Doctoral Dissertation
Doctoral Program in Electric, Electronic and Communication Engineering (XXXIV cycle)

Development and Performance Evaluation of Urban Mobility Applications and Services

Marco Rapelli
* * * * *

Supervisors
Prof. Claudio Ettore Casetti, Supervisor
Prof. Falko Dressler, Co-supervisor

Politecnico di Torino
December, 2021

. . . .

Marco Rapelli

Turin, December, 2021

Summary

Urban Mobility studies are necessary for the examination of metropolitan traffic patterns, the evaluation of new traffic policies, the testing of future vehicular applications, and the construction of efficient public transportation networks, among other purposes. 5G mobile network technology has promoted the creation of services with ultra-low latency requirements, which has facilitated, among other areas, the development of automotive applications. Concepts that have been investigated for several years, such as self-driving automobiles, vehicular communication, and real-time mobility services, are now on their way to becoming a reality. Mobility services for real-time traffic information and navigation systems play critical roles in vehicular applications, and they are becoming increasingly popular.

Mobility simulations are required for all automotive applications in order to evaluate the efficacy and reliability of the systems proposed. As is often the case, the larger the data set, the better. As a consequence, numerous projects in the disciplines of mobility and vehicular communication have sought new traffic simulators with expanded investigation regions, which could include an entire city and its surrounding suburbs. This thesis introduces TuST (Turin SUMO Traffic), a city-wide simulator for modeling traffic in Turin that makes use of the SUMO (Simulation of Urban Mobility) tool. The whole collection of vehicle traces collected over the course of 24 hours was created using TuST and made publicly available for other investigations. In fact, any vehicular simulation relies on a realistic mobility environment in order to prove its reliability and precision.

Vehicular applications that could benefit from the use of realistic traces from the TuST simulator include those that investigate micro-mobility at junctions. V^3TL (Vehicle-to-Vehicle Virtual Traffic Light) is a model of this type presented in this thesis for scheduling priority at uncontrolled intersections. The service offered by V^3TL can be offered to self-driving and connected cars at crossings that are not controlled by a traffic light in a totally distributed way. Applications based on Vehicle-to-Network (V2N) communication are another example of mobility services that might be envisioned in the near future for any connected car. In such scenarios, the flow of messages created by any automotive entity must be processed and dispatched by a cloud node, which is often a Broker server. In this thesis, we illustrate the technique that was used for the development of two use cases that

make use of V2N communication in real-world testing settings.

Besides automotive applications, urban mobility studies are essential for local municipalities in order to offer suitable infrastructure and services to their citizens. Knowing the number and shape of customer flows permits a local authority to correctly scale an already existing service or to propose a new one that is tailored to the needs of the community. By installing a specialized sensor on a public bus, we were able to construct an Automatic Passenger Counting System (APCS). By utilizing a complex software procedure, we are able to provide GTT, the Turin public transportation agency, with a real-time counting system that is useful for determining ideal bus frequencies on public transportation lines and for complying with the capacity limits imposed by COVID-19 restrictions. Eventually, we examined the prospect of creating a new application that would be targeted at pedestrians who were walking through a train station or metro station. Specifically, the model described in this thesis is a content sharing system that operates in a micro cloud environment. As a result of cashback or discounts offered by the transportation authority or by the store owners, we demonstrate how users may distribute information that is beneficial to others, such as train timetables and service delays, as well as special deals offered by coffee shops and stores.

Acknowledgements

This work was fully supported by the interdisciplinary and interdepartmental center F*UL*L, the Future *Urban Legacy* Lab of Politecnico di Torino.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The term Urban Mobility means the planning and regulation of traffic flows on private and public transport networks. The first heartbeat of Urban Mobility could be dated back to September 27^{th}, 1825, when the first line of the Stockton & Darlington Railway was officially inaugurated. It was the first railway in the world open to public passenger transport, connecting collieries near Shildon with Darlington and Stockton-on-Tees. In the same year, Gurney Goldsworthy patented the first steam bus, able to carry 15 people at a speed of 24 Km/h. In 1827, Walter Hancock improved the first patent and established several bus lines in England from 1831 to 1839. In 1839, the competition from railways and coachmen, under the pretense of a fatal accident, obtained a law prohibiting buses. It would have to wait until 1895 to see the first motor bus line between Siegen and Netphen in Germany. Some years later, in 1913, Henry Ford installed the first assembly line for the mass production of an entire automobile, establishing the first step for the creation of the modern private transportation network.

During the last century, the explosive growth of motor vehicles has been experienced in a number of major cities. Cars, buses, railways and trams have become a consistent part of our everyday lives. Like in past centuries, private and public transportation will also shape our future lifestyle, leveraging the overwhelming technological progress of the communications industry. Wired and wireless technologies have paved the way for Vehicular Ad-Hoc Networks (VANETs). Extremely dynamic topology, ultra low latency requirements and high bandwidth demand make VANETs more than a vehicular version of existing Mobile Ad-Hoc Networks (MANETs). The major standardization bodies created new protocols and solutions to meet the stringent requirements of vehicular communications. Dedicated standards gave birth to the entire set of applications enabling wireless data exchange between vehicles and their surroundings, namely Vehicle-to-Everything (V2X) communications. Depending on the particular field of application, V2X technologies can be separated into different sub-cases. Vehicle-to-Vehicle (V2V) refers to inter-vehicle communications; Vehicle-to-Infrastructure (V2I) refers to data exchanged between cars and Road Side Units (RSUs); Vehicle-to-Pedestrian (V2P) identifies communications between vehicles and Vulnerable Road Users (VRUs), usually via smartphones; Vehicle-to-Network (V2N) is for connecting the vehicle to all the set of network services and applications.

Each type of vehicular communication has to follow specific protocols defined by communication standards. The European Telecommunication Standards Institute (ETSI) and the Institute of Electrical and Electronics Engineers (IEEE) are the main standardization entities for the definition of guidelines in vehicular communication. Both Intelligent Transport System (ITS)-5G, proposed by ETSI, and the IEEE Wireless Access for Vehicular Environment (WAVE) are based on the IEEE 802.11p access layer, a dedicated WiFi solution for the vehicular environment. As an opposite solution, Cellular-V2X (C-V2X) was recently proposed by the Third Generation Partnership Project (3GPP), leveraging the emerging Long Term Evolution (LTE) 4G and the new 5G facilities.

From the definition of C-V2X, stakeholders in vehicular research started comparing cellular-based and WiFi-based solutions in terms of costs, latency, throughput, scalability and future integration. As is often the case, these studies are performed with simulation tools. Vehicular simulations reduce costs for application testing and are able to reproduce both small-scale scenarios with high levels of detail and urban-scale models with millions of vehicle trips in the span of a day. Good simulation results are paramount for having the chance of an on-field test. The more realistic the car traces used for testing vehicular applications, the more relevant the results are. Unfortunately, simulated vehicle traces resembling reality are extremely difficult to find online and large-scale simulators able to reproduce them are rare and complex objects.

In this thesis, an urban-scale simulator, named TuST (Turin SUMO Traffic), was developed using the SUMO (Simulation of Urban Mobility) tool. TuST covers a 600-Km^2 area in and around the Metropolitan City of Turin. Vehicular traces generated are validated with real traffic sensors from 5T Torino info-mobility agency and are available online [1]. In the present work, we also describe other mobility projects that can benefit from realistic vehicular traces and traffic output models from TuST. In particular, a V2V Virtual Traffic Light system, called V^{3}TL, is an example of a vehicular application defined in the context of this thesis for scheduling vehicles at intersections. In the scope of a 5GAA (5G Automotive Association) demo project and of the Rainbow European project, Advanced Message Queuing Protocol (AMQP) was used for instantiating a message Broker server.

Apart from vehicular applications, other mobility flows have had a great impact in recent years on research regarding Urban Mobility. Due to social distancing and restrictions, the COVID-19 pandemic focused great attention on pedestrian flow applications. Nevertheless, realistic pedestrian traces are even harder to find than vehicular ones, mostly due to privacy issues and collecting methodologies. In this thesis, an Automatic Passenger Counting System (APCS) is described. APCS was used on a public bus to count the number of passengers on board in real time. A second application relying on the micro cloud environment makes use of pedestrian traces to simulate a content items sharing model in a Stockholm metro station.

1.1 Research motivation and objectives

The exponential growth in vehicular technologies has caused an evolutionary development in transport policies, associated in particular with the identification of new urban paradigms and mobility problems. Mobility studies were always conducted in order to improve existing infrastructures and facilities according to the increasing demand. Nevertheless, in recent years, mobility studies are becoming more popular than before, since mobility issues have been a core part of all the main topics of the last two decades, as detailed below.

The beginning of 21^{st} century saw an overwhelming growth in all the information technology fields. The spread of smartphones in our everyday lives changed completely the paradigms of information searching and content providing, reshaped the web according to new evolving schemes and defined the current era of big data. This digital revolution benefits almost all fields involving content items providing and communication technologies. The upcoming demand for digital services defines new standards in terms of throughput, bandwidth and latency. In order to meet the requirements, standardization institutes for digital communication established new protocols for mobile data exchange, namely Legacy 3G and 4G/LTE. More recently, 5G mobile network technology has enabled the development of applications with ultra-low latency requirements, facilitating, among others, vehicular applications. Concepts studied for several years, like self-driving cars, vehicular communication and real-time mobility services, are now upcoming realities. Among vehicular applications, mobility services for real-time traffic information and navigation systems play roles of particular importance.

A different emerging topic involving urban mobility is the one regarding green mobility. In the past five years, there has been a general growing awareness of issues related to climate change and eco-sustainability. Several protests and strikes, known as "*Fridays for Future*", resulted in international agreements for stricter limits on emissions. According to the Paris Declaration on Electro-Mobility and Climate Change and Call to Action [2], transport contributes to 23% of global energy-related greenhouse gas (GHG) emissions. Considering this, thanks to the Paris Climate Accords, the American Environmental Protection Agency (EPA) and the European Commission pushed car makers to respect strict limits for vehicle emissions and to improve the research and development of electric vehicles. Along with the slow and long-term diffusion of electric vehicles, urban mobility studies could significantly help in monitoring high-emissions areas and in proposing applications and schemes for reducing vehicular pollutants. Toll-granted access policies, vehicle sharing applications and dedicated car scheduling algorithms at intersections are a few examples of urban mobility studies addressing this topic.

Eventually, due to the recent COVID-19 pandemic, new mobility issues related to social distancing and capacity constraints on public transport have gained attention in the past two years. Monitoring pedestrian flows in public areas and counting the number of passengers on a bus were two mandatory solutions adopted after the general lockdown. Pedestrian flow analysis based on person-counting sensors is a valid technological proposal for solving the problem. It is to be noticed how COVID-19 restrictions put emphasis on an already existing topic. Monitoring the number of passengers on board is fundamental for a public transportation agency in order to shape line services and transport frequencies. In a similar way, inspecting people flows in some public areas of interest (e.g., train and metro stations, airports, etc.) helps to define the network capabilities in the area and may result in dedicated content-sharing applications with contextual information.

The present thesis aims at developing innovative solutions for urban mobility issues, covering all the main thematics described above. It also provides new software tools and frameworks capable of speeding up the management and solving of urban mobility problems.

1.2 Main contributions

The main purpose of this thesis is to describe the methodology and performance evaluation used in different projects covering urban mobility aspects. Firstly, the process for the design of an urban-scale mobility simulator is described, focusing on the city of Turin. Real traffic data from vehicle counters under the road surface was used for the validation process. Vehicle traces over a 24 hour span were extracted from the simulator and are available online [1] for future research. Secondly, a vehicular application for scheduling traffic at unregulated intersections is presented. Leveraging inter-vehicular communication, we were able to develop a procedure that was able to maximize the number of passing cars for each phase and minimize the number of stop-and-go maneuvers. In order to compare our results, we simulated a scenario with a traffic-light-regulated intersection and a scenario where vehicles cross the junction in an unregulated way. Then, an AMQP Broker service is described and it is used in the context of two vehicular projects, leveraging V2N communication. In the first one, an Urban Geo-referenced Alert (UGA) system was developed with real-time traffic information. The second project describes the development of an Urban Mobility Demonstrator (UMD) for road hazard real-time notification. After that, the hardware and software solutions are reported for the development of an Automatic Passenger Counting System (APCS). We also described the interface with the bus network and the real-time mechanism of our APCS. Eventually, a micro cloud application for a content items sharing application is reported. In the context of this project, we studied an indoor scenario with pedestrian simulated traces from a metro station in Stockholm. Further details about the main contributions and the topics covered in this work are presented below.

Design, development and validation of an urban-scale simulator

As the number of motor vehicles in metropolitan areas increases, more and more mobility studies are required in order to meet the growing demand for services. In a modern environment, traffic simulations are standard procedure for studying urban patterns, testing new traffic policies, and evaluating new vehicular technologies. As is frequently the case, a large data set is beneficial in many situations. Numerous initiatives in the domains of mobility and vehicular communication have taken advantage of new traffic simulators with expanded investigation regions, which can cover an entire city and its surrounding suburbs. A 600-Km^2 simulator in and around the Municipality of Turin has been modelled with the help of the SUMO tool. Using real-time traffic data obtained from the 5T info-mobility agency, a validation procedure was carried out. The results demonstrate that an urban traffic simulator spanning such a large region may be constructed at the price of modest simplifications while still reaching a high degree of accuracy.

Within the urban-scale simulator modeling area, our contributions can be found in:

- [3] *Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. "Vehicular Traffic Simulation in the City of Turin from Raw Data".* Published in *IEEE Transactions on Mobile Computing (2021).*
  Available at *https://ieeexplore.ieee.org/document/9416842*

- [4] *Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. "TuST: from raw data to vehicular traffic simulation in Turin".* Published in *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT). IEEE. 2019, pp. 1–8.*
  Available at *https://ieeexplore.ieee.org/document/8958652*

Development of a V2V communication-based procedure for scheduling vehicles at unregulated intersections

Among the key goals of urban policies, reducing traffic congestion and pollutant emissions are definitely at the top of the list. In past years, solutions leveraging vehicular communication, such as GLOSA, have been proposed in order to reduce traffic at intersections with traffic lights. However, cities usually have a large number of unregulated intersections where queues can build up, increasing pollutants caused by the stop-and-go motion of cars. The problem is further worsened by the future presence of self-driven vehicles, which will need to be coordinated with other self-driven and normal cars at junctions. In the present work, we developed V^{3}TL, a Vehicle-to-Vehicle (V2V) Virtual Traffic Light system for infrastructure-less unregulated junctions. Our goal is to provide a low-complexity, yet effective algorithm and communication procedure for letting cars at an unregulated crossroads decide if a virtual traffic light is needed. If this is the case, vehicles will self-organize in order to establish one. The results presented demonstrate significant improvements compared to both unregulated and traffic light-based junctions. For testing the performance of V^{3}TL, various realistic scenarios were used, including isolated or consecutive three- and four-way junctions in single- and multi-lane configurations. Moreover, different generation rates were used, obtaining improvements in terms of the number of passing cars per minute, the number of stop-and-go maneuvers and scheduling fairness with respect to all comparison scenarios.

In this field, our research activity can be found in:

- [5] *Ahmadreza Jame, Marco Rapelli, and Claudio Casetti. "Reducing Pollutant Emissions through Virtual Traffic Lights".* Submitted and currently under review in *Computer Communications (2022).*
- [6] *Marco Rapelli, Claudio Casetti, and Marcello Sgarbi. "A Distributed V2V-Based Virtual Traffic Light System".* Published in *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS). IEEE. 2020, pp. 122–128. doi:10.1109/COMSNETS48256.2020.9027339.17*
  Available at *https://ieeexplore.ieee.org/document/9027339*

Design of V2N communication-based applications in real test case scenarios

The low penetration rate for inter-vehicular communication technologies makes it necessary for the development of applications relying on a single On Board Unit (OBU) device. The only applications able to meet this requirement are the ones exploiting Vehicle-to-Network (V2N) communication, using the so-called Broker servers for managing message dispatching through the network. Although a standard has not been issued yet for these kinds of transmissions, past studies and projects suggest the Advanced Message Queuing Protocol (AMQP) as the state of the art for managing a Broker server. In this chapter, we focus on the setting of an AMQP Broker server and its application in two real test cases: an Urban Georeferenced Alert (UGA) for a navigation system with real-time traffic information and an Urban Mobility Demonstrator (UMD) for a real-time hazard notification message through the network. Moreover, we also give a detailed description of our online platform used for visualizing traffic information and hazard notifications on a map in real time.

Development of an Automatic Passenger Counting System for monitoring bus capacity in real-time

The spread of the COVID-19 pandemic has made it necessary to use some restrictive measures in order to contain the contagion. In particular, after general lockdown, governments had to face the problem of how to get everyone back to their normal routine in total safety and without increasing the number of daily infections. The solution adopted to avoid people gatherings on public transport was to introduce capacity constraints on public buses and metro trains. In this chapter, we describe how we designed an Automatic Passenger Counting System (APCS) to monitor the capacity of a public bus operated by Turin Transport Group GTT (Gruppo Torinese Trasporti) agency. Our APCS is able to identify the number of people on board by analyzing WiFi probe requests from smartphones and extracting a unique fingerprint from them. Privacy issues were also considered in the context of this project. A dedicated results section shows how it is possible to achieve a good passenger count estimation with a good level of approximation.

Content sharing application in a pedestrian-based micro cloud: design and performance evaluation

The ever-increasing urban population, as well as the exorbitant development and maintenance expenses of infrastructure-based techniques, necessitate the usage of distributed solutions. Micro clouds models, particularly for automotive and pedestrian applications, are getting more and more popular among distributed systems. In this project, we designed a distributed application based on micro clouds to distribute content items in an indoor pedestrian environment. In accordance with the findings, it is possible to accomplish 100% content item dissemination while also reducing channel collisions by 66%.

Within this context, our contribution can be found in:

- [7] *Marco Rapelli, Gurjashan Singh Pannu, Falko Dressler, and Claudio Casetti. "Content Sharing in Pedestrian-based Micro Clouds".* Submitted and currently under review in *2022 IEEE 95th Vehicular Technology Conference (VTC2022). IEEE. 2022, pp. 1-7.*

1.3 Outline of the thesis

The present thesis is organized as a collection of projects. In the following, every chapter is dedicated to a different project. In particular:

Chapter 2 introduces our large-scale simulator with a detailed explanation of the selected use case and of the methodology followed in order to build it. A dedicated section reports results and the validation process.

Chapter 3 describes the V2V Virtual Traffic Light, V^{3}TL. We reported the complete procedure description, as well as the pseudo-code of the system and of the scheduling algorithm. In the results section, we compared our proposed solution with two test case scenarios: an unregulated intersection and a junction controlled by a traffic light. For every one of them, we also studied performances with different generation flows and with different intersection types.

Chapter 4 gives an initial overview of the AMQP protocol and Broker servers. Then, we give a description of the methodology to follow for creating an online urban mobility platform. Finally, the two use cases are presented with the details of the two projects where V2N communication was instantiated in real-world tests.

Chapter 5 focuses on our APCS model. An initial description of smartphone fingerprints and probe requests is followed by the software and hardware details of our system. Eventually, the results of real-time bus monitoring are given.

