significantly faster than other simulators. We would recommend it for early research stages, to evaluate the logic of different proposals and to test whether they have major drawbacks, such as local minimums where messages get stuck. We believe that *The ONE* may be easily extended to implement car following mobility models and a non-deterministic propagation model.

- Veins [118], for OMNeT++ [100], is currently the most advanced simulation framework for VN simulation. It implements a complex propagation and interference model and a fully featured medium-access model based on the 802.11p standard, with support for advanced driving models provided by SUMO. However, none of the reviewed works used this framework. In [128], authors used the INET framework [55], whose medium access model is limited to 802.11a/b/g. In [74], authors used a framework that was later integrated in the Inet framework. Because of the fine-grain simulation provided by OMNeT++ and Veins, it consumes a lot of resources in terms of memory, CPU and time, making unaffordable simulations with thousands of nodes.

- QualNet is a non-opensource simulator. Therefore, the correctness of its models cannot be easily verified. It implements a 802.11 medium-access model and a complex propagation model, as well as an interference channel model. It supports the use of trace-based mobility models, which can be obtained from mobility generators such as SUMO or VanetMobiSim. The models implemented in QualNet are less advanced than the ones implemented in Veins.

When simulating network protocols, models are more important than the simulators [107, 1, 84]. In the following subsections, we go through the models used by researchers to evaluate their proposals. However, they do not seek to be a survey on Inter-Vehicle Communication (IVC) simulation models, which can be found in [61].

2.6.3 Radio and Medium Access Models

Radio propagation models for VN must reflect the effects of path loss, shadowing, and multipath fading. The path loss defines the average received power at certain distance from the transmitter, while shadowing and multipath fading add a random component related to obstacles between the transmitter and the receiver, and the multiple delayed replicas of the signal received. A more extended discussion of these effects is not included in the scope of this chapter, and can be found in [44].

Only considering the effects of path loss results in a deterministic propagation distance, which is far from a realistic scenario, we have found that 26 out of 41 reviewed papers use a deterministic propagation model. Considering a deterministic communication range between neighbors has overly optimistic effects on the performance evaluation of the protocols.

More recent works have incorporated the effects of fading into their propagation models [74, 126, 95, 90], which is closer to propagation behavior in real environments. However, only one of the reviewed papers [128] considers the effects of buildings and obstacles when simulating urban scenarios.

In terms of interference models, we have found that only 8 works considered the effects of interference between neighbor nodes.

In our research we have found that some papers (5 out of 41) ignore or do not specify the radio propagation and channel models used. We firmly believe that the VDTN research community should make an effort to improve the quality of the propagation and channel model used to evaluate protocols.

Besides the propagation and channel model, as shown in [30], it is also important to use a fully featured IEEE 802.11p model. However, none of the reviewed papers used such an advanced model. The most advanced models were limited to a CSMA/CA model, used in 12 out of 41 papers, while 3 papers used a simplified slotted MAC. As a negative trend, 19 of the 41 reviewed papers ignore the necessity of a medium-access model, and assume that nodes within the communication range can always communicate. This assumption only holds true in very sparse networks, where the probability of interfering neighbors is negligible. Moreover, 5 papers did not define the MAC model they used, which clearly compromises the reproducibility of their simulations.

Although the medium-access model may seem less important than the propagation and channel models, from our point of view, the minimum required medium-access model is a slotted mac, where only a connection between 2 nodes in a certain area can be established. It is clear that researchers must improve the average detail of medium-access models used in VDTNs.

2.6.4 Mobility Models and Simulated Scenario

Since the mobility models are of high importance when evaluating VDTN protocols and are closely related to some important contributions of this thesis, we deeply analyzed them in this subsection. This subsection is not limited to mobility models used to evaluate VDTN, but it analyzes a wider group of publications related to VNs and VANETs in general. Only the last part of this subsection specifically focus on the mobility models used by researchers to evaluate previous VDTN proposals.

We have analyzed some of the methods commonly used to obtain suitable mobility patterns in urban vehicular scenarios. Early approaches relied on overly simple mobility models merely based on random mobility. Since these simple models do not represent vehicle mobility properly, other mobility models have recently been developed based on real-world traces, and also on artificial mobility models from the field of transportation and traffic science. Following, we briefly describe the most relevant works.

Random Vehicle Movement

At the beginning of the previous decade, the “Random Way-Point“ mobility model was extensively used in Mobile Ad-hoc NETwork (MANET) research. However, in 2003, the authors in [147] demonstrated how harmful the Random Way-Point mobility model really is in terms of result representativeness. Moreover, the negative effects described in this work become even worse when simulating VANETs. Later on, some other authors extended the “Random Way-Point“ mobility model by restricting the mobility of nodes to a map layout, as in [122]. However, this improvement does not solve the majority of the “Random Way-Point” model problems stated previously.

In our research group we developed a tool called “CityMob“ [86]. CityMob allows users to create random vehicular mobility patterns restricted to a grid. It also adds support for *downtown* definition, where a *downtown* is a region inside the simulated map which concentrates the majority of the selected routes along the simulation. Although CityMob represents a significant improvement compared to non-restricted mobility models and random mobility models, it also presents some problems; the most important one is that vehicular mobility is not influenced by other vehicles, *i.e.* two different vehicles can be at the same physical location, and no minimal distance between vehicles is required. Moreover, vehicles do not change their speed during a trip. However, in the real world, vehicles continuously change their speed according to traffic conditions and road characteristics. Last but not least, vehicles keep moving throughout the whole simulation, which especially influences the performance of protocols that keep data stored in buffers. The research community quickly realized the problems derived from inaccurate simulation patterns, and started to work using alternative methods to obtain suitable mobility traces.

Real Mobility Traces

Compared to the use of random mobility, real traces present a clear improvement. Such traces are usually obtained from a certain set of nodes, e.g. from taxis in Shangai city [75]. Mobility traces can be obtained by tracking the mobility of nodes using On-Board units, as in [75], or by using road-side equipment, as in [45]. Although real traces represent the most realistic mobility patterns, we cannot obviate the fact that mobility of tracked nodes is highly influenced by other untracked vehicles, *e.g.* taxis' mobility is influenced by other users on the road whose movement is not reflected in the collected traces. Moreover, real traces lack the flexibility to allow for an exhaustive evaluation of VANET protocols, *e.g.* changing the vehicle density without modifying their speed is clearly unreal.

Assisted Traffic Simulation

The restrictions of real traces can be overcome, with almost no loss of realism, by using mobility models taken from the field of transportation and traffic science. Several road traffic simulators are widely used among the VANET research community. One of the most widely used mobility generators is SUMO [9]. When simulating traffic mobility for VANETs, not only the vehicles' behavior is important, but also the traffic demand. SUMO allows defining traffic demand in two different ways: trips and flows. The former defines only a vehicle, its origin and its destination, while the latter defines a set of vehicles which execute the same trip. SUMO currently provides several tools to generate traffic demand:

- randomTrips.py: A random trip generator. This tool generates a trip every second having a random origin and destination. It does not check if the origin and destination are connected, or whether the trip is possible.

- duaRouter: A Dijkstra router. Given a file with trips and flows, this tool generates the actual traffic demand, expressed in vehicles with an assigned route. Routes are calculated using the Dijkstra algorithm, and every unconnected trip is discarded.

- duaIterate.py: This Python script will produce a set of optimal routes from a trip file, *i.e.* all the nodes will follow that route which minimizes the total trip-time for all nodes. This tool repeats a routing-simulation loop until optimal routes are found.

Authors have used these tools in order to generate traffic demands for SUMO. The most simplistic one is to define different flows inside the network. Although drivers usually move from certain districts to others, following patterns associated with their working and living places, defining the traffic only by creating fixed flows lacks realism, as we can see in [18] where only a few flows are defined by the user. Another common approach is to generate random trips using *randomTrips.py*. This approach presents the problem that only one vehicle is introduced every second, which leads to long transitory periods until the network reaches a steady state. A more sophisticated traffic demand generation strategy is presented in [81], where a predefined number of vehicles following random routes are randomly placed at the beginning of the simulation. Following this trend, in previous works we used C4R [36], which is a software developed by our group to automate the task of generating random vehicles with random routes at random places. To the best of our knowledge, the work presented in [120] is the only one using the *duaIterate.py* script to generate a "stable and optimal distribution of flows". This type of traffic definition presents a problem: the trip duration cannot be predicted before running the simulations, and, as a consequence, there is no way to ensure, or even determine, if the road traffic simulation will last until the end of the network simulation. As stated in previous work, this lack of realism and generality in mobility patterns can lead to biased results [117].

Bidirectionally coupled network and traffic simulations

In [118] its authors go a step further and present a new simulation framework called Veins, which includes the TraCI interface to allow the network simulator to interact with the traffic simulator running in parallel. Although it presents much novelty and offers a lot of possibilities for VANET simulation, the authors do not address the traffic demand generation problem. The main characteristics and benefits of this tool were highlighted in [119]. VaCAMobil, one of the contributions of this thesis (introduced in chapter 4), makes use of the TraCI interface to control the number of nodes in the simulation.

Mobility models in previous VDTN works

Authors such as Joerer et al. [61] have shown their concerns about mobility model specifications in VN. Fortunately, only two of the reviewed papers used the *Random Way Point* mobility model [147]. The majority of the papers used a *limited random* mobility model, *i.e.* nodes move randomly but their movements are limited by the road network topology. This model is better than pure random mobility, but it does not capture the characteristics of vehicular mobility; for example, two vehicles may occupy the same location at the same time. We also found a group of papers that implemented their own *car following* mobility model [148, 19]. Given the complexity of the models, a self-implemented car following model also compromises the reproducibility of the experiments. Finally, in only 8 papers, we found what we consider the best practice: the use of a validated micro mobility simulator. In the papers we reviewed, researchers used VanetMobiSim [49], SUMO [9] and NETSTREAM [124] as the mobility generator. SUMO is the most advanced mobility simulator, implementing a car following model and real maps and enabling researchers to run mobility and network simulation concurrently, thus allowing events in the network to influence the mobility of the nodes. It is also worth noticing that 11 of the reviewed articles used traces obtained from real vehicles to simulate the mobility of the nodes. Real traces are a good option but they lack flexibility when varying network parameters such as number of nodes, road topology, etc.

Concerning the simulated scenario, it is important to evaluate VDTN protocols in both urban and highway scenarios. We found that only 3 papers considered the highway scenario, while 22 used an urban scenario. Inside the urban scenario there is a huge variety of configurations ranging from urban grids to low-building-density suburban areas. Once again, this diversity complicates the comparison of different proposals. The mobility model and the simulated scenario can significantly affect the performance of protocols, especially VDTN protocols, where nodes tend to carry information in buffers and protocols tend to make decisions based on node mobility.

Table 2.3 summarizes our findings when analyzing the tools and models used by researchers. As previously explained, the diversity of models and simulators makes it impossible to fairly compare different proposals without re-implementing every proposal for a certain simulation environment.

2.6.5 Testbeds and Implementations

Over recent years, some researchers have pointed out the need for real tests prior to VN deployment [40]. Within the set of papers reviewed in this chapter, only [72], [41] and [11] tested their proposals in a real environment. In [72], authors run a test of the Cartorrent system, which is based on the Spawn protocol. In [41], authors extended the Controller Area Network (CAN) bus of vehicles to send its data to a base station using the DTN reference implementation [29]. In [11], authors used a testbed formed by buses inside the University of Massachusetts called UMassDieselNet.

Others authors have presented their testbed for VNs where VDTN protocols could easily be tested. In [31], authors presented Cabernet, a VN deployed over 10 Boston taxis. In [104], an implementation of a warning protocol for VDTNs was presented and tested. In [2], authors presented a *Creative Testbed* that combines simulations with testbed results to maximize flexibility while minimizing deployment cost.

We clearly identify a positive trend towards more advanced testbeds, closer to real deployment. These new testbeds allow to test new proposals and promote the full deployment of VDTNs.

2.7 Summary

The taxonomy and the analysis of previous works presented in this chapter represents one of the contributions of this thesis. Along this chapter we have reviewed the previous proposals in several fields related with VDTNs. Our analysis included more than 100 references and was not limited to describing the previous VDTN protocol proposals. We grouped them in different families according to their similitudes and identified the most suitable application for each family. Moreover, we also analyzed the different evaluation methods used by researchers. This survey helped us defining our objectives, which were stated in Section 1.2.

Part II

Contributions

Chapter 3

The MSDP Protocol

The contents of this chapter have been partially published in:

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A Map-based Sensor Data Delivery Protocol for Vehicular Networks". In: *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. Ayia Napa, Cyprus: IEEE, June 2012, pp. 1–8. ISBN: 978-1-4673-2039-9. DOI: 10.1109/MedHocNet.2012.6257118.

- S. M. Tornell, E. Hernández-Orallo, C. T. Calafate, J.-C. Cano, P. Manzoni, and E. Hernández. "An Analytical Evaluation of a Map-based Sensor-data Delivery Protocol for VANETs". In: *14th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM 2013)*. Madrid, Spain, June 2013. ISBN: 9781467358286. Core A Conference.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Assessing the Effectiveness of DTN Techniques Under Realistic Urban Environments". In: *38th Annual IEEE Conference on Local Computer Networks (LCN 2013)*. Sydney, Australia, Oct. 2013. Core A Conference.

With the purpose of collecting information from sensors deployed in vehicles and delivering that information to a control center located inside the backbone of the network, we have designed the Map-based Sensor-data Delivery Protocol (MSDP) protocol[127]. In MSDP the information is assumed to be obtained from sensors deployed in vehicles which can be retrieved using, for example, an On Board Diagnostic 2 (OBD-II) unit [56]. A message is considered to be delivered if it is correctly received by any of the RSUs, which will then send it to the control center. RSUs are supposed to be placed in strategical places by entities interested in collecting the information. The location of the RSU is provided to vehicles through a dynamic updating service.

In the design of MSDP, we assumed that the nodes of our network have an IEEE 802.11n interface for V2I communication, and an IEEE 802.11p interface for V2V

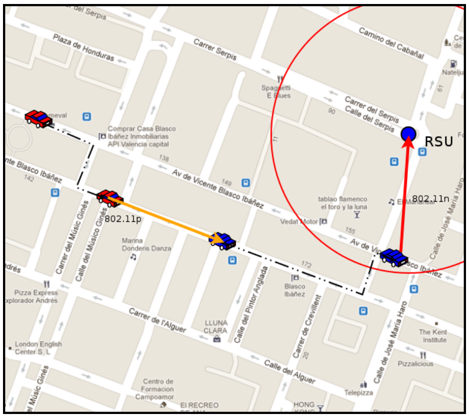


Figure 3.1: Typical situation for MSDP, dashed lines represent the movement of the nodes, while solid lines represent wireless transmissions.

communication. Moreover, we also assumed that all nodes also have some degree of knowledge about their own route obtained from a Navigation System (NS). The NS can be an integrated on board navigation device, or a preloaded static route, like in a train or a bus. Depending on the trustworthiness of that knowledge, MSDP will assign it a different reliability index that varies from 0 to 1. Considering the fast growing number of on board navigation devices sold worldwide as well as the use of such applications in smartphones, we strongly believe that the situation previously described can be considered to be true in future vehicular networks. Figure 3.1 shows a typical situation, where dashed lines represent the movement of the nodes, while solid lines represent wireless transmissions.

Briefly explained, MSDP works as follows: Data messages are generated by combining information from different sensors. Large messages are fragmented into packets that will be stored in the node's buffer. In MSDP a node with packets in its buffer to be transmitted is called *custodian*. *Custodians* announce their presence and information about the knowledge of their routes to other nodes periodically. We refer to nodes that receive this announcement as *candidates*. *Candidates* will answer to the announcements with a message containing information about their positions and an index that depend on their routes. After evaluating this information, the *custodian* will decide, as detailed below, if it is worth forwarding the packets to the best *candidate*, or if it is better to wait for future communication opportunities. It is important to remark that a *custodian* will never remove a packet from its buffer until a *candidate* has been confirmed as the new *custodian*. Finally, when a *custodian* reaches an RSU, it will try to use this communication opportunity to deliver as many packets as possible. Once an RSU has received a packet, the packet will be sent over to the service specific control center inside the backbone. The control center will then reassemble the packets into the original message and process the content.

3.1 MSDP Low Level Mechanisms

MSDP is implemented using our Generic One-Copy DTN Model (GOD) model. The GOD model simplifies implementing and configuring new VDTN protocols and is introduced in the next chapter. The GOD model allows to compare different VDTN protocols by defining a set of low level mechanisms. In detail, MSDP low level mechanisms are configured as follows:

Redundancy

After messages are fragmented into packets, we add a percentage of redundancy packets using FEC techniques. It means that, if a message needs N packets, $N * \alpha$ packets will be sent, where redundancy factor *alpha* is greater than 1 and it depends on the configuration. In our scheme all these packets are generated and distributed, but the original message can be reassembled with just N packets. This redundancy allows reducing the impact of possible packet losses.

Dynamic parameters

In MSDP some parameters are determined dynamically using the available information about the road where the car is currently located and its speed. For example, the interval between announcements is determined by the speed of the node and the speed limit of the current road.

Channel prediction

In MSDP, every message contains information about the position and velocity of the source node. Using this information, nodes are able to omit transmissions that would lead to a waste of resources, as occurs when a message is sent to a node close to the maximum transmission range.

In the next section we detail how to obtain the UtilityIndex, the metric used by MSDP to evaluate candidates nodes and decide the next forwarder of the messages.

3.2 Routing decision

The main novelty of our proposal is the way MSDP makes routing decisions. In MSDP, *custodians* use the value of a function, called UtilityIndex, to determine which is the best *candidate* to forward each packet, or even whether it is better to keep the packet in the buffer and ignore the transmission opportunity. The

UtilityIndex, which depends on four parameters, is calculated locally and communicated to neighbor nodes through MSDP messages. The higher the UtilityIndex is, the better the candidate. It is defined by the following equation:

$$UtilityIndex = \frac{P^2}{T} * Q + 1/D \tag{3.1}$$

The four parameters, P, T, Q and D are defined as follows:

Trustworthy factor (P)

This parameter tries to quantify the reliability of the Navigation System (NS), being higher values associated with more reliable NS data.

Time to reach an RSU (T)

Using the information obtained through the NS, every *candidate* estimates the time, in seconds, needed to reach the nearest RSU. Obviously, better *candidates* are associated with lower times to reach an RSU. T is defined by the following equation:

$$T = \frac{log(t+1)}{log(\tau)} \tag{3.2}$$

being t the time to reach the next RSU expressed in seconds, and τ the maximum delay considered for the application. In this equation, small variations around small t values produces big T variations, while variations around bigger t values have almost no impact on T.

Transmission availability (Q)

Using the information obtained from the NS, nodes are able to estimate the duration of the next transmission opportunity with an RSU, and also the average transmission rate of nodes connected to that RSU. Therefore, nodes can estimate the amount of data that they will be able to deliver. MSDP uses the ratio between the amount of data contained in its buffer and this estimated value (q) to prioritize those nodes with a ratio closer to zero. Q represents this availability, and it is defined by the following equation:

$$Q = max[\frac{log(\beta * (1-q))}{log(\beta)}, 0] \tag{3.3}$$

being q the rate previously mentioned, and β an application parameter that modifies the slope of the logarithmic function. Given an amount of data for the next transmission opportunity, Q decreases as the amount of data in the buffer increases.

Distance to an RSU (D)

A preliminary version of MSDP suffered of inactivity when the carrier node and all its neighbors experienced a value of zero for the first term of equation 3.3. Under this circumstance, no messages were forwarded. Through parameter D we ensure that messages will be forwarded to the closest neighbor, which increases the probability of eventually finding a node whose first term of the equation is bigger than zero.

3.3 Data format

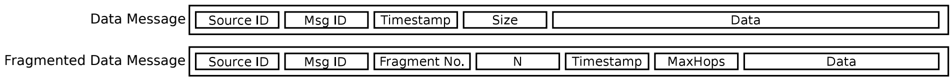


Figure 3.2: MSDP data message format.

In MSDP, nodes generate data messages following the format shown in Figure 3.2. A data message is defined by a tuple (*SourceID*, *MessageID*, *Timestamp*). The data message is then fragmented into several data packets; the number of packets depends on the size of the original data message. Redundancy information packets will also be generated if necessary. Each fragment includes the original tuple to allow reassembling the message at the destination, along with three new fields: the first of them indicates the fragment number, the second one indicates the total amount of data packets necessary to reassemble the original data message, and the last one indicates the maximum number of hops that a data packet can traverse. The size of the payload contained in an information packet is fixed.

3.4 Routing messages

Once data messages are generated and fragmented into packets, packets are individually routed through the network until they are received by an RSU which will send the packets to the control center. With that purpose, MSDP defines 5 types of routing messages that are sent by using UDP at the transport layer. All of them have a common header that includes both the subtype of the message and the message timestamp. The remaining fields are represented in Figure 3.3 and described bellow:

- *MSDP Announcement Messages:* This type of message is broadcasted periodically by *custodians*. It contains four additional fields: the unique Id of the node that generated the message, the position of the node, the velocity of the node, and its own UtilityIndex.

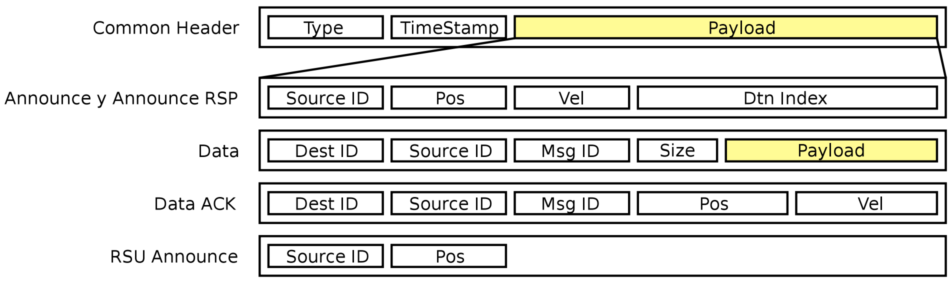


Figure 3.3: MSDP routing message format.

- *MSDP Announcement Response Messages:* This message includes the same fields as the MSDP Announcement Messages does, but it is generated by *candidates* when an MSDP Announcement is received. They are also broadcasted.
- *MSDP Data Messages:* This packet contains 4 fields: the source id, the destination id, a local unique message identifier, and the number of data packets serialized in its payload. The encapsulated data packets can belong to different sources.
- *MSDP Data ACK Messages:* This message is used to confirm that an MSDP data message has been correctly received. It contains the information required to identify the confirmed message, as well as information about the position and the speed of the sender.
- *MSDP RSU Announcement:* RSUs announce their position through this message. It contains the position and the ID of the RSU.

3.5 Nodes Behavior

In MSDP there are 3 type of nodes: *custodians*, *candidates*, and RSUs. Below we present the complete behavior of every type of node and the pseudocode for both custodians and candidates:

Custodians

Custodians nodes are those nodes that have messages in their buffers. The task of the custodian is to find the best neighbor, according to the UtilityIndex, and then to forward as many messages as possible to it. to perform their task the *custodians* behave as follows:

1. Start announcing their position periodically through MSDP Announcement Messages and then wait for replies from *candidates*.

Algorithm 1 Pseudocode of custodians nodes.

```
while $packetsInBuffer()$ do
    $wait(time)$
    $sendAnnouncement()$
    $startTimer(waitTransmissionTime)$
end while
if $receivedRSUAnnouncement(RsuId)$ then
    $startTransmission(RsuId)$
end if
if $receivedAnnouncementResponse(candidateId)$ then
    $store(candidateId, candidateList)$
end if
if $waitTimerExpired()$ then
    if $cadidateList.empty()$ then
        $cadidateId = getBestCandidate(candidateList)$
        $startTransmission(candidateId)$
    end if
end if
if $receivedACK(messageId)$ then
    $remove(messageId)$
end if
```

2. After receiving an MSDP Announcement Response the *candidate* sender ID is stored on a list; then, after a certain delay, a transmission starts with the best of the stored *candidates.*

3. In case an MSDP RSU Announcement is received, a transmission with the RSU will be immediately started.

4. Aiming at reducing the resource consumption, the next MSDP Announcement message scheduled will be omitted if an MSDP Announcement message from another neighbor custodian is received.

5. During a transmission, several data packets are encapsulated inside the payload of MSDP Data messages. The size of MSDP Data messages is limited by network's Maximum Transfer Unit (MTU).

6. When an MSDP Data ACK is received, confirmed data packets are removed from the buffer.

Algorithm 1 shows the pseudocode of custodian nodes.

Candidates

Candidates are passive nodes, their only task is to listen for beacons from *custodians*. When a beacon is received and the *candidate* represents a better opportunity than the source *custodian*, the *candidate* offers itself as a new *custodian*. The *candidates* behave as follows:

1. Remain in a passive state until an MSDP Announcement message is received from a *custodian*.

2. After receiving an MSDP Announcement message containing an UtilityIndex lower than its local UtilityIndex, they will send an MSDP Announcement Response message to announce themselves. To avoid collisions, a small random time is introduced before sending the MSDP Announcement Response.

3. To avoid wasting resources, if an Announcement Response containing an UtilityIndex better than the local UtilityIndex is received, and a new MSDP Announcement Response message was scheduled, then the last MSDP Announcement Response message will be skipped.

4. If an MSDP Data message is received, the data packets contained in its payload are decapsulated and stored in the buffer, and so the node becomes a *custodian* node.

5. If the data packets were stored properly, they will be confirmed with an MSDP Data ACK message.

Algorithm 2 shows the pseudocode of candidates nodes.

RSUs

RSUs are the most basic type of nodes in MSDP, they announce themselves using beacons and receive packets from custodians. Their behavior can be summarized as follows:

1. Announce their presence through RSU Announcement messages.

2. When an MSDP Data message is received, the data packets are immediately decapsulated and sent to the control center.

3. If the data packets are correctly received, then data packets will be confirmed with an MSDP Data ACK message.

Algorithm 2 Pseudocode of candidate nodes.

```
if $announcementRcvd(custID, custUtilIdx)$ then
    if $myUtilIdx > custnUtilIdx$ then
        $waitTime()$
        $sendAnnouncementRsp(myUtilIdx)$
    end if
end if
if $announcementeRspRcv(candUtilIdx)$ then
    if $candUtilIdx > myUtilIdx$ and $rspIsScheduled()$ then
        $cancelResponse()$
    end if
end if
if $dataMsgRcv(msgID, data)$ then
    $result = decapsulateData(data)$
    if $result == OK$ then
        $sendAck(msgID)$
    end if
end if
```

3.6 Summary

In this chapter we presented one of the contributions of this thesis, the Map-based Sensor-data Delivery Protocol (MSDP). MSDP combines information obtained from the Geographic Information Service (GIS) with the actual street/road layout obtained from the Navigation System (NS) to define a new routing metric. MSDP routes information collected at sensors deployed in vehicles to RSU deployed by operators.

The novelty of our proposal stand in the fact that MSDP is not simply based on geographic positions or distances, but routing decisions consider other variables to improve data delivery such as:

- The programmed route of the vehicle.
- The expected time to reach the message's destination.
- The amount of data stored in the vehicle's buffer.
- The amount of data expected to be exchanged with the destination
- The degree of trust of the source for all the considered information.

The performance of MSDP will be evaluated in Chapter 7. There, we will compare MSDP with other proposals, namely: The Geographical Greedy protocol [64], the GeOpps protocol [74], and the Epidemic [139] protocol.

Chapter 4

Improving VNs Simulations

The contents of this chapter have been partially published in:

- M. Báguena, S. M. Tornell, Á. Torres, C. T. Calafate, J.-C. Cano, and P. Manzoni. "VACaMobil: VANET Car Mobility Manager for OMNeT++". In: *IEEE International Conference on Communications 2013: IEEE ICC'13 - 3rd IEEE International Workshop on Smart Communication Protocols and Algorithms (SCPA 2013) (ICC'13 - IEEE ICC'13 - Workshop SCPA)*. Budapest, Hungary, June 2013, pp. 1–5.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Assessing the Effectiveness of DTN Techniques Under Realistic Urban Environments". In: *38th Annual IEEE Conference on Local Computer Networks (LCN 2013)*. Sydney, Australia, Oct. 2013. Core A Conference.

In previous chapters we identified several issues that negatively impacted reproducibility and repeatability of simulation based experiments for VDTN protocols evaluation.

The first problem is related with the complexity of the mobility models used in VDTN: mobility models usually require the definition of tens of configuration variables, however researchers rarely report all the configuration variables in their published works. Therefore, it becomes almost impossible to replicate their results. To solve this problem we have designed and implemented VACaMobil (VANET Car Mobility manager) [1] [6]. VACaMobil allow researches to easily and clearly define and configure the mobility of the vehicles in their simulations.

The second problem is related to the intrinsic complexity of VDTN protocols: Since researchers rarely report all the details of their models, it is impossible to compare VDTN protocols without reimplementing them. To solve this issue, we

[1]VACaMobil was designed and implemented in collaboration with my colleagues Alvaro Torres Cortés and Miguel Báguena Albadalejo.

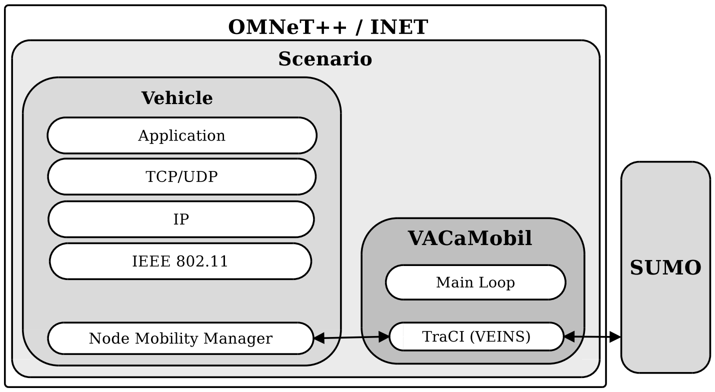


Figure 4.1: The VACaMobil module within the OMNeT++ simulation framework.

have designed the Generic One-Copy DTN Model (GOD) protocol model [128]. The GOD model allows to fairly compare and quickly implement different DTN protocols, offering a simple approach to easily define and apply common low-level transmission mechanisms in one-hop communications.

4.1 VACaMobil: Quasi Steady-State Mobility Simulations for VN

In this section we present our proposed tool to generate realistic mobility for VANETs, VACaMobil. We also provide some important implementation details. All of these characteristics were developed having two basic objectives in mind: achieving realistic-mobility scenarios with a user-defined number of nodes, and simplifying the process of simulating vehicular networks under the desired conditions.

As depicted in Figure 4.1, the VACaMobil module is implemented at the scenario level. By connecting SUMO [9] and the node mobility manager (a general interface), VACaMobil can create new nodes on both *realms* the network simulation and the road traffic simulation. In the next subsection we explain in detail how VACaMobil was implemented.

4.1.1 Implementation details

We have implemented VACaMobil as an add-on to the Veins simulation framework for OMNeT++ [118]. This scheme allow us to take advantage of the current TraCI implementation provided by Veins.

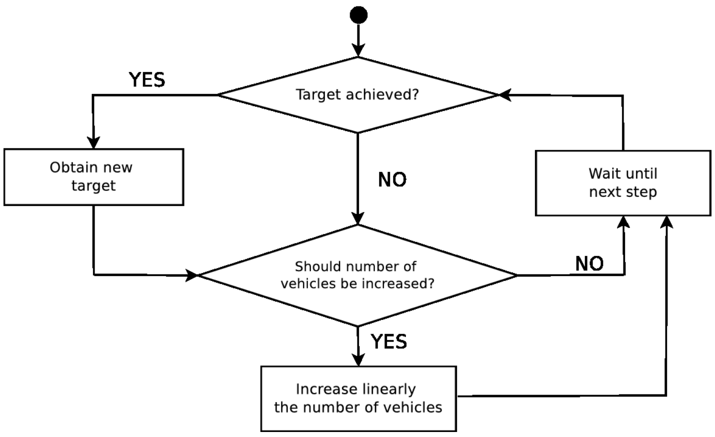

YES
Target achieved?
Obtain new target
NO
Wait until next step
Should number of vehicles be increased?
NO
YES
Increase linearly the number of vehicles

Figure 4.2: VACaMobil algorithm flowchart.

