Bruno Miguel Pinho Tavares

Transmissão Oportunística de Informação em Redes Veiculares

Universidade de Aveiro Electrónica, Telecomunicações e Informática, 2013

Bruno Tavares

Transmissão Oportunística de Informação em Redes Veiculares

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica da Professora Doutora Susana Sargento, Professora Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Professor João Barros do Departamento e Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

o júri / the jury

presidente / president

Prof. Doutor Paulo Miguel Nepomuceno Pereira Monteiro
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutora Susana Isabel Barreto de Miranda Sargento
Professora auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Doutora Teresa Maria Sá Ferreira Vazão Vasques
Professora Associada do Instituto Superior Técnico, Universidade Técnica de Lisboa

agradecimentos

Agradeço primeiramente aos meus pais por todo o apoio e esforço para que fosse possível ter uma formação superior, já que sem eles nada disto seria possível.

Agradeço a todos os meus amigos que sempre me apoiaram e a todos os colegas de laboratório.

Um agradecimento especial ao Filipe Neves, Carlos Ameixieira, Luís Coelho, Tomé Gomes e Romeu Monteiro pela disponibilidade, ajuda e companheirismo.

Um agradecimento especial ao João Gonçalves e Luís Guedes pela ajuda e disponibilidade na realização dos testes experimentais.

Resumo

A área de comunicações sem fios tem sido alvo de vários projectos de investigação ao longo dos últimos anos. A constante necessidade de "comunicar" nos mais diversos ambientes, incluindo os de alta mobilidade, requerem comunicações sem fios. Neste contexto foram criadas as Vehicular Ad hoc Networks (VANETs), que são redes baseadas no conceito Ad Hoc, permitindo a comunicação entre veículos e entre veículos e infraestruturas fixas, que aumentam a conectividade da rede. As VANETs, devido às suas carecterísticas, apresentam desafios, tais como intervalos curtos de conectividade em cenários onde a densidade de nós é reduzida, e também situações onde não existe comunicação durante longos períodos de tempo. O trabalho da presente dissertação tem como objectivo possibilitar o envio de informação não urgente de forma oportunística, rentabilizando todos os recursos da rede. O conceito de Delay/Disrupt Tolerant Network (DTN) é deste modo abordado como solução para os desafios descritos anteriormente.

Desta forma, duas implementações de DTN são estudadas e testadas para serem incorporadas em dispositivos Wireless Access in Vehicular Environments (WAVE), comunicando através da norma IEEE 802.11p para redes veiculares. Depois de vários testes realizados, o IBR-DTN mostrou ser a implementação mais robusta e mais "leve" para ser utilizada em sistemas embutidos, como é o caso das On-Board Units (OBUs) utilizadas nas VANETs. Vários problemas de implementação foram detectados e corrigidos para ser possível integrar o IBR-DTN de forma funcional num ambiente veicular real.

O conjunto de testes realizados consistiu em: dois cenários em laboratório para melhor perceber o funcionamento do IBR-DTN; e três cenários numa testbed real com veículos e estações fixas. Os dois cenários testados em laboratório permitem concluir o bom funcionamento do processo de fragmentação para diferentes tempos de ligação e diferentes tamanhos de ficheiros, onde as ligações entre os nós eram interrompidas periodicamente. Os cenários testados na testbed real mostram que o IBR-DTN funciona sem problemas usando várias velocidades com estações fixas e um carro, e com dois carros dirigindo-se em sentido contrário um ao outro. Permitem também concluir que, aumentando a velocidade, diminui o tempo de contacto entre os nós traduzindo-se num maior número de fragmentos para enviar um determinado ficheiro. Os resultados mostram também que o IBR-DTN tem uma boa resposta em ambientes de alta mobilidade como as VANETs.

Abstract

The area of wireless communications has been the subject of several research projects over the last years. The persistent need to "communicate" in various environments, including high mobility, make the use of wireless-based communications a strong requirement. In this context, VANETs were created, which are networks based on Ad Hoc concept allowing the communication between vehicles and between vehicles and fixed infrastructures, that increase network's connectivity. VANETs, due to their characteristics, introduce challenges such as shorts connectivity intervals in sparse networks, and also in situations where connectivity can be down for long periods of time. The work of this Dissertation aims to send non-urgent information in a opportunistic way, maximizing the network resources. The DTN's concept is thus addressed as a solution to the previous described challenges.

Two DTN implementations are studied and tested to be incorporated in WAVE devices communicating using the standard IEEE 802.11p for vehicular networks. After several tests, IBR-DTN proved to be the most robust and "light" implementation to be used in embedded systems, such as the OBUs used in VANETs. Several implementation problems were detected, through several tests, and corrected to be possible to provide the functional integration of IBR-DTN in a real vehicular environment.

The set of tests consisted in: two scenarios in the laboratory environment, to better understand IBR-DTN's operation; and three scenarios in a real testbed with vehicles and fixed stations. The two scenarios tested in laboratory allowed to conclude the good performance of fragmentation process for different connection time intervals and different file sizes, where the connections between the nodes were periodically interrupted. The scenarios performed on the real testbed show that IBR-DTN operates without problems for various velocities using fixed infrastructures and a car, and two cars moving towards each other in different ways. It can be also concluded that, increasing the velocity, the contact time between nodes decreases, contributing to a larger number of fragments needed to send a specific file. The results show also that IBR-DTN has a good response in high mobility environments like VANETs.

Contents

List of Figures

List of Tables

Acronyms

4G Fourth Generation of cellular Wireless Standards

ACK Acknowledge

ADU Application Data Unit

AP Access Point

API Application Programming Interface

AU Application Unit

BER Bit-Error Rate

C2C-CC Car to Car Communication Consortium

CCH Control Channel

CLA Convergence Layer Adapter

CPU Central Processing Unit

DNS Domain Name System

DSRC Dedicated Short Range Communications

DTLSR Delay Tolerant Link State Routing

DTN Delay/Disrupt Tolerant Network

DTNRG Delay-Tolerant Networking Research Group

EID Endpoint Identifier

GPRS General Packet Radio Service

GPS Global Positioning System

GSM Global System for Mobile Communications

HTTP Hypertext Transfer Protocol

ID Identifier

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

ION Interplanetary Overlay Network

IP Internet Protocol

IPC Inter-Process Communication

IPN Interplanetary Networking

IPND IP Neighbour Discovery

IRC Internet Relay Chat

IRTF Internet Research Task Force

IT Instituto de Telecomunicações

ITS Intelligent Transportation Systems

LS Link State

LSA Link State Announcement

LTP Licklider Transmission Protocol

MAC Media Access Control

MANET Mobile Ad-hoc Network

MEED Minimizing the Estimated Expected Delay

OBU On-Board Unit

OEM Original Equipment Manufacturers

OS Operating System

PAN Personal Area Network

PC Personal Computer

PDA Personal Digital Assistant

PDU Protocol Data Unit

PROPHET Probabilistic ROuting Protocol using History of Encounters and Transitivity

RSU Road Side Unit

RAM Random Access Memory

SDNV Self-Delimited Numerical Value

SCH Service Channel

SSH Secure Shell

Tcl Tool Command Language

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

UDP User Datagram Protocol

UMTS Universal Mobile Telecommunications System

URI Universal Resource Identifiers

USB Universal Serial Bus

VANET Vehicular Ad hoc Network

VDTN Vehicular Delay/Disrupt Tolerant Network

VGA Video Graphics Array

V2I Vehicle to Infrastructure

V2R Vehicle to Road

V2V Vehicle to Vehicle

WAVE Wireless Access in Vehicular Environments

Wi-Fi IEEE 802.11 a/g/n

WiMAX Worldwide Interoperability for Microwave Access

WLAN Wireless Local Area Network

WSM WAVE Short Message

ZOR Zone-of-Relevance

Chapter 1

Introduction

The main goal of the work developed under this Dissertation is to introduce and analyse the concept of "DTN" in vehicular networks in real environments.

Along this chapter, it will be presented to the reader the motivation to realize this work, the proposed objectives and the document structure.

1.1 Motivation

The past years have been extremely important to wireless communications and its evolution continues to increase at a fast pace. It is expected that, in the following years, this increase will be even higher. On the other side, vehicles industry continues to grow as well providing many services like velocity and control systems, board computers and navigation systems (GPS), etc.

Due to this evolution, it is expected that, in a near future, the vehicles can communicate with each other giving more safety to the drivers as well as comfort, creating a vehicular ad hoc network. The communication can also be possible between vehicles and infrastructure along the road considering the new technology IEEE 802.11p for vehicular communication, and with IEEE 802.11 a/g/n (Wi-Fi) hotspots.

This type of communication has its problems due to the high mobility of nodes. In some cases the density of vehicles in some area can be high, where the dissemination of information is made easily; but in other situations, where the node's density is very low, the transfer of information can be a problem. The mobility of nodes and its different velocities and directions leads to frequent loss of connection, and so, the information is not always delivered successfully.

In order to deliver information with the maximum possible success in environments like VANETs, the use of mechanisms that tolerate the lack of connectivity and high delays are of extreme interest. The introduction of this concept in these networks makes possible the transmission of partitions of information when the contacts between nodes are possible until the complete information is delivered. This type of network is desirable to deliver non-urgent information such as music, tourist guides of the city, e-mail, and they are transmitted opportunistically.

This type of network, which is capable to send information opportunistically, exists and it is called Delay/Disrupt Tolerant Network. It was designed to work on deep space environments, where the delays are high and the loss of connectivity is frequent. The evolution in

mobile ad-hoc networks contributed to the increase of interest to use the previous concept in order to augment the network's efficiency. Several projects were created using the concept of DTN in mobile ad-hoc networks, described in [1], using the standard IEEE 802.11a/g/n. None of these projects addressed real vehicular scenarios, and the vehicular standard technology IEEE 802.11p was not used as well. Tests using the vehicular standard of communication IEEE 802.11p in real vehicular environments are crucial to understand the limitations and advantages of DTN in VANETs.

1.2 Objectives

This Dissertation has the objective of studying and testing various implementations of DTNs in order to use it in a vehicular environment, using the vehicular communication devices with IEEE 802.11p technology and their communication characteristics. The use of this implementation will be useful for the opportunistically transfer of information that is not safety-related or critical.

The objectives of this work are the following:

- Evaluation and study of DTN mechanisms and protocols;
- Evaluation and study of DTN implementations;
- Installation of the implementations in the vehicular equipment;
- Evaluation and tests of the implementation with the vehicular devices;
- Debug and correction of problems and missing functionalities present in the implementation;
- Development of software to analyse the data of various tests;
- Tests on laboratory to analyse various characteristics like throughput, time of transfers, size of fragments and number of fragments;
- Tests on the road to analyse the same characteristics but in a real vehicular environment, and to evaluate the feasibility of these approaches in a vehicular environment.

The work performed in this Dissertation will be published in an international conference.

1.3 Dissertation Structure

The present document has the following organisation:

- Chapter 1 starts this Dissertation, where the motivation, objectives and document organization are presented.
- In Chapter 2, an overview of the main concepts of VANETs is presented such as VANET's architectures, its characteristics, challenges, etc. The concepts of DTNs are also introduced, specifying its main characteristics, applications and architecture. The last part of the chapter is reserved to present the work/investigation done in the area of Vehicular Delay/Disrupt Tolerant Networks.

- Chapter 3 gives to the reader the knowledge about the DTN implementations used, their architectures, applications and features.
- Chapter 4 presents the development made to realize all the experimental tests, as well as the problems found in the used DTNs implementations. It will be presented the problems and the corrections made to the implementation.
- Chapter 5 provides the several elaborated scenarios and its results, evaluating and discussing them.
- Chapter 6 is reserved to a conclusion of the developed work, and to improvements that can be implemented, as future work, in order to contribute to the continuous development of the work.

The work of this dissertation is being submitted to a publication in an International Conference: Opportunistic transfer of information in VANETs.

1.4 Contributions

As a major contribution to the research on vehicular networks, the work developed under this Master Dissertation results into one scientific paper.

The paper refers to the study of DTNs in vehicular environments, where real scenarios were deployed using the standard IEEE 802.11p for communication.

In order to report the new results and disseminate the information to the large community of scientists, this paper will be submitted to a scientific journal.

1.5 Summary

This chapter presented the motivations of the Dissertation as well as its main objectives, and a brief information about the document structure.

The next chapter will present to the reader a general study about VANETs, DTNs, and the work that can be related to the VDTNs.

Chapter 2

State of Art

2.1 Introduction

This chapter presents a summary of the work developed in the Dissertation area, using books and papers published. The basic concepts will be presented, familiarizing the reader with the subject.

Therefore, section 2.2 presents the concepts of VANET, as well as its definition, architectures, challenges and routing. Section 2.3 follows the previous one, introducing the concept of DTN, exploring its applications, architecture and routing.

The last section, 2.4, focus on what was already made using the concept of DTN in VANETs, VDTNs, making reference to several projects and potential applications.

2.2 Vehicular Networks

The evolution of technology allowed the improvement in the automotive area, giving the passengers more safety and comfort. The speed control and the parking sensors are an example of this improvement. Usually, these mechanisms make use of sensors and actuators to collect information from the exterior.

The fast growing and evolution in wireless communications made possible the idea of having a big, sparse and dynamic network composed by vehicles and infrastructures, all communicating with each other. This type of network is called VANET, which is a decentralized wireless network that does not need to rely on a pre-existing infrastructure, like routers or Access Points (APs); each node is responsible for forwarding data to other nodes based on network connectivity. This allows the communication between vehicles and between vehicles and infrastructures along the road, RSUs.

The main goal of this new communication system is to provide a variable set of services for passengers and drivers including safety, information and entertainment services.

2.2.1 Definition

As it was mentioned before, a vehicular network is a wireless network that emerged due to the evolution of wireless networks and automotive industry. This network is formed by moving vehicles equipped with wireless interfaces (homogeneous or heterogeneous technologies). VANETs are considered as one of the ad hoc network real-life applications, allowing

communication between close vehicles and vehicles with fixed infrastructures along the road, RSUs. The vehicles composing this network can either be private or belong to public transportation, like buses or police cars while the fixed equipment, along the road, might belong to the government, to private network operators or service providers.

The potential of VANETs are seducing the research community as well as the automotive industry. In North America, a Dedicated Short Range Communications (DSRC) system appeared, where 75 MHz of spectrum was reserved by the U.S. FCC (Federal Communication Commission) in 2003 for this type of communication. In Europe this new technology caught a lot of attention and the Car to Car Communication Consortium (C2C-CC) was initiated by car manufacturers and automotive Original Equipment Manufacturers (OEM), to increase the road traffic safety and efficiency, using this inter-vehicle communication. Institute of Electrical and Electronics Engineers (IEEE) also advanced with IEEE 1609 family of standards for WAVE.

2.2.2 Architecture

According to Figure 2.1 there are three main types of architectures.

- `pure Vehicle to Vehicle (V2V) ad-hoc network`: the vehicles communicate without any support of external elements. This means that they work as routers and the traffic is routed by multiple hops. This type is the simplest one, due to the absence of external infrastructure, but the great disadvantage is the network connectivity depending on the density and car's mobility.

- `Infrastructured Vehicle to Infrastructure (V2I)`: in this case the connectivity problem from the previous case is overpassed because of the infrastructure present along the streets. These nodes act like APs and centralize all the traffic from the network, being intermediates in the communication. The main advantage of this architecture is the possibility of connecting to other types of networks, e.g. like Internet. In the meanwhile, the connectivity is only granted if the number of fixed nodes is large, making the network expensive.

- `hybrid Vehicle to Road (V2R)`: this is the intermediate solution between the previous two because it combines the two of them. The number of RSUs is minimal just to increase the connectivity and to provide services. The option of communication V2V is possible as well. It is usual to refer to VANETs as the combination of the two types of architectures.

A reference architecture for VANETs was proposed within the C2C-CC, which makes distinction between three domains, as it is possible to see in figure 2.2: in-vehicle, ad-hoc and infrastructure domain:

- `in-vehicle domain`: consists in a local network inside each vehicle. This network is composed by two types of units: an OBU and one or more Application Units (AUs). The OBU is the device responsible for the communication (can communicate via wireless or wired), while the AU is the device that executes applications making use of the OBU's communication capabilities. An AU can be an integrated part of vehicle always connected to the OBU; it can also be a portable device (laptop or Personal Digital Assistant (PDA)) which can be attached and detached from the OBU. It is also possible to have the two units together in a single device.

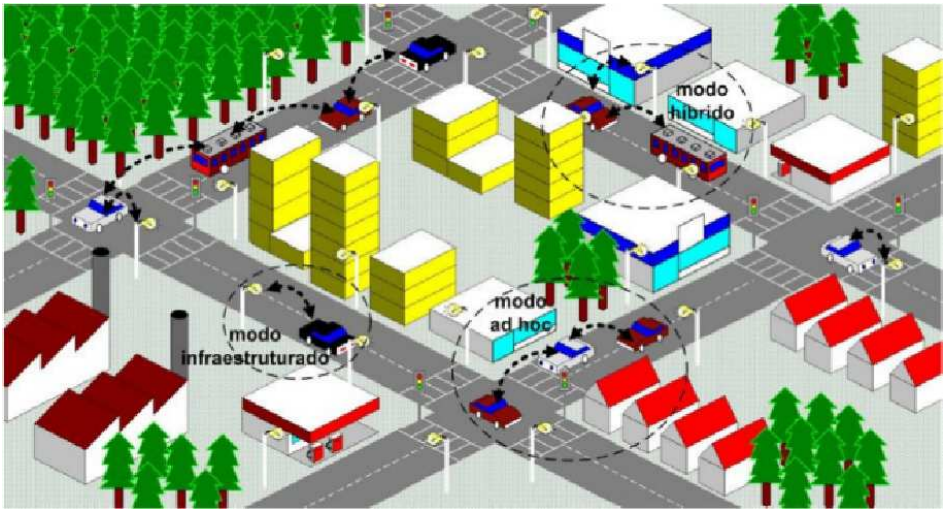

Figure 2.1: VANETs Architecture [2]

- `ad-hoc domain`: this network is formed by the vehicles equipped with OBUs (mobile nodes) and RSUs, that are stationary nodes along the road. The OBUs from the different vehicles form the Mobile Ad-hoc Network (MANET) where the OBU is equipped with communication devices, including, in the simplest form, a short-range wireless communication device for road safety. RSUs can be connected to infrastructure networks making possible the Internet access. The primary role of RSUs is to improve road safety communicating with each other directly or via multihop. The mobile nodes and the stationary nodes form this way the ad-hoc domain.

- `infrastructure domain`: in this domain there are two types of infrastructures, IEEE 802.11p RSUs and Wi-Fi hotspots. OBUs can be connected to RSUs and, consequently, be connected to the Internet. It is also possible that an OBU can connect directly to Wi-Fi hotspots to connect to the Internet. If RSUs and hotspots are not in the range of a vehicle, the OBU can make use of its communication capabilities of cellular networks (Global System for Mobile Communications (GSM), General Packet Radio Service (GPRS), Universal Mobile Telecommunications System (UMTS), Worldwide Interoperability for Microwave Access (WiMAX) and Fourth Generation of cellular Wireless Standards (4G)), if they are integrated in the device.

2.2.3 Technical Challenges

The special characteristics of VANETs bring some challenges regarding the communication between vehicles. These technical issues need to be resolved in order to provide useful services for drivers and passengers. For example, the scalability and interoperability are two issues of extreme importance that have to be satisfied. The mechanisms and protocols used should be scalable to numerous vehicles and interoperable with different wireless technologies. A vehicle should be enabled to communicate, in addition to other vehicles and RSUs, with hotspots that can be part of the network.

The following subsections will focus on some important technical challenges of vehicular networks [3].

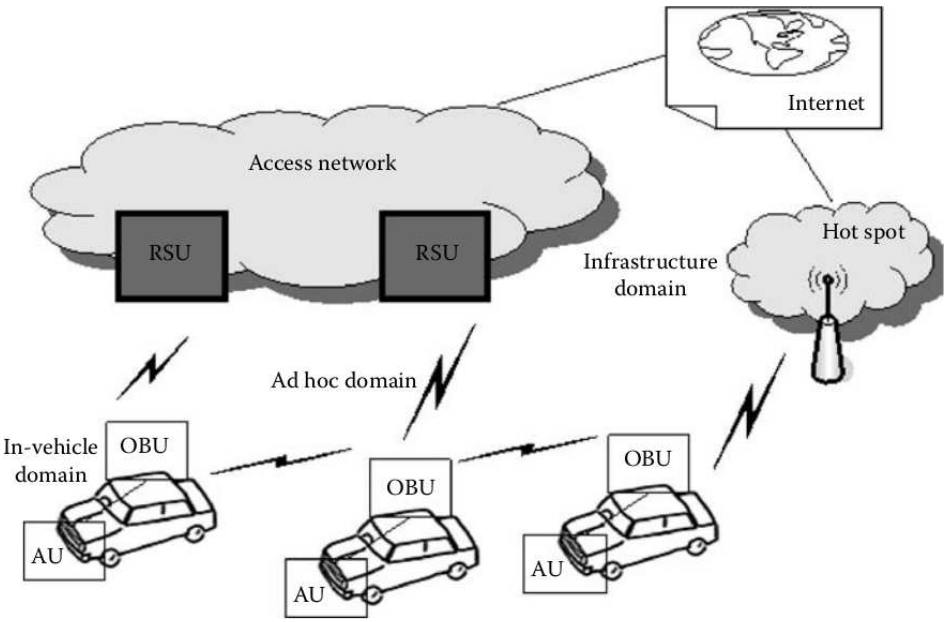

Figure 2.2: VANETs C2C-CC Reference Architecture [3]

2.2.3.1 Reliable Communication and MAC Protocols

The multihop is highly important in vehicular communications, which can extend the operator fixed infrastructure providing, this way, virtual infrastructure among the moving vehicles. Its major challenge is regarding the reliability of communication, so efficient Media Access Control (MAC) protocols need to be in place while adapting to the extreme dynamic environment of VANETs, considering message priority of certain applications, like accident warnings. Because of the short period of connectivity, the association needs to be fast and the latency low in order to guarantee:

- service reliability for safety-related applications while taking into consideration the time sensitivity during message transfer.
- the quality and continuity of service for other types of applications, non-safety ones.

Additionally, MAC protocols should guarantee reliable communications between different technologies (Wi-Fi and GSM).

2.2.3.2 Routing and Dissemination

VANETs differ from conventional ad-hoc wireless networks, not only by the rapid changes in wireless link connections, but also because they have to deal with different types of network densities [4]. The density in VANETs is dependent of the local and time; for example, during the evening the number of vehicles on a rural freeway is lower comparing to the rush hour traffic in the morning and late afternoon on a urban freeway. It is expected that VANETs can handle a large range of applications, which can go from safety to leisure ones. Thereafter, dissemination and routing algorithms need to be efficient and adaptive, according to characteristics of the network and applications, allowing different transmission rates and priorities

corresponding to application demands. Most of vehicular network research focuses on the analysis of routing algorithms to handle the broadcast problem [5][6] in a very dense network, assuming that all nodes are well connected. Since this innovative technology is somehow recent, its penetration is considerably weak which causes dependency from infrastructure support for large scale deployment. It is expected that, in the future, the penetration will be much higher with a great number of vehicles equipped and ready to communicate with each other, reducing the number of RSUs needed. Consequently there is the need to develop reliable and efficient routing protocols considering the disconnected network problem and all the diverse network topologies.

The dissemination of messages should depend on the network density as well as the application type. Safety-related applications should be, most of times, of broadcast-type to guarantee the message propagation to the required set of vehicles, but without causing a broadcast storm. In other application types, like leisure and informative ones, it is more convenient to use unicast or multicast to transfer messages.

2.2.3.3 Security

Security in vehicular communication has a large importance regarding the future deployment and application of VANETs. The development and acceptance of services is highly dependent on security and privacy, and can not be compromised by the easiness of service discovery protocols. Due to the improvement of these discovery protocols, the users may use services in foreign networks creating several of problems to themselves and to other network users. So, it is important to propose and create solutions that ensure security in communication between users, as well as authorized and secure service access to them.

In order to improve the vehicular networks access, solutions should utilize:

- The ad-hoc multihop communication and authentication concepts, allowing secure communication and extension of infrastructure, with a minimum number of fixed stations, reducing costs to the network operator.
- The distributed-based authentication: appropriate security architectures should be implemented providing communication between vehicles and allowing diverse service access.

Authentication optimization is extremely important to be studied for both infrastructure-based and infrastructure-less communications to make easy the reauthentication process which may happen during vehicle's mobility.