Chapter 6 firstly presents the description of the use case we selected for our pedestrian-based micro cloud. Then, the methodology is followed by the result section.

Chapter 7 aims at summarizing the work and the contributions presented in this manuscript.

Chapter 2

Modeling Vehicular Traffic in the City of Turin

In August 2019, a research from the Texas A&M Transportation Institute [8] assessed the amount of time the average American commuter spends travelling through congested traffic. According to the findings, 54 extra hours per year were spent in traffic jams and delays. If you drive at jammed speeds rather than free-flowing rates for an additional hour, you are reported to have traveled an “extra hour”. Even though some American cities have more jammed traffic than European ones, the average traffic congestion in the United States is similar to that of the vast majority of large cities throughout the world. Additionally, with the continued increase in the number of people living in metropolitan areas, it is projected that traffic congestion will worsen. In order to meet existing and future traffic demands, urban planners are responsible for building sustainable transportation and appropriate infrastructure systems. The most common method for accomplishing this goal is to adopt large-scale traffic simulators, such as the one described in this chapter.

2.1 Research motivation

Simulation is the instrument of choice for performing any kind of urban mobility study since it can create a comprehensive image of ever expanding regions by utilizing more powerful computing resources and multiple sources of information. A more traditional application of simulated urban models could be the analysis of traffic patterns and the identification of residential and industrial districts for the purpose of a socio-demographic study, as performed in [9]. Such studies are incredibly significant for urban planners because they allow them to identify the key arteries that connect residential districts and work areas and determine whether they should be improved or replaced entirely. Furthermore, traffic simulators are the sole instruments available for doing predictive mobility analysis. In this spirit, the authors have already developed a model that is similar to the one presented in this chapter for the feasibility study of the Turin Limited Traffic Zone (Torino ZTL), which is a component of the Torino Centro Aperto project, commissioned by the Municipality of Turin in 2018 [10]. Taking inspiration from the previously described project, the current investigation aims to build a scenario on the scale of a whole city.

Aside from mobility analysis, large-scale urban models could be utilized effectively for city-wide pollution research by simply simulating the emission class of vehicles in the simulator, as has been conducted in [11]. Possible outcomes of a simulation-based studies include the identification of areas of likely congestion where air-quality sensors should be installed, or a rough analysis of pollution caused by traffic as opposed to other sources, when evaluating data from real samples.

Last, but not least, is the fact that urban traffic models are essential for the testing of vehicular applications. Since road testing of new automotive technology in support of ADAS (Advanced Driver Assistance Systems) is often prohibitively expensive and potentially dangerous, a major percentage of the testing process is carried out using simulated tests, such as those described in [12]. Keeping this in mind, the selection of the simulated mobility environment is critical for success. Since there are no large-scale traffic simulators available, application developers frequently choose a small use case map with only a couple of intersections at most, where interactions among a large number of vehicles and their repercussions on traffic in a large urban environment are hardly realistic. Large-scale vehicle testing, on the other hand, always produces more thorough and informative data, as was the case in [13].

As a result, it is evident why a traffic model on an urban scale is required. However, a complicated, comprehensive, large-scale model of a city is a challenging work, which may explain why there are so few instances of urban-scale traffic models in the literature. Also crucial to note is that each city has its own unique topographical, demographic, public transportation and mobility features, which must be taken into account. Comparisons of the performance of models of different cities are of limited significance, even when comparing urban regions of similar size and population. As an example, almost all avenues have service roads running alongside them, which are primarily used for parking or by local businesses and are not found in existing large-scale models (such as the ones discussed in the following section of the literature review), necessitating special consideration in the modeling effort. For example, almost all avenues have service roads running alongside them, which are primarily used for parking or by local businesses. The presence of these service roads has an impact on both the way junctions are simulated and the distribution of trips begins and finishes. For this reason, we believe that expanding the current literature on urban traffic models to include more cities is particularly significant.

2.2 Main contributions

The current study, whose contributions can be found in [3] and [4], introduces TuST (Turin SUMO Traffic), a detailed large-scale model of the Turin metropolitan region and its bordering districts during a 24-hour period, leveraging the SUMO tool. It is the goal of this work to contribute to the topic of how to establish an open-source, large-scale urban simulator by offering a full explanation of its design and proving that creating a model of such complexity is indeed achievable. 5T [14], a local company specializing in ITS and Info-mobility for its shareholders, which include the Municipality of Turin, the Metropolitan City of Turin, and the Piemonte Region, provided aggregated traffic and street sensor data, which were used for traffic demand construction and system validation, respectively. Several aspects of our work contribute to the advancement of the current literature:

- the construction of a large-scale simulator for testing a 24-hour scenario is described in depth in this study;
- it takes as input actual traffic data from a real Origin/Destination (O/D) data set along with real traffic light phases from the local traffic agency;
- it provides publicly available simulated vehicle traces from a full-day scenario [1] that has been executed through simulations; in this way, traces can be utilized as input for more complex mobility models or for the development of vehicle-to-vehicle (V2V) communication scenarios based on real-time traffic data;
- it compares simulation results with real-time traffic data from street sensors provided by the local traffic authority; this provides an objective measure of "realism" and reveals that the model we constructed is a very realistic large-scale simulator of traffic in Turin;
- it describes a new procedure to incorporate data from traffic-light-regulated intersections into the model;
- it includes a thorough explanation of the adjustments and refinements that we made to our model in order to get more reliable results and a more accurate portrayal of full-scale daily traffic flow.

In the next sections, we perform a literature review analysis and describe the initial data set and the use case selected for the purpose of this study. Then, a full and detailed description of the methodology used is given. Results, validation procedure and final remarks conclude the chapter.

2.3 Related works

There are just a few instances of traffic simulators in the literature that can be considered large-scale models. The TAPASCologne data set [15], which is an open-source project that covers the broader metropolitan area of the city of Köln in Germany, is an example of a large-scale urban traffic simulator that models a 400-Km^2 region with a total of 1.2 million individual journeys in a single day. Because of its great level of realism, it is considered to be one of the best examples of large-scale simulators currently available. Another example of an urban mobility model is the traffic simulator in Bologna, Italy [16]. It simulates a typical daily traffic pattern (22,000 cars during the morning rush-hour), focusing on a 28-Km^2 region surrounding the city center.

Among the most recent traffic models to be published, we have LuST (Luxembourg SUMO Traffic) [17], which covers highways and main roads in Luxembourg City, as well as MoST (Monaco SUMO Traffic) [18], which is focused on the Principality of Monaco and surrounding regions. Using an original data set of 14,000 cars gathered over 16 hours, the LuST scenario covers a 156-Km^2 territory and simulates morning and afternoon traffic in the region. MoST, on the other hand, mimicked 35,000 automobiles during the morning rush hour. The ITS Austria West scenario [19] is a significantly bigger model, consisting of a map with around 245,000 nodes and 320,000 edges, with a total road length of 27,000 Km. The ITS Austria West scenario has 1.2 million routes and 1.6 million cars over the course of a full day, making it one of the largest scenarios developed. However, the data set is not publicly available to the research community since part of the information it depends on is proprietary.

Additional publications have recently introduced other mobility models. En Route [20] proposes a macro-mobility model for the city of London. The authors created an O/D matrix using synthetic data from cameras in London and real taxi traces from other locations. The simulation was performed over the span of two hours and included 30,000 vehicle trips. Another example comes from a Shanghai 800-Km^2 simulator [21]. In this study, 13,750 real taxi traces were utilized in a macro-mobility study to build an Origin/Destination matrix of 17,7 million journeys for a multi-day scenario. As an outcome, a SUMO simulator was constructed, with vehicles being inspected at different times of the day: 65,000 vehicles on a weekday and 60,000 vehicles on a weekend day.

The projects discussed in this section reflect all of the major large-scale traffic models that are currently in use. Our research intends to contribute to the existing literature by offering a more complex and comprehensive model. Many of the research papers previously listed used synthetic traffic data to generate their O/D matrices. Only a few of them were able to create a simulator that could analyze a full-day situation. In order to offer a quantitative assessment of the magnitude of our model, 2,200,000 automobile trips are simulated over the course of 24 hours, making it one of the most extensive instances in the literature for a large-scale microscopic traffic simulator, as stated in the official SUMO wiki [22]. If compared to the ITS Austria West scenario, our model takes into account about twice as many vehicles in a much smaller region, resulting in a significantly higher level of complexity.

2.4 Initial data set and case study

5T is a private company owned by local governmental entities (the City of Turin, the Metropolitan City of Turin, and the Piemonte Region) that focuses on intelligent transportation systems (ITS) and mobility. It is in charge of the Traffic Operation Center (TOC), which provides a variety of mobility services at the urban and regional levels, enabled by the integration of ITS (e.g., traffic monitoring, traffic light control, limited traffic zones control, parking information and others). Regarding real-time traffic monitoring and control, the TOC incorporates traffic sensors (approximately 1,700), floating car data, 26 information display panels, 71 traffic cameras, an adaptive traffic control system (which manages over 300 urban traffic lights in the city of Turin) and a traffic SuperVision system (SV), which is devoted to real-time traffic state estimation in areas where sensors are not present.

The examination of traffic flows acquired by the 5T fixed sensor network, as well as the demand data (O/D matrices) given by the 5T SV system, yielded the results reported in this research. The most recent large-scale review of O/D matrices was carried out in 2011, utilizing data drawn from the official national Census as well as from a periodic metropolitan mobility survey. From that point on, matrices were periodically updated with historical data gathered by sensors through the use of a specialized algorithm that aims to minimize the discrepancy between estimated and measured traffic flows. The reliability of O/D matrices has been demonstrated over time, despite their non-uniform composition, and they have been utilized by local authorities as a private traffic demand data source for many years.

The content of the O/D matrices is represented as a database whose records correspond to the traffic flow measured in terms of the number of vehicles per hour (Veh/h) for each Origin/Destination combination. It is computed by the SV system using the traffic macro-simulation concept, which estimates the basic traffic states in terms of traffic flows, travel times, speed and vehicle densities, among other things. Base traffic states, which are calculated every hour of the day, are then combined with traffic data collected in real time by sensors, taking into account any events (roadworks, closures, etc.) that occur on the road network. This procedure is carried out once every 5 minutes.

It needs to be noted that distinct O/D matrices can be observed on different days of the week, specifically on weekdays, pre-public holiday days (including Saturdays), and public holiday days (including Sundays). Additionally, three main periods of the year are distinguished: school days, non-school days and "summer vacation" days (typically 2 weeks in August). For the sake of this study, we used a typical school weekday O/D matrix as input.

The origins and destinations in the matrix are denoted by Traffic Assignment Zones (TAZs), which are an aggregation of Census Cells (i.e., basic elements of the National Statistics Institute official zoning). TAZs were identified in Turin in the '90s by the Turin Metropolitan Mobility Agency and are used as a reference by local governments when designing their planning activities.

The case study under investigation is a 600-kilometer-square area in and around the Municipality of Turin, including its suburbs: Moncalieri, Nichelino, Borgaretto, Orbassano, Beinasco, Grugliasco, Collegno, Cascine Vica, Pianezza, Druento, Venaria Reale, Borgaro Torinese, Mappano, Settimo Torinese, San Mauro and Revigliasco for a total of 257 TAZs and an estimated population of more than 1,200,000 residents.

2.5 Methodology

We will now explain the methods that we used in order to develop such a comprehensive model, emphasizing the three essential phases that we took throughout the process. First of all, we discuss in detail the procedure for producing a precise road graph of the research region. In the following stage, we will go over the technique for creating all vehicle routes from the initial O/D matrix. Finally, we look at all of the additional changes and modifications that have been implemented in order to achieve the objective of a comprehensive full-day traffic simulation that is as close as possible to the traffic situation provided by municipal sensors.

2.5.1 Road graph creation and map editing

The JOSM software [23], an extensible editor for OpenStreetMap [24] written in Java 8+, was used to generate the graph representing the road network of the research region, which resulted in an OSM file. The OSM file is in the eXtensible Markup Language (XML) format, and it contains all of the information that may be geo-referenced and connected to a bi-dimensional topographical map showing highways, roads, railroads, bike lanes, and rivers. Additional objects, such as buildings, parks, rivers, and other areas of interest, can be added to the scene.

The mobility simulator that we employed in this study is SUMO (Simulator of Urban Mobility) [25], which is an open-source, highly portable, microscopic and time-continuous road traffic simulation tool that is developed to handle large road networks. In particular, we used version 1.1.0, which was created in December 2018. Due to the fact that it is a microscopic simulator, SUMO accurately models each vehicle as it moves along its own route. We used NETCONVERT [26] (one of SUMO's auxiliary tools) to convert an OSM file of the area of interest into a network file in SUMO-readable XML format, which we then imported into SUMO.

Due to the fact that this research is solely concerned with automotive traffic, the infrastructure examined in the OSM file was filtered out. The final network is composed of about 33,000 nodes, 66,000 edges, and more than 6,500 Km of roads (more than 7,700 Km considering multiple-lane roads).

This initial configuration is, however, far from precise, and the resultant network has several inaccuracies. Streets are often erroneously marked and require human post-processing, while in other cases, it is the NETCONVERT conversion procedure that is faulty. One must never forget, though, that while our goal is to create a faithful representation of the real Turin road network, it nevertheless remains a model and, as such, it has inconsistencies and approximations, just as all models do. The majority of the adjustments on the map were made either manually or through scripts. We preferably acted on the ones that were responsible for high, unrealistic traffic congestion situations.

Speed limits

A significant editing job targeted the nominal speed limits of network edges, which are assigned by SUMO. Those are notably different from the average speed of real automobiles on the equivalent streets. On the one hand, a real vehicle is frequently challenged by bad asphalt conditions or lane obstructions as a result of, for example, double-parked vehicles. On the other hand, it is not uncommon for a real car to go at speeds greater than the posted limit if traffic and road conditions permit it.

As a consequence, we set the speed limit of a network edge to the average speed that a vehicle would experience if it were the only vehicle on that street portion. These conditions are known as the Free-Flow or Flow-0 traffic conditions, and they are the input that we used to investigate how speed decreases during periods of congestion. To obtain the Flow-0 average speeds, we utilized a Google Maps API called Distance Matrix [27]. Based on the departure time and traffic conditions, this matrix calculates the realistic time in seconds that it would take a vehicle to go from an origin to a destination in a particular traffic condition (optimistic, medium or pessimistic). Being interested in a perfect circumstance with no automobiles at all, we chose the optimistic traffic condition option, which began at 4:30am and resulted in extremely minimal congestion for Turin (and for any city, for that matter).

Street width and lanes

Many roads in Turin do not have a separating line between lanes, yet they are spacious enough to accommodate two or more vehicles traveling in parallel. In many cases, these roads are inaccurately shown in SUMO as being single-lane or, more generally, as having fewer lanes than intended. As a result, we frequently identify traffic congestion in regions where traffic is actually fluid because cars are being forced into a single lane.

Due to the lack of an online database that records the correct number of lanes, the whole length of several primary roads was reviewed using Google Maps Street View, and the precise number of nominal or theoretical lanes was corrected. To be more specific, we manually revised more than 100 primary roads in total.

Following the correction outlined above, the network architecture that we obtained is depicted in Fig. 2.1.

Intersection management

The management of junctions presented further issues: in SUMO, priority for intersections is only right-before-left, although roundabouts are typically regulated by a left-before-right priority. The priority of more than 500 roundabouts was therefore manually fixed.

Another significant editing work was focused on the phases of traffic signals. NETCONVERT generates fictitious traffic light phases at crossings that have been marked as controlled by a traffic light in the OSM file. These phases are clearly oblivious to the actual traffic flows, and in the specific example of the Turin map, they frequently result in long queues at the intersection. A new problem presented itself when we attempted to correct the phases of more than 900 traffic lanterns using data provided by the 5T SV system. Phases in Turin's traffic lights change during the day according to an algorithm called UTOPIA [28], which takes into account real-time traffic patterns and adjusts the phases as needed. If we had included such a behavior in our model, the complexity of our model would have risen manyfold. Therefore, we took the average of phase durations over a single day and we inserted the mean values into more than 800 junctions with traffic signals. Their distribution over the model map is seen in Fig. 2.2. We gave more priority to traffic signals that had a greater influence on the flow of traffic than others.

It was also necessary to have some sort of synchronization in order to achieve the so-called green-wave effect on a road. SUMO provides a Python script called `tlsCoordinator.py` that manages the traffic light synchronization given the traffic pattern of a single hour. The `tlsCoordinator.py` script accomplishes its work, but it still needs some manual modifications to function properly. A compromise was needed in order to account for the unavoidable inconsistencies between our manual interventions and the actual traffic light situation in Turin. We observed that the activation of all traffic-light-regulated intersections was the primary cause of the increased congestion. Thus, through an appropriate SUMO option, we enabled traffic lights only for traffic-saturated edges, while all the intersections with low traffic conditions were treated as priority-based, i.e., without a traffic light. Thus, a mixed scenario is created in which traffic lights are triggered only in overloaded junctions, avoiding inaccurately-set traffic lights from hindering traffic flow

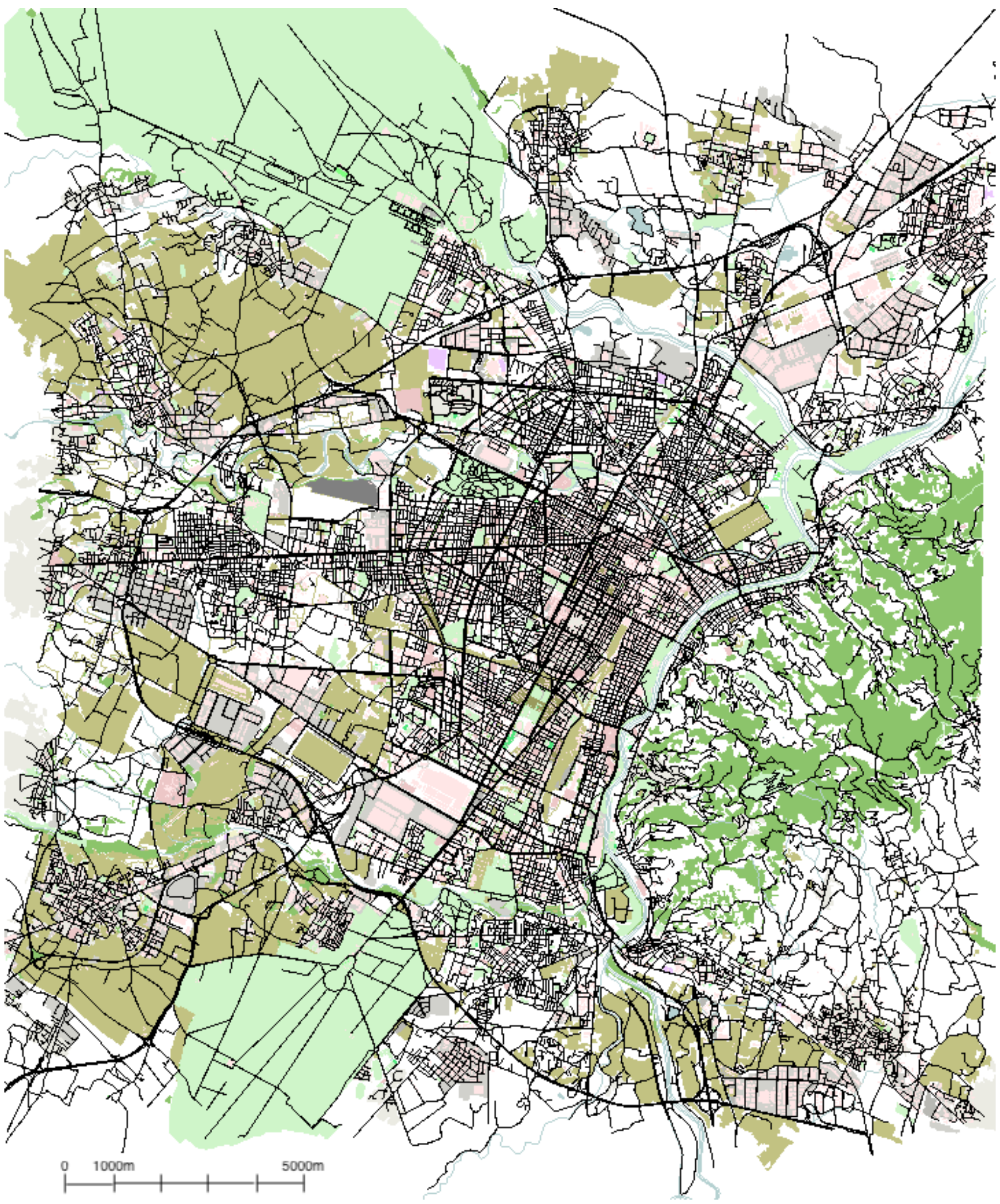


Figure 2.1: Turin SUMO Traffic case study map.

in low-traffic areas. When a low-traffic condition occurs at a traffic-light-regulated intersection, SUMO allows vehicles to proceed in accordance with the standard priority-based approach as if all traffic lights were flashing amber.