Due to the high modularity of the OMNeT++ simulator, VACaMobil is available for two of the most used network simulation environments for OMNeT++: INET [55] and MIXIM [93][2] Both implementations of VACaMobil can be downloaded from our github account[3].

4.1.2 Average number of vehicles

One of the objectives of VACaMobil is to guarantee a steady number of vehicles throughout the entire simulation. The user can define some degree of variability through the *carVariability* parameter and VACaMobil will ensure that the number of vehicles is always within the defined interval:

$$[average + carVariability, average - carVariability]$$

To control the current number of vehicles, VACaMobil selects a *current target number of vehicles* which should be achieved. Then it adds new vehicles, in case the current number of vehicles is smaller than the *current target number*, or waits until vehicles arrive to their final destination otherwise. A flow chart of this process is shown in Figure 4.2.

To avoid the insertion of a large number of vehicles in a short period of time, VACaMobil stores the duration of the last period where the number of vehicles decreased, and then it takes the same amount of time to insert new vehicles into the network.

[2]By the time I am writing this thesis, VACaMobil is only compatible with INET 2.6. Veins and INET have been reimplemented and VACaMobil needs to be updated.

[3]https://github.com/grclab

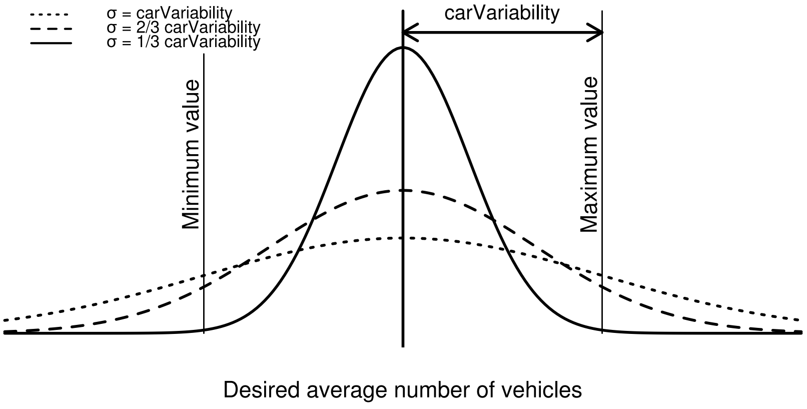


Figure 4.3: Distribution of the target number of vehicles.

The *current target number* of vehicles is obtained from a normal distribution, whose mean ($\bar{x}$) is the desired average value and whose standard deviation (σ) is equal to $1/3 \times carVariability$.

The value of σ is not arbitrary; it has been selected to guarantee that at least 99% of the values obtained from the normal distribution will be inside the user-defined bounds. Figure 4.3 illustrates the effect of σ in the normal distribution shape. If we had set σ equal to *carVariability*, only 68% of the values obtained from the normal distribution would be inside the bounds defined by the user. On the contrary, by setting the standard deviation to $1/3 \times carVariability$, more than 99% of the values returned by the normal distribution are within the defined bounds. Finally, to deal with those values falling outside the user-defined bounds, we filtered the distribution output, being the final *current target number distribution* as follows:

$$N = \begin{cases} y = norm(x,\sigma) & if\, x - carVariability < y \\ & \&\, y < x + carVariability \\ x - carVaribility & if\, y < x - carVaribility \\ x + carVaribility & if\, y > x + carVaribility \end{cases}$$

By selecting this distribution and setting its standard deviation we obtain a great degree of variability, while avoiding extreme values and ensuring that most of the simulation time the number of vehicles is maintained near the average value desired by the user.

4.1.3 Different types of vehicles

SUMO [9] supports the definition of different types of vehicles, which can have different characteristics such as maximum speed, acceleration and deceleration values. The list of different vehicles can be obtained via TraCI[118].

VACaMobil allows the user to set different probabilities associated to each type. In this case, every time a new vehicle is generated, we obtain a uniform random value to select the corresponding vehicle type. If no probability is defined for a certain type of vehicle, we assume it is equal to zero. However, if no probability value is assigned to any of the defined vehicle types, only vehicles of the first type obtained via TraCI will be generated. This feature allows users to easily define heterogeneous networks composed by different vehicles.

Although SUMO itself is able to provide this behavior, VACaMobil adds the possibility of easily changing the vehicles' associated probability between different simulation runs.

4.1.4 Route generation

VACaMobil does not include the ability to dynamically generate random routes. Instead, it includes *randomRoutes.py*, a script that makes use of two well-known tools included in SUMO (*randomTrips.py* and *duaIterate.py*). Thanks to those tools we can generate a large set of random different routes which can be loaded into SUMO.

The *randomRoutes.py* script generates a set of trips between random points of the map by using *randomTrips.py*, and then it computes the *optimal vehicles distribution* using *duaIterate.py*. Finally, it extracts the generated routes and create a new file containing only routes' definitions.

This method also guarantees that all the defined routes are valid, and that all the vehicles that are inserted into the simulation scenario will eventually arrive to their final destination.

4.1.5 Route selection per vehicle

At startup, when SUMO loads all the different routes, VACaMobil will retrieve them through TraCI. Later, VACaMobil will randomly select a route from the pre-loaded set each and every time a new vehicle is introduced in the network.

Since routes are defined as a list of consecutive edges, vehicles are introduced in the network at the beginning of the first edge. It is impossible to insert a new vehicle when another previously created vehicle is already located at the beginning of the selected route. To minimize the impact of this restriction, VACaMobil first tries to insert the vehicle in any of the lanes of the route's first edge. If the previous step does not succeed, VACaMobil selects a new route and tries it again until it finds a free place to insert the vehicle. It may occur that none of the loaded routes allows VACaMobil to introduce a new vehicle. In such a case VACaMobil assumes that its main objective cannot be satisfied and the simulation is aborted. This situation typically occurs at the beginning of the simulation, when VACaMobil must introduce a large number of vehicles in a short period of time. To avoid interrupting the simulation, the user can modify a variable called *warmUpSeconds* which defines the time period at the beginning of the simulation during which VACaMobil requirements are relaxed. During this warm-up time, VACaMobil introduces only a fraction of the desired number of vehicles in every step of the simulation, avoiding the problem previously described. After the warm-up, time VACaMobil ensures that the number of vehicles in the simulation is equal to the value defined by the user.

4.1.6 Repeatability, scalability, and usability

Thanks to the use of the standard random number generators available in OMNeT++, VACaMobil ensures the repeatability of the different scenarios including route and vehicle selection.

Despite the goodness of the characteristics previously presented, the best improvement introduced by VACaMobil is the ability to optimize the researcher's work-flow, as will be detailed in section 8.2. Currently, if a researcher wants to repeat the simulation N times for a certain vehicle density while varying the vehicle routes to decouple the results from the vehicle mobility, the researcher must create N different route files and ensure that the vehicle density is the same along all the simulations. Moreover, the path of those files must be manually introduced into the OMNeT++ configuration file, which is a time consuming task as well as prone to errors.

When using VACaMobil the researcher can take advantage of one of the most important features in OMNeT++, which allows specifying the number of independent repetitions required for every simulation.

As explained later, in section 8.2, VACaMobil also simplifies the process of repeatedly simulating different amount of vehicles in a network.

4.2 The GOD Model: Easy and Reproducible Implementation of One Copy DTN Protocols

A DTN protocol can be subdivided into various components, the most important one is the forwarding criteria. The forwarding criteria, also known as the routing metric, represents the criteria the protocol uses to chose the next forwarding node. Other minor and more generic mechanisms, such as the use of ACK packets, or the adoption of flow control mechanisms, can anyway heavily influence the performance of the DTN protocol as well.

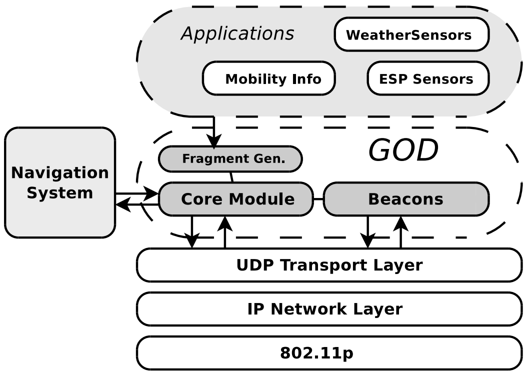


Figure 4.4: Generic One-Copy DTN protocol architecture.

We have designed the GOD model to simplify the evaluation of VDTN routing protocols. In order to simplify the implementation we have omitted issues such as the requirement of a location service which, when required, must be implemented at the application layer. The GOD model assumes that all the messages must be forwarded to any of the RSUs deployed in the scenario, which behave as *sinks*. This case allows us to focus exclusively on the performance of the routing protocol and ignore the dependency on other required services for more complex scenarios.

In order to achieve a fair comparison between different DTN protocols we have developed the Generic One-Copy DTN Model (GOD) model, which can be considered as a super-class of every DTN protocol. Our approach not only simplifies the comparison of different DTN solutions, but it also speeds up the implementation of new protocols.

Figure 4.4 shows how our model integrates within the TCP/IP protocol architecture. It is represented as a new layer between the application and the transport layers, allowing our DTN protocol to work independently of the IP routing protocol.

Our GOD model implements the following configurable generic mechanisms:

- ACK Messages: ACK messages are used to confirm every one-hop transmission. They ensure that no fragment is lost during any one-hop transmission.
- Unconfirmed messages: maximum number of unconfirmed data messages. This is specially useful when the communication channel is unstable.
- Redundancy: redundancy fragments can be added to reduce the impact of fragment losses.
- Location: an interface to the Navigation System (NS) to allow the use of local geographic data, such as location, direction and the programmed route,

among others. The information provided by the NS is limited to the current vehicle.

The GOD structure is based on three different modules which work coupled: the beacon module, the fragment generator module, and the core module. Next we explain these three modules in detail.

4.2.1 The beacon module

The beacon module implements the announcements mechanism. It periodically broadcasts the node information, and it also manages the collected information in order to construct a list of the current neighbors. Every beacon packet contains the source address as well as its location, its velocity, its direction, and the node type obtained from the Navigation System (NS). Moreover, extra information may be included within the beacon packet payload if required by a specific DTN protocol; Figure 4.5 shows the beacons message format.

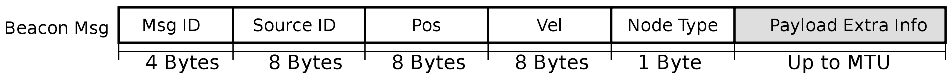


Figure 4.5: Beacon messages format.

A *New Neighbor Event* is notified to the **core module** every time a new neighbor is detected. When a certain number of consecutive beacons from the same neighbor are lost, a *Neighbor Disconnected Event* is notified to the **core module**. This module also notifies the **core module** when the information about a neighbor has been updated based on a newly received beacon.

Inside this module, the inter-beacon time can be defined to meet the protocol requirements.

4.2.2 The fragments generator module

This module is directly connected to the application layer and it is in charge, if required, of dividing large messages into fragments smaller than the MTU. Since the

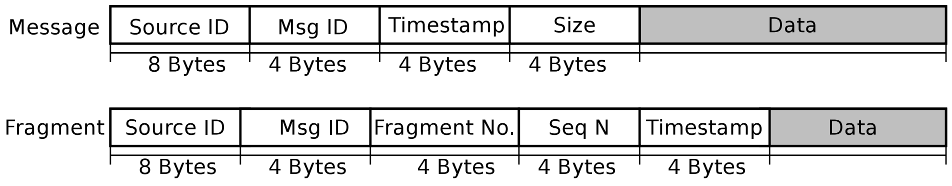
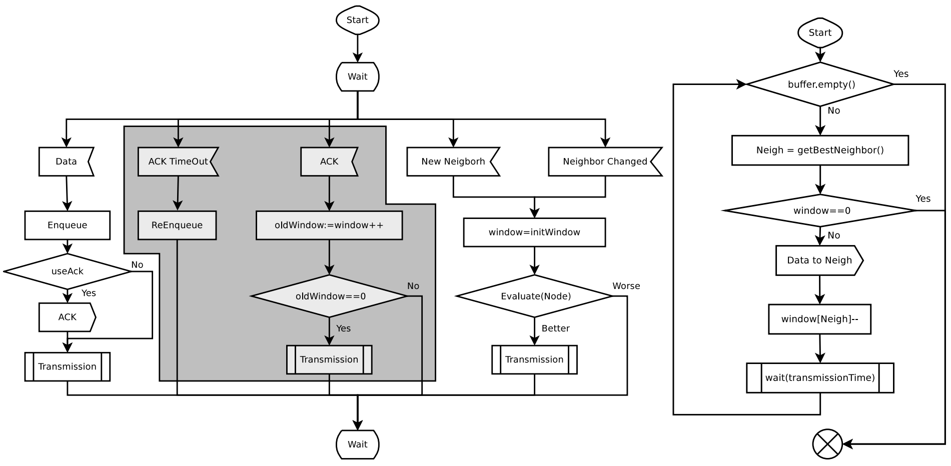


Figure 4.6: Information message format and fragment format.



(a) Dtn module flowchart.

(b) Transmission process flowchart.

Figure 4.7: Generic One-copy DTN Model flowchart.

GOD uses UDP packets for one-hop communication, large messages must be split-up to avoid IP fragmentation, which would interfere with routing decisions. Figure 4.6 shows the relationship between the information messages and their fragments. The fragments generator module can also create **redundancy** to increase reliability. When redundancy is enabled, a percentage of extra fragments are created using FEC techniques. That is, if a message is divided into N fragments, $N * \alpha$ total fragments will be generated, where redundancy factor α is greater than 1 and depends on the configuration. The basic hypothesis is that the original message can be reassembled with whatever subset of size N of the sent fragments. This redundancy allows reducing the impact of possible fragment losses.

4.2.3 The core module

This module is connected to the beacon module, and it is in charge of managing transmission opportunities between nodes. The core module has a storage buffer where fragments are enqueued and dequeued based on their timestamp: older fragments receive higher priority. The events notified by the beacon module are used to keep a sorted list of neighbors according to the chosen routing metric. A schematic overview of this module behavior is described in Figure 4.7a, where the shadowed parts of the chart are only executed when the ACK mechanism is active. This module is activated by five different types of event:

- *Data Messages*: A Data message can arrive from the fragments generator module or from the network, i.e., from a neighbor. When a Data message arrives, the fragments contained in it are enqueued in the local buffer and, in case it is enabled, its reception is confirmed through an ACK message. This ACK allows the previous source node to remove the confirmed fragments from its buffer.
- *ACK messages*: An ACK message confirms the reception of a Data message, and the fragments contained inside the confirmed Data message can now be definitely deleted from the buffer.
- *ACK Time Out*: When an ACK Time Out occurs, the data fragments contained in the unconfirmed data message are re-enqueued in the sending buffer.
- *New Neighbor and Neighbor Changed*: The neighbor is evaluated using the metric defined by the specific DTN protocol, and, in case the neighbor is evaluated as a "better" node than the current carrier, a new transmission process is started. Obviously, if the neighbor is detected as an RSU, it is always evaluated as the best possible neighbor and a new transmission process is immediately started. Whenever one of these events is notified, the value of the transmission window is initialized to its original value.

The transmission process tries to transmit as many fragments as possible to the best neighbor. Its behavior, represented in Figure 4.7b, works as follows:

1. The module checks if there are fragments in the buffer to be transmitted.
2. It obtains the best next node from the node list according to the protocol-specific metric.
3. It checks if the best node's location estimation is inside the defined transmission range.
4. It checks if the transmission window of the best node is equal to 0; in this case, the process is over.
5. It sends a data packet containing as many fragments as possible. This action dequeues the fragments from the buffer.
6. If the ACK mechanism is enabled, it schedules an ACK Time Out for the previously sent data message. If it is not running, it removes the sent fragments.
7. It waits for the time required to send a data packet and its corresponding ACK packet.

8. It starts a new transmission process.

9. The loop is broken when there are no more message in the buffer.

It is worth noticing that, since all the fragments have the same destination (any of the RSUs), the next forwarding node selection is based only on the state of the current carrier and its neighbors. Moreover, we consider that all the RSUs behave as *sinks*. They simply send beacons to announce their presence to vehicles. When vehicles send Data messages to the RSUs, they may confirm their reception through an ACK message if required, and the fragments encapsulated in it are then forwarded to the control center through the backbone network.

With our GOD model, when the ACK mechanism is active, it is ensured that no fragment is removed from the buffer until a neighbor have been confirmed as the new carrier. ACK messages are also used to limit the number of data messages pending confirmation. The use of ACK, as well as the communication type, *i.e.* broadcast or unicast, can be configured in order to model different DTN protocol variants.

4.3 Summary

In this chapter we have presented in detail our contributions to repeatability and reproducibility of VDTN protocols simulation based experiments. We have presented VACaMobil, a mobility manager for the INET simulation framework [55] which allows easily defining the number of nodes present and active in a simulation, and the GOD model, a set of modules designed for INET that allows to fairly compare and quickly implement different DTN protocols, making it easy to define and apply common low-level transmission mechanisms in one-hop communications.

The novelty of VACaMobil has been also appreciated by other researchers that have used it for their own experiments [91, 39, 38].

A comparison between VACaMobil and other mobility generation methods is assessed in Chapter 8. In the previous chapter, the GOD model was used to implement our proposal, the Map-based Sensor-data Delivery Protocol (MSDP). The GOD model was also used in Chapter 7 to implement the Geographical Greedy [64], GeOpps [74], and the Epidemic [139] protocols in order to evaluate our proposal, the MSDP.

Chapter 5

Developing an ITS Application for Smartphones

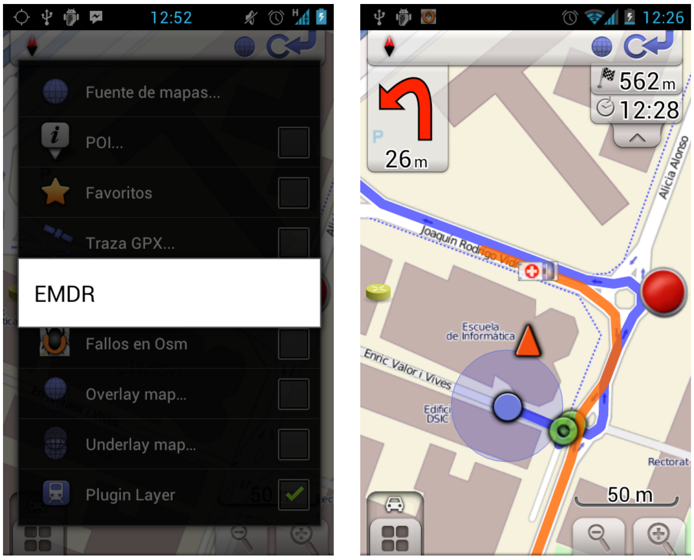
The contents of this chapter have been partially published in:

- S. M. Tornell, C. T. Calafate, J.-C. Cano, P. Manzoni, M. Fogue, and F. J. Martinez. "Evaluating the Feasibility of Using Smartphones for ITS Safety Applications". In: *VTC Spring 2013 IEEE Vehicular Technology Conference*. 2013. ISBN: 9781467363372. Core B Conference.

In previous chapters, the contributions of this thesis have been focused on VNs simulations. As we have seen, the technology is ready for deployment and intensive research has been done. From now on, we will focus in how to bring VNs from research simulations to real world.

As we introduced in Section 2.1, little or none VN systems have been widely deployed. One of the main reasons is that manufacturers have implemented IEEE 802.11p and other VN technologies in the form of OBUs integrated in the dashboards. OBUs are rarely updated and quickly become obsolete. Besides, until now, the services OBUs have offered have been brand specific, leading to compatibility problems between brands. And finally, brands tend to include new technology only in luxury cars. These facts are delaying the real deployment and adoption of VNs.

One of the main objectives of this thesis is "to develop a new platform that integrates user devices into vehicular networks". In our efforts to integrate common user devices in VNs, we explored the implementation of a warning dissemination application for smartphones. We found that, in smartphones, not only ad-hoc communication is not available out of the box, but its activation may void the



(a) Layer selection screen. (b) Application screenshot

Figure 5.1: Ambulance Warning Application screenshots.

guarantee of the devices. We also found that only one interface can be activated at a time. Taken into account these limitations, in this chapter we illustrate the implementation of this application and the experiments we performed to measure its efficiency.

5.1 A Warning Dissemination Application for Smartphones

We built a warning dissemination application upon standard smartphones based on the Android operating system. The objective was to avoid the requirement of a special approval for installation in vehicles, and to take advantage of the ubiquity and relatively low price of smartphones. Moreover, the number of smartphones already sold reduces the cost of deployment, since it does not requires installing new hardware on every vehicle.

In this case, we envisioned an environment where VN applications will directly use the Android networking API for network communications, and the file input/output API for the logging system. Our application was based on the OsmAnd [102] navigation software to which we added new communication features and a new map layer. This new map layer displays information about local warning events which are disseminated by nodes using ad-hoc communication. To disseminate the information we implemented the enhanced Message Dissemination based on Roadmaps (eMDR) [87] dissemination protocol.

Our ITS application broadcasts the location and the route of an approaching ambulance using the eMDR protocol warning mode, informing nearby drivers, which will be able to act accordingly to favor the progress of the ambulance toward its destination.

Figure 5.1b shows our “Warning Ambulance Application” in execution. We can see two idle neighbors represented by green circles, and a neighbor in alarm mode, represented by an ambulance icon; the orange line is the ambulance’s route, and the blue line represents the vehicle programmed route. Both the red button on the right and the route-shaped button on the left, are used for testing purposes. The red button activates the alarm mode and, if present, broadcasts the programmed route; it is supposed to be available only to authorized devices, such as ambulances, police-cars, etc. The other button is used to select between three forwarding modes: (i) normal forwarding, *i.e.* following eMDR rules, (ii) unconditional forwarding, *i.e.* every alarm message is rebroadcasted, and (iii) forwarding disabled, *i.e.* no alarm message is rebroadcasted. Figure 5.1a shows the interface through which users will select the new layer, integrated in the OSMAnd application, that will contain geographic information obtained from eMDR.

5.2 Implementation Details

In this section we briefly describe the implementation details of our application architecture as well as the eMDR protocol implementation.

5.2.1 Architecture

To take advantage of available software, we decided to integrate our application in an existing open source navigation software. Our main requisites were: (i) to have a free map data source to avoid royalties issues, (ii) to have an off-line route calculation system, and (iii) to present an easily expandable structure. With these premises in mind, after scouting the Android market, we chose OsmAnd [102], a navigation software that uses maps and route layout information from OpenStreetMaps [101]. The map rendering process in OsmAnd is composed by different layers that are rendered sequentially. Therefore the different applications can be programmed as a special layer that not only draws new data on the map, but it also communicates with the dedicated protocol threads which use the socket API to communicate with other vehicles through the ad-hoc interface. We have developed a class, called ‘GeoHelper”, to simplify the use of geographical data provided by the integrated GPS, and to deal with map issues related with the navigation service. Figure 5.2 shows how our application is integrated into the OsmAnd architecture. Summarizing, the OsmAnd map layer calls the *draw()*

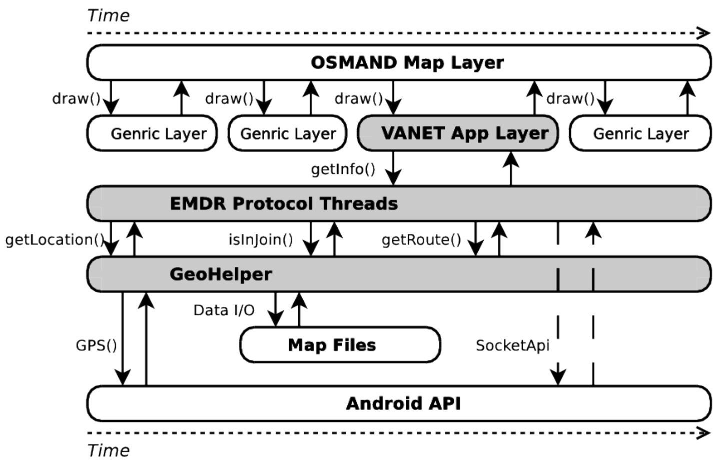


Figure 5.2: Architecture overview.

method of our new VN App Layer, which obtains the required info from the network protocol threads. Simultaneously, the protocol threads communicates with our GeoHelper class, which runs under its own thread, to obtain geographical info.

5.2.2 User Interface

Our application is implemented as a new layer that adds information obtained from the VN to the map view in OsmAnd. Besides implementing our application, we have also created a new interface called *GeoPluginLayer* that offers a common structure to implement new geographic information related layers using class heritage. This design allows us to quickly implement different protocols reusing common code.

5.2.3 Wireless Radio

Smartphones are usually equipped with an 802.11b/g interface that, in our case, has been configured in ad-hoc mode to allow direct communication between nodes. The typical transmission power of these devices is 16 dBm (40mW), which is far from the maximum allowed transmission power for IEEE 802.11g in Europe (20 dBm, i.e. 100 mW). These values are really low compared with the maximum transmission power for IEEE 802.11p (33 dBm, i.e. 2 W), which is the standard PHY/MAC protocol for VANETs. It is worth noticing that the use of the ad-hoc mode avoids any setting up delay due to the association and authentication processes. To configure the network interface in ad-hoc mode we had to perform a process called *to root the phones*, this process is not available in every android smartphone and usually void the guarantee.

5.2.4 GeoHelper Class

To simplify the management of geographical data we have developed a class called "GeoHelper". This class collects and processes data from files and from the services provided by both OsmAnd and the Android operating system (*i.e.* GPS, routing service, etc), offering different methods to our application.

The most important method provided by our GeoHelper class is the *getCurrentLocation()* method, which returns, an estimated current location based on the last two updates provided by the GPS interface and the direction of the current road. The difference between the last two updates is used to estimate the speed vector of the vehicle, if a *getLocation()* call occurs between two consecutive updates; the current position is estimated using the estimated speed vector and the last known location. In the case of *getLocationOnStreet()* calls, the estimated location is restricted to be on the closest road, and the speed vector also lays on the current road.

Concerning the *findRoute(location)*, we found that the time required by OsmAnd to calculate a route between two locations was in the order of tens of seconds, and that this value is strongly dependent on the number of possible routes. In our opinion, these issues clearly difficult the usage of some routing protocols that calculate the shortest route for every sent packet, as in [142].

5.2.5 EMDR Protocol in Detail

In order to disseminate warnings as quickly as possible, eMDR works as follows: When $vehicle_i$ starts the broadcast of a message, it sends m to all of its neighbors. When any nearby vehicle receives m for the first time, it rebroadcasts it by further relaying m to its neighbors. Depending on their characteristics, every vehicle repeats the $send(warning)$ or the $send(beacon)$ operations periodically with different periods (T_w and T_b, respectively). It is worth noting that beacons are used to simply inform neighbors of a node about node characteristics, while warnings contain emergency information and must be disseminated to as many nodes as possible. When a new message m is received, the vehicle tests whether m has already been received. To evaluate this condition, each vehicle maintains a list of received messages. A new warning message is rebroadcasted to the surrounding vehicles only when the distance d between sender and receiver is higher than a distance threshold D, or the receiver is in a different street than the sender. eMDR considers that two vehicles are in a different street when: (i) both are indeed in different roads (this information is obtained by on-board GPS systems with integrated street maps), (ii) the receiver, in spite of being in the same street, is near to an intersection, or (iii) the receiver detects that it has neighbors in different streets. Hence, warnings can be rebroadcasted to vehicles which are traveling on other streets, overcoming the radio signal interference due to the presence of buildings.

Following a divide and conquer paradigm we have structured our implementation of the eMDR protocol in three different classes: an upper class, called eMDRPluginLayer, that handles *onDraw()* calls from the map and all user interface related events (buttons). And two lower classes that implement the eMDR protocol, namely: *eMDRPacketGenerator* and *eMDRProtocol*. The former is in charge of packet generation, while the latter is in charge of receiving, processing, and forwarding the messages received from other nodes following eMDR rules. The eMDR protocol will send a beacon every second, or warning messages instead when in alarm mode. In addition, if a route is programmed, it will be included in the warning messages payload as an array of points.

5.3 Evaluating Smartphones for Vehicular Applications

The application described in the previous section has been tested in a real scenario. We focused on the effects of the wireless channel and the behavior of the GPS interface.

5.3.1 Devices' detailed description

For our experiments, we selected three different devices from the same manufacturer, *i.e.* HTC. In an attempt to prove that very advanced smartphones are in fact not required for these applications, we chose three devices whose performance varies a lot and that are definitely not the most recent smartphones currently in the market:

1. HTC Desire: Released in 2010, 1Ghz CPU core, 512 MB Ram, WIFI driver: bcm4329.

2. HTC Hero: Released in 2009, 588 Mhz CPU core, 288 MB Ram, WIFI driver: tiwlan1251.

3. HTC Tatoo: Released in 2009, 528 Mhz CPU core, 256 MB Ram, WIFI driver: tiwlan1251.

5.3.2 Experiments

We have designed a set of experiments to evaluate the following performance parameters: (i) message reception probability when in Line of Sight (LOS), (ii) message reception probability when nodes are in different streets, and (iii) GPS

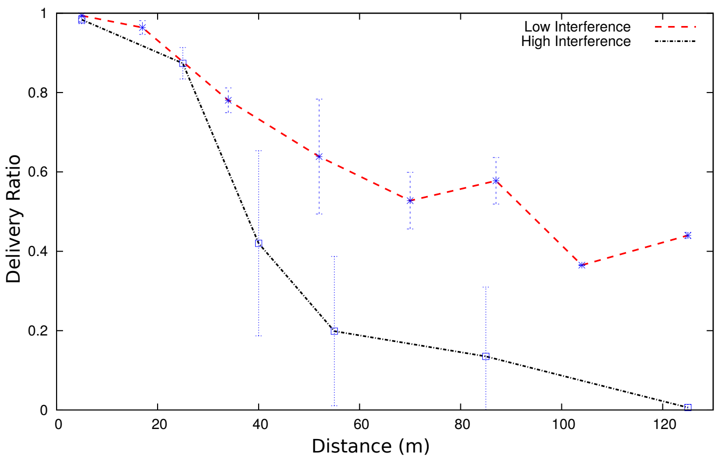


Figure 5.3: Rx Probability vs Distance.

updates inter-arrival time. All the experiments were performed in a real environment: vehicles were parked in streets with a typical traffic flow. Since all mobiles were inside vehicles, the transmissions were also affected by different issues related to adverse signal effects caused by the structure of the vehicle.

Message reception probability when in LOS

In this experiment we placed two of the handsets in different cars; then, using our application, we sent a burst of 200 warning messages and counted the number of messages successfully received. We executed two different experiments to evaluate this metric. The first one was executed in a high interference environment, and the second one was in a low interference environment. To achieve statistical significant results each experiment was repeated four times, each measurement represents the average value and the 95% confidence interval. Results, represented in figure 5.3, shows that, as expected, the reception probability decreases when the distance increases. Comparing both graphs, we can also appreciate that the presence or the absence of interferences can highly influence the performance of VANET's application in smartphones, reducing the communication rate from 80 m to merely 40 m, and increasing the variability of the results.

