POLITECNICO DI TORINO

Master Degree course in Mechatronic Engineering

Master Degree Thesis

Awareness Messages for Vulnerable Road Users

Supervisors
Prof. Claudio Ettore Casetti
Dott. Francesco Raviglione
Dott. Marco Rapelli

Candidate
Alessandro Genovese

December 2023

Acknowledgements

Before starting talking about the contents of my thesis, I would like to thank my advisors Prof. Claudio Ettore Casetti, dott. Francesco Raviglione and dott. Marco Rapelli for giving me the opportunity to join such an interesting and innovative research project and for always providing me help, support and assistance throughout the whole work.
I would also like to thank all the people with whom I have had the opportunity to work throughout these intense five years at Politecnico di Torino, including my friends and all my professors.

Abstract

In recent years, the automotive world has witnessed a significant technological leap, marked by innovations and automation. This evolution encompasses the rise of connected cars and intelligent road users, with a specific focus on communication standards. This thesis aims at exploring the critical domain of Vulnerable Road Users (VRUs), which are traffic participants such as pedestrians and cyclists that lack the protective shell of motorized vehicles and therefore experience a greatest danger if involved in road accidents.

WHO and similar institutions state that every year tens of millions of people are involved in road accidents and many of them are VRUs, for which the consequences are often quite serious due to their vulnerability. The European Telecommunications Standards Institute (ETSI) has therefore responded to the safety challenges faced by VRUs by developing a communication protocol which empowers VRUs to share their main attributes and status information with other road users. The ITS messages disseminated by VRUs are called Vehicle Awareness Messages (VAMs) and they aim at fostering VRUs' active engagement in the traffic ecosystem.

The aforementioned protocol defines the concept of VRU system, as entity made up of at least one VRU and one ITS-Station with a VRU application installed, it accommodates diverse VRU profiles, including pedestrians, cyclists, motorcyclists, e-scooters and even animals posing a potential risk to road safety, it defines the equipment needed by VRUs to be able to exchange information with other road users and it introduces concepts like VRU clustering and VAM Redundancy Mitigation, optimizing communication in areas with a high concentration of VRUs without compromising their awareness and safety.

The study also delves into a detailed study of the VRU Basic Service, exploring its role as the facilities layer entity responsible for transmitting and receiving VAMs, focusing in particular on how the generation and the dissemination of VAMs is handled in the case of both individual VAM and cluster VAMs, analysing in detail fundamental aspects like the triggering conditions for VAM transmissions.

This thesis goes beyond theoretical discussions by implementing the protocol in both simulated and real-world environments. The implementation of the protocol onto the ms-van3t vehicular network simulator and emulator allowed us to assess the protocol's functionality and performance, by running custom made simulations aimed at testing the main aspects of the VAM communication protocol, while On-Board Units (OBUs) were used to conduct field tests assessing its real-world applicability. We also carried out a comparison between the results obtained through the simulations with those obtained on the field, in such a way to analyse how the behaviour of the protocol in a real world scenario changes with respect to the way it behaves in a simulated and controlled environment.

In conclusion, this research aims at contributing to the ongoing dialogue surrounding intelligent transportation systems, emphasizing the need for robust communication standards tailored to the unique challenges faced by Vulnerable Road Users. Through an accurate exploration of the ETSI standard related to VRUs and its application in diverse

scenarios, the thesis strives to pave the way for safer and more inclusive roadways in our rapidly evolving automotive landscape.

Contents

List of Figures

Listings

Acronyms

ITS Intelligent Transportation System

ITS-S ITS-Station

C-ITS Cooperative ITS

VRU Vulnerable Road Users

VAM VRU Awareness Messages

CAM Cooperative Awareness Messages

V2X Vehicle to Everything

C-V2X Cellular-V2X

V2V Vehicle to Vehicle

V2P Vehicle to Pedestrian

V2N Vehicle to Network

ETSI European Telecommunications Standards Institute

3GPP Third Generation Partnership Project

IEEE Institute of Electrical and Electronics Engineers

C2C-CC CAR 2 CAR Communication Consortium

FCC Federal Communications Commission

WHO World Health Organization

BTP Basic Transport Protocol

GN GeoNetworking

IoT Internet of Things

RSU Roadside Unit

DSRC Dedicated Short-Range Communications

OBU On-Board Unit

SAE Society of Automotive Engineers

WAVE Wireless Access in Vehicular Environments

VANET Vehicular Ad Hoc Network

DDP Device Data Provider

PoTi Position and Time management

LDM Local Dynamic Map

PRR Packet Reception Ratio

OCABS Open CA Basic Service

AIM Automotive Integrated Map

OScar Operative System for car

NIC Network Interface Card

Chapter 1

Introduction

When talking about smart mobility and intelligent transportation systems we commonly refer to vehicles or similar road users which are provided with a certain degree of autonomy. These particular vehicles are generally equipped with devices allowing them, at least to some extent, to perceive the traffic scenario surrounding them and, according to the level of automation they are provided with, either to carry out autonomous actions or to warn their users about potential dangers they could face.
The first self-driving vehicle dates back to 1478, when the Italian inventor Leonardo Da Vinci designed and built a wagon which was able to move autonomously, without being pushed or towed.
Starting from the very beginning of the 20^{th} century, a lot of industries working in the automotive field started to invest a lot of money and resources to develop systems providing cars with some level of automation, for example, in 1920, the Japanese company Tsukuba Mechanical designed a vehicle equipped with two cameras capable of perceiving the surrounding environment during a short trip at very low speeds. In order to meet the first real self-driving vehicle we need to wait until 1984, when a research group at Carnegie Mellon University in Pittsburgh, Pennsylvania, developed a number of vehicles of different types which could be controlled remotely.
In the last 30 years all the major car industries have grown their interest in the field of autonomous vehicles and this led to the development of several communication protocols and devices allowing vehicles to exchange information about themselves and the road users surrounding them. These data, according to the level of autonomy vehicles are provided with, can be used on the one hand to allow vehicles to take autonomous decisions and perform autonomous actions in order to avoid collisions and, on the other hand, to warn their users as well as other traffic participants of potential dangerous situations on the road, eventually including suggestions about protective actions to take.
It is important to notice that vehicles and similar are not the only users active on the road and thus not the only ones to take into consideration when developing smart mobility technologies.
Another important category of road users are the so called Vulnerable Road Users (VRUs), traffic participants like pedestrians and cyclists that, in case of collisions, are the ones experiencing the greatest dangers. In the last years various institutions have started

developing communication protocols allowing these users to exchange information among themselves as well as with other road users. Vulnerable Road Users, just like vehicles, must be equipped with communication devices allowing them to share their main status and attribute information, in such a way to interact with all other traffic participants and thus become actual Intelligent Transportation Systems.

1.1 Intelligent Transportation Systems

When we talk about Intelligent Transportation System (ITS) we refer to a transportation management and service system, which includes a set of instruments designated for the management of transport networks as well as several services for road users. Intelligent transportation systems generally aim at enabling users to be better informed about the traffic environment surrounding them in order to make a safer and smarter use of the transportation network.
One of the most important characteristics of the ITS is that it generally combines the traditional transportation infrastructure with technologically advanced information systems, improved communication, sensor and control devices and advanced mathematical methods. The purpose of ITS is therefore to take advantage of all these new technologies to create a more intelligent traffic environment.
The heartbeat of an ITS is collecting, elaborating and integrating the data related to the road users and infrastructures in such a way to provide traffic participants with all the information they need to travel safely and efficiently. Other than this, an ITS has three main aims [14]:

- Mobility: the transportation system needs to be efficient and capable.
- Sustainable transportation: the traffic environment needs to be as safe as possible and it is also important to develop an environmental-friendly transportation system.
- Convenience: travelers need to be provided with an easy-to-access service.

Two very important features of an ITS are interconnection and operation. Interconnection is used to connect all the elements (vehicles, people, road and infrastructure) present in the transportation system, while operation is related to transportation management and allows to achieve goals with the minimum possible use of resources.
The architecture of an ITS is made up of 4 layers [14]:

- Physical layer: it is made up of all the elements included in the transportation system, the infrastructure, the vehicles and the people. Thanks to technological development, nowadays nearly anything can be seen as an active traffic participant, that can perceive the surrounding environment, control its actions and exchange information with other road users or infrastructures. For this reason, all traffic participants, thanks to sensors and cameras, can collect different types of traffic data and adapt their behaviour to the changes undergone by the environment.

- Communication layer: it allows the exchange of information between the different subsystems of the ITS and makes sure that this exchange of data is accurate and timely when needed.
- Operation layer: it collects all the data exchanged by the elements of the transportation system and translates them into useful information, which will then be distributed to the components of the ITS.
- Service layer: it is where services are deployed and run.

One of the most important topics related to ITS which is currently being addressed is the one about Cooperative-ITS (C-ITS), which can help improving road safety and, at the same time, help us realising full autonomous driving through the exchange of data via direct wireless short range communications. The main feature of C-ITS is that vehicles can communicate with each other as well as with infrastructure, thus improving the quality of the information related to road users, such as their location, and to the road environment. In particular, C-ITS are characterised by the presence of two or more ITS sub-systems which can cooperate with each other, thus providing better quality services. Vehicles and roadside stations exchange information through Dedicated Short-Range Communications (DSRC). DSRC operates in the 5.9 GHz frequency band and it is generally used for transmissions over short to medium distances. DSRC can be very useful for vehicular communications inside the ITS, since it has a very low latency and high reliability due to the fact that, being it used only on short distances, it receives very little interference, also in extreme weather conditions. When DSRC is used for communications between road users, such as V2V (Vehicle to Vehicle) or V2P (Vehicle to Pedestrian), it allows exchanging data through OBUs (On-Board Units), while, if the communication takes place between road users and infrastructure, DSRC allows the OBUs on road users to exchange information with the roadside units (RSUs) installed on the infrastructure.
Even though Intelligent Transportation Systems are nowadays being rapidly developed and an always growing number of institutions, research centres and industries is investing a lot of resources on their development and deployment, ITS research and application still faces a lot of complications and bottlenecks which will not be easy to eliminate.
One of the main problems of ITS is related to perception, in terms of precision, density and reliability. Intelligent Transportation Systems are indeed able to get a full sketch of the whole transport network thanks to a huge number of sensors which are both installed on the transportation infrastructure and on vehicles themselves, however, due to a number of different limitations, typically mainly related to costs, it is not feasible to cover all the desired area with enough sensors. This means, on the one hand, that it is not possible to guarantee the precision and density of raw data and, on the other, it leads to a problem of reliability of the collected data, which are used by the signal control systems to perform all necessary computations.
The second important problem related to Intelligent Transportation Systems is computational feasibility, since one of the main goals in current stage is to guarantee ITS correct operation even in huge traffic networks. Several traffic control models have been proposed in the past decades, however, if, on the one hand, sophisticated models are able to describe the system more accurately, on the other hand the use of complex and extremely

accurate models causes the computational cost to increase exponentially as the system extends and this leads these models to be infeasible in daily traffic operation. In the last few years, a number of optimal strategies with the aim of reducing the computational cost of the aforementioned models have been proposed, however the gap between theoretical models and application is still quite important. [14]

1.1.1 IoT-enabled ITS

Intelligent transportation systems integrate several different IT technologies, which include both traditional and new intelligence technologies, such as IoT, i.e. Internet of Things. In the context of ITS, IoT can play a crucial role for collecting, analyzing and managing large quantities of data, as well as it could provide safe and efficient services for all applications, anywhere and anytime.
Traffic data are usually collected from physical sensors, such as GPS, cameras and so on, however these sensors have several drawbacks, especially if we refer to specific applications, for example such devices may have reliability problems and for what concerns specifically cameras it is difficult to get clear images in adverse weather conditions. Meanwhile, nowadays people continuously use social media to publish their current locations and daily activities, which means that social networks can be used to extract a huge amount of traffic-related information that, in some cases, can be used more effectively than those retrieved through physical sensors. For example, this kind of social media data can be used to provide traffic information related to an area where there are no physical traffic detectors. The combination of the physical and network world on IoT-enabled ITS gives birth to what we can call Parallel Transportation Systems (PTS), new smart transportation systems able to gather, process and generate a vast amount of data of different nature.
Since PTS map the physical and the social system into one integrated system, their flow of working can be organised in 3 consecutive steps:

1. The actual transportation system is modeled and described by setting up artificial transportation systems.

2. Computational experiments are performed in order to predict possible future changes of the system and to evaluate control plans.

3. The two systems, the actual and the artificial ones, are executed in an interactive parallel mode, in which, instead of trying to make the status of the virtual system approach the status of the actual one, we try to make the actual system approach the ideal status of the virtual one.

The aim of this process is therefore to use the information coming from the observation of the complex phenomena governing the interactions between people and those obtained by modeling the transportation system using the basic rules of local traffic behaviour and individual road users to get a deeper understanding of traffic flow generation and evolution.
The infrastructures needed to build such a comprehensive transportation system are

provided by IoT, which allows to integrate the physical world with its virtual counterpart. On the one hand, we have physical sensors and devices of different nature, installed on cars, on roadside infrastructures or even on mobile devices, which form a distributed sensor network by exchanging data and information with each other. These information are then synthesized in such a way to elaborate traffic information such as traffic flow, vehicle trajectory, average speed, etc. Exploiting then the IoT technologies, the data gathered in the physical world are combined with those extracted from the social space using social sensors and signals. Social media like Facebook, Twitter and Instagram provide in fact a huge amount of social signals in real time, which can be used to build artificial traffic scenes using virtual reality. These artificial scenes are quite similar to the actual ones, including all the main features a physical traffic scenario would normally possess, like all the static and dynamic objects present on the road, illumination, weather conditions and so on. At this point, the social traffic scenario we have built is used to continuously process and analyse different traffic situations in real time.
If IoT technologies improve the sensing capabilities of ITS, it is also worth notice that nowadays the new theories and applications related to artificial intelligence and big data make the virtualization and modeling of physical transportation systems and scenarios much easier to perform.
IoT not only provides an increase in the amount of traffic data which can be gathered and an improvement in their quality, but it also allows to reduce the computational cost of storing and analysing such a large amount of data. Road users of all types, such as vehicles and pedestrians, are nowadays commonly equipped with smart devices provided with GPS technology, which means that they can easily collect traffic-related data of all sorts, such as speed, acceleration and position without any extra cost. It is therefore very economic to exploit these devices to collect all the traffic data we need, which, instead of being transmitted to remote data centres to be analysed, as it happens with traditional ITS, are stored and analysed locally using edge computing technologies, taking advantage of the memory and the computation resources of the smart devices themselves. The other huge advantage in terms of computational cost is that, with the help of IoT, instead of collecting a massive amount of real world data using physical devices, we can acquire a smaller amount of basic traffic information such as traffic flow and origin-to-destination census and use them to build virtual scenarios, which can be exploited to continuously generate artificial data.
The artificial transportation systems are based on a prediction model which is trained using both real and synthetic data. For this reason, at the beginning, the artificial system could generate traffic data that may not perfectly resemble the ones obtained from the real system, however, as time goes by and more and more data are collected from the real transportation system, the prediction model and consequently the artificial systems are refined and the data generated by these latters get more and more similar to the actual ones.

1.1.2 Levels of automation

SAE defines a taxonomy of driving automation which consists of six mutually exclusive levels of automation. This taxonomy is defined according to the respective roles the

human user and the automated driving system have, in fact, if the functionalities of the automated driving system change, so does the role of the human user. In particular, as the level of automation increases, the human driver performs less and less actions, his roles are progressively limited in favour of the automated driving system, which gradually takes over all driving functionalities.
The two lower levels of automation see the human driver keep performing at least some of the dynamic driving tasks also when the automation driving system is activated, in fact, in this case, the tasks performed by the automation system are referred to as "driver support" features. The three upper levels of automation are instead referred to the cases in which all the dynamic driving tasks are entirely performed by the automated driving system, which implies that the tasks performed by the automation system are in this second case called "automated driving" features.
The six SAE levels of driving automation are defined as follows [24]:

- Level 0 - No driving automation: the driver always performs all dynamic driving tasks and the driving automation system, if there is any, does not perform any driving task, while it can receive warnings about possible dangerous situations by other road users.

- Level 1 - Driver assistance: the driving automation system performs some of the dynamic driving tasks, in particular it performs either longitudinal or lateral motion control subtasks, and, as soon as the driver requests it, it disengages. The human driver therefore performs all the dynamic driving tasks not performed by the automation system and, above all, he always supervises the driving automation system in such a way to intervene whenever necessary. The engagement and disengagement of the automation system is also determined by the human user, meaning that, whenever desired, he can take back full control of the vehicle and perform all the driving tasks by himself.

- Level 2 - Partial driving automation: when engaged, the driving automation system performs part of the dynamic driving tasks and, in this case, it performs both the lateral and longitudinal motion control substaks, however, just like it happens with level 1, it disengages as soon as the human driver requests it. Also in this case, the human user performs all the remaining dynamic driving tasks and he always supervises the automation system, which engages and disengages based on the driver's request.

- Level 3 - Conditional driving automation: when the automated driving system is engaged, it performs all the driving tasks included in its operational domain and, if it determines that the limits of this latter are about to be exceeded, it requests the human driver to intervene, human driver that must therefore always pay attention to possible incoming requests from the automation system in such a way to be able to timely take the control of the vehicle and perform at least some dynamic driving tasks. Similarly, the human driver must also always be able to intervene when there is a performance-relevant failure of automated driving system, which therefore asks the human user to immediately take over the vehicle. Also in this

case, the engagement and disengagement of the automation system is determined by the human user.

- Level 4 - High driving automation: when engaged, the automated driving system performs all the dynamic driving tasks within its operational domain and again, when it detects the limits of this latter are being exceeded, it may request the human driver to take back the control of the vehicle. The automation system returns the control to the driver also when a performance-relevant system failure is detected and, similarly to level of automation 3, it is the human user who decides when to engage the automated driving system, while, for what concerns its disengagement, the automation system does it, if appropriate, only when it achieves a minimal risk condition or when the driver is performing some dynamic driving task. The human user is in this case indeed more a passenger than a driver, in fact it is not up to him to determine whether and how to achieve a minimal risk condition and, even if when it requests the automation system to disengage to take back the control of the vehicle, for safety reasons the automated driving system may decide to delay its own disengagement.

- Level 5 - Full driving automation: when engaged all driving actions are performed by the automated driving system and all transitions to a minimal risk condition are performed automatically whenever a performance-relevant failure takes place in the automation system. The human user determines whether and when to engage the automation system and, as soon as this latter is engaged, he becomes a passenger, since he doesn't need to perform any driving task and neither to supervise the automated driving system. Also in this case, the human user can ask the automation system to disengage in such a way to have the control of the vehicle back, however, just like it was with level 4, the automated driving system can delay the user request of disengagement for a number of different reasons.

1.2 V2X - Vehicle to Everything

As previously explained, communications of vehicles with each other (V2V), as well as with the infrastructure (V2I) and with other road users, such as vulnerable road users, can contribute to the safety and comfort of traffic participants and may also allow for a more efficient and widespread traffic management, in such a way to reduce or even prevent road congestion, but also to provide road users information allowing them for example to reduce fuel consumption and therefore emissions. All these different communication modes can be summarised with the term V2X, which is the acronym for Vehicle to Everything. V2X aims at creating links between the different players of a transportation network, in such a way that they can exchange information and data with each other, creating a fully connected traffic scenario.
We can classify the applications enabled by vehicular communications in 3 main categories: 1) Road safety applications, 2) Traffic efficiency and management applications and 3) Infotainment applications. [15]

Road safety applications are the ones which are more likely to have a very high market penetration and, due to the fact that they are designed to avoid collisions among vehicles and between vehicles and other road users, they set the benchmark in terms of reliability, message frequency and maximum latency. In terms of hazardous situations on the road, safety applications take advantage of the data collected using sensors like cameras, radars and lidars and they integrate them with all the traffic information exchanged through vehicular networks in such a way to detect potential dangerous situations and consequently perform evasive actions to avoid them. Integrating the data acquired with physical sensors with information coming from other road users also allows safety applications to work properly in all-weather conditions, also when the visibility is low, in fact these applications are designed to be fully effective also when based entirely on vehicular communications, when the data coming from sensors are unavailable. The existing sensors will be mainly used as complementary devices to the communication network, indeed nowadays road users' position and velocity can be easily acquired from GPS data and the reliability of positioning will further improve as soon as high precision maps designed for autonomous driving will be available. Road users cal also use these data to build and maintain a dynamic map of their surroundings, allowing them for situational awareness and to identify potential dangers. [13]
Safety-related V2X communications typically have very low latency, in the range of milliseconds, and they can cope with high relative speeds and substantial distances between the involved ITS-Stations as well as with a high network load.
The other two categories can be classified as non-safety applications, meaning that they are not involved in the avoidance of hazardous situations on the road. In particular, traffic efficiency and management applications focus on providing an efficient management of the traffic flow, in such a way that a more sustainable and smart mobility can be achieved. Infotainment applications, on the other hand, focus instead on the integration of social network and entertainment features with the aim of improving the final user experience, implementing several innovative services that will revolutionise the way users experience mobility.
Nowadays there are several different communication protocols enabling V2X communications, which can be either based on Wireless Local Area Network (WLAN) or on cellular network.
For what concerns WLAN-based protocols, they use an adapted version of the IEEE standard 802.11 for WiFi communication, called 802.11p, specifically designed to meet the aforementioned requirements, as access layer. The 802.11p amendment defines the physical layer for communication and, on top of it, for the higher layers, the ETSI ITS-G5 and WAVE standards have been developed, in Europe and in the US respectively.
In order to facilitate and promote the development of vehicular communications, the various telecommunication organisations have decided to allocate a range of frequencies for the exclusive usage of ITS services.
In the US, the FCC (Federal Communications Commission) has allocated 75 MHz, between 5.850 and 5.925 GHz, for vehicular communications. This frequency band is divided into seven 10 MHz channels, each provided with a 5 MHz guard band. These channels are then divided into a Control Channel (CCH) and a Service Channel (SCH), with the

first one used to advertise services and transmit control messages and the second one used instead for the transmission of all data related to applications and services.

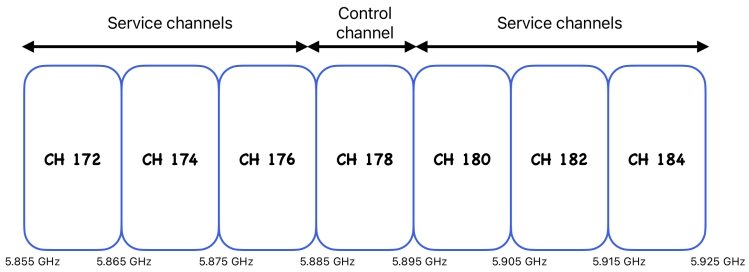

Figure 1.1. FCC spectrum allocation for vehicular communications. Inspired by [15].

In Europe, the European Commission reserved the frequency band spanning from 5.855 to 5.925 GHz to vehicular communications. Also in this case, this band is divided into seven 10 MHz channels, each used for a specific purpose. As reported in Figure 1.2, the first two channels, 172 and 174, are used for non-safety applications, while channels 176 and 180 are reserved to safety applications. Channel 178 serves as Control Channel and channels 182 and 184 are reserved for future applications. Another interesting characteristic of the European standard is that ETSI ITS-G5 introduces a fine-grained Service Channel assignment, thus 4 different types of channel are defined [15]:

- ITS-G5A: channels 172 and 174, they are reserved to non-safety related applications.
- ITS-G5B: channel 176, 178 and 180, they are used for safety-related applications.
- ITS-G5C: channels in 5.6 GHz band, they are used for infrastructure-based broadband radio access networks.
- ITS-G5D: channels 182 and 184, they are reserved for future ITS applications.

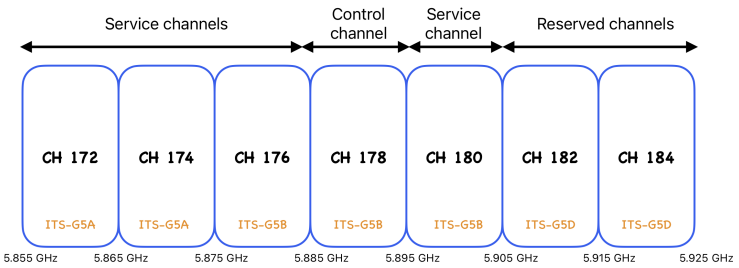

Figure 1.2. EU spectrum allocation for vehicular communications. Inspired by [15].

On the other hand, V2X communications can be also based on cellular networks and in this case we talk about C-V2X. C-V2X was standardised by 3GPP and, unlike standard cellular communications, it doesn't need any SIM, carrier access or licensed spectrum to work. With respect to IEEE 802.11 based standards, which are characterised by a peer to peer communication and by messages being directly sent to any listening user, C-V2X needs a network operator to operate and it needs to pass the uplink and downlink channels of the cellular network to reach the desired destination.
In terms of spectrum, in Europe C-V2X and 802.11p can coexist by being placed on different channels in the ITS band, while in the USA FCC decided to reserve the upper 5.9 GHz band to the exclusive use of C-V2X.

1.3 Importance of smart mobility

According to the World Health Organization (WHO), each year nearly 1.35 million people die worldwide due to car accidents and between 20 and 50 million people more are involved in non-fatal road accidents, with many of them being disabled due to the injuries. Other than this, road accidents cost most countries on average 3% of their GDP. [8]
The National Safety Council (NSC) reports that, in the U.S., VRU fatalities have been growing steeply in the last 10 to 15 years. In 2016, for example, a 9% increase with respect to the previous year was registered in terms of pedestrian fatalities, bringing the total count up to 5987 casualties and we can notice that a similar trend is exhibited by the number of fatalities involving any other type of VRU, from cyclists to e-scooters. These data become even more alarming if we consider that, in the same year, pedestrian fatalities accounted for 16% of traffic casualties, which is clearly an enormous percentage and which means that pedestrians and more in general VRUs are by far the class of road users experiencing the greatest danger on the road. The NSC then concludes its study by stating that, according to the data reported by the National Highway Traffic Safety Administration (NHTSA), if it was possible in the U.S. to get rid of collisions between vehicles and VRUs, over 11000 lives could be saved each year in the country. [16]
Given these numbers, implementing strategies and technologies allowing to reduce the number of road accidents, especially when vulnerable road users are involved, is of vital importance for all countries all over the world and intelligent transportation systems, together with the related infrastructure, can help achieving this result. For example, Traffic Management Systems can be used not only to enforce road safety regulations, but also to collect data coming from different sources and they can analyse them looking for potential dangerous situations with the final aim of delivering facilities to mitigate them. More in general, real-time traffic network control services are designed to provide traffic participants with tools which can help them cope with network congestion.
Traffic safety depends on several factors, such as roads architecture and configuration, the efficiency of traffic laws as well as the efficacy in their execution and non-human factors like weather conditions. Since traffic congestion is a recurring problem, Traffic Management Systems must be designed to deal with it and Vehicular Ad Hoc Networks (VANETs), by using vehicles' capability to track their own displacement and to compute

and distribute traffic information to other road users, can be used to make these latters more effective. It is important to take into account that the use of wireless and cellular technologies allows VANETs to also include RSUs and many different types of road users rather than just vehicles, like VRUs.
The importance of VANETs and connected road users is also evident when dealing with secondary crashes, which take place when other vehicles or different types of road users are exposed to a risk of collision due to a nearby impact. For example, there are several studies showing that rear-end collisions between vehicles are a common secondary crash type [8]. Vehicular communications can play a crucial role in avoiding this kind of collisions, since, whenever a road user detects an imminent risk of conflict, it could broadcast a message warning nearby traffic participants of the potential danger, so that these latters can proactively trigger protective actions in order not be involved in the collision.
Since traffic accidents contribute to the loss of an enormous amount of lives each year, especially when vulnerable road users like pedestrians are involved, the final aim of this document is to define a specific type of communication protocol allowing VRUs to communicate with each other as well as with other road users, such as vehicles, making them aware of each others' presence in such a way that, ultimately, collision avoidance actions can be automatically triggered on motorised road users, thus improving the safety of VRUs.

Chapter 2

Standards for Vulnerable Road Users

2.1 Vulnerable Road Users

Vulnerable Road Users (VRUs) are in general defined as road users which are not equipped with a motor and which are not protected by an outer shield. Since these road users are defined as vulnerable, one of the fundamental aspects to take into account when talking about VRUs is that their vulnerability can be defined either with respect to their degree of protection in traffic or with respect to their degree of mobility.
According to ETSI, which is the European institute for telecommunication standards, there are several different types of road users that can be considered as VRUs. The most important ones are the following [11]:

- Pedestrians and roadway workers
- Animals that present a risk to other road users
- Wheelchair users
- Bicycles
- Users of skateboards
- Riders of cycles or other types of personal transporters equipped with an electric motor with maximum speed limited to 25 km/h
- Riders of motorcycles
- Riders of three-wheel mopeds and quadricycles having maximum speed limited to 45 km/h

. Among the road users listed above and that can be therefore classified as VRUs, it is possible to notice that some of them are drivers or passengers of motorized vehicles. It is however important to highlight that, in these cases, it is not the vehicle to be considered

as a VRU, but its riders, since these latters, being them unprotected by some sort of outside shield, are characterized by a greater risk of injury in case of collision with another road user.

2.1.1 VRU system

A VRU system is a system which is made up of at least one VRU and one ITS-Station (ITS-S) having a VRU application installed upon it. It is indeed important to mention that VRU applications can be found in any ITS-S, meaning that we can find them both in VRU ITS-Stations and in non-VRU ITS-Stations, such as vehicles and roadside units, since the aim of these applications is that of providing VRU-relevant information to all entities on the road, thus increasing the awareness of VRUs, providing other road users warnings about possible collisions with VRUs or even triggering automated actions in vehicles provided with a certain degree of autonomy. To this purpose, VRU applications mainly use information received from other ITS-Stations, however they may also use additional data provided by the sensors of the ITS-S itself.
Taking into account the definition of VRU system we have just given, a vulnerable road user is simply an actor interacting with this latter in a given use case scenario. In particular, the VRU can directly interact with another VRU or another ITS-S only if both of them are provided with a VRU device, while, on the other hand, if the VRU is not equipped with such a device, this latter can only interact indirectly, since the other ITS-Stations inside the VRU system will only be able to detect the VRU through their sensors and cameras.
With respect to the equipment it can be provided with, a VRU can be classified as follows [10]:

- (Unequipped) VRU: the VRU is not equipped with any VRU device
- VRU-Tx: the VRU is equipped with an ITS-S, provided only with with a transmitter disseminating awareness messages containing information about the originating VRU
- VRU-Rx: the VRU is equipped with with an ITS-S, which only has a receiver, in such a way that the VRU can receive and decode the messages transmitted by other ITS-Stations
- VRU-St: in this case the VRU is equipped with an ITS-S that is provided with both a transmitter and a receiver, so that the VRU can carry out both the functionalities which have been previously described

For what concerns the physical architecture of a VRU system, it shall be as shown in the following figure.

As you can see from Fig.2.1, the physical architecture of a VRU system is made up of three main ingredients:

- VRUs

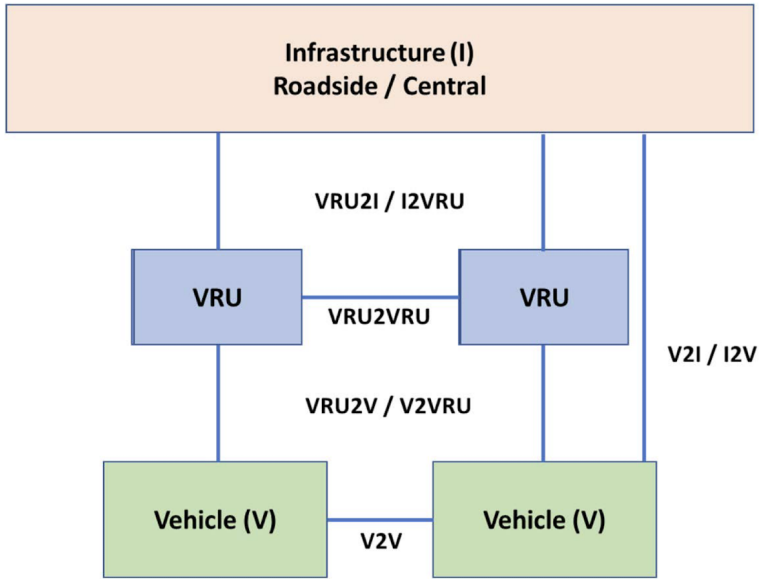

Figure 2.1. VRU system physical architecture. Taken from [10].

- Vehicles which are not considered as VRUs
- Infrastructure

In conclusion, the environment we have just described consists of ITS-Stations of different nature, which can communicate directly with each other as shown in Fig.2.1.