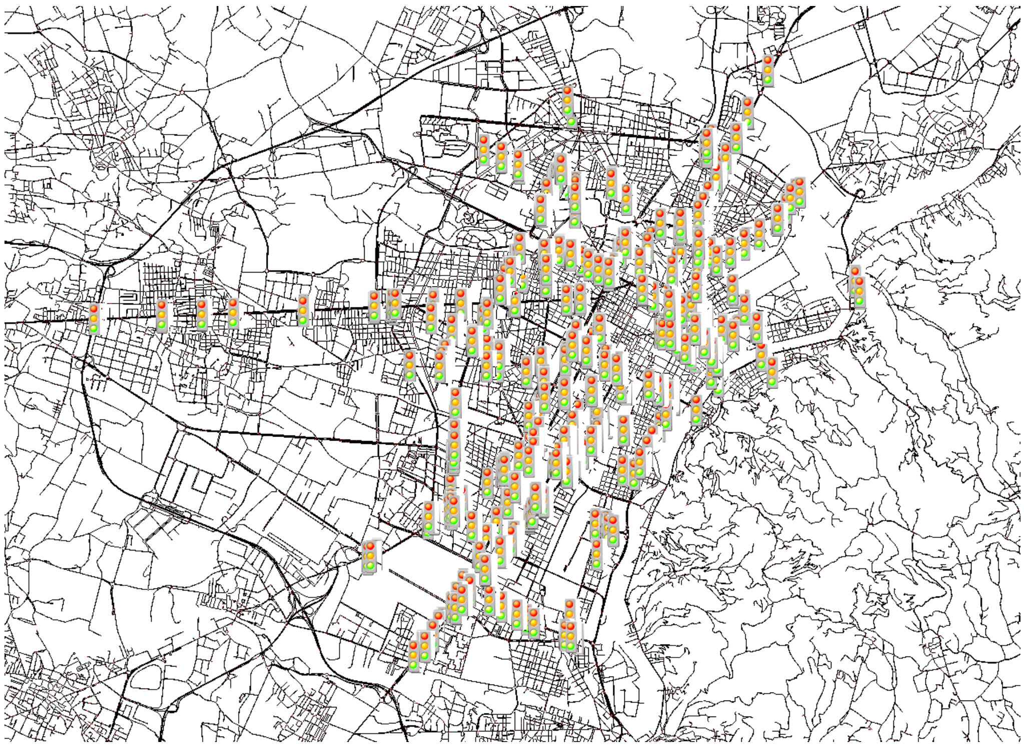


Figure 2.2: Traffic lights distribution over the map.

2.5.2 Traffic Assignment

Creating a realistic map would be useless unless it was accompanied by realistic input traffic. Therefore, we exploited the 5T O/D matrices to populate the map with traffic that was as close to the real-world as possible. Firstly, we isolated the traffic flows with Origin and Destination in a TAZ inside the map presented in the preceding section, taking only a typical school weekday into consideration as a reference. The O/D matrix was then converted into SUMO trips, with each trip being assigned an origin edge, a destination edge, and a corresponding departure time (the latter generated randomly within each hour).

The task of generating an O/D edge pair and a depart time in accordance with the flow is quite simple. Given an O/D pair and a time of departure for each vehicle, the challenge is the choice of a route that leads the system to a stable situation even in congested conditions. Consider how a human driver behaves (particularly when equipped with a navigator) when confronted with growing traffic congestion: if possible, he or she will avoid adding another car to the queue and will instead search for alternate routes to avoid the traffic jam completely.

In mobility simulation, this task is typically performed by Traffic Assignment algorithms, of which there are several instances in the literature, beginning with some early research by Wardrop [29] in 1952 and subsequently by Liu and Fricker [30] in 1996 on Stochastic User Equilibrium (SUE) with the Logit method. The SUE-Gawron assignment was specified in Gawron's dissertation [31], which was completed in 1998. Other recent research, such as [32] and [33], examine the performance of different Traffic Assignment algorithms, concluding that there is no ideal Traffic Assignment algorithm and that its performance is greatly dependent on the use case. As a result, in [34], many application cases are examined, and the Traffic Assignment problem of many cities is addressed. Unfortunately, Turin is not one of the use cases that have been examined in the mentioned project.

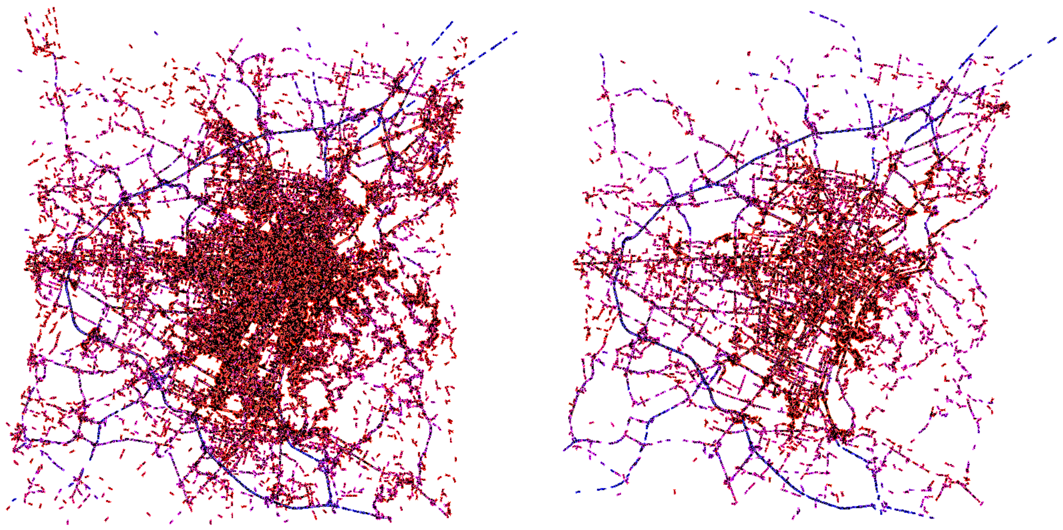
As a consequence, different Traffic Assignment algorithms were evaluated in this study in order to determine which one best suited our model. A detailed description of some fundamental Traffic Assignment algorithms, their implementation in SUMO, and their applicability to our model is provided below.

Algorithms for Traffic Assignment

Among all the Traffic Assignment algorithms, the All-or-Nothing assignment is the simplest one. It works by using a simple Dijkstra algorithm [35] to perform a shortest path search in a graph with weights. In this specific case, the weights of the graph correspond to the travel times of street edges. All-or-Nothing assignment is implemented in SUMO with the DUAROUTER tool [36], which performs a shortest path computation at the beginning of the simulation for all vehicles in the model. It is clear how this algorithm leads to an unrealistic outcome. We can consider, as an example, two parallel streets sharing the same origin and destination points but with slightly different travel times. Running a DUAROUTER method on them will result in all vehicles choosing the path with the lower nominal travel time and no one choosing the other, which is clearly a solution far from reality.

The User Equilibrium assignment method is based on Wardrop's principle [29], which states that, after a user equilibrium condition is reached, no driver can unilaterally reduce their travel time by shifting to another route. This assignment approach results in an equilibrium solution. However, it is not practicable for large-scale models: every new vehicle entering the system will result in a travel time modification for all the other drivers in the simulation, which should recompute the travel times over all the paths present on the map. Moreover, User Equilibrium assignment does not take into account the possible different time perceptions that drivers may have over the same path. For this reason, it is not a realistic solution, even if it leads to an optimum equilibrium scenario.

The Stochastic User Equilibrium (SUE) assignment extends the User Equilibrium assignment by introducing randomization in the route-choice procedure for the selection among shortest paths. In this way, it is considered the fact that drivers' perceptions of trip time vary from one another. Exploiting such a schema, two different route-choice procedures were developed: the Gawron method [31] and the Logit method [30]. The Gawron method computes the probability of choosing



(a) (b)

Figure 2.3: Comparison between Traffic Assignment methods.

a route based on three metrics: i) the travel time along the used route in the previous iteration step, ii) the sum of the edge travel times over the alternate paths and iii) the previous probability of choosing that route. The Logit method, on the other hand, applies a fixed formula to each route to calculate the new selection probability, ignoring the old costs and probabilities of the previous iteration step. The new selection criteria are calculated from an exponential function scaled by the sum of all the current route values. As for the User Equilibrium algorithm, the SUE assignment leads to an equilibrium solution. Nonetheless, it suffers from the same scalability issue as the simple User Equilibrium algorithm and as a result, it appears more appropriate for low-congestion traffic conditions, such as off-peak periods or lightly traveled rural areas.

The Incremental Traffic Assignment (ITA) method is a greedy algorithm which does not lead to an optimum equilibrium solution. ITA assigns fractions of traffic volumes in steps, in a similar way to the All-or-Nothing method. The notable difference is that travel times are computed at the vehicle's departure time and not at the beginning of the simulation for all the drivers in the model. In this way, a vehicle will compute its optimal route based on the current traffic conditions, which is actually a very good approximation of what happens in real life.

It must be noticed that no Traffic Assignment algorithm is optimum over a single run. In order to achieve a good result, every algorithm has to be used in an iterative way. As an example, the difference between a single-run method and an iterative one is shown in Fig. 2.3, where red vehicles are slow or stopped, while blue ones are moving fast. Fig. 2.3a depicts an All-or-Nothing Assignment on a single iteration, whereas Fig. 2.3b shows an Incremental Assignment at the 50th iteration.

It is possible to do an iterative assignment using one of two methods: the Microscopic Traffic Assignment method or the Macroscopic Traffic Assignment method. In the former, a simulation is run for each iteration and travel times on network edges are measured. Travel times determined in this manner are then provided as input to the chosen Traffic Assignment method for the subsequent iteration phase, and the cycle continues until the maximum number of iterations is achieved. The downside of this strategy is that it is prohibitively time-consuming: thousands of simulations must be done in order to arrive at the optimal Traffic Assignment.

The Macroscopic Traffic Assignment attempts to replicate the Microscopic Traffic Assignment but in a more rapid manner. Indeed, instead of executing a simulation for each step, it employs mathematical resistive functions that simulate the travel time increase as traffic flows surge. When used in conjunction with the SUMO traffic simulator, this strategy is implemented by MAROUTER [37], a tool that computes trip times and flows from traffic density by using a hard-coded capacity-constraint function based on speed limits, lane numbers, and edge priorities. The Macroscopic Traffic Assignment method is, of course, less accurate than the Microscopic Traffic Assignment method: because of approximations of mathematical functions, trip durations are estimated rather than measured. MAROUTER supports the SUE algorithm with both Gawron and Logit methods and the ITA algorithm. It is also possible to tune the assignment by selecting the number of alternate paths to consider and the number of iterations to perform in nested loops cycles before stopping the equilibrium search.

In the next subsection, we compare the performances over different parameter usages and across different algorithms.

Comparison of Traffic Assignment algorithms

The results reported in the following paragraphs were averaged over five simulation runs and show the outcomes of the so-called *stability study*. A stability study consists of introducing a certain traffic pattern into our model with the aim of analyzing the time needed to let all drivers complete their journey. The main metric to inspect is the time needed to empty the network.

A first instance of a stability study is the one reported in Fig. 2.4, where a stability study was performed over a single iteration of an All-or-Nothing assignment algorithm. In the figure, three different traffic patterns are represented: one hour of the morning peak in green, one hour of the average daily hour in yellow, and one hour of the evening traffic pattern in blue. Because of the differences in traffic loads throughout those periods of the day, we can observe that the maximum reached by the three lines is different as well. It is important to note how the blue line begins to flatten out before reaching its maximum: this behavior suggests that the number of cars exiting the network is becoming equal to the number of vehicles entering it, and that a condition of stability is on the horizon. The other two lines,

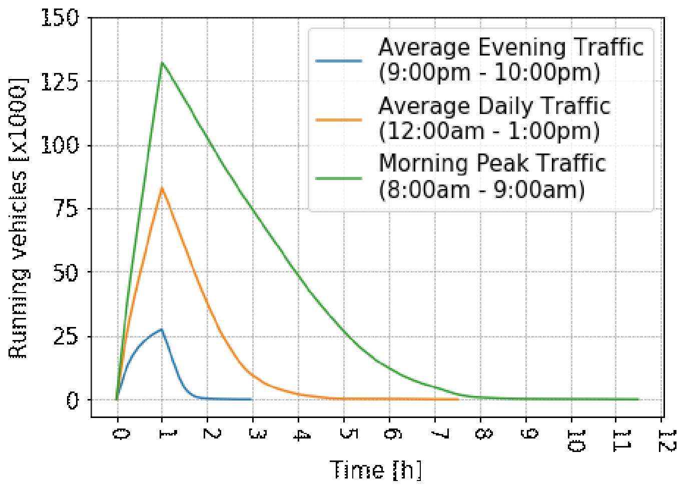
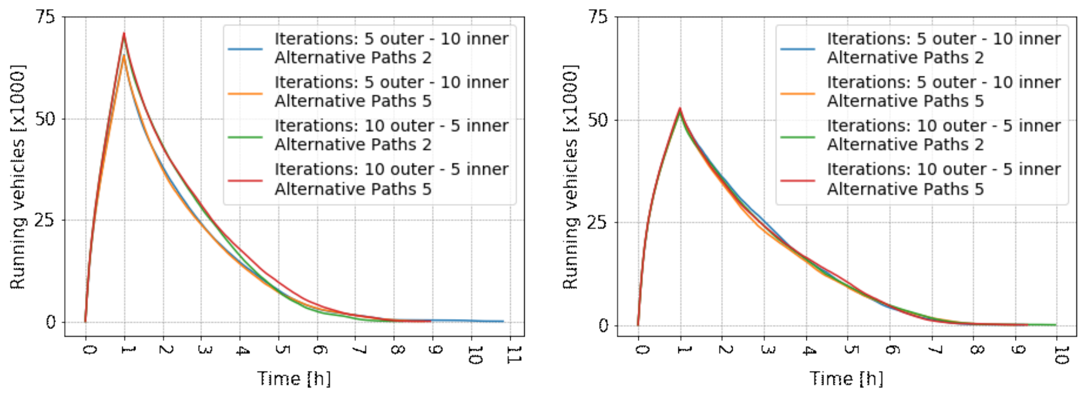


Figure 2.4: Stability study of All-or-Nothing algorithm.

on the other hand, do not achieve this level of stability. The graph shows that more than 7 hours were required to service all drivers during an average daily hour of simulation, while more than 11 hours were needed to empty the network during a morning peak hour. These underwhelming results are due to the method of Traffic Assignment that was chosen; we will see later on how they will be improved by using different Traffic Assignment algorithms and by doing multiple iteration runs. In order to get the best results, we will always focus on the morning peak hour (i.e., the green line in the previous plot), from 8:00am to 9:00am, because it covers the maximum number of vehicles injected into a daily traffic pattern.

Fig. 2.5 reports the stability study applied to the SUE Traffic Assignment algorithm. In particular, in Fig. 2.5a the study was conducted over the Gawron method, while Fig. 2.5b depicts the results of the experiment over the Logit method. For both the SUE methods inspected, we used different parameter choices for the number of iterations of internal nested loop cycles and the maximum number of alternative paths to select. First and foremost, it should be noted that there is essentially no difference between all of the options. However, there is a huge improvement between the two mathematical methods. Indeed, while we had approximately 75,000 cars on the map at the end of the injection for Gawron, we only had 50,000 vehicles for the identical traffic pattern using Logit. As a result, the amount of time required to empty the map was significantly reduced by the Logit method. Furthermore, it is feasible to see how the Logit curve bends more than the Gawron curve before reaching its maximum, suggesting that the situation is more stable compared to the Gawron curve. However, such results are still unsatisfactory because it took more than 9 hours to serve all the drivers within a single peak hour. It is important to underline that both SUE-Gawron and SUE-Logit are good Traffic Assignment algorithms. The performance of such algorithms, on the other hand, is



(a) (b)

Figure 2.5: Stability study of SUE algorithm with (a) Gawron and (b) Logit methods.

highly dependent on the model to which they are applied.

Eventually, the output of the stability study using the ITA algorithm is reported in Fig. 2.6 for different numbers of iteration runs. We can see how the maximum number of vehicles obtained after the injection is slightly lower than the SUE Logit scenario (less than 50,000 vehicles), while more remarkably, the number of hours required to empty the network has been reduced to 3 and a half. Furthermore, it can be appreciated how the difference between the curves decreases when the number of iterations increases and also how the curves bend before their maximum. Those signals indicate that a stability condition has been reached and it is notable that 50 iterations are sufficient to obtain it.

From the analysis of the results here reported, we selected ITA as the Traffic Assignment algorithm most suited to our model. Drivers in Turin's actual traffic during the morning rush hour may expect to encounter extremely high levels of congestion, particularly on the outer ring road, which can easily result in hour-long trips across town. Despite that, the stability study highlighted a situation far from reality, as it is possible to observe from the amount of time needed to solve the peak hour of traffic. Our simulator is still not capable of handling the nominal full-day traffic pattern at this stage of development. As it is possible to read from previous plots, only a small percentage of the full-day traffic can be successfully introduced at this point. Therefore, in order to increase the traffic demand, some enhancements have been implemented and are reported in the following subsection.