Message reception probability when in N-LOS

In this experiment cars where located in perpendicular streets. One of them was located 25 meters away from the intersection, and the second vehicle was moving away from the intersection. Figure 5.4 represents the location of the cars. As expected, the moving car stopped receiving messages as soon as it moved a few

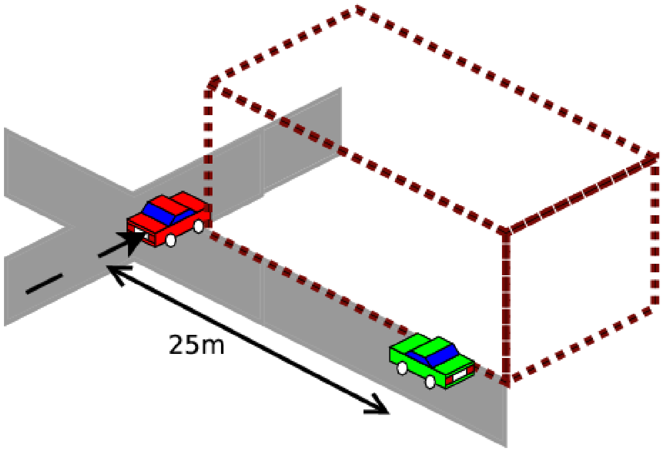


Figure 5.4: Location of the cars for NLOS measuring.

meters away from the intersection. With these results in mind, we decided that the threshold distance under eMDR to consider that a vehicle is “near to an intersection” would be configured to 10 meters or less. Also, experiments have shown that the best parameter to detect if a vehicle is close to an intersection is the detection of neighbors from different streets by using the information contained in neighbors’ beacons.

GPS updates inter-arrival time

Another important issue when checking the feasibility of our solution is the freshness of the GPS data in smartphones. During the previous experiments we collected around 12000 GPS measurements. By analyzing their inter-arrival time, we found that the average inter-arrival time for GPS updates was 1.07 s, while the maximum value was of 15 s, and only in 1 % of the total measurements it differed from 1.0 s. Although we have configured the GPS interface to notify our application about location changes as soon as possible, the minimum time between updates that the system was able to provide was of 1.0 s. If we consider a vehicle with a speed of 25 m/s, a maximum acceleration of 0.8 m/s^2, and a maximum deceleration of 4.5 m/s^2, it can typically travel between 21.40 m and 25.40 m per second, our position estimation system, which assumes a constant direction and speed during the inter-update time, will introduce a maximum error of 3.6 m due to mobility. We believe that this value is small enough to be used in VANETs.

5.4 Conclusions and Lessons Learned

In this chapter we presented an implementation of an ITS warning service based on VANETs for Android smartphones. The selected application advertises emergency vehicle's (ambulances, police cars, etc) warnings to nearby vehicles, besides, it can be easily extended to support any type of warning alert such as a slippery road, or an accident. This application offers a smart interface to geographic information through the integration in a Navigation Software, that allows to augment the location information with information about the road map layout.

Our measurements demonstrated that smartphones provide a reasonable connectivity and enough geographic precision for information dissemination. On the other hand, we discovered that enabling and configuring ad-hoc connectivity in smartphones is a tedious device-dependent process. Moreover, not only is the out-of-the-box smartphone's connectivity restricted to infrastructure networks, such as WiFi or 3G/4G networks, but also the number of simultaneous active network interfaces is limited to one. These restrictions frustrate the adoption of smartphones for applications based on opportunistic connectivity in vehicular scenarios and leaded us to the design and implementation of our GRCBox architecture, that will be presented in the next chapter.

Chapter 6

The GRCBox: Simplifying the in-vehicle connectivity for ITS applications

The contents of this chapter have been partially published in:

- S. M. Tornell, S. Patra, C. T. Calafate, J.-C. Cano, and P. Manzoni. "GRCBox: Extending Smartphone Connectivity in Vehicular Networks". In: *International Journal of Distributed Sensor Networks* 2015.Article ID 478064 (2015), p. 13. DOI: 10.1155/2015/478064. Impact Factor: 0.665(Q4).

- S. M. Tornell, T. Kärkkäinen, J. Ott, C. Calafate, J.-C. Cano, and P. Manzoni. "Simplifying The In-vehicle Connectivity for ITS Applications". In: *MOBIQUITOUS 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2015. DOI: 10.4108/eai.22-7-2015.2260058. Core A Conference.

In this chapter we present one of the contributions of this thesis: the **GRCBox Architecture** [135]. The GRCBox architecture enables applications running in user devices such as smartphones or tablets to extend its infrastructure connectivity by adding infrastructure-less in-vehicle to in-vehicle communication.

6.1 Introduction

While VNs technology is close to be ready for deployment, it is expected that car manufacturers will introduce it gradually, starting at high-level models. This issue, coupled with the low renovation rate of the vehicle fleet, will slow down and delay the deployment of VNs.

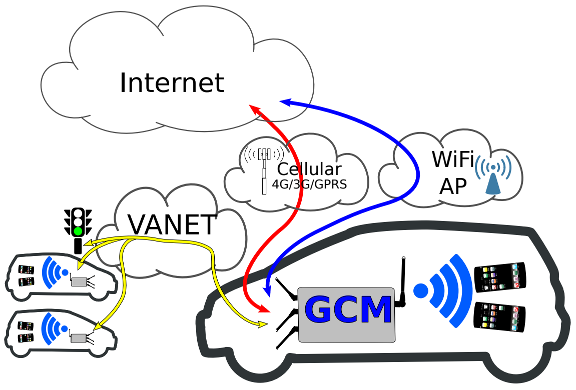


Figure 6.1: An example GCM connected to several networks.

To face this problem, part of the industry has proposed vendor-specific alternatives devices that integrate user devices like smartphones in infrastructure based VNs. The Car Connectivity Consortium (CCC), which integrates companies from the automotive and the telecommunications sector, released Mirrorlink [15], a standard technology that moves the computing tasks from the OBU to the smartphone, and present the information on the OBU's display. Users can also interact with the smartphone through the dashboard elements. Google and Apple, two of the biggest technology companies, have also proposed their own solutions, Android Auto [42], and CarPlay [3], respectively. However, all these proposals rely on the Internet infrastructure to provide in-vehicle connectivity, ignoring the advantages of V2V communication and opportunistic contacts. Moreover, these proposals are heavily dependent on companies and centralized service providers.

To extend in-vehicular connectivity to external networks such as VANETs, we have designed the GRCBox Architecture. The GRCBox Architecture is based on the GRCBox Connectivity Manager (GCM), which is responsible for creating an intra-vehicle WiFi network. User devices inside the vehicle can connect to this network to share contents and to reach any of the external networks, as depicted in Figure 6.1. GRCBox allows implementing Internet-independent solutions that focus on applications that exploit local connectivity to provide new services, such as platoon-oriented applications where friends or workers share information while traveling together in different vehicles. Opportunistic applications are especially suitable for remote areas where infrastructure is expensive to deploy. Moreover, the short life and local propagation of the information favors privacy. The GRCBox Architecture provides a Representational State Transfer (REST) interface [35] and it is based on basic IP networking, thereby minimizing the modifications required to create GRCBox-aware applications. GRCBox also removes the dependency on car manufacturers when implementing V2V communications; by using GRCBox, users can now implement their own VANETs while accessing the already deployed infrastructure networks.

To the best of our knowledge, the GRCBox is the first effort aimed at increasing the user device in-vehicle connectivity in order to allow users to create their own autonomous VN and test innovative VN applications.

The rest of this chapter is organized as follows: In section 6.2 we present the GRCBox architecture. In section 6.2, we detail the interaction between the GRCBox client and the GRCBox Connectivity Manager (GCM), and introduce the client library. In section 6.3 we describe in detail the implementation of the GRCBox core module. Finally, in section 6.4 we summarize the contents of this chapter.

6.2 The GRCBox Architecture

The GRCBox architecture defines both the GRCBox Connectivity Manager (GCM) placed in the vehicle and a client-server REST API that allows applications to interact with the GCM to reach external networks. Figure 6.2 represents the architecture including both, the GCM and the client API. To implement the REST API we used the RESTlet framework [83], which simplifies the implementation. An example of a GCM placed in a vehicle and connected to three different external networks, is shown in Figure 6.1. In this example an application running in the user device may choose to connect to the VANET for local communication, or connect to either the cellular network or the WiFi network to reach the Internet. In this section, we first offer a general overview of the interaction between the GCM and the User Application, then we detail the client API.

6.2.1 User Device-GCM Interaction

The GCM creates a WiFi access point to which smartphones, tablets, and other user devices in the vehicle will associate. Once the user devices connect to the GRCBox's wireless network, they can share contents between them, as well as access the external networks. By default, every new connection is forwarded from the GCM through the default Internet connection. In case an application requires the use of any other available interface, it must notify it to the GCM. In this section we enumerate the steps a GRCBox application must follow to communicate through any network interface that differs from the default one. A rule enables an application to choose the outgoing interface for a certain connection, or to register as listeners for a defined incoming connection. A rule is a packet filter defined by the following elements:

- Rule Type: The GRCBox Architecture defines three different kind of rules, *Incoming*, *Outgoing*, and *Multicast*. Multicast rules define bi-directional mul-

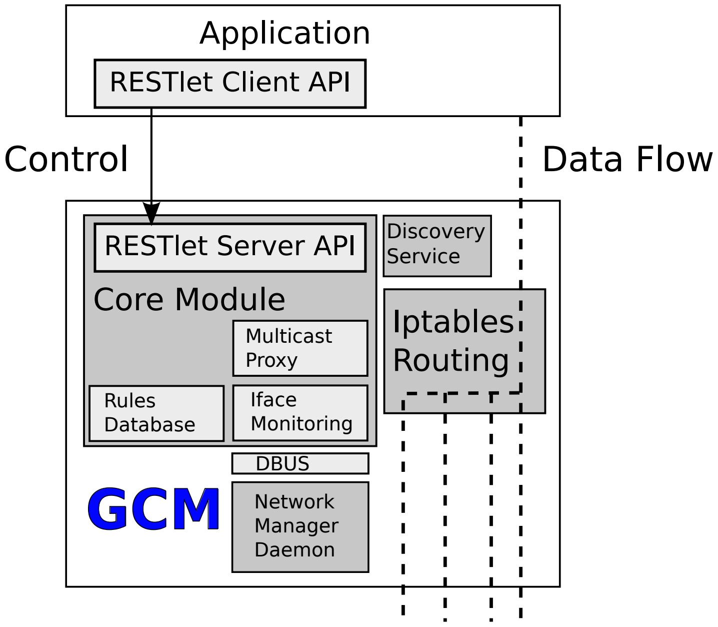


Figure 6.2: GRCBox Architecture with GCM modules in detail.

ticast packet flows between the internal interface and one of the external interfaces.

- Interface Name: The name of the outer interface to which the rule applies.
- Protocol: The protocol of the connection. Currently, GRCBox supports UDP and TCP, though we expect to implement more protocols, such as SCMP and ICMP, in the future.
- Source Port: The source port of the connection.
- Source Address: The source address of the connection.
- Destination Port: The destination port of the connection.
- Destination Address: The destination IP address of the connection.

The steps that GRCBox applications should follow are the following:

1. **Check GRCBox availability:** Once the device is associated to the GRCBox wireless network, the application must check if a GCM is available. To do so, the application will try to connect to the “http://grcbox/” url to check the status of the GCM.

2. **Application Registration:** After checking the availability of the GCM, an application must register itself to get a key. This key will be used for later application-server interactions to ensure no other application but the owner of a rule can renew, remove, or modify it.

3. **Check the Status of the Interfaces:** The next step is to check the status of the different network interfaces to identify if the desired interface is available. At this point the application can also check other previously registered rules to avoid conflicts.

4. **Register the desired rule[s]:** Now the application can register as many rules as required to configure the GCM to forward specific incoming and outgoing connections, or to forward multicast packets to external interfaces.

5. **Transmit Data:** At this point the application can effectively use the registered connections which will be forwarded according to the defined rules.

6. **Close the Connection:** When a rule is no longer required, it must be removed from the GCM. This step is optional since rules are always removed from the GCM database if the application is disconnected.

7. **Application Disconnection:** Once the application ends its interaction with the GCM, it should notify it to allow removing its registered rules.

The interactions between GRCBox applications and the GCM rely on the RESTlet API exposed by the GCM. The details of this API are described in annex B. Chapter 9 includes examples of multiple case studies to clarify the communication between the user device and the GCM. Chapter 9 also includes a detailed performance evaluation for the presented scenarios.

The GRCBox also supports the integration of third party applications by providing a management application that enables the interactive definition of new rules for non-GRCBox applications. Thereby, the user can define rules for well-known application protocols such as HTTP, POP3, etc.

6.2.2 Client Implementation Details

The GRCBox public REST API is OS independent. Until now, we have implemented two different client libraries: A C# library and an Android library. Both libraries avoid developers to deal directly with the details of http requests and json serialization. Both libraries offer the following methods:

register
Register a new application in the GRCBoxServer database.

deregister
Remove the calling application from the GRCBox database

getInterfaces
Return a list of the available external interfaces.

getApps
Return a list of the applications registered in the GRCBoxServer.

getRules
Return a list containing the rules owned by the calling application.

registerNewRule(rule)
Register a new rule in the rules database.

removeRule
Remove a rule previously registered by the calling application.

getMulticastPlugins
Get the list of implemented third party multicast plugins (see section 9.3).

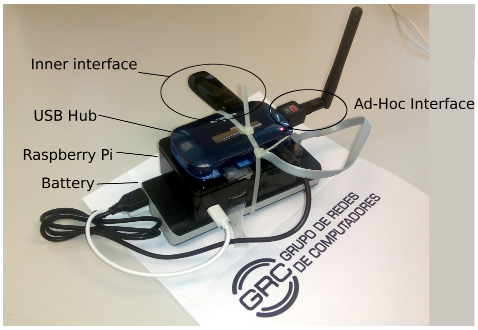
6.3 The GRCBox Connectivity Manager (GCM)

The GRCBox Connectivity Manager (GCM), which is placed inside vehicles, must have at least one WiFi interface to which user devices are connected to (called *inner interface*), and one or more *external interfaces* used to provide connectivity to external networks. The GCM is composed of several modules that work together. A scheme of the different components, their connections, and the paths traversed by data flows is presented in Figure 6.2. The GCM software is based on a Linux operating system, and it takes advantage of several well-known Linux services to provide the desired functionality. The different components running in the GCM are the following:

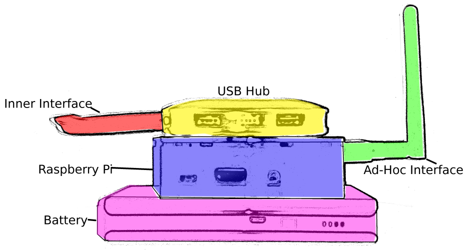
Discovery Service: The Linux daemon *dnsmasq* is used to answer DHCP and DNS requests. It is configured to resolve the "grcbox" domain name to the GCM inner interface. This way clients on the inner network can connect to the GCM without information about its IP address by directly attempting a connection to "http://grcbox/".

Packet Forwarding: To define fine grained, per connection routing, GCM uses Iptables for connection filtering and labeling, and the Linux kernel support for "Policy Routing".

Ifaces Monitoring: To monitor the status of the network interfaces, GCM connects to the NetworkManager daemon using the D-Bus interface to perform event subscribing tasks (D-Bus is a linux message bus system) [25].



(a) The GRCBox connected to a battery and an ad-hoc interface.



(b) Schematic view of GRCBox.

Figure 6.3: Minimum GRCBox setup to provide ad-hoc connectivity.

Core Module: The most important part of the GCM is its core module. The core module performs several activities: it listens to clients' requests through the REST API, maintains a database of all registered rules, starts and stops multicast proxies when needed, and performs actions when events on the interfaces are notified. The details of the core module are described in the next section.

To implement our GRCBox hardware module we have chosen an embedded computer called RaspberryPi[1] [92]. The RaspberryPi is a credit-card size computer whose cost is only 35$, but that has enough power to perform low-scale network routing. In this computer we have installed a Raspbian [125] distribution, which is a general-purpose Linux distribution based on Debian and optimized for the RaspberryPi. Raspbian supports most current networking hardware, avoiding common problems of other embedded operating systems. Figure 6.3 shows a GRCBox implemented in a Raspberry Pi with only an external ad-hoc interface. We added a battery to simplify the logistics of our experiments.

6.3.1 Core Module Implementation Details

In this section we detail the implementation of the "Core Module"[2]. The GRCBox "Core Module" is implemented as a Java application called "GRCBoxServer". The tasks performed by this application have been divided in different java classes which are represented in Figure 6.4. In detail, each class performs the following tasks:

[1]http://www.raspberrypi.org/

[2]The "Core Module" source code is available from our github account `https://github.com/GRCDEV/GrcBox`

GrcBoxApplication

This class is the starting point of the GRCBoxServer application. It initializes the RulesDB class and register all the resource implementations on the RESTlet server. This class also starts the http server.

Resources

Each REST resource corresponds to a java class. Each resource is accessed by a different URL. For example the url `http://grcbox/ifaces` correspond to the java class `es.upv.grc.grcbox.common.resources.IfacesResource`. In the GRCBoxServer, resources simply parse the client request and translate it to a call of the RulesDB class. The resources defined by the GRCBoxServer are listed in the Annex B.

RulesDB

This is the most important class of the GrcBoxServer. It keeps a database of the applications and rules registered in the server. It also instantiates a NetworkInterfaceManager to monitor the status of the network interfaces. When a new rule is added or a rule has to be removed, it calls the class IpTablesManager.

IpTablesManager

This class is in charge of registering the rules in the system. To do so, the IpTablesManager class starts the "iptables-restore" program when the server is started and then new rules are written through a file pipe to the "iptables-restore" program which provides a real time interface with iptables.

NetworkInterfaceManager

This class monitors the status of the network interfaces and notifies the RulesDB class every time an event occurs. To monitor the status of the interfaces it uses the NetworkManager's DBus interface.

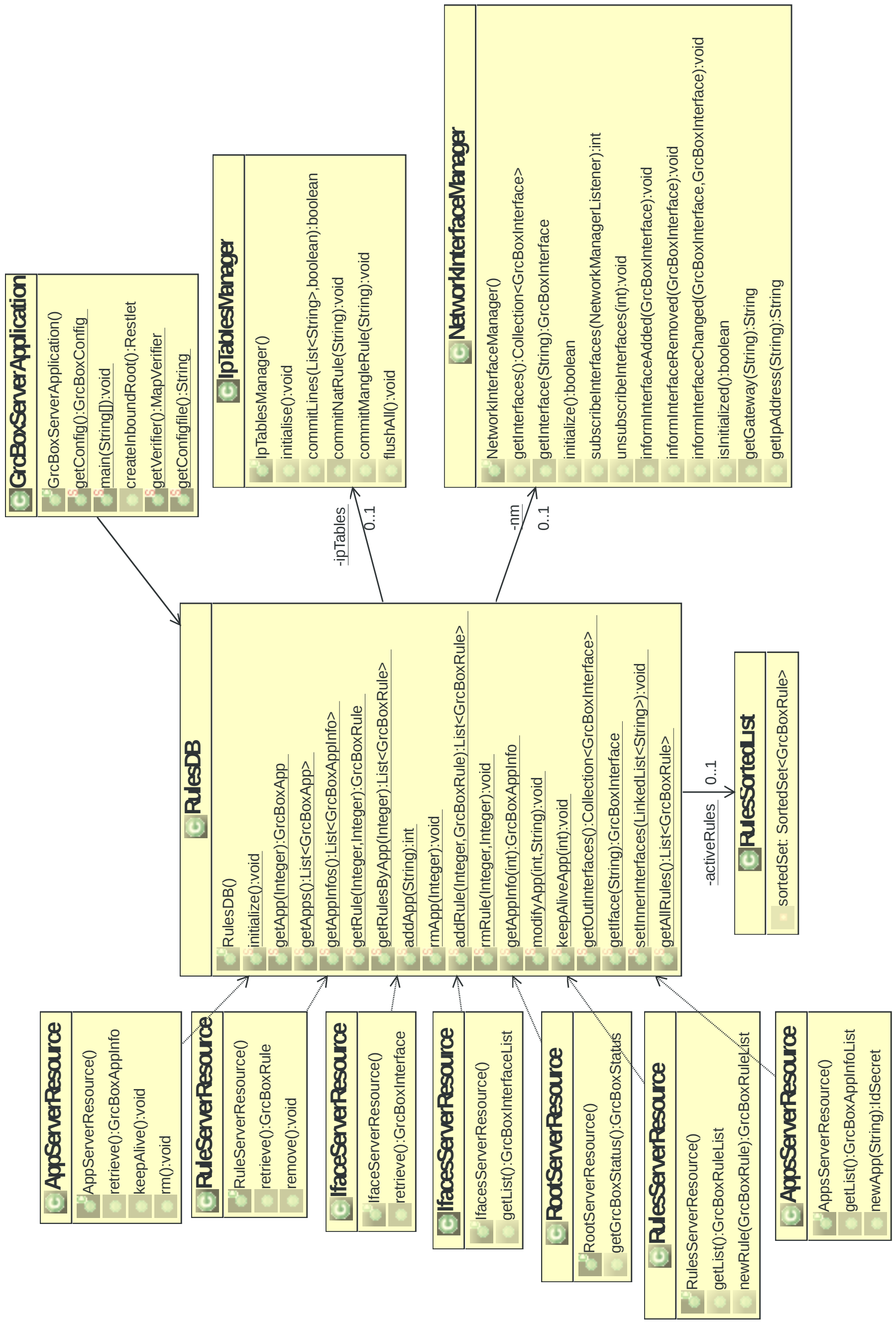

GrcBoxServerApplication
GrcBoxServerApplication()
getConfig():GrcBoxConfig
main(String[]):void
createInboundRoot():Restlet
getVerifier():MapVerifier
getConfigfile():String
IpTablesManager
IpTablesManager()
initialise():void
commitLines(List<String>,boolean):boolean
commitNatRule(String):void
commitMangleRule(String):void
flushAll():void
NetworkInterfaceManager
NetworkInterfaceManager()
getInterfaces():Collection<GrcBoxInterface>
getInterface(String):GrcBoxInterface
initialize():boolean
subscribeInterfaces(NetworkManagerListener):int
unsubscribeInterfaces(int):void
informInterfaceAdded(GrcBoxInterface):void
informInterfaceRemoved(GrcBoxInterface):void
informInterfaceChanged(GrcBoxInterface,GrcBoxInterface):void
isInitialized():boolean
getGateway(String):String
getIpAddress(String):String
-ipTables
0..1
-nm
0..1
RulesDB
RulesDB()
initialize():void
getApp(Integer):GrcBoxApp
getApps():List<GrcBoxApp>
getAppInfos():List<GrcBoxAppInfo>
getRule(Integer,Integer):GrcBoxRule
getRulesByApp(Integer):List<GrcBoxRule>
addApp(String):int
rmApp(Integer):void
addRule(Integer,GrcBoxRule):List<GrcBoxRule>
rmRule(Integer,Integer):void
getAppInfo(int):GrcBoxAppInfo
modifyApp(int,String):void
keepAliveApp(int):void
getOutInterfaces():Collection<GrcBoxInterface>
getIface(String):GrcBoxInterface
setInnerInterfaces(LinkedList<String>):void
getAllRules():List<GrcBoxRule>
-activeRules
0..1
RulesSortedList
sortedSet: SortedSet<GrcBoxRule>
AppServerResource
AppServerResource()
retrieve():GrcBoxAppInfo
keepAlive():void
rm():void
RuleServerResource
RuleServerResource()
retrieve():GrcBoxRule
remove():void
IfaceServerResource
IfaceServerResource()
retrieve():GrcBoxInterface
IfacesServerResource
IfacesServerResource()
getList():GrcBoxInterfaceList
RootServerResource
RootServerResource()
getGrcBoxStatus():GrcBoxStatus
RulesServerResource
RulesServerResource()
getList():GrcBoxRuleList
newRule(GrcBoxRule):GrcBoxRuleList
AppsServerResource
AppsServerResource()
getList():GrcBoxAppInfoList
newApp(String):IdSecret

Figure 6.4: GRCBoxServer UML diagram.

6.4 Summary

In this chapter we have presented our GRCBox architecture. The GRCBox architecture enables developers to make the most of opportunistic communications using in-vehicle user devices. With the GRCBox users can create their own VANETs to communicate with their peers using their laptops, smartphones or tablets. Our proposal is implemented in a low-cost RaspberryPi hardware device and focus on providing connectivity to several networks while the core of the applications runs on user devices. In Chapter 9 we will present some examples of new applications implemented using the GRCBox. Besides, we will present some performance results to illustrate the low impact on communication performance.

Part III

Results & Experimentation

Chapter 7

Performance Evaluation of the MSDP Protocol

The contents of this chapter have been partially published in:

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. “A Map-based Sensor Data Delivery Protocol for Vehicular Networks”. In: *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. Ayia Napa, Cyprus: IEEE, June 2012, pp. 1–8. ISBN: 978-1-4673-2039-9. DOI: 10.1109/MedHocNet.2012.6257118.

- S. M. Tornell, E. Hernández-Orallo, C. T. Calafate, J.-C. Cano, P. Manzoni, and E. Hernández. “An Analytical Evaluation of a Map-based Sensor-data Delivery Protocol for VANETs”. In: *14th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM 2013)*. Madrid, Spain, June 2013. ISBN: 9781467358286. Core A Conference.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Assessing the Effectiveness of DTN Techniques Under Realistic Urban Environments”. In: *38th Annual IEEE Conference on Local Computer Networks (LCN 2013)*. Sydney, Australia, Oct. 2013. Core A Conference.

In Chapter 3 we presented the Map-based Sensor-data Delivery Protocol (MSDP). MSDP is a new VDTN routing protocol that combines information obtained from the Geographic Information Service (GIS) with the actual street/road layout obtained from the Navigation System (NS) to define a new routing metric. In this chapter we evaluate the performance of our proposal from two different points of view. First, we compare MSDP with the Epidemic protocol [139] using analytical models. Later, we compare it with the Greedy-DTN, and the GeOpps[74] protocols using the GOD model presented in Chapter 4. Both, analytic and simulation comparisons, confirm that our proposal, the MSDP, performs better than the compared protocols.

7.1 Analytical Evaluation

In this section we model the performance of the MSDP and the Epidemic routing using Markov chains. The goal is to obtain the time that a packet needs to be delivered to the destination nodes (that is, the RSU nodes) and the cost (the number of hops or transmitted messages). Using this model we can compare our MSDP scheme with the Epidemic routing approach. The Epidemic routing is optimal in delivery time, but it assumes that all nodes have sufficient space to store all packets. However, mobile nodes have limited storage capacity, so we also compare the MSDP routing with the more realistic constrained buffer Epidemic routing (we call it, the *restricted* epidemic routing).

For our models we assume that the rate of contacts between two mobile nodes and a mobile node and a static node (that is, the RSU node) follows an exponential distribution. Recent works show that the occurrence of contacts between two mobile nodes follows an exponential distribution with rate λ [47, 153, 76]. This has been shown valid specially for VANETs, considering vehicle-to-vehicle communications as well as with the roadside infrastructure (vehicle-to-roadside communications) [153]. There is some controversy about whether or not this exponential distribution can reflect some real mobility patterns. Empirical results have shown that the aggregated inter-contact times distribution follows a power-law and has a long tail [17]. In [13] it is shown that, in a bounded domain (such as the one selected along this paper), the inter-contact distribution is exponential but in an unbounded domain, it follows a power-law distribution instead. The work in [37] analyzed some popular mobility traces and found that over 85% of the *individual pair distributions* fit an exponential distribution. Therefore, we consider that using an exponential fit is a good choice to model inter-contact times. Moreover, by using exponential distributions we can formulate analytical models using Markov chains.

The network is modeled as a set of M wireless mobile nodes and R fixed destination nodes (RSU nodes). There are two vehicles contact rates: λ_M is the mean contact rate between mobile nodes (that is, inside the set of M nodes) and λ_R is the mean contact rate between mobile nodes and RSU nodes (that is, between the two sets). Upon contact, the packet can be transmitted. Nevertheless, a contact does not always imply a transmission. There are several factors that can reduce this transmission, for example the contact duration is too short to transmit the packet, other packets are transmitted before, or error transmissions occur. Thus, we introduce two new parameters into the model: the probability that a packet is successfully transmitted (or forwarded) between mobile nodes (p_{tM}), and the probability of transmission between a mobile node and the RSU nodes (p_{tR}).

7.1.1 Modelling Epidemic diffusion

In this subsection we derive a model for evaluating the time and cost of reaching the destination node when using epidemic routing. First, we introduce a model for unrestricted epidemic diffusion (there is no buffer limitation in the nodes), and then we introduce a model for constrained buffer epidemic diffusion.

Several models has been proposed to evaluate the performance of Epidemic routing. Markov chain models were introduced in [47] for epidemic routing and 2-hop forwarding, deriving the average source-to-destination delivery delay and the number of existing copies of a packet at the time of delivery. The model in [150], which is based on Ordinary Differential Equations (ODE), obtained similar results. The previous models assume that all nodes are mobile with a unique contact rate and full probability of transmission when a contact occurs ($p_{tM} = 1$). Thus, we extend the Markov Chain model to include the mobile and destination set of nodes with their different contact rates (λ_M and λ_R) and the probabilities of transmission (p_{tM} and p_{tR}).

The basis of the model is a 2D Continuous Time Markov chain (2D-CTMC) with states $(d(t), m(t))_{t\geq 0}$, where $m(t)$ and $d(t)$ represent respectively the number of mobile and destination nodes that have the packet at time t. At the beginning only one mobile node (the sender node) has the packet. Then, when a mobile contact occurs, m can be increased by one with probability p_{tM}. Alternatively, when a mobile contacts with a destination node (with rate λ_R), d can be increased by one with probability p_{tR}. The final absorbing states are when $d > 0$. Thus, this 2D-CTMC has an initial state $s_1 = (0, 1)$, M transient states (from $s_1 = (0, 1)$ to $s_\tau = (0, M)$ states) and M absorbing states (from $s_{\tau+1} = (1, 1)$ to $s_{\tau+\upsilon} = (1, M)$)[1]. We define τ as the number of transient states ($\tau = M$) and υ as the number of absorbing states ($\upsilon = M$). This model can be expressed using the following transition matrix $\mathbf{P}$ in the canonical form:

$$\mathbf{P} = \begin{pmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{7.1}$$

where $\mathbf{I}$ is a $\upsilon \times \upsilon$ identity matrix, $\mathbf{0}$ is a $\upsilon \times \tau$ zero matrix, $\mathbf{Q}$ is a $\tau \times \tau$ matrix with elements p_{ij} denoting the transition rate from transient state s_i to transient state s_j and $\mathbf{R}$ is a $\tau \times \upsilon$ matrix with elements p_{ij} denoting the transition rate from transient state s_i to the absorbing state s_j.

Now, we derive the transition rates p_{ij}. Given the state $s_i = (d, m)$[2] the following transitions can occur:

- (d, m) to $(d, m + 1)$: A new mobile node has the packet, due to a contact between mobiles nodes with rate λ_M. Thus, the transition probability is

[1]Note that each state number i is mapped as $i = d \cdot M + m$

[2]For simplicity, we omit the time in the states (that is $(d, m) = (d(t), m(t))$)

$t_m = \lambda_M p_{tM} \cdot m(M-m)$ where $(M-m)$ represents the number of pending mobiles nodes that can receive the packet.

- $(0,m)$ to $(1,m)$: An RSU node has the packet, due a contact between a mobile node and a destination node with rate λ_R. Thus, the transition probability is $t_r = \lambda_R p_{tR} \cdot mR$.

- (d,p) to (d,p): This is the probability of no changes and is $1 - \sum_{j \neq i} p_{ij}$.

Using the transition matrix $\mathbf{P}$ we can derive the delivery time T_d. From the 2D-CTMC, we can obtain how long will it take for the process to be absorbed. Using the fundamental matrix $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$, we can obtain a vector $\mathbf{t}$ of the expected time to absorption as $\mathbf{t} = \mathbf{N}\mathbf{v}$, where $\mathbf{v}$ is a column vector of ones ($\mathbf{v} = [1, 1, \dots, 1]^T$). Each entry t_i of $\mathbf{t}$ represents the expected time to absorption from state s_i. Since we only need the expected time from state $s_1 = (0,1)$ to absorption, the delivery time T_d, is:

$$T_d = E[T] = \mathbf{v_1}\mathbf{N}\mathbf{v} \tag{7.2}$$

where T is a random variable denoting the delivery time for all nodes and $\mathbf{v_1} = [1, 0, \dots, 0]$.