Each node is important to the security of communication and service delivery, since it can threat it, so its consideration is no less important. Since the vehicular networks are open and dynamic, node's cooperation is an aspect that should be taken into account to allow successful communication among vehicles. Nodes can act selfishly by not forwarding data to others to save bandwidth and power, or just because of security and priority matters. Then, appropriate mechanisms should be implemented to detect selfishness to force node's cooperation.

2.2.3.4 IP Configuration and Mobility Management

The V2I architecture can allow vehicular Internet access as well as Internet-related services to users. Nevertheless, there are two technical challenges under this situation: Internet

Protocol (IP) address configuration and mobility management. These two challenges can compromise the service continuity and its quality. VANET characteristics demands an automatic and distributed IP configuration. Until now it does not exist a standard for IP autoconfiguration in ad-hoc networks, which can be a problem with relative complexity in VANETs. There are works in progress by some standardization bodies to try to resolve this issue. In addition to Internet Engineering Task Force (IETF) efforts through *Autoconf WG* [7] for developing IPv6 solutions to ad-hoc networks, including VANETs, all international committees are defining architectures for vehicular communication including an IPv6 stack in their protocol stacks.

Regarding the mobility management, this is a critical problem to non-safety applications where the dissemination of information is not made in broadcast. The lack of mobility management mechanisms turns difficult the commercialization of services in VANETs and loses the advantage of V2I architecture, because all Internet-related services would not guarantee quality of service and its continuity.

2.2.4 Concepts

The understanding of vehicular network applications is highly dependent on the basic concepts of this technology. In this section, concepts like basic equipment, access network technologies, addressing and data dissemination will be exploited to give the reader a better framework to understand the further work [8].

2.2.4.1 On-Board Equipment

To be a part of a VANET, vehicles need to be equipped with an OBU. It is assumed by various articles the following hardware present:

- a Central Processing Unit (CPU) which is responsible for the implementation of applications and communication protocols.
- a wireless transceiver responsible to transmit and receive data to and from vehicles and infrastructures nearby.
- a Global Positioning System (GPS) receiver to give a accurate positioning and time synchronization information.
- sensors that can measure the necessary parameters to be transmitted.
- input/output interface to allow human interaction with the system.

2.2.4.2 Network Access Technologies

Nowadays, there are several communication standards that, eventually, can be used as access networks for VANET applications. For each standard there is a set of advantages and disadvantages depending on the specific scenario, and also depending on the application type. The constant mobility of nodes brings problems to the radio channel due to the fast and slow fading.

The most used Wireless Local Area Network (WLAN), IEEE 802.11, was the predictable choice to the common use as access technology. Nevertheless, results in [9] prove, with real propagation models, that the Bit-Error Rate (BER) of this access technology can be very

high, imposing significant difficulties in upper layers. This problem is due to the design of 802.11a because this norm was optimized for local area networks with very low or no mobility at all. The high mobility of VANET nodes make this norm extremely unsuited, so a new standard was created specifically for vehicular communication, IEEE 802.11p [10], based on the standard 802.11. This new standard improves communication between vehicles and between vehicles and infrastructures fixed along the road.

This technology uses the band of 5.9 GHZ (5.85-5.925 GHz) licensed by Intelligent Transportation Systems (ITS). This standard has better responses against fading, increasing the reach with less losses.

2.2.4.3 Addressing

According to [8], VANETs behave most of times like ad-hoc networks. Hence, the addressing schemes of ad-hoc networks can be used in vehicular networks. Two types of addressing can be used:

- `Fixed adressing`: in this case, a fixed address is assigned to each node when it joins the network. This address is used while the node is part of the network. The Internet uses, commonly, this mechanism for addressing, except the mobile IP [11].

- `Geographical adressing`: the nodes are characterized by their geographical position, which means that the address given to each node is not fixed as in the previous case. The mobility of the node changes its address. It is possible to select a group of target vehicles with additional attributes like the direction of vehicle's movement, the identification of the road/street, the vehicle's type, velocity and size of the vehicle, and the driver's appearance.

2.2.4.4 Data Dissemination

Some VANET applications aim to spread data between the nodes in the network, vehicles and RSUs. There are important concepts regarding the data dissemination, which are described below.

Data spread can be single-hop or multihop as it is shown in Figure 2.3. Figure 2.3(a) illustrates a single-hop dissemination which is usually implemented with broadcast on the MAC layer, where it is shown the vehicle A spreading messages to the vehicles on its transmission range; therefore, the vehicle B never receives the data that A sends. This mechanism can be used when RSUs are responsible for the communication control. Multihop is in the top list to have a good functional VANET. The data is transmitted in several hops, where the vehicles in the middle act as relays to the destination one as it is shown in Figure 2.3(b), where vehicle C can relay the message to vehicles that are not in range's transmission of A (e.g., vehicle B). This means that multihop systems need a network layer that supports multihop routing. Some variants have been proposed as well, where, giving an example, data is transmitted via multihop to the closest RSU and then transmitted to the relevant vehicles using single-hop.

The data can be disseminated in three ways: unicast, multicast or broadcast.

In unicast, there is only one receiver of the data, where the scheme of addressing is not relevant, which means that it can be used fixed addressing as geographical addressing.

The multicast mode is defined by one group of receivers. Normally the sender does not know the exact number of receivers; it only knows the specific destination group to which

the data needs to be sent. A good example for the multicast use is an application regarding traffic safety, where a specific group of vehicles (multicast group) need to receive information about an accident. This group is formed by the vehicles in that accident's area and to vehicles moving towards that area.

Broadcast is used when the data is supposed to be sent to all vehicles, but because a VANET can be extremely sparse, the broadcast is performed on a specific area, called Zone-of-Relevance (ZOR) [8].

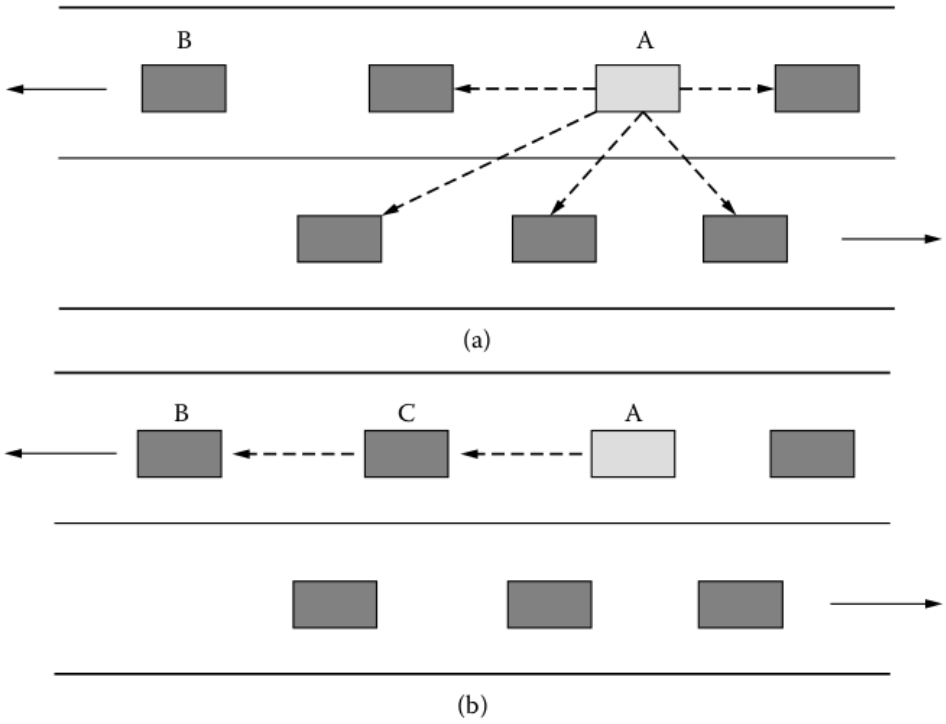

Figure 2.3: Single-hop (a) and multihop (b) data dissemination [8]

2.2.5 Routing Protocols

Routing in VANETs is, most probably, one of the key research issues in VANETs, since many applications require multihop transmissions using vehicles as relays to reach the destination. These networks are extremely unique due to their properties, like the possible large number of vehicles, its dynamics and the quick change of vehicle's density in the network. The large number of vehicles present in the network sometimes is not enough to make the data flow among vehicles without problems, because of the constrained mobility, traffic lights, intersections, etc. This all leads to partitions in the network and uncertain densities. These properties contribute to real challenges for routing protocols implementations. However, there are some characteristics in VANETs that provide a good support to routing protocols, such as predictable mobility and the access to information like geographic coordinates, maps, schedule time for transport systems, etc. These informations can be exploited to improve the efficiency of routing protocols decisions.

Therefore, the requirements and challenges that routing protocol's designers must explore are listed below [12].

- `Localized operation`: nodes take routing decisions purely based on local available information. It reduces the overhead because nodes do not need to know the entire network topology.

- `Neighborhood discovery`: beacons with reliable information are sent with different time periods depending on the network state.

- `Identification of the destination`: it uses query messages waiting for the destination's reply to know its position and direction if its moving.

- `Trajectory precomputation`: create a route for the packets to flow to the destination, i.e., mark the packets to flow within certain roads and, if the network demands it, the relay nodes should act to find other routes if the one that the source chooses is not available due to network changes.

- `Data forwarding`: Nodes should act at the time of the forwarding; this means that each relay should be able to choose the best node to forward the packets at the instant it receives them, based on destination's position.

- `Dealing with network partitions`: the high dynamic characteristics of VANETs sometimes creates partitions within the network. The scenario of a relay node not having other node to transfer the information is a possibility. DTNs are one solution to this problem implementing store, carry and forward mechanisms.

- `Prediction of future events`: information about vehicles, like direction of movement and velocity can be helpful to know the vehicle's position in a given instant.

- `Use of additional information`: information about traffic's state can be a strong help to routing protocols.

The routing protocols in VANETs can be classified in five categories [12]: topological, geographical, hierarchical, movement-based and broadcast-based routing protocols. There are also protocols specific for the type of communication: unicast, multicast and broadcast. Finally, they can be applied to different types of network densities, some of them are better suited to different environments. Table 2.1 represents a classification of routing protocols, as shown in [12].

2.2.5.1 Topological Routing Protocols

Topology-based routing protocols use link's information present in the network to deliver data packets from the source to the destination, using the nodes composing the network. These protocols can be divided into proactive and reactive. The proactive protocols create tables even when there is no message to route, while reactive ones only determine a route to the destination when it is requested. According to [13], proactive routing protocols have low latency for real time applications: there is no need for route discovery, but otherwise, the unused paths waste a considerable part of bandwidth. The reactive ones, on the other hand, save bandwidth because they are beaconless and there is no need to flood the network on every routing table's update. The large disadvantage of this type of routing protocols is the high latency for route finding.

Traffic Type	Communication Type				
	Unicast		Multicast		Broadcast
	Topology	Position	Geocast	Mobility	
Adapted to sparse networks		Epidemic MDDV VADD		Epidemic MDDV VADD	Epidemic MDDV VADD
General	AODV DSR fast OLSR OLSR	DREAM GSR MGF MORA MURU	DRG GAMER IVG LBM MGF	RBM VTRADE	DREAM
Adapted to dense networks	CBRP HSR	CAR LORA-CBF GPCR GPSR	GeoGRID	LBF OABS ODAM SB SOTIS UMB	

Table 2.1: Routing Protocols Classification [12]

2.2.5.2 Geographical Routing Protocols

The geographical-based routing protocols rely on positioning data information given by GPS receivers on each node, like GAMER and LBM [14]. The address specifies an area, while in topology-based, the address specifies a single node. Each node knows its own and neighbour's position, and these protocols do not maintain routing tables. The routing decisions are made based on GPS information.

These types of protocols are scalable, suitable for high mobility environments and do not need route discovery. The inconveniences are the requirement of services to determine positions, and the fact that GPS does not work in tunnels.

2.2.5.3 Hierarchical Routing Protocols

Hierarchical routing protocols assume that the network is formed with several clusters, which are defined by a set of nodes connected for a defined period of time, like CBRP [15] and HSR [16]. The messages are transferred from cluster to cluster using gateway nodes.

This type of routing protocol tries to optimize the resource usage, but has some disadvantages, such as the increase of overhead needed to build the clusters as the network dynamics increase, and the fact that the dynamics turn the clusters compositions highly unstable. Concluding, the use of this routing solution should not rely uniquely on clusters, because in some cases they might be composed by a small number of nodes, or clusters might not exist at all. This solution should be implemented together with other routing solutions in order to make use of clusters when it is possible and convenient, augmenting the performance of routing strategies by limiting the number of re-transmitters.

2.2.5.4 Movement-Based Routing Protocols

In this approach, the messages are forwarded to the destination by means of a node's movement, which means that a node can carry information until it meets the destination, like epidemic routing protocol [17]. Since it is not possible to change the vehicles movement to a specific target, this strategy is not always sufficient or practical; therefore, movement-based routing protocols should be used with other routing strategies.

2.3 Delay Tolerant Networks

This section will present the main subject inherent to this Dissertation, DTNs. A brief history will be presented about this type of networks, its characteristics and applications, routing protocols and main challenges.

DTNs are networks that enable communication where connectivity issues exist, like sparse and intermittent connectivity, long and variable delay, high latency, high error rates, highly asymmetric data rate, and even no end-to-end connectivity [1].

A first architecture of DTNs was initially projected for the Interplanetary Networking (IPN) [18][19] to tolerate long delays and predictably-interrupted communications over long distances, in the deep space. Later, when the first draft was published by RFC 4838 [20], the authors showed intention to extend the concept to other types of networks, including specifically terrestrial wireless networks in a way that the messages within the networks could be sent in an opportunistic manner.

2.3.1 Applications

The use of Transmission Control Protocol/Internet Protocol (TCP/IP) in certain environments is not adequate or even possible, since this service was projected to end-to-end communications, which means that there is a path between the source and the destination, to communications where the maximum round-trip time between a pair of nodes is not significantly large and the packet drop is small. Several challenging networks can violate these principles turning TCP/IP model unsuited to satisfy these network needs. This is where DTNs appear to allow the communication in these sparse networks where "disconnection" is a popular word.

The previous networks facing the challenges presented before can be the following [21]:

- `Terrestrial mobile networks`: the node's mobility may cause the partition of the network unexpectedly, for a certain period of time, or it can be predictable to the network. Consequently, the traditional TCP/IP would not work due to the absence of an end-to-end connection. Then, DTNs can solve the problem because it is possible for a node to deliver information from one source to its destination without the connection of these two nodes. It receives the message from the source, stores it, and when (and if) connected to the destination, it delivers the message. A specific example of these networks are the VANETs.

- `Military ad-hoc Networks`: Hostile environments as the military ones are typical for disconnections due to the mobility, environmental factors and intentional breaks of connections. The limited bandwidth may have to be shared by services with high level

priority with the ones with lower requirements, making the last ones wait for several seconds or even more.

- `Sensor/Actuator Networks`: in this type of networks, nodes are characterized by their limited power, memory and processor capabilities. The number of nodes are usually considerably high, and they can go from thousands to millions of nodes. In order to save power, the communications are often scheduled. These properties make this kind of networks a potential target to DTNs.

2.3.2 Architecture

DTNs are characterized by their store-and-forward mechanism, where the messages are sent from node to node, until the final destination, when no path exists from source to destination. Due to the connection losses, the nodes should be able to store data for considerable time (hours, days). Consequently, they need to be equipped with hard drives, flash memories, etc. The store-and-forward mechanism is provided by a new layer in the protocol stack: the bundle layer shown in figure 2.4.

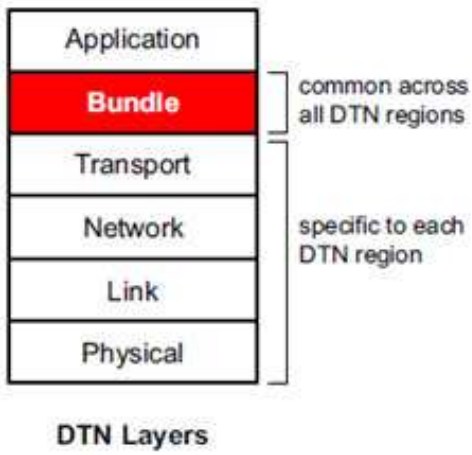

Figure 2.4: Bundle Layer [22]

As it is shown in the previous figure, there is a new layer between the Application and Transport layers responsible for the store-and-carry mechanisms. The DTN architecture will be explained later in this section.

The figure 2.5 illustrates a simple scenario of messages being transferred in a network based on a DTN. The message is transferred from A to D via node B and C through the mobility of the nodes without an end-to-end path.

The DTN architecture was designed to deal with heterogeneity found in networks, not only to accommodate network connection disruptions. IP protocol model supports heterogeneity too, but it does so by requiring the following in every node:

- a common IP address;
- packet format with specific semantics;

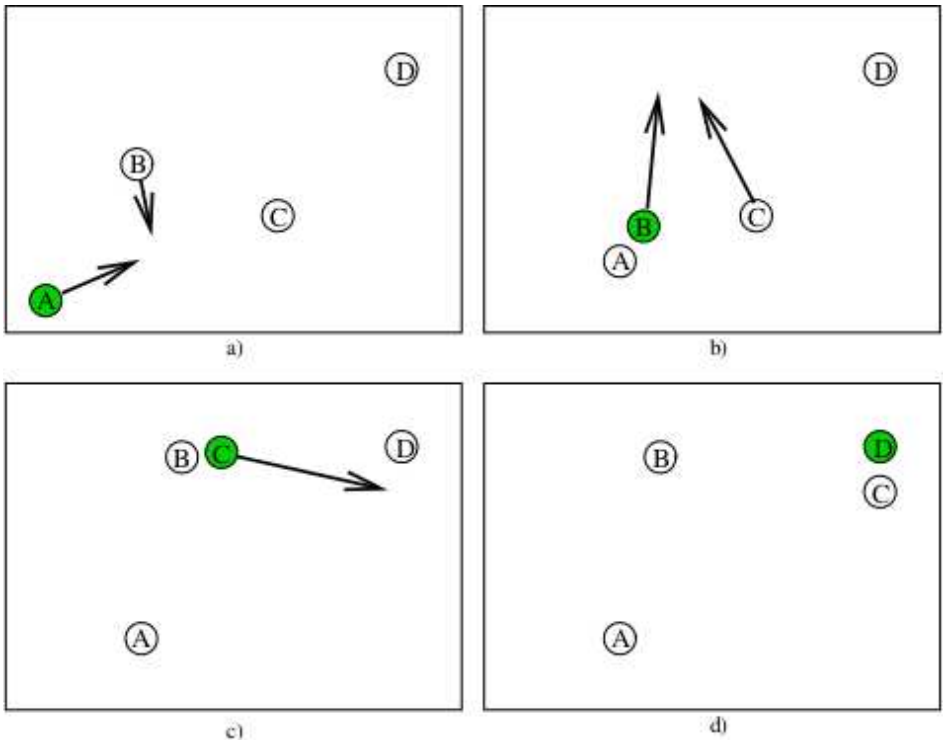

Figure 2.5: Message being transferred from A to D (node with information in green) [23]

- routing methodology that assumes a connecting routing graph to achieve interoperability.

On the other side, DTN architecture uses naming, layering, encapsulation and persistent storage to link heterogeneous portions of a larger network, independently of formal layer.

DTN can use a large number of delivery protocols including TCP/IP, raw Ethernet, serial lines or hand-carried storage drives for delivery. For each protocol used, different semantics can be used, consequently there is a collection of protocol-specific Convergence Layer Adapters (CLAs) providing all the necessary functions to carry DTN's protocol data units (bundles) on each of the corresponding protocols.

As it can be seen from figure 2.6, the bundle forwarder is the central unit who is responsible for moving bundles between storage, CLAs and applications depending on decisions made by routing algorithms. The arrows in the figure can represent bundles moving between entities or directives, management, routing decisions, etc.

2.3.2.1 Naming, Addressing and Binding

Naming and addressing are considered to be some of the most important and challenging aspects of a network architecture. Names are usually variable-length strings, while addresses are fixed-length identifiers. Mapping and binding functions are used to convert names into addresses. On Internet, Domain Name System (DNS) is the responsible for this task, while in other overlay network systems, it can be done by a locally-executed hash function.

During DTN's architecture evolution, nodes have always had identifiers which are used in the context of the bundle protocol, which is the responsible for providing basic message

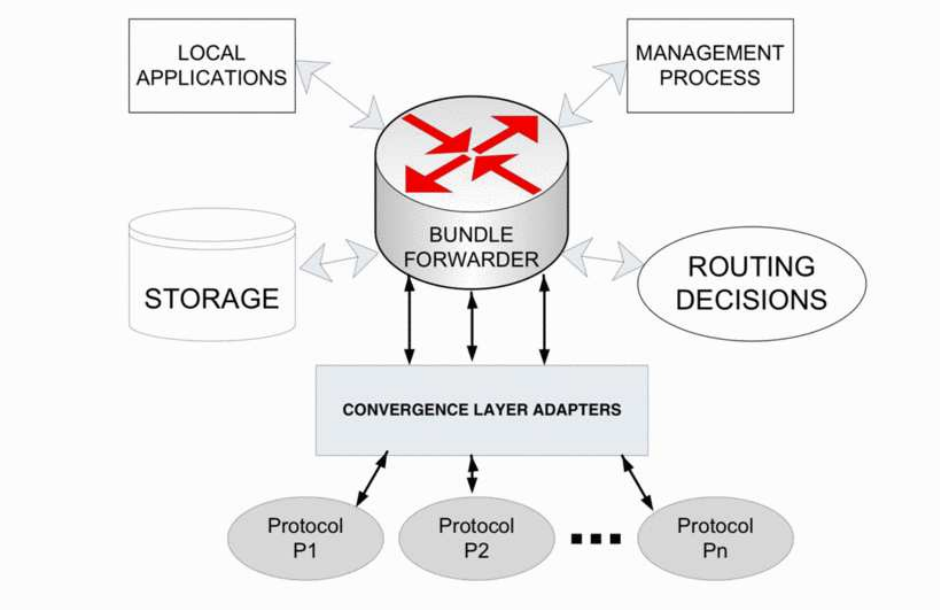

Figure 2.6: Bundle forwarder interaction architecture [24]

delivery service for DTNs. In the beginning of DTN's project, identifiers in the bundle protocol were constructed considering three parameters: region, host and application, making possible to identify a specific application of interest in the host and not only the host itself. A region is a portion of the network topology assumed to represent a well-connected area surrounding a planet (in the original IPN design).

After some time and consideration, the region construct was modified considerably because of the mobility of nodes and the possibility of having multiple network interfaces. Additional flexibility was required in how nodes were named, so multiple namespaces were supported with differing naming semantics. This characteristic allows hosts to have multiple identifiers.

In order to specify nodes with multiple identifiers, IETF developed work in the area of generalized naming systems, in the form of Universal Resource Identifierss (URIs). URIs have a few important properties, such as:

- `Allocated Name Spaces`: a URI usually has the form <scheme>:<scheme-specific-part>, where the scheme part is a string allocated from a set of several scheme names such as http, sip, file, etc.

- `Variable length`: URIs are essentially free-form except for a few reserved characters that have special semantics.

- `Structured Semantics`: URIs have a general syntax and semantics that they have to obey, but a new scheme may define new additional semantics, which are applied to all URIs.

In DTNs, URIs are referred to as Endpoint Identifiers (EIDs).

For messages containing DTN URIs with symbolic names, some "binding" step is done by one or more node along the path to the destination. In the Internet, when DNS is called at the sending node, it is called a case of "early binding", where a DNS name is linked to

an IP address. Since DTNs support direct forwarding based on names, "early binding" is not required at all. Due to this property, messages are passed along nodes towards their destinations based on forwarding entries present at DTN nodes that match the specific name. This is called "late binding", which is one of the characteristics of DTN's architecture.

2.3.2.2 Primary Bundle Fields

The applications in the DTN's architecture operate on messages carried in variable-length protocol Protocol Data Units (PDUs) called bundles [24]. Bundles contain a collection of typed blocks. Each one of them has meta-data and some of them also have application data.

The primary/first block of each bundle, present in figure 2.7, contains several fields:

- `Creation Timestamp` which is the concatenation of bundle's creation time and an increasing sequence number in order to guarantee its uniqueness for each Application Data Unit (ADU) generated at the same source. Creation Timestamp is based on the time-of-day an application requested an ADU to be sent and not when the bundle is sent through the network. This means that DTN nodes should have time synchronization capability.