2.5.3 Improvements and Assumptions

First, we observed that, during the traffic assignment phase, the route creation process tended to prioritize the use of large highways with numerous lanes while,

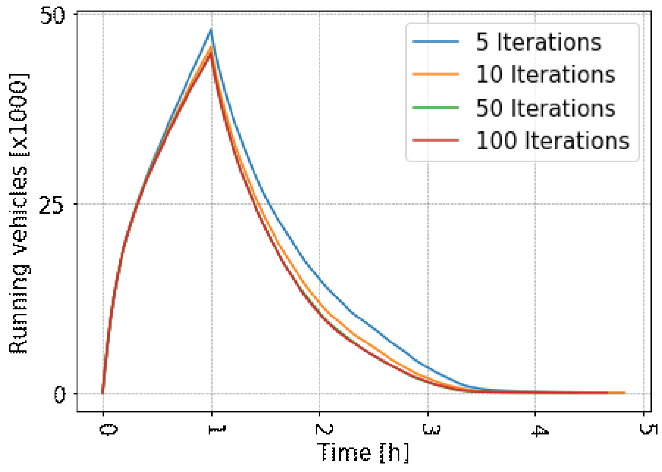


Figure 2.6: Stability study of ITA algorithm.

on the other hand, assigning few or even no vehicles to minor roads and small residential streets. This behavior may be considered quite realistic given that most drivers prefer to travel on fast, multiple-lane roads rather than small, one-lane streets in residential neighborhoods. However, this is not always the case at the start and end of a car trip. Indeed, it is reasonable for drivers to start and/or end their journey near a parking lot or close to their home, which are typically located on residential streets. Turin's main avenues are often paired with service roads (known as "controviali" in Italian), which is a characteristic of the city. They are single-lane streets running alongside both directions of an avenue, and are mainly used for parking or by local businesses to avoid hindering faster traffic on the main avenue. Even though service roads have a moderate average speed and are not considered main roads, it is reasonable to assume that a vehicle trip will begin and terminate there.

Thus, we developed a Python script to extend the first and last segments of each vehicle trip. To achieve this goal, origins and destinations inside TAZs are no longer chosen randomly among all the road edges of the zone, but only among residential streets and service roads inside the area. To connect the new trip origin with the previous one, a shortest path technique using DUAROUTER is performed, followed by a similar approach for the destination. Finally, the new route that has been generated in this manner is substituted to the old route in the route file.

Because such an operation takes a significant amount of time, it is not feasible to complete this procedure for each of the 2,200,000 vehicle trips. Consequently, only a subset of all trips was revised in this manner in order to create a moderate distribution of cars over the whole map. We also noted an increase in the amount of traffic our model is capable of handling as a result of this modification, since the route assignment now distributes more traffic among several secondary and tertiary

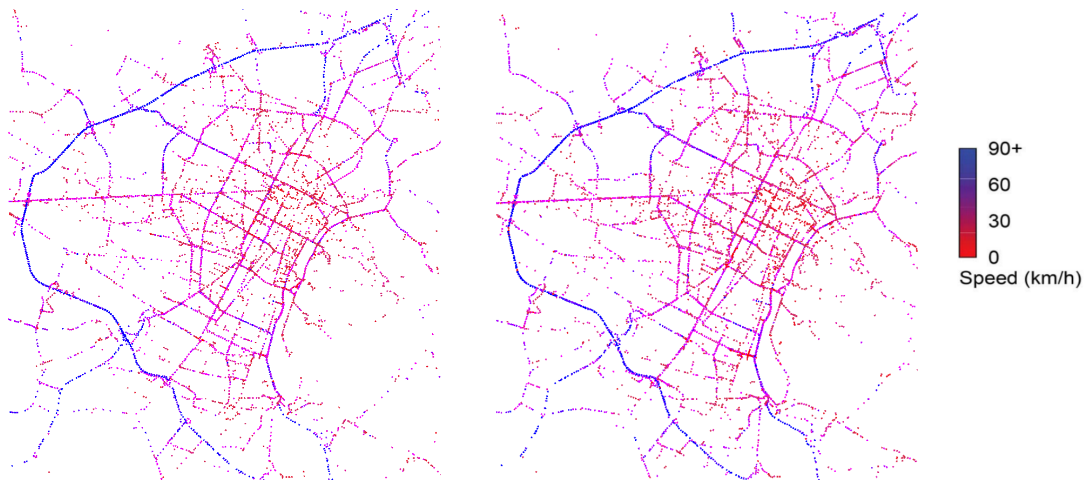


Figure 2.7: Snapshots from the final TuST data set. Morning peak of 8:00am (left) and afternoon peak of 5:00pm (right).

routes that were previously unused. We will get back to this topic when discussing results in the next section.

The usage of the SUMO option for mesoscopic simulations, known as MESO [38], was the adjustment that had the biggest influence on the outcome. Applying the MESO option, cars on every edge are grouped into traffic queues, with vehicles at the top of each queue being allowed to "jump" into the next queue on an adjacent edge. The system treats each queue in a distinct manner, allowing for considerably simpler control of the overall simulation. Indeed, computing vehicle movements with queues makes simulations run up to 100 times faster than the microscopic model of SUMO without the MESO option. Moreover, due to the jump movements of vehicles from one queue to another, it is more tolerant of network modeling errors than SUMO without the MESO option. Consequently, the proportion of traffic demand that our model is capable of accepting as input significantly improves.

Despite all of the advantages outlined above, the MESO option has some drawbacks as well. Its queue-based management model is excellent for controlling the edges of very large-scale systems, but it loses all the microscopic mobility at intersections. Furthermore, since the mesoscopic model is edge-based, it is not possible to produce a lane-based output. All of the information related to individual lanes of a street are aggregated into an edge-based output. Anyway, as detailed later, output traces can still be considered microscopic and can be used as an input for a microscopic model, too.

We are finally able to execute a full-day traffic pattern simulation after launching the SUMO command. As a result, the road traffic at the morning and afternoon peaks, in Fig. 2.7, looks like a very reasonable picture of the actual traffic pattern of Turin (as will be quantitatively assessed later). In the figure, the orbital motorway around the city is outlined in bright blue, corresponding to speeds higher than 90 Km/h. Large sections of urban roads are coloured violet, suggesting that traffic is moving fast. The presence of traffic congestion is indicated by the presence of red regions: dark red regions indicate vehicles moving at slower speeds, between 30 and 50 Km/h; bright red areas indicate traffic situations in which vehicles are stuck in queues.

Following the methods mentioned above, we were able to produce a huge traffic model that covers a large region in and around the city of Turin. To better understand the magnitude of our model, some characteristics are reported in Table 2.1.

2.6 Results

This section contains examples of results obtained using the TuST simulator. The findings we show are intended to demonstrate the validity of the model; a mobility study of the examined region is not the goal of this research. Since TuST can

Table 2.1: TuST Scenario in Numbers

Area	602.61 Km^2
Traffic Assignment Zones	257
Total nodes	32,936
Total edges	66,296
Total junctions	22,209
Traffic lights	856
Roundabouts	501
Total length edges	6,570.28 Km
Total length lanes	7,723.40 Km
Vehicle length	4.3 m
Total vehicle trips	2,202,814
Ended vehicle trips	2,197,672
Final waiting vehicles	0
Final running vehicles	5142
Teleports	334
Collisions	0
Halting	19

be used to produce a variety of measures, we chose to group the results depending on the map element they are focused on. The global output covers the entire map and the simulation as a whole, providing a useful indication to assess whether the simulation is meaningful or not. After that, we shift our attention to traffic on the single-street sections. Real-time traffic data from street sensors, as well as these types of measures, are extremely important for the validation process. Finally, the inspection of vehicle routes is discussed. They constitute the vehicle traces we also provide in our GitHub repository [1], and they can be used as a starting point for any kind of mobility analysis or testing of vehicular applications.

2.6.1 Global Output

TuST generates a number of key files that may be used to examine any measure derived from simulations. The *Summary* file contains, for each time step, all the aggregated data on the vehicles in the simulation, such as the total number of vehicles running, the total number of vehicles that have completed their route, and the total number of cars that are waiting to enter the simulation. The waiting vehicles are those that are scheduled to enter at a given time, but, due to congestion in the origin area, do not find room in the map. Those vehicles will have to wait for the congestion in that region to decrease before they can be introduced, which will be later than originally intended. Together with the number of active cars in the model, for each time step, it constitutes the main index of congestion in the model.

From the *Summary* output, we can extract global information about the whole simulated environment. In particular, if the number of running cars, as well as the number of waiting cars, increases exponentially, this is a strong indicator of an unstable simulation, which suggests that the traffic demand provided in the input has not been properly sized. Stability analysis over the full-day scenario is thus necessary in a similar way to the one done in the preceding section for choosing the proper Traffic Assignment algorithm.

The stability study plot representing a stable simulation is shown in Fig. 2.8, where the injected vehicles per hour, according to the O/D matrix, are shown in red; the blue line depicts the number of running vehicles and the orange line represents the waiting ones. In the plot, a full-day traffic pattern is presented, showing that a complete simulation, with all trips according to the O/D matrix, can be successfully achieved. It is possible to observe how the blue line of running vehicles resembles the red one of input traffic demand throughout the day. The waiting vehicles peak around 5pm, highlighting the evening rush hour. Nonetheless, the traffic congestion is alleviated within an hour of its occurrence. With no time constraint and a full-day simulation, the remaining automobiles complete their travels in less than 28 minutes and 57 seconds, demonstrating that there is no persistent congestion affecting the simulation.

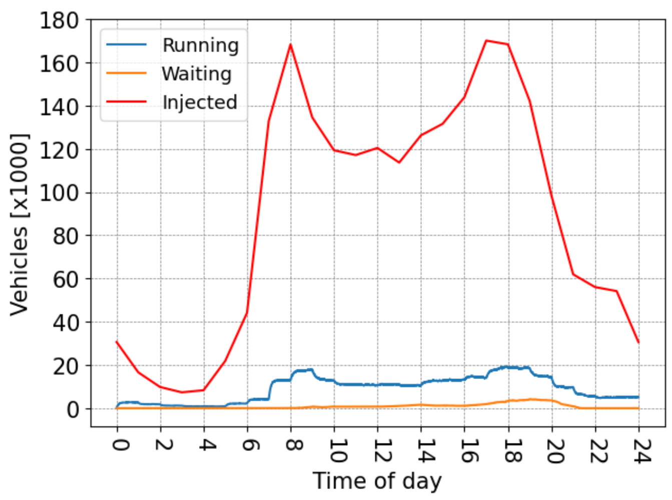


Figure 2.8: Stability study result on full-day traffic demand.

2.6.2 Road-Level Traffic Inspection

If we want to examine an edge-based output, such as the traffic on a given route, we must first extract different data from the simulation. In order to do this, a detector object has been created on every edge of our map. Detectors are in charge of counting the number of vehicles passing on a road, thus creating a *Detectors Output* file for edge-based output with a given sample frequency.

As illustrated in Fig. 2.9, we can examine the distribution of vehicles on the map using the *Detectors Output* file. Edges highlighted in red are the streets traveled on by more than two vehicles per minute; in yellow, we can observe edges with more than one vehicle per minute, while less than one vehicle per minute travels on green streets. The edges colored in black are unused streets, with no traffic throughout the simulation. According to what we can see, the red roads are the orbital motorways or principal roads, which were constructed to handle a significant percentage of traffic demand. On the contrary, the green and black margins represent residential and neighborhood streets, which are rarely used by automobiles, even in densely populated metropolitan areas. The actual position of street sensors utilized for model validation is represented by blue icons, which are detailed in further detail below.

As previously mentioned, we extended the very first and the very last parts of car trips in order to make our model more realistic and to obtain a better vehicle scatter on the map. As illustrated by the tail of the distribution in Fig. 2.10, this adjustment allows up to 94% of the map's edges to be used. Using the path-extension methodology on all edges, we could expect to further increase the percentage of edges travelled by vehicles. However, the technique would be prohibitively time-consuming in comparison to the gains it would provide, which would be statistically

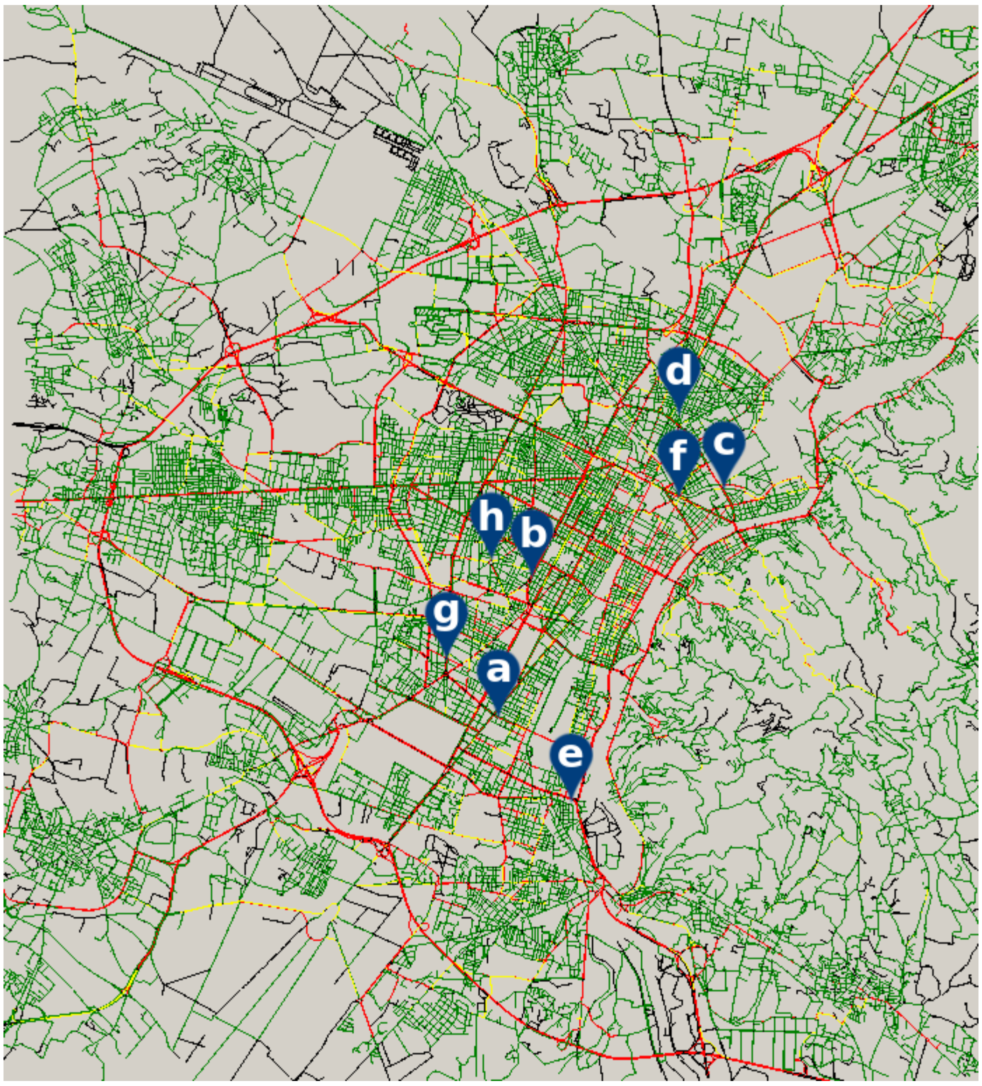


Figure 2.9: Vehicles distribution over the map.

insignificant and have little influence on performance.

To validate the edge-based output, data from 5T street sensors on many primary roads were used. A street sensor counts the rate of vehicles transiting on the portion of road under which it is placed. 5T installed hundreds of sensors under Turin's road surfaces. However, many of them have been inactive or have been damaged as a consequence of roadworks throughout the years. More than 300 active sensors are still in use today, and they are installed under the surface of several roads. By setting the sample frequency of our detectors to 5 minutes, which corresponded to the frequency of 5T sensors, we were able to compare data from simulations with real-world traffic data, therefore validating our model.

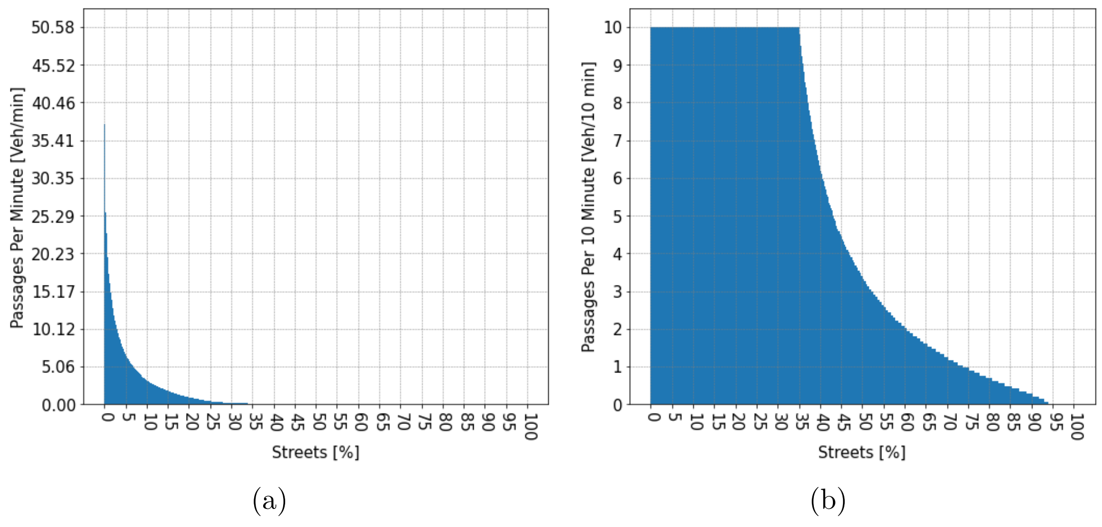
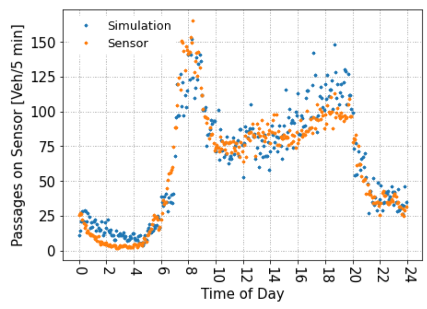


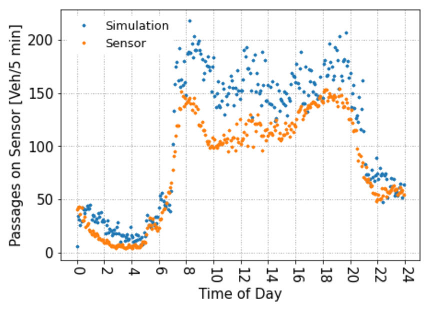
Figure 2.10: Distribution of vehicle densities on map edges: (a) total distribution; (b) distribution zoomed over the tail

In Fig. 2.11, we can see some examples of sensor data being compared to data obtained from simulations. Fig. 2.9 shows the actual locations of the streets for which the comparisons are being conducted, which are indicated by blue icons. In particular, we have: (a) Corso Eusebio Giambone, (b) Corso Mediterraneo, (c) Corso Novara, (d) Corso Palermo, (e) Corso Piero Maroncelli, (f) Corso Regina Margherita, (g) Corso Siracusa and (h) Via Monginevro. As we can see, there is very little difference between the sensed data, which is represented by orange dots, and the measured data, which is represented by blue dots. Interestingly, there is no bias in the comparison, with sensed data either overestimated or underestimated by some percentage, depending on the street.