2.1.2 VRU profiles

In general, a VRU can only be a living being, however not all living beings are always considered as VRUs, indeed they become as such only when they are involved in a safety related traffic environment. This means, for example, that if a person, in a certain moment, is inside a house, this latter cannot be considered as a VRU, however, as soon as this person comes sufficiently close to the street, for example 2 m or 3 m, he becomes part of the aforementioned safety related traffic environment and therefore the person in question starts to be considered as a VRU.
There are several parameters according to which VRU profiles can be classified, which are however out of the scope of this document. We will only focus on the outcome of this classification, which results as follows [11]:

- VRU profile 1: pedestrians, meaning road users who do not make use of any mechanical device, motorized or not, for their trip. VRUs included in this profile are characterised by a very low maximum speed, the typical value is 5 km/h, a low communication range, especially in a urban or sub-urban environment, and a very high trajectory ambiguity, which makes them extremely important to take into account when dealing with collision avoidance.

- VRU profile 2: bicyclists. In this category we can find bicyclists and similar, such as riders of light vehicles, which can also be provided with small electric engines. This profile also includes wheelchair users, e-scooters, roller-skaters and others of this kind. The road users belonging to this category are characterised by a higher typical speed than those of the previous case, i.e. 20 km/h, a slightly higher communication range, which however, especially in a urban and sub-urban scenarios, remains quite limited, and also a lower trajectory ambiguity, which makes them a bit easier to deal with when it comes to trajectory prediction for collision avoidance.
- VRU profile 3: motorcyclists not equipped with engines, meaning that only the riders (driver and passengers) of motorcycles or similar motorized two wheel vehicles belong to this VRU profile. It is fundamental to highlight that the users of the aforementioned vehicles can be considered as part of this category of VRUs only if they are actually riding these two wheelers, otherwise they might still be considered as VRUs, but they will belong to one of the other profiles. Clearly, this VRU profile is characterised by higher speed values, similar to those of passenger cars in the case of motorcycles, a similar communication range to that of the VRUs belonging to the previous profile, with however a slightly higher maximum value, and generally also a low trajectory ambiguity.
- VRU profile 4: animals that present a potential danger to other road users, such as dogs, horses, wild animals, etc. The majority of the VRUs belonging to this profile will only be indirectly detected and, as a consequence, reported, however some of them might also have their own ITS-Stations. These VRUs typically have a low typical speed, around 5 km/h, a similar communication range to the ones previously mentioned and, just like pedestrians, a high trajectory ambiguity.

2.1.3 VRU cluster

There are certain situations in which the number of VRUs located in a given area can be very high. It is therefore fundamental, in this cases, to group VRUs together, in such a way to reduce the amount of messages exchanged and consequently the resource usage. This approach leads to the creation of VRU clusters, which are groups of VRUs that can be made up of vulnerable road users of the same type, in this case we refer to homogeneous VRU clusters, made up of for example only pedestrians, or of different types and in this case we talk about heterogeneous VRU clusters, consisting for example of both pedestrians and cyclists.
One important aspect to highlight about VRU clusters is that they are considered as a single object, which means that only the VRU which was designed to be the leader of the cluster, the so-called VRU cluster leader, can transmit and receive ITS messages. All the parameters related to the cluster, such as its dimension or cardinality, are encoded in the messages transmitted by the leader of the cluster, in fact the ITS messages sent by this latter are slightly different from those transmitted by an individual VRU, as they will include an optional container containing all the main information related to the cluster itself and to the VRU that is leading the cluster. Among other things, the cluster leader will also indicate in its messages if the cluster it is leading is heterogeneous or

homogeneous, which is a fundamental information to include since it can provide insights about the possible trajectories and behaviours of the individual VRUs when the cluster is broken up.
As we have previously mentioned, VRU clusters are fundamental since they can help reducing the occupancy of the communication channel, however there are some basic requirements two or more VRUs have to meet in order to build a cluster. In particular, all the VRUs forming a cluster must move in a coherent manner, which means that they must move with coherent velocities and directions and within a bounding box.
The first condition specifically means that the velocities and the directions of motion of the VRUs being part of the cluster must be such that the speed and heading differences between any of the VRUs in the cluster are below predefined thresholds, which will be discussed later in this document. The second condition, on the other hand, refers to the fact that all VRUs in the cluster must be located inside a virtual bounding box and must make contact with the surface of this latter at approximately the same elevation. If two or more VRUs respect these conditions, we can affirm that the messages they send will contain very similar motion information, since their position, speed and heading are almost the same, which is the reason why, in order to reduce the load affecting the communication channel, they can be grouped in a cluster and the leader of this latter will speak for all its members.
Since a VRU cluster is considered as a single entity, as we have just said, only one ITS message is generated per cluster and it contains all the most important information about the cluster, which are its ID, which is relevant when more clusters are generated, the number of participants, also known as cardinality of the cluster, the dimension and the shape of the bounding box, its speed, which is the same as the speed of the leader of the cluster, and its position, which again is the same as the position of the cluster leader and it is often referred to as the cluster reference position.
It is now important to understand how a VRU cluster is managed by its leader and, in particular, how this latter handles its creation and breaking up as well as how it behaves when a VRU decides to join or leave the cluster [10]:

- Creating a cluster: the future cluster leader decides whether to create a new cluster based on the information included inside the messages it receives from other VRUs. If a VRU determines to create a cluster, it broadcasts a specific message indicating that it will lead the cluster and it specifies the identifier of the cluster.

- Joining a cluster: a VRU decides whether or not to join an existing cluster based on the information included inside the messages it receives from the leader of the cluster. If the VRU decides to join the cluster, because its measured position and kinematic state are similar to those indicated in the last message transmitted by the cluster leader and, in particular, they are such that all the aforementioned requirements are satisfied, it must broadcast a message saying that it is joining a cluster and it must specify the identifier of the cluster and that it will stop transmitting messages afterwards. This procedure is very important since it allows the VRU cluster leader to always know the composition of the cluster it is leading, in particular if it is heterogeneous or homogeneous, its cardinality and its main

parameters such as its dimension, its position and its speed. The VRU that decides to join the cluster then waits for a response from the cluster leader and, if it finds out that this latter has not received its last message with cluster join indication, it broadcasts a new message saying that it will leave the cluster and that it will resume its normal operation.

- Leaving a cluster: similar to the previous case, a VRU decides whether or not to leave a cluster according to the information included inside the messages it receives from the cluster leader. If the VRU determines that its position and kinematic state do not satisfy anymore the conditions about the creation of a VRU cluster that were previously described, then it must break its silence by transmitting a message indicating that it is leaving the cluster and that it will resume its normal operation.
- Determining cluster leader loss: in some cases it might happen that the leader of the VRU cluster loses communication with the other members of the cluster or simply fails as a node. Since normally it is the cluster leader that broadcasts ITS messages on behalf of the cluster, if there is no message transmission for a predefined amount of time, the VRUs in the cluster assume the leader to be lost and leave the cluster following the exact same procedure described above.
- Breaking up a cluster: when the cluster leader determines to break up the cluster, it must send a message saying that it will break up the cluster and it must specify the identifier of the cluster. Typically, the breaking up indication are sent by the cluster leader a predefined amount of times in consecutive messages.

2.2 VRU awareness

It is possible to distinguish three different types of road traffic situations for what concerns the safety of VRUs:

- An upcoming collision of a VRU with another road user causing a safety risk for the VRU itself.
- In order to avoid the the collision proactively, it is possible to increase the safety of a VRU by raising awareness of this latter in the other road users.
- If a VRU with special needs is present in the traffic scenario, the traffic efficiency can be increased according to the needs of the VRU.

We can therefore say that, in general, a VRU can be considered as particularly vulnerable in traffic situations where collisions with other road users can take place. In particular, the intersection of the trajectories of the VRU in the x,y and z planes with those of the other road user represents the traffic conflict point and the collision will take place if both users reach the conflict point at the exact same time. In order to avoid the collision, at least one of the two road users has to perform an emergency manoeuvre in order to adapt his speed and/or path accordingly.

According to the results of the VRUITS project funded by the European Union, the highest frequency of collisions between pedestrians and vehicles occurs when the pedestrian was crossing the road at mid-block, which can be therefore identified as the most critical traffic situation for pedestrians. On the other hand, the majority of accidents between cyclists and vehicles were found to take place at road junctions. [11]
For what concerns the safety risk of a VRU, it can be measured taking into account the speed difference, the distance or the time interval between the VRU and the other road users when passing through the conflict point. It is generally expressed by means of specific safety parameters, such as the Time-To-Collision or the Post-Encroachment-Time, which represents the time difference between a road user leaving the area of encroachment and another road user entering in the same area.
Traffic accidents depend on the topology of the road and on the travel direction of the VRU and the other road users and their analysis is very important to assess the potential risks faced by the VRUs in different conditions and thus develop new technologies which can help preventing collisions on the road.
The ability of a VRU to detect other road users and to exchange relevant information, other than the possible actions ITS-Stations can take on dangerous vehicles as well as the required behaviour of VRU applications strongly depend on the environment in which the VRU system operates. This environment consists of many different elements, all playing a crucial role in providing a safer traffic scenario to the VRU, from the roadside equipment used for for detection and communication to the vehicles and VRU devices located in the scenario, but also all the obstacles affecting detection or communication.
In terms of road layout, sidewalks and cycle lanes, borders and even simple vertical and horizontal road markings help increasing the separation distance between VRUs and other road users, thus reducing potential collisions among them, especially in the case of mid-block situations. On the other hand, clearly visible traffic signs at crossings and intersections can also be used to make potential conflict situations less likely. Other than this, even if this will not be examined in this document, it is important to highlight that the road layout defines the type of warnings, actions and awareness and the expected functionality of VRU applications, since it determines the legal road use in a certain situation and the traffic rules. For example, the VRU application may decide to disseminate warnings when the VRU is crossing an un-signalized crossing or even when the VRU himself or a vehicle crosses an intersection violating a red light. In general, it is indeed important that VRU applications take into consideration traffic scenarios where road users violate the road layout and where traffic legislation differs between different countries.
In order to allow vehicles and VRUs to exchange information and become aware of each other's presence, it is crucial that vehicles are provided with a communication device and a VRU application, so that they can receive and decode the messages broadcast by the VRUs. Some vehicles can also be provided with pedestrian detection systems, such as sensors or cameras, to improve their awareness of other users that might be present on the road.
Roadside equipment can also be used to improve VRU safety and reduce dangerous situations on the road. They are characterised by two different modes of operation:

- Passive VRU detection: there is an IoT device, such as a camera or a presence detector, which detects the presence of the VRU and reports it to the VRU application in the Road ITS-Station.
- Active VRU detection: it is the VRU himself that informs the roadside equipment of his presence through a message which will then be forwarded by the roadside station itself to it VRU application.

In both cases, the VRU application inside the roadside station processes the information and, in the case of any safety issue, it broadcasts suitable event notifications.
There are in general different ways of avoiding the risk of collisions, as it is possible to make other road users aware of the presence of a VRU proactively, as well as VRUs can also be made aware of other road users' presence so that they can initiate protective actions in advance. VRU safety can be further increased by providing VRUs and other road users simultaneous awareness of each other. Even though awareness is always extremely important in terms of safety of VRUs, there are some traffic situations in which it becomes particularly relevant:

- The so-called black spot locations, for example schools and bus stops.
- Zebra crossings
- Traffic light conflicts between vehicles and VRUs
- The VRU enters inside an unexpected area, for example a pedestrian, for some reason, ends up on a bicycle lane or a highway.

Awareness is the basis for avoiding collisions between vehicles and VRUs. There are several different strategies to achieve collision avoidance between the two types of road users previously mentioned:

- When a vehicle detects a VRU, either with its own sensors or through an alert received from another vehicle or roadside station, it can slow down and/or change its trajectory.
- If it is the VRU that detects a collision risk with another road user, he can slow down and/or change his trajectory just like vehicles do.

2.3 Use cases

We now want to focus on the main potential use cases involving VRUs and we will analyse both the use cases presented by ETSI and those presented by the C2C communication consortium. It is important to highlight that these use cases are classified according to the different road users present in the scenario and to the different stakeholders that could help improving the safety of VRUs on the road. We will also present and describe some exemplary use cases where VRUs face a risk of collision with other road users and we will discuss how the C-ITS system can help mitigating it. Finally, some of the main challenges identified in the description of the different use cases will also be briefly

discussed, as they need to be taken into account when we will talk about the structure and the functionalities of the VRU basic awareness service.
Analysing and understanding these use cases is of crucial importance, since they are used to develop all the specifications of the VRU system functional architecture which will be discussed in chapter 2 of the present document.

2.3.1 ETSI use cases

According to ETSI categorization, use cases can be classified as shown in Figure 2.2.

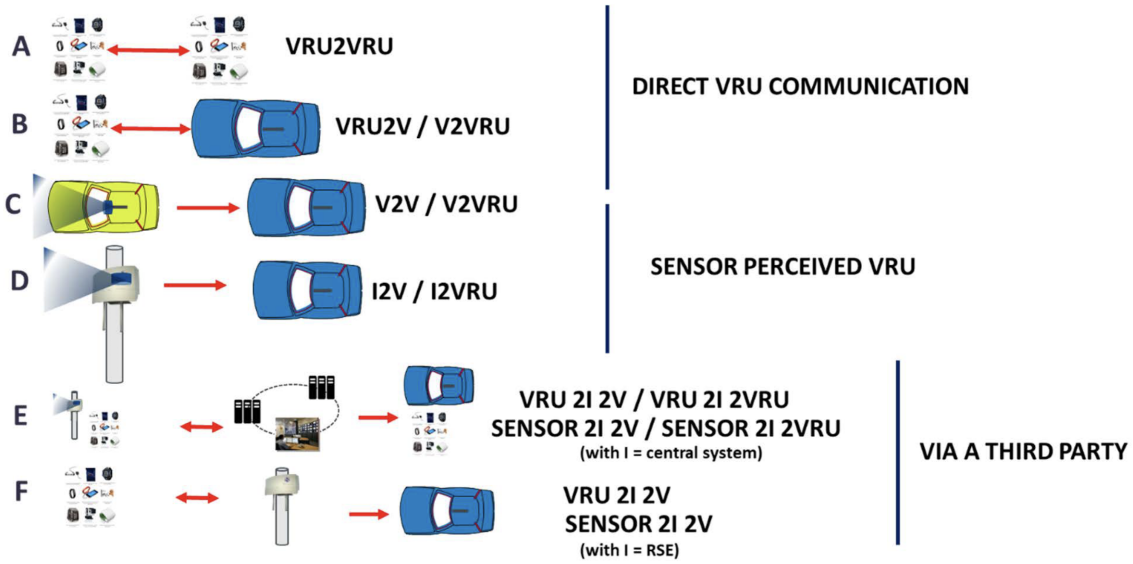

Figure 2.2. Use cases categorization according to ETSI. Taken from [11].

Figure 2.2 shows six categories of use cases, which cover all the possible interactions between VRUs and other road users or roadside equipment, however, since the scope of this document is to focus on the interaction among VRUs, especially pedestrians, and between VRUs and vehicles, only use cases A and B will be discussed and analysed. It is also important to underline that the categorization of use cases presented above is environment independent, as it doesn't focus on any specific environment and it can be applied to any environment one can think of.

Category A

As shown in Fig.2.2, use case A deals with direct VRU communication. In this case, VRUs have to be equipped with a communication device having at least one ITS-Station installed. This device can either be a transmission only device (VRU-Tx), a reception only device (VRU-Rx) or a device which incorporates both functionalities (VRU-St). Some significant examples of traffic situations belonging to this use case are for example a pedestrian or a bicycle crossing the road and a powered two wheeler either going straight or turning at the intersection or pedestrians and bicycles moving in a longitudinal traffic

flow with either the same or opposite directions.
The first use case belonging to category A we can analyse is a sharing sidewalk between pedestrians and bicycles, which is probably the most typical example of VRU to VRU direct cooperation. In this case, each VRU has his own ITS-S and the VRUs are constantly exchanging messages allowing them to detect any risk of collision between them. Whenever a possible collision is detected, an automatic action, such as an alert message, can be triggered to avoid the conflict. The only actors present in this use case are VRUs belonging to two different profiles, which are profile 1 and 2, as stated in chapter 2.1.2, sharing a given area which can be arranged or not into separate dedicated lanes. When a risk of collision between VRUs is detected, a of collision avoidance action is triggered, which is generally an alarm, possibly provided with a recommendation, sent to those VRUs which are equipped with communication devices of type VRU-Rx or VRU-St.
The normal flow of information is reported in the following figure.

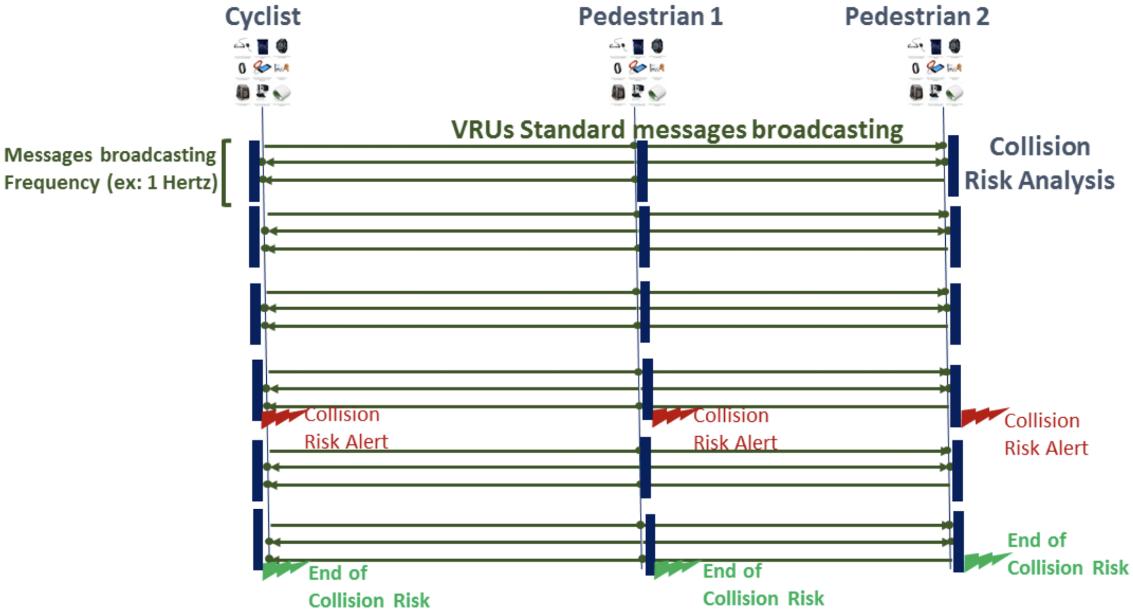

Figure 2.3. Flow diagram for use case 1, category A. Taken from [11].

Normally, VRU devices continuously broadcast VRU standard messages at a configurable frequency, in Fig.2.3 it is for example 1 Hz maximum. The ITS-S of the receiving VRU then processes the received messages in order to perform collision risk analysis. If one or more devices detect a collision risk, the VRUs of interest are alerted through an alarm together with a complementary recommendation advice, in such a way that the collision can be avoided. Once the collision risk has been processed, the system resumes its normal operation, thus it starts again broadcasting standard messages and analysing the content of the received ones, looking for a new risk of collision to be processed. This flow of operation performed by the VRU ITS-S keeps on going until the VRU decides to deactivate it.

This category of use cases only foresees alerts and advices to be sent to VRUs whenever a collision risk is detected, so no automated actions on VRUs' mobility is considered. For this reason, the risk in case of a cyberattack is very low, therefore a high security level for the system is not a priority. On the other hand, the accuracy of the geo-position of the VRU is extremely important, while, since in this case no motor vehicles are involved, the latency time of the system is less critical, due to the lower speeds.
Another possible use case belonging to this category is the situation of a pedestrian crossing a road with a road user such as and e-scooter approaching. In this case, there are one or more VRUs equipped with VRU-St equipment type, so able to both transmit and receive ITS messages, positioned at a crossroad with an electric scooter, equipped with a VRU device as well, approaching. In our example the e-scooter is equipped with a VRU-Tx ITS-S, so it is only able to transmit messages, meaning that the risk assessment has to be performed by the pedestrian, being this latter also able to receive messages. Other than being necessary for collision avoidance, the risk assessment performed by the VRU can also assist in adjusting the communication frequency, in such a way to reduce the network congestion, and it can be based on using the signals received from other road users to build a dynamic map or on context perception.
In terms of flow of operation, VRU standard messages are broadcast with a suitable frequency by the VRU-St and the VRU-Tx and the VRU-St performs the risk assessment. Whenever it identifies that the VRU is on a collision course with the e-scooter, the VRU-St warns its user, the pedestrian, of the imminent collision and it suggests him to wait for an appropriate amount of time before crossing the road. Once the risk assessment has been processed, the normal flow of operation is resumed. [11]
This use case presents several challenges, for example to achieve the operation previously described the positioning accuracy needs to be sufficient to determine that the two trajectories have a possibility to collide, as well as, in order to assess the risk of collision, an accurate knowledge of the VRU context is needed, in such a way to understand that the user means to cross the road.

Category B

As we can see from Fig.2.2, category B, on the other hand, deals with direct communication between VRUs and vehicles. In this case, the VRU must be equipped with a communication device, which can be of any type (VRU-Tx, VRU-Rx or VRU-St), having at least one ITS-S embedded.
A first use case we can analyze belonging to this category a traffic scenario with an active roadwork, meaning that there are human workers present and active in a certain area. Since, in this case, VRUs and vehicles are present in the same area simultaneously, they both have to be equipped with ITS-Stations complying with VRU standards, in such a way that both road users are at least able to receive VRU standard messages and so to detect and avoid possible collisions. In particular, the road workers' ITS-Stations will have to continuously broadcast VRU standard messages in order to provide other road users dynamic data elements related to their positions and states of motion. The vehicles involved in this use case can either be human driven or automated.
The following figure shows the normal flow of information for this use case.

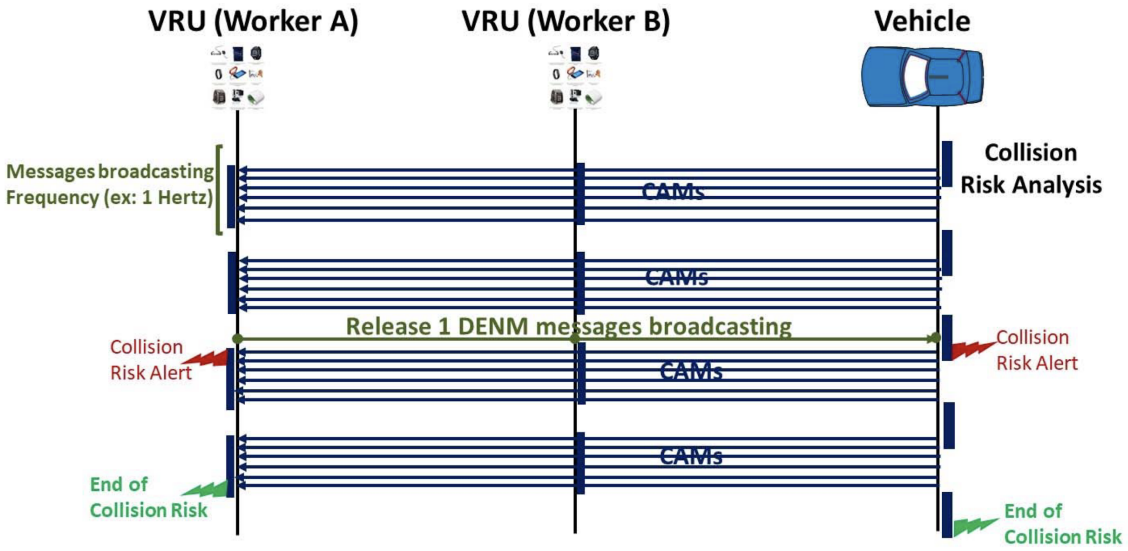

Figure 2.4. Flow diagram for use case 1, category B. Taken from [11].

As shown in Figure 2.4, vehicles constantly broadcast CAMs, which are processed by the VRU ITS-Stations for collision risk analysis. When one or more VRU devices detect a risk of collision, they alert the road workers of interest of the imminent collision risk, providing an alarm and eventually an advice of recommendation in such a way that the road worker and the vehicle can avoid the collision. If the VRU ITS-Stations also support DENMs, VRUs can also broadcast DENMs to vehicles when detecting a risk of collision, so that vehicles, upon reception of DENMs, can either send a warning to the driver or trigger an emergency braking. At the same time, the road workers not directly involved in the collision scenario can receive a message warning them of the presence of a dangerous vehicle and encouraging them to protect themselves. [11]
A second interesting use case belonging to category B involves a VRU crossing a road. In particular, in this case we have one or more VRUs, equipped for example with VRU-St equipment type and therefore able to both transmit and receive V2X messages, that are crossing a road. There are two possible approaches:

- Approach 1: the VRU ITS-S continuously sends VRU standard messages, but it doesn't perform any collision risk assessment since it has limited processing capabilities. In this case the risk assessment must be performed by some other road user, such as a vehicle, which evaluates the risk of having a collision with the VRU based on the VRU standard messages received and eventually also on the behavioural models of the VRU.
- Approach 2: the VRU ITS-S also performs collision risk assessment, which can also help in controlling the communication frequency in such a way to reduce the network congestion, since it has sufficient processing capabilities to do so. In this second case, in addition to the risk assessment performed by the vehicle, the level

of risk is also assessed by the VRU ITS-S based on context perception.

In terms of flow of operation, VRU standard messages are broadcast by the VRU-St, either continuously (approach 1) or when the assessed collision risk is higher (approach 2). If the risk assessment performed by either the vehicle ITS-S only or by both the vehicle ITS-S and the VRU ITS-S identifies that the two road users are going to collide with each other, it warns the ITS-Stations' users of the imminent collision. A collision warning is also transmitted to other relevant road users, to warn them of the presence of a nearby danger. The relevant ITS-Stations receive the warning and warn their users at the appropriate time. [11]

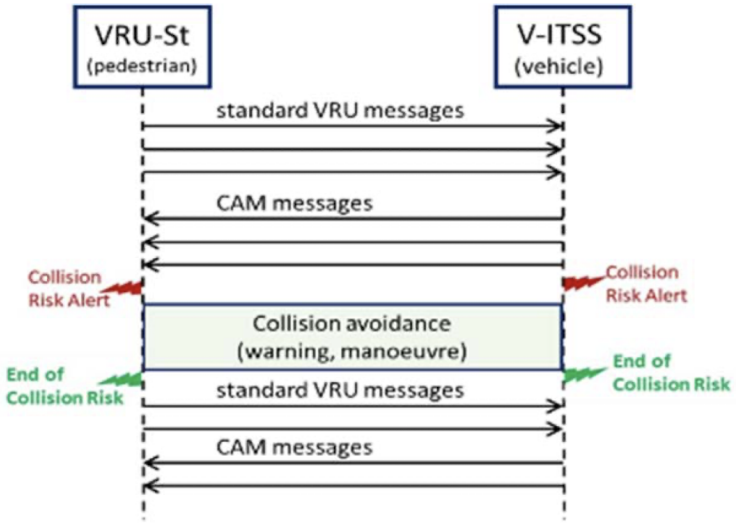

Figure 2.5. Flow diagram for use case 2, category B. Taken from [11].

2.3.2 C2C-CC use cases

Among all the use cases presented by the C2C Communication Consortium, we will analyse only those in which VRUs are directly involved. [7]
A first very basic scenario we can discuss is the VRU Presence Awareness use case, in which VRUs and other road users, such as vehicles, are informed about their mutual presence, especially in situations where, due to various factors, their awareness is limited or incomplete, such as in case of reduced visibility towards each other.
We can distinguish three different possible system architectures and implementation options to deal with the aforementioned use case:

1. Road users participating to the traffic scenario that are exchanging information and data related to the driving environment of the VRU or related to the current and future kinematic states of the VRU, such as connected vehicles reporting about themselves or about other ITS-Stations, can enable the awareness of the VRU presence. In this case, in order to make other ITS-Stations aware of their presence,

VRUs broadcast ITS messages which include information related to their current state of motion and their predicted trajectories. In return, the vehicles receiving these messages may decide to share with the VRUs information related to their kinematic states by transmitting CAMs (Cooperative Awareness Messages). Other traffic participants may decide to join the conversation by sharing their perception of the VRUs and other road users sensed through their on-board sensors by transmitting CPMs (Call Protocol Messages), a message protocol which will not be analysed in the present document.

2. There are cases in which an infrastructure able to detect VRUs and inform other traffic participants of their presence can be available. This infrastructure, typically made up of several RSUs, uses its sensors to detect VRUs and, more in general, road users in its vicinity and warns them of their mutual presence. If the RSU detects a traffic participant that could collide with a certain VRU, it informs this latter about the incoming danger by transmitting CPMs.

3. Mobile communication networks can also be used to detect VRUs and inform other traffic participants of their presence. The V2X communications from road users related to VRUs and other traffic participants in their proximity are used to collect meaningful data about the traffic scenario and the state of motion of the users active on the road. Once these data have gathered and suitably aggregated, they are transmitted to the involved ITS-Stations over specific links allowing the information to be sent of to the pertinent stations. These data may be encoded in CPMs or use some other format. This type of implementation can help avoiding flooding the V2X communication channel with VRU presence awareness messages, however it is generally not used for collision warning, since this latter has much more stringent latency requirements which would be very hard to meet with such a protocol.

A second interesting use case we case we can analyse is the VRU collision warning, in which information about the current state of motion of VRUs and other traffic participants are used to detect potential collisions and warn the involved road users.
In this case we can distinguish between two possible architectures and implementation options:

1. VRUs and other road users exchange information about their current state of motion and their predicted trajectories and this exchange of kinematic data enables the VRU collision warning, which can either be implemented by means of explicit collision warnings sent to the involved traffic participants or through an exchange of implicit information based on the dissemination of the kinematic states of the users active on the road. In order to better evaluate the traffic situation and the potential dangers coming from other traffic participants, road users can use information broadcast by means of CPMs or DENMs, other than the conventional kinematic information disseminated through the more conventional awareness messages, to activate VRU collision warning.

2. A specific roadside infrastructure equipped with several RSUs enables VRU collision warning by tracking all active users on the road and by sending them explicit or

implicit collision warnings. The use of RSUs to perform VRU collision warning is particularly important in traffic scenarios which can present high safety risks for VRUs. In order to allow RSUs to significantly contribute to the safety of VRUs, they need to be equipped with well-placed sensors, allowing them to cover the whole critical area and to accurately track all traffic participants. Typically, collision warnings are sent out through CPMs and DENMs, in such a way that it is not required to have additional ITS-Stations reporting the situation.

One last use case we can discuss is the VRU Brake or Steering Intervention, which differs from standard VRU collision warning since VRUs, that are in this case provided with a higher level of automation, can brake or steer by their own, without requiring active human intervention. Collision avoidance is in this case based on the flow of information between ITS-Stations related to VRUs and their kinematic states.
Just like in the previous case, there are two possible system architectures and implementations:

1. There are traffic participants provided with a very high level of automation that, when they receive information related to a potential imminent collision with a VRU, enable the VRU brake or steering intervention. This latter can either be activated by means of explicit collision warnings or by exchanging implicit information based on the dissemination of the kinematic states of the active road users. As for collision warning, road users can also take advantage of object information broadcast by other traffic participants by means of CPMs and DENMs, other than the kinematic information stored in the awareness messages, to enable VRU brake or steering intervention.

2. VRU brake or steering intervention is enabled by roadside units which track traffic participants and transmit information in the form of explicit or implicit warnings to enable automated driving actions providing VRU protection. RSUs are equipped with tracking sensors sharing data with VRUs by means of CPMs, which, in order to improve collision avoidance actions, can also include VRUs' predicted trajectories. Again, this system architecture is typically used in areas presenting a high safety risk for VRUs.

2.4 VRU-related functions in the ITS station architecture

We now have to talk about the main elements of the ITS-Station architecture that have a direct impact on the operation of the VRU system.
As shown in Figure 2.6, the ITS-S architecture can be divided into 4 layers: the access layer, the networking and transport layer, the facilities layer and the application layer [10]. Each layer provides different functionalities and, in particular, a new functional entity is introduced in the facilities layer with respect to other ITS-S architectures, the VRU Basic Service.