- `Lifetime` corresponds to the time-of-day that the bundles expires, it is not useful any more. If a bundle is present in the network, including the source node, when the lifetime is reached, it can be discarded. Lifetime is expressed as an offset relative to its creation timestamp.

- `Class of Service Flags` indicates the delivery options and the priority class of the bundle.

- `Source EID` is the EID of the source of the bundle, the first node that sends it.

- `Destination EID` is the EID of destination, the node which it is intended to be the receiver of the sent bundle.

- `Report-to EID` identifies the node where reports should be sent. It can or cannot identify the EID of the source.

- `Custodian EID` is the EID of the current custodian of the bundle.

Most fields are variable in length and use a compact notation called Self-Delimited Numerical Values (SDNVs).

2.3.2.3 Fragmentation and Reassembly

DTN's fragmentation and reassembly improve the efficiency of these networks by ensuring that parts of bundles sent previously are not sent again. It means that, when time or resources are not sufficient to send the entire bundle to the next node, fragmentation occurs and on the next transmission the same part will not be re-transmitted; it will be retransmitted only the part that was not transmitted and the several fragments will be reassembled together forming the entire bundle.

Fragmentation can be classified in two different types [20]:

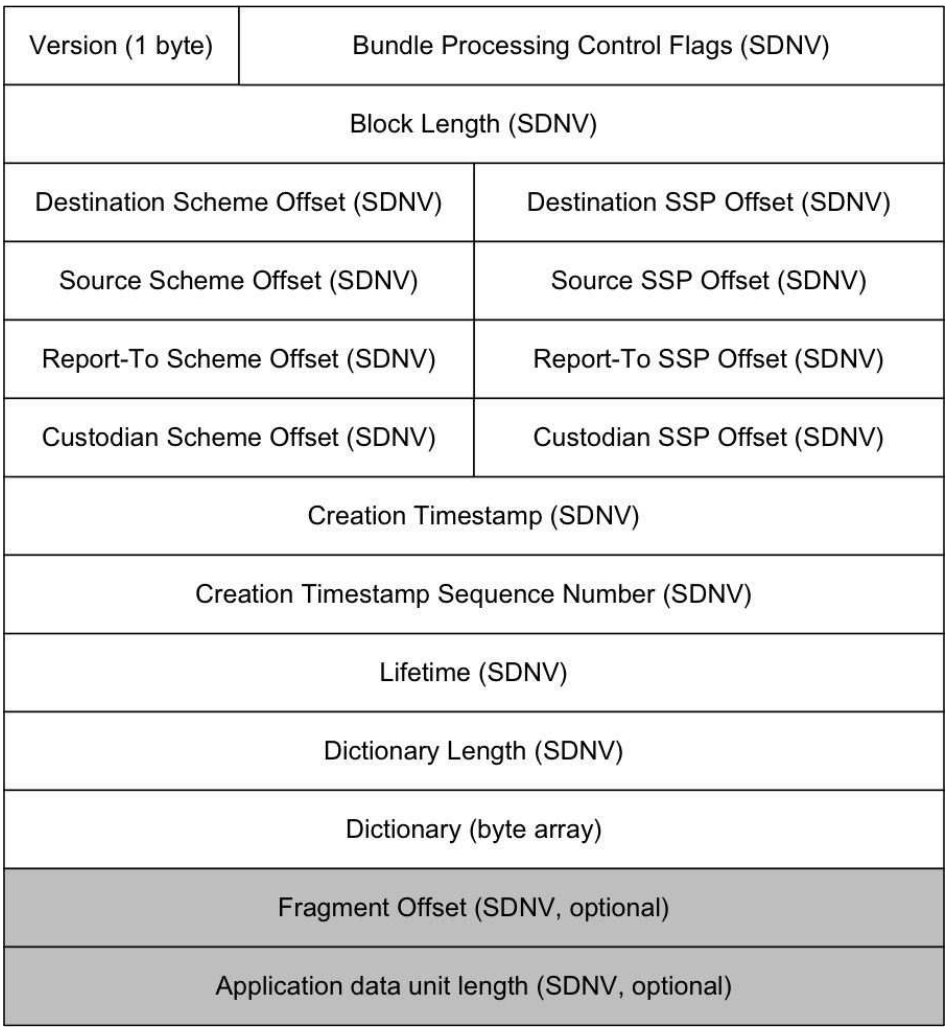

Figure 2.7: Primary Bundle Fields [24]

- `Reactive Fragmentation`: the process of fragmentation happens after the attempt to transmit the entire bundle and for some reason the transmission ceases, fragmenting the bundle. A portion of the bundle is delivered to the next-hop, and the remaining is transferred when contacts become available (different next-hops are possible).

- `Proactive Fragmentation`: on the other side, in this case, the division of the entire bundle per multiple blocks is done before the transmission, which means that every smaller block is transmitted individually as an independent bundle. The destination node is the responsible to extract the smaller blocks from incoming bundles and re-assembling them into the original entire block. Its is called proactive fragmentation because it is used primarily (before transmission) when contacts are predicted in advance.

2.3.2.4 Custody Transfer and Reliability

The bundle layer provides two options with the goal of enhancing the delivery reliability, which are the end-to-end acknowledgements and custody transfer. The custody transfer mechanism consists in passing the bundle's responsibility for reliable delivery to several nodes along the network. The nodes receiving the bundles (along their way to destination) are called "custodians" as long as they agree to accept the reliable delivery responsibility. The movement of a bundle from node to node, including reliable responsibility, is called "custody transfer". This mechanism allows the source to pass the retransmission responsibility and recover its retransmissions-related resources sooner, after sending the bundle.

This process provides as well a method for tracking bundles with special "cares" and to identify the nodes that participated in the transfer. It provides also a mechanism for enhancing the reliability of message delivery. When custody transfer is requested (which cannot be as well), the bundle layer provides an additional timeout and retransmission mechanism, and an acknowledgement mechanism between the two custodian nodes. When it is not requested, these additional mechanisms are not used and so, the delivery depends only on the underlying protocols [20].

2.3.2.5 Security

As it was mentioned before, DTNs are usually used in environments where resources are extremely scarce, and in these cases there must be some form of authentication and access control is essential. Some situations are not acceptable, like an unauthorized user flooding the network with traffic denying resources to authorized users. In many cases, it is not acceptable as well unauthorized traffic to flow over certain links within the network (mission-critical links).

Due to these facts, acording to [20], several rules are established to grant the security in DTN architecture:

- Deny unauthorized applications from having their data stored or carried through the DTN;
- Deny unauthorized applications from having control over the DTN infrastructure;
- Control the authorized applications in order to control their rate or class of service when sending bundles;
- Discard bundles that are corrupted or improperly modified in transit;
- Detect and deny access to compromised entities.

Many existing authentication and access control protocols designed for operation in low-delay and connected environments may not perform well in DTNs due to the known characteristics of these networks.

In order to satisfy the previous requirements, DTN adopts standard security architecture (DTNSEC) [25], which uses hop-by-hop and end-to-end authentication mechanisms. The reason to use both approaches is to be able to handle access control for data forwarding and storage apart from application-layer data integrity. The end-to-end mechanism is used to authenticate a principal such as a user, while the hop-by-hop mechanism is used to authenticate DTN nodes as legitimate transceivers of bundles between them.

2.3.3 Routing

Routing is one of the fundamental problems that appears when designing networks that have to deal with disconnections and long delays. Many challenges come across this kind of networks, not present on traditional ones. In DTNs, a contact is whenever a node has connection with another one, making transfer of messages possible [21]. The main challenges in these networks are:

- `Contact Schedules`, since the times between contacts can range from seconds to days, depending on the area of application used.

- `Contact Capacity`, which is the amount of data transferred during a contact. It can be a real issue when the amount of traffic in the network increases (large number of users) and the amount of data transferred per time is low. When the size of messages to be sent are very small compared to the links capacity, contact capacity is not a problem; but when the size is bigger, it is important to take this problem into account.

- `Buffer Space`, when the disconnections are long, the intermediate nodes need to store and carry messages for long time. In some cases these nodes may not have a lot of available space, so this parameter, in some cases, should be taken into account in routing decisions.

- `Processing Power`, in the case of sensor networks, as an example, where the devices are very small and their processing capacity is small as well, they might not be able to run complex routing protocols compromising this way routing strategies.

- `Energy`, it is possible that some nodes have limited energy, either because they are mobile or just because they are located in places where the connections to power supply are impossible. Heavy routing protocols send more messages between nodes causing more power use. Some researchers have investigated techniques for saving power in DTNs [26].

According to [27] there are two properties that can catalogue delay-tolerant routing strategies: replication and knowledge. The first property uses copies of the messages to be sent as a routing strategy to deliver them to the destination(s), while the second one makes use of the network information to make decisions.

There are two families of strategies according to the previous properties to classify them: the flooding and the forwarding strategies. The first one relies on replication of messages with the intent to deliver them to the destination, while the last strategy relies on network's knowledge to discover the best path to the destination.

Regarding flooding strategies, the simplest routing protocol consists in making copies of messages and deliver them to the relays whenever a contact is made. In this case, relays receive information about the number of copies to make, and the messages will be transferred from node to node until they reach the final destination. In this approach all nodes in the network will eventually receive the messages.

Forwarding strategies select the best path to send messages based on network topology information. Usually they send a single message along the best path, not using replication. This path can be found:

- `using location-based routing`, this approach requires the least information about the network, since it only needs to assign coordinates to each node. A message is forwarded to a next hop if this potential next custodian is closer in the coordinate space than the current node. The great advantage of this method is the lack of network's information needed by nodes, where routing tables are not used reducing the control overhead. The only relevant information a node needs to determine the best path is its own coordinates, destination's coordinates and the coordinates of potential next hops. Two problems appear using this strategy. The first one is the connectivity between two nodes close to each other. Even if they are close there is no guarantee that they are able to communicate to each other (e.g. objects like walls can be blocking the communication between the two). The second problem is the fact that node's coordinates can change due to their possible movements within the network. It complicates routing because the source needs to know the coordinates of destination. Lebrun *et al.* proposed the use of motion vectors to predict the future location of nodes [28].

- `using gradient routing`, which consists in assigning a weight to each node representing its "capacity" to deliver messages to a given destination. If a custodian contacts a node with a better metric for the message's destination, it passes it to the given node. This method demands more network knowledge than the previous one because each node needs to store metrics for all potential destinations, and sufficient information must be propagated through the network, to all nodes to compute their metric for all destinations. The shortcoming of this routing process is the initial time to find a good custodian because of the time to propagate all values for node's metric; moreover, the metric values in the region of the initial custodian are equally low. One approach to solve this problem is to initially use random forwarding until the metric values reach a certain threshold [29]. Burgess *et al.* use an equivalent mechanism to, in a first instance, propagate a new message in their epidemic routing variant [30].

- `or by assigning metrics to links`, this technique relies on traditional network routing protocols. A topology graph is built, where weights are assigned to the links and an algorithm is ran to find the best paths between all links. This method requires the highest network knowledge of all three because each node must have enough knowledge to process a routing algorithm. The problem of this methodology is the fact that the messages can take a long time to transverse the network, and so, the timing to make routing decisions is crucial. Jones *et al.* state that the best approach is to make decisions as late as possible; this way the messages are forwarded based on the most recent information [31].

2.3.3.1 Epidemic Routing

Epidemic Routing in an example of a flooding strategy mechanism. This algorithm guarantees that all nodes will eventually receive the messages if a sufficient number of random data exchanges is provided. Whenever a message is sent, it is placed in the node's buffer tagged with an unique Identifier (ID). When a contact appears, the two nodes send each other their list of all the messages IDs they have stored in their buffers (summary vector). Using this vector, nodes only exchange messages that they do not have, synchronizing this way their databases. Vahdat and Becker used these algorithms to forward messages in a DTN [17]. This strategy is highly robust to node and network failures due to its redundancy. Every

path is tried so the message will be received by destination in the minimum possible time, if the available resources allow it.

The figure 2.8 illustrates an example of this protocol. There are six nodes labelled A through F. The contacts between them go up and down one at a time, in the sequence shown by the labels on the lines. All links are bidirectional.

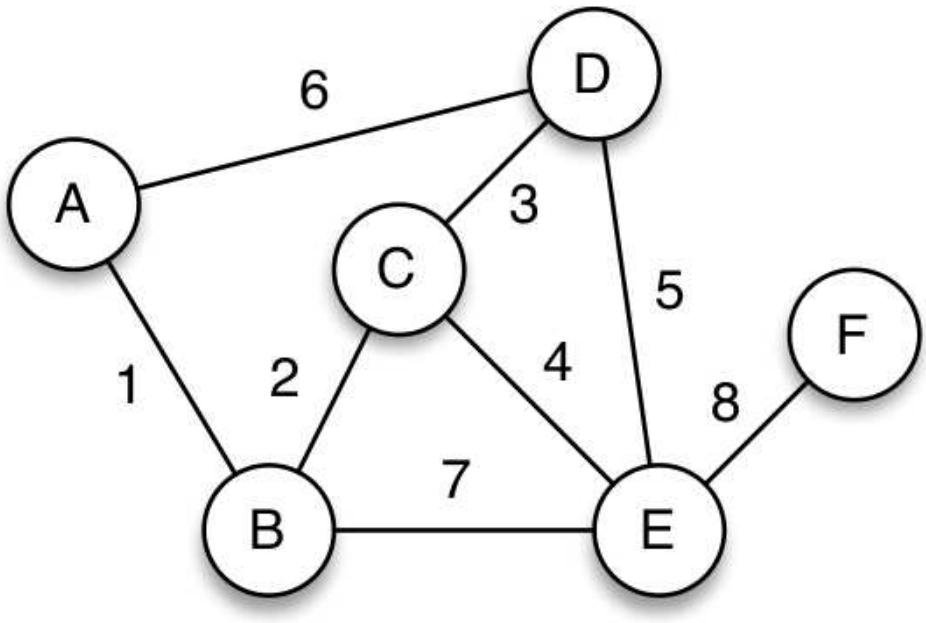

Figure 2.8: DTN Scenario - Epidemic Routing [27]

A message is generated by node A with the destination E. The delivered message from A to E will be delivered via the fastest path (A-B-C-E) as it can be seen by analysing figure 2.8. In this example all nodes will receive the message except the node F because node E will not replicate and forward messages that are destined to itself.

One of the problems of this routing protocol is the resources consumption because, after the message being received by its destination, the other nodes continue to replicate it through the network. Some proposals of "death certificates" appear in order to solve this problem where a new message is propagated to inform nodes to delete the original message and not to request it again [32].

2.3.3.2 PROPHET

Since nodes in MANETs do not move completely random, but in a predictable way, routing decisions should take this property into account when making decisions where to transfer the messages. If a node visited a specific location several times, it is likely that it will visit that same location again. Probabilistic ROuting Protocol using History of Encounters and Transitivity (PROPHET) makes uses of this fact to improve routing performance, saving networks resources.

A probabilistic metric was established, $P_{(a,b)} \in [0,1]$, at every node a for each destination b, which indicates the probability of delivering a message to that destination. PROPHET's operation is very similar to epidemic's: when there is a contact between two nodes, they

exchange their summary vectors containing the delivery predictability. The information is used to update the delivery predictability vector, and then, the summary vector is used to decide which messages, if any, are going to be exchanged between the two nodes, based on the forwarding strategy.

The calculation of the delivery predictability is composed by three stages:

- the first stage is the metric update whenever two nodes meet, increasing the delivery predictability. The calculation is given by equation 2.1 where $P_{init} \in [0, 1]$ is an initialization constant.

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

- the second stage belongs to the situation when two nodes are mismatched for a while, which means that the probability for them to deliver messages to each other is less likely, so the delivery predictability must *age* as follows in equation 2.2, where $\gamma \in [0, 1[$ is the *aging constant* and k is the number of time units that have past since the last time the metric was aged. This time unit should be defined based on the application and the expected delays in the network.

$$P_{(a,b)} = P_{(a,b)old} \times \gamma^k \tag{2.2}$$

- the third and last stage is relative to the transitivity property, that is based on the fact that if a node A frequently contacts with node B and node B frequently contacts node C, then node C is probably a good node to forward messages destined for node A. Equation 2.3 shows how the delivery predictability is affected by this property, where $\beta \in [0, 1]$ is a scaling constant deciding how large should be transitivity impact on the delivery predictability.

$$P_{(a,c)} = P_{(a,c)old} + (1 - P_{(a,c)old}) \times P_{(a,b)} \times P_{(b,c)} \times \beta \tag{2.3}$$

The forwarding of messages in DTNs is not so easy comparing to traditional networks where an end-to-end path exists most of the times. When a message arrives at a node, there might not be a path towards the destination available, so it needs to store the information; when a contact is possible, the node has to decide whether or not to deliver the message.

This decision is a delicate matter, since it depends on various aspects. In some situations, the best option is to set a fixed threshold, and the message is only sent if the delivery predictability between the nodes in contact is larger than this value. Furthermore, when encountering a node with a low delivery predictability, it is not certain if the right option to take is not to transfer because it can take a lot of time to find a node with a better metric. This way, there might be cases where the restrictions to transfer messages are not so restrictive.

Given these facts, delivering a message to a large number of nodes will obviously increase the chance of delivering it to the destination, but at the same time many resources will be wasted. On the other hand, giving a message to a small amount of nodes will save resources, but at the same time it decreases the chances of delivering it to the destination. Some considerations need to be taken into account in order to have a good relation between wasted resources and delivery ratio.

Lindgren *et al.* [33] analysed the behaviour of PROPHET and epidemic protocols in two different simulation's scenarios: scenario with random mobility and a model scenario where the nodes have a trend to go to specific places. From the results, they conclude that, in the random mobility scenario, the performance of the two protocols is identical, with slightly better results for PROPHET. The delay and delivery ratio are very similar, but PROPHET has lower communication overhead being more efficient. This better performance regarding communication overhead is related to the forwarding of messages of PROPHET, since it only forwards to "better" nodes, while epidemic routing sends messages to every nodes that enter in contact.

On the other scenario where nodes do not move randomly at all, the performance between the two protocols is very different with PROPHET showing better results. In some cases the message delivery is up to twice comparing to epidemic's.

2.3.3.3 DTLSR

Fall and Demmer [34] presented a routing protocol called Delay Tolerant Link State Routing (DTLSR) with the intent of solving problems in challenged networks, such as those in developing regions. The DTN reference implementation has as a big challenge to create a good and robust routing algorithm due to the large scope of this network's applications [35].

The developing areas targeted by this protocol experience intermittent connection due to a several set of reasons, but the usually topology has an underlying stability. Most of times nodes do not change their location, unlike the scenarios that PROPHET and epidemic are focused on. The networks dynamics are mostly the result of link failures and not from nodes mobility.

The approach followed to design this protocol was to study the existent routing protocols in MANETs, and how to change them in order to suite the targeted networks needs. Fall and Demmer decided to modify the standard Link State (LS) routing due to its great flexibility and suitability to their needs.

In LS routing, all nodes are informed about the network topology, so each node can compute its elected next hop(s). This topology is usually flooded through the network using Link State Announcements (LSAs), which may also contain node's information, location, buffer space, etc. Joining sets of LSAs gives the evolution of network topology over the time. These announcements, in DTNs, could be composed of additional information as buffer occupancy, enabling the short path computation based on the previous property. LS has also two important practical features that make this protocol very interesting to the desired environments: first, it is simpler than the other mechanisms because it is possible to know all network topology by asking it to one single node; and second, the "control" network (responsible for carrying LSAs) does not need to be the same network that carries data.

The standard LS routing only sends a message when an end-to-end path is available at the moment of transfer. As it is known, in challenging networks, as those in developing regions, that may not happen frequently, so modifications had to be made to this standard routing to overtake this issue. If a certain link is not currently available, it may become so in the future. The probability of being available again is related to its past, so there is a need to require persistent stored LSAs to check its dynamic history.

Given the previous aspects, DTLSR is closely modelled on classic LS algorithms. As the network changes, LSAs are flooded through the network containing all necessary information about nodes, links, etc. These LSAs are flooded as bundles to all adjacent nodes. Unlike the

regular LS, DTLSR sends LSAs with very long lifetimes (hours or even days) and all nodes maintain a copy of the most recent one from other nodes in the area. These announcement updates are queued in nodes in case of a network partition, and when a link is again available to a neighbour, the flooding process checks if there are cached messages to send. New announcements are only generated if some new network changes happens affecting routes weight computations.

Regarding the calculation of the best paths, the challenge for DTLSR is to determine how to utilize paths that may not be available at the time when a node needs to make a routing decision, which may be available before the message expires. Conclusions from previous work on DTN routing [36], induced Fall and Demmer to opt for the mechanism of minimizing the expected delay for all messages to maximize the total delivery rate, following the Minimizing the Estimated Expected Delay (MEED) approach, by Jones *et al* [31].

The calculation of paths is accomplished by using link connectivity knowledge gathered from LSAs. Two different calculations are made for two different cases: for links which are thought to be available, and for the links who are thought not to be, described in [34].

The main difference between DTLSR and LS is the calculation of the paths because DTLSR considers current-unavailable links to best paths, while LS only considers the current-available ones.

Fall and Demmer elaborated several simulations to compare the performance of these two protocols, described in [34]. Their results for DTLSR were much better on the evaluated metrics, delivery ratio and delay. Finally, they conclude that small changes in traditional routing algorithms result in a good and effective routing system for intermittent networks, due to the often stable underlying topology.

2.4 Vehicular Delay Tolerant Networks

The main difference between VANETs and VDTNs is essentially the same between traditional networks and DTNs. The former assumes that an end-to-end connectivity path exists while the latter does not [37][38][39]. The VANET's concept is more suitable to dense networks with a considerable number of vehicles and RSUs, while the VDTN concept is appropriated to sparse networks where lack of connectivity is frequent, making use of the store-carry-forward mechanism.

The use of DTN characteristics is very useful due to the scarce transmission opportunities and intermittent connectivity, specially in mountain and rural areas where the number of nodes in the same location at the same time is low.

The majority of problems in VANETs is related with the speed and mobility of vehicles causing short durations contacts. The existence of buildings and tunnels compose radio obstacles leading to disruption and intermittent connectivity. All these conditions make these networks subject to frequent fragmentation/partition (contributing to the non-existence of end-to-end connectivity).

2.4.1 Potential Applications

There are several examples for VDTNs applications:

- `optimize the traffic flow`, preventing road congestion informing passengers about traffic conditions.

- `sensor networks`, collecting data from the outside like weather conditions and road surface conditions.
- `commercial applications`, advertisements, marketing data, tourist information (restaurants, hotels, etc), parking space availability, etc.
- `provide connectivity to remote areas`, like transfer of files, electronic mail.

Accordingly to [40], applications in vehicular communications are classified in four classes: 1-general information services; 2-information services for vehicle safety; 3-individual motion control using inter-vehicle communication; and 4-group motion control using inter-vehicle communication.

Regarding class 1 applications, delayed or lost information does not compromise user's safety, so DTN's competences can improve the quality of these services by increasing the delivery ratio and decreasing the delays in changeling environments. An example is a relay node on the base of a mountain gathering ice and fog warnings from cars coming down the mountain to provide them to cars going up.

On the other hand, services from class 2 cannot tolerate delayed information not to compromise safety. It can be informations about accidents on the road, alerts of potential dangers, etc. These services only tolerate short delays and DTNs can not guarantee them.

Lastly, services from classes 3 and 4 are used for example to control vehicles throttle and brakes on real-time, making these services sensitive to delays and loss of data, so DTNs are not suited to this class of services also.

2.4.2 VDTN projects

The table 2.2 shows several projects in VDTNs as well as their applications.

Unfortunately, VANET projects do not use the DTN concept, but if they did the communication efficiency would certainly increase [30][49][50]. However, the only alterations that have to be made are software modifications so it is expected that in the future DTN concept will be more used in these type of projects improving this way the delivery efficiency.

Projects like KioskNet [41] were fundamental to show that theoretical concepts of DTNs actually can bring a lot of improvements in non-real time vehicle-to-vehicle communications. However, none of them tested DTNs in a real vehicular environment with IEEE 802.11p technology.

2.5 Summary

This chapter provided an overview of vehicular networks based on the literature. In a first instance, a concept's definition was given, its architecture, routing protocols and several technical challenges.

Then, it was introduced the concept of DTN, which is the base concept of this Dissertation. The DTN architecture and its applications were presented, as well as several routing protocols and the challenges regarding the routing approach.

Finally, the application of DTNs in VANETs was presented, informing the reader about the main advantages and applications in introducing this concept in networks like VANETs. Several projects in VDTNs were referred to show the type of work that has been made in this specific area.

