University of Minho
School of Engineering
Department of Informatics

Bruno Daniel Mestre Viana Ribeiro

Platooning Simulation in ITS Communications

Master dissertation
Engineering of Computer Networks and Telematic Services

Dissertation supervised by
Prof. Alexandre Santos
Prof. Maria João Nicolau

October 2016
Acknowledgments

This research work would not have been possible without the help of many people, for their support and valuable contributions.

I would like to express my appreciation and sincere thanks to my supervisors from University of Minho, professors Alexandre Santos and Maria João Nicolau. I am very grateful for their great effort in giving me useful advices and guidance throughout the development of my thesis.

I would also like to thank all my colleagues from Algoritmi Center at University of Minho for their support, in particular to Fábio Gonçalves for his truly valuable input and interesting discussions on Platooning.

I must express my gratitude to my parents and family for their inexhaustible support and endless encouragement throughout my life.

And last, but not least, a very special word of appraise goes to my girlfriend Raquel for her patience and for being supportive during the most stressful times.

Part of this research has been sponsored by the Portugal Incentive System for Research and Technological Development. Project in co-promotion nº 002797/2015 (INNOVCAR 2015-2018).
Abstract

Vehicular Ad Hoc Networks (VANETs) is a term used to describe networks of moving vehicles equipped with devices that allow spontaneous communication with other vehicles and infrastructures. Developing collaborative driving applications for VANETs is currently a hot topic and has an increasing popularity in the Intelligent Transportation Systems (ITS) domain. The goal of this thesis is to study the development and testing of advanced ITS applications, using Platooning as a use case. It presents a state of the art on typical ITS applications, its evaluation and corresponding implementation and testing methods. The Platooning Management Protocol (PMP) was then implemented and tested by means of simulation, resorting to the V2X Simulation Runtime Infrastructure (VSimRTI) framework, which couples Simulation of Urban MObility (SUMO) and Network Simulator 3 (ns-3). Results show that it is able to work in a smooth and efficient manner: the maneuvers happen during an acceptable interval, the proposed communication requirements are met and the lane capacity is increased.
Resumo

Vehicular Ad Hoc Networks (VANETs) é um termo usado para descrever redes de veículos em movimento, equipados com dispositivos que permitem uma comunicação espontânea com outros veículos e infraestruturas. Desenvolver aplicações de condução colaborativa para VANETs é atualmente um tópico muito estudado e cuja popularidade tem crescido no domínio dos Intelligent Transportation Systems (ITS) - sistemas de transporte inteligentes. O objetivo desta tese é o estudo, desenvolvimento e teste de aplicações de ITS avançadas, utilizando Platooning como caso de uso. Neste documento é apresentado o estado da arte relativamente às aplicações ITS tipicamente avaliadas e os respetivos métodos de implementação e teste. O Platooning Management Protocol (PMP) foi implementado e testado através de simulação, utilizando a ferramenta V2X Simulation Runtime Infrastructure (VSimRTI), que acopla as ferramentas Simulation of Urban MObility (SUMO) e Network Simulator 3 (ns-3). Os resultados mostram que o protocolo funciona de forma leve e eficiente: as manobras decorrem num intervalo de tempo aceitável, os requisitos de comunicações são cumpridos e a capacidade das faixas é aumentada.
## Contents

1 Introduction ........................................ 1
   1.1 Background .................................. 2
      1.1.1 Intelligent Transportation Systems ...... 2
      1.1.2 ITS Applications .......................... 5
      1.1.3 Advanced ITS Applications ............... 7
   1.2 Thesis Objectives .............................. 9
   1.3 Thesis Outline ................................ 10

2 Related Work ....................................... 11
   2.1 Related Literature ............................. 11
   2.2 Related Projects .............................. 16
   2.3 Summary ..................................... 18

3 ITS Simulation ..................................... 19
   3.1 Simulation Tools .............................. 20
      3.1.1 Network Simulation ....................... 20
      3.1.2 Traffic Simulation ....................... 21
      3.1.3 Coupled Simulation ...................... 23
   3.2 Summary ..................................... 24

4 ITS Applications Taxonomy ......................... 25
   4.1 Application Taxonomy ......................... 25
   4.2 Platooning Use Case ........................... 34
   4.3 Platooning Management Protocol ............... 36
      4.3.1 Maneuvers ............................... 36
List of Figures

1.1 Vehicle Communication Modes [1] .......................... 3
1.2 A classification of the existing ITS applications [2] ........... 6

4.1 Platoon of vehicles ........................................ 34
4.2 Create maneuver ............................................. 37
4.3 Join maneuver - Steps 1 and 2 ............................. 39
4.4 Join maneuver - Step 3 to 6 ................................ 39
4.5 Join maneuver - Step 7 and 8 ............................. 39
4.6 Join maneuver - Final state ............................... 39
4.7 Dissolve maneuver - Step 1 ............................... 40
4.8 Dissolve maneuver - Step 2 ............................... 40
4.9 Dissolve maneuver - Step 3 ............................... 41
4.10 Dissolve maneuver - Final State ......................... 41
4.11 Leave maneuver - Step 1 .................................. 42
4.12 Leave maneuver - Steps 2 to 4 ......................... 42
4.13 Leave maneuver - Step 5 and 6 ......................... 42
4.14 Leave maneuver - Step 7 ............................... 42
4.15 Leave maneuver - Final State ........................... 43
4.16 Merge maneuver - Steps 1 and 2 ....................... 44
4.17 Merge maneuver - Step 3 ............................... 44
4.18 Merge maneuver - Steps 4 to 6 ......................... 44
4.19 Merge maneuver - Steps 7 and 8 ....................... 44
4.20 Merge maneuver - Final State ........................... 45

5.1 Scenario Maps ............................................... 53
5.2 Working Environment ...................................... 54
5.3 Leader Vehicle_0 creates platoon A ........................................ 59
5.4 Vehicle_1 sends a Join Request towards Vehicle_0 ............... 61
5.5 Vehicle_0 acknowledges the Join Request from Vehicle_2 and
send a Adjust Gap message to Vehicle_1 ................................. 63
5.6 Vehicle_1 informs the Leader that has reached the correct ad-
justing position ................................................................. 64
5.7 Vehicle_2 reaches the correct joining position and sends a Dis-
tance Achieved message towards Vehicle_0 .......................... 64
5.8 Leader Vehicle_0 sends a Start Maneuver message to Vehi-
cle_2. When the Vehicle_2 informs that the maneuver is com-
pleted, Vehicle_0 sends Platoon Updated messages to the string,
and the platoon reaches a stable state - String is now composed of
[Vehicle_0, Vehicle_2, Vehicle_1] ........................................... 65
5.9 Vehicle_3 awaiting instructions from Vehicle_0 ................. 67
5.10 Leader Vehicle_4 acknowledges the Join Request from Vehi-
cle_6 and then proceeds to reject the request from Vehicle_7
(maximum string size is achieved) ........................................ 68
5.11 Platoon B stabilized after the Join maneuvers from Vehicle_5
and Vehicle_6 - String is now composed of [Vehicle_4, Vehi-
cle_6, Vehicle_5] .......................................................... 68
5.12 Leave Maneuvers I ......................................................... 69
5.13 Leave Maneuvers II ......................................................... 70
5.14 Vehicle_8 joins Platoon B at the rear ................................. 71
5.15 Stabilized Platoons ....................................................... 71
5.16 Vehicle_4 (Leader B) sends a Merge Request geocast message . 73
5.17 Vehicle_4 (and Platoon B) approaching Platoon A ............. 74
5.18 Merge maneuver final state - Platoon A is now composed of
[Vehicle_0, Vehicle_1, Vehicle_3, Vehicle_4, Vehicle_6, Vehicle_8] 75
5.19 Vehicle_8 Leave maneuver .............................................. 76
5.20 Stabilized Platoon A - [Vehicle_0, Vehicle_1, Vehicle_3, Vehi-
cle_4, Vehicle_6] .......................................................... 77
5.21 Dissolved ................................................................. 79
6.1 Lane Capacity during simulation time . . . . . . . . . . . . . . 89
6.2 Mean message delivery delay values for each simulation . . . . 96
6.3 Vehicle’s distance to go value during simulation . . . . . . . . . 97
6.4 Vehicle’s speed value during simulation . . . . . . . . . . . . . . 99
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>ITS applications and their communication requirements [3]</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>Basic Set of Applications (BSA) Definition [4]</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>Requirements of Intelligent Transportation Systems (ITS) use</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>cases [4]</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>General requirements for applications [5]</td>
<td>31</td>
</tr>
<tr>
<td>4.5</td>
<td>European Telecommunications Standards Institute (ETSI) applications</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>according to CAR2CAR classification</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>Applications Requirements Summary</td>
<td>33</td>
</tr>
<tr>
<td>4.7</td>
<td>Potential Wireless Communication Technologies for ITS Applications</td>
<td>34</td>
</tr>
<tr>
<td>4.8</td>
<td>Platoon Maneuvers Requirements</td>
<td>46</td>
</tr>
<tr>
<td>5.1</td>
<td>Vehicle colors in Simulation of Urban MObility (SUMO)</td>
<td>59</td>
</tr>
<tr>
<td>6.1</td>
<td>Mean capacity values during simulation runtime</td>
<td>89</td>
</tr>
<tr>
<td>6.2</td>
<td>Join maneuver duration</td>
<td>90</td>
</tr>
<tr>
<td>6.3</td>
<td>Rejected Join Request duration</td>
<td>91</td>
</tr>
<tr>
<td>6.4</td>
<td>Leave maneuver duration</td>
<td>92</td>
</tr>
<tr>
<td>6.5</td>
<td>Adjusting Gap durations</td>
<td>93</td>
</tr>
<tr>
<td>6.6</td>
<td>Merge maneuver duration</td>
<td>93</td>
</tr>
<tr>
<td>6.7</td>
<td>Dissolve maneuver duration</td>
<td>94</td>
</tr>
<tr>
<td>6.8</td>
<td>Statistic results from messages latency of the first simulation run</td>
<td>94</td>
</tr>
<tr>
<td>6.9</td>
<td>Messages with latency greater than 100 ms</td>
<td>95</td>
</tr>
</tbody>
</table>
Acronyms

ACC  Adaptive Cruise Control
AES  Advanced Encryption Standard
ARC  Adaptive Route Change
BSA  Basic Set of Applications
C2C  TMC to TMC
CACC  Cooperative Adaptive Cruise Control
CAM  Cooperative Awareness Message
CanuMobiSim  CANU Mobility Simulation Environment
CDG  Constant Distance Gap
CEN  European Committee for Standardization
COLOMBO  Cooperative Self-Organizing System for low Carbon Mobility at low Penetration Rates
COMPANION  Cooperative Dynamic Formation of Platoons for Safe and Energy-Optimized Goods Transportation
CTG  Constant Time Gap
DCC  Decentralized Congestion Control
DENM  Decentralized Environmental Notification Message
DSRC  Dedicated Short-Range Communications

DynB  Dynamic Beaconing

Energy ITS  Development of Energy-Saving ITS Technology

ETSI  European Telecommunications Standards Institute

EWA  Emergency Warning Application

GCDC  Grand Cooperative Driving Challenge

GLOSA  Green Light Optimized Speed Advisory

GPS  Global Positioning System

GTNetS  Georgia Tech Network Simulator

GUI  Graphical User Interface

HMI  Human-Machine Interface

I2C  Infrastructure to TMC

I2I  Infrastructure to Infrastructure

I2V  Infrastructure to Vehicle

iCS  iTETRIS Control System

IEEE  Institute of Electrical and Electronics Engineers

iGAME  Interoperable Grand Cooperative Driving Challenge AutoMation Experience

IP  Internet Protocol

ISO  International Organization for Standardization

iTETRIS  Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions
ITS  Intelligent Transportation Systems

JiST  Java in Simulation Time

JiST/SWANS  Java in Simulation Time/Scalable Wireless Ad-Hoc Network Simulator

MiXiM  Mixed Simulator

MOTO  Mobile Opportunistic Traffic Offloading

MOVE  MObility model generator for VEhicular networks

NHTSA  National Highway Traffic Safety Administration

ns-2  Network Simulator 2

ns-3  Network Simulator 3

OBU  On-Board Units

OEM  Original Equipment Manufacturer

OMNeT++  Objective Modular Network Testbed in C++

OSM  OpenStreetMap

PATH  Partners for Advanced Transportation Technology

PER  Packet Error Rate

PHEM  Passenger Cars and Heavy Duty Emissions Model

PMP  Platooning Management Protocol

RSA  Rivest-Shamir-Adleman

RSU  Roadside Units

SAE  Society of Automotive Engineers
SARTRE  Safe Road Trains for the Environment
SHA-256  Secure Hash Algorithm - 256
SUMO  Simulation of Urban MObility
SWANS  Scalable Wireless Ad-Hoc Network Simulator
TCP  Transmission Control Protocol
TMC  Traffic Management Center
TraCI  Traffic Control Interface
TSIS-CORSIM  Traffic Software Integrated System - Corridor Simulation
V2C  Vehicle to TMC
V2I  Vehicle to Infrastructure
V2V  Vehicle to Vehicle
V2X  Vehicle to Anything
VANETs  Vehicular Ad Hoc Networks
VCS  Vehicular Communication Systems
VEINS  Vehicles in Network Simulation
VENTOS  VEHicular NeTwork Open Simulator
VISSIM  Verkehr In Stadtten SIMulationsmodell
VSimRTI  V2X Simulation Runtime Infrastructure
WAVE  Wireless Access in Vehicular Environments
According to some researches [6] [7], it is expected that global Vehicle to Anything (V2X) communication modules in new vehicles will reach 62% and that Original Equipment Manufacturer (OEM) and aftermarket V2X modules will grow to 423 million by 2027. Cadillac recently announced [8] that we should expect Vehicle to Vehicle (V2V) technology in 2017 Cadillac CTS. Toyota has already introduced vehicle-infrastructure communication systems in 2015 car models [9]. These are just some examples of the current state of vehicle communications in the automotive industry.

Vehicles equipped with communication capabilities may be able to exchange important information that can help prevent accidents, save lives, ease traffic flow or even improve the environment. Vehicles that are connected require very efficient and smooth Vehicular Communication Systems (VCS), essentially due to the very short delays that this type of connection depends upon (specially when the communication concerns the safety of the drivers).

Some communication systems that may be adopted are traditional 3G/4G cellular or Wi-Fi, or specially designed systems like Dedicated Short-Range Communications (DSRC) or ITS G5. Cellular networks are typically applied in Vehicle to Infrastructure (V2I) solutions, whereas ad hoc networks are practically the only technology considered in V2V communications [10].

Intelligent Transportation Systems (ITS) are systems consisting of an intricate set of technologies applied to vehicles and infrastructures that ensure an
efficient and smart usage of the roads in general. They allow the control of traffic operations and even influence drivers behavior. These systems have the potential to improve safety, efficiency and productivity or even decrease levels of pollution.

ITS enable the rise of several applications relying on the exchange of information between vehicles themselves and infrastructures, allowing drivers to either manually or automatically make smarter driving choices. These application range from simple applications such as Emergency Vehicle Warning to more advanced and complex solutions like Platooning.

There are two main approaches when assessing vehicular networks applications (with different benefits and drawbacks): through simulation or field operational trials. Simulation allows the control of scenario parameters and the replication varying conditions, but its development is a challenging task. Field operational trials allow a realistic evaluation of systems but they are very expensive and possess serious safety and also technological constraints.

1.1 Background

This section provides an overview on the background of the project and its context. It contains a description of ITS and how the communication work on these systems, and a complete review on ITS applications and use cases, including advanced applications.

1.1.1 Intelligent Transportation Systems

ITS are defined by the European Parliament [11] as systems in which information and communication technologies are applied in the field of road transport, including infrastructure, vehicles and users, traffic management and mobility management, as well as for interfaces with other modes of transport. There are a wide range of ITS applications, such as road safety applications, traffic efficiency and infotainment.

The ITS standards [12] are defined so that one can establish and enable interoperability, by specifying who communicates with whom (e.g. vehicle,
road-side infrastructure, etc.) and what type of messages must be used, which protocol (e.g. Internet Protocol (IP)) should be used and in which communication media (e.g. wireless frequency band), etc.

Since the regularity constrains differ from region to region (even country to country), there are several ITS standards developing organizations: Institute of Electrical and Electronics Engineers (IEEE) and Society of Automotive Engineers (SAE) in the USA, European Committee for Standardization (CEN), International Organization for Standardization (ISO) and ETSI in Europe, etc.

Vehicle Communications

Luca Delgrossi states [1] that the idea of vehicle communication is that vehicles, roadside infrastructure (roads and highways), and back-end (telecommunications and Internet backbones) work together.

VCS are networks where vehicle’s On-Board Units (OBUs) and Roadside Units (RSUs) work as communicating nodes that share information with each other, like traffic information or safety warnings. Since VCS use a cooperative approach, accidents and traffic congestions avoidance can become more effective than by using the traditional way (where vehicles try to solve problems by themselves). Figure 1.1 illustrates how typical vehicle communication modes work. Below are described V2V modes of communication.

![Vehicle Communication Modes](image)

Figure 1.1: Vehicle Communication Modes [1]
**V2V Local Broadcast** In *V2V Local Broadcast*, a vehicle disseminates messages to all other vehicles that are within the respective communication range. This communication mode is used to support cooperative applications, mostly aimed to prevent collisions. This messages are used to inform nearby cars of current position, speed and such. However, due to the highly dynamic nature of traffic flow, the set of neighbors (vehicles) changes frequently. Thus, short range radios with native broadcasting capabilities are the best way to support this mode.

**V2V Multi-Hop Dissemination** In *V2V Multi-hop Dissemination*, messages from a given vehicle are relayed by others in order to reach vehicles that are out of the source’s communication range. This mode may be used to support hard safety applications, but only if the number of hops is very low.