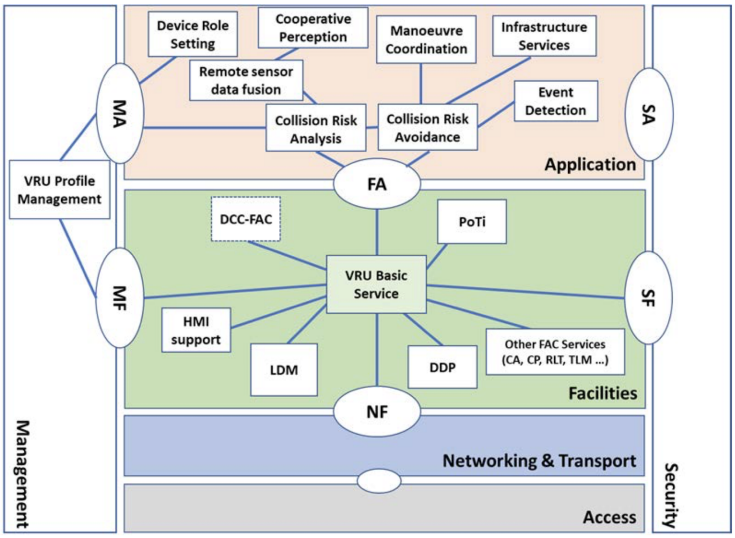

Figure 2.6. VRU-related functionalities mapped to the ITS-S architecture. Taken from [10].

As we can easily understand from Figure 2.6, all VRU-related functionalities shall be centred around the VRU Basic Service, which shall be therefore linked with the other entities, such as the application layer and the networking and transport layer, and shall also be responsible of transmitting the ITS messages, which, in the case of VRUs, are called VAMs (VRU Awareness Messages). Other than this, the VRU Basic Service must also interact with the VRU profile management entity, located in the management layer, in order to retrieve the state of the VRU role, which, as we know, can be enabled or switched off.
At this point, we can focus on the two top layers of the ITS-Station architecture, the application and the facilities ones, and have a look at their main functionalities.

2.4.1 Application layer

The application layer is the upmost layer of the ITS-Station architecture and its function is to interface and provide all the needed services to the applications.
As shown in Figure 2.6, in this case the application layer must fulfill many different functions and the most important ones are the following [10]:

- Device role setting: a VRU, in order to be able to exchange ITS messages, is generally equipped with a portable device which must be initially configured and which state, due to several factors, might evolve during its operation. For what concerns its initial configuration, it can be set-up automatically as soon as the device is powered up and of particular interest it is the setting of the type of the VRU device, which, as we have already discussed, can be VRU-Tx, which means that the device can only transmit ITS messages, VRU-Rx, meaning that the communication device is only able to receive messages, or VRU-St, which makes the device capable

of both broadcasting and receiving ITS messages. Other than this, due to context changes, the VRU profile might change during its operation, which implies that the communication device must be able to adapt its role accordingly.

- Remote sensor data fusion and actuator: VRU systems might be provided with elements which, collecting remote data, can help augmenting the local perception obtained via the data collected by physical local sensors. In this case, it is needed to fuse these remote data with the data collected by the local sensors and this data fusion may provide three possible different results:

  1. The remote data and the local data are not coherent with each other and therefore it is the system's duty to decide which data source to trust, ignoring the other one.
  2. Only one source of data is available, which means that the system is compelled to trust the only available source.
  3. The remote and the local are coherent with each other and, in this case, the two sets of data can be fused with each other, thus providing an augmented local perception.

  If available, it is also possible to make use of artificial intelligence to not only recognise and classify the detected objects, but also to retrieve their associated dynamics, which would make local perception even more effective.

- Cooperative perception: the overall perception of a VRU system can be seen as the result of the fusion of different independent perceptions at different times. First of all we have the local perception, which provides data by collecting information from the environment surrounding the ITS-Station and it is usually performed using several sensors of different types, like radars, lidars, cameras of different nature and so on. A second type of perception is the remote one, which provides the system mainly data collected via V2X communication. It is then necessary to check if the data collected via different perception functions are consistent and, if not, select the most reliable one according to the confidence level associated to the various perception variables. Associating a confidence level to each perception is very important, since, in case of differences between local and remote perception, the overall result originated by the different perceptions will also depend on it. The result of the cooperative perception should then be consistent with the required level of position accuracy.

- Collision risk analysis: this function analyses the motion prediction of the considered objects together with their degree of reliability with the goal of estimating the probability to have a collision. If a high risk of collision is detected, the collision risk analysis functionality tries to evaluate as precisely as possible the time to collision, in such a way to have a reliable estimate of when the collision is going to take place, giving in this way the possibility to the VRU to take evasive actions. Obviously, in order to be able to compute the time to collision, there have to be at least two moving objects whose trajectories of motion are going to intersect at some

point, called potential conflict point, and whose motion dynamics are maintained in time. Since VRUs' motion dynamic can be very unpredictable, the computed time to collision can be considered as reliable only when the VRU enters the collision risk area, which is typically very late, just before the conflict point is reached. The potential conflict point is located in the middle of the collision risk area, which can be defined on the basis of the lane and vehicle widths. The time to collision is only one of the many variables which can be used to detect a possible collision, for example the three variables Longitudinal Distance, Lateral Distance and Vertical Distance, together with the corresponding thresholds, called respectively Minimum Safe Longitudinal Distance (MSLoD), Minimum Safe Lateral Distance (MSLaD) and Minimum Safe Vertical Distance (MSVD), can be used for the same purpose. As it will be further investigated later in this document, the likelihood of a collision can be used to trigger the transmission of ITS messages.

- Collision risk avoidance: if the collision risk analysis provides some variables to help the ITS-Station detect possible collisions, the collision risk avoidance exploits the results given by the risk analysis to select suitable collision avoidance strategies. The collision avoidance strategy needs to take into account a lot of factors, for example the environmental conditions, the VRU capabilities, which depend on its profile, as well as the vehicle capabilities, the remaining time to collision and the conditions of the road. For example, when the road and weather conditions are good, the collision avoidance action can be triggered when the remaining time to collision is higher than 2 seconds, while below 2 seconds the vehicle can be considered in a pre-crash situation and therefore it has to trigger an impact mitigation action instead of a collision avoidance action, which would not be successful, in order to mitigate the consequences of the impact for the VRU. If roadside units are also present, the RSE can also implement collision risk analysis and avoidance, in such a way to suggest neighbouring vehicles and VRUs possible strategies to avoid or at least mitigate possible collisions.

- Event detection: when a VRU transits from one state to another, the even detection function assists the VRU Basic Service during this operation. In particular, the most common events related to this functionality are a change of role of the VRU, from on to off or vice versa, and a change of profile of a VRU, when this latter enters a cluster or when the cluster it is part of breaks up or when it starts using some mechanical element, such as a bicycle or motorbike. Some other events to be considered, at application level, are a risk of collision between VRUs themselves or between VRUs and vehicles, a change of the dynamic properties of the VRU, such as its trajectory and speed, and the change of the status of some road infrastructure which impacts in some way the movements of the VRU.

- Manoeuvre coordination: this function performs the collision avoidance actions which were decided by the collision avoidance strategy. This function must be present for all vehicles, while, in the case of VRUs, its presence depends on their profile. If roadside stations are present, they can suggest vehicles manoeuvre coordination, since they typically have a better perception of the dynamic states of the

objects in the traffic scenario. For what concerns VRUs, on the other hand, manoeuvre coordination can be enabled by sharing the Trajectory Interception Indication among the VRU and the surrounding ITS-Stations, which reflects the probability with which the VRU trajectory is going to be intercepted by the trajectory of some other ITS-S, and the Manoeuvre Identifier, which indicates the type of manoeuvre the VRU needs to perform to avoid the risk of collision.

2.4.2 Facilities layer

The facilities layer is a middleware made up of different facilities, which are functionalities, services or data used by the the application layer to provide ITS services.
As shown in Figure 2.6, the VRU Basic Service is the main entity belonging to the facilities layer and it is supported by the following functions:

- VRU basic service management
- VRU Cluster management
- VAM Reception management
- VAM Transmission management
- VAM encoding
- VAM decoding

The VRU Basic Service then interacts with other functions belonging to the facilities layer, as well as with other entities of the ITS-S, such as the application layer and the Transport and Networking layer.
Apart from the VRU Basic Service, another interesting facility present in the facilities layer is the Local Dyamic Map (LDM), which is a database containing all the dynamic data elements related to the functions which support the operations of the VRU Basic Service. VRUs and vehicles can access the content of the LDM through an exposed interface from dedicated functions in the facilities layer and the access to the LDM is controlled by the LDM management function, which can also be used to store and update data elements inside the LDM.

2.5 VRU Basic Service

The VRU Basic Service (VBS) is the facilities layer entity which supports the ITS applications managing the transmission and reception of VAMs, which are the ITS messages containing all the main information about the originating VRU ITS-S. VAMs format and specifications will be further investigated in chapter 2.6.
There are typically many applications which rely on the VRU Basic Service, however, besides the support of applications, the VBS can also be exploited in the networking and

transport layer for the position dependent dissemination of messages.
The VRU Basic service provides three main services [12]:

- Handling the VRU role: the VRU profile management entity informs the VBS whether the user device has to be considered as a VRU or not, according to the context it is in. There are two possible roles of the VRU during the VBS operation, which remains operational in both states:

  1. VRU_ROLE_ON: the user device is considered as a VRU, meaning that it can both transmit and receive VAMs. According to the information received from the VRU profile management entity, the VBS shall check the the profile of the VRU as well as its type, it must handle its own clustering state and it must provide services to other entities.
  2. VRU_ROLE_OFF: the user device is not considered as a VRU and therefore it must neither send nor receive VAMs. When in this state, the VRU is typically located in a zero-risk geographical area, such as a car, a bus or a pedestrian area, however the VBS remains operational to monitor when the VRU role changes to VRU_ROLE_ON.

- Sending VAMs: when a VRU wants to send a VAM, it must first generate this latter and then it can transmit it. For what concerns the generation of the message, the originating ITS-S first composes the VAM and then it delivers it to the ITS networking and transport layer, which will take care of its dissemination. During VAM transmission, instead, the message is transmitted over one or more communications media through one or more networking and transport protocols, in particular typically the originating ITS-S transmits the VAM to all ITS-Stations within its direct communication range. The VRU Basic Service of the originating ITS-S determines the frequency with which VAMs are generated and transmitted, based on the change of kinematic state and on the location of the VRU, as well as on the occupation of the communication channel, however, as previously mentioned in chapter 2.1.3, if the VRU ITS-S is part of a cluster, it does not transmit any message.

- Receiving VAMs: as soon as a VRU ITS-S receives a VAM, the VBS decodes the message, applying all necessary security measures, and it makes its content available to the applications and to other facilities belonging to the receiving ITS-Station, such as its Local Dynamic Map.

Going into more detail, the VRU Basic Service is not only responsible for the transmission and reception of VAM messages, but it is also the one in charge of identifying whether the VRU is part of a cluster and of enabling the assessment of potential collision risks by ITS applications. In order to collect useful information for VAM generation and transmission, the VBS consumes data from and provides data to other services located in the facilities layer. The Device Data Provider (DDP) and the Position and Time management (PoTi) are the facilities layer entities responsible for the collection of data and, in particular, the DDP exploits its local perception entities to provide status information to the device, while the PoTi retrieves the position of the ITS-S and the time information.

The Local Dynamic Map (LDM), on the other hand, acts as receiving terminal for the incoming data and it is a database in the ITS-S which is updated every time a new VAM is received, with the information included in this latter.
The VBS shall also interact with other entities outside the facilities layer, such as the VRU profile management entity in the Management entity to learn the role of the VRU ITS-S, its profile and its type, but also if it is part of a cluster and, in this case, if it is the leader of this latter or if it acts as an individual VRU. The VBS is also in charge of selecting the best available network and communication profile for the transmission of VAMs.

2.5.1 VRU Basic Service architecture

We will now analyse in detail the most important functionalities of the VRU Basic Service [12], as they are show in Figure 2.7.

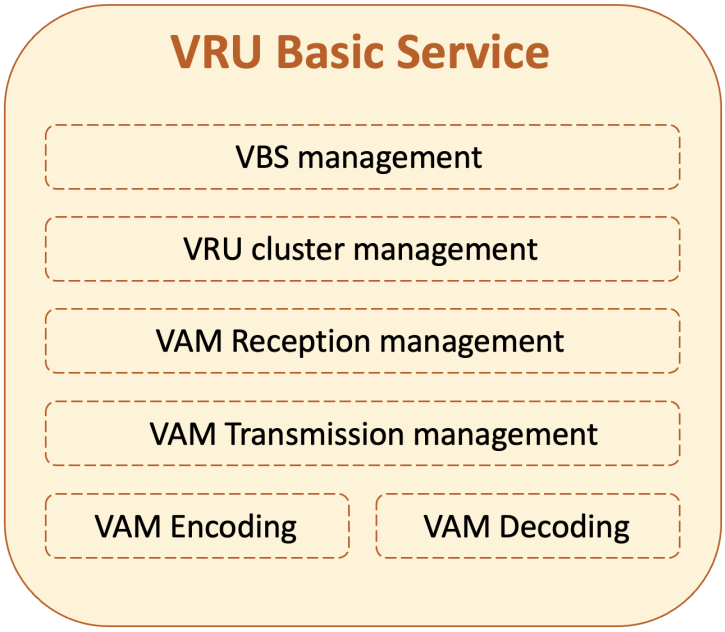

Figure 2.7. VRU Basic Service functional architecture. Inspired by [12].

- VBS management: the VRU Basic Service management is in charge of storing the port numbers to be used for the VBS and the configuration of the VRU at a given time for the encoding of the VAM, managing the interaction with other basic services, such as DDP and PoTi, activating and deactivating the transmission of VAMs according to the parameters of the VRU profile management and managing the triggering conditions for the transmission of VAMs taking into account the rules related to the network congestion control.

- VRU cluster management: this functionality is not mandatory in the VBS, however, when implemented, it should detect if the VRU is capable of becoming the leader of a cluster, manage both the creation and breaking up of a cluster, compute and store all the cluster parameters needed for generating a cluster VAM, detect if the VRU should join or leave a cluster, manage the state machine of the VBS according to the cluster events which are detected and, according to the VBS clustering state, activate or deactivate the transmission of VAMs.

- VAM decoding: this function shall decode the received VAM and extract all the relevant data elements from this latter. The extracted information are then handed over to the VAM reception management functionality, in fact the VAM decoding function is available only if also the VAM reception management function is available.

- VAM reception management: this function comes into play straight after the VAM message has been decoded and it must check if the received message is relevant, according to its current state and motion characteristics, determine if the received VAM meets all the main security conditions and delete or store the data elements of the received message in the LDM according to the outcomes of the previous operations. This function is not present in all VRUs as its presence depends on the VRU equipment type, in particular, if this latter is for example of type VRU-Tx and therefore only supports the transmission of messages, it will not include a reception management function.

- VAM transmission management: upon request of the VBS management function, this functionality is required to assemble the VAM data elements and send the constructed VAM to the encoding function. Just like in the previous case, the presence of this function entirely depends on the VRU equipment type, if indeed the VRU equipment is of type VRU-Rx and therefore only enabled to receive VAMs, but not to send them, it will not include any VAM transmission management function.

- VAM encoding: this function takes care of encoding the constructed VAM provided by the VAM transmission management function and it shall then trigger the transmission of the encoded VAM, via the dedicated port, to the Networking and Transport layer, selecting a suitable communication profile. This functionality is only available if also the VAM transmission management function is available.

2.5.2 VRU clustering function

The clustering function in the VRU Basic Service is meant to optimize the resource usage, both spectrum resources and processing resources, in the ITS system. This functionality is extremely important for the VRU Basic Services of ITS-Stations, since a huge number of VRUs all in the same area would result on the one hand in a huge amount of VAMs transmitted by the ITS-Stations and therefore a significant use of spectrum resources and on the other in an enormous amount of messages to be processed by the receiving ITS-Stations.

As it was previously mentioned, there is only on VRU per cluster enabled to send VAMs and it is the leader of the cluster, which sends messages containing information about the entire cluster. All the other members of the cluster are only allowed to send VAMs when joining or leaving the cluster, they can never send any message containing cluster information and operations.
The support of clustering is not mandatory and it is implementation dependent for all VRU profiles. If the clustering function is supported and activated, the basic operations the VBS needs to perform are the following [12]:

- Cluster identification: identification of the cluster by cluster participants themselves.
- Cluster creation: a cluster of VRUs is created including all the nearby VRU devices having similar speeds and headings.
- Cluster joining: it is the process through which a new VRU satisfying all the requirements related to VRU clusters is added to a cluster.
- Cluster leaving: when a VRU belonging to a cluster does not satisfy the conditions mentioned in chapter 2.1.3 anymore, it must be removed from the cluster.
- Cluster extension or shrinking: the operations of increasing or decreasing the size of a VRU cluster, both in terms of area or cardinality.
- Cluster breaking up: when the VRU cluster no longer participates in the traffic related scenario or when its cardinality drops below a predefined threshold it must be disbanded.

If a VRU cluster leader device decides to start leading another cluster, it must first break up its own, since each cluster leader can only lead one VRU cluster at a time. Similarly, cluster members can only be included in one VRU cluster at a time, so before joining a new cluster they must always first leave the previous one.
Regardless of whether the VRU clustering function is activated or not, the VRU Basic Service of a VRU ITS-S must always be in one of the following cluster states:

- VRU-IDLE: the user device is not considered as a VRU and therefore the VRU role as it was defined in chapter 2.5 corresponds to VRU_ROLE_OFF.
- VRU-ACTIVE-STANDALONE: this time the user device is considered as a VRU and individual VAMs are transmitted, which are VAMs containing only the information about the originating VRU ITS-S. When in this state, the VRU ITS-S may also reveal its intention to join a cluster or indicate that it has just left one.
- VRU-ACTIVE-CLUSTER-LEADER: when in this state, the VRU is the leader of a cluster and the originating ITS-S transmits VAMs including a special container with data elements specifically related to the cluster.

- VRU-PASSIVE: the user device is still considered as a VRU, however it is not allowed to transmit any message. This may happen when the VRU is a member of a cluster, without however being the leader of this latter, or when it is located in a low-risk geographical area.

We can now have a quick look at the main events related to cluster operation triggering a VBS state transition [12].
We have already said that, when the VBS is in VRU-IDLE state, the associated user device is not considered as a VRU and its role is VRU_ROLE_OFF. When the VBS in VRU-IDLE state realises that the user device has changed its role to VRU_ROLE_ON, it changes its state to VRU-ACTIVE-STANDALONE and starts transmitting VAMs.
When the VBS is in VRU-ACTIVE-STANDALONE state, it has three possibilities.
First of all, if the VBS realises that the VRU user device has changed its role to VRU_ROLE_OFF, it changes its state to VRU-IDLE and stops the transmission of VAMs.
The second possibility is instead related to the creation of a VRU cluster. If the VBS in VRU-ACTIVE-STANDALONE state determines, based on the VAMs it has received from other VRUs, that it can form as cluster, it shall perform the following actions:

1. Generate a random cluster identifier, which shall not be zero and shall be different from any cluster identifier included in any VAM received by the VBS in the last *timeClusterUniquenessThreshold* seconds, where *timeClusterUniquenessThreshold* is a predefined threshold.

2. Define an initial dimension for the cluster, in such a way to delimit its bounding box. To avoid false positives, the initial bounding box shall be such that it only includes the leader of the cluster.

3. Set the size of the VRU cluster to a predefined value called *minClusterSize*.

4. Change the VBS state to VRU-ACTIVE-CLUSTER-LEADER and start transmitting cluster VAMs.

For what concerns the third option, when the VBS of a user device is in VRU-ACTIVE-STANDALONE state and, on the basis of the information included inside the cluster VAMs received from a cluster leader, decides to join a cluster, this VRU shall transmit VAMs indicating its intention to join the cluster with a given identifier for an amount of time equal to *timeClusterJoinNotification*, which has a predefined value. The VRU shall also include in these VAMs the time at which it intends to join the cluster and, as a consequence, stop transmitting individual VAMs. Once the VRU has transmitted an appropriate number of cluster join notifications, it shall join the VRU cluster and therefore stop transmitting individual VAMs and start monitoring the cluster VAMs broadcast by the cluster leader. The VBS changes its state to VRU-PASSIVE.
It is however important to highlight that some problems may arise while a VRU tries to join a cluster. In particular, two things might happen:

1. The VBS, after having started the joining operation, determines that it will not join the cluster anymore, for a number of different reasons, therefore it stops including the cluster join notification in the individual VAMs it is transmitting and it

starts sending VAMs with the cluster leave notification for an amount of time corresponding to *timeClusterLeaveNotification*. The VBS remains in VRU-ACTIVE-STANDALONE state.

2. Once it has stopped sending individual VAMs, the VBS realises that the leader of the cluster has not updated the cluster state to include the new member or even that the cluster it wanted to join does not exist anymore. In this case, the VRU shall leave the cluster, which means that it shall remain in its VRU-ACTIVE-STANDALONE state and start transmitting individual VAMs again.

If a VRU device manages to successfully create a cluster, at some point the VBS in VRU-ACTIVE-CLUSTER-LEADER state might decide it is time to break up the cluster. In order to do this, it must include a VRU cluster operation field in its cluster VAMs specifying that it will break up the cluster it is leading, indicating also the VRU cluster's identifier and the reason why it wants to disband the cluster. Once the cluster leader has transmitted these indications in consecutive VAMs for an amount of time equal to *timeClusterBreakupWarning*, it shall stop broadcasting cluster VAMs and all VRUs in the cluster shall change their state to VRU-ACTIVE-STANDALONE and resume the transmission of individual VAMs. If instead a VRU managed to successfully join a cluster, two things might happen. First of all, if the VBS, on the basis of the information included in the messages received from the leader of the cluster, determines it is time to leave the cluster, it shall send consecutive VAMs for an amount of time equal to *timeClusterLeaveNotification* including the identifier of the cluster it is leaving and the reason why it wants to leave it. Once it has done this, the VBS changes its state from VRU-PASSIVE to VRU-ACTIVE-STANDALONE and resumes the transmission of individual VAMs. It is important to say that, in this case, a VRU that leaves a cluster shall use a different identifier from the one it used before it joined the cluster. The second possibility is related to a loss of the cluster leader. If indeed the VRU cluster leader loses connection or simply fails as a node, its VBS cannot send VAMs on behalf of the cluster anymore, which means that the members of this latter, as soon as they determine no cluster VAMs are transmitted by the leader of the cluster for a time corresponding to *timeClusterContinuity*, assume the cluster leader is lost and resume the transmission of individual VAMs, changing their state from VRU-PASSIVE to VRU-ACTIVE-STANDALONE.
We have now presented all the most important clustering events which can trigger a VBS state transition, however it is also fundamental to mention that there are some conditions that a VRU device with a VBS in VRU-ACTIVE-STANDALONE state must fulfill if it wants to create a cluster:

- It must have sufficient processing power.
- It must have been configured with VRU equipment type VRU-St.
- It is receiving VAMs from a number of different VRUs equal to *numCreateCluster* which are not further away than *maxClusterDistance*.
- It has not identified any cluster it could join.

Parameter	Type	Recommended value
numCreateCluster	Integer	[3,5]
maxClusterDistance	Distance (m)	[3,5]
minClusterSize	Integer	1
maxClusterSize	Integer	20
timeClusterUniquenessThreshold	Time (s)	30
timeClusterBreakupWarning	Time (s)	3
timeClusterJoinNotification	Time (s)	3
timeClusterContinuity	Time (s)	2
timeClusterLeaveNotification	Time (s)	1

Table 2.1. Parameters related to VRU clusters. Inspired by [12].

2.6 VRU Awareness Message

A VRU Awareness Message (VAM) is an ITS message containing all the status and attribute information related to the originating ITS-Station. The content of a VAM varies according to the profile of the VRU ITS-S, however typical status information include position, time, motion state, cluster status and similar data, while typical attribute information include data related to the VRU profile, its type, its dimensions and similar. When the receiving ITS-S receives a VAM, it becomes aware of the presence of the originating VRU ITS-S, of its type and of its status. At this point, the receiving ITS-S can use the received information to support several different VRU related ITS applications, for example, by making a comparison between its own status and the status of the originating VRU ITS-S, it can estimate the risk of having a collision with the originating ITS-Station.
We can therefore say that, in general, VAMs are messages used by VRU ITS-Stations to create and maintain awareness of vulnerable road users participating in the traffic scenario, which is why all types of road users, including infrastructure and vehicular ITS-Stations, shall be capable of receiving VAMs.

2.6.1 VAM Format

A VRU Awareness Message is made up of the following fields [12]:

- ITS PDU header
- Generation delta time
- Basic container, which includes the type of the originating ITS-S and its position
- VRU high frequency container, which includes the dynamic properties of the VRU, such as its motion characteristics, its speed, its acceleration, etc.

- VRU low frequency container, which includes the physical properties of the originating VRU
- Cluster information container
- Cluster operation container
- Motion prediction container

The first four fields, so the PDU header, the generation delta time, the basic container and the high frequency container are all mandatory, while the remaining containers are optional, depending on the nature and the operation of the originating VRU ITS-S.

Figure 2.8. VAM Format

We can now proceed by analysing in detail the aforementioned fields.

ITS PDU header and generation time

The ITS PDU header of a VAM is composed of three different fields [12]:

- the *protocolVersion* field, which, for the present version, shall be set to 3.
- the *messageID* field, which, for VAM, shall be set to 16.

- the *StationID* field, which does not have a predefined value, since each VRU will have its own *StationID*, however it must be locally unique. As it was previously mentioned in chapter 2.5.2, the *StationID* field must change when the VRU resumes transmitting individual VAMs after being part of a cluster. The only exception is if a VRU device fails to join a cluster, in this case it can keep using the *StationID* and the other identifiers it used before attempting to join the cluster.

The *GenerationDeltaTime* field represents the generation time of the VAM and it corresponds to the reference position time in the VAM, which is a measure of the number of milliseconds elapsed since ITS epoch, represented in modulo 2^{16} (65536).

Basic container

The basic container provides the following basic information about the originating VRU ITS-Station [12]:

- Station type of the originating ITS-S. It shall take the following values: 1 for pedestrians, 2 for bicyclists, 3 for mopeds, 4 for motorcycles, 12 for light VRU vehicles and 13 for animals. All the other values of *stationType* must not be used in the basic container.
- The latest geographical position of the originating VRU ITS-S the VBS was able to obtain at the time of VAM generation. This information provides both the value and the confidence of the measured position of the VRU, which shall correspond to the ground position of the centre of the front side of the bounding box of this latter, and the measurement time shall correspond to the generation delta time.

The basic container is a mandatory field of any VRU Awareness Message, so it must always be present for all VAMs generated by the ITS-Stations implementing the VBS.

VRU specific containers

The first VRU specific container we need to discuss is the VRU High Frequency container (VRU HF), which must be included in all VAMs generated by a VRU ITS-S. This container includes all the potentially fast-changing status information related to the originating VRU ITS-S, such as its heading or its speed. In particular, the VRU HF is made up of 3 mandatory fields, which are the VRU heading, its speed and its acceleration, specifically the longitudinal one, plus a number of optional fields, which presence depends on the type of VRU and on its operation. The optional information are the lane position of the VRU, its curvature value, its yaw rate, its lateral acceleration, its orientation, its roll angle, the VRU device usage and the movement control, which indicates the mechanism used to control the longitudinal movement of the VRU.
Another important VRU specific container is the Low Frequency container (VRU LF), which contains all the potentially slow-changing information related to the VRU ITS-S. The VRU LF shall be characterised by a customisable frequency and it has the following content [12]:

- *profileAndSubprofile* field: it provides information about the profile and sub-profile of the originating VRU ITS-S and it is mandatory whenever the VRU LF is present. Specifically, the profile identifies to which of the four types of VRU profiles described in chapter 2.1.2 the originating VRU ITS-S belongs to and the sub-profile gives a further specification of the type of VRU we are dealing with.

- *sizeClass* field: it contains information related to the size of the VRU, which, together with the profile type, helps other road users retrieve the range of dimensions of the VRU.

- *exteriorLights* field: it provides the status of the main exterior lights switches of the originating VRU ITS-S.

The *sizeClass* and *exteriorLights* fields are not mandatory information to include in the VRU LF, their presence typically depends on the VRU profile.
The VRU cluster containers, on the other hand, contain all the information and operations related to VRU clusters. There are two different types of cluster containers, the VRU cluster information container and the VRU cluster operation container, which differ in the characteristics of the included data.
The cluster information container is present in the VAMs disseminated from the leader of a cluster and it provides information relevant to the VRU cluster, such as its ID, called *clusterID*, the shape of its bounding box, referred to as *ClusterBoundingBoxShape*, its cardinality and the profiles of the VRUs present in the cluster (*clusterProfiles*). The ID of the cluster must always be different from zero and it must be locally unique. The shape of the bounding box, on the other hand, can be circular, rectangular or polygon.
The VRU cluster operation container contains all the information related to a change of cluster state and composition and we can find this container included in VAMs transmitted by either a leader or a member of a cluster. This container provides the following information [12]:

- *clusterJoinInfo*, when a new member joins the VRU cluster. The new member of the cluster must specify the ID of the cluster it intends to join, which is the same as the *clusterID* component of the cluster information container, and the *joinTime*, which is the time after which the VRU that is joining the cluster will stop transmitting individual VAMs.

- *clusterLeaveInfo*, when a member of an existing cluster decides to leave the VRU cluster. This VRU must specify the ID of the cluster it is leaving, which again corresponds to the *clusterID* field of the cluster information container, and the *clusterLeaveReason*, which is the reason why it is leaving the cluster.

- *clusterBreakupInfo*, transmitted by the leader of a cluster when it decides to disband the VRU cluster it is leading. The cluster leader, in this case, must specify the *clusterBreakupReason*, which is the reason why it intends to disband the cluster, and the *breakupTime*, which indicates the time after which it will stop broadcasting cluster VAMs.

- *clusterIdChangeTimeInfo*, used by the cluster leader to indicate that it is planning to change the cluster ID at a given time, which must always be specified.

The last VRU specific container we need to mention is the Motion Prediction Container, which contains information related to the past and future motion state of the VRU. In particular, this container shall contain information related to the past locations of the VRU, which are data of type *PathHistory*, and to its predicted future locations, indications related to safe distances between the VRU and other road users, which are information of type *SequenceOfSafeDistanceIndication*, and to possible trajectory interceptions of the VRU with other VRUs or different types of road users, which shall be data of type *SequenceOfTrajectoryInterceptionIndication*, and data of type *AccelerationChangeIndication*, which represent changes in the acceleration of the VRU, *HeadingChangeIndication*, used to communicate heading changes, and *StabilityChangeIndication*, which refer to changes in the stability of the VRU [12].

2.6.2 VAM dissemination

VAM transmission is typically a point-to-multipoint communication. This type of communication topology consists of a central base station that supports several subscriber stations, thus offering network access from one single location to multiple locations, allowing them to share the same network resources. [5]
We call VAM generation event the event resulting in the generation of one VAM. The minimum time that needs to elapse between the start of two consecutive VAM generation events must be equal to or larger than a parameter called T_GenVam. T_GenVam shall always be included in the range $[T\_GenVamMin, T\_GenVamMax]$, where $T\_GenVamMin$ and $T\_GenVamMax$ have two predefined values, corresponding respectively to 100 ms and 5 s. If a cluster VAM is broadcast, its T_GenVam could be smaller than that of an individual VAM. The only exception to this rule is given by two consecutive VAMs containing low frequency containers, as, in this case, $T\_GenVamMin$ must not be smaller than to 2 s. This happens because, as it was described in detail in chapter 2.6.1, the VRU low frequency container contains slow changing attribute information about the originating VRU ITS-S, which means that there is no need to have a very high transmission frequency and the reduction of this latter contributes to bandwidth efficiency and power saving.
It is also important to mention that the time required for generating a new VAM shall always be less than $T\_AssembleVAM$, whose value corresponds to 50 ms.
We have highlighted that the minimum time elapsed between two consecutive VAMs carrying the low frequency container must be bigger than 2 s and this latter is included in a VAM either on a periodical basis or when the VRU cluster container is present. Typically, the VRU low frequency container, even if its presence is not mandatory, shall be included in the first VAM transmitted since the activation of the VBS and then it shall be included every time the time elapsed since the last VAM generation with low frequency container is equal to or greater than 2 s.
The VRU Basic Service shall check the conditions for triggering a new VAM generation every $T\_CheckVamGen$, which must be equal to or lower than $T\_GenVamMin$.