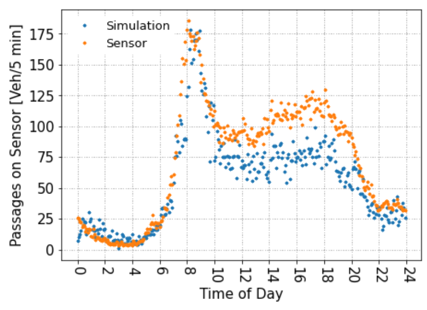
Finally, a quantitative validation was performed. Indeed, it is reasonable to assume that simulation data depicts a more accurate image of primary roads than of service and residential streets. For this reason, an affinity index was measured for sensors placed under the surface of main streets only. Calculating the affinity index was done by comparing the difference between simulated and sensed data. Fig. 2.12 shows how 80% of the considered sensors have more than 75% affinity with simulation data, and that only 5% of sensors have less than 50% affinity. Overall, this comparison demonstrates that the simulator accurately simulates the actual traffic flow in Turin on all primary roads represented on our map, with extremely close approximations.



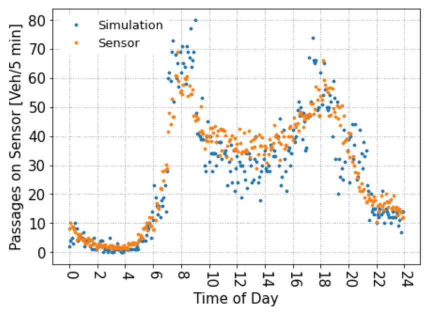
(a)



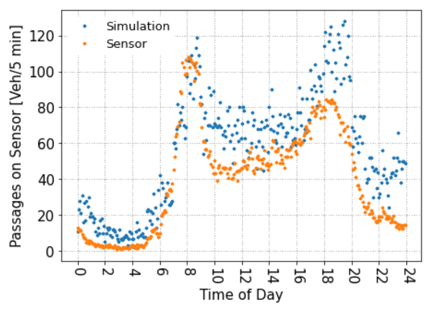
(b)



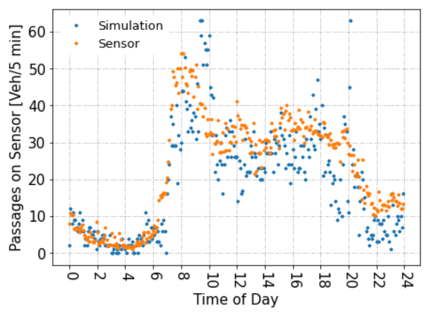
(c)



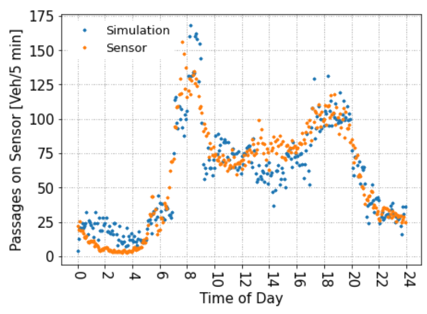
(d)



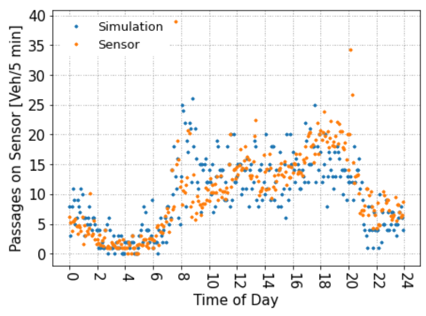
(e)



(f)



(g)



(h)

Figure 2.11: Validation of simulation measurements (blue dots) with data from 5T street sensors (orange dots). Refer to icons in Fig. 2.9 for the exact location.

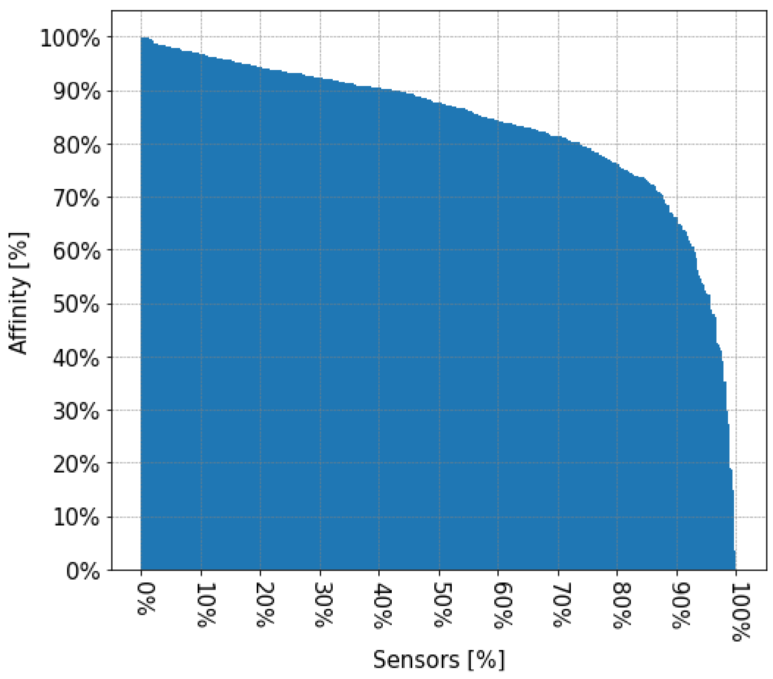


Figure 2.12: Quantitative validation analysis on main street sensors.

2.6.3 Vehicle Route Inspection

The third type of output we can extract from simulations is the *Vehicle Route Output* file. It constitutes a very important type of output, containing all vehicle traces. For each trace, alongside the complete route flow, some important characteristics of the vehicle path are measured, i.e., the exit times of every edge of the route, the total length of the complete path and the time at which the considered vehicle reaches its final destination.

We can thus compute some very significant metrics regarding vehicle paths. We can see an example in Fig. 2.13, where the average route length is shown in blue and the average travel time of a vehicle is plotted in red. The peaks of the morning and afternoon are clearly visible in both graphs. At such periods of the day, we may expect to see an increase in the number of car trips on average due to commuters' travels, which can be considered longer than other trips. In a similar manner, we can observe longer average travel times, which is a result of both the increased duration of journeys and the increased congestion. The night time, on the other hand, is highly unpredictable. Trips in this period are night-life trips, resulting in various average lengths of time. However, regardless of route length, drivers encounter very low traffic conditions at this time, allowing them to enjoy faster journeys. Overall, we observed that the majority of car trips last less than 10 minutes and in particular, 95% of them are less than 15 minutes and 36 seconds.

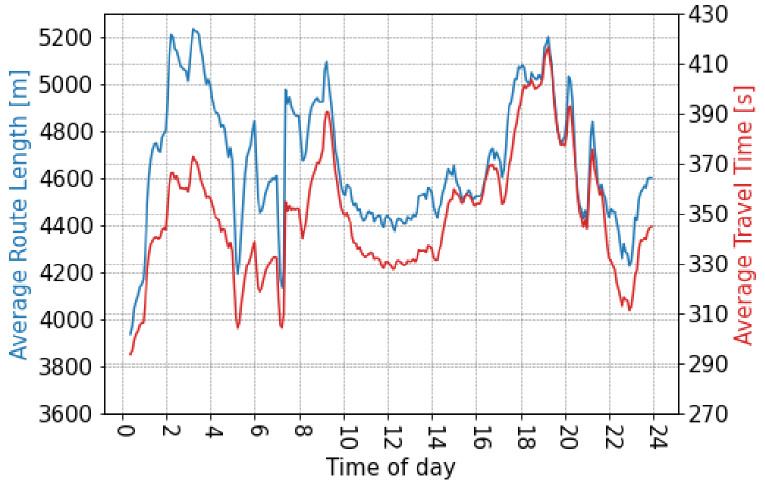


Figure 2.13: Mobility analysis: average route length versus average travel time.

Besides this simple route analysis, vehicle-based outputs can yield a variety of other relevant outcomes, such as the identification of traffic congestion hotspots or a detailed examination of polluted regions. However, vehicle routes constitute a very important output by themselves, since they can be used as realistic car traces for other simulations.

In such a spirit, full vehicle traces and other simulation output have been made available in an open-source fashion for everyone. They can be found on the GitHub main page dedicated to TuST [1], which is also linked by the SUMO wiki page related to large-scale scenarios [22].

As stated before, using SUMO with the MESO option enabled for mesoscopic modeling, we lost all the microscopic mobility in our system and we can consider the vehicle routes as mesoscopic. Despite this, the *Vehicle Route Output* file contains all of the route edges that may be used as input to a microscopic model, too. In order to prove that, we compared a microscopic simulation with 5% of injected traffic (i.e., the greatest traffic demand we are capable of giving as input to a microscopic simulation in order to obtain a stable result) with its correspondent macroscopic one. Fig. 2.14 illustrates the tiny difference between the running vehicles of a mesoscopic simulation, which are represented in blue in the figure, and the running vehicles of a microscopic simulation, which are in green. Like in previous plots, injected traffic is plotted in red and waiting vehicles are orange. It was decided on a measure of "discrepancy" in order to compare the models. We compared the transit rate of vehicles on the edges in each simulation and we measured an average discrepancy of 9.18% between the two models.

We can therefore claim that traces carrying greater traffic can also be fed to microscopic models. Even though utilizing the whole data set on a 600-Km^2 map will almost certainly result in an unstable outcome in a microscopic model, a more

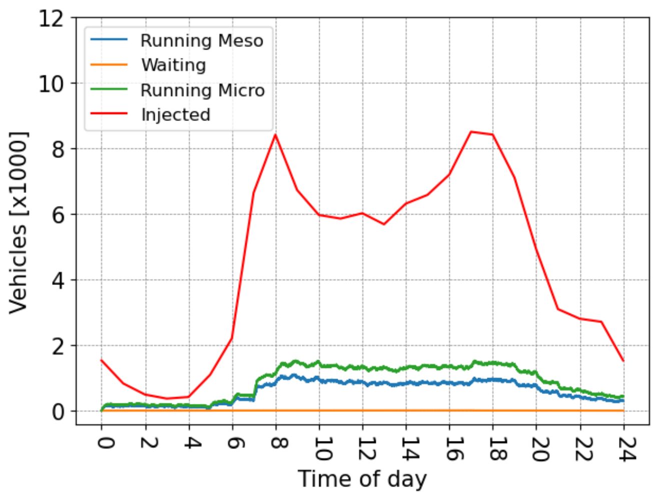


Figure 2.14: Comparison between mesoscopic and non mesoscopic simulations.

relaxed map may be constructed by using only a few TAZs instead. From the complete *Vehicle Route Output* file, only flows that travel on the considered TAZs can be extracted and a corresponding O/D matrix can be computed, creating a realistic portion of Turin's traffic with origin, destination and transient flows for the considered areas. Building a tiny simulator in this manner might result in a district-scale scenario for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication scenarios with real traffic traces, providing findings with remarkable detail.

2.7 Conclusions and future work

Continuously increasing traffic congestion in urban areas results in the waste of time and money for thousands of drivers stuck in queues and the worsening of already critical pollution conditions, calling for hard future urban planning choices. Research is increasingly concentrating on large-scale simulation tools to execute urban analysis on mobility and emissions, as well as evaluation of ITS applications for vehicular technologies and Smart Cities. However, the literature concerning models that can investigate wide-ranging scenarios, up to entire cities, is surprisingly scarce.

In this study, we presented a contribution in the form of a large-scale urban mobility simulation model and data set, describing how to construct a complex model, demonstrating which types of data are required for its realization, and highlighting all of the critical aspects of the system's construction. We focused on a 600-Km^2 area including the municipality of Turin and its surrounding districts, starting from O/D matrices extracted from the local ITS authority data and TAZs defined by the

Turin Metropolitan Mobility Agency and used for every transportation/economical/statistical analysis.

We compared mesoscopic and microscopic simulations methods, showing that both methods are valid. The decision of what should be used is determined by whether the models are to be used for large-scale mobility analysis or for the investigation of micro-mobility issues. Results demonstrate how such a large model is capable to absorb a full-day traffic demand in input at a cost of minor simplifications.

Validations using traffic sensors of the local Info-mobility Agency were successfully performed, resulting in acceptably little discrepancy levels on many sensors of the primary roads we inspected.

With future work, the model could be further improved. An automated deep learning approach for image processing might be used to correct the remaining map mistakes on secondary and residential streets. This would lead to a model with higher realism on not-crowded roads as well. The potential applications of our approach might lead to the completion of crucial emissions studies. One can also test large-scale data dissemination policies or evaluate how V2V and V2I applications perform in a scenario as wide as an entire city, using real traffic flows.

Chapter 3

Designing a Virtual Traffic Light Application

An urban-scale traffic simulator, as the one described in the the previous chapter, is instrumental for performing several mobility studies. In particular, the design of such a model can be paramount for testing vehicular application with real traffic traces of urban scenarios. In this chapter we present a Vehicle-to-Vehicle Virtual Traffic Light, namely V^{3}TL, as an example of application that could benefit from a real traffic trace. The aim of V^{3}TL is to reduce cars pollutant by targeting congestion at unregulated intersections. We also provide a complete detail of the scheduling procedure and the description of our scheduler algorithm. Integration of the model with real traffic traces from TuST are planned as future works.

3.1 Research motivation

Air pollution is a serious environmental health risk. In 2016, the World Health Organization (WHO) estimated about 7 million deaths caused by air pollution, largely as a result of heart or pulmonary diseases [39]. Among different kinds of pollutants, particulate matters (PM10 and PM2.5) are classified as Group 1 carcinogens by IARC (International Agency for Research on Cancer) and WHO itself.

Particular matters are generated by fossil fuel combustion and the greatest amounts of PM density are measured in urban environments. The main causes of those, as well as other pollutants, were found to be the emissions of industrial plants, aged heating systems and vehicular emissions. Regarding vehicular emissions, the manufacturing and distribution of hybrid and electric cars is going to mitigate the problem. However, the high cost of electric vehicles on the market prevents this from being a short-term solution. For this reason, alternative answers have to be considered.

It is a well-known fact that vehicle emissions depend on the way the driver uses the car itself. Driving at variable speeds with abrupt accelerations and frequent stop-and-go maneuvers is known to increase pollutant emissions [40], [41]. It is common to find this kind of behavior in high traffic congestion areas, where long queues delay drivers and increase their level of impatience. Apart from car accidents, roadworks or other unpredictable events, the greatest part of daily congestion is due to regulated and unregulated intersections. This is why urban planning is becoming paramount for a good mobility environment. In particular, Intelligent Transport Systems (ITS) have recently helped significantly by suggesting vehicular applications to improve road safety as well as mobility efficiency and eco-friendliness. An example of an infrastructure-based ITS solution is GLOSA, which has been shown to reduce both CO_2 emissions and fuel consumption by regulating the flow of vehicles at intersections [42].

3.2 Main contributions

In this study, we present the V2V-communication-based Virtual Traffic Light (V^{3}TL) system, a dynamic, distributed solution for infrastructure-less management of unregulated intersections. Other contribution for this work can be found in our conference paper [6] or in our journal article [5] which is currently under review. The proposed system is technology-agnostic, meaning that it can be implemented both using legacy WAVE/ITS-G5 protocols as well as the novel C-V2X technology proposed by 3GPP.
This work presents several improvements with respect to the existing literature, namely:

- we introduce a simplified, yet effective, intersection representation system encoding in few bytes all legal movements of vehicles;
- we introduce a distributed protocol, which allows selected vehicles to have a full view of the intersection before running the scheduling algorithm;
- we describe a flexible leader election protocol, introducing the possibility of resigning from leadership and joining an incomplete group of vehicles;
- we describe a heuristic scheduler based on decision trees, whose aim is maximize the number of cars crossing the intersection in a given time and minimize the number of stop-and-gos per vehicle;
- we present a complete complexity analysis of our heuristic scheduler and a comparison with an exhaustive-search algorithm;
- we reproduce and evaluate our system in different intersection environments, i.e., isolated or consecutive three- and four-way junctions, in single- and multi-lanes configuration;

- we take into consideration realistic mobility scenarios, like the balanced and unbalanced traffic scenarios.

Section 3.3 of this chapter is dedicated to related works, while in Section 3.4 technologies and use cases considered are described. Section 3.5 describes the four-steps procedure followed, the scheduling algorithm and the algorithm complexity. Section 3.6 reports the parameters setup used in our simulations, while Section 3.7 is dedicated to results. Conclusions and future works are presented in Section 3.8.

3.3 Related Work

In the last decades, several projects aimed to reduce the waiting time experienced by drivers at intersections and, consequently, to reduce emissions due to stop-and-go maneuvers. Back in 2009, the Travolution project [43] developed the Green Light Optimal Speed Advisory (GLOSA) system. GLOSA has the aim of reducing vehicle fuel consumption by predicting the next green traffic light phase and informing drivers whether they can pass in the current green phase or not. Thanks to GLOSA a vehicle can reduce its emissions by up to 22% in a single-car scenario or up to 8% in a congested scenario [44]. In 2012, MIZAR Automazione S.p.a. developed an urban traffic control system architecture called UTOPIA [28]. UTOPIA aims at synchronizing phases of different traffic lights on the same path according to live traffic patterns. Thanks to the UTOPIA phase-adaptation algorithm, travel times can be reduced by more than 15%.
Both systems cited above are currently used worldwide in many cities as working solutions for reducing traffic in regulated intersections. However, addressing the traffic in infrastructure-less intersections still remains an open problem. The definition of a Local Dynamic Map (LDM) [45] by the European Telecommunication Standard Institute (ETSI) and consensus finding studies [46] for decentralized information exchange between vehicles have shown a significant improvement in this research field. Leveraging those concepts, many studies in the literature have proposed car-to-car communication systems to schedule traffic at unregulated junctions. Ferreira et al. [47], [48] was the first to introduce the idea of a Virtual Traffic Light (VTL) system. VTL exploits beacon communication between cars and leader-based message exchange in a completely infrastructure-less scenario. This system was designed to inform all the vehicles approaching the crossroads, of the current VTL phase, although no optimization of the phases themselves was attempted.
From this idea, Hagenauer, Sommer et al. [49], [50] introduced a novel leader election algorithm and Bazzi et al. [51] performed a field test using vehicles equipped with low-cost IEEE 802.11p devices. Neither, however, introduced a scheduling algorithm to optimize the number of cars crossing the intersection in each phase. In more recent studies, Casimiro et al. [52] proposed a priority-based scheduling algorithm for autonomous vehicles only.