Now, we calculate the overhead, that is, the mean number of copies (or replicas) of the packet until the delivery time. If we assume that a packet is not transmitted again to a node that already has it, the number of copies is equivalent to the number of transmissions. Therefore, the number of copies is done calculating the average number of packets transmitted in each state s_i. To do this, we obtain the duration of each state s_i using the fundamental matrix $\mathbf{N}$. By definition, the elements of the first row of $\mathbf{N}$ are the expected times in each state starting from state 0. Then, the duration of state s_i is $\mathbf{N}(1,i)$. In state $s_1 = (0,1)$ only one node has the packet, and this packet can be transmitted to all nodes (except himself), that is $M-1$ nodes, for the duration of this state (denoted as $\mathbf{N}(1,1)$) with a rate λ_M and probability p_c. Then for state $s_2 = (0,2)$ two nodes have the packet and it can be transmitted to $M-2$ nodes. Thus, for state $s_i = (0,m)$, $i \leq \tau$, the average number of copies in this state is $\lambda_M p_{tM} \cdot \mathbf{N}(1,i) \cdot m(M-m)$. Summing up, the overhead (or the expected number of copies) is:

$$O_d = E[C] = \lambda_M p_{tM} \sum_{m=1}^{\tau} \mathbf{N}(1,i) m(M-m) \tag{7.3}$$

Note that previous expressions for time and copies obtain the same results than equations in [150] when $\lambda_M = \lambda_R$, $R = 1$ and $p_{tM} = p_{tR} = 1$, that is, $T_d = \frac{\log M}{\lambda(M-1)}$ and $O_d = \frac{M-1}{2}$.

7.1.2 Constrained buffer

In the *unrestricted* epidemic routing there are no constraints on the number of packet replicas in the network. Now we derive a model for Epidemic routing under constrained buffer (the *restricted* epidemic routing). In this case, we assume that mobile nodes have a limited buffer of size B (that is, they can only store B packets). For the destination nodes, we keep the assumption of unrestricted buffer size (they are fixed nodes, so memory is not a problem).

First, we need to obtain the average buffer occupancy. We consider the approximation derived in [150] for the case of F unicast flows. Each flow generates packets following a Poisson process with rate δ. Then, the average queue size is:

$$E[Q] = \frac{F\delta}{M\lambda_M} 2E[C] \tag{7.4}$$

Using this expression we simply define a new probability of transmission P_t that will depend on the average buffer occupancy. That is, if the buffer is full then we can transmit the packet if another one is dropped from the buffer. Assuming a random dropping, we have the following probability of transmission:

$$P_t = \begin{cases} p_{tM} & E[Q] < B \\ \frac{B}{E[Q]+1} p_{tM} & E[Q] \geq B \end{cases} \tag{7.5}$$

This value is used for calculating the transition probability of (d, m) to $(d, m+1)$, $t_m = \lambda_M P_t \cdot m(M - m)$. Using this new transition probability we can obtain the time and overhead using equations 7.2 and 7.3. Note, that in order to obtain $E[Q]$ we need a prior value of $E[C]$, that is one of the results of the model. So this value is iteratively approximated from an initial value $E[C]^0$ obtained with the unrestricted epidemic model, and then calculating values of $E[C]^{x+1}$ using the restricted epidemic model with $E[C]^x$ until a given convergence criteria is reached (that is, the difference between the successive values is less than a given error ϵ)

7.1.3 Modeling MSDP

Now, we are going to model our MSDP protocol. Without loss of generality we focus our study to only one destination node ($R = 1$). In the MSDP protocol there is only one packet in the network that is stored in the *custodian* node. When a contact occurs the packet is transmitted to a new node if the *UtiliyIndex* (UI) of the receiver (*candidate*) node is greater than the UI of the sender node. This way the UI reflects how near is a node to the destination RSU node. Basically, the higher the UI, the nearer to the RSU node. Figure 3.1 shows an example of packet delivery. It starts with m_1 as the sender node. When a contact occurs with m_2, this node has an UI greater than m_1 so the packet is transmitted (first hop, H_1). For the following hop (H_2), the UI of the node that has the packet is increased.

Finally, the packet reaches the destination node. The UI has another property, the locality. Two nodes with similar UI values are prone to be neighbors. So it is more frequent that a contact occurs between these nodes and other nodes in their neighborhood (that is, they have a greater contact rate). Following the example of figure 3.1, when the packet is in m_1, the contact rate with all the nodes inside the circumference defined by the boundary of the node is λ_{M1}. When the packet is transmitted to m_2, the circumference is reduced and the contact rate is increased $\lambda_{M2} \geq \lambda_{M1}$. Therefore, we expect a direct relation between the UI of a node and the contact rate.

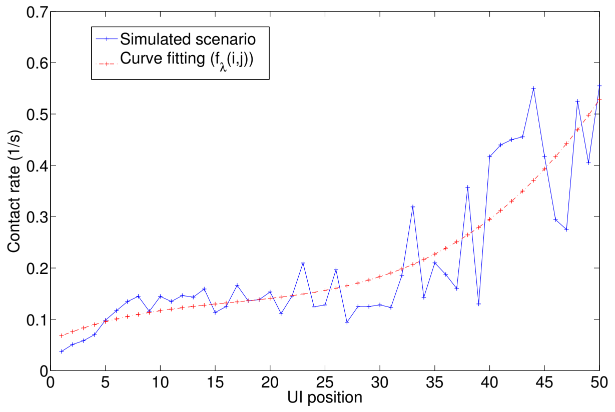


Figure 7.1: Contact rate depending on the UtilityIndex.

This is confirmed with the following experiment. From the simulation scenario used in section 7.2 we obtained all the inter-contact times between nodes and the UI of the sender node when a contact occurs. If we sort the nodes by increasing UI values, the UI position is the index on this list. Figure 7.1 shows the plot of the contact rate depending on the UI position for M nodes. We can clearly observe that the contact rate increases with the UI position. That is, if we have M mobile nodes, the list is: $\{\mathrm{UI}_1, \mathrm{UI}_2, \ldots \mathrm{UI}_i, \mathrm{UI}_j, \ldots \mathrm{UI}_M\}$, so $\mathrm{UI}_j \geq \mathrm{UI}_i \, \forall j > i$. This list is dynamic, so a node can change its position over time. The probability of changing one position is defined as p_u. As we sort nodes by their UI values, we can establish a direct relation between the UI position and the contact rate. Thus, we can fit a third degree polynomial function $f_\lambda(i,j)$, that gives the contact rate of two nodes with position index i and j:

$$f_\lambda(i,j) \approx c_4 + c_3 k + c_2 k^2 + c_1 k^3 \quad k = \min(i,j) \quad i \neq j \tag{7.6}$$

Note that $k = \min(i,j)$ reflects the fact that the contact rate for two nodes is determined by the lowest index. In the example of figure 7.1, the contact rate between nodes m_1 and m_3 is λ_{M1}. Finally, figure 7.1 shows the result of fitting this curve to the values obtained from the scenario. We can see the effect of the logarithm effect in the calculus of the time to reach factor (T) of the UI expression, specially for UI positions greater than 30.

The contact rate for the destination nodes (RSU nodes) follows a similar distribution, so the higher the index i of a node the higher the contact rate, and we can also fit a similar equation $f'_\lambda(i)$.

Using a CMTC we can obtain the time to reach the destination and the overhead (in this case, the number of hops until the packet arrives to any of the RSU nodes). We introduce H as the maximum number of possible hops ($H \leq M$). Following the same process that in the epidemic model, we have a 3D-CMTC with states $(d(t), u(t), h(t))_{t \geq 0}$ where $h(t)$ is the number of hops at time t, $u(t)$ is the position on the list of UI at time t and $d(t)$ represents if the destination node have the packet at time t. At the beginning we start with $h = 0$ hops, but we assume that the sender can be any node of the mobile nodes so its average index position u is in the middle: $\lfloor M/2 \rfloor$. Therefore, the starting state is $s_\alpha = (0, \lfloor M/2 \rfloor, 0)$[3]. The final (absorbing) states is when $d = 1$. Thus, this 3D-CTMC has $M(H+1)$ transient states (from $s_1 = (0, 1, 0)$ to $s_\tau = (0, M, H)$ states) and $M(H+1)$ absorbing states (from $s_{\tau+1} = (1, 1, 0)$ to $s_{\tau+\upsilon} = (1, M, H)$).

Now, we derive the transition rates p_{ij}. Given the state $s_i = (d, m, h)$ the following transitions can occur:

- (d, u, h) to $(d, u + \Delta, h + 1)$, $\Delta = 1 \ldots (M - u)$: The packet is transmitted to a new node with a greater UI. The contact rate depends on the difference of the UI of the nodes contacted: $f_\lambda(i, j)$. Thus, the transition probability is $t_{uh} = f_\lambda(i, j) \cdot p_{tM}$.
- (d, u, h) to $(d, u \pm 1, h)$: This transition reflects that the node that has the packet increases (decreases) one position in the UI list. The transition probability is simply p_u.
- $(0, u, h)$ to $(1, u, h + 1)$: An RSU node has the packet. The contact rate depends on the value of u : $f'_\lambda(u)$. Thus, the transition probability is $t_d = f'_\lambda(u) \cdot p_{tR}$.
- (d, u, h) to (d, u, h): This is the probability of no changes and is $1 - \sum_{j \neq i} p_{ij}$.

Using the transition matrix $\mathbf{P}$ we derive the delivery time T_d using an expression similar to equation 7.2:

$$T_d = E[T] = \mathbf{v}_\alpha \mathbf{N} \mathbf{v} \tag{7.7}$$

where $\mathbf{v}_\alpha$ is a vector with a 1 in the start state α.

[3]We can convert from an state (d, u, h) to a state number i using the following expression: $i = State(d, u, h) = d \cdot M(H+1) + u(H+1) + h + 1$, so the starting state number α is $\lfloor M/2 \rfloor (H+1)$

Now, we derive the overhead (the number of hops or retransmissions). First, we obtain the matrix of absorption probabilities as $\mathbf{B} = \mathbf{N} \cdot \mathbf{R}$. Then, we obtain the probability of absorption (p_H) depending on the number of hops starting from state number α:

$$p_H(h) = \sum_{u=1}^{M} B(\alpha, State(0, u, h)) \qquad h = 1 \ldots H \tag{7.8}$$

Thus, p_H is the probability mass function (*pmf*), that gives the probability of absorption (that is, the packet reaches the destination) with h hops. Using this *pmf* we can obtain the cumulative distribution functions $F_H(h)$. Then, the average number of hops needed to reach the RSU node $E[H]$ is the greater value of h that make true the expression $F_H(h) \leq 0.5$. That is:

$$O_d = E[H] = \max\{h \mid F_H(h) = 0.5\} \tag{7.9}$$

As in the Epidemic model, we can also consider the effect of the buffer, although in the MSPD its influence will be limited. Assuming the same F unicast flows, the arrival rate of new packet to the network is $F\delta$, and by Little's law, the average number of packets in the system is $F\delta E[T]$, where $E[T] = T_d$ is precisely the average packet lifetime. If all these packets are equally divided among the M nodes we have that the average queue size is:

$$E[Q] = \frac{F\delta}{M} E[T] \tag{7.10}$$

Using this average queue size we can obtain the probability of transmission P_t using equation 7.5 in order to calculate the new transition probability $t_{uh} = f_\lambda(i, j) \cdot P_t$.

7.1.4 Analytical Performance Evaluation

Using the models previously developed we made an analytical comparison of the performance of the MSDP protocol with the epidemic routing approaches. In this evaluation we use the following parameters that were derived from the simulation scenario described previously: $\lambda_M = 0.141$, $\lambda_R = 0.046$, $p_{tM} = 0.5$, $p_{tR} = 0.7$, $p_U = 0.05$, $H = 20$, $R = 1$. The coefficients of the f_λ functions were obtained through a curve fit based on the simulation results, as shown previously. Figure 7.3 presents the time and overhead depending on the number of mobile nodes. In figure 7.2 we can see the delivery time. Regarding the unrestricted protocols, results show that for MSDP the delivery time is about ten times greater than for Epidemic. Note that the epidemic routing is optimal in delivery time, but has a great overhead, as we can see in figure 7.3. The average number of transmissions for the Epidemic protocols increases linearly with M, while for MSDP it increases more slowly.

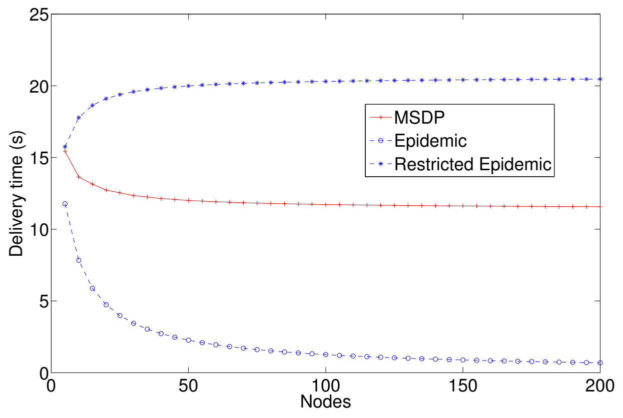


Figure 7.2: Delivery time.

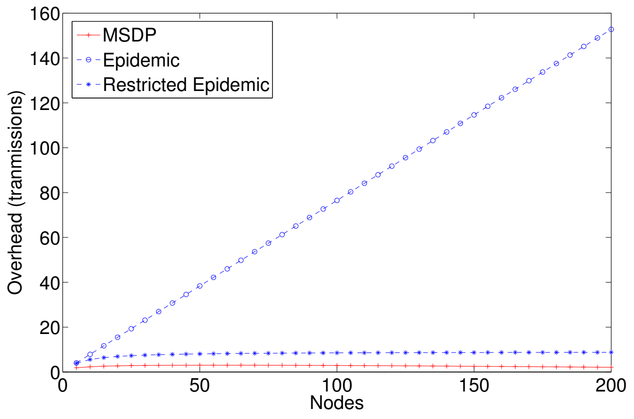


Figure 7.3: Overhead: average number of transmitted packets.

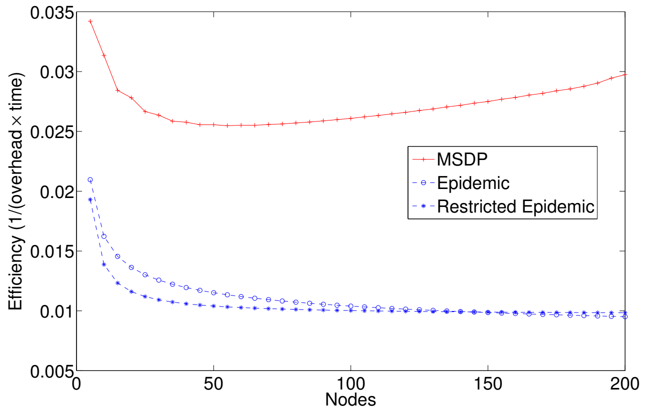


Figure 7.4: Efficiency of the protocols.

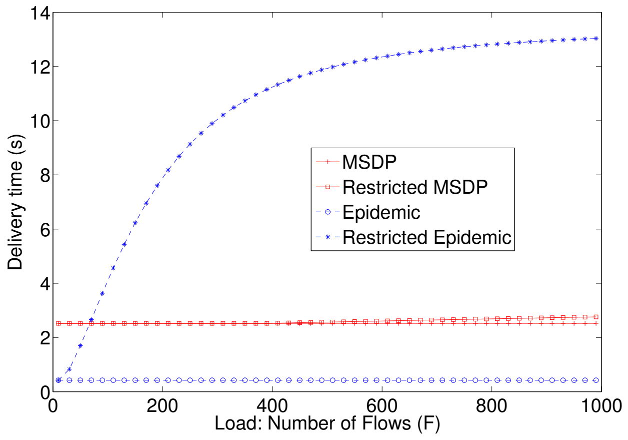


Figure 7.5: Delivery time depending on load (F =number of flows).

The results for buffer restricted epidemic protocols are totally different. We used the following values: a buffer of 50 packets ($B = 50$) and all nodes sending a message to the RSU ($F = M$) every five seconds (a similar load that in the simulated scenario). This message is fragmented in ten packets so $\delta = 2$. Note that MSDP has only one copy of each packet in the network, and so this does not imply an increase on network load; also, the effect of the buffer restriction in this evaluation is negligible. Regarding the overhead, we also see that the number of copies is reduced when the load increases. Finally, in figure 7.4 we can see the efficiency of the protocols obtained as $(O_d \times T_d)^{-1}$, so a higher value implies a more efficient protocol. Thus, MSDP is about 10-20 times more efficient than the epidemic protocols.

Figure 7.5 shows the delivery time depending on the number of flows (F) in a network with 100 mobile nodes ($M = 100$) using the same parameters of previous experiments. We can see an exponential growth of the time for low values of F. The limit is reached when the network buffers are saturated, so the delivery is made through a direct contact between the sender and the receiver. For restricted MSDP, as only one copy of each packet is present in the network, the effect is reduced, as we can appreciate in the same figure. The increase on the delivery time is minimal (about 5% for 1000 flows of load).

The previous evaluations show that the effect of network load has low influence on the efficiency of the MSDP protocol, allowing to obtain good delivery times in a very efficient way.

7.2 Simulation Based Evaluation

In this Section we use our Generic One-Copy DTN Model (GOD) to compare our MSDP against the *Greedy-DTN* and *GeOpps*[74] protocols through simulations. The Greedy-DTN and GeOpps protocols were described in detail in subsection 2.4.3.

We consider as the reference scenario the one selected in [127]. In that work vehicles used a DTN protocol to collect information from a vehicular sensor network and to deliver it to a remote control center. The information is obtained from in-vehicle sensors, which retrieve data using an OBD-II unit [56]. The information messages are then fragmented and routed. A fragment is considered to be delivered when any of the RSUs correctly receive it. Once an RSU receives a fragment, the fragment is sent over the backbone network to the control center. The control center will then reassemble the fragments into the original message and process the content. RSUs are supposed to be placed in strategical places by entities interested in collecting the information, like city councils, or road administrators. The locations of the RSUs are available to vehicles' NSs through a dynamic updating service.

We implemented all the models using the Inet framework for the Omnet++

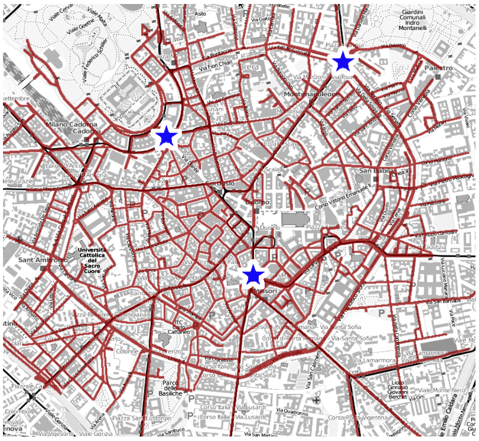


Figure 7.6: Map used in our simulations(2.6 x 2.6 km), stars indicate the location of the RSUs.

event-driven simulator [100]. The Inet framework includes detailed implementations of the 802.11 physical and MAC layers.

In our simulations we have generated the node mobility by using VaCaMobil [7]. The mobility of the nodes is simulated using Simulation of Urban MObility (SUMO) [9] according to mobility models originated in the field of transportation. For the layout we used a 6.7 km^{2}area map of the city center of Milan, which has a typical European old city structure. Figure 7.6 shows the portion of the map used, and the locations of the RSUs are indicated by stars.

We consider that the use of a very simplistic propagation model is one of the main drawbacks of previous studies in this topic. To accurately model real world conditions, we used the propagation model presented in [5], which combines the Nakagami fading model [108] with a visibility model that deals with power losses due to the effect of obstacles.

Every node in our network scenario generates a 2 kBytes data message every 10 seconds. The GOD has been configured to generated a new beacon every second. The fragment size is 450 Bytes. The simulation lasts 3600 seconds, and nodes will generate traffic throughout the entire simulation. Concerning the communication interfaces, each node has two 5.9 GHz 802.11p interfaces tuned at different channels, where the first one is used for beacon broadcasting, while the second one is used for one-hop transmissions. This double-interface model mimics the multi-channel communications scheme for Dedicated short-range communications (DSRC) [60]. The transmission power of both interfaces is configured to 18 dBm, while the antenna gain is 5 dBi, making a total Equivalent isotropically radiated power (Eirp) of 23dBm which is the maximum allowed Eirp in the USA for the operation channel.

To achieve reasonably conclusive results, we have simulated every scenario 50 times, varying the seed of the simulation. Measurements are represented with a 95% confidence interval.

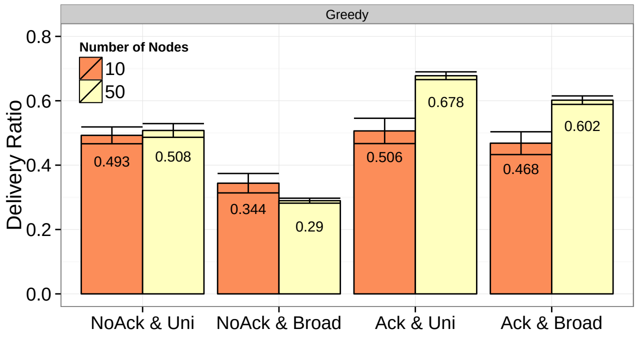
7.2.1 Results

To determine the most relevant reference scenario, we first analyzed the impact of: (a) using ACK messages, and (b) using broadcast or unicast communications. The best-performing transmission mode is then used to compare the performance of MSDP, Greedy-DTN and GeOpps for different network densities. We evaluate the delivery ratio, the overhead, and the end-to-end delay for each configuration scheme.

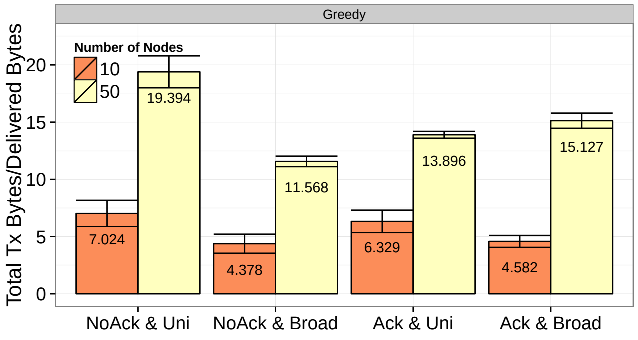
7.2.2 Evaluating the impact of ACK messages

As explained in Chapter 2, even simple mechanisms such as the use of one-hop ACK messages to confirm that another node will carry the forwarded fragment, or whether the implemented protocol uses broadcast or unicast communication, can heavily impact the performance of a DTN protocol. In order to evaluate its impact we have simulated four different cases: 1) without ACK messages and using unicast communication, 2) without ACK messages and using broadcast communication, 3) using ACK messages and unicast communication, 4) using ACK messages and broadcast communication. To take scalability into consideration, we varied the number of nodes introduced in our network from 10 to 50, therefore varying the node density from 1.47 to 7.4 nodes/km^2.

The first metric we used was the *Delivery Ratio*, *i.e.* the number of messages successfully received by an RSU divided by the total number of messages sent. Figure 7.7a shows the result we obtained for the Greedy-DTN protocol. Case 2, *i.e.* without ACK messages and using broadcast communication, presents the lowest delivery ratio for both shown densities. The other 3 cases present a similar performance when only 10 nodes are deployed in the network. However, when we increase the amount of nodes in the network up to 50, case 3, *i.e.* using ACK messages and unicast communication, clearly stands up as the best option. Figure 7.7b shows the overhead added in the four cases, we may miss-conclude that case 2 is the best option, since it requires fewer bytes per delivered byte; however, this result is heavily biased by its low delivery ratio. Eventually, since the three other cases introduce a similar overhead, we conclude that the best option is case 3, *i.e.* using ACK messages and unicast communication, since it is the one that experiences the best delivery ratio at a similar cost in terms of resources. From now on, all the evaluated protocols will be configured as in case 3.



(a) Delivery Ratio.



(b) Overhead.

Figure 7.7: Comparison between different ACK and Broadcast/Unicast combinations.

7.2.3 Evaluating the impact of network density

We now use the reference best performing transmission mode, determined in the previous subsection, to compare the three evaluated protocols according to: the delivery ratio, the end-to-end delay of delivered messages, and the consumed resources (overhead). To evaluate the impact of node density we again vary the number of nodes in the network from 10 to 50, thus varying the node density from 1.47 to 7.4 nodes/km^2.

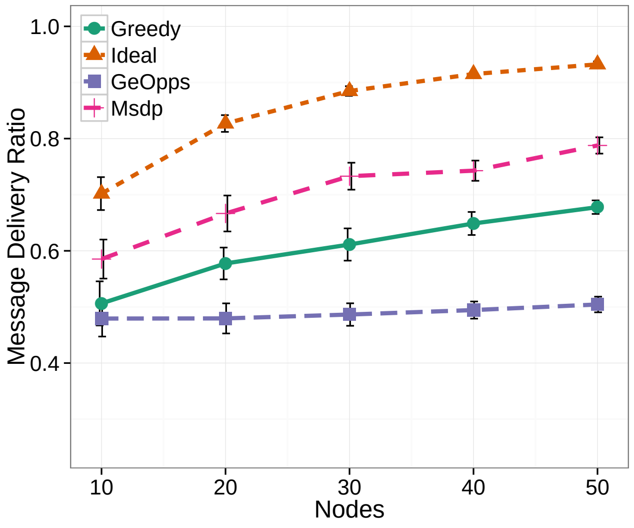


Figure 7.8: Delivery Ratio.

Delivery Ratio

Due to mobility-related effects, it is typically impossible to achieve a perfect delivery ratio. To determine a reference upper bound for this metric we have implemented an *Ideal Protocol*. Our *Ideal Protocol* is an implementation of the Epidemic protocol [139] that runs on the top of a collision-free MAC layer which only considers propagation and transmission delays, and defines a limited transmission range. Since a copy of every fragment is sent to all neighbors every time a contact occurs, the first copy of a fragment arriving to any of the sinks must have traversed the best possible path in terms of delay.

Figure 7.8 shows the delivery ratio of the different protocols; it is worth noticing

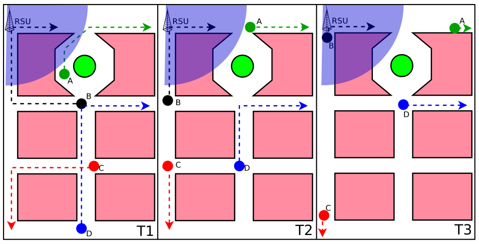


Figure 7.9: Node D has a packet to be delivered to the RSU.

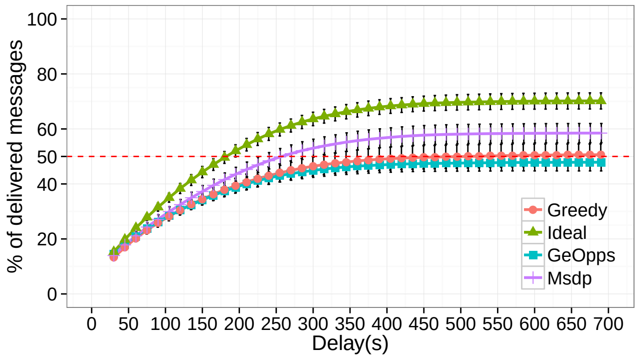
that not even the Ideal protocol reaches a perfect delivery ratio. When there are very few nodes in the network, the delivery ratio depends a lot on the mobility patterns of nodes, which explains the high confidence intervals obtained. On the other side, as we increase the number of nodes in the network, the delivery ratio also increases, because more transmission opportunities become available.

As can be seen, MSDP performs better than any other of the compared protocols, and GeOpps performs worse than Greedy-DTN. One of the reasons why MSDP outperforms the other proposals is shown in Figure 7.9. When there is only a fragment stored in node D's buffer to be delivered, and assuming that the communication range is approximately equal to the width of a block of buildings (pink rectangles), the considered protocols would behave as follows:

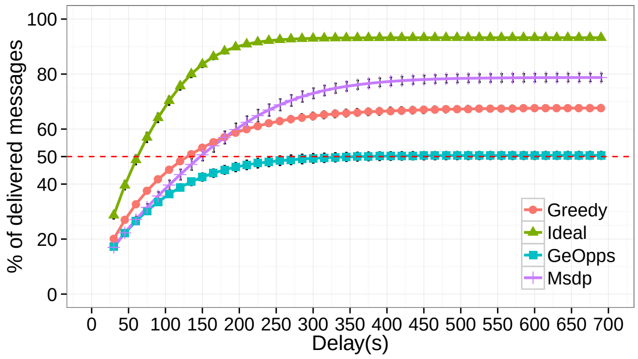
- When using the *Greedy-DTN* protocol, the fragment would be immediately forwarded to node A, which is the node closest to the RSU. Then, in $T2$ and $T3$, node A would move away from the RSU with no chance of forwarding the message to any other node.
- When using GeOpps, node D would keep the message in its buffer, since in $T1$ and $T2$ its Minimum Estimated Time of Delivery (METD) is better than the one of node C. However, at $T3$, when node D arrives to its closest point, it would not find any neighbor to forward the message.
- When using MSDP, in $T1$ the UtilityIndex of node C would be bigger than the UtilityIndex of node D due to the second term of the equation 3.1. Therefore, the fragment is forwarded to node C. Then, according to figure 4.7a, node C would immediately start a new transmission, forwarding the fragment to node B, which would keep the fragment in its buffer, since its UtilityIndex is bigger than the ones of nodes A and C. In $T3$ node B would be able to deliver the fragment to the RSU.

Delay

The delay is the time elapsed since a message is generated until it is successfully received by an RSU. To evaluate the delay suffered by messages when using each of the protocols, we decided to represent the delay cumulative distribution, which shows the percentage of sent messages that experience a delay smaller than the one represented in the X axis. Figures 7.10a and 7.10b show the results we obtained for two different vehicle densities. When only 10 nodes are introduced in the network, which results in a very low node density, our MSDP delivers messages faster than any of the other protocols, for example, MSDP delivers 50% of the messages in less than 250s, while the Greedy-DTN protocol delivers less than the 45% of the messages during that same period. When the number of nodes increases up to 50 nodes (7.4 nodes/km^2), the connectivity of the network increases, and, therefore,



(a) 10 Nodes.



(b) 50 Nodes.

Figure 7.10: Cumulative distribution of data message delay.

the Greedy-DTN protocol is able to deliver more messages quickly. In fact, the Greedy-DTN protocol delivers the 50% of the packets in less time than our MSDP; however, MSDP is able to achieve a better final delivery ratio on the long term. Since MSDP makes an intensive use of node mobility, the delay experienced by messages, compared to the Greedy-DTN protocol, is slightly incremented when a bigger number of direct multihop routes are available. This fact also explains why the difference between the Ideal model and our MSDP is bigger as the density of nodes increases.

Overhead

Last but not the least, we evaluate the overhead introduced by each protocol. To obtain the overhead ratio of the evaluated protocols, we divide the total number of

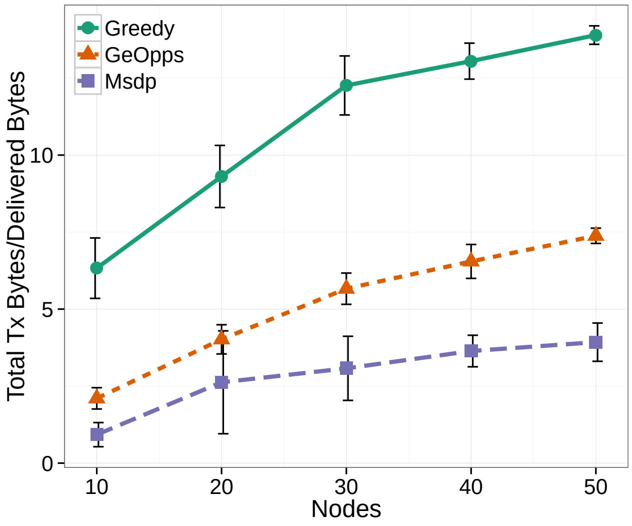


Figure 7.11: Total transmitted Bytes per delivered Bytes.

transmitted bytes by the number of delivered bytes. This measurement is closely related with the number of hops a fragment must traverse before it arrives to an RSU. Since it does not take into account the amount of data interchanged between nodes and RSUp, its value may be smaller than one when most of the delivered fragments are directly transmitted to an RSU.