For what concerns the control of the parameter T_GenVam, its value shall be provided in the unit of milliseconds by the VBS management entity. Since we know that T_GenVam's value must always be in the range [$T\_GenVamMin$, $T\_GenVamMax$], if the value provided by the management entity is greater than $T\_GenVamMax$, T_GenVam shall be set to $T\_GenVamMax$, while, if the management entity provides a value below $T\_GenVamMin$ or if it doesn't provide any value, T_GenVam shall be set to $T\_GenVamMin$.
In case of ETSI ITS-G5, it shall be the Decentralized Congestion Control (DCC) to manage T_GenVam according to the channel usage requirements.
As it was previously mentioned in chapter 2.5.2, VRU ITS-Stations shall send individual VAMs when in VRU-ACTIVE-STANDALONE state, while, when a VRU is the leader of a cluster and therefore its VBS state is VRU-ACTIVE-CLUSTER-LEADER, it shall broadcast cluster VAMs on behalf of the cluster. When a VRU is a member of a cluster, its VBS state is VRU-PASSIVE, therefore it shall only send individual VAMs containing the VRU cluster operation container while leaving the cluster. Similarly, whenever a VRU in VRU-ACTIVE-STANDALONE state wants to join a cluster, it shall transmit individual VAMs containing the VRU Cluster Operation Container. [12]

Triggering conditions

It is very important to distinguish between individual and cluster VAMs, since the VRU Basic Service of the originating ITS-S manages them in two different ways.
For what concerns individual VAMs, the first time they shall be generated is immediately after the activation of the VRU Basic Service. After the first transmission, a new individual VAM shall be generated at the earliest time instant possible whenever one of the following conditions is satisfied and if the transmission is not subject to redundancy mitigation [12]:

1. A VRU changes its VBS state from VRU-IDLE to VRU-ACTIVE-STANDALONE.

2. A member of a VRU in VRU-PASSIVE state decides to leave the cluster and enters VRU-ACTIVE-STANDALONE state.

3. A member of a VRU cluster in VRU-PASSIVE state determines that the VRU cluster leader is lost and decides to enter VRU-ACTIVE-STANDALONE state.

4. The leader of a VRU cluster in VRU-ACTIVE-CLUSTER-LEADER state decides to break up the cluster, so it transmits one last cluster VAM containing the disband indications and then it enters VRU-ACTIVE-STANDALONE state.

Once the VBS of a VRU ITS-S is in VRU-ACTIVE-STANDALONE state, the VRU ITS-S keeps broadcasting individual VAMs until the state of its VRU Basic Service changes. In particular, the generation of consecutive individual VAMs shall occur at an interval equal to or larger than T_GenVam, whose range of possible values has already been described. A new individual VAM shall therefore be generated and transmitted

if and only if the VBS of the originating VRU ITS-Station is still in VRU-ACTIVE-STANDALONE state, the transmission is not subject to redundancy mitigation and at least one of the following conditions is satisfied [12]:

1. The time elapsed since the last individual VAM transmission exceeds *T_GenVamMax*.

2. The Euclidean absolute distance between the current estimated position of the VRU and the the estimated position included in the last individual VAM transmitted is larger than a predefined threshold, called *minReferencePointPositionChangeThreshold*.

3. The absolute difference between the current estimated speed of the VRU and the estimated speed of the VRU included in the last individual VAM transmitted is larger than a predefined threshold called *minGroundSpeedChangeThreshold*.

4. The absolute difference between the orientation of the current estimated velocity vector of the VRU and the orientation of the velocity vector included in the last individual VAM transmitted is larger than a predefined threshold called *minGroundVelocityOrientationChangeThreshold*.

5. The absolute difference between the current estimated trajectory interception probability with other VRUs or vehicles and the trajectory interception probability with other VRUs or vehicles included in the last individual VAM transmitted is larger than a predefined threshold called *minTrajectoryInterceptionProbChangeThreshold*.

6. The originating VRU ITS-S is in VRU-ACTIVE-STANDALONE state and decides to join a VRU cluster.

7. The VRU ITS-S, after its last individual VAM transmission, has determined that one or more VRUs or vehicles have satisfied the following conditions simultaneously:

   - their absolute lateral distance is lower than a predefined threshold called Minimum Safe Lateral Distance (MSLaD)
   - their absolute longitudinal distance is lower than a predefined threshold called Minimum Safe Longitudinal Distance (MSLoD)
   - their absolute vertical distance is lower than a predefined threshold called Minimum Safe Vertical Distance (MSVD)

For what concerns, on the other hand, cluster VAMs, the first time a VRU shall generate them is immediately after the activation of its VRU Basic Service. After the first transmission, a new cluster VAM shall also be generated and transmitted at the earliest time possible whenever at least one of the following conditions is satisfied and if the transmission of cluster VAMs is not subject to redundancy mitigation [12]:

1. The time elapsed since the last cluster VAM transmission is larger than *T_GenVamMax*.

2. The Euclidean absolute distance between the current estimated reference position of the VRU cluster and the the estimated reference position included in the last cluster VAM transmitted is larger than a predefined threshold, called *minReferencePointPositionChangeThreshold*.

3. The absolute difference between the current estimated distance from the boundary of the cluster and the same distance based on the last cluster VAM transmitted is larger than a predefined threshold, called *minClusterDistanceChangeThreshold*

4. The difference between the current estimated reference speed of the VRU cluster and the estimated reference speed of the VRU cluster included in the last cluster VAM transmitted is larger than a predefined threshold called *minGroundSpeedChangeThreshold*.

5. The difference between the orientation of the current estimated reference velocity vector of the VRU cluster and the orientation of the reference velocity vector included in the last cluster VAM transmitted is larger than a predefined threshold called *minGroundVelocityOrientationChangeThreshold*.

6. The difference between the current estimated trajectory interception probability with other VRUs or vehicles and the trajectory interception probability with other VRUs or vehicles included in the last cluster VAM transmitted is larger than a predefined threshold called *minTrajectoryInterceptionProbChangeThreshold*.

7. The type of the VRU cluster, which can be either homogeneous or heterogeneous, has changed after the last cluster VAM transmission.

8. The leader of the cluster has decided to break up the cluster after last cluster VAM transmission.

9. After previous cluster VAM transmission, more than a predefined number of new VRUs has joined the cluster.

10. After previous cluster VAM transmission, more than a predefined number of members of the VRU cluster has decided to leave this latter.

11. The leader of the VRU cluster, after its last cluster VAM transmission, has determined that one or more non-member VRUs or vehicles have satisfied the following conditions simultaneously:

    - their absolute lateral distance from the reference point of the cluster is lower than a predefined threshold called Minimum Safe Lateral Distance (MSLaD)
    - their absolute longitudinal distance from the reference point of the cluster is lower than a predefined threshold called Minimum Safe Longitudinal Distance (MSLoD)
    - their absolute vertical distance from the reference point of the cluster is lower than a predefined threshold called Minimum Safe Vertical Distance (MSVD)

VAM Redundancy Mitigation

It is very important to find a good balance between the frequency of VAM generation and transmission and the occupancy of the communication channel, without impacting the awareness and safety of VRUs. When a VAM generation event occurs, the originating VRU ITS-S shall skip the transmission of a new individual VAM *numSkipVamsForRedundancyMitigation* if the following conditions are simultaneously satisfied [12]:

1. The time elapsed since the last individual VAM transmission doesn't exceed *numSkipVamsForRedundancyMitigation* $*$ *T_GenVamMax*.
2. The Euclidean absolute distance between the current estimated position of the originating VRU ITS-S and the estimated position included in the VAM received from another VRU ITS-S is lower than a predefined threshold called *minReferencePointPositionChangeThreshold*.
3. The absolute difference between the current estimated speed of the originating VRU ITS-S and the estimated speed included in the VAM received from another VRU ITS-S is lower than a predefined threshold called *minGroundSpeedChangeThreshold*.
4. The absolute difference between the orientation of the current estimated velocity vector of the originating VRU ITS-S and the orientation of the estimated velocity vector included in the VAM received from another VRU ITS-S is lower than a predefined threshold called *minGroundVelocityOrientationChangeThreshold*

The transmission of a new individual VAM shall also be skipped if at least one of the following conditions is satisfied [12]:

- The VRU determines that it is located inside a protected or non-drivable area.
- The VRU is located inside a pedestrian only geographical area.
- The VRU is a member of a cluster and the cluster leader has not transmitted the cluster break up message yet.
- Some other ITS-S has reported the information about the ego-VRU within *T_GenVam*.

Parameter	**Type**	**Recommended range**
minReferencePointPositionChangeThreshold	Distance (m)	4
minGroundSpeedChangeThreshold	Speed (m/s)	± 0.5
minGroundVelocityOrientationChangeThreshold	Orientation (degrees)	± 4
minTrajectoryInterceptionProbChangeThreshold	Probability (%)	10
numSkipVamsForRedundancyMitigation	Number of times	[2,10]
minClusterDistanceChangeThreshold	Length (m)	2
minimumSafeLateralDistance	Length (m)	Max $[2, A]$, where A is the maximum lateral distance the VRU can travel in $T\_GenVamMax$ seconds.
minimumSafeLongitudinalDistance	Length (m)	B, which is the maximum longitudinal distance the VRU can travel in $T\_GenVamMax$ seconds.
miniumSafeVerticalDistance	Length (m)	5

Table 2.2. Parameters related to VAM triggering conditions. Inspired by [12].

Chapter 3

Implementation on ms-van3t

The aim of this chapter is to describe how the communication protocol explained in detail in the previous section was implemented on ms-van3t, a simulator for vehicular networks developed by Politecnico di Torino.
First of all, we need to specify that the communication standard described in chapter 2 has not been fully implemented on ms-van3t, meaning that some of its features are not currently supported by this latter and their implementation is left to further study. In particular, for what concerns the communication protocol itself, it was decided not to implement the VRU clustering functionality, in order to make the code a bit easier to develop and maintain and the data retrieved from the simulations easier to interpret.
It is also worth mentioning that ms-van3t currently supports only VRUs belonging to VRU profile 1, which are, in other words, pedestrians. This decision was taken since pedestrians are the most common type of vulnerable road users we can find on the road and also the ones subjected to the greatest risk in case of collision, so, for what concerns the testing of the communication protocol, both on a simulator and in a real world environment, pedestrians are the VRUs of greatest interest. Other than this, by only implementing one profile of VRUs, we could make the simulated scenario a bit easier to build and maintain, which turns out to be very useful especially when it comes to retrieving and analysing the results of the simulation, which are used to test the effectiveness of the communication protocol under study.

3.1 ms-van3t

ms-van3t is an open source simulator and emulator for vehicular networks designed and developed by Politecnico di Torino. It includes a full implementation of ETSI ITS-G5 for V2X simulations and its purpose is to orchestrate the interaction between two other simulation softwares, which are ns-3, a network simulator, and SUMO, a road traffic simulator. Other than orchestrating the interaction between these two simulators, ms-van3t also provides additional ns-3 modules, which are used to build and simulate V2X applications using the two aforementioned softwares themselves [1]:

- *gps-tc*, which is a module used to leverage GNSS traces to model the mobility of vehicles.

- *automotive*, which consists in a full implementation of the ETSI ITS-G5 stack, allowing vehicles and other road users to exchange CAMs and DENMs. The functionality of this module has been extended in order to also support the transmission and reception of VAMs.
- *prr-supervisor*, which is a module used for the collection of metrics, interfacing with SUMO and ns-3 in order to return accurate measurements of the one-way latency and the packet reception ratio (PRR) from the simulated scenario.

ms-van3t allows to easily switch between the models of different communication technologies and it also allows to remove, if needed, SUMO from the loop and use real GPS traces to manage the mobility of implemented road users instead.
At the moment, ms-van3t supports three different access models [6]:

- 802.11p, used for V2V, V2P and V2I communications.
- LTE for V2N communications.
- C-V2X for V2V communications and possibly, in future releases, also for V2P communications.

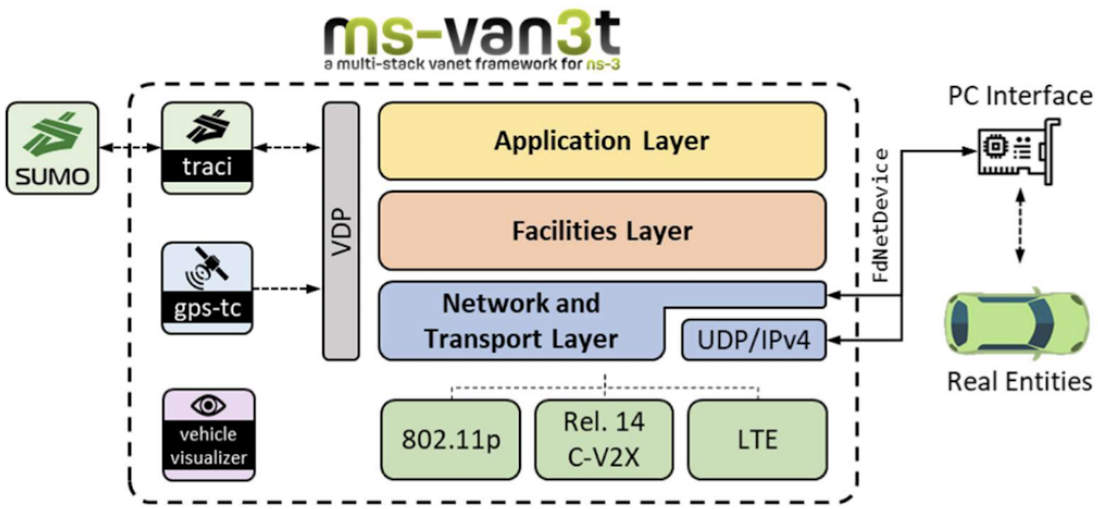

Figure 3.1. ms-van3t architecture. Taken from [1].

3.1.1 ns-3

ns-3 is a discrete-event network simulator based on event scheduling paradigm made up of a simulation core and several models, mainly implemented in C++ [2]. It is an open-source project mainly intended for educational and research purposes.

ns-3's main purpose is to provide users a simulation engine which can be used to conduct simulation experiments and it is aimed at providing several models showing how packet data networks work and what their performances are. The model set focuses on modeling Internet protocols and networks, however ns-3's users can also use it to model non-Internet-based systems [3].
This network simulator is built as a set of libraries which can be linked together as well as with external software libraries and they can also be either statically or dynamically linked to a main program written in C++, which is used to start the simulation and to define its topology. On top of that, it is also possible to export almost all ns-3 APIs to Python, in such a way that Python programs can import ns-3 modules in almost the same way as ns-3 libraries are linked together in C++.
For what concerns its source code, it is manly located inside the *src* directory and, as it was previously mentioned, it is organised in modules. The two lowest modules are the core and network ones, that, together, build a simulation core which can be used by different kinds of networks, Internet-based and non-Internet based. ns-3 is also characterised by specific modules supplementing the core APIs which allow ns-3 programs to access all APIs directly or by using the helper API, whose function is to create a wrapper of low-level API calls. It is important to highlight that each module's functionalities only depend on the functionalities of the modules beneath it. [2]
All functions to send and receive packets and to configure the network are very similar to the ones a real-world Linux system uses and the simulations are written directly in C++ files [1].

3.1.2 SUMO

SUMO, which stands for "Simulation of Urban Mbility", is an open source traffic simulation package which can be used for time-discrete microsimulations and which is designed to handle large networks. It allows to simulate all types of road users, including pedestrians and, more in general, VRUs, and it provides a lot of different tools to be used for the creation of traffic scenarios.
SUMO allows to simulate how a given traffic demand, made up of single vehicles and other types of road users, moves through a certain road network and the simulation is purely microscopic, which means that each traffic participant has its own route and moves individually across the network. It is also important to highlight that simulations are completely deterministic.
SUMO includes a lot of different applications which are used to prepare first and then perform traffic simulations. These simulations are characterised by road users having space-continuous and time-discrete movements and, other than different types of road users, it is also possible to implement streets with multiple lanes and with lane changing. It is also possible to implement different right-of-way rules and it makes use of real maps from the Geodatabase of OpenStreetMap or of maps directly available in ms-van3t. [19]
In order to couple SUMO and ns-3 functionalities, the TraCI interface was used. [6]

3.1.3 TraCI

TraCI stands for Traffic Control Interface and it serves as interface between the network simulator ns-3 and the SUMO mobility simulator. In particular, TraCI allows to retrieve the values of the simulated objects and to manipulate their behaviour by using an API that treats the SUMO simulation as a server.
In order to provide access to SUMO, TraCI uses a TCP based client/server architecture, which means that SUMO acts as a server that is started with specific additional command line options and, once it has been launched, SUMO's only purpose is to prepare the simulation and then it waits for the external applications to connect and take over the control. It is important to highlight that, when SUMO is running as a TraCI server, it keeps running until the client demands the simulation to end.
SUMO can also work with multiple clients, however, in multiple clients scenarios, it is important to always specify the order of execution of the clients. Each client must indeed specify a unique integer value and, within each simulation step, the commands of the clients will be handled starting from the lowest to the highest value. The simulation does not advance to the next step until all clients have been properly synchronised. [20]
ms-van3t includes a full C++ implementation of TraCI to bridge ns-3 and SUMO.

3.2 VRU Basic Service

As it was explained in chapter 2.5, the purpose of the VRU Basic Service is to manage the transmission and reception of VAMs.
In order to implement the VBS on ms-van3t, a specific class, called *VRUBasicService*, has been created. This class is made up of several functions which are used to implement all the functionalities of the VBS related to the transmission and reception of individual VAMs, since, as it was previously mentioned, VRU clusters are not currently supported. First of all, even if, in our case, the role of the VRUs will always be VRU_ROLE_ON and the clustering function is not supported, it was decided to implement a state machine, whose state evolution, for the aforementioned reasons, is defined a priori, which is used to constantly keep track of the role and VBS state of all the VRUs taking part to the simulation, in such a way that, if, in the future, the support to VRU clusters will be added, the state machine allowing this implementation will be already present.
The VBS also implements a number of functions which are used to set the *stationType* and the *stationID* of the ego-VRU both in the VBS itself and in the BTP (Basic Transport Protocol), which is the transport layer protocol that ETSI defines for ITS-G5. In order to access to the BTP, a specific pointer, called *m_btp*, is used. At this point we need to talk specifically about how the generation, transmission and reception of VAMs are handled by the VRU Basic Service.

3.2.1 Generation and encoding of VAMs

The VBS handles the generation and encoding of VAMs through the *generateAndEncodeVam* function, which is used first to generate the VAM by filling its mandatory fields with all the needed information and then to encode the VAM and send it to the BTP,

which will then handle its dissemination.
For what concerns the generation of VAMs, the *generateAndEncodeVam* function first fills the ITS PDU header of the VAM with its message ID, which corresponds to 16, the protocol version, which currently corresponds to 3, and the station ID of the VRU ITS-S which wants to transmit the message (lines 2 to 4 of the snippet) and the generation delta time field with the timestamp, wrapped to 65536, of the VAM generation (line 7 of the snippet).
At this point, we need to retrieve the data to be included in the two mandatory containers of a VAM, which are the basic container and the high frequency container. This is done through a specific function implemented in the VRU data provider, which will be discussed later, called *getVAMMandatoryData*, which returns all the mandatory fields that must always be included in the aforementioned containers (line 12 of the snippet).
It was decided to only implement the mandatory containers and not the optional ones, since VRU clusters are not supported and no motion prediction was carried out during the research activity.

```
1  /* Fill the header */
2  asn1cpp::setField(vam->header.messageID, FIX_VAMID);
3  asn1cpp::setField(vam->header.protocolVersion , 3);
4  asn1cpp::setField(vam->header.stationID, m_station_id);
5
6  /* Compute the generationDeltaTime */
7  asn1cpp::setField(vam->vam.generationDeltaTime, compute_timestampIts (
     m_real_time) % 65536);
8
9  /* Fill the basicContainer's station type */
10 asn1cpp::setField(vam->vam.vamParameters.basicContainer.stationType,
     m_stationtype);
11
12 vam_mandatory_data = m_VRUdp->getVAMMandatoryData();
13
14 /* Fill the basicContainer */
15 asn1cpp::setField(vam->vam.vamParameters.basicContainer.
     referencePosition.altitude.altitudeValue, vam_mandatory_data.altitude.
     getValue ());
16 asn1cpp::setField(vam->vam.vamParameters.basicContainer.
     referencePosition.altitude.altitudeConfidence, vam_mandatory_data.
     altitude.getConfidence ());
17 asn1cpp::setField(vam->vam.vamParameters.basicContainer.
     referencePosition.latitude, vam_mandatory_data.latitude);
18 asn1cpp::setField(vam->vam.vamParameters.basicContainer.
     referencePosition.longitude, vam_mandatory_data.longitude);
19 asn1cpp::setField(vam->vam.vamParameters.basicContainer.
     referencePosition.positionConfidenceEllipse.semiMajorConfidence,
     vam_mandatory_data.posConfidenceEllipse.semiMajorConfidence);
20 asn1cpp::setField(vam->vam.vamParameters.basicContainer.
     referencePosition.positionConfidenceEllipse.semiMinorConfidence,
     vam_mandatory_data.posConfidenceEllipse.semiMinorConfidence);
21 asn1cpp::setField(vam->vam.vamParameters.basicContainer.
     referencePosition.positionConfidenceEllipse.semiMajorOrientation,
     vam_mandatory_data.posConfidenceEllipse.semiMajorOrientation);
```

```
22
23  /* Fill the highFrequencyContainer */
24  asn1cpp::setField(vam->vam.vamParameters.vruHighFrequencyContainer.
      heading.value, vam_mandatory_data.heading.getValue ());
25  asn1cpp::setField(vam->vam.vamParameters.vruHighFrequencyContainer.
      heading.confidence, vam_mandatory_data.heading.getConfidence ());
26  asn1cpp::setField(vam->vam.vamParameters.vruHighFrequencyContainer.speed
      .speedValue, vam_mandatory_data.speed.getValue ());
27  asn1cpp::setField(vam->vam.vamParameters.vruHighFrequencyContainer.speed
      .speedConfidence, vam_mandatory_data.speed.getConfidence ());
28  asn1cpp::setField(vam->vam.vamParameters.vruHighFrequencyContainer.
      longitudinalAcceleration.longitudinalAccelerationValue,
29  vam_mandatory_data.longAcceleration.getValue ());
30  asn1cpp::setField(vam->vam.vamParameters.vruHighFrequencyContainer.
      longitudinalAcceleration.longitudinalAccelerationConfidence,
31  vam_mandatory_data.longAcceleration.getConfidence ());
```

Listing 3.1. VAM generation

At this point, once the VAM to be broadcast has been generated, the current position, speed and heading of the originating VRU ITS-S are stored for future use, which will be explained later, and the VAM is encoded and passed to the BTP which will handle its dissemination.

```
1   /* VAM encoding */
2   std::string encode_result = asn1cpp::uper::encode(vam);
3
4   if(encode_result.size()<1)
5   {
6     return VAM_ASN1_UPER_ENC_ERROR;
7   }
8
9   packet = Create<Packet> ((uint8_t*) encode_result.c_str(), encode_result
      .size());
10
11  dataRequest.BTPType = BTP_B; //!< BTP-B
12  dataRequest.destPort = VA_PORT;
13  dataRequest.destPInfo = 0;
14  dataRequest.GNType = TSB;
15  dataRequest.GNCommProfile = UNSPECIFIED;
16  dataRequest.GNRepInt =0;
17  dataRequest.GNMaxRepInt=0;
18  dataRequest.GNMaxLife = 1;
19  dataRequest.GNMaxHL = 1;
20  dataRequest.GNTraClass = 0x02; // Store carry foward: no - Channel
      offload: no - Traffic Class ID: 2
21  dataRequest.lenght = packet->GetSize ();
22  dataRequest.data = packet;
23  m_btp->sendBTP(dataRequest);
```

Listing 3.2. VAM encoding

3.2.2 Transmission of VAMs

As it was described in chapter 2.6.2, the minimum time which has to elapse between two consecutive VAM generation events corresponds to *T_GenVam*, which is a parameter ranging from 100 ms up to 5 s, if the VRU low frequency container was not present in the last VAM transmitted.
In our case, VAM dissemination is started by a function in the VBS called *startVamDissemination*, which is called directly by the application, whose duty is to schedule immediately after or after a random time interval the execution of the *initDissemination* function, which is the one actually starting the broadcast of VAMs.
As soon as the dissemination of VAMs is started, the *checkVamConditions* function is scheduled to be executed every 100 ms and its task is to check if any of the triggering conditions described in chapter 2.6.2 has been satisfied since the last VAM transmission and, in case this has happened, to trigger a new VAM transmission by calling the *generateAndEncodeVam* function, which was previously described. Not all seven triggering conditions have been implemented on ms-van3t, in particular condition 5, related to trajectory interception probability, was not implemented since no motion prediction container is present in the simulated VAMs and condition 6, related to VRU clusters, is not checked since the VRU clustering functionality is not currently supported. All 5 other triggering conditions are fully implemented and functioning and the following piece of code shows how one of them, in particular the one related to speed change, is specified on ms-van3t.

```
1  /* 1c)
2   * The absolute difference between the current speed of the originating
     ITS-S
3   * and the speed included in the VAM previously transmitted by the
     originating
4   * ITS-S exceeds 0,5 m/s.
5  */
6  double speed_diff = m_VRUdp->getPedSpeedValue () - m_prev_speed;
7  if (!condition_verified && (speed_diff > 0.5 || speed_diff < -0.5))
8    {
9      if(!redundancy_mitigation && (m_N_GenVam_red==0 || m_N_GenVam_red==
     m_N_GenVam_max_red)){
10           m_N_GenVam_red = 0;
11
12           m_trigg_cond = SPEED_CHANGE;
13           vam_error = generateAndEncodeVam ();
14           if(vam_error==VAM_NO_ERROR)
15             {
16               condition_verified = true;
17             } else {
18               NS_LOG_ERROR("Cannot generate VAM. Error code: "<<vam_error)
     ;
19             }
20       } else{
21         m_N_GenVam_red++;
22         condition_verified = true;
23       }
```

```
24      }
```

Listing 3.3. Speed change VAM triggering condition

As we can notice from the previous piece of code, the *checkVamConditions* function must also implement the VAM redundancy mitigation technique, which is used to reduce the occupancy of the communication channel without impacting the safety and awareness of VRUs.
As it was explained in chapter 2.6.2, VAM redundancy mitigation takes place whenever the ego-VRU receives a VAM from another VRU such that the position, speed and heading differences between the two are below given thresholds.
In order to implement VAM redundancy mitigation, a specific function, called *checkVamRedundancyMitigation*, is provided and its task is to check if there is any nearby VRU which satisfies the aforementioned conditions. In order to do this, the *checkVamRedundancyMitigation* function scans the Local Dynamic Map of the ego-VRU, which contains the information retrieved from the ITS messages received from nearby road users, looking for any VRU whose position, speed and heading are similar to the ones of the ego-VRU. In particular, the function, trough dedicated methods, selects all the VRUs stored in the LDM whose distance from the ego-VRU is lower than *minReferencePointPositionChangeThreshold* (line 2 of the snippet), whose value was defined in table 2.2. At this point, the selected VRUs are further examined, looking for any VRU ITS-S whose speed difference with respect to the speed of the ego-VRU is lower than *minGroundSpeedChangeThreshold* and whose heading difference with respect to the heading of the ego-VRU is lower than *minGroundVelocityOrientationThreshold* (lines 6 to 11 of the snippet), whose values are again defined in table 2.2.
If the analysis performed by the aforementioned function gives as result that VAM redundancy mitigation has to be performed, each time one of the five implemented triggering conditions is verified, the current individual VAM transmission is skipped for *numSkipVamsForRedundancyMitigation*.
The main problem with VAM redundancy mitigation is that it might happen that two or more VRUs start performing it at almost the same time. In particular, this happens every time there are two or more VRUs with similar position, heading and speed that are exchanging VAMs. In this case, all VRUs involved stop disseminating VAMs at almost the same time and they all skip *numSkipVamsForRedundancyMitigation* transmissions, which means that, in the worst case, all these VRUs stay silent for *numSkipVamsForRedundancyMitigation* $*$ *T_GenVamMax* seconds, where, as specified in chapter 2.6.2, *T_GenVamMax* corresponds to 5 seconds. Unfortunately this problem cannot be erased, however, in order to at least mitigate it, it was decided to assign the parameter *numSkipVamsForRedundancyMitigation* a new random integer value in the [2,10] interval every time a VRU starts performing VAM redundancy mitigation (line 13 of the snippet), in such a way that the redundancy mitigations performed by these VRUs are at least desynchronized.

```
1       if(now-lastVamGen < m_N_GenVam_max_red*5000){
2         m_LDM->rangeSelect (4,ped_pos.lat,ped_pos.lon,selectedStations);
3
```

```
4       for(std::vector<LDM::returnedVehicleData_t>::iterator it =
    selectedStations.begin (); it!=selectedStations.end () && !
    redundancy_mitigation; ++it){
5           if(it->vehData.stationType == StationType_pedestrian){
6               double speed_diff = it->vehData.speed_ms - ped_speed;
7               double near_VRU_heading = it->vehData.heading;
8               near_VRU_heading += (near_VRU_heading>180.0) ? -360.0 : (
    near_VRU_heading<-180.0) ? 360.0 : 0.0;
9               double heading_diff = near_VRU_heading - ped_heading;
10
11              if((speed_diff < 0.5 && speed_diff > -0.5) && (heading_diff
    < 4 && heading_diff > -4)){
12                  redundancy_mitigation = true;
13                  m_N_GenVam_max_red = (int16_t)std::round(((double)std::
    rand()/RAND_MAX)*8) + 2;
14              }
15          }
16      }
17  }
```

Listing 3.4. VAM redundancy mitigation check

3.2.3 Reception and decoding of VAMs

When a VRU ITS-S receives a VAM, it must decode the received message in order to extract all the relevant data from this latter and this is done by the *receiveVam* function. Once the received VAM has been properly decoded (line 2 of the snippet), the extracted data are then stored inside the LDM of the ego-VRU through a specific function called *vLDM_handler* (line 11 of the snippet), whose task is to update the LDM with the new data included in the received VAM. If the LDM of the ego-VRU already contains information about the VRU ITS-S which transmitted the received message, then the data already included in the LDM will be updated with the new ones which have been just received, otherwise a new field will be created inside the LDM to store the received information. At this point, once the LDM has been suitably updated, the *receiveVam* function will call a callback function called *m_VAMReceiveCallback* (line 15 of the snippet), which is defined in the application and then passed to the VRU Basic Service and it is called every time a new VAM is received by a certain node. Its purpose is to simplify the handling of the received VAMs.

```
1   /** Decoding **/
2   decoded_vam = asn1cpp::uper::decode(packetContent, VAM);
3
4   if(bool(decoded_vam)==false) {
5       NS_LOG_ERROR("Warning: unable to decode a received VAM.");
6       return;
7   }
8
9   if(m_LDM != NULL){
10      //Update LDM
11      vLDM_handler(decoded_vam);
12  }
```

```
13
14      if(m_VAMReceiveCallback!=nullptr) {
15          m_VAMReceiveCallback(decoded_vam,from);
16      }
```

Listing 3.5. Reception of VAMs

3.3 VRU data provider

ETSI does not foresee any data provider for filling the fields of the ITS messages transmitted by VRUs, unlike what it happens with vehicles, for which ETSI foresees the presence of a vehicle data provider in order to carry out the aforementioned function.
The data to be included in a VAM to be transmitted could also be retrieved directly within the *generateAndEncodeVam* function inside the VRU Basic Service, though it was decided to implement a data provider also for VRUs, called VRU data provider, even if it is not mentioned in ETSI standards, in order to make the acquisition of the data to be included in the VAM format simpler.
In particular, the VRU data provider implements some simple functions which can be used to fill the different fields of a VAM. At the moment, the implemented functions are meant to provide all the necessary data to only fill the mandatory fields of a VAM, since, as it was stated earlier, only the mandatory containers are currently implemented on ms-van3t, however the functionalities of the VRU data provider could be easily extended in the future to also provide the necessary data to fill the non-mandatory fields of VAMs.
The most important function present inside the VRU data provider class is the so-called *getVAMMandatoryData*, which, as the name suggests, is meant to provide all the information which must be included inside the mandatory fields of a VAM. In particular, the aforementioned function returns a custom type structure containing most of the mandatory data to be included in a VAM, like the VRU speed, its heading, its longitudinal acceleration and similar.
In order to access the motion information of the associated VRU, the VRU data provider takes advantage of a pointer to the SUMO mobility client, called *m_traci_client*, which is a TraCI client object used to access the TraCI interface and therefore to gather the aforementioned information from the SUMO scenario.

```
1       /* Speed [0.01 m/s] */
2       VAMdata.speed = VRUdpValueConfidence<>(m_traci_client->TraCIAPI::
        person.getSpeed (m_id)*CENTI,
3                                              SpeedConfidence_unavailable);
4
5       /* Longitudinal acceleration [0.1 m/s^2] */
6       if(m_compute_acceleration){
7           if(m_first_transmission){
8             VAMdata.longAcceleration = VRUdpValueConfidence<>(
        LongitudinalAccelerationValue_unavailable,
        AccelerationConfidence_unavailable);
9             m_first_transmission = false;
10          } else{
```

```
11          VAMdata.longAcceleration = VRUdpValueConfidence<>((VAMdata.speed
    .getValue()-m_prev_speed)*DECI/(now-m_prev_gen_time),
12
    AccelerationConfidence_unavailable);
13      }
14
15      m_prev_speed = VAMdata.speed.getValue ();
16      m_prev_gen_time = now;
17  } else
18      VAMdata.longAcceleration = VRUdpValueConfidence<>(
    LongitudinalAccelerationValue_unavailable,
    AccelerationConfidence_unavailable);
19
20  /* Position */
21  libsumo::TraCIPosition pos = m_traci_client->TraCIAPI::person.
    getPosition(m_id);
22  pos=m_traci_client->TraCIAPI::simulation.convertXYtoLonLat (pos.x,pos.
    y);
23
24  // longitude WGS84 [0,1 microdegree]
25  VAMdata.longitude=(Longitude_t)(pos.x*DOT_ONE_MICRO);
26  // latitude WGS84 [0,1 microdegree]
27  VAMdata.latitude=(Latitude_t)(pos.y*DOT_ONE_MICRO);
28
29  /* Altitude [0,01 m] */
30  VAMdata.altitude = VRUdpValueConfidence<>(pos.z*CENTI,
31                                            AltitudeConfidence_unavailable);
32
33  /* Heading WGS84 north [0.1 degree] */
34  VAMdata.heading = VRUdpValueConfidence<>(m_traci_client->TraCIAPI::
    person.getAngle (m_id) * DECI,
35                                           HeadingConfidence_unavailable);
```

Listing 3.6. Acquisition of VAM mandatory data

In addition to the *getVAMMandatoryData*, the VRU data provider contains two other functions which are used to retrieve the position, in terms of latitude and longitude, and the longitudinal acceleration of the associated VRU and a function which returns the nearest vehicle and the nearest pedestrian to the ego-VRU. This latter function is needed for the implementation of the seventh triggering condition, which is defined in chapter 2.6.2.