**Infrastructure Communications**

The following subsection describes how V2I and Infrastructure to Vehicle (I2V) communications work.

**I2V Local Broadcast** Vehicles receive local broadcasts from the roadside infrastructure (e.g. Poor road conditions information or traffic controller information). *I2V Local Broadcast* can be implemented through the use of radio transceivers deployed along the roadside. It may also be implemented by using other strategies like cellular or satellite services, etc.

**V2I Bidirectional Communications** *V2I Bidirectional Communications* are required by a lot of mobility and convenience applications. Some examples are Internet access navigation and media download. *V2I Communications* can also be used to broadcast messages from a vehicle to other vehicles through the applications servers infrastructure. This mode can be supported by either using short range or long range radios. Additionally, it is possible for infrastructures to communicate with each other (Infrastructure to Infrastructure (I2I)) - this mode is mostly used for relaying
Traffic Management Center Communications

The Traffic Management Center (TMC) is used for monitoring and controlling traffic and the road network. There are three communication modes available for TMC: Vehicle to TMC (V2C), Infrastructure to TMC (I2C) and TMC to TMC (C2C). Through this communications, it is able to monitor the surrounding environmental conditions and manage the transportation resources in order to provide support for a better traffic flow (e.g. keep traffic moving safely after traffic incidents).

1.1.2 ITS Applications

ITS applications are developed to optimize driver safety and comfort. Vehicle communications bring a lot of benefits to users, such as improved safety for the driver (with solutions like Situation Awareness, Collision Avoidance or Post-crash Assistance), enhanced driving experience (e.g. Navigation, Routing, Points of Interest) or extended connectivity (office activity available in the car), etc. Given the wide range of existing (and upcoming) applications, one of the most critical tasks is to standardize the diverse communication requirements for each type of application.

Vehicle Communications Applications are typically divided [2, 13, 14] in two main categories: safety and non-safety. Figure 1.2 illustrates an example of how ITS applications can be described, according to [2].

The non-safety applications can be divided in two subcategories: Mobility (Efficiency) and Comfort (Connectivity and Convenience). Mobility applications focus mainly in improving and managing the traffic flow. Resorting to live traffic updates, route guidance, navigation and such, traffic behavior can in fact be enhanced. These applications provide updated local information, maps and information of relevance, in order to improve the traffic flow, coordination and assistance. This type of traffic applications is typically used in V2I scenarios. Connectivity and Convenience applications are more focused in assuring that the driving experience is more pleasant and
enjoyable to the driver. They provide functionalities needed by passengers and drivers for a convenient travel. They offer a wide range of features like email, media, access to social networks, etc. These applications are often delivered through the use of electronic devices (most of times through the driver’s smartphones) or in-vehicle devices. These applications may tolerate some delays but can also occasionally have a high data throughput demand. *Connectivity and Convenience* applications are also referred to as in-car comfort entertainment - they do not use inter-vehicular communications in most
situations. Safety applications can also be sub-divided [15, 16] in two kinds: Soft Safety and Hard Safety. Soft Safety applications mean to increase drivers safety but, since they do not require immediate attention, they are less time-critical. They notify the driver of not so imminent dangers like icy roads, traffic jams, reduced visibility, etc. Usually, the typical driver reaction to this notifications is to proceed more carefully or to take an alternate route. On other hand, Hard Safety applications are used to prevent accidents or, at least, minimize the severity if the crash is unavoidable. Thus, this kind of applications requires a minimum communication latency and high reliability when delivering messages, so that the driver can react in time. Forward Collision Warning and Emergency Electronic Brake Light are examples of Hard Safety applications. Safety applications rely on real-time information and typically resort to V2V communications. They are used to avoid traffic accidents. Vehicles and RSUs share information that is used to predict dangerous situations.

1.1.3 Advanced ITS Applications

ITS applications vary from basic management systems (such as car navigation and variable message signs) to more advanced applications that rely on real-time live information and feedback from surrounding sources. These kind of advanced solutions aim to provide innovative services, enabling the users to use the network in a more informed and coordinated way. Two very important advanced ITS applications that are heavily researched by the community are vehicle Platooning and Cooperative Adaptive Cruise Control (CACC). Platooning is defined as a set of vehicles connected via wireless communication that travel in conjunction and in formation, coordinated by a leader vehicle. The lead vehicle is responsible for general cooperative actions such as steering or braking and it is assumed that the leading driver must possess proper training to control a platoon. This solution allows the following drivers to perform tasks that otherwise were not advised to be performed (e.g. use a phone) while their vehicles are being controlled based on
sensor information. Still, these drivers must be able to control the vehicle if, for some reason, the platoon is dissolved or no longer controlled by the lead - deciding if it is safer to remain in the platoon or to drive manually becomes a challenge for every following driver. This way of traveling introduces many benefits such as increased road capacity, fuel efficiency and enhanced safety (fewer traffic collisions). There are several projects focused on the concept of Platooning, such as SARTRE [17], COMPANION [18] and i-GAME [19]. The CACC application is an evolution from the Adaptive Cruise Control (ACC) application that resorts to V2V communication to enable the exchange of useful information to and from neighbor vehicles. CACC is a system for vehicles that automatically adjusts the traveling speed to maintain a safe distance from vehicles ahead. Besides allowing a vehicle’s cruise control to maintain a proper distance to the following car, it also allows them to communicate between each other and cooperate in an adaptive mode. The neighbor’s information on acceleration is typically transmitted using CAM messages from the Wireless Access in Vehicular Environments (WAVE) technology. More than providing the driver with a more comfortable experience, CACC has the potential of improving traffic safety and efficiency. The ultimate goal of ITS applications is to reach the point of full automated driving and these applications are the next step towards it. Automated driving is the enabling key to eliminate the potential errors made by humans in transportation systems. Future driver-independent systems will result in fewer accidents, traffic jams and even lower emissions.

These two application have been often mistaken in recent years and assumed to be the same. Although both applications are part of a broader class of automatic vehicle control systems, there are important differences between them. The first difference is that CACC only controls longitudinal distance (the driver is responsible for steering and monitor the vehicle) while Platooning is capable of controlling both longitudinal and lateral distances. Also, according to SAE [20] and National Highway Traffic Safety Administration (NHTSA) [21], CACC will fit on a lower level of automation when comparing to Platooning.

Another difference between the two is the use of distinct vehicle following
control strategies. *Platooning* solutions typically resort to a Constant Distance Gap (CDG) strategy, coupling vehicles very close to each other with a constant distance, ignoring the vehicles velocity. With this approach, vehicles give the illusion of a chain link. This solution requires a very strict control mechanism and is not very tolerable to disrupt communications due to security issues - if communication fails during an emergency maneuver, following drivers may not be able to react in time in these very short distances. In other hand, CACC typically uses the Constant Time Gap (CTG) strategy, which is similar to human driver behavior. In this approach, the distance between vehicles depends on the vehicles speed - the distance is proportional to the speed.

### 1.2 Thesis Objectives

The development of advanced applications is currently a hot topic and is becoming more and more popular in the ITS field of work. The primary goal of this thesis is to develop and test advanced ITS applications, using *Platooning* as use case. ITS applications will be tested by means of simulation. The primary goal may be divided in sub-goals, listed as follows:

- Study the ITS applications development state of the art;
- Survey the existent VANETs simulation tools;
- Build a taxonomy for ITS applications, including a *Platooning* use case;
- Analyze and test the Platooning Management Protocol (PMP), including descriptions on the application behavior and maneuvers;
- Design and build the simulation environment;
- Obtain and discuss the results from simulations.
1.3 Thesis Outline

The *Introduction* chapter has already presented a summary about the context in which this work has arisen and its objectives. The remaining thesis is structured as follows. Chapter 2 provides an overview on the related work and literature. Chapter 3 introduces the characteristics of simulation tools. Chapter 4 addresses the ITS applications, in particular the *Platooning* use case, presenting the design of the PMP. Chapter 5 presents the process of evaluating the performance of the application. Chapter 6 debates the results obtained from the simulation runs. Finally, Chapter 7 provides the main achievements and future research directions.
Chapter 2

Related Work

This related work review chapter is essentially divided in two parts. The first part intends to provide a brief overview on available publications that cover subjects related to V2X applications, with special attention to advanced applications (e.g. Platooning and CACC). Although essentially focused on V2X applications simulation, it presents some important work that describes systems designs, implementations, communication strategies and so on. The second part discusses some important projects/consortium that have similar goals and aims regarding advanced applications in Vehicular Ad Hoc Networks (VANETs). Not being practicable to introduce all the related projects existent, the main focus has been directed into funded projects, looking into their main objectives and results.

2.1 Related Literature

Milanés et al. [22] present the design, development, implementation and testing of a CACC system. The design of the system is based on controllers that determinate the maneuvers in the platoon: the leader vehicle approaching maneuver and the car-following regulation maneuver. The solution aims to reduce significantly the gaps between the vehicles, taking advantage from information exchanged using DSRC wireless communication. In order to experiment its performance against the commercially available ACC, the system
was implemented and tested on four Infiniti M56s vehicles, in three different scenarios: The first evaluated the behavior of the following cars when the gap settings are changed, the second tested cut-in and cut-out situations and the third compared the CACC performance of the vehicles following a speed change profile with the performance of the traditional ACC system. The authors conclude that the system performed well and was able to reduce the gap variability and handle the cut-in and cut-out situations. Additionally, the CACC improved the response time and platoon stability, when in comparison to the ACC system, proving that the system may be able to improve traffic flow and capacity.

Katsaros et al. [23] propose an implementation of a Green Light Optimized Speed Advisory (GLOSA) application, and evaluates its performance using the V2X Simulation Runtime Infrastructure (VSimRTI) simulation tool. Their goal was to measure the GLOSA effect on fuel and traffic efficiency, resorting to average fuel consumption and average stop time behind a traffic light metrics. The GLOSA application consists in providing the vehicles accurate traffic lights information (using I2V communication), and giving them speed advice, in order to assure a more fluid speed with less stopping time. To test the application, an urban area traffic scenario with a single route and two traffic lights was defined, using the SUMO tool. The simulations had different penetration rates of equipped vehicles and traffic density. The authors conclude that the system could in fact improve the fuel consumption and traffic congestion levels and that the higher the penetration rate is, the more benefits the application can provide, specially in high density scenarios.

Bergenhem et al. [24] survey the state of the art and present an overview of projects related to vehicle Platooning that took place at the time the article was written - Safe Road Trains for the Environment (SARTRE), Partners for Advanced Transportation Technology (PATH), Grand Cooperative Driving Challenge (GCDC), Development of Energy-Saving ITS Technology (Energy ITS) and SCANIAPlatooning project. They conclude their work with a summary and comparison between the different projects using the following parameters: vehicle type, automatic control direction, requirements/potential changes to infrastructure, integration, mainly used primary onboard sensors.
and the main goals of the project regarding Platooning.

Guvenc et al. [25] present the Team Mekar’s CACC implementation at the GCDC. They show both experimental and simulation results, discussing the challenges found. Their CACC implementation was based on V2V communication - exchanging information about the Global Positioning System (GPS) position and velocity, and preceding vehicle acceleration data, resorting to IEEE 802.11p WAVE protocol. The scenarios included both a traffic light and an urban environment.

Katsaros et al. [26] evaluate the impacts of a GLOSA application implementation on efficiency of traffic in urban areas and present its performance resorting to the VSimRTI tool, with Java in Simulation Time/Scalable Wireless Ad-Hoc Network Simulator (JiST/SWANS) and SUMO simulators. Additionally, they also study the behavior of the Adaptive Route Change (ARC) application (route alternation through V2I/V2V communication). The metrics used to analyze the applications were average fuel consumption, average stop time behind a traffic light and average trip time. The authors defined different urban scenarios that included traffic lights, using different penetration rates of vehicles running the applications, vehicles that complied to the recommended speed and traffic density. Their results show that this systems could in fact improve fuel consumption levels and also reduce traffic jams and trip time.

Segata et al. [27] analyze the interference of non-automated vehicles, when a given vehicle is joining a platoon. They define a protocol that supports the join maneuver, and validate it using the Platooning extension (Plexe [28]) from Vehicles in Network Simulation (VEINS), showing if a maneuver can be performed with success or safely aborted. Furthermore, they analyze the impact of the Packet Error Rate (PER) on the maneuver’s rate of success. The results show that their protocol can support complex joining maneuvers and that the dangerous maneuvers can be safely aborted. Additionally, they conclude that platoons remain stable even with high loss rates and that maneuvers are aborted only when networking conditions are bad (PER larger than 10%).

Amoozadeh et al. [29] developed a CACC management protocol based on
IEEE 802.11p communication, including three basic maneuvers (along with commands to accomplish them). These maneuvers enable several operations such as joining and leaving the platoon. The protocol was implemented in the VEhicular NeTwork Open Simulator (VENTOS) simulation platform (which couples SUMO and Objective Modular Network Testbed in C++ (OMNeT++) simulators) and tested in different scenarios and settings (including platoon communication structure, inter-platoon spacing, time-gap and size). To build a better model, they also implemented a more complex longitudinal control system in SUMO. They conclude that the protocol is able to ensure a stable traffic flow and theoretical throughput. The protocol is also able to react to loss in communication, either by using retransmissions or downgrading to ACC mode.

Aissaoui et al. [30] propose a real-time traffic monitoring system that aims to enhance typical reliability, accuracy, and granularity levels, providing information to vehicles with the lowest overhead possible. To accomplished that, they propose a cluster solution, where only the cluster head (the one responsible for providing RSUs the data about all vehicles in the cluster) updates the information. To evaluate the system, they used the Network Simulator 3 (ns-3) version made available by the Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions (iTETRIS) platform (which combines ns-3 and SUMO simulators). They conclude that by using this mechanism it is possible to reduce overhead to one quarter when in comparison to typical approaches, while gathering more than 99% of the existent data.

Bellavista et al. [31] work focus on the implementation and evaluation of protocols that arisen from the COLOMBO project, developed on the iTETRIS platform. In particular, the COLOMBO project aimed to develop local-based cooperative protocols that determine traffic characteristics without the need to communicate with global data centers. COLOMBO aspired to achieve accurate traffic estimations with low penetrations rates of V2X technology equipped vehicles. In this paper, they describe how they implemented the protocols on the platform, together with the learned lessons. The results from the simulations (based on real traffic traces and a real topology) prove
the feasibility of the proposed approach (even with low penetration rates). Segata et al. [32] investigate communication strategies for Platooning and compare the proposed approaches to two beaconing protocols that have been designed for cooperative awareness applications: Decentralized Congestion Control (DCC) and Dynamic Beaconing (DynB). The simulation models were validated and parameterized based on real world experiments. To establish the comparison between the approaches they resorted to PLEXE, which is based on the VEINS tool. Their goal was to evaluate the performance and behavior of the protocols under ”stressful” configurations, regarding the networking and application perspectives. The obtained results show the protocols can greatly improve the performance, both in the application and network layers.

Goebel et al. [33] introduce a trace-based simulation tool chain for V2X applications resorting to cellular networks and real world traces as a basis, assuring that some unpredictability present in other simulation approaches is minimized. To prove their concept, they used the VSimRTI framework (coupling SUMO and OMNeT++ simulators) to compare the performance of the ETSI defined Emergency Warning Application (EWA) against their optimized version. The results show that the developed version is superior in its performance.

Fazio et al. [34] propose an application that aims to advise danger on emergency situations on VANETs resorting to V2V and V2I message exchanges working on top of the IEEE 802.11p standard. To perform an evaluation on the application, the authors executed simulations on various scenarios, in order to retrieve general information that is independent form the scenario. The selected framework was VSimRTI, which permitted the coupling of the SUMO and JiST/SWANS simulators. From the obtained results, they proved that the proposed emergency management application is able to improve the reaction for general vehicles and emergency ones. The designed solution allows vehicles (communicating with the WAVE protocol) to update their path in order to avoid traffic jams near accidents, saving time for reaching their destinations. It allows for emergency vehicles to intervene in less time, managing the accident in a better and safer way, since it also brings
benefits to the injured people, which can be treated faster (the emergency vehicle average speed is higher when running the application). Additionally, the normal traffic flow can also be restored in less time. The application benefits are proportional to the number of vehicles running the application. The application tends to achieve better results in the suburbs, since the roads are more sparse than the central ones.

Krajzewicz et al. [35] evaluates the simulation of the GLOSA application (included in BSA), measuring its effects on traffic efficiency and predicting the results on real-world environments. The GLOSA system suggests vehicles a speed that will allow them to pass the traffic lights when they are green. To evaluate the system, the simulations were performed using SUMO together with a communication model coupled via Traffic Control Interface (TraCI). The main goal of the work was to validate the developed simulation models capabilities. The authors conclude that GLOSA helps in fact vehicles to go through the network, although some vehicles do not need to halt at all and some have to stop (which is assumed to happen due to receiving the traffic lights states too late). The evaluations shown that increasing the number of vehicles reduces the application benefits (even with all vehicles equipped).

2.2 Related Projects

Cooperative Dynamic Formation of Platoons for Safe and Energy-Optimized Goods Transportation (COMPANION) [18] is an ongoing European project that aims to develop co-operative mobility solutions for Platooning, in order to enhance fuel efficiency and safety [36]. They intend to develop and validate a system that is able to form, create and manage platoons. In particular, they expect to create both off-board and on-board coordination and interface solutions and analyze the legislative conditions of Platooning. Finally, the last objective is to demonstrate the validity of the system in proper tracks and through simulation.

Interoperable Grand Cooperative Driving Challenge AutoMation Experience (iGAME) [?] is an ongoing European project that aims to speed real-life implementation and interoperability of automated driving via wireless com-
munication. Its goals is to design a functional architecture and requirements for automated driving, and demonstrating its validity in a multi-vendor challenge. Additionally, they aim to create a control system, an interaction protocol for cooperative applications, tools used for validation and verification, and to standardize interaction messages. The project also specifies scenarios and the needed requirements (functional, communication, etc.).