Figure 7.11 shows the results we obtained for the different protocols; We appreciate that our MSDP is the protocol experiencing the smallest overhead ratio. Both GeOpps and MSDP perform better than the Greedy-DTN protocol, which consumes significantly more resources. This difference is explained by the presence of loops; when two nodes move in opposite directions, one node moving away from the RSU, and another one moving close to it, the fragments are sent firstly to the closest one and then, after they cross, the fragments are sent back to the original carrier, which is moving closer to the RSU; the same situation occurs when a vehicle overtakes another vehicle. The UtilityIndex of MSDP, as well as the METD of GeOpps, are more stable and, therefore, most loops are avoided.

7.3 Summary

In this chapter we have evaluated the performance of our proposed Map-based Sensor-data Delivery Protocol (MSDP). To evaluate our protocol we first compared it analytically with the Epidemic routing protocol, then, by using simulations, we compared the MSDP with the Greedy-DTN protocol and the GeOpps protocol.

Our analytical results shown that the MSDP protocol scales better than the Epidemic protocol when the size of the network increases.

To fairly compare MSDP against the Greedy-DTN and GeOpps protocols we used our Generic One-Copy DTN Model (GOD) model, presented in section 4.2. To find the best one-hop communication strategy we first carried out a study to evaluate the impact of the use of i) one-hop ACK messages, to confirm that another node will carry the forwarded fragment, or ii) whether the implemented protocol uses broadcast or unicast communication. The results obtained show that MSDP outperforms both Greedy and GeOpps protocols in terms of both delivery ratio and introduced overhead. On the contrary, when the node density of the network increases, the Greedy-DTN protocol delivers more messages during an initial short-time period. Nevertheless, our MSDP is able to deliver more messages on the long term. We will analyze the nature of this behavior and use this analysis to improve MSDP in future works.

Chapter 8

VACaMobil Evaluation

The contents of this chapter have been partially published in:

- M. Báguena, S. M. Tornell, Á. Torres, C. T. Calafate, J.-C. Cano, and P. Manzoni. "VACaMobil: VANET Car Mobility Manager for OMNeT++". In: *IEEE International Conference on Communications 2013: IEEE ICC'13 - 3rd IEEE International Workshop on Smart Communication Protocols and Algorithms (SCPA 2013) (ICC'13 - IEEE ICC'13 - Workshop SCPA)*. Budapest, Hungary, June 2013, pp. 1–5.
- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Assessing the Effectiveness of DTN Techniques Under Realistic Urban Environments". In: *38th Annual IEEE Conference on Local Computer Networks (LCN 2013)*. Sydney, Australia, Oct. 2013. Core A Conference.

In section 4.1 we presented our mobility manager for the INET framework called VACaMobil [6]. VACaMobil provides the researcher a way to define the number of nodes in a simulation while the mobility of the nodes is driven by the road traffic simulator SUMO [9]. VACaMobil also automates the generation of independent mobility patterns for replicating simulations, so reducing the number of steps required to perform independent repetitions of a experiment.

To prove the improvements offered by VACaMobil and illustrate its use we compare the performance of VACaMobil with the tools originally included in SUMO, *i.e. duaRouter and duaIterate.py*, which were introduced in subsection 2.6.4. We will compare them in terms of vehicle distribution along the map area, the evolution of the number of vehicles along the simulation time, the impact of mobility in network protocol performance, and the number of steps required to independently replicate a simulation experiment several times.

The sections of this chapter are organized as follows: first, we briefly summarize the characteristics of duaRouter and duaIterate.py, which are already included in

SUMO. Section 8.2 compares how to configure vehicle mobility using VACaMobil or using duaRouter or duaIterate.py. In Section 8.3 we introduce the maps we will use later in Sections 8.4, 8.5 and 8.6 to evaluate the vehicles distribution over the road map, the evolution of vehicles density along the simulation time, and the impact of mobility in network protocols evaluation respectively. Finally, Section 8.7 summarizes the results presented in this chapter.

8.1 Compared tools

SUMO allows defining traffic demand in two different ways: trips and flows. The former defines only a vehicle, its origin and its destination, while the latter defines a set of vehicles which execute the same trip. SUMO currently provides several tools to generate traffic demand:

duaRouter: A Dijkstra router. Given a file with trips and flows, this tool generates the actual traffic demand, expressed in vehicles with an assigned route. Routes are calculated using the Dijkstra algorithm, and every unconnected trip is discarded.

duaIterate.py: This Python script will produce a set of optimal routes from a trip file, *i.e.* all the nodes will follow that route which minimizes the total trip-time for all nodes. This tool repeats a routing-simulation loop until optimal routes are found.

8.2 Work-flow Comparison

In this section we describe the typical steps required to obtain statistically significant simulation results when using VACaMobil, and compare them with *duaRouter* and *duaIterate.py*. This section does not aim to be a tutorial or a guide, but instead to illustrate how VACaMobil makes it easier to avoid biased results due to mobility effects. Here, we detail the process to obtain N non-correlated simulation repetitions of K different experiments with a different number of nodes in each experiment. Generally, simulating vehicles mobility involves these steps:

1. Create the road map.
2. Generate the road traffic demand.
3. Configure the network simulator.

In next subsections, we highlight the advantages of VACaMobil for each step.

8.2.1 Create the Road Map

The first step to simulate our network is to obtain or generate our road map. It can be downloaded from OpenStreetMap [101] and then converted to SUMO format using the tools included with SUMO (*netconvert*). This process is the one we will use in the next section to generate the urban scenarios. The road network can also be synthetically generated, as we will do for the Manhattan Synthetic scenario. This step is always required in order to simulate a vehicular network using SUMO to manage nodes' mobility. We assume that the same map is used for our N non-correlated repetitions. Therefore, no matter the tool used to generate the vehicles mobility, the road map is required to be generated only once.

8.2.2 Road Traffic Demand Generation

Once we have generated or downloaded our road map, we need to generate the traffic demand. How to generate it varies between *duaRouter*, *duaIterate.py*, and VACaMobil.

Duarouter

To generate random traffic demand for *duaRouter* we need to invoke the *randomTrips.sh* script presented in section 2.6.4, and call *duaRouter* to compute the shortest routes. This process will produce a file containing a set of vehicles that will be simulated by SUMO. If our objective is to repeat N non-correlated simulations we need to repeat these steps N times, once for every simulation, and generate N configuration files that will be made available to SUMO through the OMNeT++ configuration file. To vary the number of nodes in our simulation and simulate K different scenarios, we need to repeat these steps $K * N$ times in order to generate N different SUMO configuration files for the K possible scenarios.

DuaIterate.py

As referred above *duaIterate.py* refines the traffic demand obtained with *duaRouter* to minimize the time cost of the routes. As a consequence, it adds a new step to the *duaRouter* procedure. This new step must also be repeated for every simulation. As well as *duaRouter, duaIterate.py* generates a file containing a set of vehicles that will be simulated by SUMO.

VACaMobil

In VACaMobil, the traffic demand consists of a set of routes that can be generated immediately after obtaining the network map. VACaMobil includes a script, called *randomRoutes.sh,* that generates a large-enough set of random routes to be used.

8.2.3 OMNeT++ Configuration

All the tools evaluated in this chapter are designed to be used with the INET framework for the Omnet++ simulator. The simulator is configured through a plain text configuration file. Omnet++ simulation parameters can be defined as an array, through which the simulator will iterate and perform as many repetitions as values the array has. Using this feature, the simulation mobility must be specified as follows:

Duarouter and duaIterate.py

In order to configure OMNeT++ to use the mobility generated by *duaRouter* or *duaIterate.py,* we need to specify the path for each and every one of the different $K * N$ SUMO configuration files generated in the previous step.

VACaMobil

Listing 8.1: Omnet++ mobility configuration using VACaMobil.

```
repetitions = 5
*.manager.launchConfig = xmldoc("MySumoConfig.launch.xml")
*.manager.meanNumberOfCars = ${10,20,30,40}
```

Listing 8.2: Omnet++ mobility configuration using duaRouter or duaIterate.py.

```
repetitions = 1
*.manager.launchConfig = xmldoc({"MySumoConfig10_0.launch.xml",\
"MySumoConfig10_1.launch.xml", "MySumoConfig10_2.launch.xml",\
"MySumoConfig10_3.launch.xml", "MySumoConfig10_4.launch.xml",\
"MySumoConfig20_0.launch.xml", "MySumoConfig20_1.launch.xml",\
"MySumoConfig20_2.launch.xml", "MySumoConfig20_3.launch.xml",\
"MySumoConfig20_4.launch.xml", "MySumoConfig30_0.launch.xml",\
"MySumoConfig30_1.launch.xml", "MySumoConfig30_2.launch.xml",\
"MySumoConfig30_3.launch.xml", "MySumoConfig30_4.launch.xml",\
"MySumoConfig40_0.launch.xml", "MySumoConfig40_1.launch.xml",\
"MySumoConfig40_2.launch.xml", "MySumoConfig40_3.launch.xml",\
"MySumoConfig40_4.launch.xml"})
```

In order to simulate the desired scenario with VACaMobil we only need to specify the number of repetitions (N) and an array containing the (K) different number of nodes that should be generated into the network. OMNeT++ and

Table 8.1: Steps required to define nodes mobility.

Method	#Files	#Commands	#Rep Steps
duaRouter	4·N·K	2·N·K	2·N
duaiterate	4·N·K	3·N·K	3·N
VACaMobil	4	4	0

VACaMobil will manage the whole set of simulations, storing their results in different files, ready to be processed. Listings 8.1 and 8.2 illustrate the differences between configuring Omnet++ when using VACaMobil and configuring it when using duaRouter or duaIterate.py. The configuration using VACaMobil is simpler and clearer than using any of the tools included in SUMO.

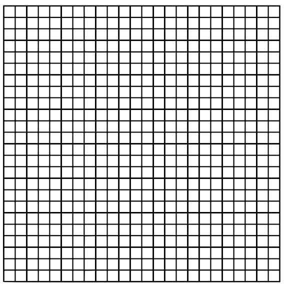
8.2.4 Total Number of Steps

In this subsection we detail the number of commands required to repeat N times the simulation of K scenarios which differ in the number of nodes introduced into the network. Table 8.1 shows the number of files we need to create, as well as the number of steps required to configure our simulations. The "#Rep Steps" column contains the number of steps we need to repeat in case we decide to add a new scenario $K+1$. The number of steps and files required when using VACaMobil is constant, while, when using duaRouter or duaIterate, the complexity of mobility definition increases when the number of scenarios or repetitions is increased. It is clear that VACaMobil simplifies the complexity of the mobility definition, which makes it less prone to errors. In the next sections we evaluate the *quality* of the mobility scenarios generated by VACaMobil by comparing them with duaRoute and duaIterate.py.

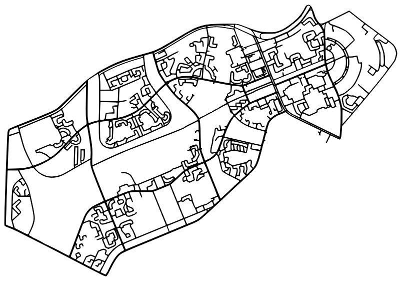
8.3 Map Scenarios

One of the most important issues when simulating vehicles mobility in VNs simulation is the city map. To obtain general results we compare VACaMobil in four different map scenarios. Those maps cover the most common scenarios in urban mobility. The maps we choose were:

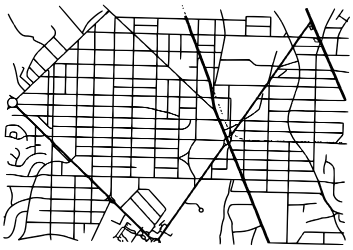
- **Synthetic Manhattan scenario:** We created a road map consisting of a 25 x 25 grid with segments of 200 meters (Figure 8.1a).
- **Suburban real map scenario:** We extracted a suburban road map from the OpenStreetMap database. The selected scenario is a $\tilde{1}2$ km^2 area from the city of Moscow characterized by long road segments and a low road density (Figure 8.1b).



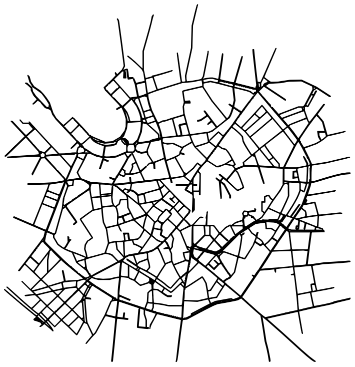
(a) Synthetic Manhattan map.



(b) Suburban real map (Moscow).



(c) Urban grid real map (Washington).



(d) Urban downtown real map (Milano).

Figure 8.1: Maps used for evaluation.

- **Urban grid real map scenario:** We extracted an urban road map from the OpenStreetMap database. This scenario is a 6 km^2 area extracted from the city of Washington DC. It is characterized by long road segments and a high road density (Figure 8.1c).

- **Urban downtown real map scenario:** We extracted an urban road map from the OpenStreetMap database. This scenario is a 7 km^2 downtown area selected from the city of Milano. It is characterized by short road segments and a high road density (Figure 8.1d).

In all the scenarios, the set of random routes provided to VACaMobil is extracted from the traffic demand generated by *duaIterate.py*. In the following subsection, we compare the vehicle density and its evolution along the simulation time for the different tools and scenarios.

8.4 Vehicle Map Distribution Study

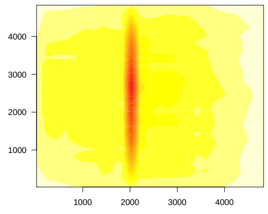
We first evaluate one of the most important issues in vehicular mobility: how vehicles are distributed over the simulated road map. To do so, we have created several heat maps that collect information about the total number of vehicles that reported a certain location during the simulation. The parts of the map colored in red experienced the maximum density values, while parts colored in white experienced the minimum density values. All the heat maps that belong to the same scenario share the same scale.

Figure 8.2 shows the vehicle distribution in the Synthetic Manhattan scenario. Due to its lack of randomness, *duaRouter* is unable to select different routes for vehicles when there are several streets with the same travel-time. This prevents the simulator from properly distributing vehicles, and so all of them are routed through the same street. When using the *duaIterate.py* script, a better distribution of the vehicles is achieved since many simulations are sequentially executed to optimize vehicle routes. Since the VACaMobil's random routes set is obtained from *duaIterate.py*, in this case it achieves a similar nodes distribution.

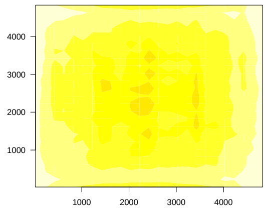
Figure 8.3 shows the vehicle distribution in the suburban real map scenario (Moscow). In this case, *duaRouter* concentrates all the traffic in the inner roads. Therefore, the typical ring roads around the map are underused, which is not a common driver behavior. When using the *duaIterate.py* or VACaMobil, vehicles are distributed among the two bigger streets, i.e., the inner and the outer biggest avenues.

Figure 8.4 shows the vehicle distribution in the urban grid scenario (Washington). Since this map includes big roads, few congested areas are present with *duaRouter*. However, a smarter driver behavior can also be seen when using *duaIterate.py* or VACaMobil. *DuaIterate.py* and VACaMobil do not present congested areas because vehicles take advantage of alternative paths in this map to reduce their travel times.

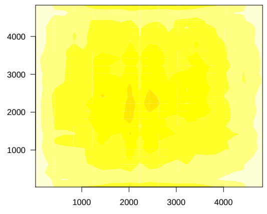
Finally, Figure 8.5 shows the vehicle distribution in the urban downtown scenario (Milan). In this case, *duaRouter* is also unable to spread the vehicles properly. Since some roads are faster than others, all the vehicles are routed through them, even when these streets are congested. Therefore, an undesired traffic congestion is created in the fastest inner roads. However, this is an unrealistic scenario because drivers tend to avoid traffic jams whenever possible. When using either *duaIterate.py* or VACaMobil, vehicles are routed through alternative streets, avoiding traffic jams. This strategy has a higher degree of similitude compared to real road traffic, since drivers prefer faster roads but often change their route to avoid traffic jams.



(a) duaRouter.

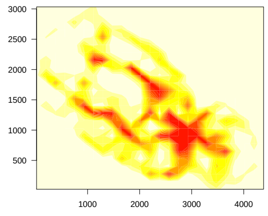


(b) duaIterate.py.

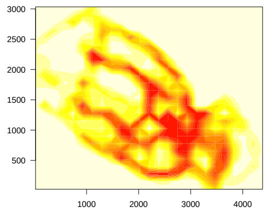


(c) VACaMobil.

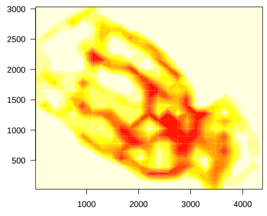
Figure 8.2: Heat map for the Synthetic Manhattan scenario (*Synthetic*).



(a) duaRouter.

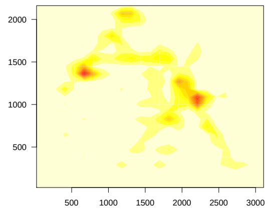


(b) duaIterate.py.

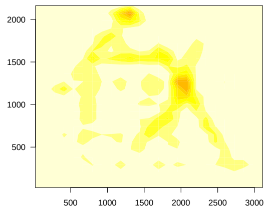


(c) VACaMobil.

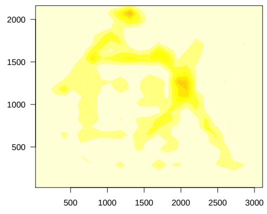
Figure 8.3: Heat map for the suburban scenario (*Moscow*).



(a) duaRouter.

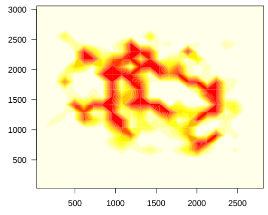


(b) duaIterate.py.

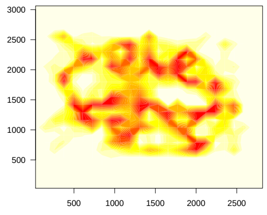


(c) VACaMobil.

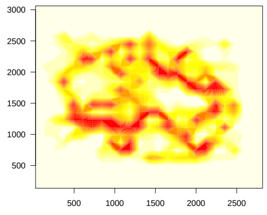
Figure 8.4: Heat map for the urban grid scenario (*Washington*).



(a) duaRouter.



(b) duaIterate.py.



(c) VACaMobil.

Figure 8.5: Heat map for the urban downtown scenario (*Milano*).

8.5 Vehicle Density Study

To make simulation results comparable, a similar vehicle density is desirable in every simulated road maps. DuaRouter and duaIterate.py cannot correctly handle this requirement. In order to compare the behavior of the three methods previously presented, we have measured the average number of vehicles, its standard deviation, and its evolution along simulation time.

Since neither *duaRouter* or *duaIterate.py* allow defining the number of cars in the simulation, we configured VACaMobil according to values in Table 8.2 to mimic the average values obtained previously when using *duaRouter* and *duaIterate*. Table 8.3 shows the differences in terms of number of vehicles for the four different scenarios for each different traffic generation tool. In the simplest scenario (Synthetic Manhattan), the three methods achieved a stable value for the average vehicle density with a low standard deviation. However, it is worth noting that neither duaRouter nor duaIterate allow to *a priori* configure the value of this parameter. On the contrary, VACaMobil is not only able to quickly populate the network with the desired number of vehicles, but it also allows defining a maximum and a minimum number of vehicles, which will bound the standard deviation value. In complex maps like the urban downtown scenario (Milano), VACaMobil is the only tool able to maintain the number of vehicles within the predefined bounds.

To better understand the aforementioned values, Figures 8.6 to 8.9 show the evolution on the number of vehicles for each tool in the different scenarios along time. Since *duaRouter* and *duaIterate.py* are able to add only one vehicle per second, the user cannot predict when vehicles will arrive to their destination and disappear from the network. Therefore, for the Synthetic Manhattan scenario, the number of vehicles when the simulation reaches the steady-state is not known *a priori*, turning protocol analysis based on the number of vehicles into a mere

Table 8.2: VACaMobil configuration.

	Vehicle number	carVariability	Std.dev.
Synthetic Manhattan	320	20	6.33
Suburban scenario	225	25	8.33
Urban grid scenario	175	20	6.33
Urban downtown scenario	370	25	8.33

Table 8.3: Vehicle statistics summary.

	Synthetic Manhattan		Suburban scenario		Urban grid scenario		Urban downtown scenario	
	mean	std. dev.	mean	std. dev.	mean	std. dev.	mean	std. dev.
duaRouter	313.767	58.8271	319.165	79.4342	214.711	39.0532	880.546	465.716
duaIterate.py	304.487	55.5174	219.610	34.2232	183.009	36.6651	393.717	96.414
VACaMobil	319.349	6.14267	223.718	8.32201	174.615	6.79294	369.691	7.84640

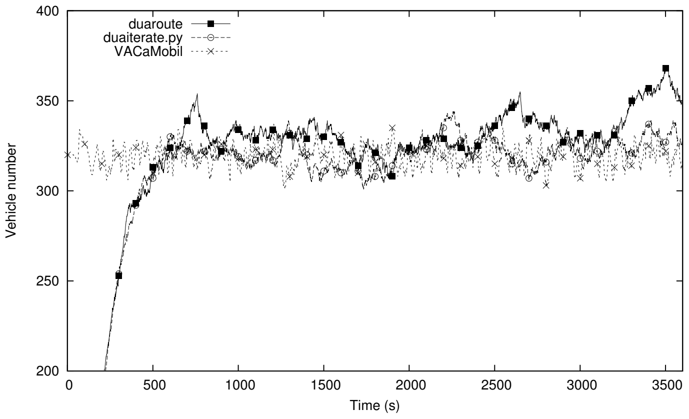


Figure 8.6: Evolution of the number of vehicles along simulation time in the Synthetic Manhattan scenario.

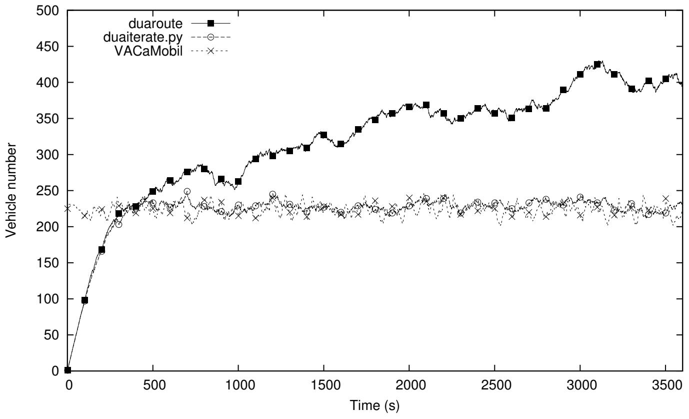


Figure 8.7: Evolution of the number of vehicles along simulation time for the suburban scenario (Moscow).

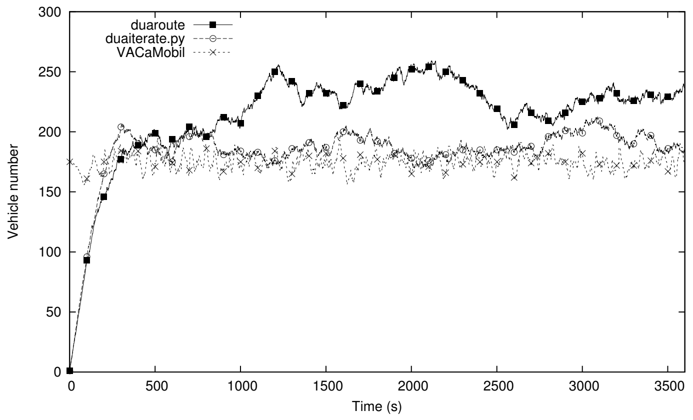


Figure 8.8: Evolution of the number of vehicles along simulation time for the urban grid scenario (Washington).

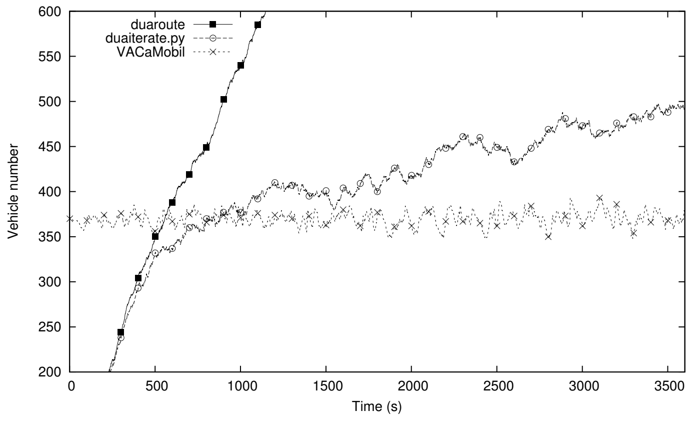


Figure 8.9: Evolution of the number of vehicles along simulation time for the urban downtown scenario (Milan).

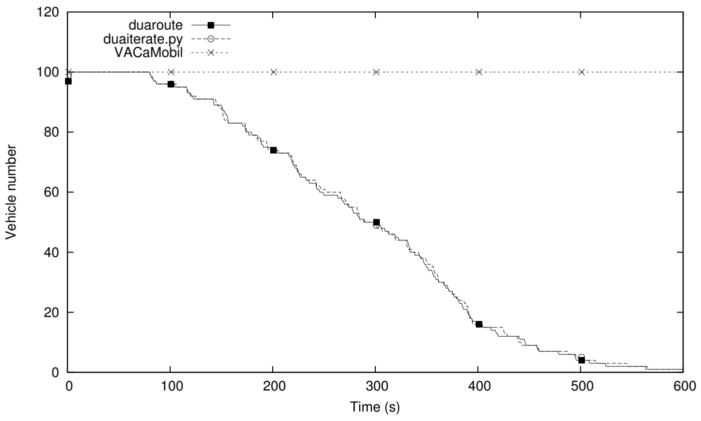


Figure 8.10: Number of nodes in the network after introducing 100 cars at the beginning of the simulation.

act of faith. Moreover, in maps where traffic jams are common, as in the urban downtown scenario, it takes more time for vehicles to reach their final destination and leave the network, which causes a constantly increasing number of vehicles in the network when not using VACaMobil. Comparing the configuration in Table 8.2 and the results in Table 8.3, we can conclude that both the target number of vehicles and the expected standard deviation are achieved only with VACaMobil.

8.6 Measuring Mobility Impact on Network Protocols

In this section we evaluate how nodes' mobility can impact network protocols performance. To demonstrate its effects, we have simulated the same network topology using *duaRouter, duaIterate.py,* and VACaMobil respectively to manage node's mobility.

8.6.1 Mobility specification

To perform this test, and for sake of simplicity, we chose the synthetic Manhattan scenario presented before. In contrast to previous evaluations, we have manually modified the configuration files generated by *duaRouter* and *duaIterate.py* in order to simultaneously introduce 100 vehicles with random routes at the beginning of the simulation. VACaMobil was configured to maintain 100 nodes in the network during the whole simulation. Figure 8.10 shows the number of actives nodes. As can be appreciated, when using both, *duaRouter* and *duaIterate.py*, the number of cars in the network decreases, whereas when using VACaMobil it is kept at its desired value along the entire simulation.

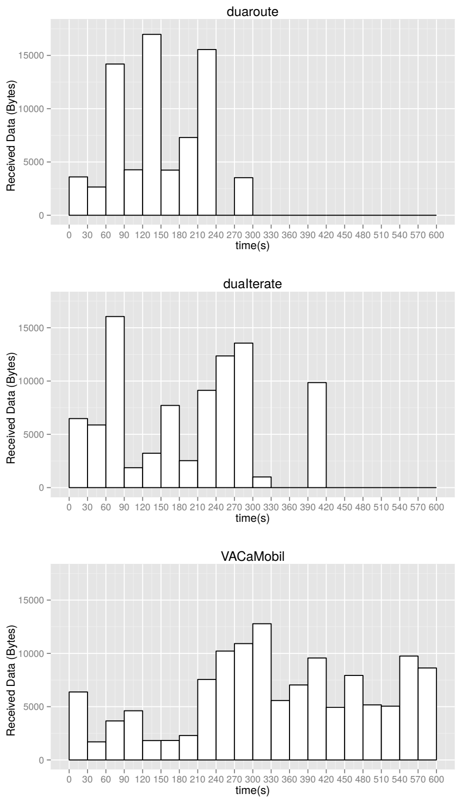


Figure 8.11: Number of received bytes along time.

8.6.2 Network Topology

To measure the impact of mobility in network protocols we measure the connectivity between two RSU in terms of bytes interchanged between them. In order to maximize the impact of mobility, we placed both RSUs in the most-traversed road of the scenario. The distance between them was 900 m and the communication range was approximately 200 m. Each RSU sends a 512 Bytes UDP unicast packet to the other RSU every second. The vehicles were configured to use the DYMO routing protocol [106].

8.6.3 Network Impact Results

To assess the impact of mobility we compared the connectivity between the two RSU along the simulation time when using different mobility alternatives. To do so, we have accounted the total amount of bytes interchanged between the two RSUs for each 30-second slot. Figure 8.11 shows the results we obtained after averaging the values from 10 different simulations. Notice that, when using *duaRouter* or *duaIterate.py*, the connectivity of the network experiences an initial period where it is higher than when using VACaMobil. However, as long as the number of vehicles decreases according to Figure 8.10, the network gradually loses its connectivity. On the contrary, when using VACaMobil, the connectivity of the network fluctuates throughout the simulation with peaks and valleys due to the random nature of cars mobility. This experiment highlights one of the effects of mobility on network protocols. When evaluating more complicated scenarios, such as diffusion protocols, delay-tolerant networks, or cooperative sensors, new problems related with the lifetime of the nodes arise [117] .

8.7 Summary

The results presented in this chapter illustrate the usefulness of VACaMobil[1]. To the best of our knowledge, VACaMobil is the first tool able to simulate SUMO [9] driven nodes in a vehicular network while ensuring the stability of certain user-defined parameters, such as the average, maximum, and minimum number of vehicles. In particular, we added critical features to previously existing tools, such as ensuring a constant number of vehicles during the entire simulation time and disseminating vehicles throughout the whole road map. Comparing the configuration in Table 8.2 and the results in Table 8.3, we can conclude that both the target number of vehicles and the expected standard deviation are achieved only with VACaMobil. In contrast to other existing tools, VACaMobil is able to keep the mean number of vehicles and the standard deviation value within user-defined bounds.

When we released VACaMobil, it was the only tool that allows studying a vehicular network in a steady-state situation without losing the realistic vehicle behavior provided by SUMO[2].

[1]VACaMobil is freely available at `http://www.grc.upv.es/software`.

[2]By the time I am writing this thesis, a new version of Veins has added support for a static number of nodes in the simulation. However, VACaMobil still is a more advanced tool.

Chapter 9

GRCBox Uses Cases and Evaluation

The contents of this chapter have been partially published in:

- S. M. Tornell, S. Patra, C. T. Calafate, J.-C. Cano, and P. Manzoni. "GRCBox: Extending Smartphone Connectivity in Vehicular Networks". In: *International Journal of Distributed Sensor Networks* 2015.Article ID 478064 (2015), p. 13. DOI: `10.1155/2015/478064`. Impact Factor:0.665 (Q4).
- S. M. Tornell, T. Kärkkäinen, J. Ott, C. Calafate, J.-C. Cano, and P. Manzoni. "Simplifying The In-vehicle Connectivity for ITS Applications". In: *MOBIQUITOUS 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2015. DOI: `10.4108/eai.22-7-2015.2260058`. Core A Conference.
- S. M. Tornell, S. Patra, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A novel On-board Unit to Accelerate the Penetration of ITS Services". In: *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC 2016)*. Las Vegas, USA, Jan. 2016, pp. 474–479. Core B Conference.