3.4 Local Dynamic Map

The Local Dynamic Map (LDM) of a VRU ITS-Station is a database which contains the information included in the VAMs received by the ego-VRU.
On ms-van3t, the LDM is implemented as a class made up of several functions implementing all the functionalities needed to manipulate the data stored inside the database. Among the notable functions, the *insert* function is used to insert a new vehicle or VRU inside the LDM if this latter is not present inside the database yet or to update its information if the road user is already present in the LDM, while the *rangeSelect* function is used to retrieve the information about the vehicles and VRUs stored inside the LDM

within a certain radius centered on a given latitude and longitude.
It is important to mention that the information stored inside the database are periodically erased through the deleteOlderThan function, since the meaning of the LDM is to only provide the ego-VRU the information related to nearby road users. For this reason, the previously mentioned function is scheduled to act every half a second deleting all the entries of the LDM older than one second, which are time parameters decided autonomously, since ETSI does not specify anything about them. [1]

3.5 Update of TraCI

As it was explained in chapter 3.1.3, TraCI is the interface connecting ns-3 and SUMO and it is used to retrieve the values of the simulated objects and to manipulate their behaviour.
When it was first implemented on ms-van3t, TraCI was designed to only support vehicles, since they were the only type of road user running in the simulations. For this reason, when the support to VRUs was added to ms-van3t, it was also necessary to update TraCI in such a way that VRUs, in particular pedestrians, were able to communicate with each other as well as with vehicles and in order to be able to retrieve their motion parameters. In order to add the support to VRUs, the TraCI client module was updated, allowing TraCI to also recognise and gather information from the pedestrians running in the simulation. The function which underwent the biggest update was the *SynchroniseNodeMap* function, which was suitably modified in order to also recognise the pedestrians present in the simulated scenario. In particular, the aforementioned function creates a map of nodes in order to keep track of all vehicles and pedestrians present in the SUMO simulation. These road users are in fact mapped in ns3 as nodes and, whenever a new road user enters the simulation, this function adds the corresponding node to the map, while, when a road user leaves the simulation, the *SynchroniseNodeMap* function removes the associated node from the map of nodes.

```
1     // Get all pedestrians present in the simulation
2     std::vector<std::string> sumoPed = this->TraCIAPI::simulation.
      getPedList ();
3     if(!sumoPed.empty()){
4         m_pedlist_empty = false;
5     }
6
7     if(!m_pedlist_empty){
8         // Iterate over all pedestrians present in the simulation
9         for (std::vector<std::string>::iterator it = sumoPed.begin(); it
      != sumoPed.end(); ++it){
10            // Get current pedestrian
11            std::string ped(*it);
12
13            // Search for pedestrian in the node map
14            std::map<std::string, std::pair< StationType_t, Ptr<Node> >
      >::iterator pos = m_NodeMap.find(ped);
15
```

```
16              // If the pedestrian is not present in the node map yet,
    include it
17              if (pos == m_NodeMap.end()){
18                  // Create the new node by calling the include function
19                  std::pair<StationType_t, Ptr<ns3::Node>> inNode_ped;
20                  inNode_ped.first = StationType_pedestrian;
21                  inNode_ped.second = m_includeNode(ped);
22
23                  // Register the new node in the map
24                  m_NodeMap.insert(std::pair<std::string, std::pair<
    StationType_t, Ptr<ns3::Node>>>(ped, inNode_ped));
25              }
26          }
27      }
```

Listing 3.7. Adding pedestrians to the map of nodes

Besides the *SynchroniseNodeMap* function, other TraCI client functions were also updated to fully support pedestrians on ms-van3t, such as the *UpdatePositions*, needed to successfully locate the vehicles and pedestrians present in the simulated scenario.

Chapter 4

Implementation on customisable On-Board Units

Once the implementation of the communication protocol described in chapter 2 on ms-van3t was completed, it was decided to implement the same protocol on an On-Board Unit (OBU), in order to test the effectiveness of the standard in a real world scenario, comparing the results of the real world tests with those obtained through the simulations. We started from two existing projects, called OCABS (Open CA Basic Service) and AIM (Automotive Integrated Map), we first merged them, since OCABS would only manage the transmission of CAMs, while AIM was designed to only handle the reception of CAMs and DENMs, and then we integrated the code needed to handle the transmission and reception of VAMs, thus creating a new project called OScar (Operative System for car)
For what concerns OCABS, it consists in a C++ implementation of the CA Basic Service designed to be used for deployment into a real hardware. In particular, this project includes a full implementation of the ETSI CA Basic Service, as defined by the standard ETSI EN 302 637-2 V1.4.1 [9], including the Transport and GeoNetworking layers. OCABS is only able to handle CAMs version 2 and, in order to work correctly, it needs a source of Position-Velocity-Time GNSS data through *gpsd*, which means that a GNSS device must be available and connected to a *gpsd* instance. [18]
OCABS also supports an enhancement to CAM messages, the so-called Enhanced CAMs, which can be used together with the Local Dynamic Map implemented in the AIM project. Since the transmission of these Enhanced CAMs is not supported by all standard-compliant stacks, it is enabled only when explicitly specified through a specific option. When merging the two projects to create OScar, it was decided to eliminate the support to Enhanced CAMs, since their implementation was out of the scope of this project.
AIM, on the other hand, is an enriched C++ implementation of the Local Dynamic Map of a vehicle. AIM is capable of receiving and decoding version 2 CAMs and to store the information included in these latters in a database. As it was previously mentioned, AIM is also able to work with Enhanced CAMs, which carry additional information about nearby vehicles, and their reception must be manually enabled through a specific option. If the related option is enabled, AIM is also able to receive and decode version 2 DENMs, which are however not yet used to update the database containing the information of

nearby vehicles. For this reason, their reception will only make AIM print some additional lines on a log file with some information included in the received DENM. [17]

4.1 On-Board Unit

The On-Board Unit (OBU) utilized in our field tests comes from the evolution of a DSRC platform, which consists in a Linux-based open-source platform which can be used with any IEEE 802.11p-compatible wireless Network Interface Card (NIC) and which encompasses all the necessary hardware and software components for deploying vehicular testbeds. The foundation of this platform involves customisable off-the-shelf hardware, specifically a Unex wireless module for data transmission over the 5.8/5.9 GHz frequency bands and the APU board by PC Engines, tailored for networking applications and supporting various wireless NICs, including the Unex modules.
Concerning the software, the platform was initially developed from a specific version of the OpenWrt embedded Linux distribution, particularly version 18.06.1 with Linux kernel 4.14.63. Subsequently, the project was upgraded to OpenWrt-V2X 21.02.1. The platform then underwent further enhancements, adopting newer APU2 boards with 4 GB of RAM and integrating additional technologies and services. This evolution aimed to create an OBU which could be easily installed on vehicles and similar road users for research and experimentation purposes. The OBU, running OpenWrt-V2X, incorporates technologies beyond standard IEEE 802.11p, including standard 4G connectivity, LTE-V2X Mode 4 release 14 and standard WiFi at 5 GHz. Notably, the design emphasizes ease of installation in existing non-connected vehicles, integration of various access technologies and the provision of an open-source C++ implementation of the ETSI ITS-G5 stack. All these features make the software highly customisable and extendable, rendering the OBU ideal for conducting the real-world tests we needed.
In addition to the APU2 board, the OBU integrates a lane-level accurate Multi GNSS receiver linked to the board through USB 3.0. Recognizing that most commercial GNSS devices lack the precision needed for safety-critical applications, as they can only provide an accuracy of 2 to 3 metres, the APU2 board interfaces with an advanced ArduSimple simpleRTK2B-F9R GNSS RTK and Inertial Navigation System device. This device, based on the u-box ZED-F9R dual-band GNSS chipset, offers configurable update rates up to 30 Hz and real-time kinematic (RTK) corrections. [23]
Notably, the OCABS and AIM projects, as previously discussed, along with the OScar project, which will be explored in Chapter 4.2, are all designed to operate on OpenWrt-V2X.

4.2 OScar

As it was previously mentioned, the OScar project was built by merging the OCABS and AIM projects, in such a way to create a project supporting both the transmission and the reception of ITS messages, and by integrating the support to VRU Awareness Messages. Based on what was previously said about OCABS and AIM, OScar could be defined as a C++ open source implementation of the CA and VRU Basic Services to be used on a

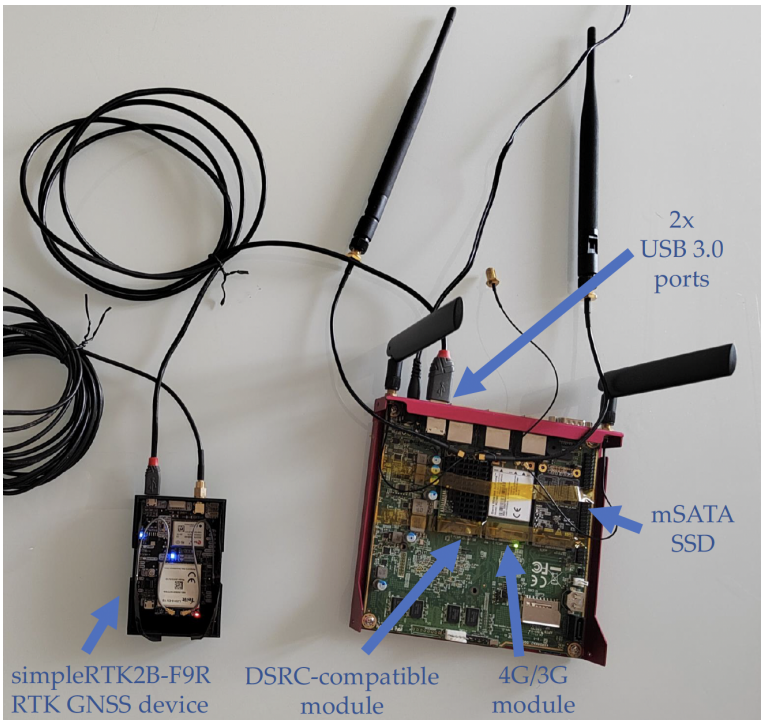

Figure 4.1. On-Board Unit. Taken from [23].

real hardware, containing also a C++ implementation of the Local Dynamic Map of a vehicle or VRU.

4.2.1 Merging of OCABS and AIM

Before integrating the support to VRU Awareness Messages, we needed to create an application capable of both transmitting and receiving ITS messages. For this reason, the first thing which was done was to merge the two projects OCABS and AIM in such a way to create a new project, called OScar, supporting both the transmission and reception of ITS messages, in particular CAMs.
In order to do this, we started from the OCABS project and we gradually integrated all the functionalities which were peculiar of AIM, in particular its ability to receive and decode ITS messages and the implementation of the Local Dynamic Map, used to store all the most important information related to nearby vehicles.
First of all we focused on creating an application which could handle both the transmission and the reception of CAMs and, in order to do this, we first updated the transport and networking layers of OCABS, since they were originally designed to only support the dissemination of CAMs. We therefore merged the functionalities of the BTP and GeoNetworking layers of OCABS and AIM in such a way to make these two layers capable of handling both the dissemination and the reception of ITS messages by vehicles. In particular, starting from the BTP and GeoNetworking layers of OCABS, we needed to

add some functions allowing them to properly receive and decode ITS messages, functions that we derived from the BTP and GeoNetworking layers of AIM.
Once the transport and networking layers had been suitably updated, we focused on adding to OCABS all the functionalities needed to successfully decode the received ITS messages, in particular CAMs and DENMs, and, in case of CAMs, the decoded information also need to be stored inside an internal database called Local Dynamic Map. This was done by importing to OCABS the *etsiDecoderFrontend* and the *SocketClient* C++ classes already present in AIM.
The *etsiDecoderFrontend* class contains some useful functions and information for decoding CAMs and DENMs, but the most important class in terms of reception of ITS messages is the *SocketClient*, which performs the decoding of both CAMs and DENMs and updates the internal database with the information included in the received CAMs. In particular, the *SocketClient* contains a function called *rxThr*, which is used to properly receive the incoming messages, and another function called *manageMessage*, which is instead used to decode the received ITS messages and, in case of CAMs, it processes the information included in these latters and uses them to update the internal database. The information included in the received messages, in this case both CAMs and DENMs, can also be printed on a log file, if needed. The *SocketClient* class also contains two functions used to start and stop the reception of ITS messages.
Since the information included in the received CAMs need to be stored inside an internal database, our project also needs to include an implementation of the Local Dynamic Map of a vehicle, which we imported from the AIM project. The LDM implementation is more or less the same to the one we described in chapter 3.4 when talking about the implementation of the VAM communication protocol on ms-van3t, in fact, also in this case, we have a *LDM* class including functions used to update the database with new data, to retrieve information from this latter and to periodically delete old data from the database.
For what concerns the transmission of CAMs, on the other hand, OCABS contains a full implementation of the CA Basic Service and it also includes a class called *gpsc* providing all the functionalities of the vehicle data provider (VDP). The only difference with the vehicle data provider implemented on ms-van3t is that *gpsc* retrieves the data to be included in a CAM directly from a GPS device.
The last thing to do in order to successfully merge OCABS with AIM was to update the main file in order to suitably set up the whole application making it ready to start the dissemination of CAMs and the reception of both CAMs and DENMs. The main file is therefore organised in two parts: the first half of the code prepares the dissemination of CAMs, by setting the dissemination interface, the socket used for broadcasting the messages and the BTP and GeoNetworking layers, by creating a VDP GPS Client object and by initialising the CA Basic Service, which is used to start the dissemination of CAMs, while the second half of the code is used to set up the reception of ITS messages, in particular CAMs and DENMs, by setting the socket used for reception and the reception interface, by creating a new LDM object to store the data included in the received CAMs together with a JSON server object to be used by the client to retrieve the database data and by creating and initialising the main *SocketClient* object, which is also used to start

the reception of messages.
The main file also contains a function called *DBcleaner_callback* which is used to periodically delete the older entries of the internal database of the vehicle. In particular, this function schedules the *deleteOlderThan* function of the LDM object every 5 seconds and this latter deletes all the entries of the database older than 7 seconds.

4.2.2 Integration of the VRU Basic Service

Once the OCABS and AIM projects had been correctly merged, the following step was to integrate the support to VRU Awareness Messages and this was done by first adding the VRU Basic Service together with the VRU data provider to the project described in the previous section and then by slightly modifying the main file and the classes used for the dissemination and reception of ITS messages in order to fully support VRUs, in particular pedestrians, and VAMs.
First of all we need to talk about the integration of the VRU Basic Service in the OScar project. The VBS class of the OScar project is very similar to the one we have discussed in chapter 3.2 when talking about the implementation of the VBS as described by the standard ETSI TS 103 300-3 V2.2.1 [12] on ms-van3t, which means that the functions of the two classes are almost the same and they provide the same functionalities, however, since the VBS class of ms-van3t schedules the dissemination of messages in a simulated scenario through the ns-3 network simulator, while the VBS class implemented in OScar needs to operate on an On-Board Unit in a real world scenario, they are characterised by two slightly different ways of handling the dissemination of VAMs. In particular, since the ns-3 *schedule* function cannot be used in this case, we took advantage of a custom made macro, called *SCHEDULE*, which can be used to schedule the asynchronous execution of a function after a certain amount of milliseconds. We used this macro to schedule first the *initDissemination* function, which is used to start the dissemination of VAMs, and then the first execution of the *checkVamConditions*, which is used to check if any of the triggering conditions described in chapter 2.6.2 has been satisfied since last VAM transmission. The following executions of the *checkVamConditions* are not scheduled through the aforementioned macro, but through a timer designed to periodically check the VAM conditions every 100 ms, as stated by the standard ETSI TS 103 300-3 V2.2.1 [12], and the check of the aforementioned conditions takes place inside a while cycle which makes the dissemination of VAMs go on until it is manually interrupted.
The *checkVamConditions* function is also used to save on a log file the condition which triggered a new VAM transmission, which will be labelled as *NONE* if no VAM was generated at a given time instant, together with the motion data of the ego-VRU each time this function is executed.
The VRU Basic Service implemented in OScar implements VAM redundancy mitigation, which is checked through a dedicated function, called *checkVamRedundancyMitigation*, and acts in the exact same way as on ms-van3t.
The generation and encoding of a new VAM to transmit are handled by the *generateAndEncodeVam*, which is almost identical to the one implemented on ms-van3t. Also in this case, in fact, only the mandatory fields of the VAM were filled and the data to be included in the new VAM were retrieved using the *getVamMandatoryData* function

implemented in the VRU data provider.
The VRU data provider class is again very similar to the one implemented on ms-van3t and it is used to provide the mandatory data to be included in a new VAM to transmit. The most important function is the *getVamMandatoryData*, which is identical to the one we have discussed in chapter 3.3, except for the way data are retrieved. If, indeed, on ms-van3t, the data to be included in the VAM format were retrieved from the simulated scenario taking advantage of the SUMO mobility client, in this case these data must be retrieved directly from a GPS device and this is done through a suitable structure allowing us to access the motion parameters measured by the GPS device, like the position, in terms of longitude, latitude and altitude, of the VRU, its speed and its heading. The VRU data provider also contains three dedicated functions, called *getPedPosition*, *getPedSpeedValue* and *getPedHeadingValue*, which can be used to retrieve respectively the position, the speed and the heading of the VRU from the GPS device. Also in this case, the VRU data provider contains the *get_min_distance* function to retrieve the distances of the ego-VRU from the nearest VRU and vehicle and one other function, called *convertLatLontoXYZ*, used to convert the position of the VRU expressed in terms of longitude, latitude and altitude in Cartesian coordinates. Lastly, the two functions *openConnection* and *closeConnection* are used to set up and close the connection to the GNSS device.
Once the VRU Basic Service and the corresponding data provider were ready, we moved on by integrating the support to the reception of VAMs in the *SocketClient* class. In order to do this, we simply had to update the *manageMessage* function in such a way to make it able to also decode VAMs other than CAMs and DENMs. Just like it happens with CAMs, the main information derived from the received VAMs have to be stored inside the internal database of the vehicle/VRU, which is not a problem since no changes had to be made to the LDM class implemented in OScar to also support VRUs and VAMs. The *etsiDecoderFrontend* class also had to be slightly modified, adding some useful information needed for the correct reception and decoding of VAMs.
We then integrated the support to VAMs in the BTP and GeoNetworking layers of OScar, by adding the support to the VRU data provider and by updating the long position vector in such a way that VRUs are correctly located when disseminating VAMs and, as last thing, we updated the main file in order to suitably set up the dissemination of VAMs. In order to do this, an if statement was added to the transmission cycle, in such as way that, if the dissemination of CAMs is enabled, everything works as described in section 4.2.1, while, if the dissemination of VAMs is enabled, a VRU data provider object is created, the VRU Basic Service is initialised and the dissemination of VAMs is started. For what concerns instead the setup of the reception cycle, no modifications were needed.

Chapter 5

Results

Now that we have carefully described the VAM communication protocol, as defined by the standard ETSI TS 103 300-3 V2.2.1 [12], and its implementation on both ms-van3t and the On-Board Unit, we will present the results obtained from the ms-van3t simulations and the field tests, concluding with a comparison between the two.
The testing of the aforementioned communication protocol, both using the simulator and on the field, was intended at showing the effectiveness of the communication standard, by measuring parameters like the latency and the packet reception ratio (PRR) of VAMs and comparing them with those of CAMs, but also at understanding which triggering conditions trigger the majority of VAMs' transmissions, thus reflecting on the effectiveness and correctness of the thresholds characterising these triggering conditions.

5.1 Simulation on ms-van3t

First of all, before going on the field, we tested the communication protocol as it was implemented on ms-van3t using the mobility simulator SUMO to create a traffic scenario made up of both vehicles sending CAMs and pedestrians sending VAMs.
Many simulations were conducted using different numbers of vehicles and pedestrians, different transmission powers and different data rates and two different metrics were collected, related to both CAMs and VAMs, namely the latency and the PRR, which will be defined in section 5.1.2.
We also collected the conditions triggering each new VAM transmission, which we used to make some statistics in order to figure out which are the most common triggering conditions, and the main motion parameters of the VRUs, in such a way to check if all triggering conditions were implemented properly and if the related thresholds are appropriate.

5.1.1 Traffic scenario

The location chosen for building our simulated traffic scenario was Turin and, in particular, the part of corso Castelfidardo next to Poltecnico di Torino. This location, as we can see in figure 5.1, gave us the opportunity to build a complete and realistic traffic scenario,

which is the reason why it was chosen. In particular, this map is characterised by two central and two secondary roadways, the central ones made up of two lanes, while the secondary ones made up of one lane only, all reserved to vehicle traffic, allowing vehicles to travel in both directions. On the two sides of the map we also have two sidewalks, where only pedestrians can travel, and on the right side there is also a cycle path, for future use, since, at the moment, pedestrians are the only type of VRU supported by ms-van3t. The main reason why this location was chosen, however, are the 3 pedestrian crossings located in the upper, middle and lower part of the map, which are regulated by simple priority rules when crossing the two lateral secondary roadways and by traffic lights when crossing the two central ones. These 3 crossings gave us the opportunity to make vehicles and pedestrians interact with each other by exchanging ITS messages, namely CAMs and VAMs, and, thanks to this active interaction between different types of road users, this scenario is also perfect for future tests related to collision avoidance.
The map was downloaded from the OpenStreetMap website [4], which provides and open source access to worldwide geographic data. This map was then converted into a *xml* file using SUMO's *netconvert* tool [21], which is used to import digital road networks from different sources, including OpenStreetMap, and to generate road networks that can be used and manipulated by other tools provided by SUMO. In particular, the *netconvert* tool allowed us to generate a *.net.xml* file, which we then manipulated using SUMO's *netedit* tool [22], which is a graphical network editor which can be used for both creating from scratch new traffic networks and for modifying existing networks, which is exactly our case. We in fact used *netedit* to modify and, in particular, simplify the map downloaded from OpenStreetMap, in order to build a suitable traffic map to be used to effectively test the VAM communication protocol and to compare its performances with those of the CAM protocol. For this reason, all roads, including the cycle path and the sidewalks, were prioritised accordingly and the traffic lights cycles were suitably defined.
Once the map was completed, we defined two route files, one for vehicles and the other one for pedestrians, defining the number of road users present in the simulation, specifically 45 vehicles and 25 pedestrians, the time instants in which each of them is included in this latter and their itineraries. The definition of the time instants in which each road user enters the simulation was essential to build a traffic scenario providing realistic results, in particular pedestrians had to be spread throughout the map as much as possible in order not to form groups, which, according to the standard ETSI TS 103 300-3 V2.2.1 [12] and as already described in chapter 2.1.3 of the present document, shall be treated as VRU clusters and these latters are not currently supported by ms-van3t. For what concerns, on the other hand, the travelling speeds of the road users taking part to the simulation, we defined the maximum speeds allowed on the different types of roads, in particular, on the two central roadways a maximum speed of 50 km/h was set, maximum speed which is decreased up to 30 km/h on the two lateral roadways, just like it happens in the real world, while, in terms of sidewalks, we defined a maximum speed for the pedestrians of 5 km/h, in such way to provide a traffic scenario as realistic as possible. Even though they are not currently used, we also set the maximum speed allowed on the cycle paths to 20 km/h, in order to also take into account e-bikes.

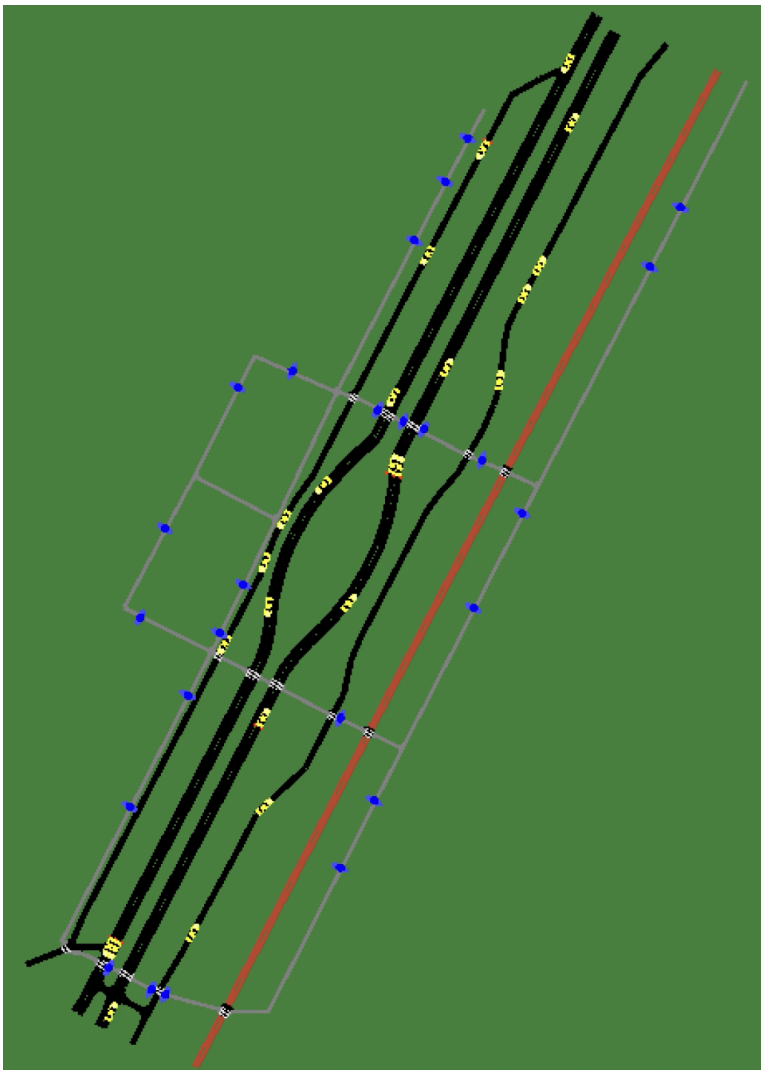

Figure 5.1. SUMO traffic scenario

5.1.2 Application

After building the traffic scenario, we needed to create an application, which we called "*v2p_vam_80211p*", in order to properly set up the simulation. In particular, this application consists in creating and setting up the nodes to be associated to the road users taking part to the simulation and the communication channel used for the dissemination and reception of the ITS messages and in setting up all the functions and entities which will be used by the simulated ITS-Stations during the simulation.
First of all, in the main file, all the parameters related to the simulation are set up, specifying the range of possible data rate and transmission power values which can be used and the map and route files to be used for the SUMO simulation. It is worth highlighting that all the parameters related to the application can be adjusted through dedicated

command line options, which are suitably decoded and interpreted in the main file of the application itself.
Once all parameters are properly set, we first need to read the vehicles and pedestrians route files, in order to derive the number of vehicles and pedestrians which will be created during the simulation. This is done through a dedicated C++ class implemented on ms-van3t, called *sumo_xml_parser*, which was suitably updated to make it able to also recognise the XML elements of type <person> and not only those of type <vehicle>. The number of vehicles and pedestrians retrieved from the aforementioned route files is then used to set up the number of nodes to be created, which are grouped inside a container of type *NodeContainer*. The same container will contain both the nodes to be associated to vehicles and those to be associated to pedestrians, since vehicles and pedestrians need to be able to exchange messages. At this point, we have a number of nodes equal to the overall number of road users present in the simulation, which are however not connected to each other. For this reason, we now need to create and set up the communication channel.
First of all, we define the IEEE 802.11p model, which defines the physical layer for communication. We then define a 802.11p object, which is needed to tell ns-3 that we want to simulate the IEEE 802.11p standard. At this point, we create and set up the MAC layer and we assign our IEEE 802.11p wireless channel a certain physical data rate and transmission power. We then need to connect the channel to the nodes, therefore we proceed by installing proper IEEE 802.11p Network Interface Cards on the nodes, which are now connected with each other through the Wi-Fi channel. We then give packet socket powers to the nodes, so that they are able to communicate thanks to the IEEE 802.11p channel, set up the mobility and position node pool, set up TraCI and start SUMO.
The last steps consist in creating a PRR supervisor object, which will be used to collect all the relevant metrics related to both CAMs and VAMs, and the csv file where we can store the collected metrics and to create two callback functions, called *setupNewWifiNode* and *shutdownWifiNode*, used for nodes creation and shutdown respectively. The *setupNewWifiNode*, in particular, performs the following actions:

- Identification of the station type of the road user (vehicle or pedestrian) and extraction of the corresponding node from the NodeContainer.
- Creation of the GeoNetworking packet socket and setting of the proper access category through the specified user priority.
- Creation and setup of a new Basic Service Container object (*BSContainer*), which is used to set up both the ETSI CA Basic Service and the ETSI VRU Basic Service for the transmission and reception of CAMs and VAMs. This container is simply a wrapper class enabling an easier handling of both CAMs and VAMs and, if needed, it can also be used to set up the ETSI DEN Basic Service for the transmission and reception of event-based DENMs, which we have not considered in this document. It is important to highlight that the ETSI CA and VRU Basic Services are installed on all nodes created, regardless of the station type of the associated road user, since vehicles and pedestrians need to be able to receive both CAMs and VAMs and, in order to do this, they need to have access to both basic services. The Basic Service

container object is then used to set up the PRR supervisor, the file used to store the metrics related to VAMs and the functions which will be called each time a new ITS message, CAM or VAM being, is received. These functions are the reception callbacks already defined in chapter 3.2.3 and they are used by the corresponding basic service to make the handling of the received CAMs and VAMs simpler. Each node created will be provided with two different reception callbacks, one for its CA Basic Service and the other one for its VRU Basic Service. Lastly, the Basic Service container object is stored inside a local global *BSMap*, which is a structure allowing to easily retrieve the right *BSContainer* given a vehicle/VRU ID.

- Start the transmission of CAMs in case of vehicles and of VAMs in case of pedestrians.

The *shutdownWifiNode*, on the other hand, is used to set the position of the node leaving the simulation outside of the communication range, to terminate its transmission of CAMs or VAMs and to clean the associated Basic Service container.
Lastly, TraCI client is started and the simulation as well, which, in our case, lasts for 250 seconds.
The last part of the main function is dedicated at storing, on the aforementioned csv file, the metrics of interest, related to both CAMs and VAMs, and to eventually print in the terminal window the overall number of CAMs and VAMs successfully transmitted and received.

5.1.3 Simulation and results

We will now discuss the main results related to the VAM communication protocol derived from the simulations.
First of all we can have a look at the Wireshark trace of one of the pedestrians present in the simulation, which shows all the ITS messages transmitted and received by the chosen pedestrian. As we can see from figure 5.2, VAMs are indicated as BTPB, since the current version of Wireshark does not support the VAM protocol yet, however all information stored inside them are correctly transmitted and received. It is also possible to notice that, as previously highlighted, pedestrians are only able to transmit VAMs, but they are able to receive both VAMs and CAMs, just like vehicles can only send CAMs, but they can receive both types of messages.