Project	Applications	Protocol Stack	Routing Protocols
KioskNet [41]	Internet access for rural sites	DTN standard stack	Epidemic
DieselNet [42]	Internet access for rural buses	DTN standard stack	MaxProp, RAPID
VDTN [43]	Internet access for vehicles	Bundle layer below network layer. Separate data and control planes	Epidemic, Spray-and-wait
CarTel [44]	Detection of road pavement defects	Mule adaptation layer below network layer	Static, Epidemic
EMMA [45]	Pollution measurements, Traffic information	DTN standard stack	Epidemic
Drive-Thru Internet [46]	E-mail, Web browsing	Session layer above the transport layer	Through infrastructure
CONDOR [47]	E-mail, Web browsing, IRC, Voice mail	DTN standard stack	Static

Table 2.2: VDTN's Projects [48]

All projects referred above used the DTN concept in several scenarios, but they never tested a DTN in a real vehicular environment with vehicles moving at different velocities, using the standard IEEE 802.11p. This Dissertation work will focus on introducing this concept in a real vehicular environment, using real OBUs communicating via IEEE 802.11p technology, testing different real scenarios with vehicles and fixed infrastructures.

Chapter 3

DTN Implementations

3.1 Introduction

The previous chapter presented to the reader the theoretical concepts that are behind the Dissertation work. This chapter will approach the real implementations of the bundle protocol used during the realization of the Dissertation's work.

Section 3.2 presents the first implementation of the bundle protocol used, DTN2, focusing on its architecture and applications. Section 3.3 studies the second implementation used, IBR-DTN.

Section 3.4 compares the two implementations used, through their advantages and disadvantages in tested scenarios.

The bundle protocol was defined by Delay-Tolerant Networking Research Group (DTNRG) [51], which is a research group chartered as part of the Internet Research Task Force (IRTF). This group is concerned with the architectural and protocol design principles to use in challenging environments where continuous end-to-end connectivity cannot be assumed. DTNRG is working on two protocols for which there is code publicly available. The bundle protocol is a general overlay network. The Licklider Transmission Protocol (LTP) [52] is a point-to-point protocol to work on very high delay links, such as those used in deep space communications.

Several implementations of the DTN architecture were created [53]:

- `DTN2`, the reference implementation of the bundle protocol, which also provides a framework for experimentation, extension and real-world deployment [54].
- `Interplanetary Overlay Network (ION)`, which was implemented to support high-speed, small-footprint deployment of DTN in embedded systems like robotic spacecraft [55].
- `Postellation` is a DTN implementation, running on Windows, MacOSX, Linux and RTEMS [56].
- `IBR-DTN` is a C++ implementation of the bundle protocol developed at IBR [57] and aims to be very portable and is designed to run on embedded systems. The project can be found in [58].
- `JDTN` is a Java implementation containing the DTN "core" that runs on every platform that supports Java [59].

There are other implementations of the DTN architecture as well as simulation platforms like DTNSIM2 Simulator [60].

DTN2 and IBR-DTN will be analysed in detail. The first one used was DTN2, since it is the reference implementation of the bundle protocol. For reasons which are going to be explained later, DTN2 had to be discarded. IBR-DTN was then used as the second implementation due to its portable design run on embedded systems, which is the case of the boards used on vehicles.

3.2 DTN2

DTN2, as it was mentioned before, is a reference implementation of the DTN protocol and it is designed as an experimental platform for researchers.

3.2.1 DTN2 architecture

Figure 3.1 shows a diagram with the major components of DTN2 architecture [61] and how they interact with each other. It shows that the module Bundle Router is the most central component of the implementation. Routing decisions are based on information that is passed to this module; therefore, the information should be as much detailed as possible. These decisions are passed as a set of instructions to the Bundle Forwarder, which is the responsible for executing them. The distinction/separation between control and function allows easy extension, modification and replacement of the router module.

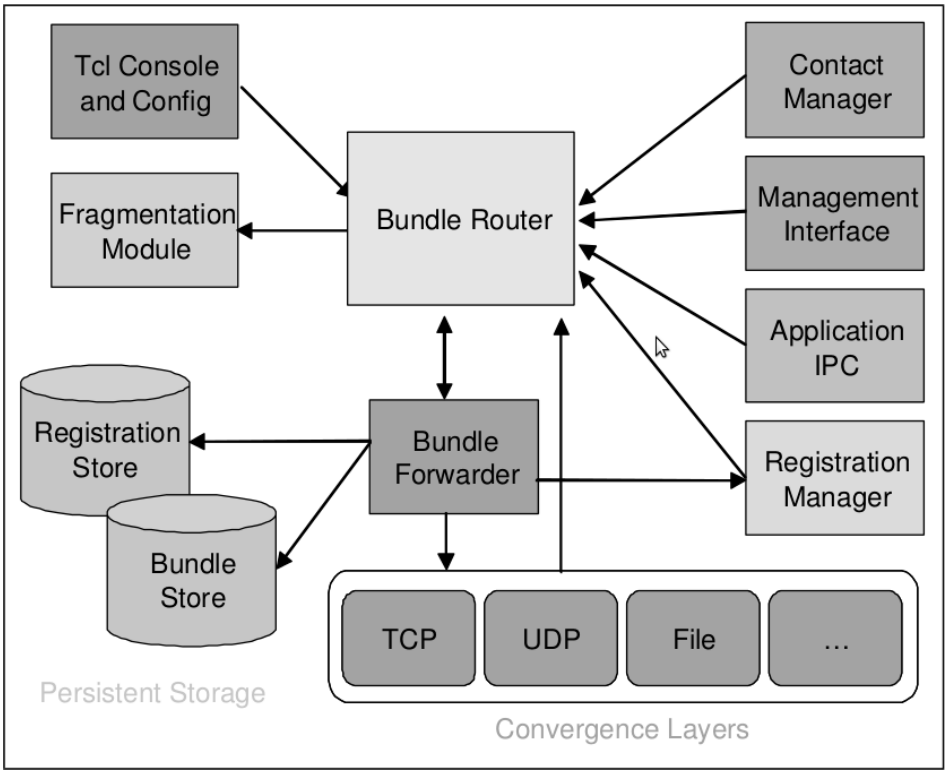

Figure 3.1: DTN2 Architecture Overview [61]

The following subsections characterize the several modules of figure 3.1.

3.2.1.1 Bundle Router and Bundle Forwarder

All the route selection and decision making are responsibility of the router component. It takes as input several events that can affect routing decisions and pass these decisions (as a set of instructions) to the bundle forwarder, which is the module compelled to execute these instructions. The forwarder executes these instructions by interacting with the convergence layers, the registrations and the persistent storage. The separation between the bundle forwarder and the bundle router coincides with separating the calculation of instructions from their execution, helping to separate the routing code from possible changes in other Application Programming Interfaces (APIs).

3.2.1.2 Convergence Layers

Convergence layers behave themselves as adapters between DTN bundling protocols and several underlying transport protocols. Basically, these components execute data plane functions where a particular layer must be able to send and receive bundles over a single hop. Sporadically, they also process signalling messages required by the bundle router like failed connections or restarts. DTN2 supports several convergence layers, such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Bluetooth, etc.

3.2.1.3 Persistent Storage

This module is responsible for the storage of bundles during the store-and-forward process. It is possible to choose the type of storage desired, the bundles can be stored in a database or stored in the conventional file system. This option can be taken at runtime due to the common abstraction for persistent storage.

3.2.1.4 Fragmentation Module

Fragmentation module is responsible for fragmenting and reassembling bundle fragments. There are two types of fragmentations: proactive and reactive. The first one consists in dividing a bundle into several pieces with the desired length when the contact has small volume; the latter is used whenever a transfer fails unexpectedly. This module has the capacity to signal the bundle router when all fragments of a specific bundle have been received.

3.2.1.5 Contact Manager

The present module is responsible for keeping track of links currently available, historic information regarding their connectivity and performance, and information about future contacts schedules that may become available. The primary function of this module is to transform the learnt information about contacts, from its own environment mechanisms, into mechanisms known by the bundle router (to set the instructions).

3.2.1.6 Management Interface

This module provides special signalling to the bundle router regarding policy constraints or preferences that may affect its routing decisions about transfer of data. It supports multiple

applications processing due to its implementation of interprocess communication capabilities.

3.2.1.7 Console/Config

The console/configuration module provides to users a command line interface and an event loop for tests and debugs of the implementation; it also provides a method to set the initial configuration options. The interpreter used to parse and execute user commands and settings is the Tool Command Language (Tcl) interpreter.

3.2.1.8 Application Inter-Process Communication (IPC)/Registration Module

DTN applications are built to use a thin library that makes communication with the bundle router via a communication channel. Its main activities are related to sending and receiving application messages, and message demultiplexing bindings.

3.2.2 Applications

There is a set of applications in DTN2 that can be used to test the implementation. These tools are used to send bundles through the DTN network. The most basic tools are *dtnsend* and *dtnrecv*. *dtnsend* is used to create a bundle with various options attached. The bundle will eventually pass over the network until it arrives to the destination node. *dtnrecv* is used in the final destination to retrieve the bundle. To use *dtnsend* the instruction needs the source, destination and the type of data to send as well as its location. Other command line options can be used to set optional flags in the bundle.

The *dtnrecv* only needs the destination node of the bundle in its command operation. It is possible to check on real-time the bundles in storage on each node by typing *bundle list* on the window where *dtnd* (dtn daemon) is running.

Other available tools are *dtncp* and *dtncpd*. These are used to move files from one to another node. *dtncp* picks up a file from local filesystem and sends it as a bundle. *dtncpd* is the daemon which contacts with bundle router requesting a notification when a bundle is received, sent by *dtncp*. The bundle is copied to a location set on the configuration file. The nodes, in order to have the files copied, need to run *dtncpd*.

3.2.3 Routing

Regarding routing, DTN2 has some routing protocols implemented, such as PROPHET [62], DTLSR [63], flood [64], tca-router [65], tca-client [66], external [67] and static [68] routing. To use these routing protocols with several nodes, DTN2 has a service discovery implemented as well, where each node announces to the network its IP address and when a contact appears, the node gets the other node's IP address. It is possible as well to create static links instead of using the service discovery.

3.3 IBR-DTN

The previous bundle protocol reference, DTN2, is not suited for embedded systems, compromising many applications that are constrained by costs or energy. In this section it will be

presented a flexible bundle protocol daemon designed to run on OpenWRT [69], a Linux distribution built for embedded systems. Due to these characteristics, IBR-DTN is completely compatible with uClibc [70], a system library for embedded systems.

3.3.1 IBR-DTN Architecture

The main goals of IBR-DTN implementation are to use the minimum external requirements such as libraries, and keep the software implementation as modularized as possible. It is implemented with coupled modules communicating using an event mechanism as it can be seen from figure 3.2. Due to this mechanism operation, IBR-DTN, when compiled, can be specifically adapted to the targeted platforms. Any functionalities depending on external libraries (like the bundle protocol security extensions are based on OpenSSL), apply only to the optional module.

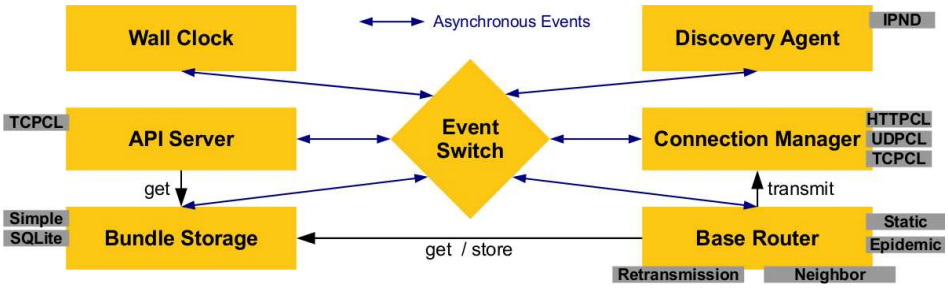

Figure 3.2: IBR-DTN Architecture Overview [71]

The following subsections will present the standard base modules composing IBR-DTN architecture.

3.3.1.1 Event Switch

The central module of this architecture is the *Event Switch*, which is responsible for dispatching raised events to all applicable sub-modules. IBR-DTN is capable of achieving high degree of concurrency between the several daemon modules due to its capacity of queueing events (requiring extensive processing in the module) to a private work queue of the module's thread. The standard modules of IBR-DTN are built to allow the creation of new-specific applications. Existing and new potential modules can raise and receive events to interact with other parts of the daemon. The actual implementation has standard events related with storage and routing of bundles, and events regarding the availability of nodes.

3.3.1.2 Discovery Agent

This module supports node's discovery through the implementation of DTN, IP Neighbour Discovery (IPND) version 1 and 2 [72], and a compatible version with DTN2. The control of appearance and disappearance of nodes is the responsibility of the present module, which generates events whenever a node is detected or lost, making the routing modules check if there are any bundles to send to the new node.

3.3.1.3 Connection Manager

In IBR-DTN, the Connection Manager controls the instances of the modules (based on the configuration file) which provide connectivity between the DTN daemons. These modules are the convergence layers, where each one of them provides a different interface to transfer the bundles to other nodes. Events are generated in case of failed transfers or incoming bundles to be stored in *Bundle Storage*. IBR-DTN comes with four implemented convergence layers:

- `TCP Convergence Layer`, compatible with IETF draft [73]. It uses handshake mechanism between the daemons and it is equipped with the capability of splitting bundles into segments, which are then acknowledged by the receiving daemon.
- `UDP Convergence Layer`, compatible with IETF draft [74]. The bundle has size limit to fit in a single UDP datagram.
- `Hypertext Transfer Protocol (HTTP) Convergence Layer`, it is based on libcurl [75] and it can use an HTTP server to send and receive bundles.
- `LowPersonal Area Network (PAN) Convergence Layer`, it supports the 802.15.4 MAC protocol [76] which is used in sensor networks.

3.3.1.4 Bundle Storage

The store-and-forward mechanism makes necessary the existence of a storage to buffer the bundles for possible long intervals of time. The *Bundle Storage* module was created to provide the needs to the previous mechanism. IBR-DTN has three types of storage to be used:

- `Memory`, its a non-persistent bundle storage and it is chosen by default when no path for storage is set. The bundles are stored in Random Access Memory (RAM).
- `File based storage`, it is used when a path is set and all bundles will be persistently stored in that location based on simple files. Although the functionality is the same comparing to *Memory* storage, in this case the bundles are saved on disk and their existence survives to daemon's shut-downs and losses of power. Since the memory is not being used to store bundles, the daemon requires much less memory to operate.
- `SQLite`, this type of storage uses an SQLite [77] database. It is a good approach for more complex routing protocols.

3.3.1.5 Base Router

This module is responsible for managing different routing modules interacting with the *Bundle Storage* and the *Discovery Agent*. It receives events about nodes entering in contact and leaving, and about bundles arriving to the storage. When a routing module wants to send a bundle, it contacts the *Connection Manager* and requests the desired convergence layer to make the transfer. There are several routing modules included in IBR-DTN:

- `Static`: routes and links are configured a priori. It assumes that all configured links are permanently available.

- `Neighbour`: the packets are routed to the available neighbour nodes discovered by the Discovery Agent.
- `Epidemic`: IBR-DTN also has an implementation of epidemic routing protocol based on [17]. The principles of this routing protocol were explained in 2.3.3.1. The difference is that IBR-DTN uses a BloomFilter [78] mechanism instead of summary vectors. This routing protocol also manages a purge vector capable of deleting delivered bundles from nodes.
- `PROPHET`: there is also an implementation of PROPHET routing protocol based on [79].
- `Retransmission`: it is responsible for signalling whenever an error during bundle's transmission occurs. The bundle is re queued and stays in the storage. If possible, the convergence layer will try to retransmit the bundle.

3.3.1.6 Wall Clock

The local time is determined by the local host's clock which is managed by the *Wall Clock*. The timestamps in DTN count the seconds since 1/1/2000. This module also provides a global time tick event which is delivered every second through the *Discovery Agent* to all the remaining modules.

3.3.1.7 IBR-DTN API

The bundle streaming protocol is reused to provide a socket based API interface. The constant re-implementation of bundle streaming protocol in each application could be very painful, that is why a library is linked to applications simplifying the creation of bundles. This approach has the great advantage that all supported features of the daemon are immediately ready. The implementation does not support out-of-band messages, so configuration on-real time is not possible (configuration of routing modules for example).

3.3.2 Applications

Like DTN2, IBR-DTN also has several implemented applications. *dtnsend* and *dtnrecv* have the same objective as in DTN2. There is an application, *dtnping*, that sends out bundles to a specific DTN EID and waits until it receives a bundle with the same payload as answer (it retrieves a printed out response time). *dtntracepath* makes use of the *bundle forwarder* module to discover the path of a bundle.

dtninbox and *dtnoutbox* are used to transfer files between two directories on different machines.

3.4 IBR-DTN vs DTN2

Several tests were realized to compare the two DTN implementations. According to [71], two different scenario tests were evaluated: API and bundle storage performance, and networking performance.

To evaluate API and bundle storage performance, they utilized *dtnsend* and *dtnrecv*. Two types of storage were tested in each implementation:*Memory* and *File* storage for IBR-DTN; for DTN2, *BerkelyDB* (filesystem) and *Memory* (RAM).

Regarding the bundle storage performance, IBR-DTN overcomes DTN2 when using disk based storage and the small bundle's size. DTN2 only shows better behaviour than IBR-DTN when bundles are really small and the type of storage is memory-based. When the payload size is bigger, the two implementations have identical performances.

For reception of bundles (*dtnrecv*), IBR-DTN and DTN2 have identical performance for file and memory storage cases. For bundles bigger than 5kbyte, IBR-DTN slightly outperforms DTN2 in both storage cases.

For networking performance test, IBR-DTN outperforms DTN2 in both memory and disk-based storage, showing better results regarding the throughput.

Several emulation tests were also computed in [80] using a large-scale network emulation testbed. In these tests DTN2 used two different routing protocols, DTLSR and flood, while IBR-DTN only used the flood since it does not have the DTLSR routing protocol.

They conclude that DTN2's performance tends to degrade with the number of nodes in the network. DTLSR performed with good results in scenarios with good connectivity, but in sparse scenarios its performance is lower than what was expected.

On the other side, IBR-DTN maintains the expected performance in the several scenarios.

3.5 Summary

This chapter introduced to the reader the two used DTN implementations during this work. Both architectures were presented to explain the two systems work as well as their embedded applications, routing protocols, etc.

The two implementations were compared based on experiments. It can be concluded that IBR-DTN is more suitable for embedded systems where space and memory are a constraint. This implementation also presented better results comparing to DTN2 in the majority of the tested cases explained previously, so it will be the one used in the following chapters.

Chapter 4

Integration and Development

4.1 Introduction

The previous chapter introduced the existing implementations of DTN, focusing on the two used, DTN2 and IBR-DTN.

The current chapter will present the steps taken to develop the final tests on the experimental testbed, describing all the used equipment, tests, problems found and their resolution.

Section 4.2 will present the used material to run the several tests, such as the boards (OBUs), cars, RSUs.

Section 4.3 describes the tests using DTN2, on the boards and on Personal Computers (PCs), as well as the problems found on the several tests.

Section 4.4 describes also the tests performed using the IBR-DTN. During these tests, several problems were found, which were solved in the scope of this Dissertation.

4.2 Equipment

The following section will present the equipments used in all tests and the scenarios described on chapter 5 to test DTN implementations on VANET's devices.

The following list and figures present the modules as well as the Operating System (OS) and driver used on boards (OBUs):

- `ALIX module`: it's the main board of the system. PCEngines Alix3D3 with a 500 MHz AMD Geode LX800 processor (32-bits x86 architecture) [81];
- `Wi-Fi module` able for IEEE 802.11a/b/g, for control operations.
- `Wi-Fi module` able for IEEE 802.11p/DSRC, [5.86-5.92]GHz in figure 4.1 [82];
- `Memory cards` kingston (8GB) and CF (4GB);
- `L-Com omnidirectional antenna` with magnetic support and extension for [5.150-5.9]GHz , 5dBi, IEEE 802.11p standard - used in vehicles;
- `Control Antenna` at 2.4GHz, 5dBi, 802.11a/b/g standards [84];
- `GPS` GlobalTop (MediaTek MT3329);

Figure 4.1: Wi-Fi Module containing 802.11p/DSRC Standard [83]

- `OS` buildroot 2012 02, kernel's version 3.3.7 [85];
- `Driver ath5k` modified [86][87].

Figure 4.2 shows the equipment ready to be used in vehicular communication; in figure 4.3 it can be seen the several connectors existing in the board. There are connectors on both sides of the board: two connectors to plug-in the two antennas (standard g and p), a power connector, Ethernet port, a serial port (RS-232), which is used to receive the values from GPS. There are also a Video Graphics Array (VGA) connector and two Universal Serial Bus (USB) ports.

The driver ath5k was modified to support the IEEE 802.11p/WAVE requirements, enabling the support of fast switching between control and service channels, reduced associations times and reliable support for emergency messages. The unmodified driver implements the requirements specified in IEEE 802.11 a/b/g. Since IEEE 802.11p is an amendment to IEEE 802.11 and IEEE 1609.4 presents improvements to the capabilities standardized in the former, the ath5k driver had to be modified to implement the DSRC required functionalities.

In [86], the several changed modules, *ath5k*, *mac80211* and *cfg80211*, are presented as well as their interaction to support the WAVE MAC sub-layer.

Figure 4.2: Board for vehicular communication [83]

Two cars were used in the real testbed, shown in figure 4.4. A Renault Clio with five doors (on the left) and a Seat Toledo with five doors as well (on the right).

Figure 4.5 shows the board installed on a car ready to be used, using a battery (right side of the figure) for power, and the connectors: GPS connector and the extensor connected to the 2.4GHz connector. The Ethernet cable was used to connect the pc to the board, allowing the management of board's services.

The antenna for standard IEEE 802.11p communication was placed inside the car, as it can be seen in figure 4.6, to evaluate the implementation in the most possible real scenario.

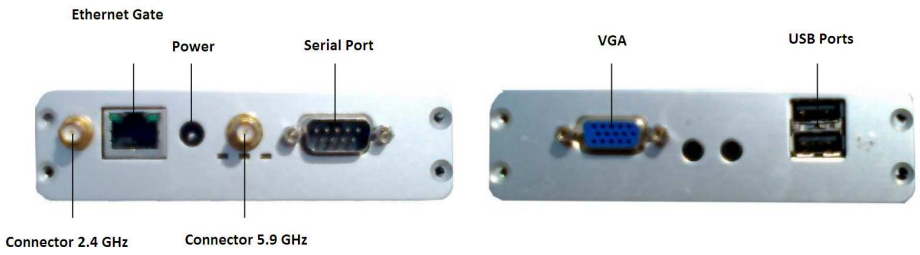

Figure 4.3: Available Connectors [83]

Figure 4.4: Renault Clio (left) and Seat Toledo (right)

This antenna was connected to the board via an extensor, while the board was placed in the trunk of the car.

RSUs were also used in the experimental testbed, placed along the road, and are shown in figure 4.7. On the top of the tripod there is the antenna for the communication using 802.11p, connected to the board, placed on the floor by an extensor.

4.3 DTN2

The first implementation used was DTN2 (version 2.9.0). This section will present the tests performed to evaluate its behaviour, problems and capabilities. First, the implementation is tested using PCs in different scenarios, and later is installed on the boards and tested using the vehicular communication (standard IEEE 802.11p).

4.3.1 DTN2 on PCs

The first experiments are performed using two PCs connected via Ethernet (figure 4.8) using static routing, where the link between the two nodes is created a priori in the configuration file.

This experiment allows to test DTN2 in the simplest scenario possible: transfer of files from A to B and vice versa.