SARTRE [17] is a project funded by the European Commission that aimed to develop Platooning solutions, with significant environmental, safety and comfort benefits. SARTRE defines Platooning as a set of vehicles driving together with a lead vehicle (driven by a professional) and several followers fully automatically, with short gaps between them. The project investigated the human and safety factors of Platooning. To prove the concept, they developed a system with a lead truck, a following truck, and 3 following cars. Additionally they developed a solution to help vehicles find and join a suitable platoon, but this solution was not integrated in the system. The system was tested on specific tracks in order to measure the fuel consumption benefits. Finally, they studied the commercial viability of the product, infrastructure and environmental concerns, and defined recommended policies to achieve a wider impact.

Cooperative Self-Organizing System for low Carbon Mobility at low Penetration Rates (COLOMBO) project [37] aimed to overcome the problem of most cooperative systems requiring high penetration rates of equipped vehicles for their functionality to be assured. The project’s main goal was to design traffic management components that enable the correct functioning of the system with penetration rates below 10%. The key objectives were to decrease costs and vehicular emissions. The researchers focused on two topics: traffic surveillance (determine the traffic state by collecting data from V2X messages) and traffic light control algorithms (making traffic lights adjust according to the traffic state). As part of the results, the project concludes that the self-organizing traffic control algorithm performs well even with very low penetration rates (1%), based on the metrics of waiting time, number of stops per vehicle in intersections and the emission levels. Additionally, they made available and public a series of open source software and simulation
scenarios. In particular, they developed extension for commonly used tools such as iTETRIS, SUMO, ns-3 and the Passenger Cars and Heavy Duty Emissions Model (PHEM).

There are also other relatively relevant projects that deserve a mention, such as KONVOI [38], CHAUFFEUR II [39], Energy ITS [40], Mobile Opportunistic Traffic Offloading (MOTO) [41] or Partners for Advanced Transportation Technology (PATH) [42], but detailed descriptions have been omitted here.

2.3 Summary

This chapter presented an overview on related work regarding ITS applications (including advanced applications) and on important projects/consortium relative to the same field of study.

Most related literature that aims to analyze the behavior of a given application or solution (e.g. GLOSA, CACC, Platooning, etc.) tends to resort to simulation using frameworks such as VSimRTI and VEINS to evaluate their performance. However, there are several works that test the solutions in experimental environments, which potentially present more accurate results.

Related work focus mostly on information and solutions related to CACC and Platooning, including full-system designs, communication performance analysis or even state of the art surveys.

The related projects focus on advanced applications, giving special emphasis to Platooning, which is currently a very hot topic in the research community.
Chapter 3

ITS Simulation

The development of efficient VANETs advanced systems or applications requires the determination of the main system properties and consequent evaluation of its performance. Performing field tests in vehicular environments is a tough challenge for a series of reasons. The large number of existent vehicles and traffic scenarios makes it very difficult to collect data in the experiments. Moreover, the development of real prototypes is usually very expensive and they take too much time to be prepared and processed. In other hand, resorting to simulations is a popular solution when it comes to evaluate the performance of transportation and communication systems. When using simulation tools, researchers are able to control parameters, configurations, conditions and input data, which are major advantages. Furthermore, tests can be easily accomplished and repeated, and using widely known tools does not require researchers to validate individual models, allowing them to focus only on developing solutions. Although both solutions (testbed and simulation) allow the study of the system, simulation is able to perform assessments in a larger scale. However, the simulation method normally assumes the use of simpler models, which may reduce the system realism - the accuracy of their implementation may affect the results veracity. In summary, simulating VANETs brings benefits in terms of cost, repeatability, scale and practical requirements. For this reason, researches tend to resort to simulation rather than field testing.
3.1 Simulation Tools

To perform a proper simulation of VANETs, both a traffic and a network simulator are required, although they may work independently. The aim of this section is to identify and describe the most widely used tools and frameworks when performing VANETs simulations. First, some important networks simulators are presented, followed by traffic simulators and finally also some important platforms/tools that integrate both traffic and networks simulators.

3.1.1 Network Simulation

Network simulation is one of the most prominent evaluation methods in computer networks. This method is commonly used for developing new communication protocols and architectures.

Network Simulator 3 (ns-3) [43] is a well-known discrete event simulator for communication networks that evolved from Network Simulator 2 (ns-2) [44]. This evolution brought several improvements such as scalability and memory usage - ns-3’s design incorporated architectural concepts from Georgia Tech Network Simulator (GTNetS) [45], known for being a simulator with good scalability. ns-3 supports realistic models of wireless communications for cooperative ITS applications - it provides models to emulate radio propagation effects and functionalities and protocols for all layers of the ITS-G5 stack.

The Objective Modular Network Testbed in C++ (OMNeT++) [46] open source tool is an extensible, component-based C++ discrete event network simulator, known for scaling well for large networks. Although not providing any specific components for computer network simulations, these models are developed independently - its user community has provided support mainly for standard wired and wireless IP networks. OMNeT++ has two main frameworks that benefit from a specially designed propagation model for V2X: Mixed Simulator (MiXiM) [47] and INET Framework [48].

Java in Simulation Time (JiST) [49] is a discrete event simulation engine written in Java, that allows the simulation of real Java applications. Although allowing general purpose network simulation, it is commonly used
together with Scalable Wireless Ad-Hoc Network Simulator (SWANS) [49] - a highly scalable mobile ad hoc networks simulator, that is able to simulate scenarios of more than ten thousand nodes. Unfortunately, the official development of JiST is no longer maintained, and the latest version does not include neither mobility nor propagation models for VANETs.

### 3.1.2 Traffic Simulation

Traffic simulation tools emulate the traffic flow in transportation networks. Traditionally, traffic simulators can be classified as microscopic, macroscopic, and mesoscopic, depending on the granularity: microscopic simulators model traffic at a large scale, treating each entity (e.g. cars, trains) individually; macroscopic tools are used to model traffic at a large scale, treating traffic like a fluid; mesoscopic simulation combines the properties of both microscopic and macroscopic simulation models, describing the reality in terms of aggregated traffic platoons of vehicles characteristics, avoiding the time and complexity issues of implementing microscopic scenarios. However, VANETs scenarios required accurate models of communication between nodes and their exact position, which make both macroscopic and mesoscopic not so reliable solutions since they offer less detail when in comparison to microscopic simulations.

Simulation of Urban MObility (SUMO) [50] is an open source and highly portable microscopic simulator widely used for VANETs research and typically used to simulate automatic driving or traffic management strategies [51]. It possesses several features such as large road networks, collision free vehicle movement, different vehicle types, single-vehicle routing, multi-lane streets with lane changing, etc. Additionally, SUMO includes a visualizer that shows the road topology and nodes movement during the simulation runtime. It is possible to combine SUMO with OpenStreetMap (OSM) [52] and simulate traffic in any location in the world. Unfortunately, SUMO is a pure traffic generator, which means that the traces generated by the tool cannot be easily used by network simulators (in a direct way). The TraCI API existent in SUMO allows it to act as a server and connecting to an ex-
ternal application using a Transmission Control Protocol (TCP) socket.
MObility model generator for VElicular networks (MOVE) [53] is a realistic
mobility models generator that is built on top of SUMO. MOVE generates
mobility traces with data from realistic vehicles movements that can later be
used in other network simulators (e.g. ns-2). Additionally, MOVE provides
a Graphical User Interface (GUI) that allows an easy way to generate sce-
narios.
Verkehr In Stadten SIMulationsmodell (VISSIM) [54] is a multi-modal mi-
croscopic, time-step and behavior-based traffic simulation tool. VISSIM is
considered a leader in the field of micro-simulation software. It is a frame-
work that provides a powerful GUI that allows users to define maps and
scenarios, and the traffic conditions can be visualized in a very high level
of detail. This tool implements a car-following traffic model that takes into
consideration psychological aspects of drivers. Additionally, it also provides
a pedestrian mobility model, which sometimes is useful in urban scenarios. It
is able to analyze traffic under constraints such as lane configuration, speed
limits, traffic signals, and so on. VISSIM allows the following types of traffic
to be simulate: vehicles (cars and trucks), trams, buses, bicycles, motorcy-
cles, pedestrians and even rickshaws.
VanetMobiSim [55] is an extension for the CANU Mobility Simulation Envi-
ronment (CanuMobiSim) [56], a flexible framework that is used for modeling
mobility. CanuMobiSim is able to generate trace files in different formats,
supporting different mobility simulation software. VanetMobiSim provides
vehicular mobility, with both microscopic and macroscopic realistic vehic-
ular movement models. At macroscopic level, it supports multi-lanes, bi-
directional flows, differentiated speed constraints and traffic signs. At mi-
croscopic level, VanetMobiSim implements V2V and V2I models that enable
vehicles to regulate speed, overtake maneuvers and respecting traffic signs
in intersections. VanetMobiSim mobility models have been validated by the
well known Traffic Software Integrated System - Corridor Simulation (TSIS-
CORSIM) [57] traffic generator.
### 3.1.3 Coupled Simulation

Although mobility and network simulators work in an independent way, there are some tools that allow their interconnection, which enables them to interact with each other. These systems are usually of type: i) isolated, ii) federated or iii) integrated. In i) isolated solutions, the mobility tool generates a trace file which is later processed by the network simulator. In ii) federated mode, the simulators communicate through a two-way interface: the network simulator controls the traffic simulator through commands and the traffic tool reports back information about the vehicles (e.g. position). In the iii) integrated mode, a single simulator handles both traffic and network simulation simultaneously.

Vehicles in Network Simulation (VEINS) [58] is an open source framework that provides a control interface between SUMO and OMNeT++ via TCP sockets. VEINS framework development was based on MiXiM. The communication between the simulators (network and mobility) is accomplished though the use of SUMO’s TraCI. Both simulators are bidirectionally coupled and run in parallel, which means traffic has direct impact on network performance and vice versa. In this framework, the OMNeT++ simulator is directly extended - it is able to control and send commands to vehicles directly (e.g. orders to change speed or path). VEINS contains modules that implement the IEEE 802.11p standard and the higher layer of the DSRC stack.

Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions (iTETRIS) [59] is an open source simulation platform developed under the European FP7 Program and it is characterized by a modular architecture that allows the integration of two widely used traffic and wireless simulators (SUMO and ns-3), and supports the implementation of cooperative ITS applications in various programming languages. iTETRIS implements a new central block called iTETRIS Control System (iCS). The iCS handles the SUMO and ns-3 interaction, in addition to preparing, triggering, coordinating and controlling the execution of simulations. The modularity of iTETRIS source code allows the coupled use of other open source
simulators, without having to modify their internal modules.

V2X Simulation Runtime Infrastructure (VSimRTI) [60] is a general framework for the evaluation of VANETs solutions that is able to couple different simulators each for a particular domain (networking and mobility). The VSimRTI modules are responsible for handling all the management tasks - time synchronization, interaction, communication, data exchange and so on. The integration of simulators is enabled by the implementation of generic VSimRTI interfaces. This framework provides also a simple network and a cellular communication simulator, a Java-based application simulator, and a battery simulator. Additionally, it provides visualizer and analysis tools. Furthermore, VSimRTI is being extended to enable the simulation of electric mobility environments.

3.2 Summary

The increasing popularity of VANETs lead the research community to develop accurate and realistic simulation tools for these environments. Given the fact that test fielding in VANETs is a tough challenge, researches resort to the use of simulation tools to evaluate the performance of applications, communication systems and so on.

This chapter reviewed some basic general aspects of ITS simulation and surveyed the most prominent tools available to perform network, traffic and coupled simulations. Regarding networking, the ns-3 and OMNeT++ are tools widely used and validated by the research community, while JiST development is no longer maintained. In traffic and mobility simulation, SUMO is a very popular solution that has proved its reliability and performance within the ITS community. Although not as strong as SUMO, VanetMobiSim is also often used for modeling mobility in VANETs. For these reasons, it is quite natural that most coupled frameworks tend to resort to the use of these powerful simulators, in order to render the simulation results as accurate as possible when comparing to real life.
Chapter 4

ITS Applications Taxonomy

Considering the availability of a wide range of applications (with their corresponding requirements), it is important for the research community to establish reference values for the requirements of each type of existing applications and use cases. This chapter presents an overview of the related work and a summary of the main aspects that build a taxonomy of applications. Furthermore, this chapter introduces a detailed description of the Platooning use case and presents the proposed Platooning Management Protocol (PMP).

4.1 Application Taxonomy

Regarding the related work of characterizing applications, Lèbre et al. [61] introduce concepts and specific vocabulary in order to classify current innovations or ideas on the emerging topic of smart cars. They organized the solutions according to their societal and scalable evolution.

Dar et al. [2] summarize the communication characteristics and features of typical applications from the three main types (Safety, Efficiency and Comfort). Additionally, they provide a mapping methodology that allows to select for each class of applications a suitable communication media.

Papadimitratos et al. [62] summarize the state-of-the-art solutions (available at the time) from a broad range of projects, and present a representative list of vehicle applications and their requirements. The applications
names are closely compatible with those used by projects such as Car2Car Communication Consortium (C2C-CC) [63], SAFESPOT [64] or Cooperative Vehicle-Infrastructure Systems (CVIS) [65]. Although their work illustrates requirements for different kinds of applications, the values are not proposed, but rather based on several technical reports, such as [66] or [4].

Sepulcre and Gozalvez [67] illustrate the importance of considering the requirements of cooperative applications when designing congestion control protocols. They demonstrate the application requirements impact on the communication settings of each vehicle and on the overall channel load.

Nekovee [68] specifies bounds for the maximum message delivery latency and reliability requirements of the V2V communication protocols for rear-end collision avoidance applications.

Karagiannis et al. [69] compile the safety use cases and their respective communication performance requirements from the main existent initiatives in US, Europe and Japan. Additionally, they establish relations between applications and the possible carriers based on their requirements.

Mahmod et al. [3] establish communication requirements for applications suggested for the CVIS and SAFESPOT projects, as well as general safety applications that are related to the applications from the PReVENT project (not existing anymore). To quantify the communication requirements the authors assigned values within a certain range: latency is classified as very short (VS - less than 100 ms), short (S - ms to one second), medium (M - seconds to one minute), and long (L - minutes); A small bandwidth (S) is often used for short safety messages, medium (M) for other traffic related information, and high (H) for multimedia services; Communication range can be short (S - up to 1000m), medium (M - few km) or long (L - several km); Reliability can assume the values of high (H - critical data), medium (M - non-critical data), and low (L - media data download); Priority is usually high (H) for information that requires immediate attention from the driver, medium (M - if it requires special attention but does not pose imminent danger) or low (L - e.g., commercial or entertainment information). Table 4.1 illustrates the summary of their work - in short, the characterization of the requirements for the applications introduced by the related projects.
<table>
<thead>
<tr>
<th>Application</th>
<th>Information Transmission Control</th>
<th>Com. Mode</th>
<th>Addressing</th>
<th>Direction</th>
<th>Latency</th>
<th>Bandwidth</th>
<th>Range</th>
<th>Reliability</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVIS CURB: Flexible Lane Allocation</td>
<td>Event</td>
<td>V2I</td>
<td>E2I</td>
<td>E2C</td>
<td>P2P</td>
<td>Two-Way</td>
<td>M-L</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>CVIS CURB: Local Traffic Control</td>
<td>Event</td>
<td>V2I</td>
<td>E2I</td>
<td>E2C</td>
<td>P2P</td>
<td>Two-Way</td>
<td>S-M</td>
<td>M</td>
<td>S</td>
</tr>
<tr>
<td>CVIS CINT: Travelers Assistance</td>
<td>Event</td>
<td>V2I</td>
<td>E2I</td>
<td>E2C</td>
<td>P2MultiP</td>
<td>Two-Way</td>
<td>M</td>
<td>M</td>
<td>M-L</td>
</tr>
<tr>
<td>SAFESPOT - General Active Safety Applications</td>
<td>Event</td>
<td>V2V</td>
<td>V2I</td>
<td>P2MultiP</td>
<td>One-Way</td>
<td>VS-S</td>
<td>S</td>
<td>S</td>
<td>H</td>
</tr>
</tbody>
</table>

Table 4.1: ITS applications and their communication requirements [3]

The **ETSI TR 102 638** standard [4] describes a Basic Set of Applications (BSA), focusing on V2V, V2I and I2V communications in the V2X dedicated frequency band (although it does not exclude the use of alternative technologies, such as cellular networks). The BSA was developed in conjunction with other European research projects such as *Car2Car* [63] and *EU PRE-DRIVE C2X* (information on the project now available on [70]). The document serves as a reference for developing standards for applications in the BSA and other ITS services. Each application/use case specification has taken into consideration several criteria, such as strategic requirements, system capabilities
requirements, legal requirements and so on. These specifications are the first step taken in order to deploy these standardized applications. ETSI defines four classes of applications: *Active road safety* (equivalent to *Safety* applications), *cooperative traffic efficiency* (identical to *Mobility* and *Efficiency* applications), *co-operative local services* and *global internet services* (correspondent to *Connectivity* and *Convenience* applications). *Co-operative local services* are provided from within the ITS network, while *global internet services* are obtained from providers in the wider internet. The BSA is illustrated in Table 4.2.

*Annex C* from the standard contains a catalog of V2X use cases on which the BSA was based. The catalog of applications and use cases represents only part of the nowadays existing solutions/proposals, since it was based on the state of the art at the time of publication. It provides knowledge on latency, minimum frequency, messaging type, communication mode and some other requirements for every use case. Table 4.3 sums up the information on the requirements for a few typical use cases from each class type. The table does not address all applications from the BSA to ease its reading and comprehension. Nonetheless, it contains all crucial information and the reference values for application classes.