We then performed some simulations with the aim of retrieving the trend of the PRR (Packet Reception Ratio) and the latency of both VAMs and CAMs with respect to parameters like the transmission power, the data rate and the number of pedestrians present in the traffic scenario.
The Packet Reception Ratio (PRR) can be defined as the probability that a certain data packet is correctly received from all road users within a certain baseline, called PRR baseline, of the sender, which, in our case, corresponds to 150 meters. The latency, on the other hand, can be defined as the time it takes for data to travel from one road user to another. In our case, we will always consider average measures, meaning that we will report the average PRR and one-way latency either of a single pedestrian or of all pedestrians or vehicles present in the simulation.

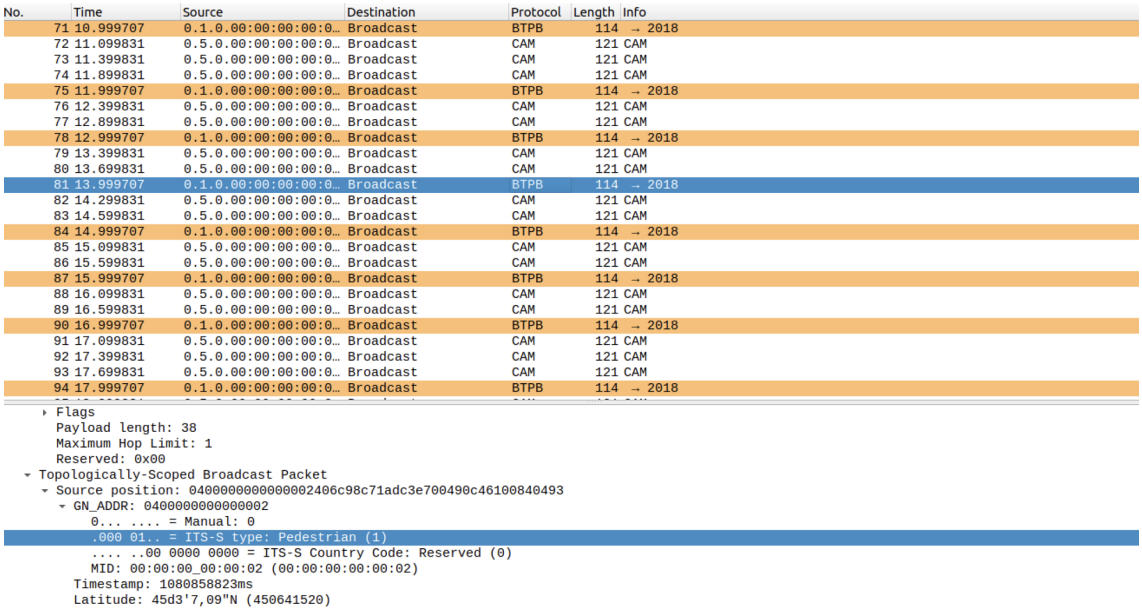

Figure 5.2. Wireshark trace of a pedestrian

Starting with the average PRR and one-way latency of the VAMs disseminated by a single pedestrian, we can have a look at their trends with respect to the transmission power, while keeping the data rate fixed at 12 MHz.

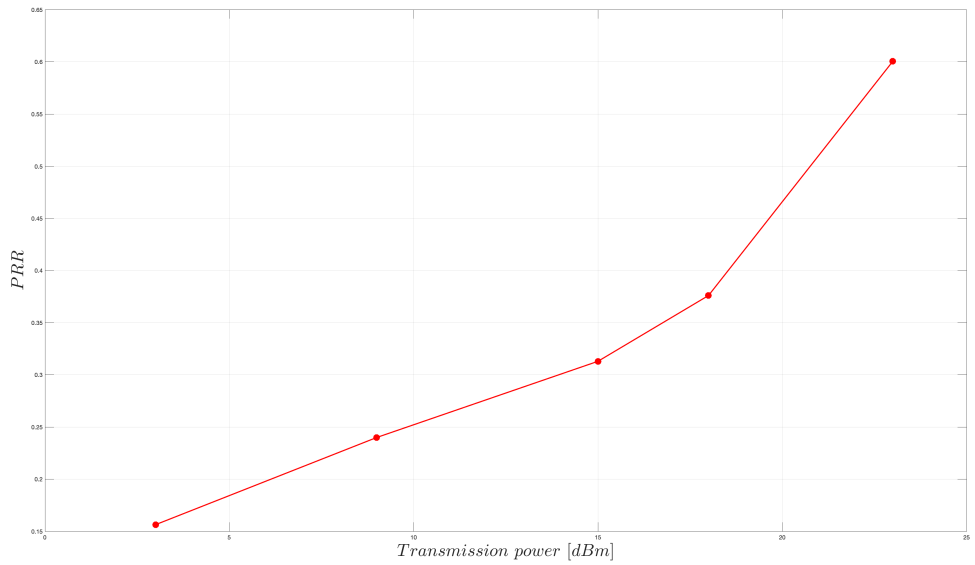

Figure 5.3. Average PRR of one pedestrian vs transmission power

As we can see from figure 5.3, the PRR increases as the transmission power increases,

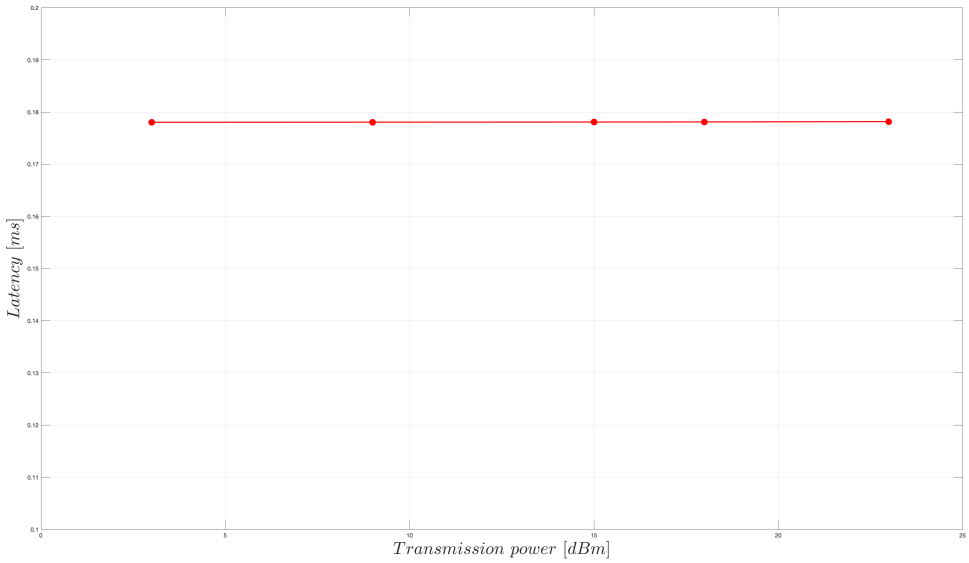

Figure 5.4. Average one-way latency of one pedestrian vs transmission power

which means that the probability that a transmitted ITS message is correctly received increases with an increase of the transmission power. This behaviour could be expected since a higher transmission power means that the originating ITS-S will be able to reach more vehicles and pedestrians, therefore, with respect to the total number of road users present in the simulation, the probability that the transmitted VAM will be correctly received by one or more ITS-Stations will increase as well.
If we look at figure 5.4, we can instead notice that the average one-way latency of the VAMs transmitted by one pedestrian is basically not affected at all by the transmission power, it remains almost perfectly constant as this latter varies. Again, the result obtained seems reasonable and it means that a change in the transmission power of the originating ITS-S does not affect the time taken for the transmitted data packet to reach its final destination.
We can now analyse how the PRR and one-way latency change with respect to the data rate, keeping the transmission power fixed at 23 dBm.

As we can see from figure 5.5, the PRR decreases as the data rate increases, which is a result we could expect, since it is normal that more mistakes are made as the transmission frequency with which VAMs are sent increases. In particular, we can notice that the PRR decrease is slight between 3 MHz and 18 MHz and then it gets bigger, which means that, after a certain frequency threshold, the transmission protocol starts to become less and less reliable. On the other hand, figure 5.6 shows that the average one-way latency decreases as the data rate increases, as it was expected, since the higher the velocity of the transmission the lower it is the time it takes for data to travel from one node to the other.
We can then have a look at how the PRR and the latency of the VAMs transmitted by one pedestrian change according to the number of pedestrians present in the simulation,

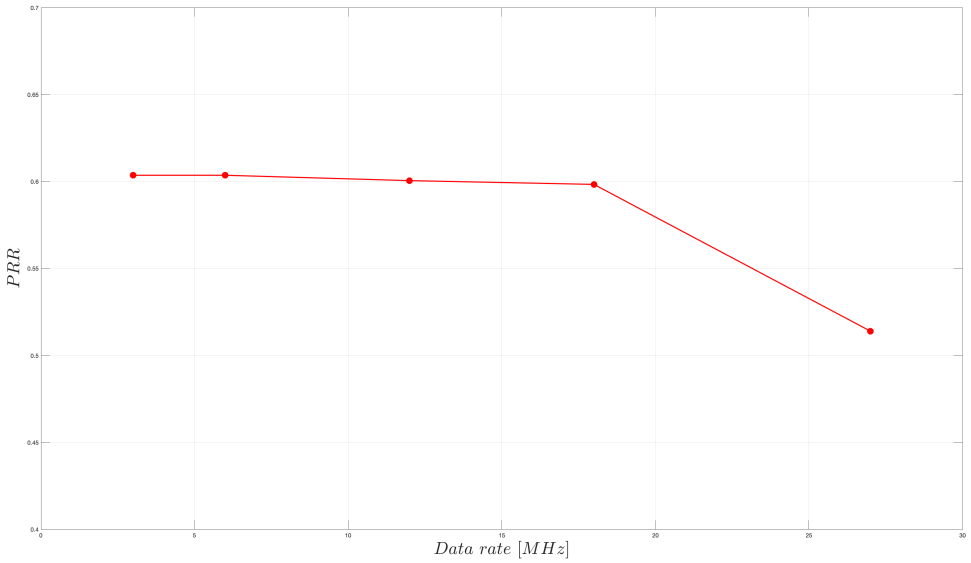

Figure 5.5. Average PRR of one pedestrian vs data rate

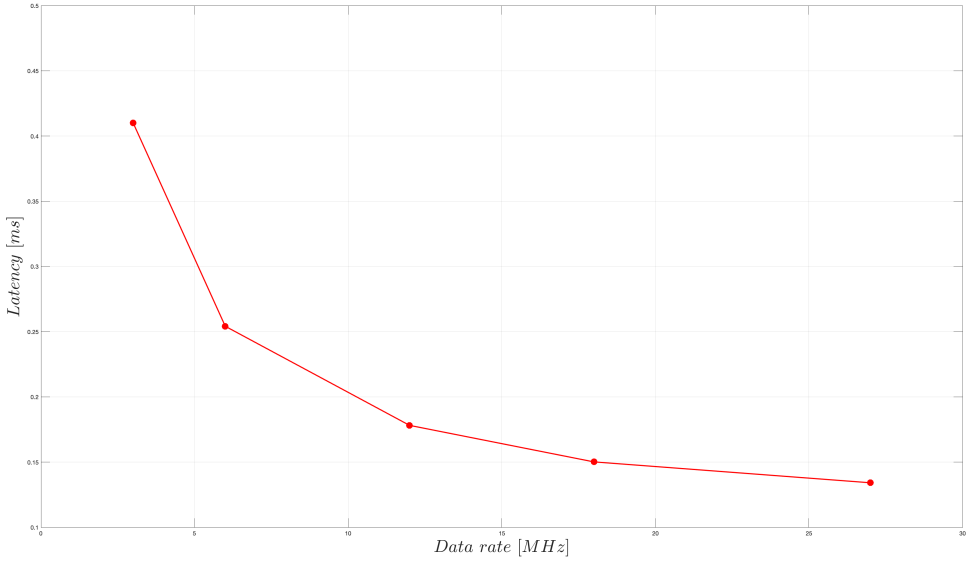

Figure 5.6. Average one-way latency of one pedestrian vs data rate

keeping the transmission power fixed at 23 dBm and the data rate fixed at 12 MHz.

Figures 5.7 and 5.8 show that both PRR and latency remain more or less constant as the number of pedestrians present in the simulation varies, so we can affirm that the number of pedestrians does not affect the performances of the VAM communication protocol.

Now that we have discussed the performances of the VAM communication protocol in

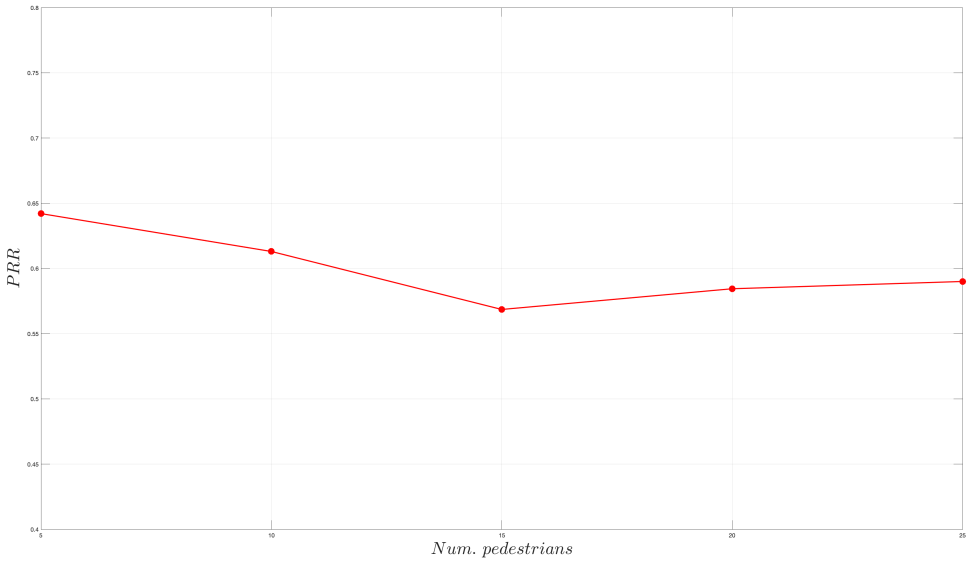

Figure 5.7. Average PRR of VAMs vs number of pedestrians

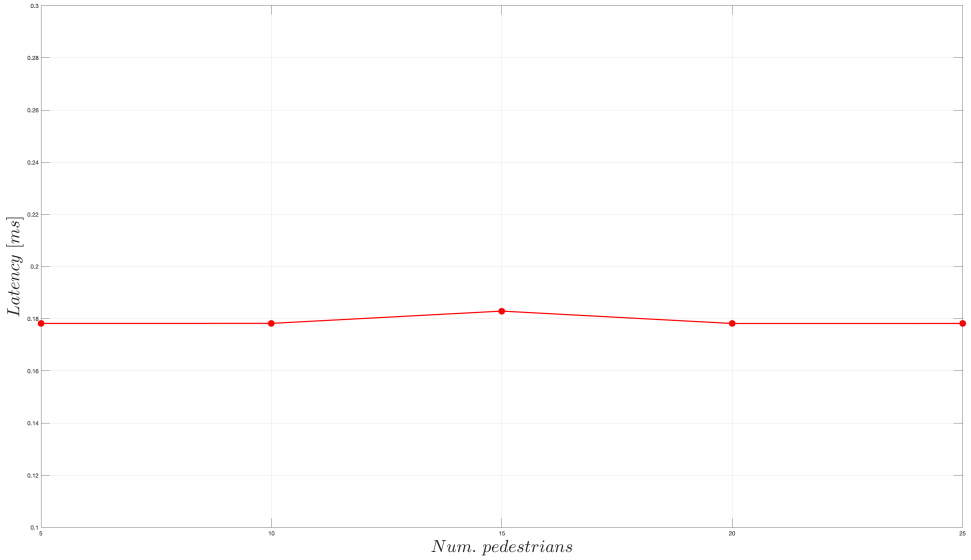

Figure 5.8. Average one-way latency of VAMs vs number of pedestrians

terms of average PRR and average one-way latency, we can compare its performances with those of the CAM communication protocol, this time taking into account the average PRR and one-way latency of all pedestrians and vehicles present in the simulation. In this case, we will analyse how these two parameters change with respect to the transmission power and the data rate.

As shown in figures 5.9, 5.10, 5.11 and 5.12 the two communications protocols have

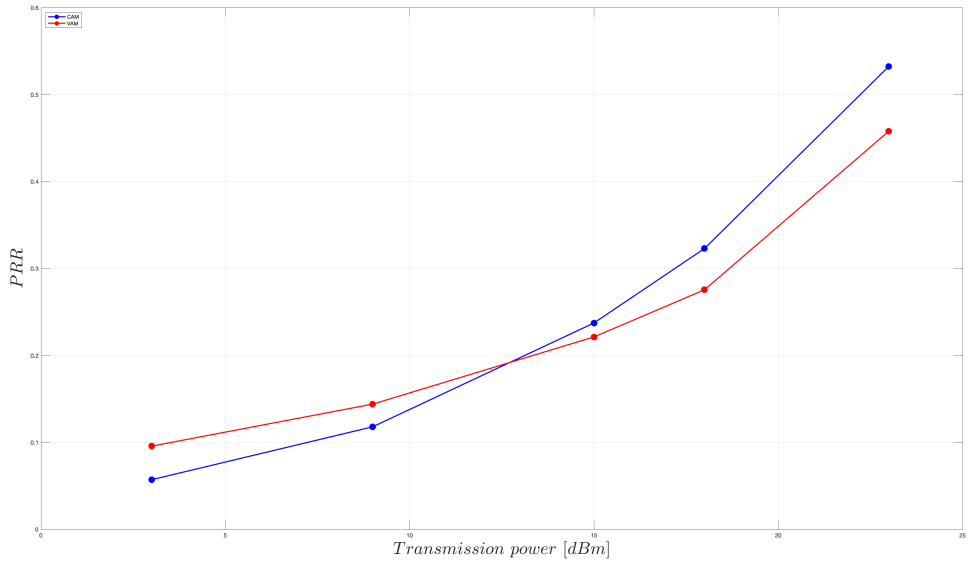

Figure 5.9. Average PRR of VAMs and CAMs vs transmission power

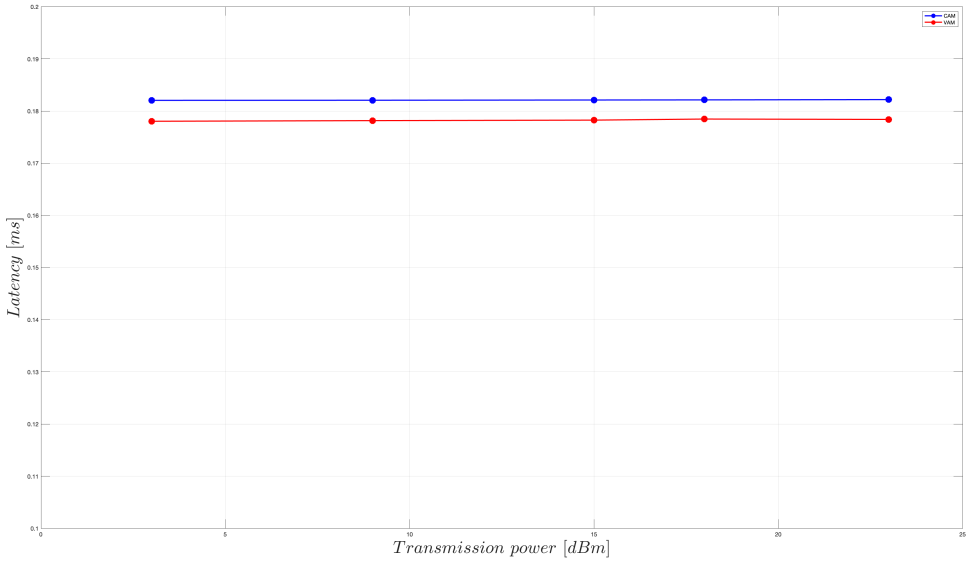

Figure 5.10. Average one-way latency of VAMs and CAMs vs transmission power

very similar performances. In particular, we can notice that CAMs have slightly better performances in terms of PRR, with the only exception of low transmission powers, below 13 dBm, while VAMs have slightly better performances in terms of latency, even though, in this case, the difference between the two communication protocols is extremely limited. The tests were conducted keeping the data rate fixed at 12 MHz when making the transmission power vary and the transmission power fixed at 23 dBm when making the

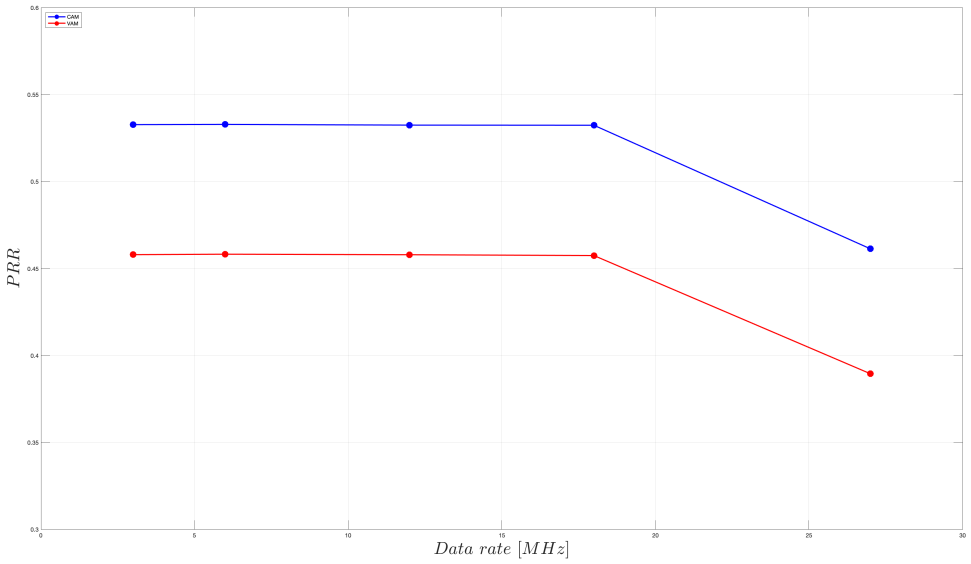

Figure 5.11. Average PRR of VAMs and CAMs vs data rate

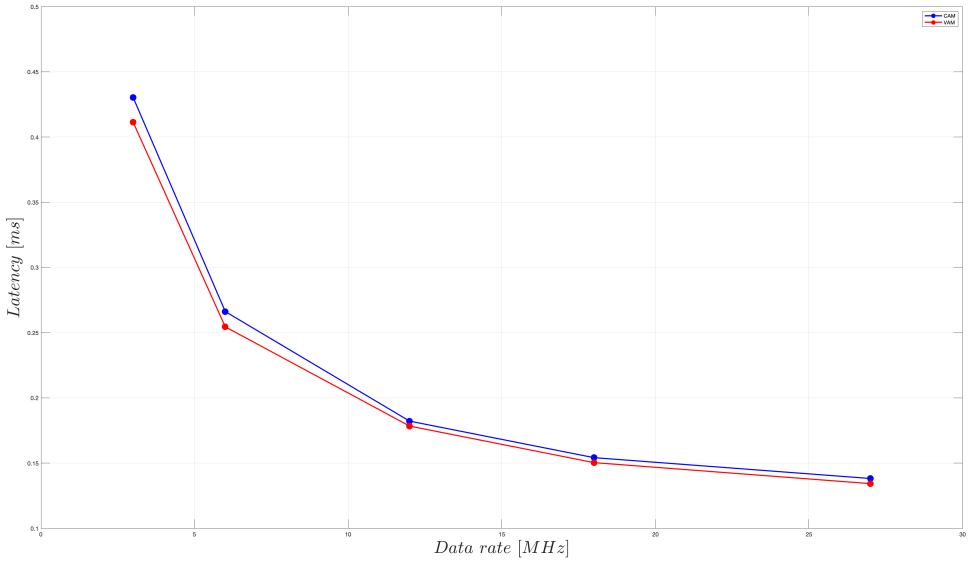

Figure 5.12. Average one-way latency of VAMs and CAMs vs data rate

data rate change.
After having discussed the performances of the VAM communication protocol and having compared them with those of the CAM communication protocol, we can focus on analysing the relation between VAM transmissions and the conditions triggering them. We want to focus in particular on how VAM transmissions are distributed among the 5 triggering conditions which are implemented on ms-van3t, to see which ones are the most

recurring ones and to understand if the related thresholds given by ETSI are reasonable or if the should be modified in order to allow for a better functioning of the communication protocol. The results obtained are reported in figures 5.13 and 5.14.

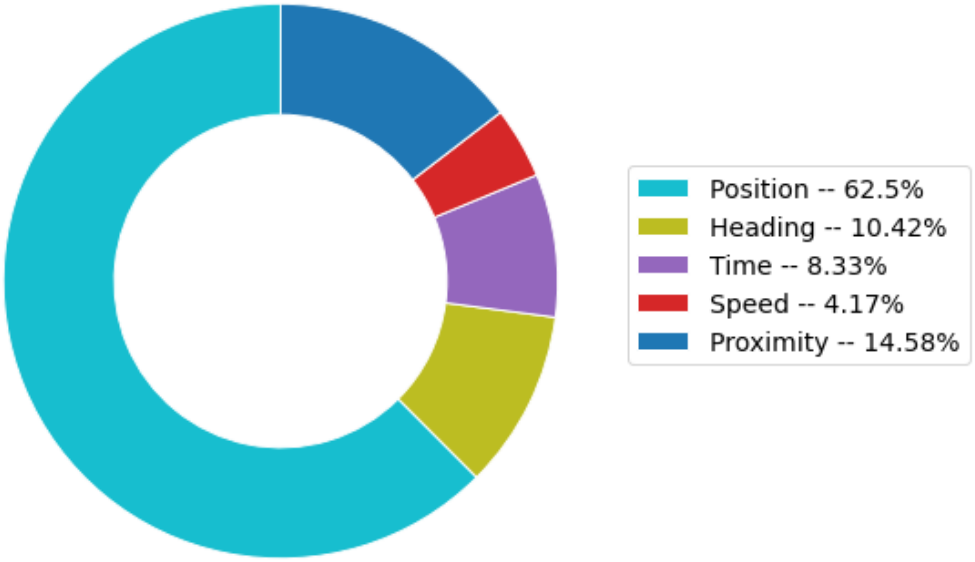

Figure 5.13. VAM transmissions percentage distribution with respect to triggering conditions

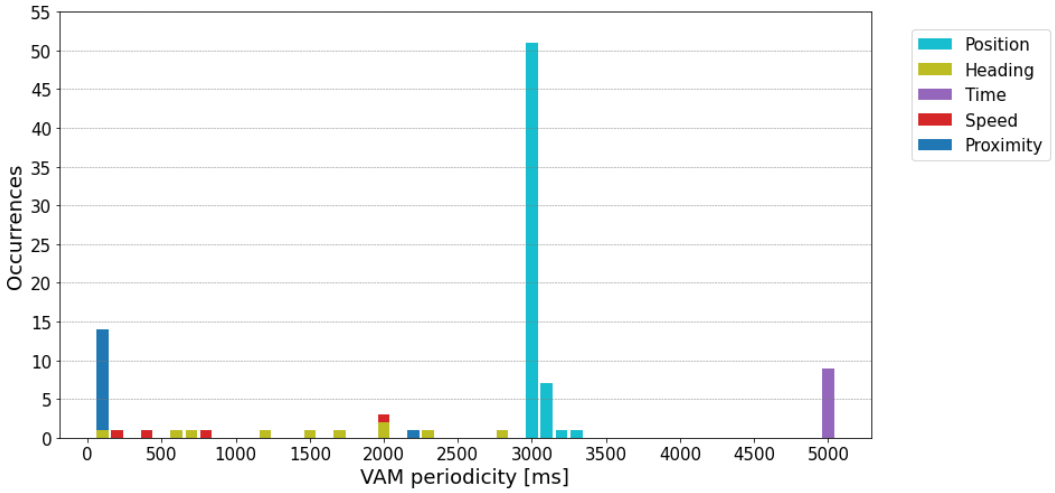

Figure 5.14. VAM transmissions occurrences vs VAM periodicity

Figure 5.13 shows the distribution of VAM transmissions with respect to the 5 triggering conditions implemented on ms-van3t. We can notice that the most recurring one is by far the position triggering condition, which, as discussed in chapter 2.6.2, triggers a new VAM transmission whenever the difference between the current position of the VRU and the position stored in the last VAM transmitted is greater or equal to 4 m. Figure 5.14 confirms that this triggering condition is working properly, since, in our simulation, pedestrians travel with a speed of around 5 km/h, which means that, if no other condition is verified, the position one should trigger a new VAM transmission more or less every 3 seconds, which is exactly what is shown in figure 5.14. The proximity triggering condition, which is the one related to VRU safe distances, is the second most recurring one and, as explained in chapter 2.6.2, it is limited by VAM redundancy mitigation, in order to avoid flooding the communication channel with unnecessary messages. The heading and time triggering conditions are more or less equally distributed, while the speed triggering condition is the less recurring one, which, being the speed change threshold related to this condition equal to 0.5 m/s, is a result we could expect, since, due to their low travelling speed, it is not so common for pedestrians to verify this triggering condition. The fact that there are only few VAM transmissions triggered by a change of speed of the pedestrian is not a problem, since their travelling speed is typically very low, therefore no danger for pedestrians themselves or for those surrounding them typically comes from their travelling speed. We can therefore conclude that the thresholds associated to the 5 VAM triggering conditions implemented on ms-van3t, which were presented in table **??**, are all reasonable. Given that the same triggering conditions and the related thresholds shall be applied to the VAMs transmitted by any type of VRU, we would obtain completely different results if, instead of pedestrians, we would for example consider cyclists, however we can affirm that, at least for pedestrians, the thresholds associated to these 5 triggering conditions were very well designed.
All the results shown in the previous figures are characterised by the presence of VAM redundancy mitigation. As explained in chapter 2.6.2, VAM redundancy mitigation is a technique used to prevent flooding the communication channel with redundant messages and it applies whenever there are two or more pedestrians very close to each other, with similar speed and heading, which are disseminating VAMs. In order to verify if VAM redundancy mitigation was working as expected we built a very simple dedicated traffic scenario characterised by the same map described in chapter 5.1.1, but having only two pedestrians travelling next to each other on a straight line for 15 seconds. The simulation was run with and without VAM redundancy mitigation and we measured, in the two cases, the number of VAMs sent when the position, speed and heading differences between the two pedestrians in the simulation were below the 3 thresholds characterising VAM redundancy mitigation, which are defined in table 2.2. The results obtained are shown in figure 5.15.