3.4 Intersection scenarios

The purpose of this work is to develop a system that acts as a virtual traffic light in unregulated intersections. The goal is to reduce the waiting times of drivers attempting to cross an intersection and to reduce the environmental impact by minimizing the number of stop-and-go maneuvers.

3.4.1 Communication settings

In the scenarios we are considering, depicting different intersections, vehicles are supposed to be able to communicate with each other at a close range, via V2V direct communication in a completely distributed environment. In the following, we always assume a 100% technology penetration rate and we discuss lower penetration rates in Section 3.8.

The messages exchanged are of BSM (Basic Safety Message) type, as defined by the SAE J2735 standard [53]. They are broadcast by every vehicle at a frequency of 10 Hz (i.e., the standard frequency mandated by IEEE).
The main fields that were used to populate the BSM for our goals are:

- Transmission timestamp
- Vehicle anonymous random identification number
- Vehicle motion information, such as: position, speed, acceleration, heading, yaw angle, signaling lights (turns and breaks) and the identification numbers of the current road, lane and next junction
- Vehicle relative position with respect to the other vehicles grouped in the same direction
- Leader Dataset (explained below)
- Junction Dataset (explained below)
- Solution Dataset (explained below)
- Intersection Flag (to notify if the intersection has been crossed)
- Scheduled Flag (to notify if a vehicle has already been considered in the scheduling process)
- Leader Elected Flag (explained below)

Among these fields, transmission timestamp, vehicle identification number, vehicle motion information and relative position are part of the standardized BSM fields. All other fields can be mapped in the optional VehicleStatus field defined by the

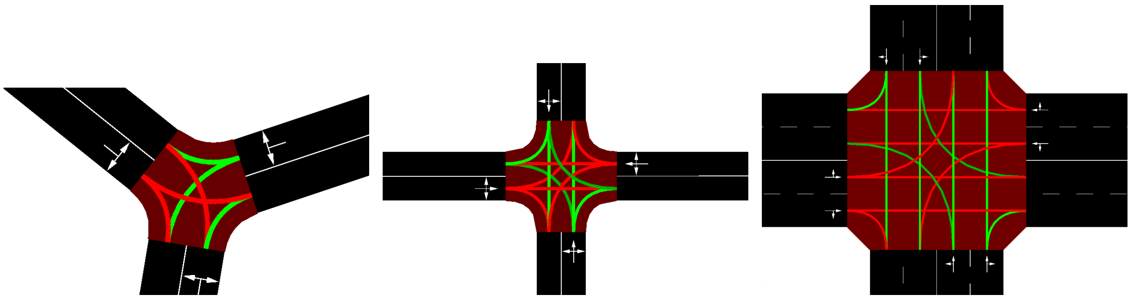


Figure 3.1: Right-to-pass priorities in use case intersections.

SAE J2735 standard [53] for the part II of a BSM, dedicated for non-critical safety applications.
The channel was modeled taking into consideration simple path loss, obstacle shadowing and Nakagami fading attenuation models. To achieve a more realistic mobility environment, buildings alongside the roads merging into the intersection(s) were introduced so as to reduce the line of sight between vehicles. From a navigational perspective, all vehicles are assumed to carry a GNSS-capable device and can place themselves on a map that allows them to identify upcoming junctions.

3.4.2 Intersection and traffic models

We focus on crossroads with no traffic light and no roadside communication infrastructure, making it a full-fledged V2V scenario. We considered different kinds of single- and multiple-lane intersections, which are frequently unregulated in an urban context, i.e., three-way (Figure 3.1a) and four-way junctions (Figure 3.1b). A scenario with two consecutive intersections (Figure 3.1c) is also modeled in order to mimic a sort of "green wave" effect. For every intersection considered, the right-to-pass priorities are shown in Figure 3.1. Green lines represent the trajectories on which vehicles are allowed to cross simultaneously. In particular, bright green lines depict the move with right-to-pass, while dark green lines are used for left turns at intersections and represent priority-based moves.

In order to provide a compact representation of the intersection to be input to the scheduling algorithm, we introduced a bitmap, where every cell corresponds to a position in a lane that can be occupied by a vehicle. If a position is occupied, the cell in the bitmap model is marked with a 1, while a 0 indicates that no vehicle is present in that slot, as shown in Figure 3.2. In detail, in Figure 3.2 the southbound vehicle in green is willing to turn left, resulting in '1's in the main bitmap diagonal. The other green vehicle from the eastbound street is turning right, as shown by the '1' in the bottom left cell of the bitmap. The other two red vehicles are stopped and they do not result in the bitmap representation. A similar bitmap, with '1's is

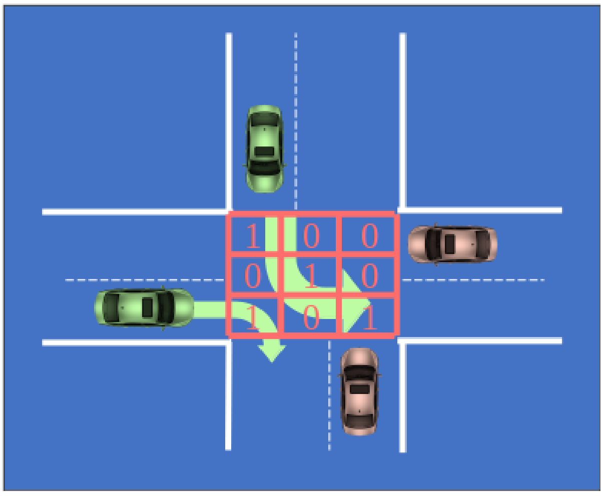


Figure 3.2: Bitmap representation of a legal movement.

used to indicate every possible legal movement of the junction.
Using this bitmap permits to distinguish between legal and forbidden movements of vehicles. There are indeed some configurations of turning intentions at an intersection that lead to a hazard situation (e.g., a vehicle turning left while an other one is crossing straight on). Those configurations are, of course, forbidden, so that only legal moves can be considered in the scheduling process, namely M. The complete set of M legal moves for different type of scenarios (three-way, four-way, multi-lanes junctions, etc.) is indeed the input of the scheduling algorithm.

3.5 V^{3}TL procedure

This Section describes the methodology and steps followed in order to define the V^{3}TL procedure. First, the main idea of the whole procedure is described in a step-by-step fashion, followed by a more detailed analysis of the scheduler algorithm. Finally, we address the algorithm complexity.

3.5.1 System overview

The V^{3}TL scheme works in a cyclic way, scheduling a variable number of vehicles per cycle. Every scheduling cycle is divided into four steps. For simplicity, we will assume that every vehicle has V2V capabilities, can create an LDM and is aware of the V^{3}TL protocol. We will discuss in Section 3.8 how to address situations where these conditions do not apply.
A representation of the scheduling process is reported in Figure 3.3 and a pseudo-code of the whole V^{3}TL procedure is reported in Algorithm 1.

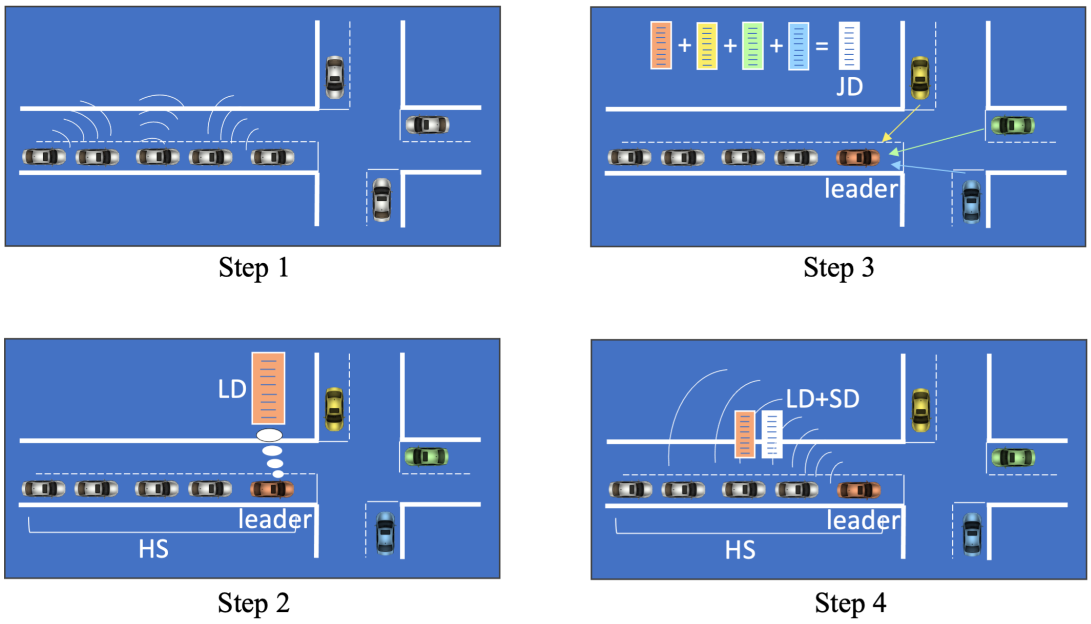


Figure 3.3: Steps of information collection in a cycle.

Discovery procedure

The first step of the cycle starts as soon as a vehicle finds itself within 300m from the next intersection on the map. This phase aims at identifying groups of vehicles approaching the intersection from a similar direction. During this phase, vehicles exchange positional and navigational information broadcasting it through BSMs in V2V communication and thus populate their LDMs. Since the goal of our model is not to regulate *any* intersection at *any* time, the next phase of the V^{3}TL procedure is triggered only when a congested situation is detected, namely as soon as the number of nearby cars is above a threshold N_c. The triggering condition to enter the next phase is as follows: as soon as a car detects $N_c - 1$ vehicles in the two quadrants of the LDM superimposed to its road lane, it sets a flag called Leader Elected Flag in its BSM and starts the next phase of the V^{3}TL procedure. This initial phase is depicted in the Step 1 part of Fig. 3.3, where we can see vehicles exchange BSMs and populate their LDMs.

Leader election and Heading Set identification

When a vehicle receives a BSM with the Leader Elected Flag set, it sets the flag on its own BSMs to propagate the information that the leader election should start everywhere around the intersection. Then, it initiates the procedure for the identification of the leader vehicle and thus the formation of the *Heading Set (HS)*, i.e., the group of vehicles that will collectively be scheduled by V^{3}TL. Ideally, consecutive vehicles find themselves in a HS if they are travelling in the same direction while approaching an intersection and if they are within radio visibility of each other. The vehicle closer to the intersection is elected[1] as leader and it takes it upon itself to form the HS by selecting the $N_c - 1$ vehicles behind, to gather data

[1]Procedures for distributed consensus finding such as those in [46] can be used.

Algorithm 1 V^{3}TL workflow.

```
Data: On BSM received
 1: Update vehicle LDM
 2: if ($N_c - 1$ other vehicles are in same lane) or (BSM has Leader Elected Flag
    set) then
 3:    if (There is no leader elected) then
 4:       Select a leader vehicle
 5:       Set the Leader Elected Flag
 6:    end if
 7:    if (I am the leader) then
 8:       Update LD
 9:    end if
10: end if
11: if (BSM contains LD of other HSs) then
12:    Leaving current HS is no more allowed
13:    if (I am leader) then
14:       Merge received LD with others
15:       if (I have a complete JD) then
16:          Start the scheduling process
17:          Broadcast LS and SD
18:       end if
19:    end if
20: else
21:    if (BSM belongs to a HS closer to the intersection) then
22:       if (The HS of the transmitter allows a join) then
23:          if (I am leader) then
24:             Handover leadership to vehicle behind
25:          end if
26:          Leave the current HS and join the other
27:       end if
28:    end if
29: end if
```

from its HS and share those information among other leaders in neighboring roads stemming from the junction. It is to be noted that, thanks to the propagation of BSMs with the Leader Elected Flag set, also vehicles approaching the intersection from other directions will have started the election procedure.
Joining and leaving a HS is still permitted in this second step of the procedure. Indeed, in a real-life situation, it is frequent that distances between vehicles are reduced in proximity of a crossroad due to traffic conditions. These kind of situations present high variability in the mobility environment, so we decided to focus on this by creating a more flexible procedure in the leader election. In particular, we envisioned the possibility for a leader to resign its leadership and to join another HS as a normal member. This may happen when a leader of a HS recognizes another HS, closer to the junction, on the very same street, with fewer than N_c members. In this case, the leader of the group far from the intersection can join the HS approaching the crossroad, resigning its leadership (and handing over all the data thus far collected as a leader) in favor of the car immediately behind. The new leader can now start the same procedure until N_c cars are grouped in the same HS and the group is thus considered complete. In such a way, we can maximize the number of vehicles we can schedule in a single scheduling cycle, despite the traffic fluctuations in proximity of intersections.
Using such a structure largely simplifies the scheduling algorithm. Indeed, it is possible to consider only a fixed number of vehicle "tiers" (at most N_c tiers of vehicles if we consider a single lane per direction), where the leaders and other cars at the head of the queue in different directions are the first tier, the second tier is formed by the cars directly behind, and so on.

This paradigm works even in scenarios where traffic is unbalanced between directions, with many vehicles queued up in a street and fewer cars approaching the intersection from other roads. Additionally, it can be easily extended to a multi-lane scenario. Vehicles in the HS are thus selected, progressively filling each tier with vehicles from different lanes that are at the same distance from the intersection, until up to N_c are selected. Then, the leader is selected as the rightmost vehicle in the first tier.

The leader is in charge of gathering information from its group members. Those data are saved in a *Leader Dataset (LD)*, which includes all the information of every vehicle in the HS. In particular, for every vehicle, the following are saved: vehicle anonymous identifier, position in the queue, current lane, intersection identifier, signaling light (or, in its absence, the intention of going straight on) and direction (e.g., eastbound, southbound, westbound or northbound in a four-way intersection). The LD is very important for collecting a common knowledge of the scenario in order to schedule all the vehicles synchronously.
This phase is described in the Step 2 part of Fig. 3.3, where we reported the leader election, the Heading Set creation and the Leader Dataset compilation.

Leaders information exchange

When leaders approach the intersection, they start sharing the LD of their HS, thus initiating step three of the procedure. In such a phase, joining or leaving a HS is no longer allowed and leaders start to merge LDs from different directions into a single *Junction Dataset (JD)*.
With the JD, leaders have a full view of the intersection and they can compute the scheduling solution, called *Solution Dataset (SD)*, as explained in the following subsection. Since leaders use the same JD as input, they will nominally output the same SD.
Step 3 of Fig. 3.3 reports this phase of the V^{3}TL procedure, where leaders exchange Leader Dataset, then merge them into the Junction Dataset.

Scheduling procedure

In the fourth and final step, leaders compute and broadcast the SD together with their LDs. In such a way, upon a SD reception, a vehicle can search in the LD if its identification number is present for the current schedule cycle. If this is the case, it sets the Scheduled Flag in its BSM and follows the schedule according to the SD. Every vehicle that was not part of the scheduled HS at the time of the JS creation will not find its identification number in the broadcast LD. This can happen if the vehicle is part of the HS that follows or if it has just reached the intersection and is not yet part of a HS. Those vehicles will not be scheduled and will have to wait for a new leader election procedure to start in the following cycle. It is possible to see the phase here described in the Step 4 of Fig. 3.3.

3.5.2 Scheduling solution computation

Given a configuration defined in the *Junction Dataset*, at most $H \cdot N_c$ cars can be scheduled at a time, where N_c is the maximum number of vehicles in a *Heading Set* and H is the number of headings in the intersection. Note that this is independent from the number of lanes P, since up to N_c vehicles are grouped per direction and not per lane. Among all the possible turning configurations of those $H \cdot N_c$ vehicles, only the M legal moves involving the first tier are selected, nominally $A_{i,1}$. For each $A_{i,1}$ with $i = 1, \dots, M$ the number of passing vehicles for that move are computed, $c(A_{i,1}) \in [1, H \cdot P]$, and likewise the number of generated stop-and-gos, $e(A_{i,1}) \in [0, H \cdot N_c - 1]$. The scheduling algorithm now proceeds as a decision tree, where every node $A_{i,j}$ is a legal movement characterized by its own $c(A_{i,j})$ and $e(A_{i,j})$ for $i = 1, \dots, M$ legal movements and for $j = 1, \dots, L$, where L represents the maximum number of levels in depth that we are willing to consider in order to curb complexity. Furthermore, every node $A_{i,j}$ has associated cumulative values of passing vehicles $C_{i,j}$ and stop-and-gos $E_{i,j}$ resulting from the sum of individual $c(A_{i,j})$ and $e(A_{i,j})$ along its branch.

For every node of the tree, multiple branches are generated, one per legal turning configuration. Notice that at level L, up to M^L values of $C_{i,j}$ are computed. When all the branches of the tree down to level L are computed, the scheduling solution $s(L)$ is chosen as the branch that maximizes $C_{i,j}$ at level L. In case of a lingering tie, the branch that both maximizes $C_{i,j}$ and minimizes $E_{i,j}$ at level L is taken as a solution. If there are two or more such branches, the solution is chosen randomly among them. It is to be remarked that, depending on the choice of L, less than $H \cdot N_c$ cars can be scheduled in this way. Therefore, this procedure is repeated in an iterative way until all the vehicles present in the *Junction Dataset* have been scheduled, as described in the pseudo-code of the scheduling procedure in Algorithm 2.

Using such a paradigm, leader-vehicles are able to compute a scheduling solution in real time and broadcast it before the vehicles in HS reach the intersection. In order to minimize the size of the *Solution Dataset* on BSMs, a compact representation can be used. It is a matrix representation, composed by $H \cdot P$ columns and $r \in [1, H \cdot N_c]$ rows, where every row represents a legal movement selected by the scheduling algorithm. Therefore, every element refers to a single vehicle and contains the instructions for a specific vehicle finding itself at the head of the lane after the previous legal movement. In particular, every element of the matrix SD(i;j) is a tuple composed by the anonymous random identification number of the vehicle to which the instruction refers and the identification number of the road and the lane to which access has been granted. If the latter refers to the street on which the vehicle is currently traveling, then it means that the vehicle is forced to brake and stop before the intersection. If there is no vehicle in a given direction, the special element $(-1; -1)$ is used for a void position.

Algorithm 2 Scheduling algorithm.

```
Data: Complete JD
 1: while (There are still vehicles to schedule in the JD) do
 2:     j=0
 3:     while ($j < L$) do
 4:         Select all legal movements of first available tier
 5:         for all (Legal movements $A_{i,j}$) do
 6:             Compute $c(A_{i,j})$, $e(A_{i,j})$, $C_{i,j}$ and $E_{i,j}$
 7:         end for
 8:         j++
 9:     end while
10:     Select the best branch of the tree according to selection explained in text
11:     Update SD with current tier solution
12: end while
```

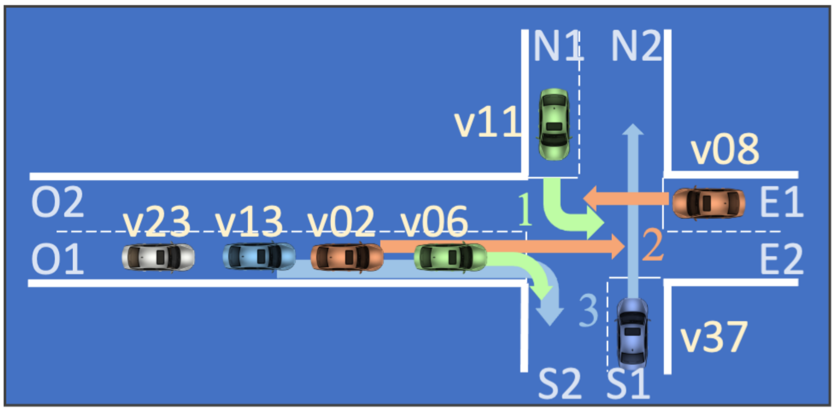


Figure 3.4: Example of three legal actions.