The *CAR 2 CAR Communication Consortium Manifesto* document [5] introduces an overview of the *CAR 2 X Communication System*. It was designed based on a vast collection of use cases and their requirements. In addition to the description of these use cases (giving emphasis to its potential regarding safety, traffic efficiency, and comfort, and the benefits of V2V communication), this work also aggregates them into six types of applications, based on their security requirements and types of information exchange (e.g. the *Emergency Electronic Brake Lights* and *Approaching Emergency Vehicle Warning* use cases are part of *Vehicle 2 Vehicle Cooperative Awareness* applications). Table 4.4 states the general requirements for each existing type of application.
<table>
<thead>
<tr>
<th>Application Class</th>
<th>Application</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active road safety</td>
<td>Driving assistance - Co-operative awareness</td>
<td>Emergency vehicle warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Slow vehicle indication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intersection collision warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Motorcycle approaching indication</td>
</tr>
<tr>
<td></td>
<td>Driving assistance - Road hazard warning</td>
<td>Emergency electronic brake lights</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wrong way driving warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stationary vehicle - accident</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stationary vehicle - vehicle problem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Traffic condition warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Signal violation warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Roadwork warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Collision risk warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decentralized floating car data - Hazardous Location</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decentralized floating car data - Precipitations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decentralized floating car data - Road adhesion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decentralized floating car data - Visibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decentralized floating car data - Wind</td>
</tr>
<tr>
<td>Cooperative traffic efficiency</td>
<td>Speed management</td>
<td>Regulatory/contextual speed limits notification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Traffic light optimal speed advisory</td>
</tr>
<tr>
<td></td>
<td>Co-operative navigation</td>
<td>Traffic information and recommended itinerary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enhanced route guidance and navigation</td>
</tr>
<tr>
<td></td>
<td>Co-operative local services</td>
<td>Limited access warning and detour notification</td>
</tr>
<tr>
<td></td>
<td>Location based services</td>
<td>In-vehicle signage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Point of interest notification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Automatic access control and parking management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TTS local electronic commerce</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Media downloading</td>
</tr>
<tr>
<td>Global internet services</td>
<td>Communities services</td>
<td>Insurance and financial services</td>
</tr>
<tr>
<td></td>
<td>TTS station life cycle management</td>
<td>Fleet management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vehicle software/data provisioning and update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vehicle and RSU data calibration</td>
</tr>
</tbody>
</table>

Table 4.2: BSA Definition [4]
<table>
<thead>
<tr>
<th>Application Class</th>
<th>Application</th>
<th>Maximum Latency</th>
<th>Minimum Frequency</th>
<th>Messaging Type</th>
<th>Com. Mode</th>
<th>Other Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Safety</td>
<td>Emergency Vehicle Warning</td>
<td>100 ms</td>
<td>10 Hz</td>
<td>Periodic Triggered</td>
<td>V2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Slow Vehicle Warning</td>
<td>100 ms</td>
<td>2 Hz</td>
<td>Periodic Triggered</td>
<td>V2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Emergency Electronic Break Lights</td>
<td>100 ms</td>
<td>10 Hz</td>
<td>Time-limited Broadcast</td>
<td>V2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stationary Vehicle Warning</td>
<td>100 ms</td>
<td>10 Hz</td>
<td>Time-limited Broadcast</td>
<td>V2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Traffic Condition Warning</td>
<td>1 Hz</td>
<td></td>
<td>Time-limited Broadcast</td>
<td>V2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Roadwork Warning</td>
<td>100 ms</td>
<td>2 Hz</td>
<td>Time-limited Broadcast</td>
<td>V2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Collision Risk Warning from RSU</td>
<td>100 ms</td>
<td>10 Hz</td>
<td>Time-limited Broadcast Event</td>
<td>V2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decentralized Floating Car Data</td>
<td>1 to 10 Hz</td>
<td></td>
<td>Time-limited Broadcast Event</td>
<td>V2V</td>
<td></td>
</tr>
<tr>
<td>Traffic</td>
<td>Regulatory Speed Limits</td>
<td>100 ms</td>
<td>1 to 10 Hz</td>
<td>Periodic Broadcast</td>
<td>I2V</td>
<td>Minimum positioning accuracy: better than 5m</td>
</tr>
<tr>
<td></td>
<td>Traffic Light Optimal Speed Advisory</td>
<td>100 ms</td>
<td>2 Hz</td>
<td>Periodic Broadcast</td>
<td>I2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Traffic Information and Recommended Itinerary</td>
<td>500 ms</td>
<td>1 to 10 Hz</td>
<td>Periodic Broadcast</td>
<td>I2V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enhanced Route Guidance and Navigation</td>
<td>500 ms</td>
<td>1 Hz</td>
<td>On-Demand</td>
<td>I2V</td>
<td>Internet access: IPv6 is required</td>
</tr>
<tr>
<td></td>
<td>Limited Access Warning, Detour Notification</td>
<td>500 ms</td>
<td>1 to 10 Hz</td>
<td>Periodic Broadcast</td>
<td>I2V</td>
<td></td>
</tr>
<tr>
<td>Comfort</td>
<td>Point of Interest Notification</td>
<td>500 ms</td>
<td>1 Hz</td>
<td>Periodic Broadcast</td>
<td>I2V</td>
<td>Internet access: IPv6 is required</td>
</tr>
<tr>
<td></td>
<td>Media Downloading</td>
<td>500 ms</td>
<td>1 Hz</td>
<td>On-Demand</td>
<td>I2V</td>
<td>Internet access: IPv6 is required</td>
</tr>
<tr>
<td></td>
<td>Insurance and Financial Services</td>
<td>500 ms</td>
<td>1 Hz</td>
<td>On-Demand</td>
<td>I2V</td>
<td>Internet access: IPv6 is required</td>
</tr>
<tr>
<td></td>
<td>Vehicle Software Data Provisioning and Update</td>
<td>500 ms</td>
<td>1 Hz</td>
<td>On-Demand</td>
<td>I2V</td>
<td>Internet access: IPv6 is required</td>
</tr>
</tbody>
</table>

Table 4.3: Requirements of ITS use cases [4]

Typical active road safety applications coverage distances go from 300 meters to 20 kilometers. Traffic efficiency and management applications vary from 300 meters to 5 kilometers if they manage speed, or between 0 meters and 1000 meters if they are of co-operative navigation type. Comfort applications have a coverage distance from 0 m to full communication range, depending on the use case.
<table>
<thead>
<tr>
<th>Application</th>
<th>Communication Type</th>
<th>Communication Range</th>
<th>Roadside Units</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>V2V Cooperative Awareness</td>
<td>Broadcast</td>
<td>300 m to 1 km</td>
<td>N/A</td>
<td>V2V Trust</td>
</tr>
<tr>
<td>V2V Unicast Exchange</td>
<td>Unicast</td>
<td>0 m to 5 km</td>
<td>N/A</td>
<td>V2V Trust</td>
</tr>
<tr>
<td>V2V Decentralized Environmental Notification</td>
<td>Broadcast</td>
<td>300 m to 20 km</td>
<td>Not required but can aid applications</td>
<td>Originator Trust</td>
</tr>
<tr>
<td></td>
<td>Geocast</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2V (One-way)</td>
<td>Broadcast</td>
<td>300 m to 5 km</td>
<td>Required</td>
<td>Vehicle must trust RSU</td>
</tr>
<tr>
<td></td>
<td>Geocast</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local RSU Connection</td>
<td>Unicast</td>
<td>0 m to 1 km</td>
<td>Required</td>
<td>RSU/OBU must trust each other</td>
</tr>
<tr>
<td>Internet Protocol</td>
<td>Unicast</td>
<td>0 m to full radio range. Can be extended by Multihop</td>
<td>Required</td>
<td>Internet Security (IPsec, application layer security)</td>
</tr>
<tr>
<td>RSU Connection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: General requirements for applications [5]

The match between the BSA and the types defined by [5] is illustrated in Table 4.5. Although the applications from the V2V unicast exchange type do not belong to the BSA, they are represented for illustrative purposes. Collision Risk Warning can be either of type V2V Cooperative Awareness or V2V Decentralized Environmental Notification, depending on the use case. Table 4.6 summarizes the minimum requisites for the application types defined before. The most strict applications in terms of safety have a maximum latency of 100 ms and a minimum frequency range between 2 and 10 Hz, with ranges between 300m and 1km. Despite being slightly more relaxed and flexible in terms of latency and range, soft safety applications have similar requirements to hard safety applications and their differences are not very significant. Traffic efficiency applications can tolerate higher latencies (usually 500 ms, with some exceptions) and have higher ranges - can go up to tens of kilometers, depending on the access technology being used. Finally, comfort applications are typically characterized for having a maximum latency of 500 ms, minimum frequencies of 1 Hz and covering full radio range.
<table>
<thead>
<tr>
<th>Type</th>
<th>Applications</th>
</tr>
</thead>
</table>
| V2V Cooperative Awareness| Emergency vehicle warning  
Intersection collision warning  
Motorcycle approaching indication  
Emergency electronic brake lights  
Wrong way driving warning  
Collision risk warning |
| V2V Unicast Exchange     | Pre-Crash Sensing/Warning  
V2V Merging Assistance  
Cooperative Vehicle-Highway Automation System (Platoon)  
Instant Messaging |
| V2V Decentralized  Environmental Notification | Slow vehicle indication  
Stationary vehicle  
Traffic condition warning  
Collision risk warning  
Decentralized floating car data |
| I2V (One-way)            | Regulatory / contextual speed limits notification  
Traffic light optimal speed advisory  
Limited access warning and detour notification  
In-vehicle signage  
Signal violation warning  
Roadwork warning |
| Local RSU Connection     | Traffic information and recommended itinerary  
Point of Interest notification  
Automatic access control and parking management  
ITS local electronic commerce  
Loading zone management  
Vehicle software / data provisioning and update |
| Internet Protocol RSU Connection | Enhanced route guidance and navigation  
Media downloading  
Insurance and financial services  
Fleet management  
Vehicle and RSU data calibration |

Table 4.5: ETSI applications according to CAR2CAR classification
Table 4.6: Applications Requirements Summary

<table>
<thead>
<tr>
<th>Applications</th>
<th>Maximum Latency</th>
<th>Minimum Frequency</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hard-Safety</td>
<td>100 ms</td>
<td>2 Hz to 10 Hz</td>
<td>300 m to 1 km</td>
</tr>
<tr>
<td>Soft-Safety</td>
<td>100 ms</td>
<td>1 Hz to 10 Hz</td>
<td>300 m to 20 km</td>
</tr>
<tr>
<td>Traffic Efficiency</td>
<td>100 ms to 500 ms</td>
<td>1 Hz to 10 Hz</td>
<td>Full radio range</td>
</tr>
<tr>
<td>Comfort</td>
<td>500 ms</td>
<td>1 Hz</td>
<td>Full radio range</td>
</tr>
</tbody>
</table>

Communication technologies for ITS applications

Dar et al. [2] present a mapping between ITS applications categories and potential communication technologies. DSRC (sometimes it is also referred as WAVE also) is a typical communication technology that is used by safety and efficiency applications - particularly when applications demand V2V and/or V2I direct communication with a range within 1 km. Cellular and WiMAX (Worldwide Interoperability for Microwave Access) technologies can be considered in V2I or I2I cases, where a long range communication is required. Cellular technologies (e.g. 3G or 4G) are a good fit for Internet access (if medium data rates are acceptable) and they have the advantage of already possessing infrastructures that are provided by network operators. On the other hand, WiMAX can also provide users with high speed Internet access - however, it requires the installations of WiMAX base stations. The DVB/DAB (Digital Video Broadcasting/Digital Audio Broadcasting) technologies have potential to be used within specific ITS applications for broadcasting purposes (such as traffic management and road condition monitoring applications). DVB-H (Digital Video Broadcasting - Handheld) can also be considered to be used in broadcasting audio/video programs for infotainment. MMWAVE (millimeter wave) can be employed as a high speed air interface for devices that are within vehicles. Table 4.7 complements the mapping between applications and recommended carriers.
### Table 4.7: Potential Wireless Communication Technologies for ITS Applications

<table>
<thead>
<tr>
<th>Applications</th>
<th>ITS Application Category</th>
<th>Recommended Carriers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety</td>
<td>Collision Avoidance</td>
<td>DSRC/WAVE</td>
</tr>
<tr>
<td></td>
<td>Road Sign Notifications</td>
<td>DSRC/WAVE, CALM, Li-Fi</td>
</tr>
<tr>
<td></td>
<td>Incident Management</td>
<td>DSRC/WAVE, Cellular Networks</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Traffic Management</td>
<td>DSRC, WiMax, Cellular Networks</td>
</tr>
<tr>
<td></td>
<td>Road Monitoring</td>
<td>DSRC/WAVE, Cellular Networks, Li-Fi, ZigBee</td>
</tr>
<tr>
<td>Comfort</td>
<td>Entertainment</td>
<td>DSRC/WAVE, Cellular Networks, Li-Fi, ZigBee, WiMax, Bluetooth</td>
</tr>
<tr>
<td></td>
<td>Contextual Information</td>
<td>DSRC/WAVE, Cellular Networks, Li-Fi</td>
</tr>
</tbody>
</table>

### 4.2 Platooning Use Case

The *Platooning* application, introduced before on advanced ITS applications section 1.1.3, is a solution that allows vehicles to travel very close to each other in groups (*platoon*) with automated velocity and steering control. Driving in *platoon* with automatic control enables the enhancement of safety, traffic flow and highway capacities, while also providing drivers with a more convenient and comfortable driving experience. Furthermore, it helps to save energy and fuel, while reducing emissions [71–74]. Figure 4.1 illustrates an example platoon of vehicles in an highway, identified in between the red frames.

![Figure 4.1: Platoon of vehicles](image)
The simplest way of implementing *Platooning* is through the use of V2V communication, where vehicles only share information with their immediate predecessor. In these systems, vehicles receive notifications from the preceding vehicles (about acceleration changes, breaking values, etc.) within constant intervals (typically, with an update frequency of 10Hz), allowing them to respond accordingly - adjusting their speed and position, and so on. However, there is a major issue regarding communication delay since the delay values accumulate from vehicle to vehicle until the last group follower receives the message. More advanced solutions disseminate information not only about the preceding vehicles but also from vehicles that are not in line of sight, providing the driver with situational awareness feedback to understand the status of his vehicle and the whole group. When vehicles possess group information in advance (sent by the group leader), they can predict the behavior of vehicles in front, which helps to stabilize the platoon - similarly to real world defensive drivers paying close attention to road to predict what might happen. Another way of overcoming the lack of visibility from the drivers is by providing a video stream from the front group leader view, but this solution is not as strong as the cooperative exchange of information. The implementation of *Platooning* systems can either be resorting to infrastructures (V2I communication) or not. Although V2V communication is often the choice for highway environments, V2I communication seems to bring more benefits on urban areas, where RSUs and TMCs provide information such as recommended speeds to vehicles.

Despite *Platooning* being a different application from CACC, there are a lot of improvements and results from previous research that are useful and can easily be moved and applied on *Platooning* as well. Since CACC and *Platooning* share the same concerns on distance gaps, one should take into account the results from several works such as Nowakowski et al. [75], which state that drivers are generally comfortable with short following time gaps (about 0.6 seconds) when using CACC, which results an increased highway capacity. The work from Shladover et al. [76] also shows that by using short following gaps in CACC systems, the lane capacity can be increased from around 2000 vehicles per hour to about 4000 vehicles, when moving from zero to one hun-
dred percent market penetration. Ploeg et al. [77] analyzed the properties of string-stability in CACC and concluded that time gaps below one second are possible. The ideal time gap value was of 0.7 seconds, although it is possible to reduce the gaps down to 0.3 seconds (when minimizing the latency of the communication).

Another main concern in Platooning is the size of the chain (usually named as string in CACC systems), for reasons such as security or performance. Although being possible to establish the maximum length of the chain as the distance between the first and last vehicle, it is typically defined as the maximum number of vehicles. The first studies [78] claimed the ideal size to be around 20 vehicles but, a more recent study [79] from the SARTRE project advises the maximum platoon size to be 15.

4.3 Platooning Management Protocol

Platooning is an application that has the potential to bring several benefits, but it requires a very efficient Platooning Management Protocol (PMP) that specifies all the required maneuvers and proper communication behaviors, so that a good vehicle’s cooperation performance is achieved. The following section describes the proposed PMP, including a full description of the maneuvers and also the specification of the requirements based on European standards.

4.3.1 Maneuvers

This subsection presents the main assumptions regarding the PMP maneuvers as well as their complete description.

Create

The Create maneuver starts when a given vehicles tries to join a platoon but there are no available strings around him (may be non-existent or already have the maximum number of vehicles permitted). Hence, the Create maneuver results from a previous failed Join maneuver (described in detail
A vehicle may create a new platoon if it possess a special license that allows him to do so. The process of creating a platoon is listed below:

1. Leader vehicle starts a new Platoon \((ID=\text{CarName}_\text{PlatoonNumber})\).

2. Leader vehicle propagates the Platoon existence, broadcasting the ID every second.

Figure 4.2 illustrates this use case.

Join

The Join maneuver is triggered when a vehicle wants to join a platoon. In this solution, the platoon choice is automatic, although the driver may be able to choose one of its preference in a real world application. An important aspect of the Join maneuver is the string ordering. The simplest solution of all is to make vehicles join the platoon tail. In contrast, joining the platoon at the front of the string is harder, since it assumes a change of rolls in the string (a new Leader is chosen). Allowing vehicles to join the platoon in any position, enables the vehicle’s string to be ordered by one of several parameters: engine, weight or braking performance, aerodynamics, distance to be traveled, and so on. In this cases, vehicles open a gap that allows the joining vehicle to merge to the correct lane. This solution is more challenging, since it requires more coordination between several vehicles. Joining the string becomes even more difficult when traffic is dense and there are vehicles...
traveling close to the \textit{platoon}. If the maneuver cannot be completed safely, it is aborted. A vehicles is able to join a \textit{platoon} if the string does not exceed its maximum length and if no other maneuver is occurring. Additionally, there may exist several other requirements that the vehicles must fulfill (e.g. vehicles must meet the platoon velocity).