As we can see from the histogram shown in figure 5.15, when VAM redundancy mitigation is active the number of VAMs sent when the aforementioned thresholds are violated is substantially decreased, meaning that VAM redundancy mitigation is effective in reducing the amount of load affecting the communication channel without impacting the

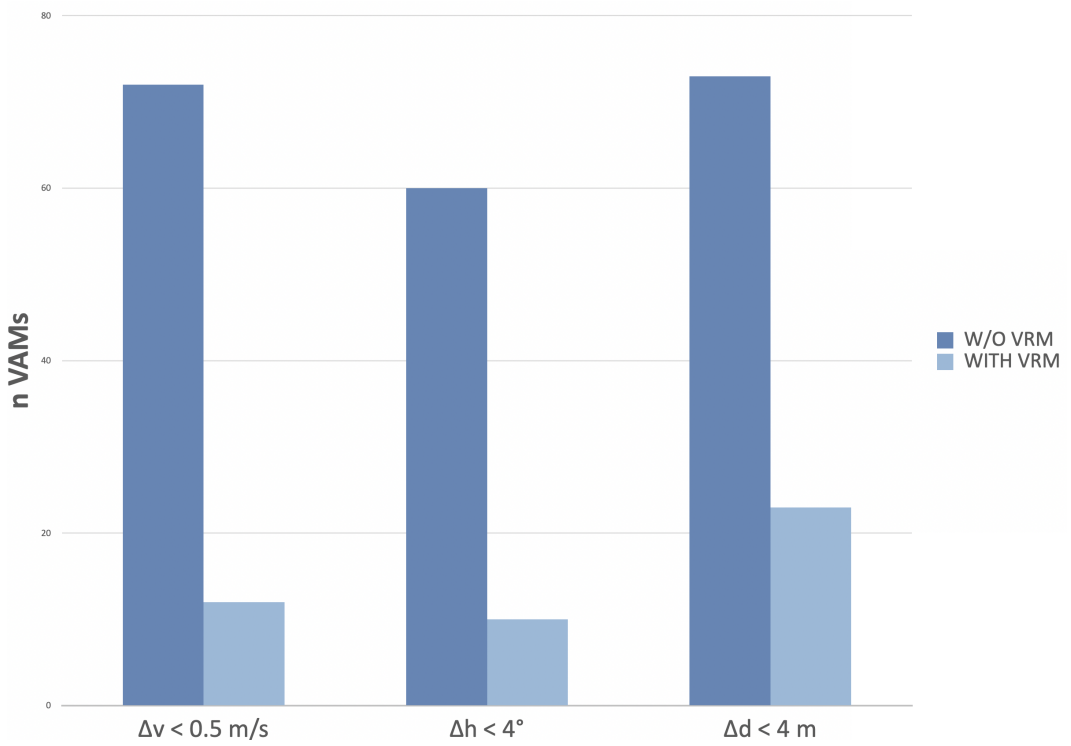

Figure 5.15. Number of VAMs sent with and without VAM redundancy mitigation - 2 pedestrians scenario

awareness of VRUs, since VAM transmissions are deleted only when the motion parameters of the two pedestrians are very similar and therefore the transmitted VAMs would contain almost the same information.
We have said that VAM redundancy mitigation is performed when the position, speed and heading differences between two close pedestrians are below the 3 aforementioned thresholds and these differences are computed by using the data stored in the LDM of the ego-VRU. In particular, the TraCI client module is used to retrieve the position, speed and heading of the ego-VRU, while the LDM of this latter is used to retrieve the same parameters of the nearby pedestrians, therefore the position, speed and heading differences between the two pedestrians are computed on the basis of the motion parameters stored inside the internal database of one of the two pedestrians involved. The effectiveness of VAM Redundancy Mitigation thus depends on how accurate the information stored inside the LDM are with respect to the real position, speed and heading of the pedestrian involved at the time in which redundancy mitigation shall be performed. For this reason, we performed a simulation, using the previously mentioned scenario, measuring, each time VAM redundancy mitigation is started, the difference between the position, speed and heading differences between the two pedestrians using the data stored in the LDM of one of them and the same differences measured using the actual motion parameters of the two pedestrians retrieved using TraCI and we obtained the data shown in table 5.1.

As we can see from table 5.1, these errors are quite small if compared to the values

Error on position difference	Error on speed difference	Error on heading difference
0.19 m	0.08 m/s	0.05 degrees

Table 5.1. Errors on motion parameters for VAM Redundancy Mitigation

of the thresholds used for VAM Redundancy Mitigation, which are, as previously mentioned, 4 m for the position difference, 0.5 m/s for the speed difference and 4 degrees for the heading difference, therefore we can affirm that the decision, made by the VRU ITS-Stations, on whether or not to perform VAM Redundancy Mitigation relies on reliable data.
When a VRU enters VAM Redundancy Mitigation, it stays silent for a certain amount of time and, since in this time interval the VRU cannot exchange information with other road users, it is subject to greater risk. In particular, it can be of interest the average distance travelled by a pedestrian between two consecutive VAM transmissions when VAM Redundancy Mitigation is active and when it is not, since, the larger this distance, the greater the risk for the pedestrian. In order to retrieve this distance, we took advantage of the two pedestrians scenario previously described and we measured the difference between the current position of the ego-VRU and the position included in the last VAM transmitted every time a new VAM is generated. We then computed the average difference between these two positions throughout the whole simulation and we performed two simulations, one with VAM Redundancy Mitigation and the other one without it. The results obtained are shown in figure 5.16

As it was expected, the average distance travelled by the pedestrian between two consecutive VAM generations increases when activating VAM Redundancy Mitigation, it almost doubles, since several VAM transmissions are skipped due to this latter and, as a consequence, the average time interval between two consecutive VAM transmissions increases.

5.2 Hardware in the loop

Before testing the VAM communication protocol on the field through the OScar project, discussed in chapter 4.2, we needed to cross-compile this latter in order to be able to run it on the On-Board Unit described in chapter 4.1. Other than this, in order to be sure that the OScar project was working properly, we also decided to test it using a fake GPS trace generated using the virtual simulation tool (*vsim*) by Cohda Wireless, in such a way to perform some measurements before going on the field allowing us to understand if the code on which OScar is based was behaving as expected.

5.2.1 Cross-compilation on OBU

The first thing to do was to cross-compile the code in the OScar project in order to make it run on the OBU which was used to perform the field tests, which runs the OpenWrt-V2X software. This was due to the fact that the OBU has a completely different architecture

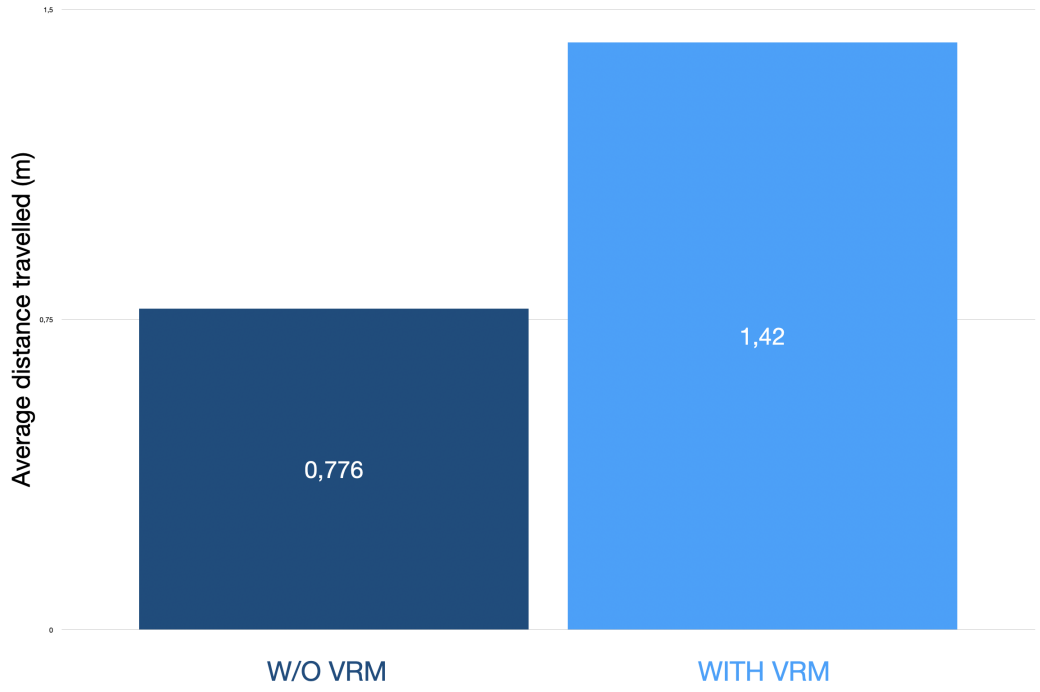

Figure 5.16. Average distance travelled by a pedestrian between two consecutive VAM generations with and without VAM Redundancy Mitigation

compared to the one of a standard PC and therefore requires executable files built for a x86_64 architecture and standard library C musl, which is a lighter version of the standard library C glibc which we commonly find on standard computers. While OScar can be easily compiled on the PC using the command *make*, the process of cross-compiling it on the OBU is slightly more complicated. Instead of using *make*, we must use the command *make compileAPU_gpsd* to cross-compile the code on the OBU architecture, however, before doing this, we must set a specific toolchain, containing all the tools and compilers needed to cross-compile the project on our PC, making it ready for the OBU. In order to properly set the toolchain, we first needed to download on our PC a patched version of OpenWrt-V2X and then make some modifications to the path in order to make it ready for use. At this point, the OScar Makefile was slightly modified in order to set the proper cross-compilers and update the name of the target.
Once the cross-compilation toolchain and the Makefile were ready, we could finally build OScar with the *compileAPU_gpsd* target, generating the binary file to be downloaded on the OBU.

5.2.2 Simulation using *vsim*

Once the OScar project was ready to be downloaded and run on our OBU, we tested it using a fake trace generated using Cohda Wireless *vsim* tool.
First of all, we built a custom traffic scenario, located in Turin and characterised by the

presence of one pedestrian only, since we were only interested in understanding if the code was working as expected and if VAMs were disseminated correctly, therefore we did not need a complex scenario with many road users of different types. For what concerns the map we used for building this traffic scenario, we decided to choose a location with few turns and long straights, since, as it will be shown in chapter 5.3, it is similar to the paths chosen for the real world tests. Figure 5.17 shows the aforementioned traffic scenario, where the orange path corresponds to the path travelled by the simulated pedestrian. Unfortunately, the *vsim* tool does not include a model for pedestrians yet, therefore we had to resort to the model of a car travelling with a very low speed, specifically 5 km/h.

Figure 5.17. OScar test traffic scenario

Considering the traffic scenario shown in figure 5.17 and the fact that the simulated pedestrian travels with a constant speed of 5 km/h, we expect almost all VAM transmissions to be triggered by the position triggering condition and the remaining ones by the heading triggering condition, due to the three turns located along the path. We expect not to register any VAM transmission triggered by the speed, time and safe distances conditions, since the simulated pedestrian travels with a constant speed, large enough not to ever make 5 seconds pass between two consecutive position triggered VAM transmissions, and it is alone in the traffic scenario. Figure 5.18 shows the distribution among the five aforementioned triggering conditions of the VAMs transmitted during the simulation.

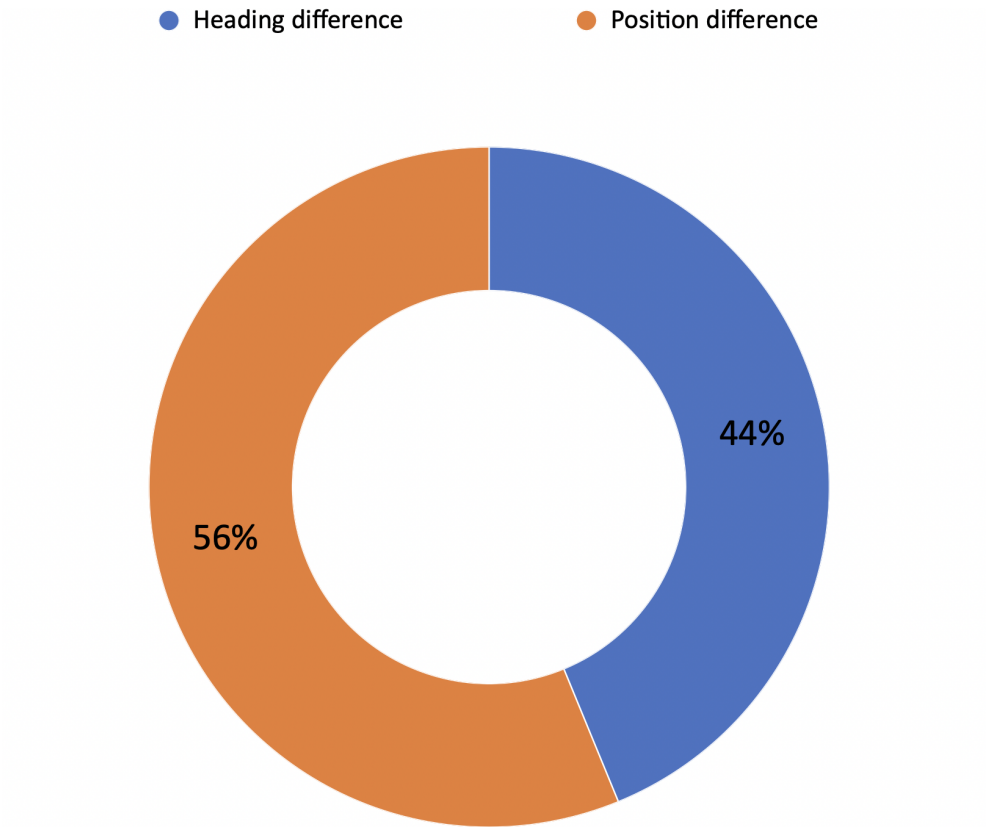

Figure 5.18. VAM transmissions percentage distribution with respect to triggering conditions in *vsim* test

As it was expected, there are no VAM transmissions triggered by the speed, time and safe distances conditions, however the distribution of the transmitted VAMs among the other two triggering conditions, namely the position and the heading ones, is not what we expected, as heading triggered transmissions are far too many if compared to the position triggered ones, considering the topology of the path travelled by the pedestrian.
The reason why the distribution of transmitted VAMs among their triggering conditions is the one shown in figure 5.18 is to be looked for in how heading triggered transmissions are distributed throughout the simulation. In particular, if we look carefully at the path shown in figure 5.17, specifically at the three turns, we can notice that the pedestrian is performing quite rounded turns, therefore we can assume that its heading, while performing these turns, slowly changes of about 90 degrees before stabilising on a new value. In order to verify our assumption, we performed a very simple simulation, using the same scenario presented above, but, in this case, we started the simulation with the pedestrian located just before the first turn and we ended it as soon as the pedestrian had finished it, storing on a log file all the VAMs triggered by the heading triggering condition and some related information. The first part of this log file is shown in figure 5.19.

As we can see from the content of this log file, all VAMs transmitted, which are

```
[VAM] VAM sent
[LOG] Timestamp=1700585917309 VAMSent=true Motivation=heading NUMVAMsSent=1 HeadDiff=8.500000 PosDiff=0.613367 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=26.900000 CurrHead=35.400000 HeadDiff=8.500000

[VAM] VAM sent
[LOG] Timestamp=1700585917410 VAMSent=true Motivation=heading NUMVAMsSent=2 HeadDiff=7.300000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=37.900000 CurrHead=45.200000 HeadDiff=7.300000


[VAM] VAM sent
[LOG] Timestamp=1700585917809 VAMSent=true Motivation=heading NUMVAMsSent=3 HeadDiff=4.900000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=46.400000 CurrHead=51.300000 HeadDiff=4.900000


[VAM] VAM sent
[LOG] Timestamp=1700585918209 VAMSent=true Motivation=heading NUMVAMsSent=4 HeadDiff=4.900000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=51.300000 CurrHead=56.200000 HeadDiff=4.900000

[VAM] VAM sent
[LOG] Timestamp=1700585918609 VAMSent=true Motivation=heading NUMVAMsSent=5 HeadDiff=4.900000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=56.200000 CurrHead=61.100000 HeadDiff=4.900000

[VAM] VAM sent
[LOG] Timestamp=1700585919009 VAMSent=true Motivation=heading NUMVAMsSent=6 HeadDiff=4.900000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=61.100000 CurrHead=66.000000 HeadDiff=4.900000

[VAM] VAM sent
[LOG] Timestamp=1700585919409 VAMSent=true Motivation=heading NUMVAMsSent=7 HeadDiff=4.900000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=66.000000 CurrHead=70.900000 HeadDiff=4.900000

[VAM] VAM sent
[LOG] Timestamp=1700585919909 VAMSent=true Motivation=heading NUMVAMsSent=8 HeadDiff=4.900000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=70.900000 CurrHead=75.800000 HeadDiff=4.900000

[VAM] VAM sent
[LOG] Timestamp=1700585920309 VAMSent=true Motivation=heading NUMVAMsSent=9 HeadDiff=4.900000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=75.800000 CurrHead=80.700000 HeadDiff=4.900000

[VAM] VAM sent
[LOG] Timestamp=1700585920709 VAMSent=true Motivation=heading NUMVAMsSent=10 HeadDiff=4.100000 PosDiff=0.000000 SpeedDiff=0.000000 TimeDiff=0
[HEADING] HeadingUnavailable=360.100006 PrevHead=80.700000 CurrHead=84.800000 HeadDiff=4.100000
```

Figure 5.19. Log file *vsim* test for checking the heading triggering condition

21 at the end of the turn, are in this case triggered by the heading condition and the *Timestamp* and *NUMVAMsSent* fields, which represent respectively the time instant corresponding to the VAM transmission and the number of VAMs sent since the beginning of the simulation, prove that these heading triggered VAMs are sent one after the other, at a very short distance of time from each other and the heading variations with respect to the previous transmission are quite small, most of the times around 4 degrees, which is exactly what we assumed.
We can then conclude that the *vsim* simulation confirms that the code on which OScar is based works properly and the excessive number of heading triggered VAMs we obtain during the simulation are simply due to the rounded turns designed by the Cohda Wireless *vsim* tool, which means that they shouldn't be a problem during the field tests.

5.3 Field tests

Once the OScar project had been suitably cross compiled and downloaded on the OBU and its correct functioning had been verified through the Cohda Wireless *vsim* simulation, we were ready to test it on the field, in a real world scenario. The goal of these field tests was to verify the behaviour of the VAM communication protocol in a real world environment and, in order to do this, we conducted some tests storing all the information related to the transmitted VAMs on log files, which were then analysed to understand how VAM transmissions were distributed among the five implemented triggering conditions which were described in chapter 2.6.2.

The location chosen for the tests, shown in figure 5.20, was the neighborhood around the main site of Politecnico di Torino and all the tests were conducted with one OBU only disseminating VAMs, since the purpose of these real world tests was mainly to verify that the dissemination of VAMs was working as expected and that VAM transmissions were distributed among the different triggering conditions in a similar way to what was obtained through the simulations on ms-van3t (5.1.3).

Figure 5.20. Location field tests

5.3.1 Setup of the equipment

First of all, we needed to set up the equipment used for conducting the tests. This equipment consists in one On-Board Unit, made up of the APU2 board and a GNSS device, described in chapter 4.1, an external supply used to power the OBU and a GNSS receiver. All these devices were installed on a stroller, which was pushed around acting as a pedestrian disseminating VAMs.

The OBU shown in figure 5.21 was running the OpenWrt-V2X software and the OScar

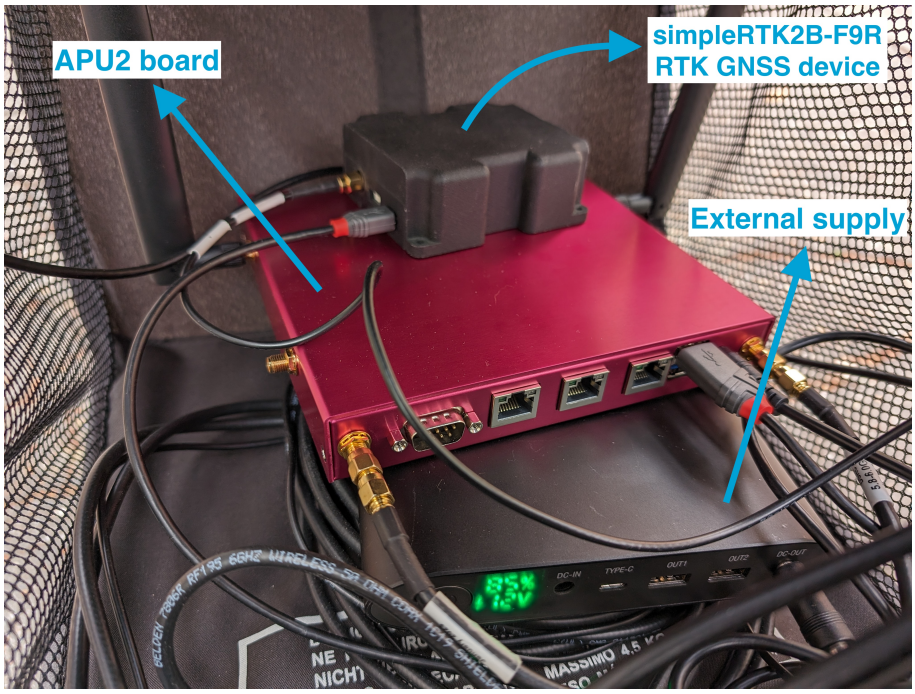

Figure 5.21. OBU setup

Figure 5.22. GNSS antenna

project code was downloaded on it in such a way to make it able to transmit and receive different types of ITS messages, namely CAMs and VAMs. For what concerns the GNSS antenna shown in figure 5.22, a u-blox antenna by ArduSimple was used. We set the

Figure 5.23. Stroller setup

antenna to work with the pedestrian model developed by u-blox, in such a way to obtain an as accurate as possible GNSS trace allowing the VAM communication protocol to work properly and effectively.

5.3.2 Tests and results

We will now present the field tests which were conducted and we will analyse the results derived from them. As it was previously mentioned, we focused on analysing the distribution of VAM transmissions among the different triggering conditions foreseen by ETSI, keeping in mind that our VBS currently supports only 5 out of the 7 triggering conditions presented in chapter 2.6.2. In this specific case, despite all these 5 triggering conditions are implemented in the VBS included in the OScar project, we will only analyse four of them, since all tests were conducted using one OBU only disseminating VAMs, therefore the safe distances triggering condition could never be satisfied. The same applies to VAM Redundancy Mitigation, which is implemented in OScar, but it could not be tested during the field tests since it implies the presence of at least two road users, which means at least two OBUs disseminating ITS messages.
All tests performed were analysed retrieving a map showing the exact location of each VAM transmission, allowing us to extract the path which was followed during the test, and building two diagrams, in particular a histogram and a pie chart, showing the distribution of the transmitted VAMs among the different conditions which triggered their transmission.

First of all, we performed a static test, in order to check that the equipment was working and that everything was installed correctly on the stroller. In particular, we were interested in verifying that the OBU was disseminating VAMs and that the GNSS data acquired by the antenna were meaningful. Clearly, since, in this case, our VRU, namely the stroller, was not moving, we expected to only have time triggered VAM transmissions every 5 seconds. The results obtained are shown in figures 5.24, 5.25 and 5.26.

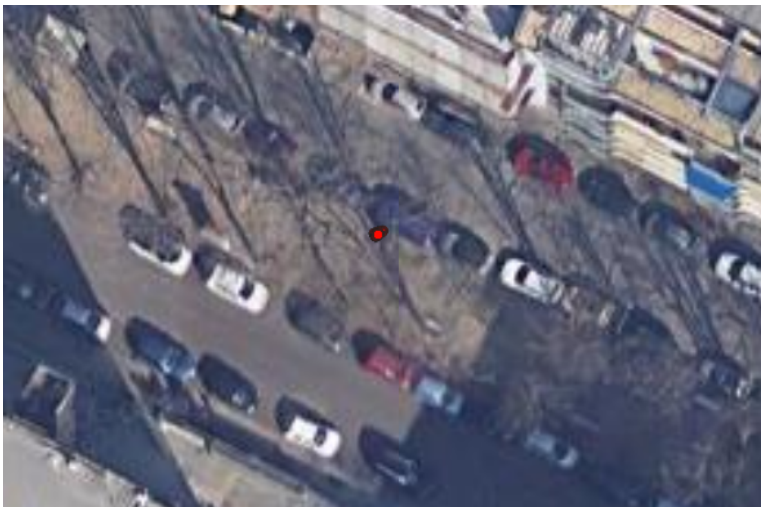

Figure 5.24. GNSS trace - static test

As we can see from figure 5.24, the GNSS trace is very accurate, since all dots representing VAM transmissions are overlapped, as you would expect from a static dissemination, and figures 5.25 and 5.26 show that almost all VAM transmission are triggered by the time condition every 5 seconds, just like we expected. The only thing we did not expect are the few heading triggered transmissions we can see in the two diagrams. These erroneous transmissions are due to fluctuations of the orientation acquired by the GNSS antenna, probably due to the fact that, even though we are using the u-blox pedestrian model, the accuracy of the signal acquired by the antenna decreases when the speed is too low and, in this case, our VRU is not moving at all.
This hypothesis is confirmed by the fact that, if we conduct a quick test in which our VRU travels on a straight line, setting the u-blox model of our antenna to be the vehicular one, we obtain a massive amount of VAM transmissions triggered by the heading condition every 100 ms, as we can see in figures 5.28 and 5.29. Also the GNSS trace, in this case, is quite inaccurate, as shown in figure 5.27, even though the path travelled was almost perfectly straight.
The vehicular model turns out to be extremely inaccurate at very low speeds, as proved by the fact that the measured orientation keeps changing continuously and also the GNSS trace is very inaccurate, since it is designed for vehicles and therefore it must work well at higher speeds. We can then affirm that also the problem identified with the static test can be addressed to an inaccurate functioning of the model implemented on the GNSS antenna when the VRU is standing still, which means that it should be a minor problem

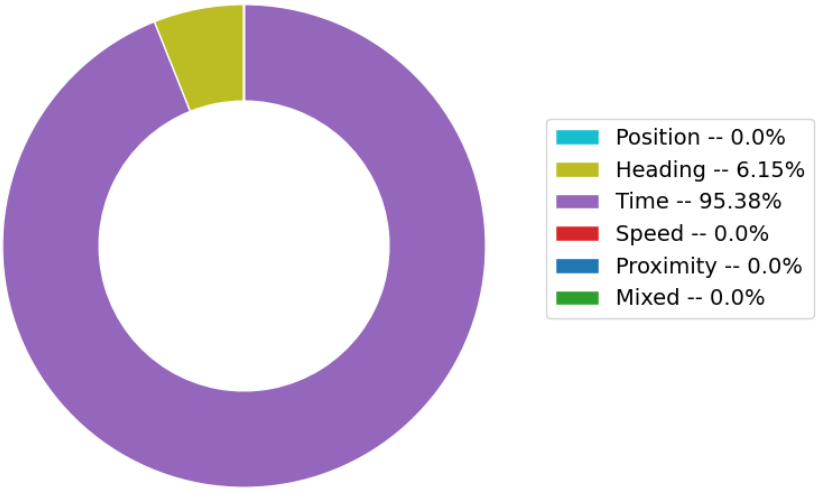

Figure 5.25. VAM transmissions percentage distribution with respect to triggering conditions - static test

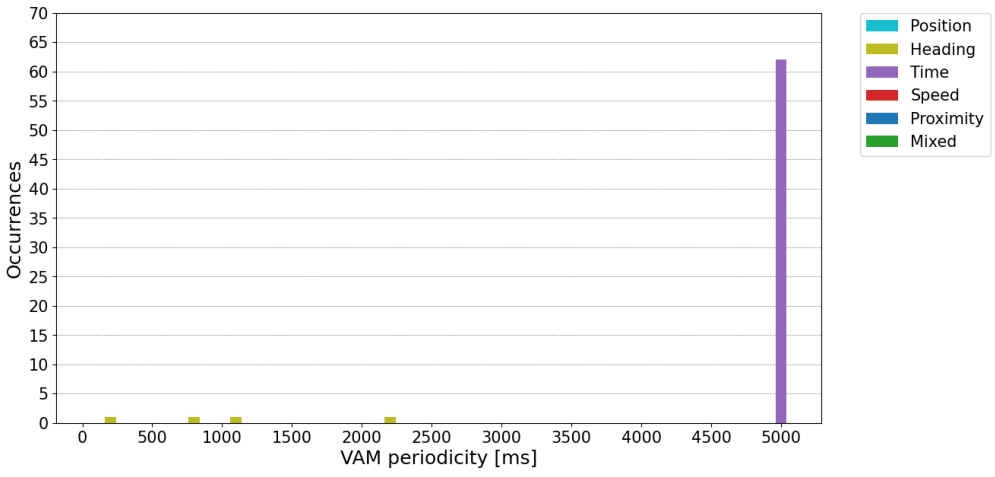

Figure 5.26. VAM transmissions occurrences - static test

when conducting the tests with the moving VRU.

We can now start analysing the three tests aimed at studying the distribution of

Figure 5.27. GNSS trace - vehicular model test

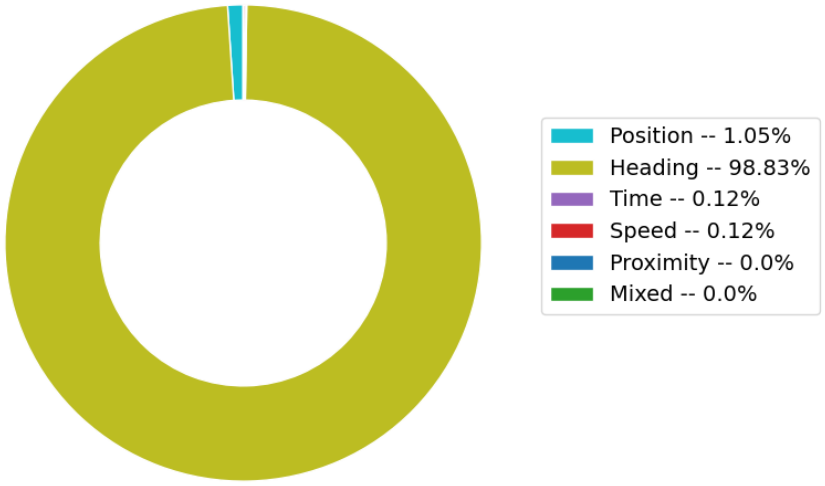

Figure 5.28. VAM transmissions percentage distribution with respect to triggering conditions - vehicular model test

VAM transmissions among the conditions triggering them in a real world scenario. Since, by doing the static and vehicular model tests, we noticed that there can be accidental variations of the heading acquired by the GNSS antenna, each one of these three tests

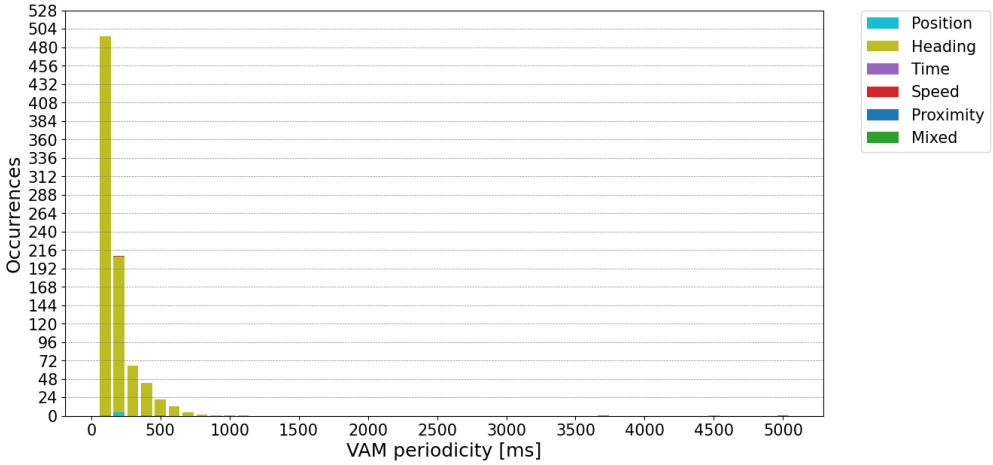

Figure 5.29. VAM transmissions occurrences - vehicular model test

was performed twice, changing the heading triggering condition threshold. As discussed in chapter 2.6.2 and as presented in standard ETSI TS 103 300-3 V2.2.1 [12], ETSI proposes some recommended values as triggering conditions thresholds, but it leaves the opportunity to increase them if necessary. For this reason, we first tried to conduct each one of these tests using 4 degrees as triggering condition threshold as suggested by ETSI and then we repeated the same test increasing the threshold to 10 degrees and we compared the results obtained in the two cases.
The first test we performed consisted in our VRU travelling on a straight path, a very similar scenario to the one of the vehicular model test. Figures 5.30, 5.31 and 5.32 show the results we obtained with heading triggering condition threshold set to 4 degrees, while figures 5.33, 5.34 and 5.35 show what happens when changing the aforementioned threshold to 10 degrees.