Figure 3.4 depicts a possible scenario after a *Solution Dataset (SD)* has been delivered to vehicles in a single-lane, four-way intersection. In particular, we can see the green vehicles from southbound and eastbound streets that are allowed to pass together, while cars from northbound and westbound are forced to stop. On the next move, the orange cars from eastbound and westbound can cross together. Finally, another vehicle from eastbound, in blue, is allowed to pass together with a vehicle from northbound. The corresponding SD for the scenario of Figure 3.4 is reported in Table 3.1.

3.5.3 Algorithm complexity

It is conceivable that, for small values of N_c, a brute-force approach can be computed with all the possible combinations of M legal moves for all the vehicles in the scheduling cycle. The output of the brute-force approach could be then stored a priori on vehicles for different values of N_c and for the most common values of $H \in \{3,4\}$ and of $P \in \{1,2\}$, i.e., three-way or four-way intersections with one or two lanes per direction. However, the complexity of the problem grows exponentially with respect to the number of L levels considered.
If we consider a H-way P-lanes intersection, we can count up to $H \cdot P$ vehicles in

N1	E1	S1	O1
(v11, E2)	(v8, E1)	(v37, S1)	(v6, S2)
(-1, -1)	(v8, O2)	(v37, S1)	(v2, E2)
(-1, -1)	(-1, -1)	(v37, N2)	(v13, S2)

Table 3.1: Solution Dataset of the example in Fig. 3.4.

the first tier. Each of those vehicles can choose among $P \cdot (H-1)$ possible direction (since the u-turn is not allowed) or to brake and stop before the junction, leading to $P \cdot (H-1)+1$ different choices. Considering only the first move, the total number of possible combinations (legal or not) are at most $[P \cdot (H-1)+1]^{H \cdot P}$, since there are $P \cdot (H-1)+1$ decisions for $H \cdot P$ vehicles. We remark that the possible movements are $[P \cdot (H-1)+1]^{H \cdot P}-1$, since we discount the case of none of the cars making any move. Considering now a decision tree with L levels in depth, we can count $\{[P \cdot (H-1)+1]^{H \cdot P}-1\}^L$ possible permutations.

However, the solution space of our problem is defined by a subset of the possible permutations above, defined by the number of M legal movements for each level L. The value of M must be manually computed for each combination of H and P, for example:

$$M = 13 \;\; for \;\; H = 3, \;\; P = 1$$

$$M = 49 \;\; for \;\; H = 4, \;\; P = 1$$

$$M = 311 \;\; for \;\; H = 4, \;\; P = 2$$

Consequently, the complexity of a brute-force approach will be exponential with $\mathcal{O}(M^L)$. Using a decision tree paradigm, we exploit a low-complexity heuristic approach with an average complexity $\mathcal{O}(L \log M)$.
It is important to notice that the L parameter refers to the number of levels in depth per iteration of the decision tree, not the number of vehicle tiers scheduled. The only chance of having L equal to the number of tiers is to schedule for each level the maximum number of vehicles, so that $c(A_{i,j}) = H \cdot P$, $\forall i = 1, \ldots, M$ and $j = 1, \ldots, L$, i.e., all cars in the first tier turning right. Since this probability is negligible in a realistic scenario, the computed complexity for scheduling a complete tier of vehicles is usually higher than for scheduling a level in depth of the decision tree.

3.6 Performance evaluation setup

Results shown in this section are obtained from simulations in which every vehicle is modeled with an On-Board Unit (OBU) equipment capable of transmitting and receiving BSMs.
SUMO (Simulator of Urban Mobility) [54] was used as a mobility simulator, while OMNeT++ [55] is the tool used in order to model the transmission between vehicles in our network. Finally, the Veins simulator [56] was used to interconnect SUMO and OMNeT++.

3.6.1 Environment

In order to evaluate the performance of the V^{3}TL procedure, three use cases were analyzed. We focused on a single-lane three-way intersection (H = 3 and P = 1), a single-lane four-way intersection (H = 4 and P = 1) and a scenario with two consecutive double-lane four-way intersections (H = 4 and P = 2). We believe that these use cases represent the vast majority of unregulated intersection types in an urban scenario.

Regarding generation flows, vehicles are created according to Poisson distributions with different values of inter-arrival time λ. In particular, for every use case, we developed a generation process where traffic is balanced between directions and λ is constant for every incoming road. In addition, for the single-lane three-way scenario and the single-lane four-way scenario, we also analyzed an unbalanced traffic model, where Poisson generation processes with different λ were used for every direction. Regarding the consecutive intersections scenario, the traffic distribution is unbalanced even if we use the same λ value for every input road. This is due to the particular mobility environment, which creates a main flow between intersections with the aim of imitating a green-wave effect.
As a result, our system is dependent on the amount of traffic we inject into the model: we are interested in high traffic densities so as to highlight the performance of our solution. In particular, we have observed that values of inter-arrival time with $\lambda > 8$ s/veh per direction results in too shallow densities which fail to trigger the V^{3}TL procedure.

The evaluation of the performance of our model was conducted through a comparison with other two scenarios.
Firstly, we inspected a simple scenario, referred to as *Unregulated*, where vehicles cross the intersection without being scheduled, therefore, in a completely unregulated manner. In this case, vehicles follow the "right-before-left" priority model used by SUMO.
A second scenario with a fixed traffic light was then developed, referred to as *Traffic Light*. Since, by default, SUMO emulates non-realistic control phases, the Webster method [57] was used. The Webster method is an algorithm used by civil engineers to set traffic light phases according to the current incoming traffic. The goal of the Webster method is to define an optimum cycle length C and optimum green light phases G_i using the following formulas:

$$C = \frac{1.5 \cdot T + 5}{1 - \sum_k y_k} \tag{3.1}$$

$$G_i = \frac{(C - T) \cdot y_k}{\sum_k y_k} \tag{3.2}$$

where T is the total lost time for every cycle due to human reaction times. In the formulas, y_k is the critical flow rate, computed as the ratio between number of vehicles per direction and the saturation flow, i.e., the maximum number of vehicles that can pass the intersection in an hour. For every single-lane use case considered, we will set a saturation flow of 3600 veh/h for both balanced and unbalanced models.

3.6.2 V^{3}TL parameters

First of all, the maximum number of vehicles that can be grouped into a *Heading Set*, namely N_c, was chosen to be 6, so as to limit the size of the queue to approximately 30 m from the intersection[2].
Recall that we defined as L the number of levels in depth for every iteration of the scheduler. On every iteration of the scheduler, the local optimum solution across L rows is written in the *Solution Dataset* matrix made of $r \in [1, H \cdot N_c]$ rows in total. However, building a complex tree every iteration dramatically affects the algorithm processing time. Since leader vehicles have to compute the scheduling solution on the fly, the decision tree procedure has to abide to tight time limits. The presence of buildings at the intersection can hinder communication between a leader and vehicles coming from perpendicular directions. Therefore, leaders may only receive transmissions from nearby crossing roads approximately 5 m before stopping at the intersection. Considering a BSM transmission frequency of 10 Hz and negligible propagation and processing times, the scheduler procedure has to output a solution within 260 ms for a scenario with vehicles moving at 50 Km/h[3]. In a single-lane four-way scenario, measuring the time it takes for the scheduling algorithm to be executed on the hardware used for our simulations (2.5-GHz Quad-Core i7), we observed 123.9 ms for $L = 2$ and 434.49 ms for $L = 3$. In order not to exceed the processing time limit of 260 ms, a scheduling algorithm with $L = 2$ levels in depth was used.

3.7 Results

We evaluated the performance of the system by inspecting two main metrics: the average number of cars per minute passing the intersection and the average number of stop-and-go maneuvers per vehicle. The first metric measures congestion, i.e., the clearance rate for the intersection. The number of stop-and-go maneuvers, instead, is considered as a secondary goal. A vehicle that performs fewer stop-and-gos on average will pollute less. Additionally, for the consecutive intersections scenario, a fairness metric was introduced to evaluate how much the flow of vehicles crossing

[2]the vehicle length of a standard passenger car in SUMO is 4.3 m.

[3]360 ms to drive 5 m at 50 Km/h minus the worst case waiting time for the next BSM transmission, 100 ms.

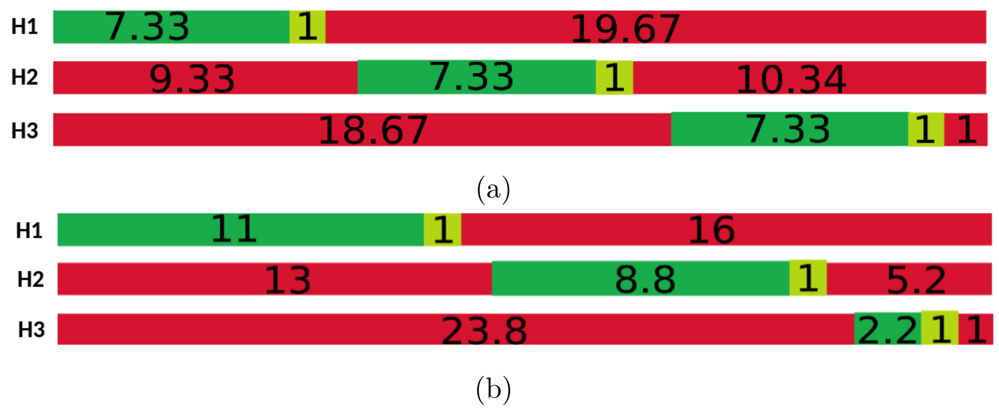


Figure 3.5: Traffic light phases of a three-way intersection (with headings H1, H2 and H3) computed with the Webster method. The phases are expressed in seconds.

both intersections is favored with respect to other flows.
All results are averaged over ten simulation runs, each corresponding to a simulated time of one hour.

3.7.1 Three-way intersection

The first use case we analyzed is a three-way intersection with a single lane per direction. For the balanced scenario, we generated vehicle flows of $v_1 = v_2 = v_3 =$ 600 vehicles, with critical flow rates of the Webster method $y_1 = y_2 = y_3 = \frac{1}{6}$ and inter-arrival time $\lambda = 6$ s/veh per direction. Assuming the lost time per phase $T = 6$ s (i.e., 1 s of amber phase and 1 s of clearance interval per phase), we obtain an optimal cycle length $C = 28$ s and green phases $G_1 = G_2 = G_3 = 7.33$ s. For the unbalanced scenario, vehicle flows of $v_1 = 900$ vehicles, $v_2 = 720$ vehicles and $v_3 = 180$ vehicles were modeled with inter-arrival times of $\lambda_1 = 4$ s/veh, $\lambda_2 = 5$ s/veh and $\lambda_3 = 20$ s/veh. The corresponding critical flow rates are $y_1 = 0.25$, $y_2 = 0.2$ and $y_3 = 0.05$, resulting in $C = 28$ s cycle length and $G_1 = 11$ s, $G_2 = 8.8$ s and $G_3 = 2.2$ s green phases (using $T = 6$ s). Traffic light phases of balanced (Fig. 3.5a) and unbalanced (Fig. 3.5b) scenarios are reported in Figure 3.5.

In Table 3.2 the performances of V^{3}TL scheduler are shown for the three-way intersection use case. In particular, we can see the average number of cars per minute passing the intersection and the average number of stop-and-gos per vehicle per minute. Metrics are reported for all comparison scenarios and for every traffic distribution. From the table, it is possible to see how the scheduler procedure outperforms both unregulated and traffic light scenarios. In particular, the scheduler procedure improves the unregulated scenario by 2.68% of passing cars per minute and by 17.07% of stop-and-go maneuvers for the balanced scenario (4.61% and 15.43% for the unbalanced traffic).

It is possible to notice how the choice of a traffic light in this intersection type is penalizing since a dedicated phase is needed for every heading (a four-way intersection only needs two phases). The V^{3}TL scheduler significantly improves the traffic light scenario, as reported in Table 3.2.

3.7.2 Four-way intersection

The second use case focuses on a single-lane, four-way intersection. In this model, a balanced scenario was created using $v_1 = v_2 = v_3 = v_4 = 900$ vehicles in every direction with inter-arrival times of $\lambda = 4$ s/veh. The corresponding critical flows are $y_{1\wedge3} = y_{2\wedge4} = 0.25$, resulting in $C = 22$ s optimum cycle length and two green phases of $G_{1\wedge3} = G_{2\wedge4} = 9$ s (we used a total lost time of $T = 4$ s, i.e., 2 s per phase divided into amber phase and clearance interval). The unbalanced scenario uses vehicle flows $v_1 = v_3 = 1440$ vehicles and $v_2 = v_4 = 360$ vehicles with corresponding inter-arrival times $\lambda_1 = 2.5$ s/veh and $\lambda_2 = 10$ s/veh. Following the Webster method, we computed critical flow rates $y_{1\wedge3} = 0.4$ and $y_{2\wedge4} = 0.1$, optimum cycle length $C = 22$ s and green phases $G_{1\wedge3} = 14.4$ s and $G_{2\wedge4} = 3.6$ s. Both the traffic light phases of balanced (Fig. 3.6a) and unbalanced (Fig. 3.6b) traffic are reported in Figure 3.6.

Table 3.3 reports the average number of passing cars per minute and the average number of stop-and-go maneuvers per vehicle per minute for both balanced and unbalanced traffic distributions of a four-way intersection. In this use case, we have two flows of vehicles from opposite directions of the junction, resulting in only two traffic light phases. For this reason, the traffic light scenario handles more vehicles per minute than the unregulated model. Conversely, the number of stop-and-gos increases due to the phase behavior of traffic lights. Although, V^{3}TL scheduler performs better with respect to both unregulated and traffic light for balanced and unbalanced scenarios with improvements up to 41.22% for scheduled vehicles and up to 34.53% for stop-and-go manoeuvres.

Traffic Distribution	**Scenario**	**Cars/min**	**Stop-and-go/min**
Balanced	V^{3}TL	48.24	1.02
	Unregulated	46.98	1.23
	TL	32.14	4.43
Unbalanced	V^{3}TL	39.46	1.37
	Unregulated	37.72	1.62
	TL	26	6.29

Table 3.2: Three-way intersection results for the V^{3}TL system, the unregulated case and the traffic light case.

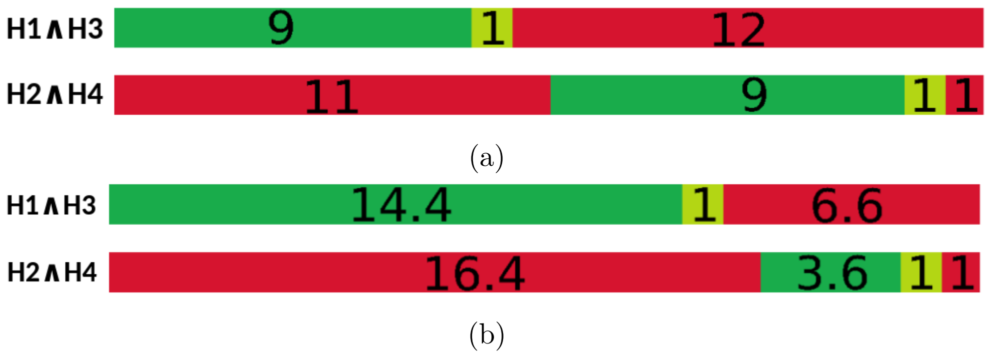


Figure 3.6: Traffic light phases of a four-way intersection (with headings H1$\wedge$H3 and H2$\wedge$H4) computed with Webster method. Numbers reported in figure are expressed in seconds.

3.7.3 Consecutive four-way intersections with double-lane

Eventually, a more realistic and complex scenario was taken into account. We modeled two consecutive four-way intersections in order to study the performance of our system for a continuous flow of vehicles in a sort of green-wave effect. For this scenario, every direction is modeled as a double-lane road, so we measured a saturation flow of 4800 veh/h per intersection. It is important to underline how the traffic distribution here differs from the other use cases: due to the turning choices of vehicles at intersections, introducing vehicles with the same inter-arrival times from every direction will not lead to a balanced traffic scenario. For this reason, a single distribution with vehicle flows $v_1 = \cdots = v_6 = 1200$ vehicles was developed, where a third of vehicles cross both intersections while others reach their destinations after passing one intersection only. Inter-arrival time is $\lambda = 4$ s/veh for all directions and the critical flow rates are $y_{1\wedge3} = y_{2\wedge4} = 0.25$ for both intersections. This results in two optimal cycles lengths of $C = 22$ s and green phases $G_{1\wedge3} = G_{2\wedge4} = 9$ s for both crossing cycles ($T = 4$ s, i.e., 2 s per phase divided in amber phase and

Traffic Distribution	**Scenario**	**Cars/min**	**Stop-and-go/min**
Balanced	V^{3}TL	37.96	1.56
	Unregulated	26.88	1.85
	TL	28.8	2.21
Unbalanced	V^{3}TL	35.14	0.91
	Unregulated	27.64	1.39
	TL	33.82	1.23

Table 3.3: Four-way intersection results for the V^{3}TL system, the unregulated case and the traffic light case.

clearance interval), as reported in Figure 3.6a.

For this use case, we are also interested in measuring the fairness of the system. Indeed, an unfair scheduling procedure may lead to long queues on roads between intersections. In order to analyze this metric, we introduced a fairness index F using Jain's Fairness formula [58]:

$$F = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{3.3}$$

where n is the number of roads in the scenario (i.e., 8 for this use case since we have two four-way intersections) and x_i, $\forall i = 1 \ldots n$ is the ratio between the measured output traffic of a direction and the ideal output traffic equally shared between directions. Values of measured x_i for every scenario are reported in Table 3.4. Since x_i is a ratio of output traffic, the sum of all directions coincides with the number of roads in our scenario.

Jain's Fairness formula is broadly used in computer networks to measure the fairness of the system considering the different data flows. It reports an index between 0 and 1, which refers to the fairness percentage of the system.
The results of the consecutive intersections use case are reported in Table 3.5. From the table, we can see how the scheduler procedure brings significant improvements for both passing cars per minute and stop-and-go manoeuvres per vehicle per minute. Additionally, it also increases the fairness of the considered system with a significant enhancement of 32.05% with respect to the unregulated scenario, while keeping in par with a traffic-light regulated intersection.