The process of joining a \textit{platoon} is listed below:

1. \textit{Joiner} sends a \textit{Join Request} (broadcasted every second) until it finds an available \textit{Platoon} or \textit{Time Out} is exceed.

2. \textit{Leader} responds with a \textit{Join Acknowledgment} if it’s possible for the vehicle to join. Otherwise, it responds with a \textit{Join Reject} and the maneuver is aborted.

3. \textit{Joiner} moves to the correct position in order to merge to correct lane and informs the \textit{Leader} with a \textit{Distance Achieved} message.

4. \textit{Leader} notifies the \textit{Followers} to open up a gap, with a \textit{Adjust Gap} message, unless the \textit{Joiner} is joining in the rear of the \textit{platoon}.

5. \textit{Followers} notify the \textit{Leader} when the adjusting process is completed with \textit{Adjust Gap Acknowledgments}.

6. \textit{Leader} sends a \textit{Start Maneuver} message, informing the \textit{Join Follower} that the maneuver can be accomplished.

7. \textit{Joiner} merges to the correct lane and enters automatic mode, notifying the \textit{Leader} with a \textit{Maneuver Completed} message.

8. \textit{Leader} sends a \textit{Platoon Update} message for all \textit{Followers} with updated information.

Figures 4.3 to 4.6 illustrate the \textit{Join} maneuver.
Dissolve

The *Dissolve* maneuver happens when the *Leader* decides to disassemble the string. This may happen for several reasons: For example, the *Leader* may leave the *platoon*, there are obstacles on the road ahead or all vehicles left the
platoon. The Leader may only follow manual driving mode after all Followers acknowledge the command to leave the string. The steps of the use case are described below and illustrated on Figures 4.7 to 4.10:

1. Leader sends a Dissolve Request.

2. Followers enter manual driving mode and send back a Dissolve Acknowledgment to the Leader.

3. When all Followers respond (if any), the Leader enters manual driving mode and dissolves the platoon. Additionally, it stops broadcasting its existence.

Figure 4.7: Dissolve maneuver - Step 1

Figure 4.8: Dissolve maneuver - Step 2
Leave

The Leave maneuver is initiated when a given Follower needs to exit the platoon (e.g. reaching its destination). It informs the Leader about its decision and then awaits for its response, in order to assume manual control and change lane. Due to complexity of the maneuver, only one vehicle may leave the platoon at a time. The maneuver steps are detailed below and in Figures 4.11 to 4.15.

1. Follower sends a Leave Request.
2. Leader computes the new gap distances for Followers, informing them with Adjust Gap message.
3. Followers acknowledge new distances, resorting to Adjust Gap Acknowledgment messages.
4. Leader returns a Start Maneuver message for the Leaver.
5. Leaver shifts to manual driving and changes lane.
6. Leaver notifies the platoon Leader with a Maneuver Completed message.
7. Leader notifies Followers with a Platoon Update message.
Merge

The *Merge* maneuver consists on the process of joining two *platoons* that are traveling on the same lane. This maneuver is only possible if the size of the two *platoons* combined is less that the maximum length allowed. Typically,
the process is initiated by the Leader of the rear platoon and the front platoon leader decides whether to accept or reject the request (in case of exceeding the maximum length or if the front platoon is performing another maneuver, which makes the Merge process not possible). The following steps and Figures 4.16 to 4.20 show how the Merge maneuver is performed. To ease the reading, the front Leader and platoon are referred as Leader A and platoon A, while the rear Leader and platoon are referred as Leader B and platoon B.

1. Leaders (both A and B) send Merge Requests through broadcast every 10 seconds.

2. Leader A receives the request and responds with a Merge Acknowledgment.

3. Leaders exchange Platoon Info messages with information to determine the new platoon.

4. Leader B sends a Adjust Gap message to Leader A.

5. Leader B moves platoon B to the rear of platoon A.

6. Leader A acknowledges the notification with a Adjust Gap Acknowledgment message.

7. Leader B sends a New Leader message to its Followers.

8. Leader B assumes a Follower role.
Figure 4.16: Merge maneuver - Steps 1 and 2

Figure 4.17: Merge maneuver - Step 3

Figure 4.18: Merge maneuver - Steps 4 to 6

Figure 4.19: Merge maneuver - Steps 7 and 8
4.3.2 Platooning Requirements

This subsection defines the most important functional and communication requirements for the platooning application. The requirements are split for each maneuver existent. The requirements were essentially based on the ETSI TR 102 638 standard [4], which provides the main requirements for a Co-operative vehicle-highway automation system (Platoon) use case. This standard defines that vehicles should be able to broadcast V2X Cooperative Awareness Message (CAM) messages and to establish unicast connections with other vehicles. The maximum latency should be of 100 ms, the minimum frequency of messages should have at least a value of 2 Hz and the vehicles relative positioning accuracy should be better than 2 m. Hence, as general requirements, the latency was defined to have a maximum value of 100 ms, the communication range should able to reach 1 km and the relative position accuracy should be equal or better than 1 m. Additionally, it is defined that the authenticity of the V2X messages should be verified. Some of the parameters were slightly adjusted depending on the maneuver and the information exchanged, since the events in the platoon don't have all the same level of demand. The requirements are presented on Table 4.8.
The ETSI TS 102 637-1 [80] introduces functional requirements for the BSA, also including requirements for communication scenarios and messages contents. Although not providing direct requirements for Platooning (since it is not part of the BSA), some of the requirements from the driving assistance - co-operative awareness application are perfectly applicable in the platooning scenario, since the environments share similar characteristics, namely the type of information exchanged and the fact that they do not resort to RSU’s to communicate. This application includes the Emergency Vehicle Warning, Slow Vehicle Indication, Intersection Collision Warning and Motorcycle Approaching Indication use cases. Below is provided a functional requirements list for this application (extracted from the standard):

- An ITS station shall announce its presence to its vicinity;
- An ITS station shall broadcasts its position, speed and moving direction to its vicinity;
- A vehicle ITS station shall broadcast its basic dynamics and status information to its vicinity;

---

Table 4.8: Platoon Maneuvers Requirements

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Create</th>
<th>Join</th>
<th>Dissolve</th>
<th>Leave</th>
<th>Merge</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirements</strong></td>
<td>V2V</td>
<td>V2V</td>
<td>V2V</td>
<td>V2V</td>
<td>V2V</td>
</tr>
<tr>
<td><strong>Communication Type</strong></td>
<td>Periodic Broadcast over short period</td>
<td>Periodic Broadcast over short period; Event-Triggered</td>
<td>Event-Triggered</td>
<td>Event-triggered</td>
<td>Periodic Broadcast; Event-Triggered</td>
</tr>
<tr>
<td><strong>Messaging Type</strong></td>
<td>V2V</td>
<td>V2V</td>
<td>V2V</td>
<td>V2V</td>
<td>V2V</td>
</tr>
<tr>
<td><strong>Message Period</strong></td>
<td>Every second</td>
<td>Every second</td>
<td>-</td>
<td>-</td>
<td>Every ten seconds</td>
</tr>
<tr>
<td><strong>Direction</strong></td>
<td>One-Way; Point to Multi-Point</td>
<td>Two-Way; Point to Point; Multi-Point</td>
<td>Two-Way; Point to Point; Multi-Point</td>
<td>Two-Way; Point to Point; Multi-Point</td>
<td>One-Way; Two-Way; Point to Point; Multi-Point</td>
</tr>
<tr>
<td><strong>Transmission Mode</strong></td>
<td>Broadcast</td>
<td>Broadcast; Unicast; Geocast</td>
<td>Geocast; Unicast</td>
<td>Geocast; Unicast</td>
<td>Broadcast; Unicast; Geocast</td>
</tr>
<tr>
<td><strong>Frequency</strong></td>
<td>1 Hz</td>
<td>10 Hz</td>
<td>10 Hz</td>
<td>10 Hz</td>
<td>0.1 Hz; 10 Hz</td>
</tr>
<tr>
<td><strong>Other Requirements</strong></td>
<td>Driver is required to have a special license.</td>
<td>String size must not exceed its maximum length; No other maneuvers occurring.</td>
<td>Leader may only dissolve the string after all vehicles acknowledge they have moved.</td>
<td>Only one vehicle may leave the platoon at a time.</td>
<td>Size of the two Platoons is less than the maximum allowed.</td>
</tr>
</tbody>
</table>

---

The ETSI TS 102 637-1 [80] introduces functional requirements for the BSA, also including requirements for communication scenarios and messages contents. Although not providing direct requirements for Platooning (since it is not part of the BSA), some of the requirements from the driving assistance - co-operative awareness application are perfectly applicable in the platooning scenario, since the environments share similar characteristics, namely the type of information exchanged and the fact that they do not resort to RSU’s to communicate. This application includes the Emergency Vehicle Warning, Slow Vehicle Indication, Intersection Collision Warning and Motorcycle Approaching Indication use cases. Below is provided a functional requirements list for this application (extracted from the standard):

- An ITS station shall announce its presence to its vicinity;
- An ITS station shall broadcasts its position, speed and moving direction to its vicinity;
- A vehicle ITS station shall broadcast its basic dynamics and status information to its vicinity;
• CAM shall provide the position information with a confidence level that is sufficient for the all use cases of the BSA;

• Vehicle ITS station shall have access to the in vehicle system to obtain the required information for the CAM construction. A receiving ITS station should update the position of the sending ITS station;

• Information included in CAM shall allow receiving ITS station to estimate the relevance of the information and the risk level;

• An ITS station shall be able to modify the sending interval of two consecutive CAMs;

• CAM shall be set with high priority for transmission;

• ITS station shall provide one hop broadcasting functionality for CAM.

The standard defines that driving assistance co-operative awareness should use broadcasted CAMs (transmission process defined in [81]) and resort to Decentralized Environmental Notification Message (DENM)s if the use case requires so, in order to provide more information on the situation. When receiving messages, the ITS station evaluates the relevance of the situation and should provide the information to the driver via Human-Machine Interface (HMI). [82] defines specifications for the Cooperative Awareness basic service, including the definition of the syntax and semantics for CAM messages (including message handling). [83] defines similar specifications for DENM messages.
Chapter 5

Simulation Deployment

This chapter details the process of deploying the Platooning Management Protocol (PMP) in a simulation environment. First, the deployment decisions are discussed, regarding the simulation tools and the application implementation choices. Finally, the simulation scenario is presented and discussed.

5.1 Deployment Decisions

The first step towards deployment is the choice of the simulation tool(s). Among all solutions to simulate VANETs, the most complete and realistic way is through the use of coupled simulators, since they integrate both mobility and network simulation tools in a transparent way to the user. According to [84], iTETRIS, VEINS and VSimRTI are strong solutions and there is no clear winner, since they all cover the required aspects for VANETs simulation.

Despite iTETRIS potential, there are some drawbacks to the use of this framework for developing ITS applications. Since iTETRIS project finished on 2011, there is not available any community or technical support, with exception to a brief and incomplete document with installation and application development guidelines. The lack of documentation, together with the language-agnostic design makes it very difficult to understand how the applications should be build and implemented. Furthermore, the project’s
virtual machine was built to communicate with an old ns-3.7 version, which now contains deprecated IEEE 802.11p code (fundamental for a proper study of any ITS system), according to the CHANGES.html file available with the ns-3.18 version - this release started to implement new IEEE 802.11p code introduced in [85]. Since the iTetris code is not in the ns-3 main branch, the evolution of other models is not included in it, causing issues. This problem is also mentioned in MOTO project [86]. Additionally, some users [87] state that they have issues installing iTETRIS on recent versions of Linux Ubuntu. The use of older operating systems is often not advised, since they usually lack support, have problems with recent hardware and so on.

For the reasons presented before, VEINS and VSimRTI seem to be more reliable solutions in comparison to iTETRIS. VEINS simulator already includes a platooning module denominated as PLEXE: A Platooning Extension for Veins [28]. However, due to the uncertainty about the flexibility and adaptability of the implemented PLEXE protocols in VEINS and the fact that VSimRTI resorts to the Java programming language (which on a personal level eases the development of applications, due to previous background and expertise), the choice falls for the latter. To simulate transportation, including mobility, transit policies, vehicles emissions and even fuel consumption, the choice is SUMO tool, since it is able to support detailed representations of large scale traffic scenarios. To support accurate and realistic simulation of communication for cooperative ITS systems, the choice is ns-3, since it includes all models to reproduce functionalities and protocols for the ITS communications stack.

According to [88], a very suitable wireless technology available today to interconnect vehicles is IEEE 802.11p. It was specifically built for vehicular environments and it is able to support applications which are very strict in terms of requirements - such is the case of platooning. For this reasons, it was the technology chosen to allow communication between vehicles in the simulation scenario.

Although the advised size of the string is of 15 vehicles, as discussed before in section 4.2, two different sizes were defined for the two different platoon strings created in the simulation. Platoons A has a maximum string size of
15 and platoon $B$ has a size of 3. This happens so that it is easier to test the cases where the string is already full: e.g. vehicles joining a fulfilled platoon and merge maneuvers cannot be performed (i.e. Platoon $A$ cannot merge into Platoon $B$).

Platooning solutions tend to resort to constant time gap to determine the distance one vehicle should keep to its preceding vehicle in the platoon. However, the distance was decided to be constant and equal to 2 meters. This choice was based on the work of [29], which states that for homogeneous vehicles case (same acceleration and deceleration capabilities), the minimum gap was set to be 2 meters.

Vehicles running in the simulation can assume three roles: Non-platoon, Leader and Follower. To achieved this in the simulation, each vehicle reads a configuration file, in order to know which role it should assume. If a vehicle is of type Leader, it creates a new platoon and starts broadcasting its existence. If it is of type Follower, it starts searching for platoons in order to join one. Additionally, a given Leader may also assume a Follower role after a Merge operation is finished. If a vehicle leaves the platoon, it becomes a Non-platoon vehicle. When in a platoon, Follower vehicles are fully automatic, always following the instructions coming from the Leader.

The ”manual” driving behavior had to be forced in SUMO simulator, with constant commands to change speed, position and driving lane on the vehicles. The Leader is responsible for all communications within the platoon (centralized solution), and he is the only vehicle in the string that manages information on the platoon configuration and vehicles parameters.

This implementation follows a sufficient and simple approach where the platoon ID is fixed, which simplifies the data propagation. However, it has the drawback of not being able to adapt to a more dynamic environment in the real world, unlike solutions such as [89, 90] which resort to multicast-based communication.

Another important aspect of this PMP is that maneuvers can happen at any point but only one maneuver is allowed at a time. Allowing more than one maneuver would make the management task extremely complex. Additionally, the right lane (lane 0 in sumo) is only used for platooning, meaning that
non-platoon vehicles are not allowed to merge to the platoon lane. When vehicles leave the platoon, they move from lane 0 to the left lane (lane 1). Regarding the vehicle parameters (acceleration and deceleration values, vehicle length, etc.), it was hard to find evidence of any reliable example values on related work that was useful for this simulation. For this reason, the defined parameters were obtained from the SUMO proposal [91].

Finally, and regarding security concerns, a basic and simple security mechanism for messaging exchanging was implemented for testing purposes. Although the [92] standard advises ITS messages (CAMs and DENMs) not to be encrypted, the non-standard application defined messages used in the simulation are all signed and encrypted, since it is considered that the information contained on the payload is sensitive. In this scenario, the vehicles are statically assigned one public key pair and one symmetric key and all public keys are pre-shared between the vehicles. The symmetric cipher algorithm used was Advanced Encryption Standard (AES) with a 128 bits key and for public key scheme it was used a Rivest-Shamir-Adleman (RSA) key with 1024 bits. First, a hash is generated from the message payload, resorting to the Secure Hash Algorithm - 256 (SHA-256) algorithm. The hash is then signed using RSA. The final message consists of the signature and the AES encrypted message payload. This way, a receiver is able to check if the signature is valid, and if so, decrypt the rest of the message.

5.2 Simulation Scenario

This section presents the general steps taken in order to create the simulation scenario. At first, the chosen simulation scenario was the scenario provided in the Tiergarten to get in acquaintance with the VSimRTI framework. But since the map was from a city center, the path was short and did not allow to fully simulate the platooning application. Hence, a new map creation process was started.

The selected simulation area comprises the main highways that connect the Portuguese cities of Braga and Porto, obtained from a OSM map file [93]. The remaining roads (secondary, rural, etc.) were cropped out of the map.
using the osmosis [94] tool (also made available by OSM). The data was then processed by VSimRTI’s scenario-convert tool, which generates a navigation database and the appropriate SUMO files. This tool allows the generation of random routes, but since random routes do not meet the application needs, the routes were defined manually, by providing the tool with two geographic points. The obtained scenario map is illustrated on Figure 5.1 below. Figure 5.1a shows the real map from the scenario and Figure 5.1b the network of roads (highways) used in SUMO. The route used in the simulation is highlighted in red.

Figure 5.1: Scenario Maps
Figure 5.2 below shows the working environment when running the simulations - the combined view of the log in the terminal with the SUMO simulator and the SUMO visualizer in detail.