If we compare the results obtained from the two tests, we can notice that, obviously, the number of heading triggered VAMs decreases considerably when changing the heading triggering condition threshold from 4 degrees to 10 degrees and, as a consequence, the number of position triggered VAMs increases. Even though the number of heading triggered VAM transmissions decreases a lot when increasing the associated threshold, as we can easily deduce by comparing figures 5.31 and 5.34, we still have 2 VAM transmissions triggered by the heading condition. These two heading triggered transmissions can be either due to some actual heading changes along the path, which is in general straight, however it is still possible to have some small heading changes causing these 2 VAM transmissions, or by some rare accidental fluctuations in the orientation retrieved by the GNSS antenna. A part from this, the few speed triggered VAMs we obtained from both tests are due to random speed changes of the VRU during the test, which, most of the times, depend on external conditions which are not under our control, and we can also notice that both GNSS traces are quite accurate, which means that the pedestrian

Figure 5.30. GNSS trace - straight path test with heading threshold set to 4 degrees

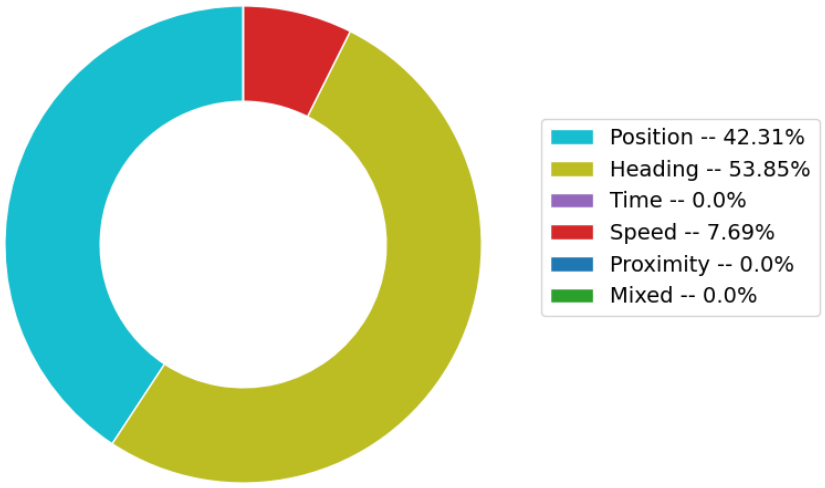

Figure 5.31. VAM transmissions percentage distribution with respect to triggering conditions - straight path test with heading threshold set to 4 degrees

u-blox model works much better when the VRU is moving.
The second and third tests consisted respectively in a shorter path and a longer one across the neighborhood around Politecnico di Torino. The paths of the two versions of

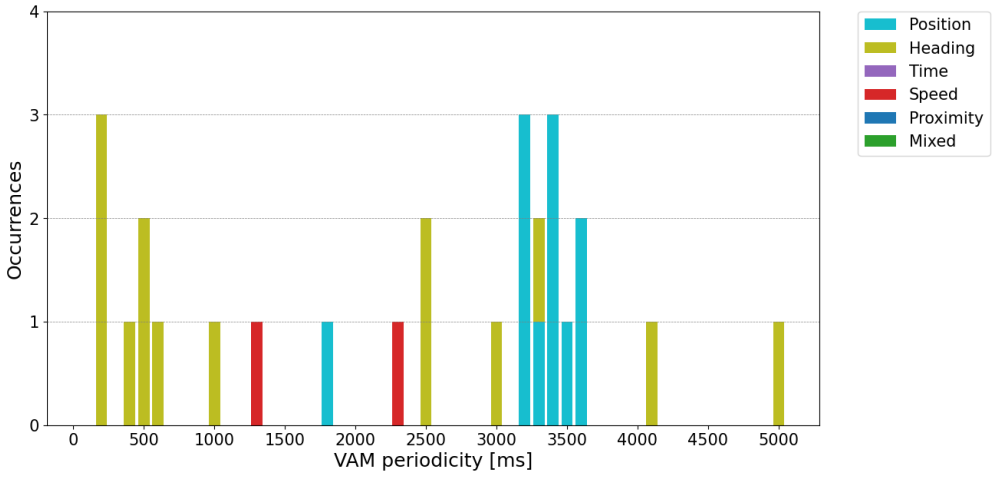

Figure 5.32. VAM transmissions occurrences - straight path test with heading threshold set to 4 degrees

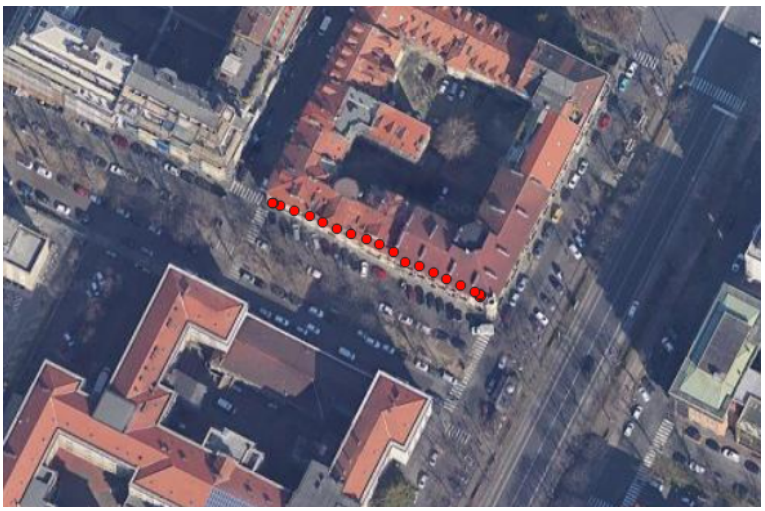

Figure 5.33. GNSS trace - straight path test with heading threshold set to 10 degrees

the short path test are shown in figures 5.36 and 5.39, while the paths of the two long path tests in figures 5.42 and 5.45, figures 5.37 and 5.38 for the short path test and 5.43 and 5.44 for the long path test show the results obtained from the two tests conducted with heading triggering threshold set to 4 degrees and figures 5.40 and 5.41 for the short path test and 5.46 and 5.47, on the other hand, show the same results with heading triggering threshold set to 10 degrees.

In both cases, as we can easily notice by comparing figure 5.37 with figure 5.40 and

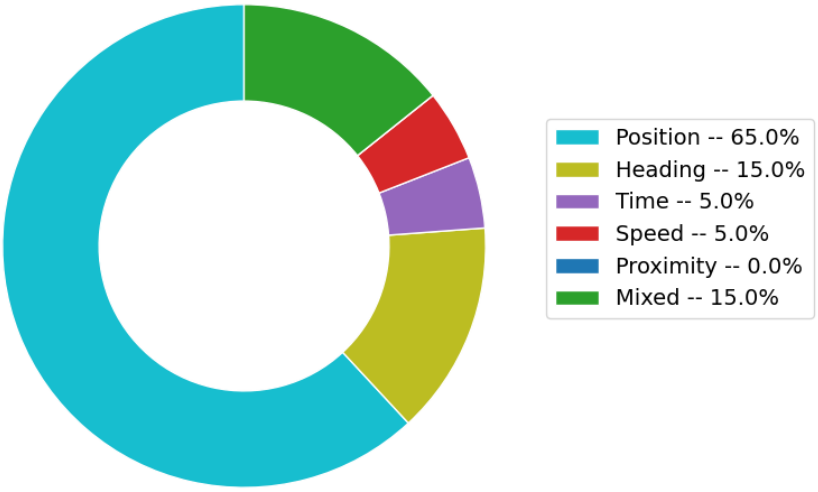

Figure 5.34. VAM transmissions percentage distribution with respect to triggering conditions - straight path test with heading threshold set to 10 degrees

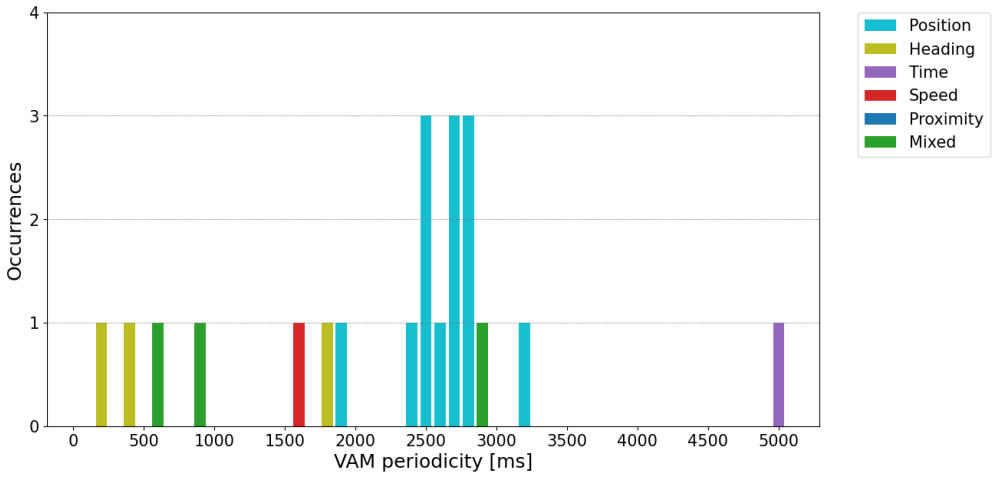

Figure 5.35. VAM transmissions occurrences - straight path test with heading threshold set to 10 degrees

figure 5.43 with figure 5.46, the increase of the threshold associated to the heading triggering condition from 4 degrees to 10 degrees causes the number of VAM transmission

Figure 5.36. GNSS trace - short round path test with heading threshold set to 4 degrees

triggered by the heading condition to decrease and, consequently, the number of transmissions triggered by the position condition to increase. With respect to the straight path test, these last two tests are more significant in terms of distribution of VAM transmissions among the different triggering conditions, since they are longer in terms of time duration and therefore more VAMs are disseminated during the tests and, above all, the paths travelled by our VRU are more complex and therefore the VAM communication protocol is stressed a bit more.
By looking at the results obtained from these field tests we can therefore conclude that the recommended values of the triggering conditions thresholds defined by ETSI are all reasonable, however, when the heading triggering condition threshold is set at 4 degrees, the number of heading triggered VAMs seems to be a bit too high. For this reason, it can be a good idea to slightly increase this threshold, around 7 to 10 degrees, which is allowed by ETSI TS 103 300-3 V2.2.1 [12], in order to limit as much as possible the number of VAMs transmitted because of accidental heading variations, without impacting the transmission of heading triggered VAMs when the VRU actually performs a turn.

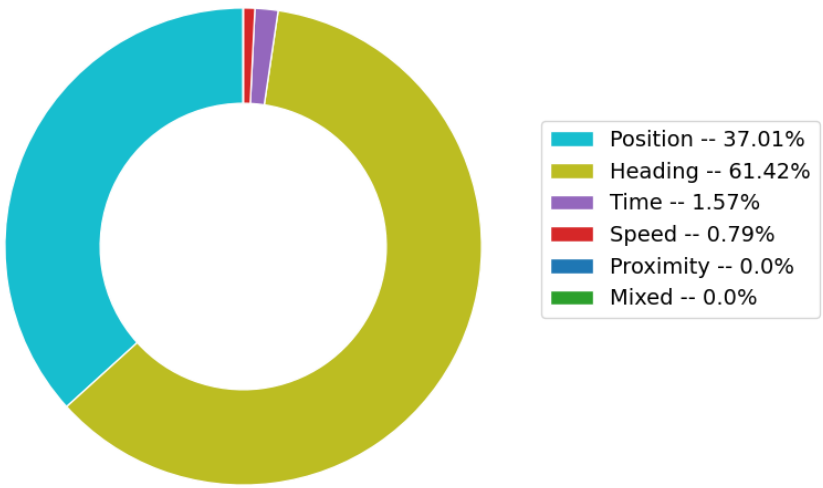

Figure 5.37. VAM transmissions percentage distribution with respect to triggering conditions - short round path test with heading threshold set to 4 degrees

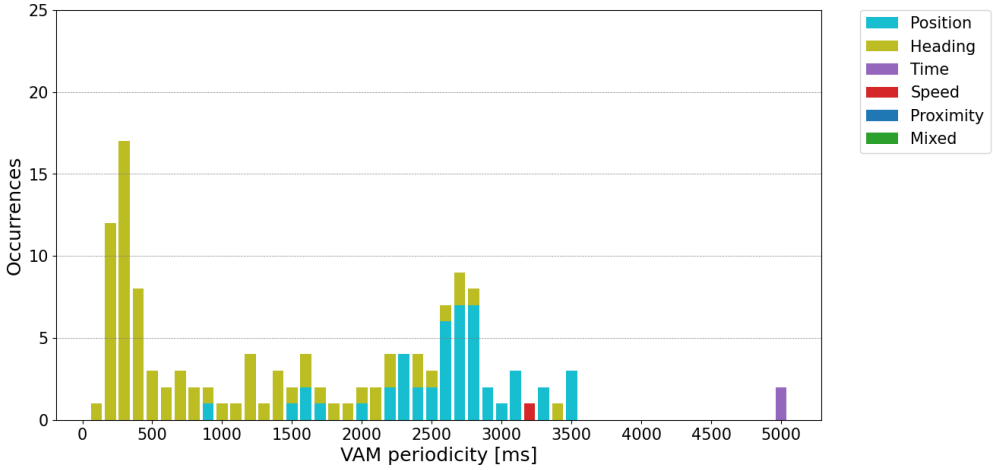

Figure 5.38. VAM transmissions occurrences - short round path test with heading threshold set to 4 degrees

Figure 5.39. GNSS trace - short round path test with heading threshold set to 10 degrees

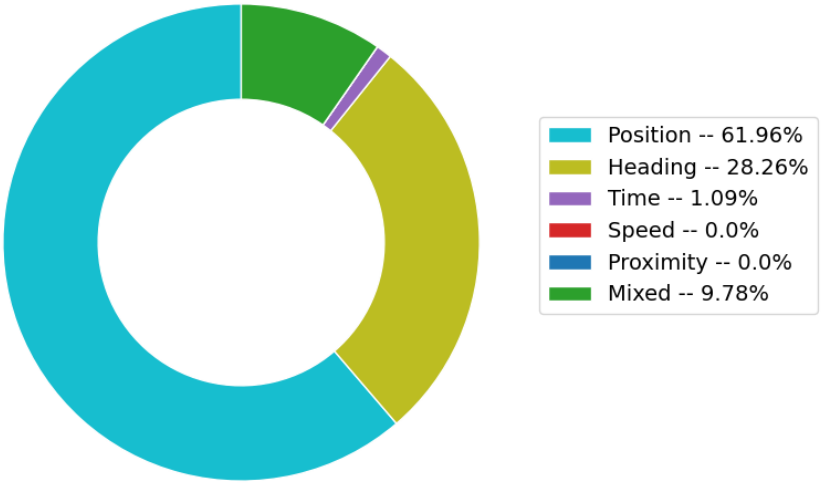

Figure 5.40. VAM transmissions percentage distribution with respect to triggering conditions - short round path test with heading threshold set to 10 degrees

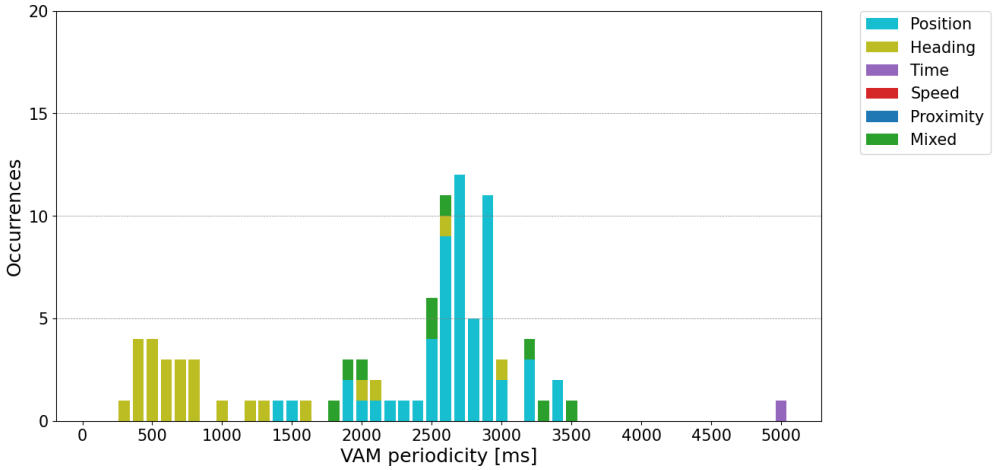

Figure 5.41. VAM transmissions occurrences - short round path test with heading threshold set to 10 degrees

Figure 5.42. GNSS trace - long round path test with heading threshold set to 4 degrees

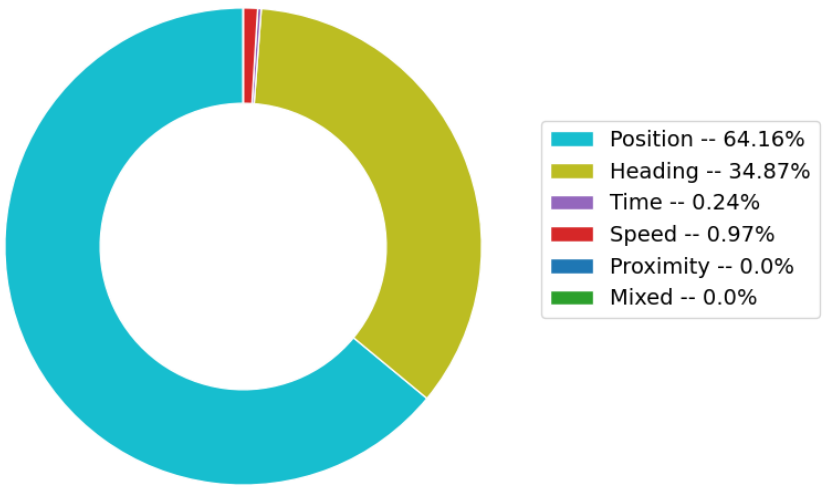

Figure 5.43. VAM transmissions percentage distribution with respect to triggering conditions - long round path test with heading threshold set to 4 degrees

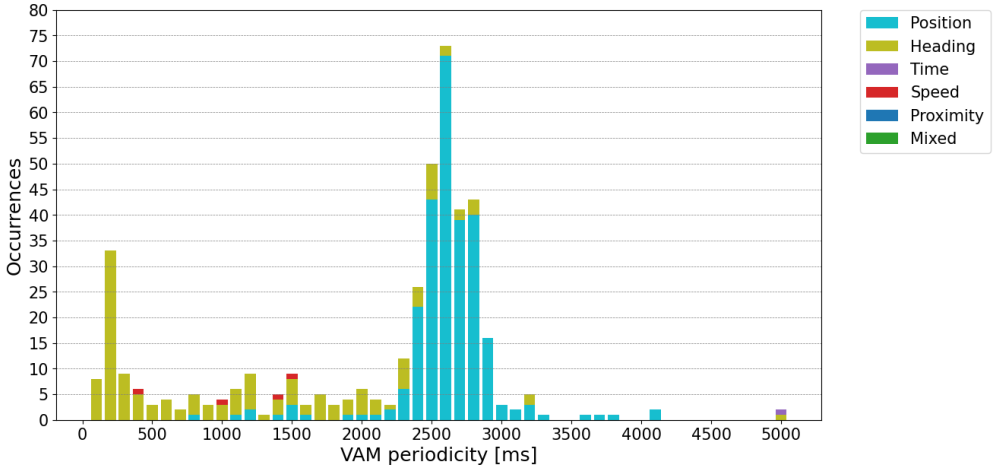

Figure 5.44. VAM transmissions occurrences - long round path test with heading threshold set to 4 degrees

Figure 5.45. GNSS trace - long round path test with heading threshold set to 10 degrees

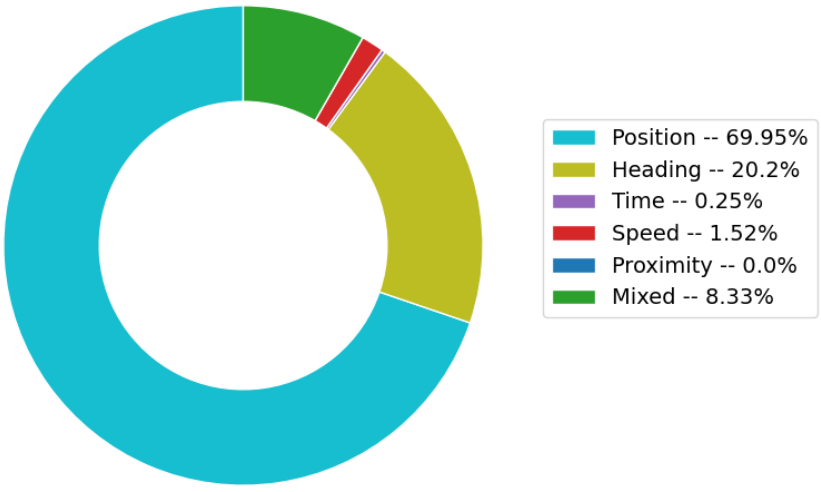

Figure 5.46. VAM transmissions percentage distribution with respect to triggering conditions - long round path test with heading threshold set to 10 degrees

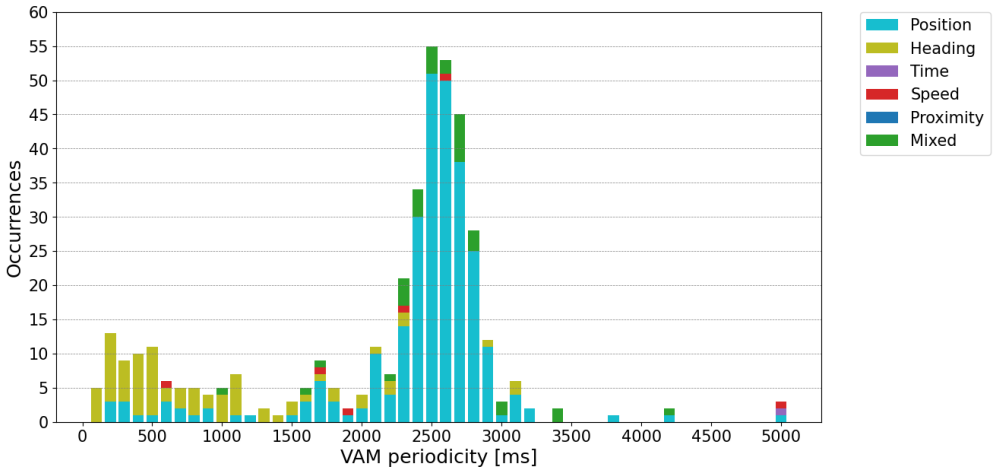

Figure 5.47. VAM transmissions occurrences - long round path test with heading threshold set to 10 degrees

Chapter 6

Conclusion

In the last few years the automotive industry and the world revolving around it have experienced an incredible evolution in terms of technological innovation and automation, with the emergence of the first communication standards related to connected cars and, more in general, to connected road users. These intelligent road users are typically equipped with devices allowing them to perceive, at least partially, the traffic scenario surrounding them and therefore to always be aware of what is happening around them.
An important category of road users are the so called Vulnerable Road Users (VRUs), which are traffic participants like pedestrians and cyclists that, in case of collision, are the ones experiencing the greatest dangers. Specifically, VRUs are defined as those road users which are not equipped with a motor and which are not protected by an outer shield. In the last few years, several institutions have started developing communication protocols allowing them to exchange information with other road users and, in order to do this, they must be equipped with devices allowing them to share their main attribute and status information.
If we have a look at the data by WHO and similar institutions, each year tens of millions of people are involved in road accidents and many of them are VRUs, which are many times involved in fatal accidents due to their vulnerable nature. For this reason, communication protocols allowing these road users to be aware of the traffic scenario surrounding them and, above all, to become an active part of this scenario, making other traffic participants like vehicles aware of them, is of paramount importance to reduce the number of road accidents involving VRUs and thus to save human lives.
With this aim in mind, ETSI has developed a communication protocol, standardised in documents ETSI TS 103 300-2 V2.2.1 [10] and ETSI TS 103 300-3 V2.2.1 [12], allowing VRUs to share their attribute and status information with the road users surrounding them. The ITS messages disseminated by VRUs are called VAMs and they contain the most important information related to the originating VRU ITS-Station.
According to the standard defined by ETSI, VRUs can be equipped in such a way that they can only transmit or receive messages or in such a way that they can do both and they can be classified according to four profiles: the first one involves pedestrians, the second one cyclists and similar, the third one motorcyclists and the fourth one all animals which might present a danger to other road users. When there are several VRUs close

to each other with similar speed and heading, they can be grouped in clusters, which are characterised by the fact that only the VRU which is leader of the cluster can disseminate VAMs, all other members have to remain silent. VRU clusters are particularly useful since, when the number of VRUs located in a certain area is very high, they can help reducing the amount of load affecting the channel without impacting the awareness of VRUs.
A VRU system can be defined as a system made up of at least one VRU and one ITS-S with a VRU application installed and all its functionalities shall be centred around the VRU Basic Service, which is the facilities layer entity responsible for the transmission and reception of VAMs, as well as for handling the role of the VRU, which can either be VRU_ROLE_ON or VRU_ROLE_OFF. On top of this, the VBS is also in charge of identifying whether the VRU is part of a cluster, in fact the VBS of a VRU ITS-S is characterised by four cluster states, which determine the ability of a VRU to transmit and receive ITS messages.
Just like all other types of intelligent transportation systems, VRUs can receive all types of ITS messages, however they can only disseminate VAMs. VAMs are made up of four mandatory fields, which are the ITS PDU header, the generation delta time field, the basic container and the high frequency container, and four optional containers, which are the low frequency, cluster information, cluster operation and motion prediction ones. The four mandatory fields are clearly the most important ones, since they include the most important status and attribute information about the originating VRU ITS-S.
VAMs transmission has a variable frequency, as the minimum time which must elapse between two consecutive VAM generations corresponds to a parameter called T_GenVam, which has a value that must always be higher or equal than 100 ms and lower or equal than 5 s. The actual time interval elapsing between two consecutive transmissions depends on the VAM triggering conditions, which shall always be checked every $T\_CheckVamGen$, that must always be equal or lower than 100 ms. These triggering conditions are related to the type of VAM, which can either be an individual or a cluster VAM. For what concerns individual VAMs, which are the ones that were analysed in detail in this document, there are seven conditions which can trigger the generation and dissemination of a new VAM, the first one is related to the time elapsed since the last VAM transmission, then there are three conditions related respectively to the position, speed and heading difference with respect to the values of the same parameters included in the last VAM transmitted, then we have one condition related to trajectory interception probability with another road user and one condition related to the joining of a cluster and the last triggering condition is related to the longitudinal, lateral and vertical safe distances that a VRU needs to have with respect to another road user. All these triggering conditions are characterised by recommended predefined thresholds. It is also important to highlight that there are some conditions that, when satisfied, cause the VRU to skip a predefined number of VAM transmissions. This phenomenon is called VAM Redundancy Mitigation and its aim is to prevent the communication channel to be flooded with unnecessary messages without impacting the awareness and safety of VRUs.
In order to test how the standard works and its effectiveness, we first implemented it on ms-van3t, a simulator for vehicular networks, and tested in a simulated traffic scenario

and then we implemented it in a project designed to run on an OBU to test it on the field, in a real world environment.
For what concerns the implementation of the standard on both ms-van3t and the OBU, we implemented a slightly simplified version of the communication protocol, since the aim was to test its basic functionalities and its main features, leaving the most complex ones for further study. Being this said, we implemented on ms-van3t an almost complete version of the VRU Basic Service, a part from the clustering functionality, the triggering conditions related to VRU clusters and to trajectory interception probability and the optional containers and we tested it using a custom made traffic scenario built on SUMO, measuring metrics like the PRR and latency of VAMs and comparing the obtained results with those of CAMs, whose basic service was already fully implemented on the simulator, and retrieving the distribution of VAM transmissions across the simulation among the different triggering conditions, in such a way to understand whether the associated thresholds defined by ETSI have reasonable recommended values. We then adapted the VBS code to be run on the architecture of our OBU, thus building the OScar project, a C++ open source implementation of the CA and VRU Basic Services, which was used to perform some field tests aimed at proving the correct functioning of the VAM communication protocol in a real world scenario, retrieving also in this case the distribution of VAM transmissions among the associated triggering conditions.
The average PRR and latency measures suggest that CAMs and VAMs have very similar performances and show us how these two parameters, in case of VAMs, are influenced by the transmission power, the data rate and the number of pedestrians present in the traffic scenario. On the other hand, if we analyse the results obtained in terms of VAM transmissions distribution among the associated triggering conditions, we can notice that we obtain quite similar results comparing the results derived from the simulations and those derived from the field tests, taking also into account that the safe distances condition could only be tested in the simulated scenario. In particular, we can see that, in both cases, the position condition is the one triggering most VAM transmissions, followed by the heading one and then all the others. For what concerns the heading triggering condition, we did not notice any problem when performing the simulations, however, during the field tests, we have noticed that there can be fluctuations in the orientation measured by the GNSS antenna, due to a number of different factors, which can cause accidental transmissions. For this reason, even though the results obtained through the simulations suggest that the recommended minimum value proposed by ETSI for the heading triggering condition threshold is appropriate, following the evaluation of the results derived from the field tests, we feel quite comfortable in suggesting to slightly increase the value used as heading triggering condition threshold, in such a way to filter out the accidental transmissions due to fluctuations of the measured orientation of the VRU. All the other thresholds, on the other hand, seem to have suitable minimum values.

Bibliography

[1] [online] ms-van3t. https://github.com/marcomali/ms-van3t.

[2] [online] ns-3 manual; release ns-3-dev; ns-3 poject. https://www.nsnam.org/docs/release/3.40/manual/ns-3-manual.pdf.

[3] [online] ns-3 tutorial; release ns-3-dev; ns-3 poject. https://www.nsnam.org/docs/release/3.40/tutorial/ns-3-tutorial.pdf.

[4] [online] openstreemap. https://www.openstreetmap.org/#map=6/42.088/12.564.

[5] [online] point-to-multipoint communication. https://www.techopedia.com/definition/26762/point-to-multipoint-communication-pmp#.

[6] Vito Battista. Development of infrastructure-to-vehicle module for vehicular communication emulation through the ms-van3t framework. 2021.

[7] C2C-CC. Use cases. Use case, CAR 2 CAR Communication Consortium, 2023.

[8] Orlean G. Dela Cruz. Managing road traffic accidents: A review on its contributing factors. In *IOP Conference Series: Earth and Environmental Science*, 2021.

[9] ETSI. ETSI EN 302 637-2 V1.4.1 (2019-04) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. Standard ETSI EN 302 637-2 V1.4.1, European Telecommunications Standards Institute, 2019.

[10] ETSI. ETSI TS 103 300-2 V2.2.1 (2021-04) - Intelligent Transport Systems (ITS); Vulnerable Road Users (VRU) awareness; Part 2: Functional Architecture and Requirements definition; Release 2. Standard ETSI TS 103 300-2 V2.2.1, European Telecommunications Standards Institute, 2021.

[11] ETSI. ETSI TR 103 300-1 V2.3.1 (2022-11) - Intelligent Transport Systems (ITS); Vulnerable Road Users (VRU) awareness; Part 1: Use Cases Definition; Release 2. Standard ETSI TR 103 300-1 V2.3.1, European Telecommunications Standards Institute, 2022.

[12] ETSI. ETSI TS 103 300-3 V2.2.1 (2023-02) - Intelligent Transport Systems (ITS); Vulnerable Road Users (VRU) awareness; Part 3: Specification of VRU awareness basic service; Release 2. Standard ETSI TS 103 300-3 V2.2.1, European Telecommunications Standards Institute, 2023.

[13] Meng Lu Kees Wevers. V2x communication for its - from ieee 802.11p towards 5g. In *IEEE 5G Tech Focus*, volume 1, 2017.

[14] Yangxin Lin, Ping Wang, and Meng Ma. Intelligent transportation system(its): Concept, challenge and opportunity. In *2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high*

performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids), pages 167–172, 2017.

[15] Marco Malinverno. Safety applications and measurement tools for connected vehicles. pages 17–20, 2021.

[16] NSC. Vulnerable Road Users. Position statement, National Safety Council, 2018.

[17] [online] Francesco Raviglione. Aim-automotiveintegratedmap. https://github.com/francescoraves483/AIM-AutomotiveIntegratedMap.

[18] [online] Francesco Raviglione. Ocabs-project. https://github.com/francescoraves483/OCABS-project.

[19] [online] Pablo Alvarez Lopez Michael Behrisch Laura Bieker-Walz Jakob Erdmann Yun-Pang Flötteröd Robert Hilbrich Leonhard Lücken Johannes Rummel Peter Wagner Evamarie Wießner. Microscopic traffic simulation using sumo. IEEE Intelligent Transportation Systems Conference (ITSC), 2018. https://sumo.dlr.de/docs/index.html.

[20] [online] Pablo Alvarez Lopez Michael Behrisch Laura Bieker-Walz Jakob Erdmann Yun-Pang Flötteröd Robert Hilbrich Leonhard Lücken Johannes Rummel Peter Wagner Evamarie Wießner. Microscopic traffic simulation using sumo. IEEE Intelligent Transportation Systems Conference (ITSC), 2018. https://sumo.dlr.de/docs/TraCI.html.

[21] [online] Pablo Alvarez Lopez Michael Behrisch Laura Bieker-Walz Jakob Erdmann Yun-Pang Flötteröd Robert Hilbrich Leonhard Lücken Johannes Rummel Peter Wagner Evamarie Wießner. Microscopic traffic simulation using sumo. IEEE Intelligent Transportation Systems Conference (ITSC), 2018. https://sumo.dlr.de/docs/netconvert.html.

[22] [online] Pablo Alvarez Lopez Michael Behrisch Laura Bieker-Walz Jakob Erdmann Yun-Pang Flötteröd Robert Hilbrich Leonhard Lücken Johannes Rummel Peter Wagner Evamarie Wießner. Microscopic traffic simulation using sumo. IEEE Intelligent Transportation Systems Conference (ITSC), 2018. https://sumo.dlr.de/docs/Netedit/index.html.

[23] Francesco Raviglione. Open platforms for connected vehicles. 2022.

[24] SAE. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Standard J3016, Society of Automotive Engineers, 2021.