In Chapter 6, we presented our GRCBox architecture. The GRCBox architecture enables in-vehicle applications running in user devices such as smartphones or tablets to extend its infrastructure connectivity by adding infrastructure-less in-vehicle to in-vehicle communication. In this chapter we illustrate with examples how to use the GRCBox. Besides, we include a performance evaluation of using our GRCBox architecture.

In section 9.1, we start this chapter with the most simple case connecting to a server on the Internet using an specific interface. We use this simple case to measure the time required to configure the GCM and the delay it introduces. Then, in Section 9.2, we describe the operations required to establish ad-hoc vehicular communication between two user devices located in different vehicles, using both UDP and TCP. Section 9.3, presents how the Scampi middleware, a DTN platform for smartphones, can be adapted to support the GRCBox Architecture. In Section 9.4, we describe how to use the GRCBox to perform an inter vehicle VoIP call between 2 different user devices. Finally, Section 9.6, concludes and summarizes this chapter.

9.1 Connecting to the Internet

When connecting to the Internet using a specific interface, a GRCBox-aware application must cover several steps before being able to communicate through the desired interface with the external network. As depicted in Figure 9.1, the operations that the application must perform are:

- Check the availability of the GCM.
- Check the available interfaces on the GCM.
- Register a new application.
- Register a new rule with the desired output interface.
- Now the application can download the file from the Internet.
- Remove the previously defined rule.
- Remove itself from the GCM applications' database.

In this section we evaluate the overhead introduced by the GRCBox architecture by measuring the average time required to perform each of these operations, and the delay introduced by the GRCBox architecture itself.

9.1.1 Test Configuration

To measure the times associated to the different steps, we have configured a GCM with three wireless interfaces: a TP-Link TL-WN727N usb adapter (chipset rt5370 from Ralink); a Linksys WUSB600N usb adapter (chipset rt2870 from Ralink); and a generic WiFi usb adapter with an rt5370 chipset from Ralink. The first interface is configured as an access point, while the other two are connected to an infrastructure access point connected to the Internet using the Universitat Politècnica de València infrastructure. We have performed two different experiments, the first one using a laptop as a user device, and the second one using a smartphone. The user device will connect to a host located in the university network through the GCM to download a file. All the devices (including the Internet-connected access point) are placed in the same room. Figure 9.2 shows how the different devices

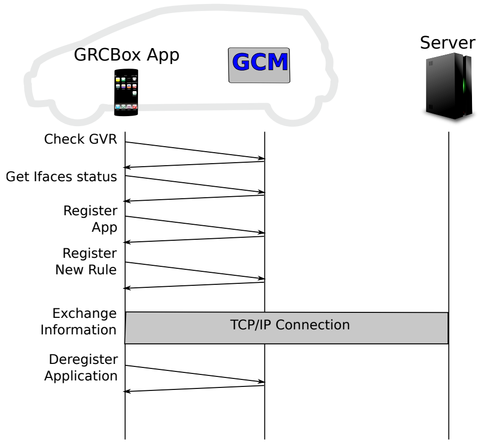


Figure 9.1: Example of a client GRCBox application connecting to a server.

are connected for this test, while Figure 9.3 shows the real devices we have used to run the test.

The experiments exposed in this section were performed using a preliminary version of the GRCBox, since then, the GRCBox has been improved. We specially improved the *Interface Monitoring* module, which we migrated from a command line based to the current D-Bus based module which is much faster.

9.1.2 Time Required to Configure the GCM

To get conclusive results, we have repeated each experiment 100 times. To check if the computation power of the user device affects the average delay added by the GRCBox architecture, we performed the same set of experiments using both the notebook and the smartphone.

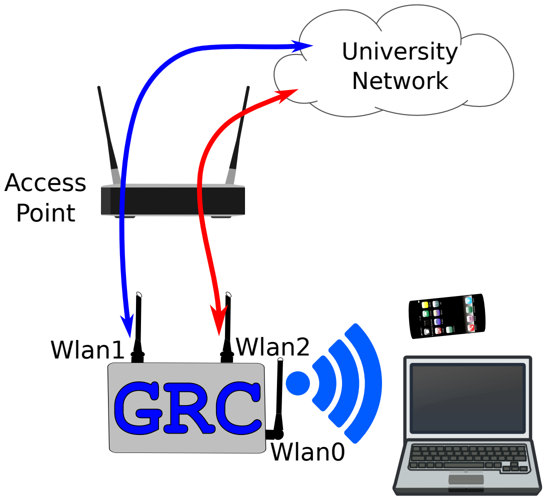


Figure 9.2: Access to the Internet test configuration.

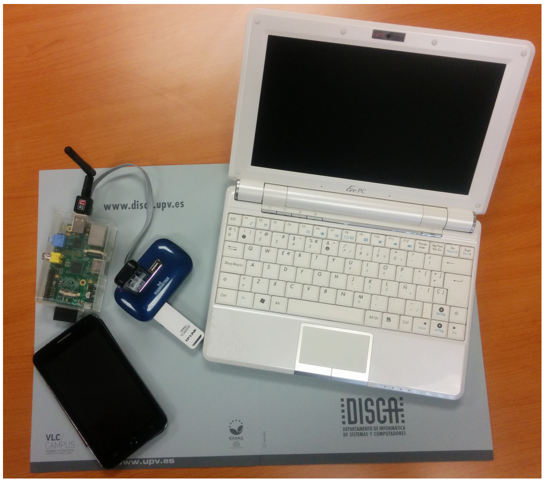


Figure 9.3: A picture of the devices used for the test.

Table 9.1: Time required for the different tasks involved when downloading a 5 MB file.

	GRCBox Notebook		GRCBox Smartphone		Non-GRCBox Notebook		Non-GRCBox Smartphone	
	Average	Conf. Int. 95%	Average	Conf. Int. 95%	Average	Conf. Int. 95%	Average	Conf. Int. 95%
Check	1.46s	±0.24	1.55s	±0.22	-	-	-	-
Ifaces	1.12s	±0.13	1.25s	±0.03	-	-	-	-
Reg. App.	1.08s	±0.05	0.96s	±0.03	-	-	-	-
Reg. Rule	1.21s	±0.09	1.29s	±0.03	-	-	-	-
Download	6.28s	±0.39	6.34s	±0.3	6.82	±0.89	7	±0.4
Rm Rule	0.19s	±0.02	0.21s	±0.02	-	-	-	-
Rm App	1.11s	±0.02	1.22s	±0.02	-	-	-	-
Total	12.5s	±0.54	12.83s	±0.41	6.82s	-	7s	-

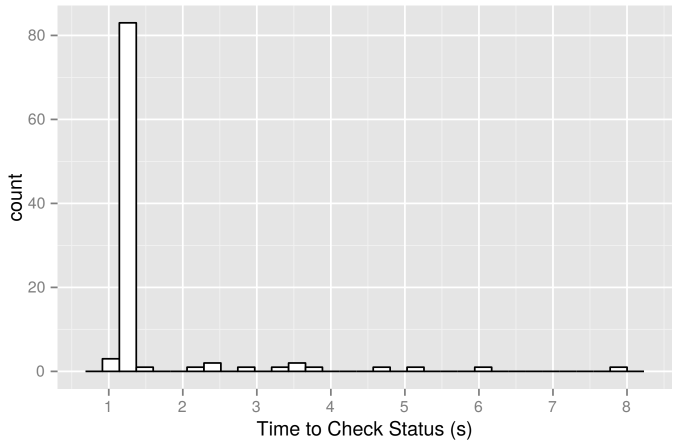


Figure 9.4: Time to check status of the GRCBox core service histogram when using a smartphone as user device.

As a reference, we have also downloaded the same files through the GCM without using the GRCBox features. Table 9.1 summarizes the obtained results, all the values represent time in seconds. The average time required for every step ranges from 0.2s to 2.4s, whereas the total required extra-time is around 6 seconds. These values are tolerable when the user wants to perform a long lasting action, such as browsing or a VoIP call. According to the values we obtained, an application should be able to check if the GRCBox core service is available, get the list of interfaces, register itself in the GRCBox core service, and register a new rule to route the expected traffic through the desired output interface, transparently to the user, while the user starts and configures the application (write a website address or dial a phone number).

Concerning *non-interactive autonomous applications*, such as a DTN services, or ad-hoc warning notification systems, which usually run continuously for hours or even days, they only need to communicate with the GRCBox at the moment they start running. Therefore, the impact of the delay introduced by the GRCBox architecture is insignificant.

By examining the confidence interval of the first step (check the status of the server), we realized that its variability was very high. To find an explanation to this fact, we inspected the individual values, whose histogram is represented in Figure 9.4, and realized that the server took up to 8s to resolve the request. By revising our code, we detected a problem in the *Interface Monitoring* module that blocks the processing of requests involving interface information during the interface updating process, which takes about 10s. Switching from a command-

Table 9.2: Round Trip Times measured using *Ping*.

	www.upv.es		google.com		www.yahoo.com	
	Average	Std.Dev.	Average	Std.Dev.	Average	Std.Dev.
GRCBox	16.96	23.97	40.222	54.584	108.899	89.240
Directly Conn.	14.210	35.9	24.402	34.101	113.774	111.254

line based monitoring to a D-Bus messages monitoring have solved this problem. The D-Bus message bus allow the GCM to subscribe to signals that are generated when interfaces' status is changed. Therefore, polling the system to find changes is no longer required, interfaces' info is updated as soon as changes occur.

9.1.3 Delay Introduced by the GRCBox

To measure the delay introduced by the GRCBox architecture, we have used the well-known *ping* tool to measure the Round Trip Time (RTT) between the user device and a server on the Internet in two different conditions: connected through the GRCBox, and connected directly to the access point in our lab. To get a wide set of results we *pinged* 100 times 3 different hosts under different domains. Table 9.2 contains the results we obtained. The first important thing we notice is the high value of the *standard deviation*, which means that the network, in all the cases, was very unstable. Given this condition, it is hard, or even impossible, to conclude that there is a difference between the RTT experienced when using GRCBox, and when connected directly to the access point acting as Internet gateway. In addition, and despite it is obvious that adding an extra hop to the path between the user device and the Internet server increases the RTT, the obtained data shows that this increment is negligible if we compare it to the effects of network instability.

9.2 Ad-hoc V2V Connectivity

This is a case of direct client-server ad-hoc communication between devices. It assumes that the *client* and *server* connectivity has been configured using some auto-configuration system like the one presented in [14]. Therefore, the *client* knows the public IP of the vehicle connected to the VANET where the device acting as a *server* is connected. For UDP flows and TCP connections, only one rule on the server side must be registered. That rule must specify the port where the *server* is listening for connections. Thus the GCM can forward connections attempts on the external interface to the node by performing Network Address Translation (NAT). As depicted in Figure 9.5, the steps performed by the *server* device for establishing such communication are the following:

1. The device must check the availability of the GCM.

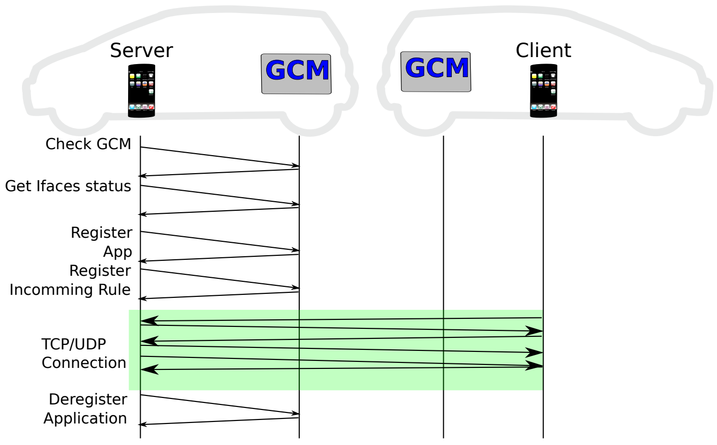


Figure 9.5: GRCBox interaction for client-server ad-hoc communication.

2. Connect to the GCM and register itself.

3. Check the status of the interfaces and select the VANET interface.

4. Register a new rule to forward incoming connections.

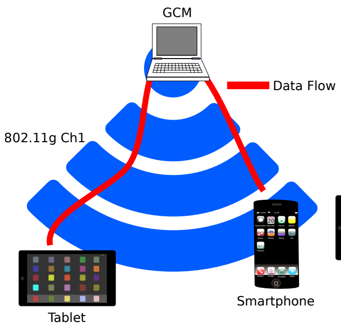
5. Wait for incoming connections from *clients* and process them.

6. When the server is stopped, the application should deregister itself from the GCM, which will also remove all its rules.

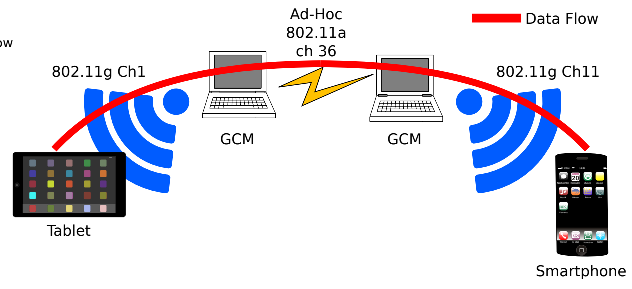
The *client* does not need to perform any interaction with its GCM, since the GCM will forward the connection through the VANET interface based on the destination IP.

9.2.1 Test Configuration

We have run experiments to evaluate the performance of the GRCBox Architecture in 2 different scenarios: In the first scenario, we used an Android Nexus 7 tablet and a BQ4.5E smartphone connected to the same GCM, which acted as a standard WiFi Access Point, since connections can be established without interaction with the GCM, this is the baseline scenario. In the second scenario we connected each device to different GCMs, which were then connected to the same ad-hoc network, in this case the connection must be established through the GCMs. Table 9.3 and Figure 9.6 summarize the configuration of both scenarios.



(a) Intra-GCM Scenario



(b) Inter-GCM Scenario

Figure 9.6: Scenarios used in our experiments.

Table 9.3: Devices Hardware Characteristics.

Element	Characteristics
Tablet	Google Nexus 7 (2012)
Smartphone	BQ Aquaris 4.5E
GCMs	Asus EeePc 1000h, Intel Atom N270
Ad-Hoc Network	802.11a, Frequency:5.18 GHz
WiFi 1	802.11g, Frequency:2.462 GHz
WiFi 2	802.11g, Frequency:2.412 GHz

When we first implemented the GCM using a Raspberry Pi model B[1], we found that the board presented power management and instability issues that prevent connecting more than one wireless interface. To overcome this limitation we temporarily used an Asus EeePC netbook with low computational power to run the GCM. The notebooks were running a Debian Linux distribution with an 802.11a USB wireless network interface configured in the Ad-Hoc mode.

The GRCBox management application was used to configure the required rules on the GCMs.

9.2.2 Performance Analysis

We have run two different experiments in the two different previously presented scenarios: the first experiment is an analysis of the maximum throughput. while the second experiment analyses the UDP RTT between client and server.

[1]A new Raspberry Pi model was released in Q1 2015, this model, called Raspberry Pi 2, solved the power related issues.

Maximum Throughput

To evaluate the impact of the GRCBox Architecture on the maximum throughput experienced by an ad-hoc client-server connection when using the GRCBox, we tested the network performance using the *iperf* [58] tool. We have collected measurements for both UDP and TCP protocols. Each experiment was repeated 60 times to discard random effects, and the role of the user devices was interchanged after half of the experiments to discard the effects associated to the device's performance. Results are shown in Figure 9.7a in a boxplot chart.

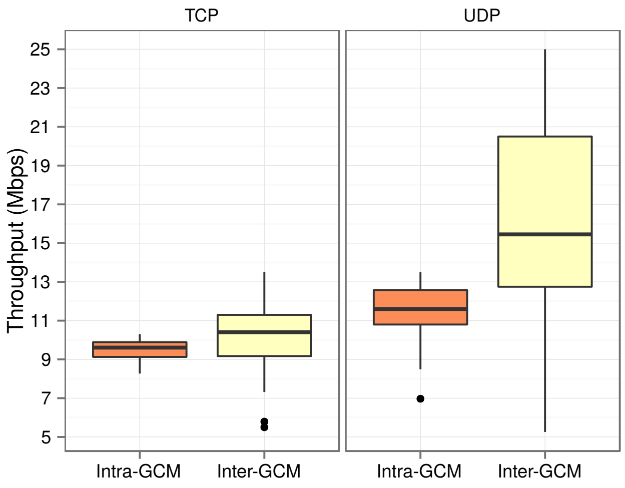
Notice that, no matter whether TCP or UDP is used, the maximum throughput achieved when using the GRCBox Architecture (Scenario 2) is slightly better than the one achieved when both devices are connected to the same WiFi Network (Scenario 1). The main cause behind this difference is the use of a different channel for each wireless network when using the GRCBox. This setup avoids collisions between nodes, when transmitting requests and responses. The high variability experienced in all the experiments is due to the presence of interference, which heavily affects throughput in wireless networks.

UDP Round Trip Time (RTT)

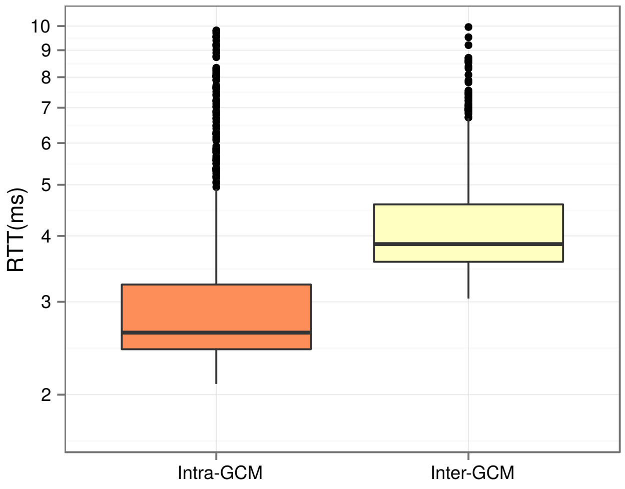
To test the delay introduced by the GRCBox Architecture when it is compared against an infrastructure network we have developed a small application that sends an UDP message to a server running on another device. The server will then send a new UDP message as a response, so the RTT can be measured at the first sender. We performed the test on both scenarios presented before, collecting more than 500 measurements per scenario. Figure 9.7b shows a boxplot that summarizes the results we obtained. We have used a logarithmic scale to be able to clearly represent infrequent values in both the low and high ranges. It can be observed that, on average, the RTT is about 2 ms higher when using GRCBox. This effect is due to the multi-hop nature of the communication: adding an extra hop between sender and receiver increases the RTT. During this experiment we discovered that the main source of delay in Android devices is the WiFi interface power management performed by the Android operating system, which in some cases increased the RTT by up to 508 ms. If we compare the delay introduced by the GRCBox against the delay introduce by the OS, we can conclude that the GRCBox impact on the RTT is negligible.

9.3 Vehicular DTN Scenario: Scampi Neighbor Discovery and Data Transmission

An example of new applications based on opportunistic peer-to-peer communications is the Scampi project [63]. Its authors developed a framework to provide opportunistic communication for smartphones. They proposed to deploy au-



(a) Throughput obtained for UDP and TCP tests.



(b) UDP RTT results.

Figure 9.7: Direct VANET communication results.

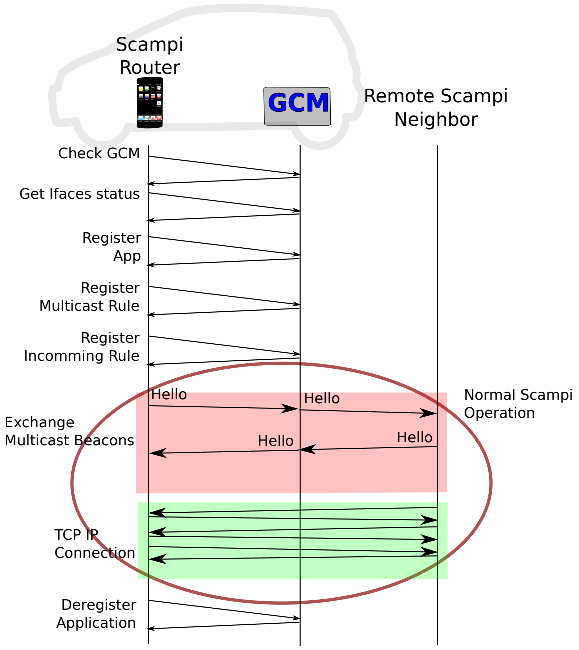


Figure 9.8: Integration between Scampi and GRCBox.

tonomous routers (called LibeRouters) that, by creating WiFi connectivity islands, provide a network for opportunistic contacts between smartphones. By using a DTN [66] architecture and taking advantage of node mobility, Scampi distributes messages to nodes connected to other Scampi routers. Since the Scampi platform requires nodes to be associated to the same router in order to exchange information, it is not suitable for VNs, where nodes move quickly and contacts based on the infrastructure last only for a short time period. Our GRCBox can complement the Scampi platform by increasing the smartphones' connectivity beyond the local Scampi router, thereby increasing the number and duration of opportunistic contacts.

To illustrate the use of a more elaborated VANET application we describe the modifications required to adapt the Scampi router [63] to the GRCBox Architecture. Although Scampi was not designed for VANETs, it is a good example of a V2V application. Scampi neighbors are discovered by multicasting beacon messages periodically. Once a new neighbor has been discovered, Scampi establishes a TCP/IP connection to exchange information. As shown in Figure 9.8, the modifications required on the Scampi router to adapt it to the GRCBox are the following:

1. Check the GCM availability.
2. Get the list of available interfaces and their status.
3. Create a new multicast proxy associated to the Ad-Hoc interface to forward multicast beacons to the external networks.
4. Register a rule for incoming connections from the neighbors reached by the multicast messages. Now the application can receive connections from remote neighbors.
5. Perform standard Scampi activities.
6. Once the application is closed, it must remove itself from the GCM's applications database, which will remove both rules.

In this case, both rules are long-lasting rules that must be active as long as the application is running.

9.3.1 Performance Analysis

As in the previous case, we have run two different experiments to evaluate the performance of the GRCBox Architecture combined with Scampi: the first experiment measures the maximum bundle throughput, and the second experiment measures the Scampi RTT of bundles between two devices. To avoid modifying the Scampi's source code we have created a *multicast plugin* that can be activated through the management application to forward the multicast beacon messages sent by the Scampi router from the inner network to the desired external network. Both experiments were run once the Scampi nodes had discovered each other and the Scampi topology was stable. We have used the same scenarios we presented in the previous section (See Figure 9.6).

Scampi Throughput

To measure the performance of the Scampi platform when combined with the GRCBox Architecture we have developed a *client* application that, by using the Scampi API, generates a burst of bundles that are passed by to the Scampi router in order to be distributed to scampi neighbors. Once the *server* application receives all the bundles, it confirms their reception by creating a new bundle. The *server* application measures the time and the amount of data received to calculate the throughput in terms of Mbps. When running the scampi middleware under heavy load it presents some issues that complicate computing the time required to exchange a certain number of bundles. Therefore, we needed to signal the beginning of each experiment through a notification bundle, that has to be confirmed by the *server*. In the same way, the reception of the last bundle of the burst

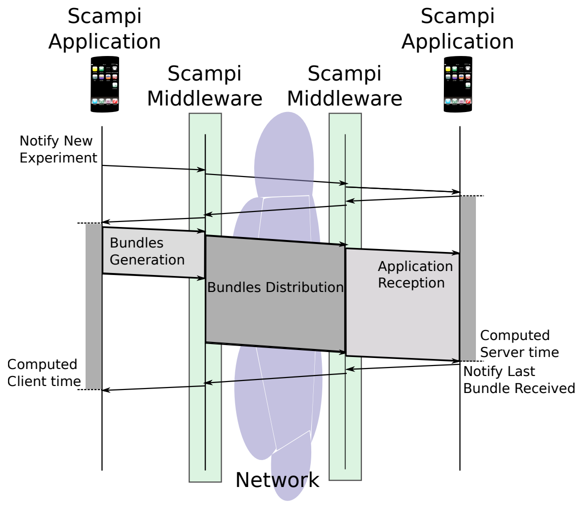


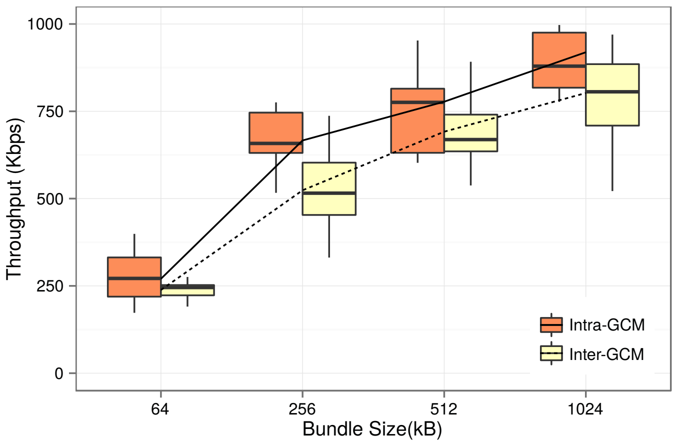
Figure 9.9: Bundles sent to measure the Scampi throughput.

has to be notified to the *client*, so it can compute the total required time. This behavior is summarized in Figure 9.9. It is important to notice that, when the scampi middleware receives a new bundle, it is not immediately delivered to the user application, and thereby the Scampi RTT does not depend on the network resources alone. We repeated the experiment varying the size of the bundles. Each experiment was repeated 10 times. Figure 9.10a shows the obtained results.

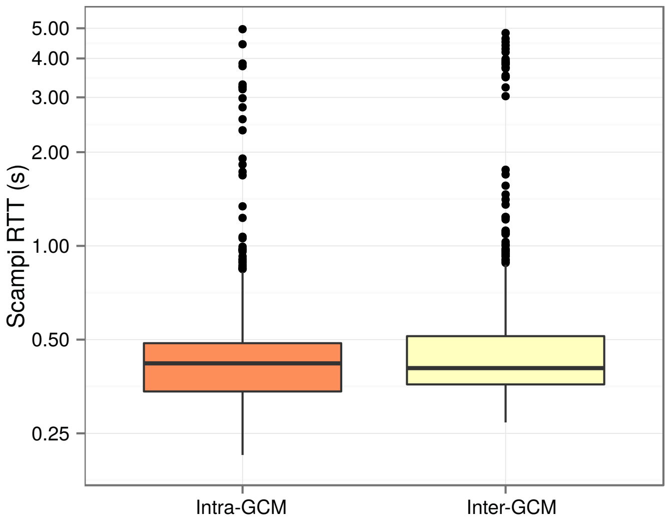
According to the results, the Scampi throughput is slightly better in the infrastructure scenario. However, the range of the boxes overlaps in most cases, which means that the differences between both scenarios are not statistically significant. Besides, the figure also shows that the performance of Scampi is really poor when compared to the results presented in section 9.2, and that it becomes worse as the bundle size becomes smaller. The reason is that the Scampi platform has a big overhead, not only because of the bundle protocol, but also due to computation overhead at the middleware layer; every bundle must be processed by several threads, including copying it to permanent storage before notifying it to the application.

Scampi Round Trip Time (RTT)

The RTT test consisted on generating a minimum-size bundle on the *source node*, that is then distributed by the Scampi router. When the application running on the *destination device* receives the bundle, it will generate a *response bundle* that confirms the reception of the first bundle. Finally, when the *source node* application receives the *response bundle* it computes the RTT. Figure 9.10b shows



(a) Scampi throughput.



(b) Scampi RTT test results.

Figure 9.10: Scampi results.

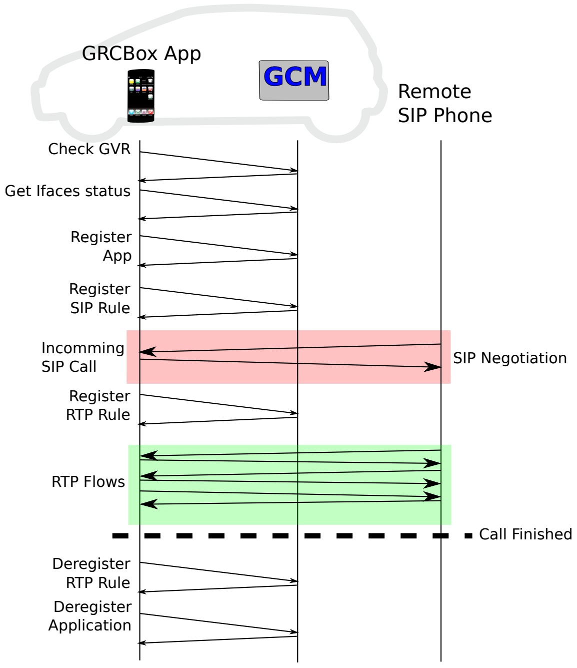


Figure 9.11: VoIP GRCBox application.

the obtained results.

As can be seen, there is almost no difference between the distributions of the RTT in both scenarios. When both devices are connected to the same AP, the Scampi RTT is slightly better than when using the GRCBox. However, when focusing on the value of the quartiles, this small difference becomes insignificant. It is worth noticing the high number of measurements that experienced a high RTT (represented as outliers). These high values are due to some instability issues found in the Scampi middleware under heavy load in Android.

9.4 VoIP Application over 3G

When making an in-vehicle VoIP call, it might be desirable to use the most stable available connection. Figure 9.11 illustrates the case of a GRCBox VoIP application attempting to use the highly-stable 3G connection to receive VoIP calls ignoring more unstable interfaces, such as WiFi. Due to the nature of VoIP connections, the application must follow these steps, which include creating 2 different rules:

1. Check the GCM availability.

2. Get the list of available interfaces and their status.

3. Register an incoming rule for Session Initiation Protocol(SIP) connections that includes only the local port, the local address, and the desired listening external interface.

4. When a SIP negotiation occurs, the GCM automatically creates as many NAT rules as required according to the parameters of the negotiated RTP flows.

5. At this time, the RTP packets can flow between the GRCBox application and its remote peer.

6. When the application is closed it must remove itself from the GCM applications database. This will also remove any other rule registered by the application.

In this case, the SIP rule is a permanent rule that must be active as long as the application is running. On the other hand, the RTP rules are managed by the *nf_nat_sip* kernel module and removed as soon as the VoIP call finishes. These steps are also valid to perform a direct V2V VoIP call.

This case illustrates how the GRCBox architecture can be used to perform real time communications between two users.

9.5 Other GRCBox Applications

We have designed GRCBox with the goal of creating a platform to easily test collaborative vehicular communication solutions for user devices. We are currently working on a variety of GRCBox applications that use the ad-hoc communication capabilities.

One of the applications is an *overtake assistant* application that streams video captured from the camera of a smartphone placed on the dashboard of the vehicle in front to the rear vehicle. The application sends periodic beacons through the GCM to announce the location of the vehicle. Using this information, rear vehicles can detect vehicles in front and ask for an overtake-assisting video stream.

We are also working on adapting our previously presented *Warning Ambulance Application*[2] [132] which was presented in Chapter 5. To run the original application, the user was forced to root its smartphone; in addition, no matter whether the device is rooted, not all the smartphones can be configured to use ad-hoc communications. By using the GRCBox architecture, the application will be easier to deploy and test.

[2]A video of this application can be found on our channel in youtube `https://www.youtube.com/watch?v=Wh4cwmdvecM`

Beside testing services that would require a high number of GCMs to be deployed, we envision GRCBox applications focused on vehicle platoons. The GRCBox architecture can be used to provide inter and intra-vehicle communication to friends or co-workers that move together using different vehicles. They can share multimedia content, directions, etc. without requiring a mobile broadband Internet connection which is usually hard to find on remote roads.

9.6 Summary

In this chapter we have presented how to use the GRCBox Architecture, which allows implementing vehicular applications by extending user devices' in-vehicle connectivity to external networks. We have presented four case study that illustrate the use of the GRCBox Architecture in different scenarios: *i)* access to the Internet, *ii)* ad-hoc V2V communication, *iii)* vehicular DTN communication, and *iv)* VoIP over 3G application. We have evaluated the performance of the GRCBox on the three first cases. Our results shown that the GRCBox Architecture is fully operational and that, in terms of throughput and delay, the penalty to pay when comparing the GRCBox against an infrastructure network is minimal and clearly compensated by the extra connectivity offered by the GRCBox.