(a) Terminal log and SUMO environment
(b) SUMO environment

Figure 5.2: Working Environment
Regarding the configuration files, the *vsimrti_config.xml* file was not modified in this scenario. This file provided in the framework contains general configurations for the VSimRTI platform itself. The SUMO configuration files were all generated automatically using the tools described before. The mapping configuration file (*mapping_config.json*) is used to define the number of vehicles of each type running in the simulation. In this case, the mapping is deterministic, which means that the mapped vehicles are the same in each simulation run. In this file, it is possible to define the values for the parameters illustrated below (configuration extracted from the file).

```json
{
    "prototypes": [
        {
            "applications": ["com.uminho.invovcar.platoon.apps.VehiclePlatoonApp"],
            "name": "PLATOON_VEHICLE",
            "accel": 1.1,
            "decel": 4.0,
            "length": 16.50,
            "maxSpeed": 36.11,
            "width": 2.55,
            "minGap": 0,
            "sigma": 0.1,
            "tau": 0.05
        },
        {
            "applications": ["com.uminho.invovcar.platoon.apps.VehiclePlatoonApp"],
            "name": "PLATOON_VEHICLE_2",
            "accel": 1.1,
            "decel": 4.0,
            "length": 16.50,
            "maxSpeed": 36.11,
            "width": 2.55,
            "minGap": 0,
            "sigma": 0.1,
            "tau": 0.05
        }
    ]
}
```
"applications": ["com.uminho.invovcar.platoon.apps.
    VehiclePlatoonApp"],
"name": "PLATOON_VEHICLE_3",
"accel": 1.1,
"decel": 4.0,
"length": 16.50,
"maxSpeed": 36.11,
"width": 2.55,
"minGap": 0,
"sigma": 0.1,
"tau": 0.06
},
"vehicles": [
{
    "startingTime": 0,
    "route": 0,
    "maxNumberVehicles": 4,
    "types": [{"name": "PLATOON_VEHICLE"}]
},
{
    "startingTime": 70,
    "route": 0,
    "maxNumberVehicles": 4,
    "types": [{"name": "PLATOON_VEHICLE_2"}]
},
{
    "startingTime": 115,
    "route": 0,
    "maxNumberVehicles": 1,
    "types": [{"name": "PLATOON_VEHICLE_3"}]
},
{
    "startingTime": 0,
    "route": 0,
    "maxNumberVehicles": 0,
    "types": [{"name": "NON_PLATOON"}]
}]
}
Additionally, some helper configuration files were written to define properties for each vehicle. This way, the vehicles read their information from the files, which prevents the rebuild of the entire project code every time a simple change is done to any of the parameters. The platoon vehicle properties files (shown below) possess information about the maneuver timings, parameters to compute distances and speeds, etc.

```plaintext
# Vehicle parameters
is_platoon=1

# Group params
group_name=GROUP_1
leader=veh_0

# Speed adjustments
adjust_weight=2.5
adjust_weight_dec=3.5

# Maneuvers
maneuver_distance=15
max_distance_value=5.0
min_distance_value=-2.0

# Leaving
leaving_v=veh_2
left_color=YELLOW
leaving_time=200

# Dissolve
dissolve=550

# Velocity variation
variation_init=450
variation_end=530
```

The group properties files possess information about the platoon group - namely the speed, minimum gap, maximum size and colors to be used in SUMO’s visualizer. The file containing the platoon A information is presented below:
# Group parameters
platoon_speed=20
platoon_distance=2
string_size=15

# Colors
leader_color=RED
slave_color=BLUE
maneuver_color=GREEN
left_color=YELLOW

Additionally, a non-platoon configuration file was created, which contains only one parameter that states that a vehicle with this properties does not have platoon capabilities:

is_platoon=0

To perform an evaluation on all the maneuvers and to obtain results from their behavior, the following set of events was defined to occur during the simulation:

1. Leader A (Vehicle_0) creates a platoon and starts broadcasting its existence.

2. Followers A1, A2 and A3 (Vehicle_1, Vehicle_2 and Vehicle_3) join Platoon A.

3. Leader B (Vehicle_4) creates a platoon and starts broadcasting its existence.

4. Followers B1 and B2 (Vehicle_5 and Vehicle_6) join Platoon B.

5. Vehicle_7 attempts to join Platoon B but the request is rejected (platoon is already full).

6. Followers A2 and B1 (Vehicle_2 and Vehicle_5) leave their respective platoons.


8. Platoons B merges into Platoon A.


The course of events is now described in detail, accompanied by screenshots from the actual simulation and some example extracts of the application code. Due to space constrains, some breaklines were forced on the code samples. Lines starting with `.method()` should be understood as the continuation of the previous line. Table 5.1 presents the colors for each state of the vehicles, to ease the understanding of the application and vehicles behavior.

<table>
<thead>
<tr>
<th>State</th>
<th>Platoon A</th>
<th>Platoon B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leader</td>
<td>Red</td>
<td>Pink</td>
</tr>
<tr>
<td>Follower</td>
<td>Blue</td>
<td>Gray</td>
</tr>
<tr>
<td>Waiting</td>
<td>Orange</td>
<td>Orange</td>
</tr>
<tr>
<td>During Maneuver</td>
<td>Green</td>
<td>Magenta</td>
</tr>
<tr>
<td>Non-Platoon</td>
<td>Yellow</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Vehicle colors in SUMO

The simulation begins with *Vehicle_0* creating *Platoon A* and broadcasting its existence (Figure 5.3).

Figure 5.3: *Leader Vehicle_0* creates *platoon A*
To broadcast the information on the platoon, the Leader uses Geocast messages. The code used to send a broadcast message is listed below. First, the Leader calculates the destination addresses within a given radius range. Then, the destination addresses are wrapped in a container created via createGeocastDestinationAddressAdHoc() and creates the routing for that message, giving as argument the destination address and the default source address. Finally, the message goes through the process of signing and ciphering (described before in this document before) being sent using the SendV2XMessage() method provided by the operating system.

```java
public class SendMessageGeocast implements SendMessage {
    private GeocastDestinationAddress gda;
    private DestinationAddressContainer dac;
    private MessageRouting routing;
    private OperatingSystem operatingSystem;
    private GeoCircle dest;

    public SendMessageGeocast(OperatingSystem operatingSystem, double radius) {
        this.operatingSystem = operatingSystem;
        dest = new GeoCircle(operatingSystem.getPosition(), radius);
        this.gda = GeocastDestinationAddress.createBroadcast(dest);
        this.dac = DestinationAddressContainer.createGeocastDestinationAddressAdHoc(gda, AdHocChannel.CCH);
        this.routing = new MessageRouting(dac, operatingSystem.generateSourceAddressContainer());
    }

    public void sendMessage(PlatoonMessage message) {
        GroupKeys groupKeys = GroupKeys.getInstance();
        SealedObject sealedObject = new SealedObject(message, groupKeys.getKey("veh_0").getAesEncrypt());
        PrivateKey privateKey = groupKeys.getKey(operatingSystem.getId()).getRsaKeys().getPrivate();
        Signature signature = Signature.getInstance("SHA256withRSA");
        // Sign and cipher the message
    }
```
SignedObject signedObject = new SignedObject(sealedObject, privateKey, signature);
EncryptedMessage encryptedMessage = new EncryptedMessage(message.getRouting(), message.getId()+"", sealedObject, signedObject);
operatingSystem.sendV2XMessage(encryptedMessage);
}
...

When a given vehicle finds a suitable platoon, it is able to perform a request in order to join it (Figure 5.4). The first vehicle to perform a Join operation is Vehicle_1. First, it sends a Join Request to the Leader and awaits for a response. Upon receiving the request, the Leader Vehicle_0 computes the performance value and the position Vehicle_1 should assume, sending this information back to the requester in a Join Acknowledge message (if the maneuver is possible). Since it is the first vehicle attempting to join, Vehicle_1 moves to the rear of the platoon and informs that he has reached the correct position and that the maneuver is finished.

![Figure 5.4: Vehicle_1 sends a Join Request towards Vehicle_0](image)

61
The method responsible for sending a *Join Request* - `joinGroup()` from `FollowerHandler.java` - is detailed below.

```java
public void joinGroup(PlatoonMessage platoonMessage, byte[] address)
    setPlatoonMSG(platoonMessage);
    setAddress(address);
    setNextEvent(1);
    PlatoonJoin platoonJoinRequest = new PlatoonJoinRequest (getPlatoonGroup().getGroupName(),
        getOperatingSystem().getId(),
        getOperatingSystem().getVehicleParameters().getMaxAcceleration(),
        getOperatingSystem().getVehicleParameters().getMaxDeceleration(),
        getOperatingSystem().getVehicleParameters().getMaxSpeed(),
        getOperatingSystem().getInitialVehicleType().getLength());
    sendUnicastMessage(platoonJoinRequest,
        getPlatoonMSG().getRouting().getSourceAddressContainer().getSourceAddress().getIPv4Address().getAddress());
    getOperatingSystem().applyVehicleParametersChange(
        getOperatingSystem().requestVehicleParametersUpdate().changeColor(
            StaticFuncions.getColor(getOperatingSystem(), "ORANGE")));
}
```

First, *Vehicle_1* saves information about the *Leader* address and then proceeds to send a unicast *Join Request* containing information about its own parameters. While awaiting for a response, the vehicle assumes the orange color.

The *Leader* method for computing the performance of the vehicle is actually random - since vehicles on this simulation are homogeneous, the performance value result would always have the same outcome. Using random values, the joining position of the request vehicles will have different outcomes, resulting in two different types of joins: *Rear* and *Side*.

The next vehicle to perform a *Join* operation is *Vehicle_2*. In this particular
case, and after receiving a *Join Request*, the *Leader* computed a performance value greater than *Vehicle_1* (that was already part of the string), which resulted in a situation where the joining vehicle should move to the front of the followers string (*side* join). Thus, in addition to acknowledge the *Join Request*, it also informs *Vehicle_1* to adjust its gap, so that the joining vehicle can join in the correct order.

When the adjusting vehicle reaches the correct position, it sends a *Acknowledge Adjust Gap* message to inform *Vehicle_0*. Similarly, the joining *Vehicle_2* vehicle informs he reached the correct joining position with a *Distance Achieved* message. *Vehicle_0* sends a *Start Maneuver* message to *Vehicle_2* only after he has received the message from both vehicles. When the maneuver is completed, *Vehicle_2* sends to the *Leader* a *Maneuver Completed* message. Upon receiving it, *Vehicle_0* notifies the vehicles that the maneuver is completed with a *Platoon Updated* message. The order of events is illustrated in Figures 5.5 to 5.8.

Figure 5.5: *Vehicle_0* acknowledges the *Join Request* from *Vehicle_2* and sends a *Adjust Gap* message to *Vehicle_1*
Figure 5.6: Vehicle_1 informs the Leader that has reached the correct adjusting position

Figure 5.7: Vehicle_2 reaches the correct joining position and sends a Distance Achieved message towards Vehicle_0
Figure 5.8: Leader Vehicle_0 sends a Start Maneuver message to Vehicle_2. When the Vehicle_2 informs that the maneuver is completed, Vehicle_0 sends Platoon Updated messages to the string, and the platoon reaches a stable state - String is now composed of [Vehicle_0, Vehicle_2, Vehicle_1]

At this point of the simulation - while the Vehicle_2 joining maneuver was still unfinished - Leader Vehicle_0 received another Join Request from Vehicle_3. Even though the maneuver was still occurring, the Leader is able to process the request and check if the maneuver can be performed later when the current one is finished. Then, it proceeds to inform the requester about its decision with a Platoon Reject.

For that reason, a boolean was defined in the Platoon Reject messages, so that the requesting vehicle is able to understand if he can join the platoon later or not. A vehicle gets a negative response if the number of vehicles in the string plus the joining vehicle and the number of awaiting vehicles equals the maximum size a string can have.

The response construction is shown on the following code extract:

```java
Boolean MAX_SIZE =
    (getVehicleHashMap().size() + WaitList.size() + 1) >=
    Integer.parseInt(propertiesRead.getProperty(
        getPlatoonGroup(), getGroupName(), Vars.STRING_SIZE));
if (MAX_SIZE) {
```
sendUnicastMessage(new PlatoonJoinReject(getPlatoonGroup().getGroupName(), id, true),
    getPlatoonMSG().getRouting().getSourceAddressContainer().getSourceAddress().getIPv4Address().getAddress());
}
else
{
    WaitListEntry waitListEntry = new WaitListEntry(platoonJoinRequest, getPlatoonMSG());
    WaitList.add(waitListEntry);
    sendUnicastMessage(new PlatoonJoinReject(getPlatoonGroup().getGroupName(), id, false),
    getPlatoonMSG().getRouting().getSourceAddressContainer().getSourceAddress().getIPv4Address().getAddress());
}

In a positive response (such as this case where Vehicle_3 is trying to join),
the vehicle waits until further instructions from the Leader (as Figure 5.9 illustrates). Otherwise, if the join maneuver is not possible, the vehicle simply leaves and continues its trip.

Since there are cases where multiple requests can appear during a maneuver,
the Leader keeps a list of requesting vehicles in a wait list. The Leader responds to those requests in order when a maneuver is finished. In this particular case, Vehicle_3 gets its response when the Join maneuver from Vehicle_2 is finished, and joins the platoon at the rear of the string.

Meanwhile, Platoon B is already composed of Leader Vehicle_4 and Vehicle_5. The Vehicle_5 Join operation was not scrutinized since it is similar to the Join maneuver of Vehicle_1 in Platoon A. At this point, Leader Vehicle_4 receives two Join Request messages from Vehicle_6 and Vehicle_7. The request from Vehicle_6 is acknowledged, while the request from Vehicle_7 is immediately rejected, following the procedure described before. This happens because it is defined that this platoon has a very small maximum string size (only for testing purposes).

After receiving a negative response (through a Join Reject message), Vehicle_7 changes to the non-platoon lane and continues its trip, following the
predefined route. This moment is captured and shown in Figure 5.10.

It is possible to observe in the figure that Vehicle_5 already adjusted its gap, and that Vehicle_6 is moving to the correct position in order to change lane. When completing the maneuver, it sends a Completed Maneuver message to Leader Vehicle_4, which later informs its followers that the maneuver is completed (with a Platoon Updated message).

Platoon B stabilizes as shown in Figure 5.11.
Figure 5.10: Leader Vehicle_4 acknowledges the Join Request from Vehicle_6 and then proceeds to reject the request from Vehicle_7 (maximum string size is achieved).

Figure 5.11: Platoon B stabilized after the Join maneuvers from Vehicle_5 and Vehicle_6 - String is now composed of [Vehicle_4, Vehicle_6, Vehicle_5]
The next maneuvers tested in the simulation were the *Leave* maneuvers. At 200 seconds of simulation time, both *Vehicle_2* and *Vehicle_5* perform a *Leave* operation in *Platoon A* and *Platoon B*, respectively. In this particular case, the *Vehicle_5* *Leave* maneuver turns out to be simpler than the *Vehicle_2* maneuver, since the vehicle is at the rear of the platoon (which implies that no other vehicles need to adjust gaps in order for him to leave).

On other hand, *Vehicle_2* requires *Vehicle_1* and *Vehicle_3* to open up gaps in order for him to leave *Platoon A* (so the maneuver can be accomplished in a safe way). *Vehicle_2* is only allowed to start the maneuver when all the adjusting vehicles have confirmed to reach the required position. When the leaving vehicles receive the *Start Maneuver* message, they change lanes and leave the group, acknowledging that the maneuver is finished with a *Maneuver Completed* message, sent to the *Leader* of the group. The *Leader* then proceeds to inform its followers that the maneuver is over. The maneuvers are illustrated in Figures 5.12 and 5.13.

![Figure 5.12: Leave Maneuvers I](image)

(a) *Vehicle_2* leaving *Platoon A*  
(b) *Vehicle_5* leaving *Platoon B*
When the Leave maneuvers are finished, another Join maneuver is performed: Vehicle_8 joins Platoon B. The vehicles joins at the rear of the platoon and the procedure is similar to the ones described before. The moment the vehicle joins is captured in Figure 5.14.

At this point, the platoons get in a stable state, as Figure 5.15 shows.

The most complex maneuver to be performed in the Platooning application is the Merge maneuver. This operation requires a lot of cooperation between the two leaders of the merging platoons. The first step of the Merge operation is the discovery of surrounding platoons. To accomplish so, Leaders send a Merge Request broadcast message every ten seconds, surveying other possible Leaders on the road for merge opportunities.

The process of sending the message is extracted from the code and presented next. Leaders set up an event every ten seconds that calls the sendMergeBroadcast() method, which is responsible for sending a Merge Request geocast message. This process is interrupted if the platoon is already performing a Merge maneuver.

```java
public class LeaderHandler extends PlatoonHandler {
    public void mergeEventSetUp()
```
Figure 5.14: Vehicle_8 joins Platoon B at the rear

(a) Platoon A stabilized - [Vehicle_0, Vehicle_1, Vehicle_3]  
(b) Platoon B stabilized - [Vehicle_4, Vehicle_6, Vehicle_8]

Figure 5.15: Stabilized Platoons

```java
    mergeBroadcastEvent = new Event(getOperatingSystem().getSimulationTime() + TIME.SECOND*10, this.getApp());
```
getOperatingSystem().getEventManager().addEvent(
    mergeBroadcastEvent);
sendMergeBroadcast();
}

private void sendMergeBroadcast()
{
    if (!merging && getPlatoonGroup() != null && getPlatoonGroup().
        getGroupName() != null && getOperatingSystem().getId() != null)
    {
        PlatoonMergeRequest message = new
            PlatoonMergeRequest(getPlatoonGroup().getGroupName(),
            getOperatingSystem().getId(),
            this.vehicleHashMap.size() +1,
            getOperatingSystem().getVehicleInfo().
                getDistanceDriven());
        sendMessage(message);
    }
}
...

Upon receiving a Merge Request message, Leaders verify if the maneuver can be performed. The maneuver is only possible if no other maneuver is occurring, the size of both strings is acceptable and if the requesting Leader is following behind on the road. If a given Leader is interested in merging and the message is received from a platoon following ahead on the road, the request is rejected but a new Merge Request towards the first requesting vehicle is immediately sent. The beginning of the maneuver is illustrated on Figure 5.16.