The bundles are injected in the sender node using one of two possible tools, *dtnsend* and *dtncp*. These two applications need a set of input arguments; in the case of sending a file

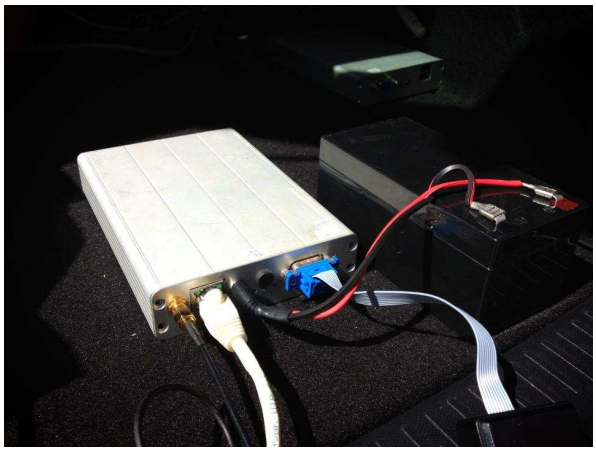

Figure 4.5: Board installed on car

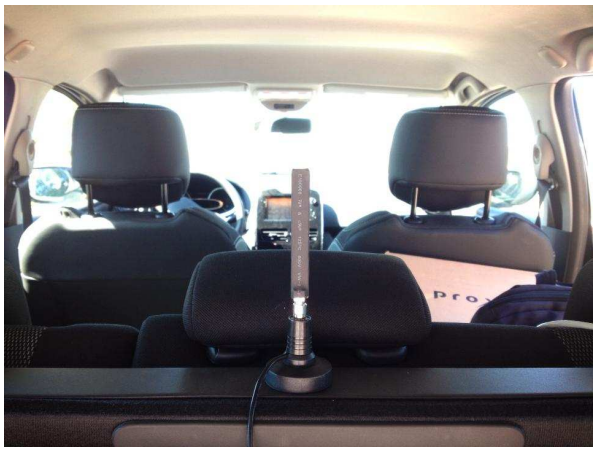

Figure 4.6: Antenna for standard IEEE 802.11p

from A to B, *dtnsend* needs:

- source node name;
- destination node name;
- location of file to send.

If the command used is *dtncp*, it only needs as input parameters the destination node name and the location of the file to send.

This scenario is also used to try the discovery mechanism (IP Neighbour Discovery), in other words, with no static links. The nodes flood the network with their own IP addresses, and when they find each other, the links are created to trade bundles, if it is the case. The links are not assumed to be always up; this means that, when two nodes stop receiving each other's messages of node's announcement, the link is assumed to be down.

In this scenario, no problems were found after setting well options in the configuration file. Between two nodes all routing protocols worked except the PROPHET routing protocol, which is not well implemented (this was one of the reasons to finally decide for IBR-DTN).

Figure 4.7: RSU

The next scenario tested was the one in figure 4.9, which evaluates the store-and-forward mechanism of the implementation.

To test this feature, bundles were sent using the same applications in the previous scenario from node A to node C (these two were not connected) through node B, acting as a relay node. In this scenario, the static routing and links were not used, only the rest of routing protocols based on IPND (flood and DTLSR). To assure that the nodes A and C were never connected, when the bundle was sent from A, C was not connected to the network and the bundle was sent to the only node available, node B. After node B has received the bundle, node A was disconnected from the network and node C was connected. After node C has connected the bundle was transferred from B to C, corroborating the store-and-forward mechanism. Some problems with the flood routing protocol were found, which will be explained in a latter section.

After running several tests with previous scenarios connecting nodes via Ethernet, the next step was to try the implementation with nodes connected via wireless. Figure 4.10 illustrates the first created scenario using wireless communications.

As it can be observed, two nodes are connected to an AP which is broadcasting a network. This simple wireless scenario is used just to see the implementation's behaviour in wireless environments. Bundles are traded between the two nodes, and the behaviour is similar compared to the scenario in figure 4.8, where the two nodes are connected via Ethernet.

The last scenario tested with DTN2 using PCs is the one represented in figure 4.11. In this scenario three PCs are connected via ad-hoc. PC A and PC C are each one broadcasting

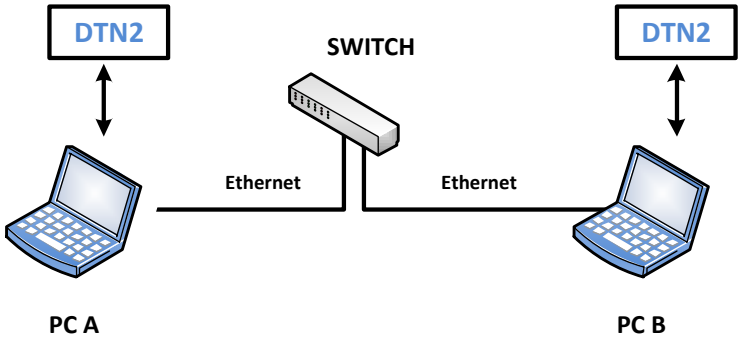

Figure 4.8: Two PCs running DTN2 connected via Ethernet

a network. In this scenario, PC B connects to each network, one at a time. Bundles are sent from PC A with destination PC C using PC B as relay, since A and B are never connected. PC B receives the bundles from A and then delivers it when connected to the network being broadcasted by PC C. The opposite way is also tested, with node C sending bundles to node A using B as relay. The problems that were found in scenario of figure 4.9 are also the ones of this scenario, which will be following explained.

4.3.1.1 Detected Problems

During the tests elaborated on the previous explained scenarios, several problems were detected, including the bad implementation of the PROPHET routing protocol.

When PROPHET routing protocol is set as the routing protocol in the configuration file of each node, it is not possible to transmit bundles from one node to another. In the scenario where two nodes are used (figure 4.10 and 4.8), when the two daemons are turned on, error messages are processed in the two daemons and one of the daemons simply crashes. After some research in DTN2 forums, it was realized that the problem was due to a bad implementation of this routing protocol.

Another problem was found when using the flood routing protocol in all scenarios presented previously. In the scenarios using two nodes (figures 4.8 and 4.10), a problem was found in the following case: when the two nodes are not connected and a bundle is ready to be sent in PC A to PC B (final destination), the bundle is sent only if it is the first bundle to be sent since the daemon was started. The transfer of a second bundle without restarting the daemon does not happen; in other words the two PCs are connected but the bundle is not sent to PC B.

When the scenario has three nodes (figures 4.9 and 4.11) something similar occurs: a first bundle is sent from node A to node C (final destination) through node B (A and C are never connected with each other) with no problems like in the previous case (two nodes). When a second bundle is sent again from node A with node C as destination, the bundle is only sent to node B and never reaches the final destination node C. In order to be possible to send a bundle again from A to C, it is necessary to restart the daemon.

The other routing protocols did not present problems in the several scenarios mentioned

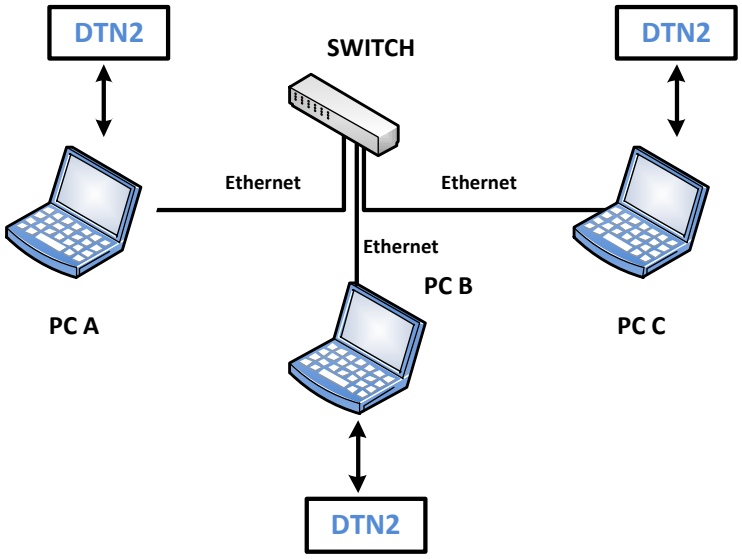

Figure 4.9: Three PCs running DTN2 connected via Ethernet

before.

4.3.2 DTN2 on Boards

After making several tests using PCs, DTN2 is evaluated using the boards that are responsible for the vehicular communication, with IEEE 802.11p implemented in our group [86].

The implementation was installed on the boards which have *buildroot* as OS. This OS is a set of Makefiles and patches that makes it easy to create a complete embedded Linux system [85].

From the several tests performed with DTN2 on the boards, the same conclusions apply compared to the tests on PCs. The first experiment scenario has two boards and it is shown in figure 4.12. The boards trade bundles like it happened with PCs, concluding that the communication with the standard 802.11p using DTN2 had no problems.

The tests with 3 nodes were also tested with the several routing protocols. The results were identical: the problems found using PCs were also detected using the boards. Since the boards are also static, it is expected a similar behaviour comparing to PCs.

In this sense, figure 4.13 represents the scenario with mobility, to evaluate DTN2 in a dynamic scenario.

Board A is ready to send a bundle and waits until a connection is set. Board B starts to move towards board A to perform the "mobility" until the contact between the two nodes is reached and the bundle can be sent. The routing protocol used in this scenario is DTLSR, since PROPHET and flood routing protocols have shown problems in the previous scenarios.

After several tests using this scenario, the conclusion was that this implementation was not suited for high mobility environments due to its behaviour when confronted with a slight

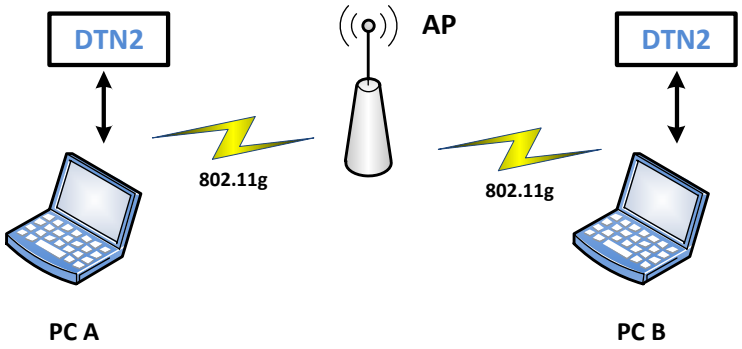

Figure 4.10: Three PCs running DTN2 connected via Ethernet

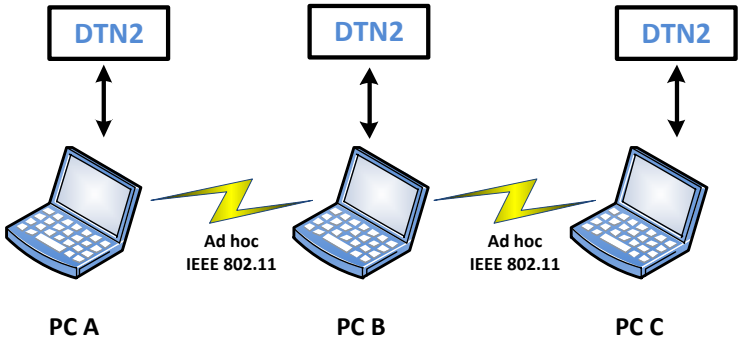

Figure 4.11: Three PCs running DTN2 connected via ad hoc

mobility situation. In some tests when the two nodes met, one of the daemons crashes and the bundles are not sent, but in some tests no problems occurred. Due to these facts, this implementation did not seem to be well suited to use in a high mobility scenario, which is the case of a VANET.

4.3.2.1 Detected Problems

The problems detected in 4.3.1.1 were also detected when using the boards. The new scenario including the slight mobility allowed to conclude that the implementation is not robust to high mobility environments.

The synchronization between the nodes is an important feature that must be respected because, when the nodes are not synchronized, there are problems between the transfer of bundles, related to the expiration time. The nodes synchronize themselves using the local host system clock as it was mentioned in section 3.3.1.6.

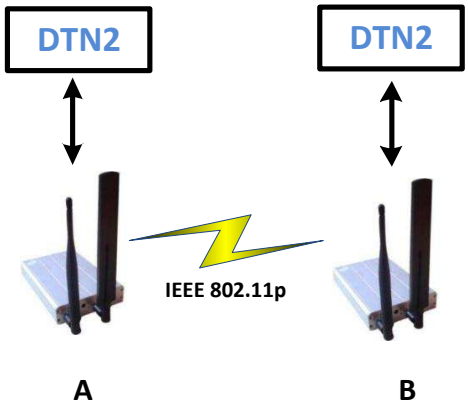

Figure 4.12: Two boards running DTN2 connected via standard IEEE 802.11p

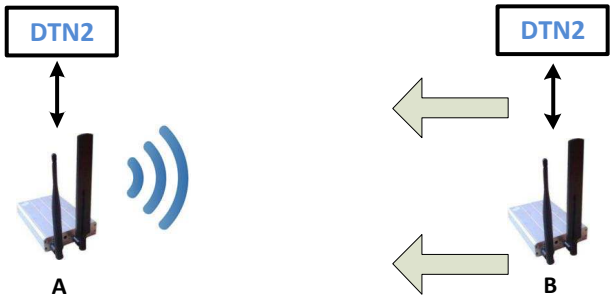

Figure 4.13: Two boards running DTN2 using standard IEEE 802.11p, movement scenario

4.4 IBR-DTN

After testing DTN2 implementation using PCs and boards, and finding about its problems, more specifically, its performance in dynamic scenarios, we decided to test another DTN implementation.

The new DTN implementation used is IBR-DTN version 0.8.0 (2012-06-01) (after some time the used version was 0.9.0), because it is suited to work on embedded systems using thin libraries, contributing this way to energy and computation costs reduction. Moreover, it has a functional implementation of PROPHET routing protocol. It is much lighter than DTN2: the size of applications and the attached libraries are much smaller and the RAM consumption is considerably lower [88].

4.4.1 IBR-DTN on PCs

Several tests are also performed using IBR-DTN, including the scenarios tested in DTN2. It was relatively easy to understand the incorporated applications of IBR-DTN, since it is

similar to DTN2.

The first experiment implemented with the PC's has three nodes represented in figure 4.14.

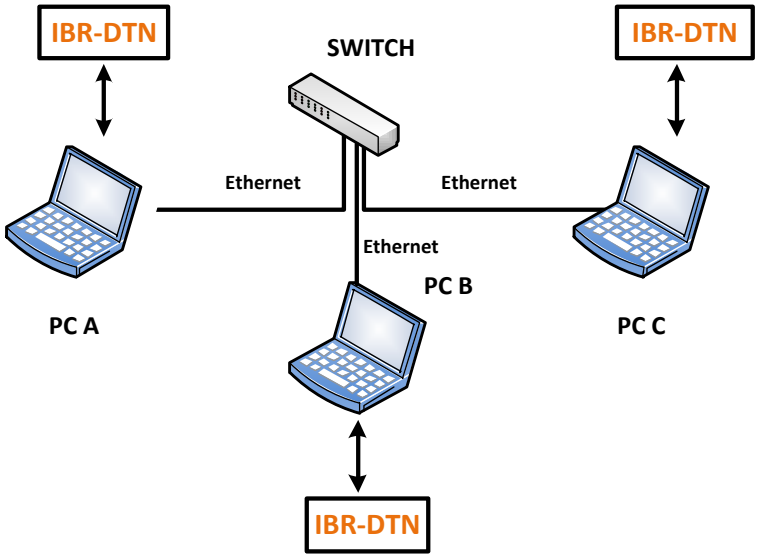

Figure 4.14: Three PCs running IBR connected via Ethernet

In this scenario, node A is never connected with node C (the same principle as in figure 4.9), and the transfer of bundles is from source A to destination C, using B as relay. The applications used to inject bundles in the network were *dtnoutbox* and *dtnsend*, and to receive the bundles were *dtninbox* and *dtnrecv*, respectively. *dtnsend* sends a specific file given by its location, while *dtnoutbox* moves all the contents in a specific folder in the source to the destination node. *dtnoutbox* needs as input arguments:

- destination node name;
- location of the folder to move its contents;
- application name, the transfer has a specific name set by the user.

dtnsend needs a destination name (containing also the application name) and the file to send.

All the routing protocols using the IPND are tested with apparent success using three nodes: flood, epidemic and PROPHET routing algorithms. All routing algorithms are capable of delivering the bundles from A to C through B, and do not suffer from the problems detected in DTN2.

4.4.2 IBR-DTN on boards

The IBR-DTN implementation is installed on boards in buildroot OS, to evaluate IBR-DTN behaviour in different scenarios with nodes communicating via standard IEEE 802.11p.

The installation of the implementation in buildroot had some problems because this OS was built to operate in devices that usually have low storage and processor capacities. The boards used do not differ from these characteristics and the OS has the minor number of libraries to make it as light as possible, which complicates the process of installing implementations which have dependencies on several libraries which is the case of IBR-DTN.

Several problems arose regarding the compilation of IBR-DTN on boards due to missing libraries needed by the implementation, wrong location paths (where to find libraries), etc. All these problems made the process of compiling IBR-DTN on boards relatively complicated.

dtnoutbox and *dtninbox* use the *TAR* function to compress all files on an specific folder to send as a bundle through the network and to decompress the files at the destination node. Due to its properties, it is only possible to use these applications after installing *GNU TAR* [89] function on the boards.

4.4.2.1 Tests

A set of different scenarios were developed to test IBR-DTN on boards, incrementing the number of nodes and changing the conditions on each scenario.

The MAC WAVE sub-layer operates using multi-channel for transmission. There are two types of channels specified in the standard, Control Channel (CCH) and Service Channel (SCH). CCH allows the devices to trade WAVE Short Message (WSM) messages without the need of pre-association; this way the devices can send and receive alert messages considered "urgent" instantaneously, while SCH is used to send and receive IP and WSM data.

Three possible ways to access the channel [87] were tested on the scenarios using the boards:

- Continuous Access: in this access, the device stays on CCH continuously and a request is sent to access the SCH when it is necessary to be used.
- Alternating Service Channel Access: in this situation, there are defined intervals for CCH and SCH alternating with each other.
- Extended Service Channel Access: this type allows the communication on SCH during several time intervals without the need to change to CCH, improving the bandwidth usage for services transfers.

Two nodes

The first scenario created with boards was the simplest, composed by two static nodes connected via IEEE 802.11p, illustrated in Figure 4.15.

Bundles were injected (using *dtnoutbox* and *dtnsend*) in the network from node A to B and vice versa, and no problems were detected, all bundles were delivered to both destinations.

Regarding the use of the Alternating Service Channel Access and Extended Service Channel Access, the implementation did dot show any problems with the constant switching of channels.

Another test performed with two nodes was the same one observed in figure 4.13, but this time using IBR-DTN, figure 4.16.

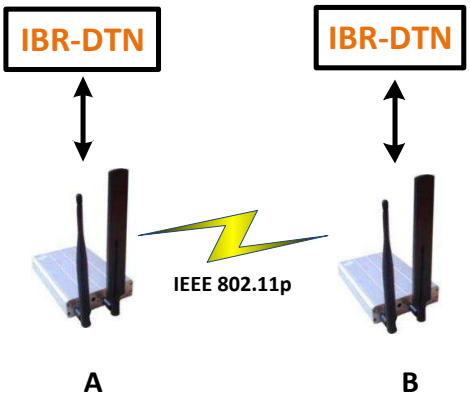

Figure 4.15: Two boards running IBR connected via standard IEEE 802.11p

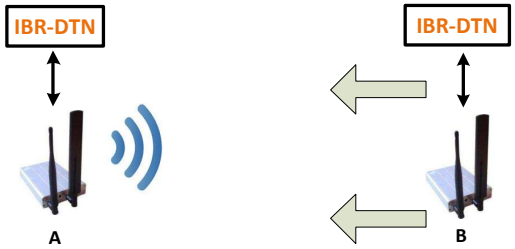

Figure 4.16: Two boards running IBR-DTN using standard IEEE 802.11p

Contrary to what happened with DTN2, IBR-DTN did not show problems with this mobility factor. The bundles were ready to be sent from node A as soon as B was on the range and, when that happened, the bundles were transferred with no problems. This process was repeated several times with PROPHET, epidemic and flood routing protocols, to ensure that it was stable.

Three nodes

The scenario developed with three nodes is shown in Figure 4.17. The boards were placed within the Instituto de Telecomunicações assuring that board A is only connected to board B, board C only connected with B, and board B connected with both to make sure that no connection is established between boards A and C. The boards were placed in different rooms within the building to respect the previous conditions of the connections between the nodes.

Bundles were sent from source A with board C as destination, and traversing board B. The store-and-forward mechanism worked well, since the node B receives the bundle from A, stores, and then transfers it to the final destination, board C. Board B only starts to transfer a bundle to C when it stops receiving it from A (store-and-forward mechanism).

A few implementation problems were found after several tests using this scenario, which

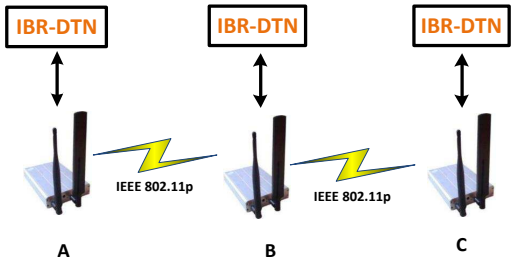

Figure 4.17: Three static boards running IBR-DTN connected via standard IEEE 802.11p

will be explained in section 4.4.3.

Four nodes

A scenario composed by four static nodes was also created to better understand how IBR-DTN operates, illustrated on Figure 4.18.

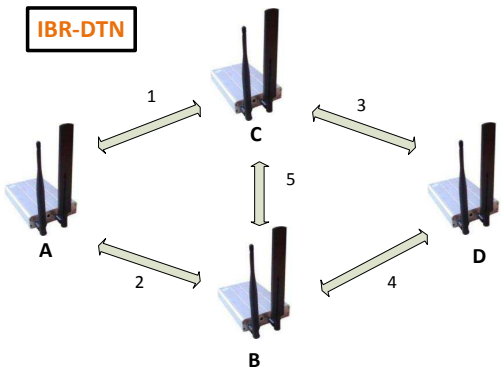

Figure 4.18: Four static boards running IBR-DTN connected via standard IEEE 802.11p

In this test, all links represented in figure were up and a bundle was ready to be sent from node A to node D as destination. The bundles are first transmitted to nodes B and C and, after then, to node D.

Node D received two bundles (from B and C) and one was deleted (the last one arriving to the node). After that, nodes B and C eliminated the bundles, due to PROPHET Acknowledge (ACK), since node D was the final destination of the bundles sent.

Other experiments made with four nodes allowed to conclude that the process of fragmentation is done node by node. Using an example, this means that it is not possible to send 30MB out of 70MB of a bundle through node B, and then send the remaining 40 MB through node C to be reassembled at node D. If a fragment of a bundle is sent through node B or C, the remaining part has to be transmitted to the same node in order to be reassembled in node D.

4.4.3 Problems Found and Solution

During the tests carried out, several problems were found compromising the good functionality of the implementation. These faults were concerning about PROPHET acknowledgement and fragmentation of bundles, which in some specific tests, with previous scenarios, it turn out impossible to deliver the intact bundle to its destination. In the next subsections, the faults will be explained as well as its detection and solution.

4.4.3.1 PROPHET Acknowledgement

When PROPHET is being used as the routing protocol, a PROPHET ACK is sent whenever a bundle reaches its final destination, confirming the reception of the bundle in question. When a node receives this PROPHET ACK, the bundle targeted by the ACK is eliminated from the storage.

The first problem encountered in IBR-DTN was related with these PROPHET ACKs, that in some cases were provoking the deletion of bundles that did not have been completely received by the final destination (only fragments of it).

This issue was detected running a specific test with the scenario presented in figure 4.18 (with PROPHET routing protocol being used at all nodes). The arrows represent the several connections between the boards. A can communicate with C and B; B can connect with A, C and D; C can communicate with A, B and D; and D can communicate with B and C. These five links were up or down simulating specifics scenarios.

The error was detected realizing the following steps:

- `first`: a bundle was ready to be sent at node A with destination node D, and all links were down;
- `second`: link 1 was turned up to a certain period of time (not enough to transmit the whole bundle from node A to C) and then turned down;
- `third`: link 3 was turned up until the entire bundle stored in node C was transmitted to node D, turning then the link down;
- `fourth`: link 2 was turned up until the entire bundle stored in node A was transmitted to node B, and the link is then turned down;
- `fifth`: link 5 was turned up (with no specific time interval) to check the behaviour between the nodes B and C.

After the second step, node C had a fragment of the entire bundle (from node A) and, in third step, this fragment was transferred to node D, from C. Since node D was the final destination of the bundle, a PROPHET ACK was generated and sent to node C, and the fragment that was stored in node C was deleted. Then, in the fourth step, the entire bundle stored was transmitted to node B and stored.

When the fifth step was taken, the whole bundle stored in node B (transmitted on the previous step) was deleted due to a PROPHET ACK sent by node C. After this, links 1, 2 and 5 were turned on and the original bundle stored in node A was also deleted due to a PROPHET ACK.

Concluding, the bundles were deleted before the destination received the entire bundle. At the end of the experience the only remaining bundle was the fragment at the node D (incomplete).

This issue was addressed to the ibr-dtn mailing list [90] and it was solved in a few days. Since this implementation is under many changes, there is a version available in a repository (version 0.9.0) which is a temporary version always in change due to bug's solving.