Our GRCBox architecture demonstrated to be a more flexible alternative to dashboard integrated OBUs. The GRCBox architecture allow users to experience the possibilities of V2V and V2I applications while interacting with devices already integrated in their daily life such as smartphones. Besides, the GRCBox architecture reduces the entry barriers by offering a low-cost alternative to expensive brand-dependent OBUs. The GRCBox is also a quick V2V applications developing environment. Developers' programming skills can be fully reused when developing for the GRCBox.

The GRCBox Architecture have been released under an open source license and can be found in our GitHub page [46]. As an open source development, we want to invite the research community to download, use, and improve our GRCBox Architecture.

Part IV

Conclusions

Chapter 10

Conclusions, Publications and Future Work

Vehicular Networks and specifically VANETs present a highly variable network topology, as well as more frequent topology partitions than others adhoc networks like MANETs. Under these conditions, DTNs are considered an alternative able to deal with VANET characteristics, being applicable to VN to provide ITS services. This thesis has focuses on the combination of VNs and DTNs, called Vehicular Delay Tolerant Networks (VDTNs).

We have proposed a novel VDTN protocol designed to collect information from vehicular sensors. Our proposal, called MSDP, combines information about the localization obtained from a GNSS system with the actual street/road layout obtained from a Navigation System (NS) to define a new routing metric.

Concerning the deployment of VNs and VANETs technologies, the technology has left behind the innovation and the standardization phases, now it is time to reach the first early adopters in the market. However, most car manufacturers have decided to implement VN devices in the form of On Board Units (OBUs), which are expensive and difficult to update. OBUs are also heavy manufacturer dependent. These facts are delaying the deployment of VN. To boost the adoption of VN, in this thesis we have proposed and developed the GRCBox architecture. The GRCBox architecture is based in a low cost device and enables users to use V2V services while using general purpose devices like smartphones, tablets or laptops. We have combined our GRCBox with the Scampi DTN architecture to implement and test a VDTN scenario.

Below we briefly summarize the most relevant contributions of this thesis:

- **A survey of VDTN proposals** was presented in chapter 2. We provided the reader with a broad view of the different proposals for VDTNs. We classified them according to their utility index, showing the relationships between different protocols and their evolution. We identified a set of common mechanisms that can be applicable to almost all VDTN protocols, and that may heavily influence their performance. We presented some applications where VDTNs can be used and finally we evaluated the suitability of the different proposals for each considered application.

  Moreover, our survey is not limited to a mere description of protocols, since it also addresses critical issues such as the reproducibility and repeatability of experiments and reviews the evaluation methods used by the different VDTN researchers.

- **The MSDP protocol**. MSDP combines information obtained from the Geographic Information Service (GIS) with the actual street/road layout obtained from the Navigation System (NS) to define a new routing metric. MSDP routes information collected at sensors deployed in vehicles to RSU deployed by operators. The presented results show that MSDP outperforms both Greedy and GeOpps protocols in terms of both delivery ratio and introduced overhead.

- **The GOD DTN model**, a set of classes that allows to quickly implement DTN protocols. The GOD provides an implementation of several common low-level mechanism used in DTN protocols. When using the GOD, DTN protocols can be specified solely by their utility function, therefore, the GOD model simplifies the definition of DTN protocols. Protocols implemented using the GOD model can be easily compared. The GOD model increases the reproducibility and repeatability of VDTN protocols simulation experiments.

- **VACaMobil**, a new mobility manager for the OMNeT++ simulator which simplifies the definition of mobility for VANET simulations. The main novelty of VACaMobil is that: to the best of our knowledge, it is the first tool able to generate SUMO driven nodes in a vehicular network while ensuring the stability of certain user-defined parameters, such as the average, maximum, and minimum number of vehicles. VACaMobil heavily reduces the complexity of mobility parameters definition, thus improving the reproducibility and the repeatability of simulation experiments.

- **The GRCBox architecture** enables developers to make the most of opportunistic communications using in-vehicle user devices. With the GRCBox, users can create their own VANETs to communicate with their peers using their laptops, smartphones or tablets. Our proposal is implemented in a low-cost RaspberryPi hardware device and focuses on providing connectivity to several networks while the core of the applications is run on user devices.

Having accomplished all our objectives, the original goal of this thesis has been achieved and so this dissertation can now be concluded. The next section enumerates the publications related to this thesis. The last section of this chapter, Section 10.2, presents some open issues to assess in the future.

10.1 Publications

This chapter lists the publications that have been produced as a result of this thesis as well as some other collaborations and non-directly related publications we published during this time.

10.1.1 International Journals

- S. M. Tornell, S. Patra, C. T. Calafate, J.-C. Cano, and P. Manzoni. "GRCBox: Extending Smartphone Connectivity in Vehicular Networks". In: *International Journal of Distributed Sensor Networks* 2015.Article ID 478064 (2015), p. 13. DOI: `10.1155/2015/478064`. Impact Factor: 0.665 (JCR Q4).

  In this paper, we presented a preliminary version of our GRCBox architecture. This paper described the GRCBox public API and describes the modules that form the GCM.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "DTN Protocols for Vehicular Networks: An Application Oriented Overview". In: *IEEE Communications Surveys & Tutorials* 17.2 (2015), pp. 868–887. ISSN: 1553-877X. DOI: `10.1109/COMST.2014.2375340`. Impact Factor: 6.806 (JCR Q1).

  In this paper, we surveyed the most relevant previously presented DTN protocols. Our survey does not simply enumerate the different protocols but also analyzes them from the application point of view. We identified the suitability of different protocols for different applications. We also analyzed the DTN protocols evaluation methodology. We found that most researchers base their evaluations and comparisons on simulations. We also found that the high diversity of simulation tools compromises the repeatability and reproducibility of experiments.

- M. Báguena, S. M. Tornell, Á. Torres, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A Tool Offering Steady-State Simulations for VANETs". In: *Recent Advances in Communications and Networking Technology* 2.2 (2013), pp. 102–112. DOI: `10.2174/22117407112016660008`.

  In this paper, we presented VACaMobil, our mobility simulator. VACaMobil is able to provide quasi-steady state mobility scenarios. Using VACaMobil repeatability and reproducibility of VNs simulation-based experiments is improved.

10.1.2 International Conferences

- S. M. Tornell, S. Patra, C. T. Calafate, J.-C. Cano, and P. Manzoni. “A novel On-board Unit to Accelerate the Penetration of ITS Services”. In: *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC 2016)*. Las Vegas, USA, Jan. 2016, pp. 474–479. (CORE B)

  In this paper, we presented an evaluation of the GRCBox. The results demonstrated that using a low-cost device to provide VN connectivity to in-vehicle user devices is an alternative to OBUs.

- S. M. Tornell, T. Kärkkäinen, J. Ott, C. Calafate, J.-C. Cano, and P. Manzoni. “Simplifying The In-vehicle Connectivity for ITS Applications”. In: *MOBIQUITOUS 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2015. DOI: `10.4108/eai.22-7-2015.2260058`. (CORE A)

  In this paper, we presented our GRCBox architecture. We evaluated the impact on performance of the GRCBox architecture in both cases: when performing direct ad-hoc communications between vehicles, and when using a DTN application like Scampi.

- S. Patra, S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Video-Based Overtaking Assistance Now A Realit”. In: *Demonstrations of the 40th Annual IEEE Conference on Local Computer Networks (LCN-Demos 2015)*. Clearwater, USA. (CORE A)

  In this demo, we presented the EYES application, an overtaking assistance application.

- M. Báguena, S. M. Tornell, Á. Torres, C. T. Calafate, J.-C. Cano, and P. Manzoni. “VACaMobil: VANET Car Mobility Manager for OMNeT++”. In: *IEEE International Conference on Communications 2013: IEEE ICC’13 - 3rd IEEE International Workshop on Smart Communication Protocols and Algorithms (SCPA 2013) (ICC’13 - IEEE ICC’13 - Workshop SCPA)*. Budapest, Hungary, June 2013, pp. 1–5.

  In this paper, we presented a preliminary version of VACaMobil.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. “Assessing the Effectiveness of DTN Techniques Under Realistic Urban Environments”. In: *38th Annual IEEE Conference on Local Computer Networks (LCN 2013)*. Sydney, Australia, Oct. 2013. (CORE A)

In this paper, we compared through simulations the performance of our MSDP proposal against others DTN routing protocols. Our results demonstrated the superior performance of the MSDP.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, P. Manzoni, M. Fogue, and F. J. Martinez. "Evaluating the Feasibility of Using Smartphones for ITS Safety Applications". In: *VTC Spring 2013 IEEE Vehicular Technology Conference.* 2013. ISBN: 9781467363372. (CORE B)

  In this paper, we explored the use of smartphone for VN. Our results pointed out the limitations of smartphones and drove us to the design and the implementation of the GRCBox architecture.

- S. M. Tornell, E. Hernández-Orallo, C. T. Calafate, J.-C. Cano, P. Manzoni, and E. Hernández. "An Analytical Evaluation of a Map-based Sensor-data Delivery Protocol for VANETs". In: *14th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM 2013).* Madrid, Spain, June 2013. ISBN: 9781467358286. (CORE A)

  In this paper, we mathematically modeled the MSDP and the Epidemic protocol in order to compare them analytically. Our results showed that our proposal exhibit a superior performance.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, P. Manzoni, M. Fogue, and F. J. Martinez. "Implementing and Testing a Driving Safety Application for Smartphones Based on the eMDR Protocol". In: *2012 IFIP Wireless Days.* IEEE, Nov. 2012, pp. 1–3. ISBN: 978-1-4673-4404-3. DOI: 10.1109/WD.2012.6402816.

  In this paper, we presented a preliminary version of our ambulance warning application for smartphones. We used ad-hoc communication to disseminate warnings from a emergency vehicle.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A Map-based Sensor Data Delivery Protocol for Vehicular Networks". In: *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net).* Ayia Napa, Cyprus: IEEE, June 2012, pp. 1–8. ISBN: 978-1-4673-2039-9. DOI: 10.1109/MedHocNet.2012.6257118.

  In this paper, we presented the MSDP. We compared its performance with the epidemic protocol performance using simulations.

10.1.3 Other Publications

- J. M. Marquez-Barja, H. Ahmadi, S. M. Tornell, C. T. Calafate, J.-C. Cano, P. Manzoni, and L. A. DaSilva. "Breaking the Vehicular Wireless Communications Barriers: Vertical Handover Techniques for Heterogeneous Networks". In: *IEEE Transactions on Vehicular Technology* 64.12 (Dec. 2015), pp. 5878–5890. ISSN: 0018-9545. DOI: `10.1109/TVT.2014.2386911`. Impact Factor: 1.978 (JCR Q1)

In this paper, we present an overview of vertical handover techniques and propose an algorithm empowered by the IEEE 802.21 standard, which considers the particularities of the vehicular networks (VNs), the surrounding context, the application requirements, the user preferences, and the different available wireless networks [i.e., Wireless Fidelity (Wi-Fi), Worldwide Interoperability for Microwave Access (WiMAX), and Universal Mobile Telecommunications System (UMTS)] to improve users' quality of experience (QoE). Our results demonstrate that our approach, under the considered scenario, is able to meet application requirements while ensuring user preferences are also met.

- S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Accelerating Vehicle Network Simulations in Urban Scenarios Through Caching". In: *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2014)*. July 2014, pp. 654–661. DOI: `10.1109/SPECTS.2014.6880007`

  In this paper, we present an optimization applicable to the obstacle model included in VEINS, a well-known VN simulator, which relays on a cache table to accelerate ray-tracing calculations at the physical layer. Our results show that the proposed optimization can reduce by up to 75% the simulation time with minimal differences in terms of simulation results.

10.2 Future Work

In this section we enumerate several improvements to the contributions presented in this thesis:

- MSDP possibilities may be increased by using cellular communication technologies such as 3G or 4G. In the future, it would be interesting to study the combination of different technologies.
- When using MSDP, aggregating the information coming from several sensors will improve the performance of the protocol. In the future, we will explore different aggregation and disaggregation mechanisms.
- We will analyze the optimal routes used by the Ideal DTN model in order to mimic them and make new improvements to our MSDP.
- VACaMobil may be extended to be able to associate different node roles to different types of vehicles, for example, "buses" could have a bigger transmission power than "cars".

- We plan to improve GRCBox by implementing high-level rules, thereby creating a new semantic able to support rule definitions such as “use the most stable interface for VoIP calls” or “forward multicast packets to every ad-hoc or infrastructure network”.

- To test the GRCBox, we have only tested 802.11a and 802.11g, it would be desirable to install 802.11p hardware and update the GRCBox API to support WAVE applications.

- The GRCBox may also be connected to vehicles’ internal data buses such as the OBD-II bus or the CAN bus to read car status related information and make it available to user devices through the public API.

- Currently, when using the GRCBox management application, the user can only define which interface wants to use, being unable to configure the interface properties. We plan to support remote interface configuration. The management application can also be improved by including some predefined rules for some common third-party applications.

- We have designed the GRCBox Architecture as a quick deployment platform for research, allowing to easily test vehicular applications for smartphones; therefore, we have considered that all the devices in the network were trustworthy. Currently, it would be simple to create a malicious application that compromises the system. In case of commercial deployment plans, further research in this direction must be carried out.

10.3 Special Acknowledgements

This thesis has been supported by the *Ministerio de Economía y Competitividad, Programa nacional de formación de Recursos Humanos, Subprograma de Formación de Personal Investigador* under Grant BES-2012-052673.

10.3.1 Scholarships

Between August, 2014 and November, 2014 I did a scholarship in Äalto University, Espoo, Finland. This scholarship was funded by the *Ministerio de Economía y Competitividad, Programa nacional de formación de Recursos Humanos, Subprograma de Formación de Personal Investigador* under Grant EEBB-I-14-07890.

Between September, 2015 and December, 2015 I did a scholarship in Universidade da Beira Interior, Covilhã, Portugal. This scholarship was funded by the *Ministerio de Economía y Competitividad, Programa nacional de formación de Recursos Humanos, Subprograma de Formación de Personal Investigador* under Grant EEBB-I-15-09222.

10.3.2 Projects

During this thesis I participated in several projects, namely:

- "Walkie-Talkie: Vehicular Communication Systems to Enable Safer, Smarter, and Greener Transportation", which was funded by the *Ministerio de Economía y Competitividad*, Spain, under Grant TIN2011-27543-C03-01.
- "Smart@CarPhone: Toward Seamless Smartphone and Vehicle Integration to Connect Drivers with Sensors and the Environment in a Holistic Service-Oriented Architecture" which was funded by *Ministerio de Economía y Competitividad, Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, Proyectos I+D+I 2014*, Spain, under Grant TEC2014-52690-R

Part V

Appendices and References

Appendix A

Acronyms

ADPBSW	Average Delivery Probability Binary Spray and Wait	29
API	Application Programming Interface	11
BS&W	Binary Spray & Wait	23
CCC	Car Connectivity Consortium	84
CAN	Controller Area Network	47
CDF	Cumulative Distribution Function	
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance	10
DDT	Distance Defer Transmission	19
DRTAR	Distributed Real-time Data Traffic Statistics Assisted Routing	31
DSCF	Directional Store-Carry-Forward	19
DSRC	Dedicated short-range communications	106
DTFR	Delay Tolerant Firework Routing	29
DTN	Delay Tolerant Network	3
DTNRG	Delay Tolerant Network Research Group	4
DV-CAST	Distributed Vehicular BroadCast	19
EC	European Commission	10
Eirp	Equivalent isotropically radiated power	106
eMDR	enhanced Message Dissemination based on Roadmaps	75
ETSI	European Telecommunications Standards Institute	10

FCC	Federal Communications Commission . . . 9
FEC	Forward Error Correction . . . 35
FFRDV	Fastest Ferry Routing in DTN-enabled Vehicular Ad-Hoc . . . 19
GCM	GRCBox Connectivity Manager . . . xx
GIS	Geographic Information Service . . . 5
GOD	Generic One-Copy DTN Model . . . 5
GORF	Generic Opportunistic Routing Framework . . . 13
GPS	Global Positioning System . . . 21
IPN	InterPlanetary Network . . . 4
ITS	Intelligent Transport System . . . 3
IVC	Inter-Vehicle Communication . . . 43
LOS	Line of Sight . . . 78
MANET	Mobile Ad-hoc NETwork . . . 3
METD	Minimum Estimated Time of Delivery . . . 29
MSDP	Map-based Sensor-data Delivery Protocol . . . 5
MTU	Maximum Transfer Unit . . . 57
NAT	Network Address Translation . . . 135
NS	Navigation System . . . 5
OBD-II	On Board Diagnostic 2 . . . 51
OBU	On Board Unit . . . 4
P2P	Peer-to-Peer . . . 35
PER	Packet Error Rate . . . 34
RENA	REgioN-bAsed . . . 29
REST	Representational State Transfer . . . 84
ROD	Road Oriented Dissemination . . . 19
ROI	Region of Interest . . . 20
RSU	Road Side Unit . . . 14
RTT	Round Trip Time . . . 135
SADV	Static-Node-Assisted Adaptive Data Dissemination in Vehicular Networks . . . 31
SUMO	Simulation of Urban MObility . . . 106
TTL	Time To Live . . . 33
UV-CAST	Urban Vehicular BroadCast . . . 19

V2V	Vehicle to Vehicle	3
V2I	Vehicle to Infrastructure	3
VADD	Vehicle-Assisted Data Delivery	31
VANET	Vehicular Ad-Hoc Network	3
VDTN	Vehicular Delay Tolerant Network	4
VN	Vehicular Network	3
WAVE	Wireless Access for Vehicular Environment	9
WSMP	WAVE Short Message Protocol	10

Appendix B

GRCBox REST API

- Root resource "/":
  - Method GET: information about the status of the server and the number of rules already registered in the database.
- Ifaces resource "/ifaces":
  - Method GET: a simplified list of all the available outgoing interfaces.
- Iface resource "/ifaces/{ifId}":
  - Method GET: information of a specific interface.
  - Method POST: at this moment this is not implemented, but we expect to allow authorized applications to remotely configure certain interface parameters such as the SSID or the Password for wireless interfaces.
- Applications resource "/apps":
  - Method GET: a list of the currently registered applications in the system.
  - Method POST: register a new application and return a secret password for later authentication.
- Application resource "/apps/{appId}": When a new application is registered a new specific resource is created. Access to the POST and DELETE methods is restricted to the original application.

  - Method GET: information about the specific application, its name and its last-seen value.
  - Method POST: a call to this method is interpreted as a keep-alive signal by the server. If an application does not post to its ID for a certain amount of time, the application is deregistered and its defined rules are deleted from the database and from the system.
  - Method DELETE: remove an application and all its rules from the database and from the system.

- Rules resource "/apps/{appId}/rules": Each registered application can access to its list of rules. Access to the POST method is restricted to the *owner* of the resource.

  - Method GET: a list of the rules defined by this application.
  - Method POST: create a new rule.

- Rule resource "/apps/{appId}/rules/{ruleId}": This resource is accessible when a new rule is created.

  - Method GET: details of the rule.
  - Method DELETE: remove a rule from database and system.

Bibliography

[1] W. Alasmary and W. Zhuang. "The Mobility Impact in IEEE 802.11p Infrastructureless Vehicular Networks." In: *Vehicular Technology Conference Fall VTC 2010 IEEE 72nd*. Vol. 10. 2. Ottawa, ON, 2010, pp. 222–230. ISBN: 9781424435746 (cited on p. 43).

[2] A. Amoroso, G. Marfia, M. Roccetti, and G. Pau. "Creative Testbeds for VANET Research: A New Methodology." In: *2012 IEEE Consumer Communications and Networking Conference (CCNC)*. Las Vegas, NV, Jan. 2012, pp. 477–481. ISBN: 978-1-4577-2071-0. DOI: `10.1109/CCNC.2012.6181007` (cited on p. 48).

[3] Apple Inc. *CarPlay*. https://www.apple.com/ios/carplay/. Feb. 2016 (cited on p. 84).

[4] Arada Systems. *Arada WAVE white paper*. http://www.aradasystems.com/pdfs/Arada_IEEE_WAVE_paper.pdf (cited on p. 11).

[5] M. Baguena, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Towards Realistic Vehicular Network Simulation Models." In: *2012 IFIP Wireless Days*. Dublin, Ireland: IEEE, Nov. 2012, pp. 1–3. ISBN: 978-1-4673-4404-3. DOI: `10.1109/WD.2012.6402805` (cited on p. 106).

[6] M. Báguena, S. M. Tornell, Á. Torres, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A Tool Offering Steady-State Simulations for VANETs." In: *Recent Advances in Communications and Networking Technology* 2.2 (2013),

pp. 102–112. doi: 10.2174/22117407112016660008 (cited on pp. 7, 61, 115, 151).

[7] M. Báguena, S. M. Tornell, Á. Torres, C. T. Calafate, J.-C. Cano, and P. Manzoni. “VACaMobil: VANET Car Mobility Manager for OMNeT++.” In: *IEEE International Conference on Communications 2013: IEEE ICC'13 - 3rd IEEE International Workshop on Smart Communication Protocols and Algorithms (SCPA 2013) (ICC'13 - IEEE ICC'13 - Workshop SCPA).* Budapest, Hungary, June 2013, pp. 1–5 (cited on pp. 61, 106, 115, 152).

[8] A. Balasubramanian, B. Levine, and A. Venkataramani. “DTN Routing As a Resource Allocation Problem.” In: *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.* SIGCOMM '07. Kyoto, Japan: ACM, 2007, pp. 373–384. isbn: 978-1-59593-713-1. doi: 10.1145/1282380.1282422 (cited on pp. 17, 22, 35, 40).

[9] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. “SUMO–Simulation of Urban MObility: An Overview.” In: *SIMUL 2011, The Third International Conference on Advances in System Simulation.* Barcelona, Spain, Oct. 2011, pp. 63–68 (cited on pp. 45, 47, 62, 65, 106, 115, 128).

[10] S. M. Bilal, A. U. R. Khan, S. U. Khan, S. A. Madani, B. Nazir, and M. Othman. “Road Oriented Traffic Information System for Vehicular Ad hoc Networks.” In: *Wireless Personal Communications* (Feb. 2014). issn: 0929-6212. doi: 10.1007/s11277-014-1651-0 (cited on p. 32).

[11] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. “MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks.” In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings.* Vol. 6. c. Barcelona, Spain, Apr. 2006, pp. 1–11. isbn: 1-4244-0221-2. doi: 10.1109/INFOCOM.2006.228 (cited on pp. 14, 17, 22, 35, 40, 47, 48).

[12] S. Burleigh, A. Hooke, and L. Torgerson. “Delay-tolerant Networking: an Approach to Interplanetary Internet.” In: *IEEE Communications Magazine* 41.6 (2003), pp. 128–136 (cited on p. 11).

[13] H. Cai and D. Y. Eun. “Crossing Over the Bounded Domain: From Exponential to Power-Law Intermeeting Time in Mobile Ad Hoc Networks.” In: *IEEE/ACM Transactions on Networking* 17.5 (Aug. 2009), pp. 1578–1591 (cited on p. 96).

[14] J. Cano, J.-C. Cano, C. K. Toh, C. T. Calafate, and P. Manzoni. “Easy-MANET: An extensible and configurable platform for service provisioning in MANET environments.” In: *IEEE Communications Magazine* 48 (2010), pp. 159–167. ISSN: 01636804. DOI: `10.1109/MCOM.2010.5673087` (cited on p. 135).

[15] Car Connectivity Consortium (CCC). *MirrorLink*. http://www.mirrorlink.com/. Feb. 2016 (cited on p. 84).

[16] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. *Delay-Tolerant Networking Architecture*. RFC 4838 (Informational). Internet Engineering Task Force, Apr. 2007 (cited on pp. 3, 11).

[17] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. “Impact of Human Mobility on Opportunistic Forwarding Algorithms.” In: *IEEE Transactions on Mobile Computing* 6 (June 2007), pp. 606–620 (cited on p. 96).

[18] L.-J. Chen, Y.-Y. Chen, K.-c. Lan, and C.-M. Chou. “Localized Data Dissemination in Vehicular Sensing Networks.” In: *Vehicular Networking Conference (VNC), 2009 IEEE*. Oct. 2009, pp. 1–6. DOI: `10.1109/VNC.2009.5416362` (cited on p. 46).

[19] W. Chen, R. Guha, J. Chennikara-Varghese, M. Pang, R. Vuyyuru, and J. Fukuyama. “Context-driven Disruption Tolerant Networking for Vehicular Applications.” In: *2010 IEEE Vehicular Networking Conference*, pp. 33–40. ISBN: 978-1-4244-9526-9. DOI: `10.1109/VNC.2010.5698265` (cited on pp. 17, 40, 47).

[20] P.-C. Cheng, K. C. Lee, M. Gerla, and J. Härri. “GeoDTN+Nav: Geographic DTN Routing with Navigator Prediction for Urban Vehicular Environments.” In: *Mobile Networks and Applications* 15.1 (June 2009), pp. 61–82. ISSN: 1383-469X. DOI: `10.1007/s11036-009-0181-6` (cited on pp. 17, 29, 37, 41).

[21] M. Cherif, S.-M. Secouci, and B. Ducourthial. “How to Disseminate Vehicular Data Efficiently in Both Highway and Urban Environments?” In: *IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. Niagara Falls, ON, 2010, pp. 165–171 (cited on pp. 17, 19, 40).

[22] E. C. Committee. *ECC RECOMMENDATION (08)01, USE OF THE BAND 5855-5875 MHz FOR INTELLIGENT TRANSPORT SYSTEMS (ITS).* 2008 (cited on p. 10).

[23] Commsignia Ltd. *Commsignia Web Page.* http://www.commsignia.com/ (cited on p. 11).

[24] E. M. Daly and M. Haahr. "Social Network Analysis for Routing in Disconnected Delay-tolerant MANETs." In: *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc '07* (2007), p. 32. DOI: 10.1145/1288107.1288113 (cited on pp. 17, 24, 25, 40).

[25] *D-Bus.* http://dbus.freedesktop.org/. Feb. 2016 (cited on p. 88).

[26] *Delay Tolerant Networks Research Group.* https://sites.google.com/site/dtnresgroup/home. Feb. 2016 (cited on pp. 4, 11).

[27] M. Demmer, J. Ott, and S. Perreault. *Delay Tolerant Networking TCP Convergence Layer Protocol.* Internet Draft (Experimental). Internet Engineering Task Force, Sept. 2013 (cited on p. 13).

[28] Y. Ding and L. Xiao. "SADV: Static-node-assisted Adaptive Data Dissemination in Vehicular Networks." In: *IEEE Transactions on Vehicular Technology* 59.5 (2010), pp. 2445–2455 (cited on pp. 17, 31, 32, 41).

[29] *DTN reference implementation.* https://sites.google.com/site/dtnresgroup/home/code/dtn2documentation. Feb. 2016 (cited on p. 48).

[30] D. Eckhoff, C. Sommer, and F. Dressler. "On the Necessity of Accurate IEEE 802.11p Models for IVC Protocol Simulation." In: *Vehicular Technology Conference Spring.* Ivc. Yokohama, Japan, 2012, pp. 1–5. ISBN: 9781467309905 (cited on p. 43).

[31] J. Eriksson, H. Balakrishnan, and S. Madden. "Cabernet: Vehicular Content Delivery Using WiFi." In: *Proceedings of the 14th ACM international conference on Mobile computing and networking.* MobiCom '08. New York, NY, USA, pp. 199–210. ISBN: 978-1-60558-096-8. DOI: 10.1145/1409944.1409968 (cited on p. 48).

[32] ETSI. *Intelligent Transport Systems.* http://www.etsi.org/technologies-clusters/technologies/intelligent-transport (cited on p. 8).

[33] *European Automobile Manufacturers Association.* http://www.acea.be. Feb. 2016 (cited on p. 10).

[34] K. Fall. "A Delay-tolerant Network Architecture for Challenged Internets." In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '03.* SIGCOMM '03. New York, New York, USA: ACM Press, 2003, p. 27. ISBN: 1581137354. DOI: 10.1145/863955.863960 (cited on pp. 4, 11).

[35] R. T. Fielding. "Architectural styles and the design of network-based software architectures." PhD thesis. University of California, 2000 (cited on p. 84).

[36] M. Fogue, P. Garrido, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni. "Using Roadmap Profiling to Enhance the Warning Message Dissemination in Vehicular Environments." In: *36th IEEE Conference on Local Computer Networks (LCN 2011).* Bonn, Germany, Oct. 2011 (cited on p. 46).

[37] W. Gao, Q. Li, B. Zhao, and G. Cao. "Multicasting in delay tolerant networks: a social network perspective." In: *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing.* MobiHoc '09. New York, NY, USA: ACM, 2009, pp. 299–308 (cited on p. 96).

[38] E. Garcia-Lozano, C. Campo, C. Garcia-Rubio, and A. Cortes-Martin. "Bandwidth-efficient Techniques for Information Dissemination in Urban Vehicular Networks." In: *Proceedings of the 11th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor,& Ubiquitous Networks.* PE-WASUN '14. New York, NY, USA: ACM, 2014, pp. 61–68. ISBN: 978-1-4503-3025-1. DOI: 10.1145/2653481.2653483 (cited on p. 71).

[39] E. Garcia-Lozano, C. Campo, C. Garcia-Rubio, and A. Rodriguez-Carrion. "Adapting a Bandwidth-Efficient Information Dissemination Scheme for Urban VANETs." In: *Proceedings of Ubiquitous Computing and Ambient Intelligence. Sensing, Processing, and Using Environmental Information: 9th International Conference, UCAmI.* Puerto Varas, Chile: Springer International Publishing, Dec. 2015, pp. 72–83. ISBN: 978-3-319-26401-1. DOI: 10.1007/978-3-319-26401-1 (cited on p. 71).

[40] M. Gerla, J.-T. Weng, E. Giordano, and G. Pau. "Vehicular Testbeds; Validating Models and Protocols Before Large Scale Deployment." In: *2012 International Conference on Computing, Networking and Communications (ICNC).* Vol. 7. 6. Maui, HI, 2012, pp. 665–669. ISBN: 9781467300094. DOI: `10.1109/ICCNC.2012.6167506` (cited on pp. 10, 47).

[41] F. Gil-Castineira, F. Gonzalez-Castano, and L. Franck. "Extending Vehicular CAN Fieldbuses with Delay-tolerant Networks." In: *IEEE Transactions on Industrial Electronics* 55.9 (2008), pp. 3307–3314 (cited on pp. 17, 40, 47).

[42] Google Inc. *Android Auto.* http://www.android.com/auto/. Feb. 2016 (cited on p. 84).

[43] R. Gorcitz, P. Spathis, M. Dias de Amorim, R. Wakikawa, and S. Fdida. "SERVUS: Reliable Low-cost and Disconnection-aware Broadcasting in VANETs." In: *2011 7th International Wireless Communications and Mobile Computing Conference (IWCMC).* Istanbul, 2011, pp. 1760–1765 (cited on pp. 17, 19, 21, 41).

[44] J. Gozalvez, M. Sepulcre, and R. Bauza. "Impact of the Radio Channel Modelling on the Performance of VANET Communication Protocols." In: *Telecommunication Systems* 50.3 (Dec. 2010), pp. 149–167. ISSN: 1018-4864. DOI: `10.1007/s11235-010-9396-x` (cited on p. 43).

[45] M. Gramaglia, M. Calderon, and C. Bernardos. "TREBOL: Tree-Based Routing and Address Autoconfiguration for Vehicle-to-Internet Communications." In: *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring).* May 2011, pp. 1–5. DOI: `10.1109/VETECS.2011.5956233` (cited on p. 45).

[46] *GRC GitHub Account.* https://github.com/GRCDEV. Feb. 2016 (cited on p. 146).