In this case, Vehicle_0 (Leader A) receives a Merge Request from Vehicle_4 (Leader B) which is acknowledged with a Merge Request Acknowledgment. The other way around would result in a Merge Reject, since Platoon B is following in the rear and the size is not acceptable - maximum string size in Platoon B is equal to three. After receiving the acknowledgment from the
Figure 5.16: Vehicle\_4 (Leader B) sends a Merge Request geocast message front leader, Vehicle\_4 sends a unicast Merge Info message to Vehicle\_0 with information about its string - the vehicle hashmap with information about every vehicle composing Platoon B. Vehicle\_0 adds the new vehicles information to its own hashmap, and computes the distance that Vehicle\_4 should now assume (and, consequently, for the Platoon B followers), informing it with a Adjust Gap message. Vehicle\_4 starts to adjust its velocity in order to merge into the correct position, as Figure 5.17 shows.
When achieving the correct position, every vehicle from the former *platoon* is informed with a *New Leader* message that they should now follow a new vehicle - in this case, *Vehicle_0*. When the maneuver is completed, the former *Leader* informs the new *Leader* with an *Adjust Gap Acknowledge* message and becomes a *Follower* himself. The final state of the maneuver is illustrated in Figure 5.18.
After the Merge is finished, another Leave maneuver happens in the simulation, resulting from a Leave Request sent by Vehicle.8 to the new Leader of the whole platoon - Vehicle.0. Since Vehicle.8 is the last vehicle of the string, no vehicles have the need to adjust its gap - the operation is similar to the Vehicle.5 Leave maneuver. This process is illustrated in Figure 5.19.
Finally, the *platoon* reaches the last stable state (Figure 5.20) before the *Dissolve* operation. Before starting dismembering the *platoon*, *Leader_0* goes through a phase of velocity fluctuation, to test the *Followers* ability to adjust their velocities based on the messages received from the *Leader* every 100 ms. This part of the simulation is discussed later in chapter 6.
The **Dissolve** maneuver is triggered by the **Leader Vehicle_0** when the simulation reaches the *dissolve time*, given as argument from the configuration files. When reaching *dissolve time*, the **Leader** sends a unicast **Platoon Dissolve Request** to each vehicle of the string. Since the **Leader** is only allowed to leave/dissolve the *platoon* when every single vehicle has acknowledged the request, it will continue this process until it is able to do so. The code used by the application in order to perform the dissolving process of the *platoon* is presented below.

```java
public class LeaderHandler extends PlatoonHandler {
    ...

    public PlatoonHandler handleEvent(Event event) {
    ...
        int dissolveTime =
            Integer.parseInt(propertiesRead.getProperty(
                getOperatingSystem().getInitialVehicleType().getName(),
                Vars.DISSOLVE));
        if (time > dissolveTime) {
```

Figure 5.20: Stabilized *Platoon A* - [Vehicle_0, Vehicle_1, Vehicle_3, Vehicle_4, Vehicle_6]
getOperatingSystem().applyVehicleParametersChange(
    getOperatingSystem().requestVehicleParametersUpdate() 
    .changeColor(StaticFunctions.getColor(
        getOperatingSystem(), propertiesRead.getProperty(
            getPlatoonGroup().getGroupName(), 
            Vars.MANEUVER_COLOR))));
ArrayList<Vehicle> vehicleArrayList = new 
    ArrayList<>(vehicleHashMap.values());
for (int i=0; i<vehicleArrayList.size(); i++)
{
    Vehicle vehicle = vehicleArrayList.get(i);
    PlatoonDissolveReq platoonDissolveReq = new 
        PlatoonDissolveReq(getPlatoonGroup().getGroupName(), 
            vehicle.getId());
    sendUnicastMessage(platoonDissolveReq, 
        getPlatoonMSG().getRouting() 
        .getSourceAddressContainer().getSourceAddress() 
        .getIPv4Address().getAddress());
}

if(AllLeft())
{
    getOperatingSystem().applyVehicleParametersChange(
        getOperatingSystem() 
        .requestVehicleParametersUpdate().changeColor(
            StaticFuncions.getColor(getOperatingSystem()), 
            propertiesRead.getProperty( getPlatoonGroup() 
                .getGroupName(), Vars.LEFT_COLOR))));
    SecureRandom secureRandom = 
        SecureRandom.getInstance("SHA1PRNG");
    double speed = secureRandom.nextInt((int)
        (vehicleParameters.getMaxSpeed() - 1)) + 1;
    speed = Math.abs(speed);
    getOperatingSystem().changeSpeedWithInterval(speed, 
        StaticFuncions.getNextEventTime(100, 
            getOperatingSystem()));
    return new LeftHandler(getOperatingSystem(), 
        getLogger(), this.app);
}
When the process is complete and all vehicles have left the platoon, the simulation reaches its final state before shutting down, as it is possible to see in Figure 5.21.

Figure 5.21: Dissolved

5.3 Messages

This section introduces the description of the parameters of the application-defined messages and the situation in which they are used. First, the common messages are presented, containing general messages used within the group and that are used in more than one maneuver. The messages presented later are grouped by the specific type of maneuver they belong to.
5.3.1 Common

Group

The Group messages are related to the process of exchanging information of the platoon.

Platoon is Group The Platoon is Group message is used as a trick in order to force a given vehicle to join a specific platoon. This happens so that it is easier to control the simulation behavior (e.g. Vehicle_1 should search for Platoon A and not Platoon B). When a vehicle starts the application, it starts sending Platoon is Group messages until it finds the correct Platoon so that he can perform a Join maneuver.

Platoon Group This periodic broadcast message is used by the Leader to disseminate information on the group’s name, creator, route, drive direction, speed, leader position and lane. It is also used as a response to the Platoon is Group requests.

Adjust Gap

The Adjust Gap messages are related to the use cases where vehicles should open up gaps to the preceding vehicle so maneuvers can be completed. These message are extensions of the Platoon Change Distance class.

Platoon Adjust Gap These unicast messages are used by the Leader to inform its Followers on which is the distance (given as argument) that they should keep towards him. They are useful during the maneuvers that require a given set of vehicles to open up or close gaps (e.g. during a Join maneuver, vehicles need to open up space so that the joining vehicle may merge into the platoon).

Platoon Adjust Gap Acknowledgment These unicast messages are used by Followers as a response to the Platoon Adjust Gap message, informing the Leader that they reach the correct position.

Platoon Updated The messages are used by the Leader to inform the Followers that the platoon is stabilized, i.e. the maneuver is finished.
5.3.2 Join

The list of messages below is used to accomplish the Join maneuver and they are extensions of the class of Platoon Join messages.

**Platoon Join Request** These periodic broadcast messages are sent from a vehicle that wants to join a platoon. They contain information about the vehicle parameters (maximum acceleration, deceleration, speed and size) which helps a given Leader to make a decision on whether to accept or reject the request.

**Platoon Join Acknowledge – Platoon Join Reject** These unicast messages are a response to the Platoon Join Request messages. The Leader is able to accept a request (using a Platoon Join Acknowledge) or to reject it (using a Platoon Join Reject). The Leader may reject a request to join if the size of the string reached its maximum, the platoon is performing another maneuver, or if the received vehicles parameters are not in conformance with the platoon characteristics (e.g. vehicle is not able to reach the platoon traveling speed). However, this situation never occurs in the simulation, since vehicles are homogeneous. The Platoon Join Acknowledge contains information about the distance that the joining vehicle should establish from the leader and the lane the platoon is traveling. The Platoon Join Reject contains information on whether the reject is based on the string size or if a maneuver is occurring. In the last case, it means that the maneuver can be performed later.

**Platoon Join Distance Achieved** This unicast message is used by the joining vehicle to inform the Leader that it has arrived the correct position in order to merge to the correct lane. It possesses no special arguments.

**Platoon Join Start Maneuver** The Leader sends this unicast message to the joining vehicle, informing that the maneuver can be accomplished.

**Platoon Join Maneuver Completed** This unicast message is sent from
the joining vehicle to the Leader, informing that the maneuver has terminated successfully. It contains no arguments.

5.3.3 Leave

The list of messages below is used to accomplish the Leave maneuver and they are extensions of the class of Platoon Leave messages.

**Platoon Leave Request** These unicast messages are sent from a vehicle that wants to leave a platoon to the Leader. They do not contain any information.

**Platoon Leave Reject** This unicast message is sent by the Leader if the leave maneuver is not possible to perform at this moment. It contains no payload.

**Platoon Leave Start Maneuver** The Leader sends this unicast message to the leaving vehicle if the maneuver is possible. If the leaving vehicle is at the rear of the platoon, the Leader sends this message immediately. Otherwise, it is sent after the vehicles adjust their gaps, so that the leaving vehicle can perform the maneuver safely. These messages do not contain information.

**Platoon Leave Maneuver Complete** This unicast message is sent from the leaving vehicle, informing the Leader that the maneuver is completed. Similarly to the Platoon Leave Start Maneuver messages, it does not contain payload.

5.3.4 Merge

**Platoon Merge Request** This periodic broadcast message is sent from the platoon Leaders, surveying possible merge opportunities. In this message is included information on the platoon size and the position of the leader on the road. These messages provide enough information to help another Leader to accept or reject the request.
**Platoon Merge Reject – Acknowledge** These messages are used by the front *Leader* to inform the requesting *Leader* if a merge is possible or not. When it’s possible, the front *Leader* returns a *Platoon Merge Acknowledge* to the following *Leader*, with no special information. The *Leader* may reject a request (resorting to a *Platoon Merge Reject* message) if another maneuver is occurring, or if the size is not acceptable. Additionally, a *Leader* may reject a request if it comes from a *Leader* that is following more ahead on the road. On this particular case, it delivers another *Platoon Merge Request* to the *Leader* that first sent the request. This happens, so that the maneuver is easier to accomplish - the rear *Platoon* moves to the rear of the front *Platoon*.

**Platoon Merge Info** When receiving an affirmative response from the front *Leader*, the rear *Leader* sends a unicast *Platoon Merge Info* message containing the hashmap of vehicles which are part of its *Platoon*, containing information on all vehicles of the string.

**Platoon Merge Adjust Gap** After receiving a *Platoon Merge Info* message and merging the received hashmap with its own, the front *Leader* computes the distance that the rear *Leader* should now assume, and sends it as an argument via a *Platoon Merge Adjust Gap* message.

**Platoon Merge Adjust Gap Acknowledge** When the rear *Leader* arrives at the correct distance to the front *Platoon*, it sends a *Platoon Merge Adjust Gap Acknowledge* message informing that he has reached the position.

**Platoon Merge New Leader** The *Platoon Merge New Leader* messages are used by the *Leader* to inform its followers that they now should follow a new *Leader* and that the *Merge* maneuver is finished. It contains information on the distance to the front *Leader* and on the new *Platoon* Group.
5.4 Implementation Challenges

When first starting the implementation of the PMP, it was intended to use the standard ITS messages to communicate in the *platoons*. However, it was more efficient to build self-defined messages that the vehicles could easily process and understand. Initially, the PMP was defined to use *geocast* messages every time the vehicles needed to communicate. Soon this proved to be inefficient and to make the simulation slow down. Thus, only the *Platoon Group* messages stayed as *geocast*, while the remaining messages used the *Topocast* method, which proved to be more efficient. The *Topocast* method is used to communicate with an individual destination node in the direct communication range of the sender.

The default *time-step* used in SUMO lead vehicles to keep a large distance to the vehicle in front, even when the application was forcing vehicles to approach and it was not possible to send messages every 100 ms. The distance to the front leader is calculated by SUMO using $\tau$ (driver reaction time) and *minimum gap* values. For instance, if the $\tau$ value is equal to 0 and vehicle speed is 15 m/s, the minimum calculated distance to the front vehicle is 15 m (not desirable in the *platooning* application). As SUMO does not allow the *time-step* value to be greater than the $\tau$ value, the *time-step* was defined to be equal to 50 ms instead of the predefined value of 1000ms. However, this solution had as drawback the decreasing of the simulation performance. Additionally, the constant use of the *in-vehicle* sensor and the commands to change lane and adjust speed caused the simulation to degrade its performance. The constant commands could not be moved, because vehicles are not "smart" in SUMO - one must force them to move to a given place with constant indications. To overcome the sensor problem, the distance information is sometimes taken by GPS while the cars are maneuvering and by the sensors when stable in the *platoon*. Also, since SUMO and VSimRTI do not provide any special function to compare which vehicle follows in front on the road, this information is actually based on the distance driven on the defined route. However, this solution only works on this particular scenario, since the route is the same to all vehicles.
The PMP suffered some alterations from its original version when the implementation started. When testing the maneuvers in the simulation environment, some steps were modified to strengthen the solution. In the Join maneuver, the vehicle is now only allowed to change to the correct lane when every adjusting vehicles has finished its operation. At first, the vehicle would simply try to merge to the lane as soon as it reached the position, which is not advised or secure. Also, when the maneuver is finished, the joining vehicle he notifies the Leader. This allows the Leader to know when to alert the remaining Followers that the maneuver is finished and that they can now assume the correct position. Previously, vehicles assumed that a given operation was finished when they reached the exact computed position. Now they are able to finish an operation when they reach an acceptable distance to that computed point. This makes the simulation more realistic (e.g. in the real world, a vehicle will merge to the correct lane as soon as he is able to when in manual mode) and the maneuver durations become faster as well. Regarding the Merge maneuver, the Rear Leader now only informs its followers that they have a new Leader when he reaches the correct position in the merged Platoon. In the first solution, this was performed as soon as he received a Merge Acknowledgment. This way, the maneuver is safer and the adjusting gap operation of the rear platoon is smoother - rear Followers will still be "listening" to the rear platoon Leader until the maneuver is finished. In the first description of the Dissolve maneuver, the Followers did not inform the Leader that they had in fact left, and the Leader would simply leave the platoon. This could potentially lead to severe problems - for example, if a given Follower had not received the Dissolve Request, it would be still waiting for instruction from the Leader (which would already left the platoon). Finally, the Leave maneuver was also modified so that vehicles following behind the leaving vehicle would open up a gap. In the first description, the vehicle would simply change lanes. In the real world, performing the Leave maneuver in such a tight space could lead to accidents. Thus, the adjusting operation was introduced so that the maneuver can be performed safely.
Chapter 6

Results and Analysis

In general, the behavior of the vehicles in ITS applications simulation tends to be extremely dynamic: typically, the mobility simulator runs with hundreds of vehicles equipped with applications at a given penetration rate and they follow computer generated routes. However, the simulation deployment on this work was proposed to be slightly more static, in the sense that the maneuvers (and respective maneuver start timings), routes and even the vehicles running the application are predefined. Although not being the most desired situation, this happens due to the high difficulty level of controlling and evaluating the vehicles and application behavior during the complex maneuvers that result from the PMP. Nevertheless, independent simulation runs will always generate different outputs, since the Join maneuvers are dynamic (due to the performance ordering), and consequently the Leave maneuvers and adjusting operations will also be different in each run.

Although the VSimRTI framework provides some result logs, the application was built to generate more specific results regarding the exchanged messages and the behavior of each vehicle. In particular, the PMP generates logs regarding the exchanged messages (group and maneuver related messages), vehicles distance (towards their correct position), speed values and the lane capacity. The obtained results are discussed in the following sections.
6.1 Lane Capacity

The first immediate results that can be obtained from the use of Platooning is increased lane capacity. According to [78], platoons are able to increase traffic capacity due to its capability to use tight spaces between vehicles (and between different platoons). The presented formula to compute the capacity is:

\[ \phi = v \times \frac{n}{ns + (n - 1)d + D} \]  
vehicles/lane/min \quad (6.1)

where \( v \) is the steady state speed (meters/min), \( d \) the intra-platoon spacing (meters), \( D \) the intra-platoon spacing (meters), \( s \) the vehicle length (meters) and \( n \) the number of cars composing the string.

In the PMP definition, where \( v \) is equal to 1500 m/min (25 meters/second), \( d \) is 2 meters, \( D \) is 30 meters (although not defined initially and not taking into account on the application itself, 30 meters seems a reasonable value), \( s \) is 15 meters and \( n \) is 15 vehicles, the capacity result is \( \phi = 71.88 \) vehicles/lane/min. Assuming that the typical lane capacity value is \( \phi = 35 \) vehicles/lane/min [78], the PMP is able to theoretically more than double the capacity of the road.

To compare this analytical value to the actual application (which has dynamic parameters, namely the speed and number of vehicles), the capacity value was computed and logged by each leader during the runtime every second. The results were then processed in Microsoft Excel.

The obtained results that are presented on Figure 6.1 were taken from a single simulation - since the number of vehicles in the platoons and velocities are the same in each simulation run (as discussed before, the maneuvers are static), the outcome of the capacity value will always be similar.

As expected, the capacity value is typically lower than the theoretical value, since platoons have in fact different group speeds and the number of vehicles in the string is not the same as the maximum defined values. Also, these values differ in both platoons, which implies that the capacity values are different between them.

Platoon B tends to have higher capacity values, which is explained by the
fact that the group speed is higher than in *Platoon A*, even though *Platoon B* has less vehicles in the string. In general, the capacity value increases when vehicles join the string and decrease when they leave. During the speed fluctuation in *Platoon A* (around 450 seconds), it is also possible to verify the impact the speed has on the capacity - capacity increases when speed increases and vice-versa. Furthermore, the computed capacity also decreases abruptly when vehicles start dissolving (around 500 seconds of simulation time).

Table 6.1 presents the mean values for the capacity in the *platoons* during the simulation. Although the application is not able to meet the theoretical values (in this scenario), it is still possible to conclude that it is able to achieve much better results than when *platooning* is not implemented.

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Mean value (vehicles/lane/min)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Platoon A</em></td>
<td>48.388</td>
</tr>
<tr>
<td><em>Platoon B</em></td>
<td>54.166</td>
</tr>
</tbody>
</table>

Table 6.1: Mean capacity values during simulation runtime
6.2 Maneuvers

This section describes the behavior of the Platoons during the application runtime, focusing and discussing on the duration of the maneuvers and what are the possible reasons behind those results. The presented outcomes for the Join and Leave maneuver duration times result from a single simulation. Taking into account mean results of multiple simulations would not make sense, since these maneuvers are dynamic. However, the Merge and Dissolve maneuvers discuss mean results from all simulation runs.