This version (0.9.0) was compiled on the boards and the same test was realized; with these changes, the problem was solved. From this point on, the version of IBR-DTN used was 0.9.0 instead of 0.8.0.

4.4.3.2 Fragmentation of bundles

The fragmentation process in a vehicular environment is an extremely important feature due to its dynamics, making the time of contacts between nodes relatively short or even unexpectedly broken. Fragmentation provides the resume of bundle's transfer. This means that, when a connection is broken, if the nodes contact each other again, the process of transferring a bundle is not restarted, instead it transfers what was not in the first time (if the contact time is not enough, another part/fragment is transmitted). This mechanism is also responsible for reassembling the bundle's fragments at the destination node.

The problem was found in one of the experiments made with scenario from Figure 4.15. Node A was running a script in bash controlling the connectivity between the two nodes by setting the wireless interface of the boards up and down, making that the two nodes only communicate during pre-determined intervals defined in a script.

A bundle with a considerable size of 50MB was ready to be sent in node A (wireless interface of A down) to node B. The script was set to allow communication intervals of 10 seconds between the two nodes and intervals of 30 seconds of disconnection. Since the average throughput was about 1.1/1.2 MB/s (continuous channel with 23 Mbit/s of Bitrate and 23dB of transmitted power), the whole bundle could not be transmitted in one single interval of connection, allowing to divide the bundle into fragments in the destination (node B). What happened was that fragments were continuously being transferred to node B infinitely, and the transfer of the bundle never ceased.

After some debugging, the reason for this behaviour was found by analysing the size of bundles fragments received at node B. At some point of the experiment, the sum of all fragments sizes in node B was much higher than the size of original bundle in node A. Since fragments were continuously being transferred, the script was turned down and the wireless interface of board A was always up from this moment on. After some time, the transfer of fragments stopped and it was observed that the size of the two last received fragments in node B summed had the size of the original bundle in node A. Concluding, the error was in the calculation of fragment offset, because it only calculates the offset of the current fragment being transferred.

After a considerable time analysing the code from IBR-DTN regarding the fragment module, the problem was solved by adding some code needed to have a functional fragmentation process. After these adjusts, the fragmentation process was completely corrected and the previous experiments, with pre-determined intervals of connection, were repeated but this time the transfer of bundles ceased and the sum of all fragments sizes was equal to the original bundle. All fragments were also reassembled at node B, forming the original bundle which was sent from node A.

4.4.3.3 Source Bundle Deletion

After solving the previous problems, another one concerning the deletion of bundles, due to PROPHET ACKs, was detected. This time, the error was detected testing IBR-DTN in the scenario from Figure 4.17. The tests consisted in sending bundles from board A with C as final destination; the communication between nodes is the same as it was explained in the previous subsection 4.4.2.1, regarding the scenario with three static boards.

The same process was used to control the connectivity between the two nodes, using shell scripting as in the previous subsection 4.4.3.2. Unlike the previous case, this scenario has three nodes and the shell script was only controlling the connectivity between boards A and B, which means that boards B and C were always connected. This way, bundles were sent from A with destination C, traversing B. The bundle was large enough for allowing the process of fragmentation due to the intervals of connection setted in the script.

The error was found while the several fragments of the entire bundle were being transferred from A to C through B, when at some point of the test, the bundle in board A was deleted due to a PROPHET ACK, before the complete transfer of the bundle. This is a problem since the bundle was deleted from its source without being completely delivered (only several fragments). This was happening because an incorrect creation of a PROPHET ACK was being received by node A to delete the bundle before its time.

Changes were made in the source code to correct this problem, and now the bundle is only deleted from the source when the entire bundle (all fragments) is delivered to its destination, and not when only several fragments are delivered.

After the changes, the test was repeated several times to make sure everything was working as it should. From that moment on, the bundle was never deleted from the source before its whole transfer was completed.

This way another problem found in IBR-DTN was corrected contributing to the efficiency of this implementation and to the results of this Dissertation work.

4.4.4 Data Log for Experimental Testbed

The previous detections and corrections allowed the beginning of the tests in experimental testbed to get results, but first it was necessary to get the required information from IBR-DTN, creating a data log suited to our experiments.

The results, which will be presented and analysed in the next chapter, will need the following parameters:

- time when a transmission of a bundle starts;
- time when a transmission of a bundle is interrupted;
- time when a transmission of a bundle is completed;
- time of each bundle which arrived at the node;
- size of each bundle which arrived at the node;
- name of destination node and contact node which received the bundle.

All these parameters were registered in a file, which is created on each node running IBR-DTN. After the experiments, these files have the necessary information above, allowing the calculation of certain metrics to evaluate IBR-DTN using the boards.

All these informations stored in the files, generated by IBR-DTN, were implemented in the source code, and examples of this output file is shown on Figure 4.19, where all the relevant information is revealed.

```
1373821420.165798 Bundlereceived(local) [427136619.0] drivein101 teste1 drivein253 teste1  10496075
1373821453.286597 SENT [427136619.0] drivein101 teste1 drivein250
1373821458.223803 STOPED [427136619.0] drivein101 teste1 drivein253 teste1 6606848 10496000 drivein250
1373821468.292320 Bundlereceived [427136619.0.0] drivein101 teste1 drivein253 teste1 drivein101  6037584
1373821473.303150 singletonbundledelivered [427136619.0.0] drivein101 teste1 drivein253 teste1
1373821473.311349 transferofbundle [427136619.0.0] drivein101 teste1 drivein253 completed
```

Figure 4.19: Example of Information Present in Data log

As it can be seen, before each line there is always a timestamp, which was set to have the exact time information about when a specific instruction was executed. The first line in Figure 4.19 gives the information that a bundle was received (local which means that the bundle was created by the current node, creating the data log) in the node. The name of the node which created the bundle is given in the fourth word of the line (drivein101), and the name of the destination node of the bundle is in the sixth word (drivein253). Whenever a bundle is created to be sent, there is always an application name responsible for that event, in this case its name is "teste1" in the fifth and seventh words. The information about the size of bundle is also given whenever a bundle is received.

The second line notes the time a bundle begins to be transferred to another node, giving the two node's name information responsible for the transfer. The emitter ("drivein101") and the receptor ("drivein250") are in the fourth and sixth words, respectively.

The third line informs that a transfer of a bundle was interrupted, which means that the bundle was not completely transferred, only a fragment. The size of the fragment transferred is given as well as the complete bundle size, in MB, 6606848 and 10496000, respectively.

The fourth line is very similar to the first one, the difference is that the bundle received is not local (was not generated in the node) which means that the bundle was received from other node given by the last word in line, "drivein250".

The fifth line is written in file whenever a bundle/fragment is delivered to its final destination, informing about the application name, source and destination node.

The last line informs that a bundle was transferred to another node, its destination or not. In this case, the bundle was transferred for node with name "drivein253".

To have all this information, it was necessary to explore, modify and create code in IBR-DTN in a way that the information required was available. It was necessary to find in the code the exact moment that a bundle was beginning to be transferred to another contact, and to register the time, contact node, etc. It was also indispensable to have the information about the exact time a transfer was interrupted, in order to calculate throughputs, transfer times; some parts of IBR-DTN data log were used because of its usual information, but it was always needed to include extra components on each line of data log to get the required metrics.

The created log was used to generate the final results presented in the next chapter, by using other means of scripting to make use of data log information.

4.5 Summary

The following chapter presented to the reader the taken steps before testing scenarios to get results about IBR-DTN's performance. Two DTN implementations were used and tested in PCs and boards, and it was concluded that IBR-DTN was the best solution to use in a real vehicular environment.

Implementation problems were founded and solved, improving its efficiency and also contributing to a correct operation of IBR-DTN.

The next chapter will make use of IBR-DTN to analyse the experiments in laboratory and in real vehicular environments.

Chapter 5

Results

5.1 Introduction

This chapter presents the tested scenarios as well as the experiment results.

Section 5.2 presents the parameters used on the tests, in the laboratory and in the real testbed. These parameters define the bit rate, channel access and transmission power.

Section 5.3 presents the scenarios and results from the tests in the laboratory, while section 5.4 presents the real testbed scenarios and the corresponding results. In real environments, for some scenarios different velocities are tested to evaluate the IBR-DTN performance for different velocities.

Since these were the first tests on a real vehicular environment, the network is composed by a few number of nodes (two and three nodes). The scenarios tested were used to evaluate the fragmentation, transmission capacities of IBR-DTN using two nodes: one vehicle and one RSU, and two vehicles. The other tested scenario with three nodes (one vehicle and two RSUs) was used to evaluate the store and forward performance of IBR-DTN in a real vehicular environment.

All the performed tests were realized using PROPHET as the routing protocol.

5.2 Used Parameters

The scenarios tested to get the experimental results are divided in two groups: tests in laboratory and tests real vehicular testbed.

Parameters	**value**
Channel number	180
Defined TxPower	23 dBm
Channel Access	continuous
Defined Bit Rate	23 Mb/s

Table 5.1: Used Parameters for laboratory tested scenarios

For the scenarios tested in laboratory, the parameters used are presented in table 5.1. The continuous channel, referenced in 4.4.2.1, is used in the laboratory scenarios because of the GPS signal, since in indoor tests the reception of GPS signal is very weak. The larger the bit rate is, the smallest is the range of communication. Since the boards are within the Instituto de Telecomunicações (IT), it was necessary to make sure that some boards were not in the range of others as it is going to be explained later; because of this fact the bit rate was set to its maximum.

Parameters	**value**
Channel number	180
Defined TxPower	23 dBm
Channel Access	Extended
Defined Bit Rate	6 Mb/s

Table 5.2: Used Parameters for real testbed scenarios

The parameters used in the real testbed are different, comparing to the laboratory tests, and are presented in table 5.2. In these tests the access to the channel used is the extended channel (4.4.2.1), as it is the one used on the real testbed implemented on the taxis and buses in Porto. These tests are performed in a way to be more closed to a realistic vehicular environment. The extended access was set to six time intervals dedicated to the SCH and one time interval dedicated to CCH. The combination of TxPower and Bit Rate was set to the presented values, since it was studied and proven that these two values give the best commitment between bandwidth and range of communication [91].

5.3 Laboratory Scenarios

In order to evaluate IBR-DTN's performance, several tests were performed within the IT with the boards statically placed in the building corners. A couple of scenarios were performed, using two and three boards.

In these tests, bundles with different sizes are sent from a sender node to a destination one. Connection and disconnection intervals between nodes are set in order to test the fragmentation performance, and this way, evaluate the behaviour of IBR-DTN.

5.3.1 Two Boards Scenario

The first deployed scenario is composed by two boards illustrated in figure 5.1.

In this scenario the two boards are closed to each other and connected via IEEE 802.11p as it can be seen in the figure. Link "1" is not always up and the time interval of connection between the two nodes varies on each bundle sent. The disconnection time interval was always set to 30 seconds.

The size of the bundles sent, from board A to board B, were of 10, 20 and 50 MB. For each of them, several connection time intervals were set (each connection time interval is a different test):

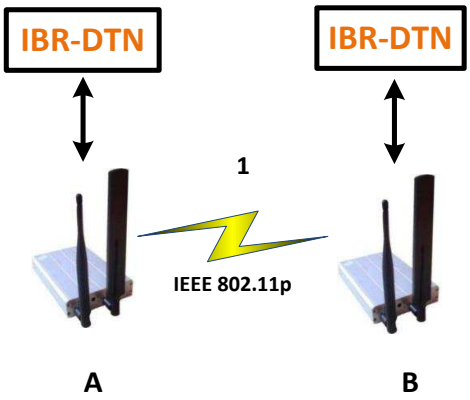

Figure 5.1: Two Boards Scenario

- **10 MB**: 5 and 10 seconds;
- **20 MB**: 5, 10, 15 and 20 seconds;
- **50 MB**: 10, 15, 20, 25, 30, 35, 40 and 45 seconds.

To better explain the operation, we detail the test of the 50 MB bundle being sent from one node to another with the connectivity interval of 10 seconds. The bundle starts to be sent during 10 seconds (link "1" up), then stops 30 seconds (link "1" down); after this disconnection interval, it starts to send the bundle again during 10 seconds (link "1" up) and so on until the bundle is completely sent.

The boards are connected to Ethernet switches allowing to check and manage the experiments on the computer also connected to a switch, accessing to the boards via Secure Shell (SSH). Each test is repeated 10 times and the results are the average of these 10 repetitions, with 95% confidence intervals.

Figures 5.2 and 5.3 illustrate the results from these tests in this scenario composed by two nodes.

Figure 5.2 illustrates four graphs:

- **(a)Bytes per second**: represents the amount of bytes sent on each second in average. The disconnection time interval (30 seconds) after each connection time interval enters in these results. It means that these values are calculated using the time since the bundle is sent from the sender until it is fully received by the destination (whole bundle), and the size of the bundle.
- **(b)Number of Fragments**: represents the number of fragments needed to send the entire bundle to the destination. This means that the number of fragments decreases with the increase of time of connection intervals.
- **(c)Transfer Time**: represents the time since a bundle is sent from the sender until it is fully received by its destination. It can be seen that its behaviour is inverse to the graph (a). When the transfer time increases, the bytes per second (a) decreases.

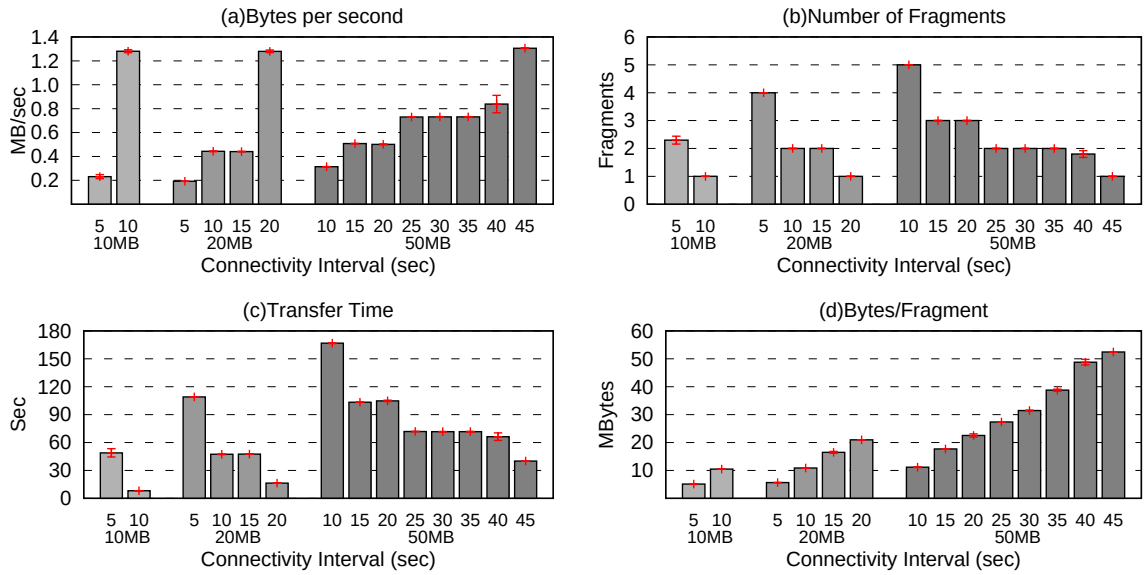

Figure 5.2: Results For Two Boards Scenario

- **(d)Bytes per Fragment**: this graph illustrates the amount of data (bytes) sent on each fragment on each test. The average of these values is calculated using the weighted average represented on equation 5.1:

$$Bytes_{fragment} = \frac{x_{frag_1} \times t_{frag_1} + x_{frag_2} \times t_{frag_2} + ... + x_{frag_n} \times t_{frag_n}}{t_{frag_1} + t_{frag_2} + ... + t_{frag_n}} \tag{5.1}$$

  where:

  - $Bytes_{fragment}$: average transferred bytes per fragment
  - x_{frag_n} : transferred bytes on fragment n
  - t_{frag_n} : time of transfer of fragment n

The graph in figure 5.2(a) represents the amount of bytes sent per second, which is given by the complete size of the bundle divided by the time since it is sent (from node A) until it is completely received by the destination (node B). As it can be seen, in all files, the tendency of graph's bars is the same; the increase of the connectivity interval increases the amount of data sent per second because less fragments are needed to send the overall file, while the connectivity interval increases (as it can be seen in graph (b) from figure 5.2 the number of fragments needed to send the file decreases with the increase of the connectivity interval). Graph (c) shows the decrease of the transfer time with the increase of the connection time intervals, and so graph (a) is related with graphs (b) and (c). When the file is sent in only one fragment, the value of the graph corresponds to the actual throughput of the file, which is illustrated in figure 5.3 for the same cases (file 10 MB for 10 seconds; file 20 MB for 20 seconds; and file 50 MB for 45 seconds). When the number of fragments is the same (for each

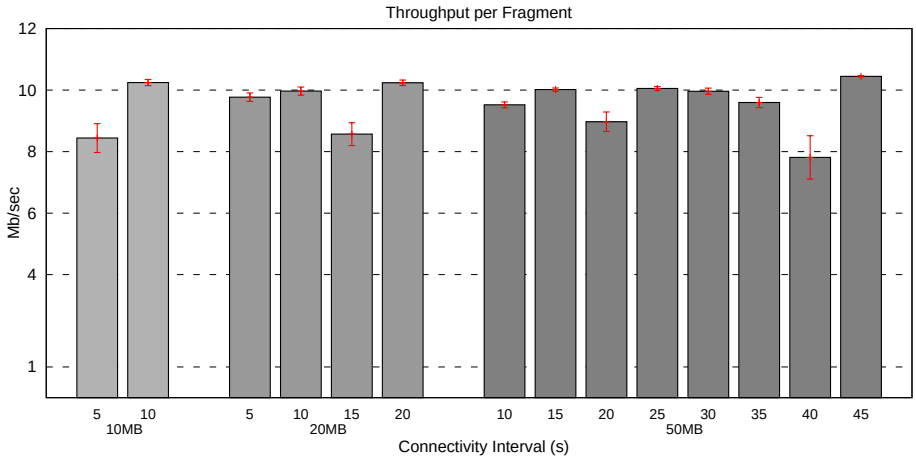

Figure 5.3: Throughput per Fragment

file) the values of bytes transferred per second is essentially the same because the transfer time for these cases is also very similar (graph (c)).

The graph (b) shows the number of fragments necessary to send the whole file from node A to B, and it is easy to verify that the number of fragments for each file size decreases with the increase of the connectivity time intervals. In some cases, the number of fragments stays constant because the increase of connectivity time interval was not enough to send the file in less fragments. The number of fragments needed to send a file increases with the increase of the file size for the same connection time intervals. In some cases the average is not exactly an integer number (case of 50 MB for 40 seconds, and the case of 10 MB for 5 seconds) because in some of the 10 tests, the number of fragments sent was different from the others. For the 50 MB for 40 seconds, in one test the file was completely sent in one fragment, while in the remaining 9 tests two fragments were needed.

Graph (c) is related to graph (b), and its behaviour is exactly the same. The increase of time intervals decreases the number of fragments, which makes the transfer time decrease in the same way. The transfer time increases with the increase of the number of fragments because of the disconnection time intervals between the connection time intervals. More fragments mean more disconnection time intervals, contributing to a higher transfer time.

Graph (d) represents the weighted average amount of bytes transferred per fragment. This means that if a fragment is sent during more time than another fragment, its weight will be higher on the average calculation (equation 5.1). This way it is possible to see that the amount of bytes transferred increases with the increase of connection time intervals. This way this graph illustrates the capacity of the link to send data with the increase of the connection time intervals. The increase of connection time intervals makes the weight of fragments transmitted in more time, more influential, more time to transfer more data. For the example of the 50 MB file, for 25, 30, 35, 40 seconds the connection time intervals increases making the weighted average of bytes sent per fragment increase as well. When the bundle is sent in only one fragment, the bytes transferred correspond to the actual size of the file.

The figure 5.3 represents the average throughput per fragment. It can be seen that the maximum throughput is achieved when a bundle is sent in only one fragment and it is similar to the three files, with the maximum mean throughput of 10.4 Mbits/sec. There are some

variations in the values of the throughput for the cases where bundles are transmitted in more than one fragment, but the largest breaks on the mean throughput are noticed in specific cases of connection time intervals (10 MB for 5 seconds; 20 MB for 15 seconds; and 50 MB for 20 and 40 seconds). The transmission of data is over TCPs connections, which means that if the length of data to send is small, the throughput will be lower due to the transmission window of TCP. The cases mentioned before can be justified looking into the graph (b) and (d) from figure 5.2. The cases where the breaks on throughput happen correspond to the last connectivity interval with that number of fragments. The next connectivity interval will have less fragments. This means that the last fragment for these cases has small size contributing to the break of the throughput. From graph (d) it is possible to prove it also by analysing the amount of data represented for the previous cases. Given a specific example, for the case of 50 MB for 20 seconds of connectivity intervals the number of fragments is three. In graph (d) the amount of weighted average data per fragment was approximately 23 MB which means that, in average, the first two fragments sent 23 MB each, and the third and last fragment had approximately 4 MB, contributing this way to the break on throughput. The case of 50 MB for 40 seconds of connectivity intervals has the biggest break, since the second and last fragment is really small.

5.3.2 Three Boards Scenario

The next deployed scenario is composed by three boards, illustrated in figure 5.4.

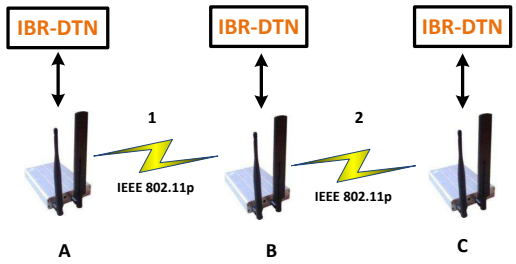

Figure 5.4: Three Boards Scenario

In this scenario, the three boards are placed within the IT building in a way that board A can only communicate with board B, and board C can only communicate with board B. Link "1" is not always up and the time interval of connection between the two nodes varies on each bundle sent. The disconnection time interval was always set to 30 seconds, like in the previous scenario with two boards. The link "2" is always up. The files used and the connection intervals are the same of the previous scenario. The three boards are all connected to switches to be possible to manage and control the experiments in a PC. In this scenario, bundles are sent from board A to board C, with board B acting as a relay since no connection between A and C exists. The bundles are received by board B, from A, stored and then sent to board C. This way the store-and-forward mechanism is tested.

The results from the experiments in this scenario are represented in figures 5.5 and 5.6.

All the results were taken exactly the same way as in the previous scenario. This scenario has two links, and the results of the number bytes per fragment (d) and figure 5.6 are the

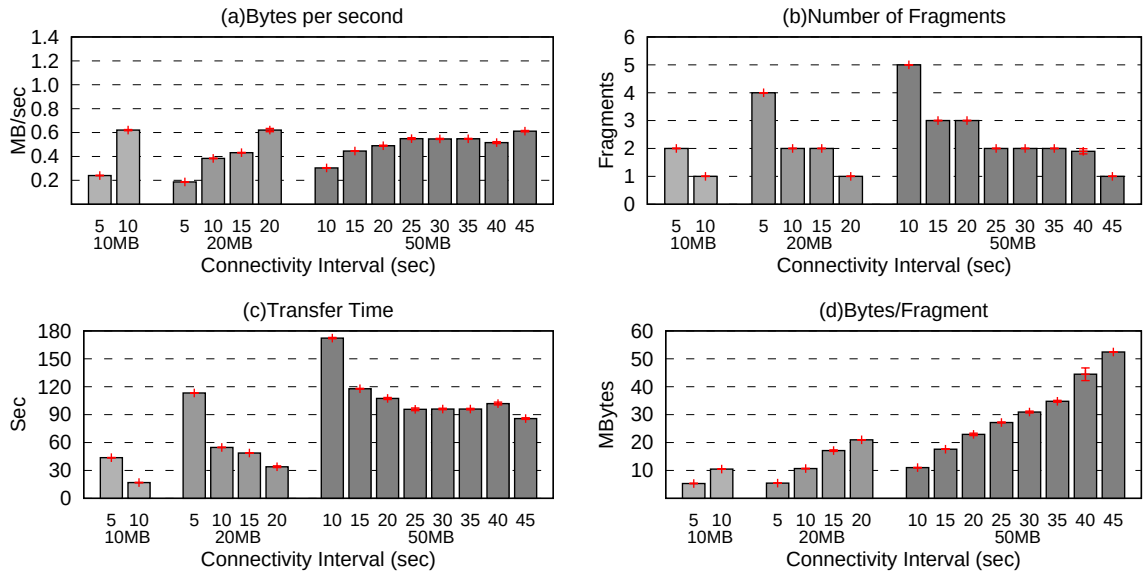

Figure 5.5: Results For Three Boards Scenario

average of the two.