[47] R. Groenevelt, P. Nain, and G. Koole. "The Message Delay in Mobile Ad Hoc Networks." In: *Performance Evaluation* 62 (Oct. 2005), pp. 210–228 (cited on pp. 96, 97).

[48] M. Grossglauser and D. N. C. Tse. "Mobility Increases the Capacity of Ad Hoc Wireless Networks." In: *IEEE/ACM Transactions on Networking* 10.4 (2002), pp. 477–486 (cited on p. 22).

[49] J. Härri, F. Filali, C. Bonnet, and M. Fiore. “VanetMobiSim: Generating Realistic Mobility Patterns for VANETs.” In: *Proceedings of the 3rd international workshop on Vehicular ad hoc networks - VANET ’06*. New York, NY, USA, 2006, pp. 96–97. ISBN: 1595935401. DOI: `10.1145/1161064.1161084` (cited on p. 47).

[50] H. Hartenstein and K. Laberteaux. “A Tutorial Survey on Vehicular Ad Hoc Networks.” In: *IEEE Communications Magazine* June (2008), pp. 164–171. DOI: `10.1109/MCOM.2008.4539481` (cited on p. 3).

[51] T.-K. Huang, C.-K. Lee, and L.-J. Chen. “PRoPHET+: An Adaptive PRoPHET-Based Routing Protocol for Opportunistic Network.” In: *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. Apr. 2010, pp. 112–119. ISBN: 978-1-4244-6695-5. DOI: `10.1109/AINA.2010.162` (cited on pp. 17, 24, 25, 41).

[52] B. Hull, V. Bychkovsky, Y. Zhang, et al. “CarTel: A Distributed Mobile Sensor Computing System.” In: *4th ACM SenSys*. Boulder, CO, Nov. 2006 (cited on p. 10).

[53] “IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments.” In: *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)* (July 2010), pp. 1–51. DOI: `10.1109/IEEESTD.2010.5514475` (cited on p. 8).

[54] *IEEE Standard for Information Technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 10: Mesh Networking*. Tech. rep. 2011, pp. 1–372. DOI: `10.1109/IEEESTD.2011.6018236` (cited on p. 3).

[55] *Inet framwork for OMNeT++*. http://inet.omnetpp.org/. Feb. 2016 (cited on pp. 42, 63, 71).

[56] International Organization for Standardization. *ISO 15765: Road vehicles, Diagnostics on Controller Area Networks (CAN)*. 2004 (cited on pp. 51, 105).

[57] *InterPlanetary Networks project.* http://www.ipnsig.org. Feb. 2016 (cited on p. 11).

[58] *Iperf homepage.* https://iperf.fr/. Feb. 2016 (cited on p. 138).

[59] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. "Optimized Link State Routing Protocol for Ad Hoc Networks." In: *IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings.* IEEE, Aug. 2001, pp. 62–68. ISBN: 0-7803-7406-1. DOI: `10.1109/INMIC.2001.995315` (cited on p. 8).

[60] D. Jiang and L. Delgrossi. "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments." In: *IEEE Vehicular Technology Conference VTC Spring 2008.* Marina Bay, Singapore: IEEE, 2008, pp. 2036–2040. ISBN: 9781424416455 (cited on pp. 9, 106).

[61] S. Joerer, C. Sommer, and F. Dressler. "Toward Reproducibility and Comparability of IVC Simulation Studies: A Literature Survey." In: *IEEE Communications Magazine* 50.10 (Oct. 2012), pp. 82–88. DOI: `10.1109/MCOM.2012.6316780` (cited on pp. 43, 47).

[62] P. Juang, H. Oki, Y. Wang, and M. Martonosi. "Energy-efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet." In: *SIGPLAN Not.* 37.10 (2002), pp. 96–107 (cited on pp. 11, 24).

[63] T. Kärkkäinen, M. Pitkänen, P. Houghton, and J. Ott. "SCAMPI Application Platform." In: *Proceedings of the Seventh ACM International Workshop on Challenged Networks.* CHANTS '12. New York, NY, USA: ACM, 2012, pp. 83–86. ISBN: 978-1-4503-1284-4. DOI: `10.1145/2348616.2348636` (cited on pp. 138, 140).

[64] B. Karp and H. T. Kung. "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks." In: *Proceedings of the 6th annual international conference on Mobile computing and networking.* MobiCom '00. Boston, Massachusetts, United States: ACM, 2000, pp. 243–254. ISBN: 1-58113-197-6. DOI: `10.1145/345910.345953` (cited on pp. 26, 31, 59, 71).

[65] A. Keränen, J. Ott, and T. Kärkkäinen. "The ONE simulator for DTN protocol evaluation." In: *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques.* Brussels, Belgium, 55:1–55:10. ISBN:

978-963-9799-45-5. DOI: `10.4108/ICST.SIMUTOOLS2009.5674` (cited on p. 42).

[66] M. Khabbaz, C. Assi, and W. Fawaz. "Disruption-tolerant networking: A Comprehensive Survey on Recent Developments and Persisting Challenges." In: *IEEE Communications Surveys & Tutorials* 14.2 (2012), pp. 607–640 (cited on p. 140).

[67] O. Khalid, S. U. Khan, J. Kolodziej, et al. "A Checkpoint Based Message Forwarding Approach For Opportunistic Communication." In: *ECMS*. 2012, pp. 512–518. ISBN: 9780956494443 (cited on p. 32).

[68] J. Kolodziej, S. U. Khan, L. Wang, N. Min-Allah, S. A. Madani, N. Ghani, and H. Li. "An Application of Markov Jump Process Model for Activity-Based Indoor Mobility Prediction in Wireless Networks." In: *Frontiers of Information Technology (FIT), 2011*. Dec. 2011, pp. 51–56. DOI: `10.1109/FIT.2011.17` (cited on p. 4).

[69] G. V. Kumar, Y. V. Reddyr, and M. Nagendra. "Current Research Work on Routing Protocols for MANET: A Literature Survey." In: *(IJCSE) International Journal on Computer Science and Engineering* 02.03 (2010), pp. 706–713 (cited on pp. 3, 8, 22).

[70] S. Kuribayashi, Y. Sakumoto, S. Hasegawa, H. Ohsaki, and M. Imase. "Performance Evaluation of Broadcast Communication Protocol DSCF (Directional Store-Carry-Forward) for VANETs with Two-Dimensional Road Model." In: *2009 10th International Symposium on Pervasive Systems Algorithms and Networks* (2009), pp. 615–619. DOI: `10.1109/I-SPAN.2009.65` (cited on pp. 17, 19, 40).

[71] J. LeBrun, C.-N. Chuah, D. Ghosal, and M. Zhang. "Knowledge-based Opportunistic Forwarding in Vehicular Wireless Ad Hoc Networks." In: *IEEE 61st Vehicular Technology Conference, VTC Spring*. Vol. 4. Dallas, Texas, 2005, pp. 2289–2293. DOI: `10.1109/VETECS.2005.1543743` (cited on pp. 17, 26, 40).

[72] K. Lee, S.-h. Lee, R. Cheung, U. Lee, and M. Gerla. "First Experience with CarTorrent in a Real Vehicular Ad Hoc Network Testbed." In: *2007 Mobile Networking for Vehicular Environments* (2007), pp. 109–114 (cited on p. 47).

[73] I. Leontiadis, P. Costa, and C. Mascolo. “Extending Access Point Connectivity through Opportunistic Routing in Vehicular Networks.” In: *Proceedings of the 29th Conference on Information Communications (INFOCOM).* Piscataway, NJ, USA, Mar. 2010, pp. 486–490. ISBN: 978-1-4244-5836-3. DOI: `10.1109/INFCOM.2010.5462185` (cited on pp. 17, 28, 29, 38).

[74] I. Leontiadis and C. Mascolo. “GeOpps: Geographical Opportunistic Routing for Vehicular Networks.” In: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007.* Helsinki, Finland, June 2007, pp. 1–6. ISBN: 978-1-4244-0993-8. DOI: `10.1109/WOWMOM.2007.4351688` (cited on pp. 17, 26, 28, 29, 37, 40, 42, 43, 59, 71, 95, 105).

[75] X. Li, W. Shu, M. Li, H. Huang, and M.-Y. Wu. “DTN Routing in Vehicular Sensor Networks.” In: *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference.* New Orleans, LO: IEEE, Nov. 2008, pp. 1–5. ISBN: 978-1-4244-2324-8. DOI: `10.1109/GLOCOM.2008.ECP.150` (cited on pp. 17, 22, 35, 40, 45).

[76] Y. Li, G. Su, D. Wu, D. Jin, L. Su, and L. Zeng. “The Impact of Node Selfishness on Multicasting in Delay Tolerant Networks.” In: *IEEE Transactions on Vehicular Technology* 60.5 (June 2011), pp. 2224–2238 (cited on p. 96).

[77] A. Lindgren, E. Davies, and A. Doria. *Epidemic Routing Module for Generic Opportunistic Routing Framework.* Internet Draft (Experimental). Internet Engineering Task Force, July 2013 (cited on p. 13).

[78] A. Lindgren, E. Davies, and A. Doria. *Generic Opportunistic Routing Framework.* Internet Draft (Experimental). Internet Engineering Task Force, July 2013 (cited on p. 13).

[79] A. Lindgren, A. Doria, O. Schelén, and O. Schelen. “Probabilistic Routing in Intermittently Connected Networks.” In: *SIGMOBILE Mob. Comput. Commun. Rev.* 7.3 (July 2003), pp. 19–20. ISSN: 15591662. DOI: `10.1145/961268.961272` (cited on pp. 13, 17, 24, 35, 40).

[80] J. Liu, J. Bi, Y. Bian, X. Liu, and Z. Li. “DSRelay: A Scheme of Cooperative Downloading Based on Dynamic Slot.” In: *2012 IEEE 12th International Conference on Computer and Information Technology.* 2011. Ottawa, ON, 2012, pp. 381–386. ISBN: 9781457720536 (cited on pp. 17, 38, 41).

[81] C.-C. Lo, J.-W. Lee, C.-H. Lin, M.-F. Horng, and Y.-H. Kuo. "A Co-operative Destination Discovery Scheme to Support Adaptive routing in VANETs." In: *Vehicular Networking Conference (VNC), 2010 IEEE.* Dec. 2010, pp. 202–208. DOI: 10.1109/VNC.2010.5698275 (cited on p. 46).

[82] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein. "Geographic routing in city scenarios." In: *SIGMOBILE Mob. Comput. Commun. Rev.* 9.1 (2005), pp. 69–72 (cited on p. 26).

[83] J. Louvel, T. Templier, and T. Boileau. *Restlet in Action: Developing RESTful Web APIs in Java.* Greenwich, CT, USA: Manning Publications Co., 2012. ISBN: 193518234X, 9781935182344 (cited on p. 85).

[84] P. Luo, H. Huang, W. Shu, M. Li, and M.-Y. Wu. "Performance Evaluation of Vehicular DTN Routing under Realistic Mobility Models." In: *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE.* Vol. 2. Las Vegas, NV, Mar. 2008, pp. 2206–2211. ISBN: 978-1-4244-1997-5. DOI: 10.1109/WCNC.2008.390 (cited on pp. 17, 22, 35, 40, 43).

[85] J. M. Marquez-Barja, H. Ahmadi, S. M. Tornell, C. T. Calafate, J.-C. Cano, P. Manzoni, and L. A. DaSilva. "Breaking the Vehicular Wireless Communications Barriers: Vertical Handover Techniques for Heterogeneous Networks." In: *IEEE Transactions on Vehicular Technology* 64.12 (Dec. 2015), pp. 5878–5890. ISSN: 0018-9545. DOI: 10.1109/TVT.2014.2386911 (cited on p. 153).

[86] F. Martinez, J.-C. Cano, C. Calafate, and P. Manzoni. "CityMob: A Mobility Model Pattern Generator for VANETs." In: *IEEE International Conference on Communications Workshops, 2008. ICC Workshops '08.* May 2008, pp. 370–374. DOI: 10.1109/ICCW.2008.76 (cited on p. 45).

[87] F. J. Martinez, M. Fogue, M. Coll, J.-C. Cano, C. T. Calafate, and P. Manzoni. "Evaluating the Impact of a Novel Warning Message Dissemination Scheme for VANETs Using Real City Maps." In: *Proceedings of the 9th IFIP TC 6 international conference on Networking.* NETWORKING'10. Chennai, India: Springer-Verlag, 2010, pp. 265–276. ISBN: 3-642-12962-5, 978-3-642-12962-9. DOI: 10.1007/978-3-642-12963-6_21 (cited on p. 75).

[88] S. Medjiah and T. Ahmed. "Orion Routing Protocol for Delay Tolerant Networks." In: *2011 IEEE International Conference on Communications*

(ICC). Kyoto, Japan, June 2011, pp. 1–6. isbn: 978-1-61284-232-5. doi: `10.1109/icc.2011.5963362` (cited on pp. 17, 29, 41).

[89] R. Meireles, M. Boban, P. Steenkiste, O. Tonguz, and J. Barros. "Experimental study on the impact of vehicular obstructions in VANETs." In: *2010 IEEE Vehicular Networking Conference* (Dec. 2010), pp. 338–345. doi: `10.1109/VNC.2010.5698233` (cited on p. 34).

[90] K. Mershad, H. Artail, and M. Gerla. "We Can Deliver Messages to Far Vehicles." In: *IEEE Transportation Systems* 13.3 (2012), pp. 1099–1115 (cited on pp. 17, 31, 33–36, 41–43).

[91] M. Milojevic and V. Rakocevic. "Distributed Road Traffic Congestion Quantification Using Cooperative VANETs." In: *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*. June 2014, pp. 203–210. doi: `10.1109/MedHocNet.2014.6849125` (cited on p. 71).

[92] G. Mitchell. "The Raspberry Pi single-board computer will revolutionise computer science teaching

ForAgainst

." In: *Engineering Technology* 7.3 (Apr. 2012), p. 26. issn: 1750-9637 (cited on p. 89).

[93] *MIXIM Framework website*. http://mixim.sourceforge.net/. July 2013 (cited on p. 63).

[94] *N4C Project*. http://www.n4c.eu. Feb. 2016 (cited on p. 11).

[95] M. Nakamura, T. Kitani, N. Shibata, K. Yasumoto, M. Ito, and W. Sun. "A Method for Improving Data Delivery Efficiency in Delay Tolerant VANET with Scheduled Routes of Cars." In: *2010 7th IEEE Consumer Communications and Networking Conference (CCNC)*. Vol. 9. 5. Las Vegas, NV, Jan. 2010, pp. 1–5. isbn: 9781424451760. doi: `10.1109/CCNC.2010.5421646` (cited on pp. 17, 28, 29, 41, 43).

[96] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Sanadidi. "Co-operative Downloading in Vehicular Ad-Hoc Wireless Networks." In: *Second Annual Conference on Wireless On-demand Network Systems and Services*. St. Moritz, Switzerland, 2005, pp. 32–41. isbn: 0-7695-2290-0. doi: `10.1109/WONS.2005.7` (cited on pp. 17, 38, 40).

[97] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. "The Broadcast Storm Problem in a Mobile Ad Hoc Network." In: *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking - MobiCom '99* (1999), pp. 151–162. ISSN: 1022-0038. DOI: 10.1145/313451.313525 (cited on p. 18).

[98] *Ns2 website*. Feb. 2016 (cited on p. 42).

[99] K. Obraczka, K. Viswanath, and G. Tsudik. "Flooding for Reliable Multicast in Multi-hop Ad Hoc Networks." In: *Wireless Networks* 7.6 (2001), pp. 627–634. DOI: 10.1023/A:1012323519059 (cited on p. 18).

[100] *Omnet++*. http://www.omnetpp.org/. Feb. 2016 (cited on pp. 42, 106).

[101] *OpenStreetMap website*. http://www.openstreetmap.org. Feb. 2016 (cited on pp. 75, 117).

[102] *OsmAnd website*. http://www.osmand.net. Feb. 2016 (cited on pp. 74, 75).

[103] S. Patra, S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Video-Based Overtaking Assistance Now A Realit." In: *Demonstrations of the 40th Annual IEEE Conference on Local Computer Networks (LCN-Demos 2015)*. Clearwater, USA (cited on p. 152).

[104] M. C. G. Paula, J. N. Isento, J. a. Dias, and J. J. P. C. Rodrigues. "A Real-world VDTN Testbed for Advanced Vehicular Services and Applications." In: *2011 IEEE 16th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (June 2011), pp. 16–20. DOI: 10.1109/CAMAD.2011.5941108 (cited on p. 48).

[105] C. E. Perkins and E. M. Royer. "Ad-hoc On-demand Distance Vector Routing." In: *Proceedings of WMCSA '99 Second IEEE Workshop on Mobile Computing Systems and Applications*. New Orleans, LA, USA, pp. 90–100. DOI: 10.1109/MCSA.1999.749281 (cited on p. 8).

[106] C. Perkins, S. Ratliff, and J. Dowden. "Dynamic MANET on-demand (DYMO) routing." In: *draft-ietf-manet-dymo-26 (work in progress)* (2013) (cited on p. 127).

[107] R. Protzmann, B. Schuunemann, and I. Radusch. "The Influences of Communication Models on the Simulated Effectiveness of V2X Applications."

In: *IEEE Communications Magazine* 49.11 (2011), pp. 149–155 (cited on p. 43).

[108] T. S. Rappaport. *Wireless Communications: Principles and Practice (2nd Edition)*. 2nd ed. Prentice Hall PTR, Jan. 2002. isbn: 9780130422323 (cited on p. 106).

[109] M. Sardari, F. Hendessi, and F. Fekri. "Infocast: A New Paradigm for Collaborative Content Distribution from Roadside Units to Vehicular Networks." In: *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. SECON'09. Rome, Italy: IEEE Press, 2009, pp. 271–279. isbn: 978-1-4244-2907-3 (cited on pp. 17, 18, 40).

[110] Savari Inc. *Savari Networks homepage*. http://www.savarinetworks.com/ (cited on p. 11).

[111] K. Scott and S. Burleigh. *Bundle Protocol Specification*. RFC 5050 (Experimental). Internet Engineering Task Force, Nov. 2007 (cited on p. 11).

[112] R. Shah, S. Roy, S. Jain, and W. Brunette. "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks." In: *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*. IEEE, 2003, pp. 30–41. isbn: 0-7803-7879-2. doi: 10.1109/SNPA.2003.1203354 (cited on p. 11).

[113] A. Sidera and S. Toumpis. "DTFR: A Geographic Routing Protocol for Wireless Delay Tolerant Networks." In: *2011 The 10th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. Favignana Island, Sicily, 2011, pp. 33–40. isbn: 9781457709005 (cited on pp. 17, 29, 30, 34, 35, 41).

[114] A. Skordylis and N. Trigoni. "Efficient Data Propagation in Traffic-monitoring Vehicular Networks." In: *IEEE Transactions on Intelligent Transportation Systems* 12.3 (Sept. 2011), pp. 680–694. issn: 1524-9050. doi: 10.1109/TITS.2011.2159857 (cited on pp. 17, 31, 32, 34, 41).

[115] T. Small and Z. J. Haas. "Resource and Performance Tradeoffs in Delay-tolerant Wireless Networks." In: *Proceeding of the 2005 ACM SIGCOMM Workshop on Delay-tolerant networking - WDTN '05*. New York, New York, USA: ACM Press, pp. 260–267. isbn: 1595930264. doi: 10.1145/1080139.1080144 (cited on p. 22).

[116] V. N. G. J. Soares, J. J. Rodrigues, and F. Farahmand. “GeoSpray: A Geographic Routing Protocol for Vehicular Delay-tolerant Networks.” In: *Inf. Fusion* 15 (Jan. 2011), pp. 102–113. ISSN: 15662535. DOI: `10.1016/j.inffus.2011.11.003` (cited on pp. 17, 23, 29, 30, 35, 41).

[117] C. Sommer and F. Dressler. “Progressing Toward Realistic Mobility Models in VANET Simulations.” In: *IEEE Communications Magazine* 46.11 (Nov. 2008), pp. 132–137. ISSN: 0163-6804. DOI: `10.1109/MCOM.2008.4689256` (cited on pp. 46, 128).

[118] C. Sommer, R. German, and F. Dressler. “Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis.” In: *IEEE Transactions on Mobile Computing* 10.1 (2011), pp. 3–15. ISSN: 1536-1233. DOI: `10.1109/TMC.2010.133` (cited on pp. 42, 46, 62, 65).

[119] C. Sommer, O. K. Tonguz, and F. Dressler. “Adaptive Beaconing for Delay-sensitive and Congestion-aware Traffic Information Systems.” In: *Vehicular Networking Conference (VNC), 2010 IEEE.* Dec. 2010, pp. 1–8. DOI: `10.1109/VNC.2010.5698242` (cited on p. 46).

[120] C. Sommer, O. K. Tonguz, and F. Dressler. “Traffic Information Systems: Efficient Message Dissemination Via Adaptive Beaconing.” In: *IEEE Communications Magazine* 49.5 (May 2011), pp. 173–179. ISSN: 0163-6804. DOI: `10.1109/MCOM.2011.5762815` (cited on p. 46).

[121] T. Spyropoulos, K. Psounis, and C. Raghavendra. “Single-copy Routing in Intermittently Connected Mobile Networks.” In: *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.* Pp. 235–244. ISBN: 0-7803-8796-1. DOI: `10.1109/SAHCN.2004.1381922` (cited on p. 22).

[122] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. “Spray and Wait: an Efficient Routing Scheme for Intermittently Connected Mobile Networks.” In: *Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking - WDTN ’05.* WDTN ’05. New York, New York, USA: ACM Press, pp. 252–259. ISBN: 1595930264. DOI: `10.1145/1080139.1080143` (cited on pp. 17, 22, 23, 35, 40, 44).

[123] M.-T. Sun, W.-c. Feng, T.-H. Lai, K. Yamada, H. Okada, and K. Fujimura. “GPS-based Message Broadcast for Adaptive Inter-vehicle Communications.” In: *IEEE-VTS Fall VTC 2000 52nd Vehicular Technology*

Conference. Boston, MA, 2000, 2685–2692 vol.6. ISBN: 0780365070 (cited on p. 19).

[124] E. Teramoto, M. Baba, and H. Mori. "Netstream: Traffic Simulator for Evaluating Traffic Information Systems." In: *ITSC '97, IEEE Conference on Intelligent Transportation System*. Boston, 1997, pp. 484–489. ISBN: 0780342690 (cited on p. 47).

[125] M. Thompson and P. Green. *Raspbian Home Page*. http://http://www.raspbian.org/. Feb. 2016 (cited on p. 89).

[126] O. Tonguz, N. Wisitpongphan, and B. Fan. "DV-CAST: A Distributed Vehicular Broadcast Protocol for Vehicular Ad Hoc Networks." In: *IEEE Wireless Communications* 17.2 (2010), pp. 47–57 (cited on pp. 17, 19, 20, 40, 43).

[127] S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A Map-based Sensor Data Delivery Protocol for Vehicular Networks." In: *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. Ayia Napa, Cyprus: IEEE, June 2012, pp. 1–8. ISBN: 978-1-4673-2039-9. DOI: `10.1109/MedHocNet.2012.6257118` (cited on pp. 51, 95, 105, 153).

[128] S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Assessing the Effectiveness of DTN Techniques Under Realistic Urban Environments." In: *38th Annual IEEE Conference on Local Computer Networks (LCN 2013)*. Sydney, Australia, Oct. 2013 (cited on pp. 17, 26, 37, 41–43, 51, 61, 62, 95, 115, 152).

[129] S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "Accelerating Vehicle Network Simulations in Urban Scenarios Through Caching." In: *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2014)*. July 2014, pp. 654–661. DOI: `10.1109/SPECTS.2014.6880007` (cited on p. 154).

[130] S. M. Tornell, C. T. Calafate, J.-C. Cano, and P. Manzoni. "DTN Protocols for Vehicular Networks: An Application Oriented Overview." In: *IEEE Communications Surveys & Tutorials* 17.2 (2015), pp. 868–887. ISSN: 1553-877X. DOI: `10.1109/COMST.2014.2375340` (cited on pp. 7, 151).

[131] S. M. Tornell, C. T. Calafate, J.-C. Cano, P. Manzoni, M. Fogue, and F. J. Martinez. "Implementing and Testing a Driving Safety Application for Smartphones Based on the eMDR Protocol." In: *2012 IFIP Wireless*

Days. IEEE, Nov. 2012, pp. 1–3. ISBN: 978-1-4673-4404-3. DOI: `10.1109/WD.2012.6402816` (cited on p. 153).

[132] S. M. Tornell, C. T. Calafate, J.-C. Cano, P. Manzoni, M. Fogue, and F. J. Martinez. “Evaluating the Feasibility of Using Smartphones for ITS Safety Applications.” In: *VTC Spring 2013 IEEE Vehicular Technology Conference*. 2013. ISBN: 9781467363372 (cited on pp. 73, 145, 153).

[133] S. M. Tornell, E. Hernández-Orallo, C. T. Calafate, J.-C. Cano, P. Manzoni, and E. Hernández. “An Analytical Evaluation of a Map-based Sensor-data Delivery Protocol for VANETs.” In: *14th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM 2013)*. Madrid, Spain, June 2013. ISBN: 9781467358286 (cited on pp. 51, 95, 153).

[134] S. M. Tornell, T. Kärkkäinen, J. Ott, C. Calafate, J.-C. Cano, and P. Manzoni. “Simplifying The In-vehicle Connectivity for ITS Applications.” In: *MOBIQUITOUS 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2015. DOI: `10.4108/eai.22-7-2015.2260058` (cited on pp. 83, 129, 152).

[135] S. M. Tornell, S. Patra, C. T. Calafate, J.-C. Cano, and P. Manzoni. “GRCBox: Extending Smartphone Connectivity in Vehicular Networks.” In: *International Journal of Distributed Sensor Networks* 2015.Article ID 478064 (2015), p. 13. DOI: `10.1155/2015/478064` (cited on pp. 83, 129, 151).

[136] S. M. Tornell, S. Patra, C. T. Calafate, J.-C. Cano, and P. Manzoni. “A novel On-board Unit to Accelerate the Penetration of ITS Services.” In: *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC 2016)*. Las Vegas, USA, Jan. 2016, pp. 474–479 (cited on pp. 129, 152).

[137] O. Trullols-Cruces, M. Fiore, and J. Barcelo-Ordinas. “Cooperative Download in Vehicular Environments.” In: *IEEE Transactions on Mobile Computing* 11.4 (Apr. 2012), pp. 663–678. ISSN: 1536-1233. DOI: `10.1109/TMC.2011.100` (cited on pp. 17, 31, 33, 38, 41).

[138] Y. Tseng, S. Ni, and E. Shih. “Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network.” In: *IEEE Transactions on Computers* 52.5 (2003), pp. 545–557 (cited on p. 18).

[139] A. Vahdat and D. Becker. “Epidemic Routing for Partially Connected Ad Hoc Networks.” In: *Technical Report CS-200006*. Duke University, Apr. 2000 (cited on pp. 13, 17, 18, 22, 35, 40, 59, 71, 95, 109).

[140] W. Viriyasitavat, F. Bai, and O. Tonguz. “UV-CAST: An urban vehicular broadcast protocol.” In: *2010 IEEE Vehicular Networking Conference (VNC)*. Vol. 49. 11. Jersey City, NJ, 2010, pp. 25–32. isbn: 9781424495252 (cited on pp. 17, 19, 20, 40).

[141] H. Wang, Y. Zhu, and Q. Zhang. “Compressive Sensing Based Monitoring with Vehicular Networks.” In: *2013 Proceedings IEEE INFOCOM* (), pp. 2823–2831. doi: `10.1109/INFCOM.2013.6567092` (cited on pp. 17, 37, 41).

[142] X. Wang and C. Song. “Distributed Real-Time Data Traffic Statistics Assisted Routing Protocol for Vehicular Networks.” In: *2010 IEEE 16th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, Dec. 2010, pp. 863–867. isbn: 978-1-4244-9727-0. doi: `10.1109/ICPADS.2010.57` (cited on pp. 17, 31, 32, 41, 77).

[143] F. Warthman. *Delay Tolerant Networking: A Tutorial*. 2003 (cited on p. 12).

[144] H. Wen, F. Ren, J. Liu, and C. Lin. “A Storage-Friendly Routing Scheme in Intermittently Connected Mobile Network.” In: *IEEE Transactions on Vehicular Technology* 60.3 (2011), pp. 1138–1149 (cited on pp. 17, 29, 30, 35, 41).

[145] J. Xue, J. Li, Y. Cao, and J. Fang. “Advanced PROPHET Routing in Delay Tolerant Network.” In: *International Conference on Communication Software and Networks, 2009. ICCSN ’09.* C. Macau, 2009, pp. 411–413. isbn: 978-0-7695-3522-7. doi: `10.1109/ICCSN.2009.44` (cited on pp. 17, 24, 25, 40).

[146] J. Xue, X. Fan, Y. Cao, J. Fang, and J. Li. “Spray and Wait Routing Based on Average Delivery Probability in Delay Tolerant Network.” In: *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*. Vol. 2. Wuhan, Hubei, Apr. 2009, pp. 500–502. isbn: 978-0-7695-3610-1. doi: `10.1109/NSWCTC.2009.284` (cited on pp. 17, 23, 29, 35, 40).

[147] J. Yoon, M. Liu, and B. Noble. “Random Waypoint Considered Harmful.” In: *22nd Annual Joint Conference of the IEEE Computer and Communica-*

tions, INFOCOM 2003. Vol. 2. C. 2003, pp. 1312–1321. ISBN: 0780377532. DOI: 10.1109/INFCOM.2003.1208967 (cited on pp. 44, 47).

[148] D. Yu and Y.-B. Ko. "FFRDV: Fastest-ferry Routing in DTN-enabled Vehicular Ad Hoc Networks." In: *11th International Conference on Advanced Communication Technology, 2009. ICACT 2009*. Vol. 02. Phoenix Park, Feb. 2009, pp. 1410–1414. ISBN: 978-89-5519-138-7 (cited on pp. 17, 19, 40, 47).

[149] D. Zhang, H. Huang, M. Chen, and X. Liao. "Empirical Study on Taxi GPS Traces for Vehicular Ad Hoc Networks." In: *2012 IEEE International Conference on Communications (ICC)* (June 2012), pp. 581–585. DOI: 10.1109/ICC.2012.6364248 (cited on p. 26).

[150] X. Zhang, G. Neglia, J. Kurose, and D. Towsley. "Performance Modeling of Epidemic Routing." In: *Computer Networks* 51.10 (July 2007), pp. 2867–2891. ISSN: 13891286. DOI: 10.1016/j.comnet.2006.11.028 (cited on pp. 22, 97–99).

[151] J. Zhao and G. Cao. "VADD: Vehicle-assisted Data Delivery in Vehicular Ad Hoc Networks." In: *IEEE Transactions on Vehicular Technology* 57.3 (2008), pp. 1910–1922 (cited on pp. 17, 31, 40).

[152] H. Zhu, M. Dong, S. Chang, Y. Zhu, M. Li, and X. Sherman Shen. "ZOOM: Scaling the Mobility for Fast Opportunistic Forwarding in Vehicular Networks." In: *2013 Proceedings IEEE INFOCOM* (Apr. 2013), pp. 2832–2840. DOI: 10.1109/INFCOM.2013.6567093 (cited on pp. 17, 24–26, 41).

[153] H. Zhu, L. Fu, G. Xue, Y. Zhu, M. Li, and L. M. Ni. "Recognizing Exponential Inter-contact Time in VANETs." In: *Proceedings of the 29th conference on Information communications*. INFOCOM'10. Piscataway, NJ, USA: IEEE, 2010, pp. 101–105 (cited on p. 96).

[154] Y. Zhu, Y. Qiu, Y. Wu, and B. Li. "On Adaptive Routing in Urban Vehicular Networks." In: *IEEE GLOBECOM 2012, Global Telecommunications Conference* (), pp. 1611–1616. DOI: 10.1109/GLOCOM.2012.6503341 (cited on pp. 17, 29, 31, 41).