6.2.1 Join Maneuver

This subsection discusses the duration of the Join maneuver. With exception to the Leader vehicles (creators of platoons), every vehicle must perform a Join maneuver in order to become part of a platoon. The duration of these maneuvers, along with their mean values, is presented in Table 6.2. The table is divided in two parts: Vehicle_1, Vehicle_2 and Vehicle_3 maneuvers were performed in Platoon A, while Vehicle_5, Vehicle_6 and Vehicle_8 maneuvers were performed in Platoon B. The waiting time value represents the amount of time a vehicle awaits until it receives a Join Acknowledgment after sending a Join Request.

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Join Type</th>
<th>Maneuver Duration (s)</th>
<th>Waiting Time (s)</th>
<th>Total Duration (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Veh_1</td>
<td>Rear</td>
<td>32.980</td>
<td>–</td>
<td>32.980</td>
</tr>
<tr>
<td>Veh_2</td>
<td>Side</td>
<td>43.001</td>
<td>16.284</td>
<td>59.285</td>
</tr>
<tr>
<td>Veh_3</td>
<td>Rear</td>
<td>35.000</td>
<td>40.181</td>
<td>75.181</td>
</tr>
<tr>
<td>Veh_5</td>
<td>Rear</td>
<td>42.943</td>
<td>–</td>
<td>42.943</td>
</tr>
<tr>
<td>Veh_6</td>
<td>Side</td>
<td>44.996</td>
<td>17.989</td>
<td>62.985</td>
</tr>
<tr>
<td>Veh_8</td>
<td>Rear</td>
<td>37.885</td>
<td>–</td>
<td>37.885</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>39.467</td>
<td>24.818</td>
<td>51.876</td>
</tr>
</tbody>
</table>

Table 6.2: Join maneuver duration
When looking at the Join duration results, the first conclusion one draws is that joining a platoon at the rear is slightly faster than joining a platoon by the side. The rear joins at Platoon B seem to take a bit longer than in Platoon A, which may be explained by the fact that Platoon B is traveling at a greater speed, which makes the maneuver harder for the joining vehicles. The second remark is that despite the speed differences, the Join maneuvers in both platoons have similar durations, with no relevant discrepancies in the values. Even regarding the wait times, the values seem acceptable and similar between them, with exception to the Vehicle_3 join operation. The value seems somewhat high, but is explained by the fact that the Join Request was sent when the Vehicle_2 side maneuver was in a early stage (and side operations by themselves already takes longer than rear operations).

Still regarding the Join maneuver, it is also important to analyze the duration of a negative response to a rejected Join Request. As shown on Table 6.3, the reject situation only happens once during the simulation runtime - when Vehicle_7 attempts to join platoon B. It is possible to observe that between the request and response, only 1 ms has elapsed. This happens because the Leader is able to respond immediately if a given requester is able to perform the maneuver, even if another maneuver is already occurring. At this point of the simulation, Platoon B was already at its maximum size.

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>Duration (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle_7</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Table 6.3: Rejected Join Request duration

6.2.2 Leave Maneuver

Similarly to the Join maneuver, the Leave maneuver may also assume two types: side and rear. In side leaves, the vehicles that follow behind the requesting vehicles are required to open up gaps, so that the maneuver can be accomplished in a safe way. In rear leaves, and since the vehicle requesting
to perform the maneuver is the last one of the string, other vehicles do not need to adjust their gaps - the leaving vehicle simply changes the lane and continues its trip. For this reasons, as it is possible to see in Table 6.4, there is a huge difference in the duration of the maneuver, depending on the type.

<table>
<thead>
<tr>
<th>Leave</th>
<th>Leave Type</th>
<th>Duration (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle_2</td>
<td>Side</td>
<td>27.000</td>
</tr>
<tr>
<td>Vehicle_5</td>
<td>Rear</td>
<td>0.999</td>
</tr>
<tr>
<td>Vehicle_8</td>
<td>Rear</td>
<td>1.000</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>9.666</td>
</tr>
</tbody>
</table>

Table 6.4: Leave maneuver duration

6.2.3 Adjusting Gaps

Although the Adjust gap situations are not qualified as maneuvers themselves, they play an important role on the maneuver’s course of actions and duration times. Vehicles may adjust their gap during the Join, Leave and Merge maneuvers. However, the results regarding the Merge maneuver are discussed later in subsection 6.2.4, and thus not included on Table 6.5. The duration of these operations seem to be in a reasonable interval of time. However, there is a discrepancy between the values for the Vehicle_2 during the Join maneuver and Vehicle_6 in a similar situation. The reason for this result is not clear, but may also be related to the fact that the speed of the platoon B is greater than platoon A.

6.2.4 Merge Maneuver

The merge maneuver can be divided in three steps: exchanging the information between the Leaders, the adjusting gap operation and the New Leader information dissemination. Table 6.6 presents the duration of each of these
### Adjusting Gaps

<table>
<thead>
<tr>
<th>Maneuver</th>
<th>Adjusting Vehicles</th>
<th>Duration (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle_2 Join Vehicle_1</td>
<td></td>
<td>6.998</td>
</tr>
<tr>
<td>Vehicle_6 Join Vehicle_5</td>
<td></td>
<td>19.999</td>
</tr>
<tr>
<td>Vehicle_2 Leave Vehicle_1, Vehicle_3</td>
<td></td>
<td>24.996</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>17.331</td>
</tr>
</tbody>
</table>

Table 6.5: Adjusting Gap durations

steps in seconds. The Merge maneuver was accomplished at a rather surprising short time - even faster than the Join maneuvers. The expected results were for the maneuver to take at least the same amount of time that the fastest Join maneuver took - a Merge operation can almost be seen as one Leader joining another platoon that follows in front of him. Still, the adjust gap operation takes most of the time - approximately 99% of the total time - while the maneuver set up and finishing steps are very fast - 8 milliseconds.

### Merge

<table>
<thead>
<tr>
<th>Info Exchange (s)</th>
<th>Adjusting Gap (s)</th>
<th>New Leader Information (s)</th>
<th>Total (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>28.172</td>
<td>0.003</td>
<td>28.180</td>
</tr>
</tbody>
</table>

Table 6.6: Merge maneuver duration

#### 6.2.5 Dissolve

The Dissolve maneuver duration time is estimated from the moment when the Dissolve Request is issued by the Leader until the last Dissolve Acknowledgment is received as a response. As seen on Table 6.7, the maneuver performs quickly - 4.1 milliseconds. This happens because all vehicles simply acknowledge the request and turn into manual driving instantaneously, which makes the result unrealistic - in a real situation, the maneuver would for sure last longer, since the driver would be required to answer to the request manually before the vehicle can stop automatic control. In the ap-
plication simulation, and since the switch is performed automatically, the operation is performed without any delay.

<table>
<thead>
<tr>
<th>Dissolve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration (s)</td>
</tr>
<tr>
<td>0.0041</td>
</tr>
</tbody>
</table>

Table 6.7: Dissolve maneuver duration

6.3 Messages

This next section presents the results regarding the delay (time that elapses from the message creation timestamp until it is received at the destination) of the messages that are sent on the application. The results comprise all types of messages exchanged during the simulation runtime, including both the maneuver messages and the Platoon Group messages sent regularly by the Leaders (every 100 ms). The obtained results were processed by the Analysis Toolpak provided by Microsoft Excel for a confidence level of 95%. One analysis example taken from the first simulation run is presented on Table 6.8.

<table>
<thead>
<tr>
<th>Messages Latency (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Standard Error</td>
</tr>
<tr>
<td>Median</td>
</tr>
<tr>
<td>Mode</td>
</tr>
<tr>
<td>Standard Deviation</td>
</tr>
<tr>
<td>Sample Variance</td>
</tr>
<tr>
<td>Range</td>
</tr>
<tr>
<td>Minimum</td>
</tr>
<tr>
<td>Maximum</td>
</tr>
</tbody>
</table>

Table 6.8: Statistic results from messages latency of the first simulation run
The first important remark to be made is that there a lot more operations to handle messages than the expected. The number of messages transmitted by *Followers* are higher due to the need to force the vehicles to constantly search for their *platoon* (using the *Platoon is Group* messages) until they can perform the *Join* operation. Additionally, the exchange of broadcast messages results in multiple copies of the same message to be received by different vehicles with different timestamps.

Nonetheless, the calculated values present a very satisfactory result. They indicate that the PMP is mostly able to deliver the messages on time, and it is able to conform with the very strict communication requirements of the *platooning* application - the messages are able to be generated every 100 ms and the requirement for the maximum delay allowed (100 ms) can be fulfilled. This also means that the impact of the security mechanisms used to encrypt and sign the messages is almost negligible - vehicles can still receive messages in a valid time interval. This way, the data content can be secured while the communication is not compromised.

Table 6.9 presents the number of messages that exceed the expected latency value. It is possible to see that there are a total of 943 messages that are delivered with a latency time greater than 100 ms. From those 943 messages, 81 of them are delivered with a latency value greater than 125 ms, 11 with more than 150 ms and only one message exceeds a latency value of 175 ms. Despite the results not being perfect, these messages represent a universe of only 0.1218% of the total of exchanged messages. Thus, the results are still considered valid.

<table>
<thead>
<tr>
<th>Latency (ms)</th>
<th>Number of Messages</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 100</td>
<td>943</td>
<td>0.1218</td>
</tr>
<tr>
<td>&gt; 125</td>
<td>81</td>
<td>0.0105</td>
</tr>
<tr>
<td>&gt; 150</td>
<td>11</td>
<td>0.0014</td>
</tr>
<tr>
<td>&gt; 175</td>
<td>1</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Table 6.9: Messages with latency greater than 100 ms

The mean values regarding the delay from each simulation run is presented in Figure 6.2. As it is possible to see, the values are very close to each other,
which shows that the application can smoothly handle the messages within
the expected latency every simulation run.

![Mean Message Delivery Delay Values](image)

Figure 6.2: Mean message delivery delay values for each simulation

### 6.4 Distances

This section presents an example result from one simulation run that ana-
lyzes the distances that the vehicles should achieve during the application
runtime. Vehicles log their *distance to go* value (distance that they should
achieve) every second, in order to conclude the behavior of the vehicles when
they try to reach the correct position. The vehicle logs are represented as
functions (one color for each vehicle) on Figure 6.3. The x axis represents
the simulation time and the y axis the distance to achieve in meters. It is
important to point out that the ”breaks” in the functions happen when the
vehicles switch from the *GPS* distance sensor to the *in-vehicle* distance sensor
(and vice-versa). The results comprise only the *Follower* logs, since *Leader*
vehicles do not keep distances (with exception to *Vehicle_4*, that becomes a
Follower after the Merge maneuver).
The function values start to be recorded when a Join Request is sent. At this point, the vehicle starts immediately to adjust its speed to move to the correct position. In the cases where the Join is immediate (e.g. Vehicle_1), the vehicle starts moving right away, which results in a fast descending curve. However, when a vehicle needs to wait for its request to be fulfilled, the distance to go value stays in a “stable” state until it receives orders to move to the correct position (e.g. Vehicle_2 enters this state around the 30 seconds of simulation time mark, lasting for sensibly 16 seconds).

At this point, if the joining maneuver requires other vehicles to adjust their gap (e.g. Vehicle_1 starts adjusting its gap when Vehicle_2 receives the Join Acknowledge), their distance to go value becomes negative. This means that the vehicle should slow down and open a space in the string. The process of adjusting is finished when the distance to go values reaches zero again - the vehicle as successfully opened a gap. When the joining vehicle finally joins, the adjusting vehicles receive a Platoon Update message from the Leader. This means that the vehicle should move forward (the distance to go value
becomes positive again), and the vehicles should adjust their speed so the distance to go is zero again. At this point, a zero value means that a vehicle is in the right position, at the expected intra-platoon distance. This process occurs every time there is a maneuver being performed in the simulation. When the vehicles are not maneuvering, the distance to go seems to be pretty stable, which means that the vehicles can rapidly adjust their speed to reach the correct position based on the frequent Platoon Group messages sent by the Leader. From 450 to 500 seconds of simulation runtime, where the Leader starts fluctuating the speed (and consequently, the group speed), the adjusting operation is also very smooth - the distance to go values barely suffer alterations.

6.5 Speed

This section presents the results from the speed values that vehicles maintain during the application runtime (vehicles log their speed value every second). These results were also obtained from the same simulation run that was used in the distance analysis.

The speed logs are represented as functions (one color for each vehicle) on Figure 6.4. The x axis represents the simulation time and the y axis the speed in meters/second. The first descending curve from the Leaders happens during the time the Leader is reading the configuration files and creating the Platoon, before assuming the correct group speed. Similarly to the distances chart, the Followers function values start to be recorded when a Join Request is sent. When a given vehicle receives a Join Acknowledge from the Leader, they start immediately adjusting their speed to move to the correct position. This actually results in vehicles decreasing their speed values as they move towards the correct distance, since vehicles travel at their maximum speed until they join a platoon (which have lower group speed values). For example, when Vehicle_1 joins, the speed decreases immediately (at around 15 seconds of simulation time) to match the platoon A group speed of 20 m/s. After joining a platoon, the vehicle’s speed stabilize and match the respec-
Figure 6.4: Vehicle’s speed value during simulation
tive platoon speed. The fact that the platoon speed is constant is a little unrealistic - in the real world, the Leader would not be traveling at exactly the same speed during the whole trip. During the forced speed fluctuations introduced from around 450 seconds until 500 seconds of simulation time, the vehicles can easily and smoothly adjust their speed to meet the Leader speed - announced to the followers through Platoon Group messages. Another important aspect of the speed values that the vehicles assume during the simulation is that they are intimately related to the computed distance to go values. This means that the vehicle must increase its speed when the distance to go value its positive and to slow down when the distance to go is negative. For example, when Vehicle_2 is joining Platoon A, Vehicle_1 starts decreasing its speed until he opens up a gap in the string in order for Vehicle_2 to merge into the lane. When this maneuver is finished, Vehicle_1 increases its speed again to meet the required intra-platoon distance.
Chapter 7

Conclusions and Future Work

Intelligent Transportation Systems (ITS) are systems that aim to assure a more efficient and improved usage of the roads, leading to potential improvements in terms of safety, traffic management and comfort, by controlling traffic operations and drivers behavior. ITS enables the creation of many applications that use the information from vehicles and infrastructures that compose Vehicular Ad Hoc Networks (VANETs) to implement better driving practices and to improve traffic flow. A typical example of one of these applications is the Platooning use case.

In the first chapters, this thesis introduces and discusses the related work and general ITS simulation considerations (along with some important tools). Then, the ITS applications taxonomy is presented, giving emphasis to the Platooning use case. The Platooning application is described in detail and the Platooning Management Protocol (PMP) is introduced, containing the description of the maneuvers and general considerations. Additionally, the PMP requirements are also presented, mostly based on ITS standards defined by the European Telecommunications Standards Institute (ETSI).

The second part of the thesis details the process of deploying the practical PMP implementation and obtaining the results from the simulation runs. The PMP application was implemented using the V2X Simulation Runtime Infrastructure (VSimRTI) framework, coupling Simulation of Urban MObil-ity (SUMO) and Network Simulator 3 (ns-3). The choice of the tools was
not a difficult decision, taking into account that these tools are proven to be very powerful and well-established within the research community.

The deployment chapter describes the process of creating the simulation scenario and associated decisions as well. The Platooning application deployment was not an easy task - from the study on the state of the art, it was possible to conclude that the application is immensely complex and sometimes very subjective. For each particular problem that arises from Platooning, there are usually a lot of different proposed solutions (e.g. some researchers believe that the distance one vehicle should keep to the preceding vehicle must be based on a constant time gap while others propose a constant distance gap). This seems to be an indicator that the Platooning specification is prone to ambiguity. Thus, this work resulted from an effort to trying to meet a "common ground" between the existent proposals. Also, the process of deploying the application required a lot of effort to overcome some lack of "intelligence" the chosen tools present (mainly SUMO simulator). The constant trade-off between a more realistic application and the simulation performance caused some difficulties to evaluate the application behavior.

From the simulation, it is possible to conclude that the PMP is able to work smoothly and efficiently - the duration of the maneuvers are within an acceptable interval, the messages are able to meet the hard communication requirements and the lane capacity is proven to be increased.

The future work could be extended by improving the PMP in order to take into account the existence of abnormal situations, such as obstacles on the road, cut-ins in the middle of the platoon, and so on. Furthermore, the simulation scenario could be improved to allow a full dynamic flow of events, in the sense that the maneuvers are not controlled and platoons and vehicles travel freely in the roads. The models to calibrate the speed and the vehicles parameters may also be improved. Finally, it would be interesting to evaluate the proposed PMP through experimental testing, instead of simulation means. The PMP could be tested using a series of On-Board Units (OBUs), with every device running the application individually. Since manufacturers wouldn’t probably let third parties use the actuators (e.g. brakes, accelera-
tor) on the vehicles for security reasons, the protocol could be adapted to give visual advise to drivers through a Graphical User Interface (GUI) running in a screen installed on the vehicle. For example, the application could advise the driver to open up a gap, to reduce the distance to the front vehicle, etc.
Bibliography


itoring System Based on V2X Communications. In 2014 IEEE International Conference on Communications (ICC), pages 2713–2718. IEEE, jun 2014.


110


111


[80] ETSI. ETSI TS 102 637-1 V1.1.1 Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 1: Functional Requirements, 2010-09.


[84] Christoph Sommer, Jérôme Härri, Fatma Hrizi, Björn Schünemann, and Falko Dressler. Simulation Tools and Techniques for Vehicular Commu-