The graphs behaviour is similar to the others on the previous scenario, but there are several changes that need to be explained. The first graph (figure 5.5 (a)) presents the expected behaviour, but there are two situations out of this expected behaviour, which can be explained with the external conditions that the boards were under of. Since they were placed in the building away from each other, external factors like people's movements affect the tests. This is probably the case of the 50 MB file for 40 seconds, where the value of bytes per second is a little lower than what it is expected. It shall have approximately the same values of 25, 30 and 35 seconds for the same file. The maximum value for this graph was achieved, as expected, when the file is sent with only one fragment, approximately 0.61 MB/sec (sensibly half comparing to the previous scenario, which is natural since the file is traversing two links).

The graph (b) represents the mean number of fragments which was exactly the same of the last scenario.

In graph (c) it can be seen that the bundles sent in only one fragment have the double time of transfer comparing to the previous scenario, since they need to traverse two links. The behaviour of the values is identical to the previous scenario, but for 50 MB for 40 seconds of connectivity time interval, the value is a bit higher than what it should (related to what happened and explained in graph (a)).

In graph (d), the increase of the connection time intervals increases the amount of weighted average of bytes transferred per fragment, just like in the previous scenario. There are slightly differences when comparing the two scenarios. For the case of 50 MB for 35 and 40 seconds, less bytes are transferred in average which is related to the break on throughput, which is going to be justified next.

Figure 5.6 represents the average throughput per fragment on three boards scenario. The maximum value achieved for this metric is a little lower than in the previous scenario. This happens because, in the present scenario, the boards are not next to each other like in the

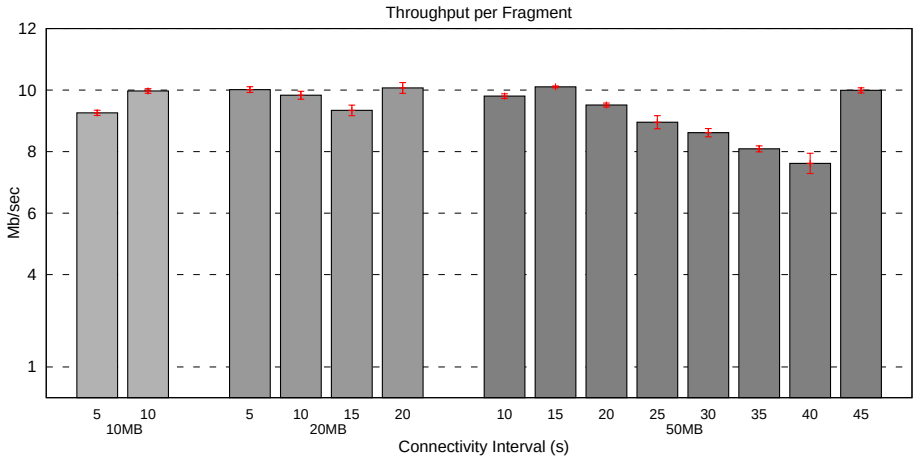

Figure 5.6: Throughput per Fragment

previous one. They are separated in different parts of the building, and the external factors mentioned before can harm the connection a little, contributing to these results. In the 50 MB files there is a break on throughput continuously from 25 to 40 seconds. The average throughput starts to drop more on each of them. This fact is due to the transmissions of different fragments at the same time: a new fragment from board A to B, and the previous fragment being transferred from B to C. The increase of the connection time intervals increases the time these two transmissions happen, at the same time contributing to the drops on the average throughput. This situation only happens for the mentioned cases, because in the others, when the new transmission starts, the previous one is already finished. These breaks on throughput per fragment contributed to the break on the average bytes transferred per fragment, mentioned before.

5.4 Testbed Results

The laboratory tests were the first step to evaluate IBR-DTN in a controlled environment to understand well its operation. Now, the next step is to test IBR-DTN in a vehicular environment, using real cars and RSUs. The conditions are highly different from laboratory tests, since in the street, with cars and RSUs, there is movement and external factors affecting the results such as other cars and the movement itself. These external factors may have influence in the results. One of the scenarios (two cars), which will be presented in subsection 5.4.2, is the most affected one. Several cars and trucks were passing in the area of the tests, and in this specific scenario, these external conditions provoked breaks in connections, shortening of contact times and the quality of the connection. Since all these factors are normal in a vehicular environment, it is a good sign that IBR-DTN, even with these barriers, had a good behaviour, but the results in some cases are a little dynamic.

Three scenarios were tested and will be presented and analysed next. In all of them, the specifications used were the same and defined in table 5.2.

5.4.1 One Car One RSU

The first vehicular scenario deployed is composed by two nodes, one static node (RSU) and one mobile node (car), illustrated in figure 5.7.

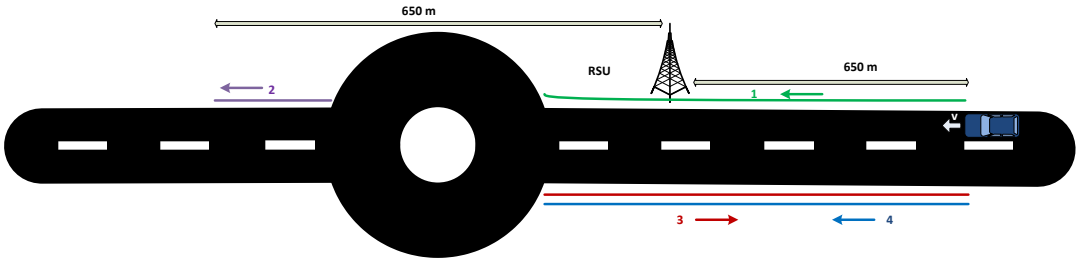

Figure 5.7: One Car One RSU

In this scenario several files are sent, one at each time, from the car to the RSU, since a computer is connected to the board on the car to manage the tests. Two velocities are tested for this scenario, 30 and 50 km/h. For each file sent, the tests are repeated 5 times and the values present an average of the 5 measurements, with 95% confidence intervals. For 30 and 50 km/h, 4 files with different sizes are sent in order to test the fragmentation process. For 30 km/h the files sent have 5, 10, 30 and 50 MB; for 50 km/h they have 5, 10, 20 and 30 MB. The car starts to communicate with the RSU at the specified velocity, and that velocity is maintained until the the two nodes stop communicating (fragmenting the file).

The graphs that will be presented next show the results taken from the tests on this scenario and present the following metrics:

- average time per fragment;
- average number of fragments;
- average throughput per fragment;
- average bytes sent per fragment.

The results for this scenario are represented in figures 5.8 (for 30 km/h) and 5.9 (for 50 km/h).

In figure 5.8 (for 30 km/h), it can be seen that the mean time per fragment varies between 50 and 60 seconds, approximately, when the fragmentation process is used (30 and 50 MB). For 5 and 10 MB, in average they do not need to be fragmented to be sent, and as it can be seen from graph (a), the time to send the 10 MB file is approximately the double to send the 5 MB one. The number of fragments (graph (b)) increases with the increase of the file size, as it is expected. The average throughput per fragment (graph (c)) varies between 1.1 and 1.5 Mb/sec. The increase of the number of fragments makes the throughput drop a little, and this has to do with fragments sent with low size, due to the conditions of the environment, as it will be explained later. The average bytes sent per fragment tend to stabilize between 10 and 12.5 MB, as it can be seen in graph (d). These graphs only show the average values;

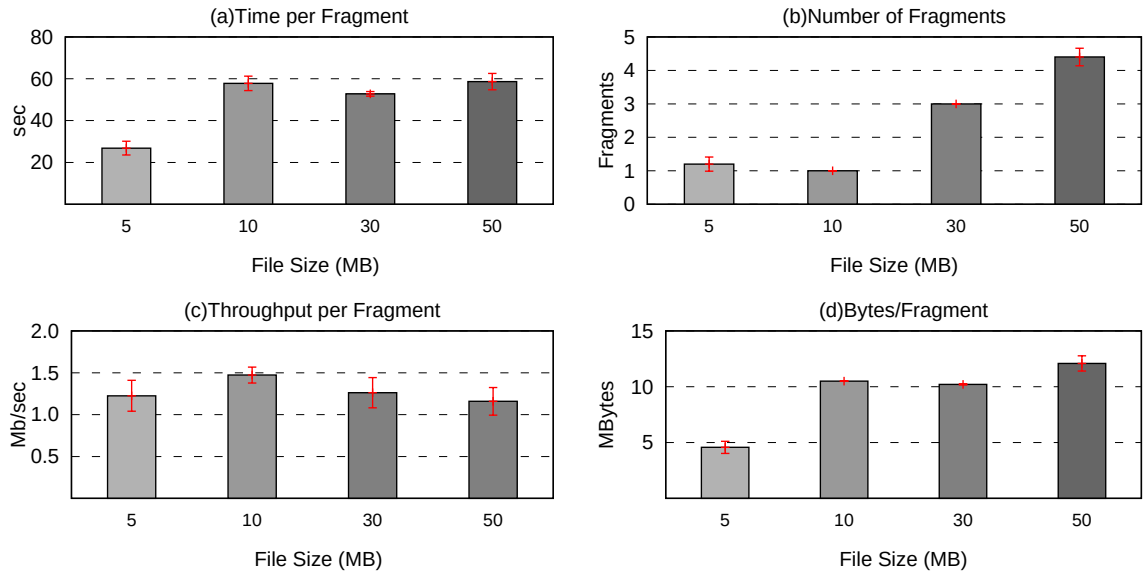

Figure 5.8: One Car One RSU Results for 30 km/h

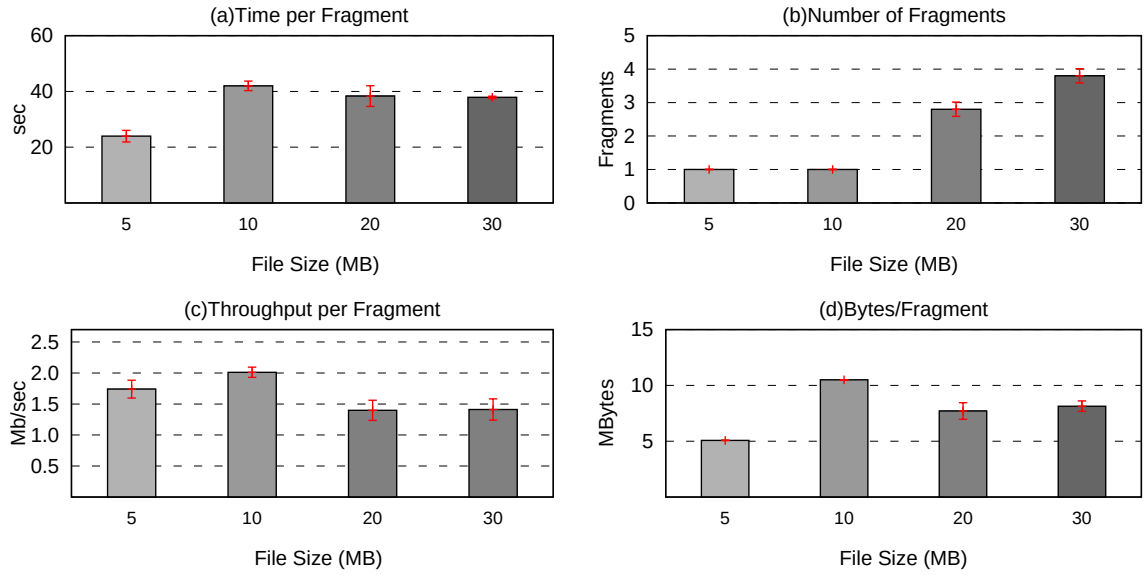

Figure 5.9: One Car One RSU Results for 50 km/h

to have a better knowledge of the experiment, a table with the average bytes sent on each fragment and their time (time of fragment) is presented in table 5.3.

Analysing table 5.3, it is possible to observe the amount of data sent on each specific fragment on average, as well as the time to send each fragment.

For the first fragment, for both 30 and 50 MB, the amount of data sent was approximately the same, and the time of data transfer (contact time) was in average 115 seconds. This first fragment corresponds to the trajectory made by the car illustrated in figure 5.7 as the number "1". This is the transfer that delivers more data as it can be seen in the table. The

	30MB		50MB	
Fragment	Trasnferred (MB)	Time (sec)	Transferred (MB)	Time (sec)
first	24,6	116	25,5	115
second	0,8	15	0,4	13
third	5,2	28	18,7	81
fourth	-	-	7,0	37

Table 5.3: Bytes Transferred and Time of Transfer - 30 km/h

second fragment corresponds to the trajectory number "2" after the roundabout. This second fragment has a low value of data in average, since this contact happens far way from the RSU (approximately 500 meters) which makes the connection weak and short. In the roundabout the connection with the RSU is lost due to the characteristics of the environment where the tests were made. The third fragment corresponds to the trajectory number "3" after the roundabout on the way back. For the case of the 30 MB file, this contact was enough to send the remaining data, but for the 50 MB it was necessary to transfer again using the trajectory number "4", which is the same as the number "1".

Figure 5.9 represents the results for 50 km/h, showing that the average time per fragment (graph (a)), when the file is sent with more than one fragment, is approximately 40 seconds, which is less than the average for 30 km/h. This is as expected, since with a high velocity the time of contact will be smaller. Since the contact time is smaller, the files sent, for the same number of fragments as for 30 km/h are now 20 and 30 MB, instead of 30 and 50 MB. In average, the 20 MB file is sent in three fragments, while the 30 MB file is sent in four fragments, as it can be seen in graph (b).

Analysing graph (c), once again the mean throughput per fragment is lower when the file is sent in more than one fragment. Comparing the mean throughput per fragment, it can be noted that the behaviour of the two graphs (graphs (c) in figures 5.8 and 5.9) is identical, but for the velocity of 50 km/h, the values for the mean throughput on each file are higher. This situation is due to the earlier detection of connection between the two nodes for the velocity of 30 km/h. This means that, since the car is moving slower, comparing to 50 km/h, the detection of a connection between the two nodes starts earlier when the connection is not very good. The bundle starts being sent earlier, but the throughput is very low since the connection is not good. In the case of the 50 km/h, the car is moving faster and the connection is better comparing to 30 km/h, making the throughput higher. The small difference between the two velocities in the mean throughput is due to this fact.

As it was expected, the graph (d) shows a decrease in the average bytes sent per fragment, since the contact time between the two nodes is smaller. The difference between the files 20 and 30 MB, and the file 10 MB is due to the second fragment sent on 20 and 30 MB (table 5.4), which makes the average of bytes sent per fragment decrease.

The table 5.4 shows the amount of data sent on each fragment (for the case of the files that are sent in more than one fragment), as well as the time of each transfer.

The first fragment on both files (20 and 30 MB) is approximately the same, and both have less data comparing to the case of 30 km/h, which is normal since the contact time between the two nodes is smaller. The maximum transfer time occurs for the first fragment (trajectory "1"), and it is of 67.3 seconds, in average. The second fragment, in this case, has more data in both files comparing to the previous case (30 km/h), and the time is expected

	20MB		30MB	
Fragment	Trasnferred (MB)	Time (sec)	Transferred (MB)	Time (sec)
first	13,9	67	15,2	64
second	2,3	12	2,6	14
third	6,1	31	10,6	54
fourth	-	-	2,8	14

Table 5.4: Bytes Transferred and Time of Transfer - 50 km/h

to be smaller as well. This happens because in one of the five tests (on both files) there was no fragment sent corresponding to the trajectory "2", which means that in those tests the file was sent with less fragments. This makes the average of bytes sent and time of transfer higher. The third fragment corresponding to trajectory "3" has lower time of transfer as it happens for the case of 50 MB file for the 30 km/h. The 20 MB file does not need to use the whole contact time of the third trajectory to send the file, as it can be seen in the time used by the third fragment.

5.4.2 Two Cars

The second scenario performed is composed by two mobile nodes (two cars) and it is illustrated in figure 5.10.

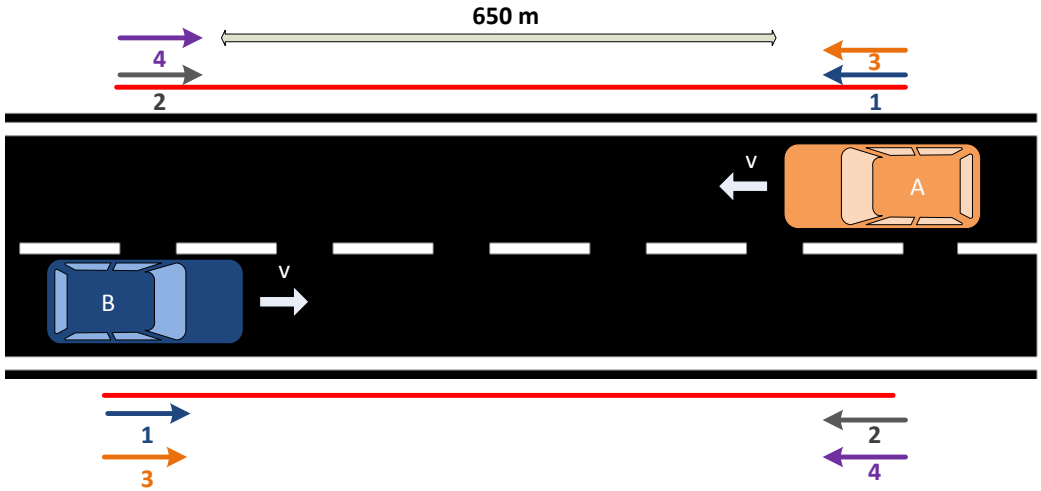

Figure 5.10: Two Cars

In this scenario, two cars are driven in the same direction but in opposite ways. This way, the sending of bundles is tested with two nodes moving, with two different velocities, 50 and 70 km/h. The sender is the car A and the destination is the car B. When the contact time between the two nodes was not enough to send the entire bundle, the cars turned around

when the connection was already lost and started moving towards each other again, over and over until the bundle has been completely sent.

For 50 km/h the files sent contain 2, 5 and 10 MB, while for 70 km/h the files sent contain 1, 5 and 10 MB. The two vehicles, when in contact, maintained the specified velocity until the connection was lost. The graphs present the same metrics and are presented in figures 5.11 (for 50 km/h) and 5.12 (for 70 km/h).

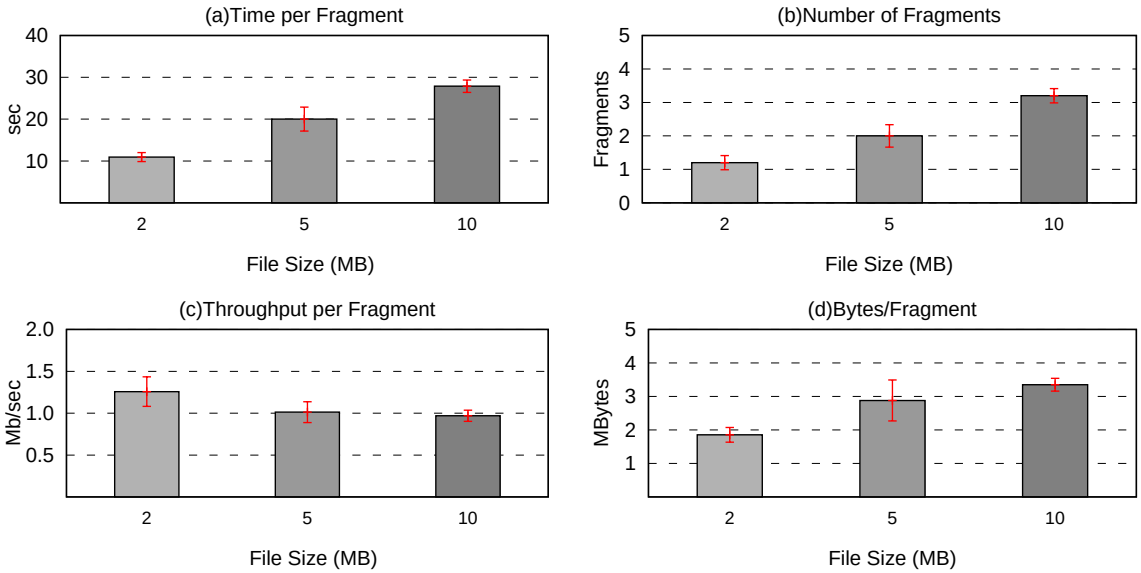

Figure 5.11: Two Cars Results for 50 km/h

Looking to figure 5.11, it is easy to observe that the files are sent with fragmentation, as opposed to what happened in the previous scenario (5 and 10 MB). As it can be seen, both of them need 2 and 3 fragments approximately, respectively (graph (b)). Due to environment's dynamics referred in the beginning of section 5.4, the results in this scenario are not very constant. The increase of the number of fragments to send a file once again makes breaks on the average throughput per fragment (graph (c)). The higher throughput noted was in the case where the file was sent, in average, in only one fragment. The average number of bytes sent per fragment increases with the file length, but this depends on the fragments sent (graph (d)).

Table 5.5 shows the average data sent on each fragment for the files with the size of 5 and 10 MB, which were not sent in one unique fragment.

	5MB		10MB	
Fragment	Trasnferred (MB)	Time (sec)	Transferred (MB)	Time (sec)
first	3,1	23	4,0	33
second	2,2	16	3,8	32
third	-	-	2,2	20

Table 5.5: Bytes Transferred and Time of Transfer - 50 km/h

The first signs of external factors affecting the results can be noted by the results for the

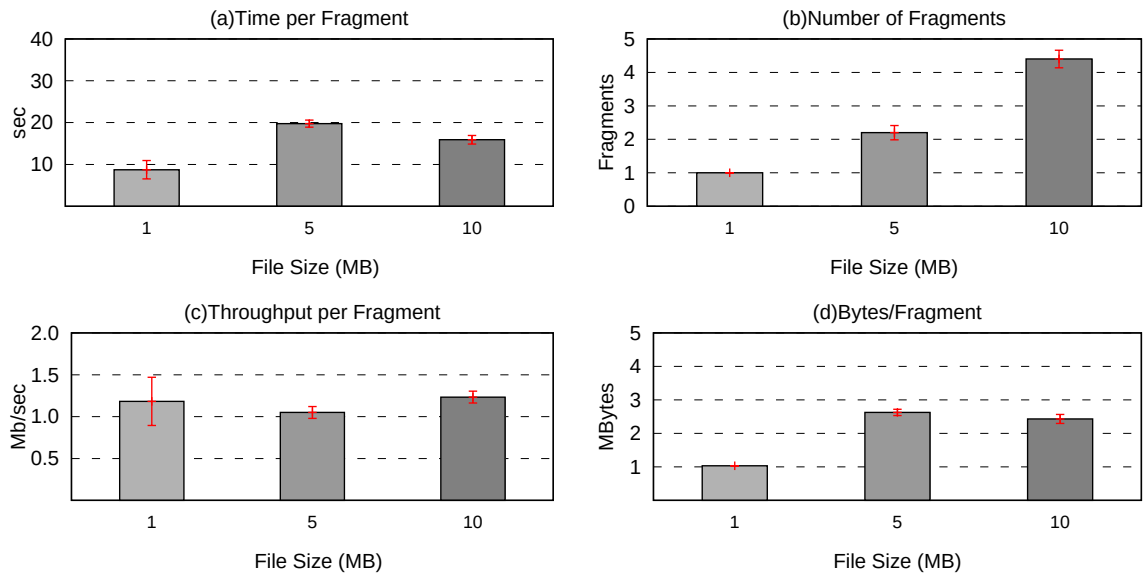

Figure 5.12: Two Cars Results for 70 km/h

file of 5 MB. The first fragment was sent in average during 22,7 seconds, while for the file of 10 MB the first fragment was sent during 33 seconds as well as the second fragment for the same file. This shows that the higher time of contact for the two nodes with the specified velocity is around 33 seconds, and for the file of 5 MB this was not achieved due to the referred external factors.

Figure 5.12 shows a different tendency in graph (a) comparing to the results for 50 km/h. In this case, the external factors had more influence for the file of 10 MB, since the average time per fragment was lower comparing to 5 MB, also shown for each fragment on table 5.6. In this case, taking a look at graph (b), to send the file of 10 MB, 4 fragments in average were necessary (due to the environment's conditions). For the 5 MB file two fragments were necessary as in the previous case. For the throughput per fragment (graph (c)), the values tend to the same value, approximately. They are a little bit above the values for 50 km/h for the same reason explained for the previous scenario, related with the velocities. The bytes sent per fragment (graph (d)) show a decrease comparing to the case of 50 km/h, which is normal since the contact time is shorter. However, the difference is not very significant.

The next table 5.6 gives a better view about what happened in average on each fragment sent for the files 5 and 10 MB for 70 km/h.

	5MB		10MB	
Fragment	Trasnferred (MB)	Time (sec)	Transferred (MB)	Time (sec)
first	3,9	29	3,3	23
second	1,8	14	1,3	9
third	-	-	3,1	19
fourth	-	-	2,7	17

Table 5.6: Bytes Transferred and Time of Transfer - 70 km/h

In this case, for the 5 MB file, the largest time contact was achieved on the first fragment (29 seconds). The second fragment delivered the remaining data that was left. The 10 MB file was more influenced by the external factors, as it can be seen from the time the contacts of the four fragments needed to send the entire bundle. If the contacts were not so influenced by the external conditions, only three fragments would be needed. The process of transferring 10 MB file have low time contacts; the highest is the first with 23 seconds. The second is the most affected with only 9 seconds. Even with these constraints the files are all sent with more or less fragments needed.

5.4.3 One Car-Two RSUs

The last scenario deployed on the street is composed by three nodes, one car and two RSUs, and it is illustrated in figure 5.13.

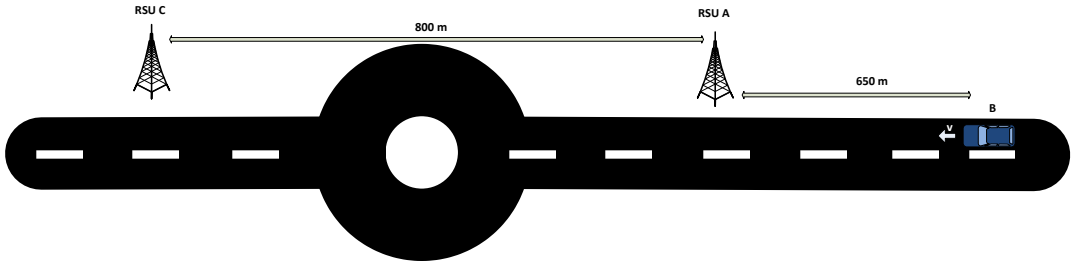

Figure 5.13: One Car Two RSUs

This scenario is used to test the store-and-forward mechanism with a relay node (car). Three different files are sent, once at a time, from RSU A with the destination RSU C. Since the two RSUs are apart to not communicate, the bundle will be sent to the car B (the car will drive at a velocity of 50 km/h), and then from car B to RSU C, reaching this way its final destination. Since the fragmentation process was already tested, the files sent in this scenario have a size that allows the sending in only one fragment, and it is not necessary to go again to the same spot to finish the transmission. The figure shows the trajectory of the car B passing through RSU A and then through RSU C.

The files sent from RSU A have the following sizes: 5, 6 and 7 MB. The results from this scenario are represented in figure 5.14. The results are an average of the two links, the link from RSU A to car B and the link from car B to RSU C.

Since the files do not differ very much in their size, unlike the other scenarios, it can be seen that the average transfer time per fragment (graph (a)) increases softly with the increase of the size of the file. The transfer time in the three cases gets around the 30 seconds. In average, the bundles are all sent in only one fragment (graph (b)), but for the case of 7 MB file, due to the previous described external factors, the connection in the second link (in some tests) was broke and the bundle was sent in more than one fragment. The mean throughput per fragment, on graph (c), does not show large variations. The largest break is in the 7 MB file due to the breaks in the connection, increasing the mean number of fragments, reducing the throughput. Since the 5 and 6 MB files were sent in only one fragment on both links, the average bytes per fragment (grpah (d)) correspond to the original size of the file. In the case

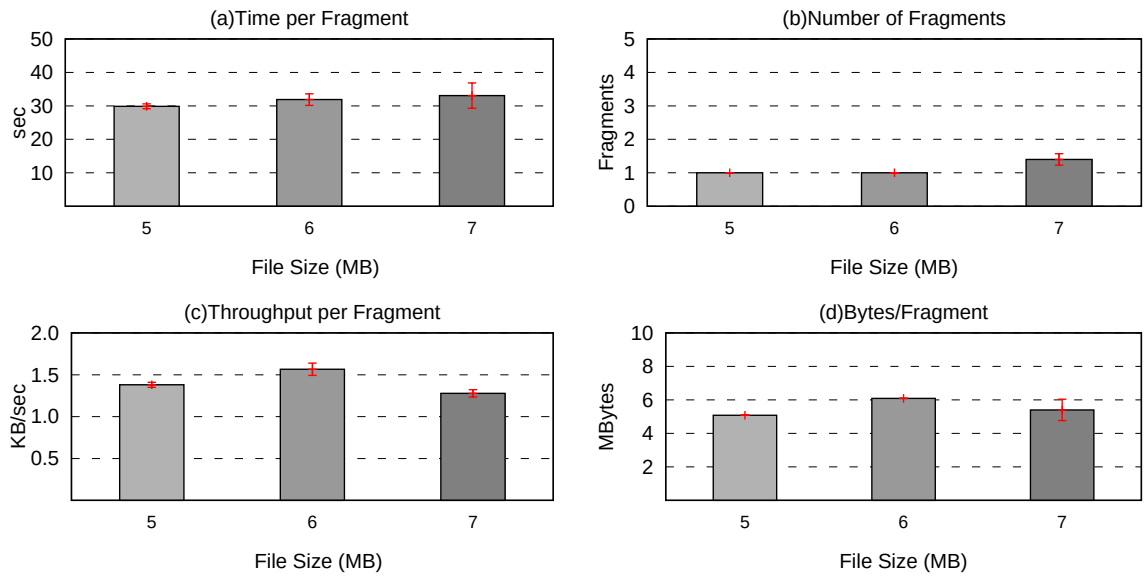

Figure 5.14: One Car and Two RSUs Results for 50 km/h

of 7 MB file, since in some tests for the link 2 the number of fragments increased, the average number of bytes sent per fragment is reduced comparing to the other files.

5.5 Summary

In this chapter, IBR-DTN was tested and evaluated. Two types of scenarios were deployed: scenarios in laboratory composed by two and three nodes, and scenarios on a real testbed using cars and RSUs.

The results generated in the laboratory enabled a better understanding of IBR-DTN, how it worked and its behaviour. The section 5.3 illustrated and explained the results of the two scenarios tested. The results show that, even with large files (50 MB) and several connection time intervals, the bundle eventually is completely received by its final destination (in both scenarios). Since everything was working well on laboratory, it was the time to test IBR-DTN in a real vehicular environment.

The section 5.4 presents the results generated in the real testbed, using the boards on cars and in RSUs, just like in a real vehicular environment. Three scenarios were tested and three velocities were tested, 30, 50 and 70 km/h. The increase of velocity in the same scenario, naturally, increased the number of fragments needed to send the same file. The increase of velocity also provoked a decrease on contact times between nodes, specially in the first fragment when the bundle cannot be sent in a single one.

The performed tests prove the good operation of IBR-DTN in a vehicular environment with mobility dynamics, and also with external factors that can break or diminish the connection's quality.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The aim of this Dissertation work was to study the existing DTNs architecture, implement it on a vehicular environment and study its behaviour.

The study of the VANETs and DTNs allowed to conclude that all the projects, involving the concept of DTNs in VANETs, did not made tests using the vehicular standard IEEE 802.11p, and did not deploy any scenarios using cars in movement and fixed infrastructures. Therefore, the developed work allowed to make the experiments using real vehicular scenarios.

Two DTN implementations were tested in the boards, since DTN2 had to be discarded due to its problems, which made impossible to use it in vehicular environments. IBR-DTN was the chosen DTN implementation to be used in the experiments using the boards. Several problems were found and solved in IBR-DTN, which allowed to conclude that the DTN concept continues in expansion and a lot of work needs to be done.

Several scenarios were deployed: first in the laboratory and then in real vehicular scenarios, always using the standard 802.11p for communication. The tests performed in the laboratory allowed to understand the operation of IBR-DTN, where several files were sent from board to board, with different sizes and different connection time intervals. These tests allowed to conclude the good operation of the implementation regarding its fragmentation process and the reassembling of the several fragments at the destination node.

Regarding the tests performed on the real testbed, three scenarios were deployed using cars and fixed infrastructures. Several velocities were tested in the deployed scenarios allowing to conclude the good operation of IBR-DTN in the real vehicular environments for different velocities and scenarios. The velocity issue did not cause any problems in IBR-DTN performance. Increasing the velocity provoked the increasing of the number of fragments required to send a file, since the contact time between the nodes is shorter. The good operation of the store-and-forward mechanism was also proved when using 1 car and 2 RSUs, since all files sent eventually reached their final destination.

The deployed scenarios allowed to conclude that it is possible to have a DTN operating in a real vehicular communication via IEEE 802.11p. It is possible, though, to increase VANET efficiency services with this mechanism to send information opportunistically.

This work was just the beginning, since it was the first time that a DTN implementation was tested in a real vehicular environment using the vehicular standard for communication 802.11p, using cars and RSUs. Since the real implementations of DTNs are a subject that

continues to be exploited, a lot of work had to be done before testing it on the road.

6.2 Future Work

Since the concept of DTNs remains to be fully exploited, there is a lot of work that can be done to continue this Dissertation's work. The real implementations of DTNs are a subject that continues to be exploited and errors were found while testing IBR-DTN on laboratory, and so a lot of debugging work needed to be done. The tests of the real testbed proved the good functionality of IBR-DTN in a vehicular environment, but these tests were only formed by the maximum of three nodes. Tests need to be done using a larger network with a considerable large number of nodes and evaluate its behaviour.

Several routing protocols incorporated in IBR-DTN, like PROPHET, flooding and epidemic can be tested in the mentioned large networks and be compared for different types of scenarios, evaluating the best one for each type of scenario. The metrics to evaluate could be the percentage of bundles delivered to its final destination, the overhead provoked by each protocol on the network, percentage of bundles discarded.

There is an alteration that can be done to improve IBR-DTN's capabilities related to the fragmentation process. As it was explained in chapter 4, the fragmentation process is made node by node, which limits the efficiency of IBR-DTN. This process could be modified to better improve its capabilities to deliver bundles, by enabling that different parts of a bundle could be received by a destination from different nodes. For example, a node A has a bundle to send to node D. First, 30 per cent of the bundle are sent to node B and then, 70 per cent are sent to node C. With the adequate changes in the code, node D should be able to receive the fragments from B and C and reassemble them together. This way, the efficiency would be highly improved.

Bibliography

[1] Paulo Rogerio Pereira, Augusto Casaca, Joel J. P. C. Rodrigues, Vasco N. G. J. Soares, Joan Triay, and Cristina Cervello-Pastor. From delay-tolerant networks to vehicular delay-tolerant networks. *Communications Surveys Tutorials, IEEE*, 14(4):1166–1182, 2012.

[2] Rafael dos S. Alves. *Livro Texto dos Minicursos do XXVII Simpsio Brasileiro de Redes de Computadores e Sistemas Distribudos*, chapter Redes Veiculares: Princpios, Aplicaes e Desafios. 2009.

[3] Hassnaa Moustafa and Yan Zhang. *Vehicular Networks Techniques, Standards and Applications*, chapter Introduction to Vehicular Networks. Auerbach Publications, 2009.

[4] O.K. Tonguz and G. Ferrari. *Ad Hoc Wireless Networks: A Communication-Theoretic Perspective*. New York: John Wiley & Sons, 2006.

[5] O.K. Tonguz, N. Wisitpongphan, J.S. Parikh, Fan Bai, P. Mudalige, and V.K. Sadekar. On the broadcast storm problem in ad hoc wireless networks. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–11, 2006.

[6] Marc Torrent-Moreno, Daniel Jiang, and Hannes Hartenstein. Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, VANET '04, pages 10–18. ACM, 2004.

[7] Internet Engineering Task Force (IETF). Ad-hoc network autoconfiguration (autoconf). `http://datatracker.ietf.org/wg/autoconf/charter/`, 2012.

[8] Hassnaa Moustafa and Yan Zhang. *Vehicular Networks Techniques, Standards and Applications*, chapter Vehicular Network Applications and Services. Auerbach Publications, 2009.

[9] Marc Torrent-Moreno, Daniel Jiang, and Hannes Hartenstein. Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, VANET '04, pages 10–18. ACM, 2004.

[10] 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. *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), pages 1–51, 2010.

[11] Internet Engineering Task Force (IETF). Ip mobility support. `http://www.ietf.org/rfc/rfc2002.txt`, 2002.

[12] Hassnaa Moustafa and Yan Zhang. *Vehicular Networks Techniques, Standards and Applications*, chapter Routing in Vehicular Networks: A User's Perspective. Auerbach Publications, 2009.

[13] Bijan Paul, Md. Ibrahim, and Md. Abu Naser Bikas. Article: Vanet routing protocols: Pros and cons. *International Journal of Computer Applications*, 20(3):28–34, 2011. Published by Foundation of Computer Science.

[14] C. Maihofer. A survey of geocast routing protocols. *Communications Surveys Tutorials, IEEE*, 6(2):32–42, 2004.

[15] Jinyang Tay Y.C. Jiang, Mingliang Li. Cluster based routing protocol(cbrp). `http://tools.ietf.org/html/draft-ietf-manet-cbrp-spec-01`, August 1999.

[16] A. Iwata, Ching-Chuan Chiang, Guangyu Pei, M. Gerla, and Tsu-Wei Chen. Scalable routing strategies for ad hoc wireless networks. *Selected Areas in Communications, IEEE Journal on*, 17(8):1369–1379, 1999.

[17] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks, 2000. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.6151.

[18] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *Communications Magazine, IEEE*, 41(6):128–136, 2003.

[19] Interplanetary networking special interest group (ipnsig). `http://ipnsig.org/`, October 2013.

[20] Internet Research Task Force. Delay-Tolerant Networking Architecture. Internet-Draft RFC 4838, April 2007.

[21] Kevin 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, pages 27–34. ACM, 2003.

[22] William Kehr. Delay tolerant networks - university of missouri-rolla, May 2006. http://web.mst.edu/-mobildat/DTN/index.html.

[23] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[24] Kevin R. Fall and Stephen Farrell. Dtn: an architectural retrospective. *IEEE Journal on Selected Areas in Communications*, 26(5):828–836, 2008.

[25] DTN Research Group. Bundle Security Protocol Specification, 2011. http://tools.ietf.org/html/draft-irtf-dtnrg-bundle-security-19.

[26] H. Jun, M.H. Ammar, and E.W. Zegura. Power management in delay tolerant networks: a framework and knowledge-based mechanisms. In *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, 2005.

[27] E. P. Jones and P. A. Ward. Routing strategies for delay-tolerant networks. *Submitted to ACM Computer Communication Review (CCR)*, 2006.

[28] J. LeBrun, Chen-Nee Chuah, D. Ghosal, and M. Zhang. Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 4, pages 2289–2293 Vol. 4, 2005.

[29] Thrasyvoulos Spyropoulos, K. Psounis, and C.S. Raghavendra. Single-copy routing in intermittently connected mobile networks. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 235–244, 2004.

[30] J. Burgess, Brian 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*, pages 1–11, 2006.

[31] E.P.C. Jones, L. Li, J.K. Schmidtke, and P.A.S. Ward. Practical routing in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 6(8):943–959, 2007.

[32] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*. ACM, 1987.

[33] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, July 2003.

[34] Michael Demmer and Kevin Fall. Dtlsr: delay tolerant routing for developing regions. In *Proceedings of the 2007 workshop on Networked systems for developing regions*, pages 5:1–5:6. ACM, 2007.

[35] Stephen Farrell and V. Cahill. *Delay- and Disruption-Tolerant Networking*. Artech House, 2006.

[36] Kevin Jain, Sushant Fall and Rabin Patra. Routing in a delay tolerant network. *SIGCOMM Comput. Commun. Rev.*, 34(4):145–158, 2004.

[37] M. Zhang and R.S. Wolff. Routing protocols for vehicular ad hoc networks in rural areas. *Communications Magazine, IEEE*, 46(11):126–131, 2008.

[38] Mingliu Zhang and Richard Wolff. A border node based routing protocol for partially connected vehicular ad hoc networks. *Journal of Communications*, 5(2), 2010.

[39] N. Wisitpongphan, Fan Bai, P. Mudalige, V. Sadekar, and O. Tonguz. Routing in sparse vehicular ad hoc wireless networks. *Selected Areas in Communications, IEEE Journal on*, 25(8):1538–1556, 2007.

[40] P. Willke, T.L. Tientrakool and N.F. Maxemchuk. A survey of inter-vehicle communication protocols and their applications. *Communications Surveys Tutorials, IEEE*, 11(2):3–20, 2009.

[41] M. H. Oliver E. A. Ur Rahman S. Guo, S. Falaki, M. A. Seth, A. Zaharia, and S. Keshav. Very low-cost internet access using kiosknet. *SIGCOMM Comput. Commun. Rev.*, 37(5):95–100, 2007.

[42] Nilanjan Balasubramanian Aruna Soroush, Hamed Banerjee, Brian Neil Corner, Mark D. Levine, and Brian Lynn. Dome: a diverse outdoor mobile testbed. In *Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements*, HotPlanet '09, pages 2:1–2:6. ACM, 2009.

[43] F. Soares, V.N.G.J. Farahmand and J. J P C Rodrigues. A layered architecture for vehicular delay-tolerant networks. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 122–127, 2009.

[44] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 125–138. ACM, 2006.

[45] Michael Lahde, Sven Doering, Wolf-Bastian Pöttner, Gerrit Lammert, and Lars Wolf. A practical analysis of communication characteristics for mobile and distributed pollution measurements on the road: Research articles. *Wirel. Commun. Mob. Comput.*, 7(10):1209–1218, 2007.

[46] Jrg Ott and Dirk Kutscher. *Mobile Ad-hoc Networks: From Theory to Reality*, chapter From Drive-thru Internet to Delay-tolerant Ad-hoc Networking. Nova Science Publishers, 2007.

[47] K. Scott. Disruption tolerant networking proxies for on-the-move tactical networks. In *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pages 3226–3231 Vol. 5, 2005.

[48] Paulo Rogerio Pereira, Augusto Casaca, Joel J. P. C. Rodrigues, Vasco N. G. J. Soares, Joan Triay, and Cristina Cervello-Pastor. From delay-tolerant networks to vehicular delay-tolerant networks. *Communications Surveys Tutorials, IEEE*, 14(4):1166–1182, 2012.

[49] L. Franck and F. Gil-Castineira. Using delay tolerant networks for car2car communications. In *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pages 2573–2578, 2007.

[50] F.J. Cabrera, V. Ros and P.M. Ruiz. Simulation-based study of common issues in vanet routing protocols. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1–5, 2009.

[51] Delay Tolerant Networking Research Group. Delay tolerant networking research group, May 2013. http://www.dtnrg.org/wiki/Home.

[52] Networking Working Group. Licklider transmission protocol - specification, September 2008. http://tools.ietf.org/html/rfc5326.

[53] Delay Tolerant Networking Research Group. Delay tolerant networking research group, July 2013. http://www.dtnrg.org/wiki/Code.

[54] DTN Research Group. DTN2 Documentation, May 2013. http://www.dtnrg.org/wiki/Dtn2Documentation.

[55] Interplanetary Overlay Network, May 2013. https://ion.ocp.ohiou.edu/.

[56] Postellation: a Lean and Deployable DTN Implementation, May 2013. http://postellation.viagenie.ca/.

[57] Prof. Dr.-Ing. Lars Wolf Prof. Dr. Sndor P. Fekete and Prof. Dr. Rdiger Kapitza. Institut fur betriebssysteme und rechnerverbund, January 2012. http://www.ibr.cs.tu-bs.de/.

[58] Ibr-dtn a modular and lightweight implementation of the bundle protocol, August 2013. http://trac.ibr.cs.tu-bs.de/project-cm-2012-ibrdtn.

[59] JDTN, May 2013. http://jdtn.sourceforge.net/.

[60] Dtnsim2 a delay-tolerant network simulator, August 2013. http://watwire.uwaterloo.ca/DTN/sim/.

[61] Eric Fall Kevin Demmer, Michael Brewer, Robin Ho, Melissa Patra, and Sushant Jain. Implementing delay tolerant networking. Technical report, 2003. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.2832.

[62] Prophet router. `http://www.dtnrg.org/docs/code/DTN2/doc/manual/ro_prophet.html`, August 2013.

[63] Dtlsr protocol. `http://www.dtnrg.org/docs/code/DTN2/doc/manual/ro_dtlsr.html`, August 2013.

[64] Flood router. `http://www.dtnrg.org/docs/code/DTN2/doc/manual/ro_flood.html`, August 2013.

[65] tca-router routing algorithm. `http://www.dtnrg.org/docs/code/DTN2/doc/manual/ro_tca-router.h` August 2013.

[66] tca-client routing algorithm. `http://www.dtnrg.org/docs/code/DTN2/doc/manual/ro_tca-client.h` August 2013.

[67] External router. `http://www.dtnrg.org/docs/code/DTN2/doc/manual/ro_external.html`, August 2013.

[68] Static routing protocol. `http://www.dtnrg.org/docs/code/DTN2/doc/manual/ro_static.html`, August 2013.

[69] Openwrt wireless freedom. `https://openwrt.org/`, August 2013.

[70] u c l i b c. `http://www.uclibc.org/`, August 2013.

[71] Sebastian Schildt, Johannes Morgenroth, Wolf-Bastian Pttner, and Lars C. Wolf. Ibr-dtn: A lightweight, modular and highly portable bundle protocol implementation. *ECEASST*, 37, 2011.

[72] D. Ellard, D. Brown. Dtn ip neighbour discovery (ipnd). `http://tools.ietf.org/html/draft-irtf-dtnrg-ipnd-01`, September 2010.

[73] J. Demmer, M. Ott. Delay tolerant networking tcp convergence layer protocol. `http://tools.ietf.org/html/draft-irtf-dtnrg-tcp-clayer-02`, May 2009.

[74] S. Kruse, H. Ostermann. Udp convergence layers for the dtn bundle and ltp protocols. `http://tools.ietf.org/html/draft-irtf-dtnrg-udp-clayer-00`, May 2009.

[75] curl. `http://curl.haxx.se/`, August 2013.

[76] IEEE Standard for Information technology. Ieee. 802.15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans), 2007.

[77] Sqlite. `http://www.sqlite.org/`, August 2013.

[78] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, 2002.

[79] A. Davies E. Grasic S. Lindgren, A. Doria. `http://tools.ietf.org/html/draft-irtf-dtnrg-prophet` October 2011.

[80] Shinsuke Beuran, Razvan Miwa and Yoichi Shinoda. Performance evaluation of dtn implementations on a large-scale network emulation testbed. In *Proceedings of the seventh ACM international workshop on Challenged networks*. ACM, 2012.

[81] PC Engines. Alix.2 / alix.3 series (system boards), 2007. Datasheet.

[82] Unex. Dcma-86p2: Wifi mini-pci module for 802.11p/dsrc application, ar5414a-b2b, 2010. Datasheet.

[83] Filipe Neves. Comunicação entre Veículos - Testes de Comunicações Reais. Master's thesis, Universidade de Aveiro, 2011.

[84] Wifi-antennas. 5dbi 2.4ghz omni antenna - rp sma. `http://www.wifi-antennas.co.uk/`, 2011.

[85] Buildroot. Buildroot. `http://buildroot.uclibc.org/`, 2013.

[86] C. Ameixieira, J. Matos, R. Moreira, A. Cardote, A. Oliveira, and S. Sargento. An ieee 802.11p / wave implementation with synchronous channel switching for seamless dual-channel access. In *IEEE Vehicular Networking Conference (VNC)*, November 2011.

[87] Carlos Ameixieira. Desenvolvimento da Sub-Camada MAC IEEE 802.11p/1609.4. Master's thesis, Universidade de Aveiro, 2011.

[88] Sven Morgenroth Johannes Doering, Michael Lahde and Lars Wolf. Ibr-dtn: an efficient implementation for embedded systems. In *Proceedings of the third ACM workshop on Challenged networks*. ACM, 2008.

[89] Tar. http://www.gnu.org/software/tar/, 2013.

[90] Ibr-dtn –. https://mail.ibr.cs.tu-bs.de/mailman/listinfo/ibr-dtn, 2013.

[91] Filipe Neves, Andre Cardote, Ricardo Moreira, and Susana Sargento. Real-world evaluation of ieee 802.11p for vehicular networks. In *Proceedings of the Eighth ACM international workshop on Vehicular inter-networking*, VANET '11, pages 89–90, New York, NY, USA, 2011. ACM.