3.8 Discussion and conclusion

This study presents V^{3}TL, a V2V Virtual Traffic Light system for managing traffic at unregulated intersections in an entirely distributed manner. We defined a

Heading	**V^{3}TL**	**Unregulated**	**Traffic Light**
1	0.29	0.42	0.28
2	0.39	0.21	0.26
3	1.27	0.58	1.07
4	1.15	0.48	1.11
5	1.28	0.56	1.44
6	1.16	1.92	1.42
7	1.19	1.76	1.15
8	1.27	2.06	1.24
SUM	8	8	8

Table 3.4: Values of x_i per direction for Jain's Fairness formula.

Scenario	**Cars/min**	**Stop-and-go/min**	**Fairness**
V^3TL	78.62	0.98	87.1%
Unregulated	55.76	1.62	65.96%
TL	73.54	1.58	84.45%

Table 3.5: Consecutive four-way intersections results for the V^3TL system, the unregulated case and the traffic light case.

procedure made of four steps in order to gather all the data required for a complete view of the intersection status and for a proper scheduling solution.

A complete description of our scheduling algorithm is also given, describing how decision trees were built for every iteration to select local optima in a heuristic way. We focused on three use cases that resemble the majority of unregulated intersection types in an urban environment. The considered scheduling criteria are the number of vehicles crossing the intersection in a minute and the number of stop-and-go maneuvers per vehicle, identified as the main metrics for traffic congestion and pollutant emissions, respectively. We also analyzed the fairness of the system in a complex, two-intersections scenario.

Simulations show how the proposed system leads to significant improvements concerning unregulated intersections and traffic-light-regulated ones. Enhancements have been observed for all inspected metrics in both scenarios with balanced and unbalanced traffic distributions.

As with all cooperative vehicular applications, our system guarantees the performance we have illustrated only for very high penetration rate values. Solving this issue is not the purpose of the present work. Nevertheless, a little insight of a fallback procedure to disable the scheduling process in the case of one or more unequipped cars is still given here.
Since a vehicle not equipped with V2V hardware is uncapable of communicating with others in any way, the first problem is how to identify the presence of such a car. A valid method could be using car sensors: if a neighbor of an unequipped car detects, through front or rear sensors, the presence of a vehicle unidentified via V2V communication, then it could assume its presence, in what is usually known as “collective perception” [59]. In such a situation, the complex scheduling process described in this study may not produce a valid outcome, since it assumes the full knowledge of the vehicles at the intersection. For this reason, when an unequipped car is sensed, the V^3TL system performs a fallback procedure consisting of a scheduling solution computed on the vehicles ahead of the unequipped one and a “special green phase”, with all other cars stopped, is reserved to let the unequipped vehicle transit. We will explore such a variant in our future work, where we also plan to inject real traffic flows into our system and inspect its behavior for more complete, urban-scale scenarios.

Chapter 4

Developing and Testing of V2N-based Applications

Vehicle-to-Network (V2N) transmission systems are becoming increasingly popular among automotive applications that rely on vehicular communication. Every year, a growing number of European projects and live demonstrations (e.g., C-Roads projects, Rainbow project and the 5GAA live demonstration) choose to concentrate on this important use case. A V2N-based application, on the other hand, is a complicated model since it usually necessitates the full set up of a dedicated network server. As a so-called Broker server, the network node (also known as the cloud node) is responsible for managing the flow of messages. For the management of a Broker server, the Advanced Message Queuing Protocol (AMQP) [60] is the most commonly used application protocol.
In this chapter, we will discuss how to replicate a cloud node for V2N-based applications that manage an AMQP Broker server. In addition, we describe two real-world use cases in which a cloud node we developed was utilized.

4.1 Research motivation

In the last five years, we have witnessed the development and standardization of new communication technologies. The 5G mobile network standard assures high throughput rates, broad bandwidth, and ultra-low latency requirements. All of these advantages make it feasible for significant advancements to be made in a variety of technical fields. The 3GPP Release 16 standard [61] specifies that a designated objective of 5G is vehicular communication. Indeed, during the past decade, there has been a marked increase in interest in vehicular applications, with a large amount of money being invested in research into assisted and autonomous driving vehicles, connected cars, and vehicular communication, among other things.

When it comes to the hardware, these studies have resulted in a large number of systems known as ADAS (Advanced Driver Assistance Systems), which are comprised of sensors, cameras, LiDAR (Light Detection and Ranging), Radar (Radio Detection and Ranging), GPS (Global Positioning System), and GNSS (Global Navigation Satellite System) technologies. Having all of these components in place allows a vehicle to achieve a high level of automation and to undertake driver-less movements in a controlled environment. An example of an automated system is the Traffic Jam Pilot, installed on the Honda Legend and the Audi A8L, which achieves a degree of automation of three out of five. Despite the considerable technical advantage, ADAS will not be able to create a totally driver-less car without relying on vehicular communication. A top-level sensor is ineffective in a variety of scenarios, such as when two cars are not in direct line of sight with one another or when, due to fast travel speed, a sensor with a meter-based range generates an alarm a few milliseconds before the collision occurs. Furthermore, when compared to vehicular communication technologies, ADAS are exceedingly costly. When participating in a test case, it is not uncommon to see automobiles equipped with camera and sensor equipment that is far more expensive than the vehicle itself.

On the other hand, the poor penetration rate of communication-based vehicle applications is a challenging factor. Consider the possibility of having a vehicle that is capable of exchanging data with other cars in a Vehicle-to-Vehicle (V2V) manner. If this vehicle is the only one in the region that is equipped with V2V technology, then the economic investment made for the communications protocol on that vehicle will be unusable and worthless. Furthermore, automotive applications are frequently dependent on information provided by several vehicles. The result is that, as demonstrated in [62] for a collision avoidance application, the lower the penetration rate, the worse the overall performance of the system. In order for a Vehicle-to-Everything (V2X) device to be considered worthwhile for purchase by a user, at the very least a complete application must ensure that the ultimate customer receives a high-quality service, even in an environment with a 0% penetration rate (i.e., relying on only one car equipped in all the selected area). This may be accomplished through the use of Vehicle-to-Infrastructure (V2I) communication-based applications. They can provide drivers with security and infotainment services by utilizing Road Side Units (RSUs) that are strategically placed along the road and are capable of collecting and analyzing traffic data. However, because a large number of Road Side Units are required to cover a given region, the development, maintenance, and deployment costs of Road Side Units are quite high.

Applications relying on Vehicle-to-Network (V2N) communication are thus the most suited ones for providing a complete service even with a 0% penetration rate. Because of their centralized nature, once a cloud node capable of receiving the flow of data provided by cars has been established, each vehicle may benefit from a comprehensive service that is offered independently of the others. As a result, several organizations are investing in this sort of vehicular communication as a short-term solution to the problem of low penetration rates.

4.2 Main contributions

Throughout this chapter, we will discuss the technique that was used in the construction of a Torino Digital Mobility online platform, an AMQP Broker server, and a City Aggregator online platform. Those objects were used during the development of two projects:

- the 5GAA live demonstration, which took place on November 14th, 2019 in Turin, Italy. In this big event we collaborated with Links Foundation, 5T S.r.l., TIM Italia S.p.a. and Stellantis group (which was named FCA at the time of the project). Our efforts were devoted to the development of the Urban Geo-referenced Alert (UGA) use case. The navigation system with real-time mobility information created for this work represents the final product of the project. The corresponding online platform, which can be found at [63], is no longer operational. A comparable simulated version has been created and is accessible at [64];
- the European Rainbow project, which is now under progress. In this project, we worked in collaboration with the Links Foundation and the Stellantis group on the construction of an Urban Mobility Demonstrator (UMD) that was specifically targeted at the city of Turin. In this use case, we want to handle flows of vehicle messages using an AMQP Broker server. Messages are used to warn drivers of a potential road hazard ahead. Furthermore, the visualization of events on a map environment is supported by an online platform available at [65].

The next section provides an overview of automotive standards for communication and message exchange between vehicles. The following section provides a full description of the Advanced Message Queuing Protocol (AMQP) and how to configure a message Broker server. A little insight on how to create an online platform is given in the next section. Following that, we discuss in two separate sections the methodological characteristics that we employed in the two projects listed above. Eventually, final remarks and future work bring the chapter to a conclusion.

4.3 Standards for vehicular communication

The basic idea behind connected cars, and VANETs in general, is to enable connections amongst all road participants (e.g. cars, bicycles, road signs, semaphores, etc.) and to improve present mobility through a variety of applications ranging from safety to infotainment. A strong, resilient, and scalable vehicular network is a critical component of such a concept since it is accountable for achieving the stringent standards set by the overlying applications.

The applications enabled by vehicular networks can be divided into three categories: 1) Road safety applications, 2) Traffic efficiency and management applications and 3) Infotainment applications [66]. Because of the socio-economic effect that car accidents and injuries have across the world, road safety applications are the most promising in terms of market penetration. Instead, traffic efficiency and management applications concentrate on effectively managing traffic flow in order to achieve the goals of more sustainable and smart mobility. Eventually, infotainment services, which will include Social Network feature integration, gamification strategies and marketing goals, will improve the overall user experience, enabling a multitude of innovative services that will transform the user's perspective on mobility.

At the time of writing, Dedicated Short-Range Communications (DSRC) [67] or cellular networks are the two available communication protocols for V2X. IEEE 802.11p is used as an access layer in DSRC-based technologies such as the American Wireless Access for Vehicular Environment (WAVE) [68, 69, 70, 71] and the European ITS-G5 [72]. Cellular-based standards such as C-V2X reuse the upper layer standardized by IEEE and ETSI by changing the access layer and enabling native communication with the cellular network infrastructure. Cellular-based solutions for V2X communications are getting more and more attention from the industry and academic world. Release 14 [73] of the 3GPP incorporated the notion of V2X for cellular networks, introducing two modes of operation:

- using the PC5 interface, which allows for one-to-many communication and direct connections between UEs (User Equipments);
- through the Uu interface, which allows connection with network infrastructure, namely the eNB (E-UTRAN NodeB), which serves as a base station in an LTE network.

Communication through the PC5 interface can be either coordinated by an eNB, or in a completely independent way. As a result, V2V-based applications can be enabled via the PC5 interface. Communication through Uu is only supported under network coverage, with the UE receiving V2X messages (unicast or broadcast) through the downlink and sending V2X messages via the uplink. It is possible to obtain access to solid edge computing capabilities, cloud apps, and V2N-based services in general via the Uu interface.

Despite growing interest in V2N communication from the scientific community, there is currently no standard that covers all of the many forms of V2N communication. Most notably, there is no clear description of how to route flows of vehicular messages via the network, nor is there a clear definition of which protocol should be used to manage a cloud node. Following the guidelines from earlier European

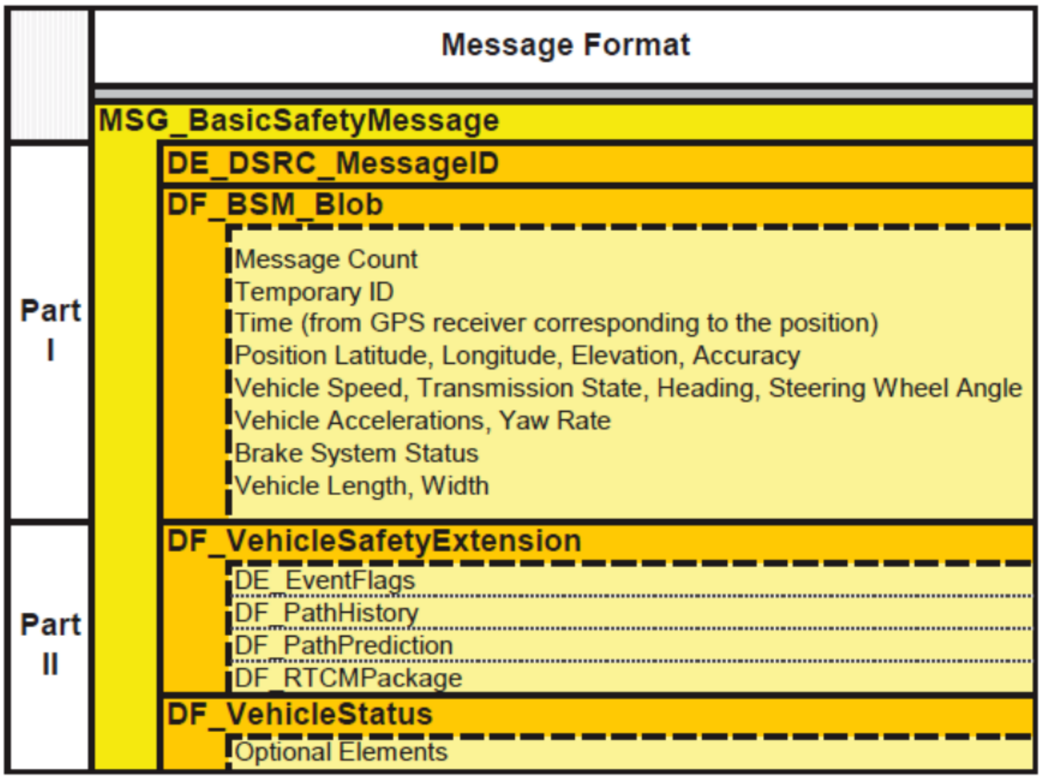


Figure 4.1: Basic Safety Message definition.

initiatives on the same issue (e.g., C-Roads projects), we will concentrate our attention in this chapter on two test cases dealing with V2N communication. In the next subsection, we will examine a small subset of the messages, as specified by American and European standards, that are essential to understanding the contents of this dissertation.

4.3.1 Definition of basic vehicular messages

Although WAVE does not specify a specific message format, the standards J2735 [53] and J2945 [74] were defined by the Society of Automotive Engineers (SAE) for this purpose. They are considered to fit the needs of the DSRC environment, which has limited resources, particularly in terms of bandwidth. For these reasons, they designed a message sublayer for safety-related applications where the information is enveloped in short packets that can be frequently and efficiently broadcast in an unacknowledged delivery mode. A dense encoding of information in defining the messages is essential to maximize the overall Cooperative-ITS (C-ITS) system capacity, which is done by Abstract Syntax Notation revision One (ASN.1) encoding. In the J2735 standard, 17 message types are defined having

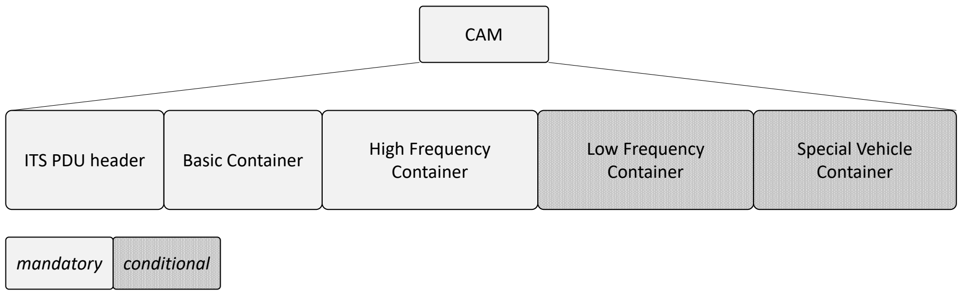


Figure 4.2: Format of a CAM message.

purposes ranging from safety to GPS (Global Positioning System) correction. This paragraph will only discuss the Basic Safety Message (BSM) and will then move on to two alert messages: Emergency Vehicle Alert (EVA) and Intersection Collision Avoidance (ICA).

- BSMs are used by most vehicular applications to exchange safety data regarding the state of the vehicle. These messages provide data required for safety and non-safety applications and they are broadcast at a high frequency to nearby cars and road entities. To avoid channel congestion, the transmission rate is adaptive and is determined by the congestion management methods used. The maximum transmission frequency is set at 10 Hz. A BSM is divided into two parts, as reported in Fig. 4.1: the first must be included in every BSM and contains data such as latitude, longitude, speed, heading, and so on. The second section is optional and offers additional safety-related information such as: path history, path prediction, exterior light status, etc.;

- EVA messages are used by emergency vehicles to broadcast warning messages, notifying nearby vehicles that an emergency situation is happening in the vicinity. This message type contains data fields that describe the event and offer advice and recommendations to the cars in the area;

- ICA messages are instead used to broadcast information concerning potential collisions. They are delivered by other vehicles or by RSUs when they enter a dangerous intersection without the right of way.

Differently from WAVE, the European ITS-G5 starts with the definition of a basic set of applications dedicated to road safety. Some of the most important applications based on the basic services of Cooperative Awareness (CA), Decentralized Environmental Notification (DEN) and Infrastructure to Vehicle Information (IVI) are presented and described in [75, 76, 77, 78]. In particular, [75] introduces the Road Hazard Signaling (RHS) application thought for the increased awareness of both ITS stations and drivers, through the dissemination of Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs)

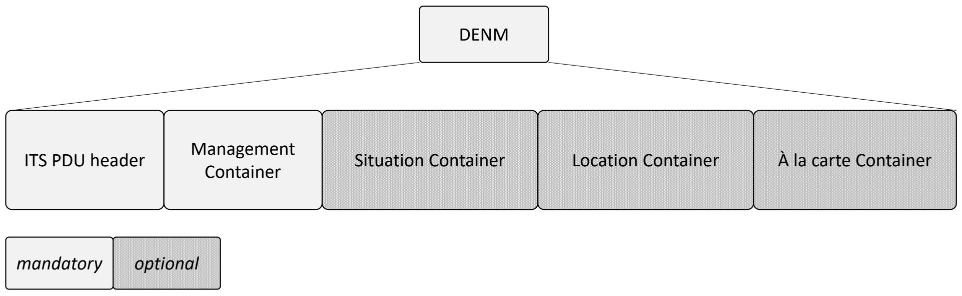


Figure 4.3: Format of a DENM message.

signaling the presence of hazardous situations. In [78] ETSI described the services to support communication between infrastructures and other traffic participants, defining several message types. We will concentrate on the Infrastructure to Vehicle Information Message (IVIM) for the purposes of this chapter.

Because CAMs, DENMs and IVIMs are so important to the topics covered in this chapter, the entities in charge of their creation will be introduced in the following. All the fields present in the CAM, in the DENM and in the IVIM messages are defined in a special common data dictionary [79].

- The facilities layer object that executes the CAM protocol is the CA basic service, as specified in [80]. The frequency of CAMs is determined by the dynamics of the generating vehicle, and it is considered to reduce channel congestion. The CAM inter-arrival time is set between 100 milliseconds and one second. The overall goal is to increase the frequency of fast-moving or fast-turning vehicles while simultaneously reducing the frequency of vehicles that are stationary or traveling slowly. The cooperative awareness granularity is preserved in this way, even when cars are moving fast (for example, on highways), and the general position perception is less scattered. [80] also defines the CAM format, which is represented in Fig. 4.2;

- the DEN basic service, as specified in [81], is a facilities layer application support module that runs the DENM protocol. Unlike the CA basic service, which creates CAMs on a regular basis without involving the higher layers, the DEN basic service is activated by applications and offers an asynchronous mechanism for notifying other ITS stations. The DEN basic service should offer primitives to produce, update, and terminate the DENM during transmission. When the information is received, it should be dispatched to the higher layers when necessary. The data contained in a DENM is related to events that may have an impact on road safety or traffic flow. Because of the nature of these events, DENM may be connected with the event rather than the producing ITS station. Unlike CAMs, a DENM can be forwarded by a third-party entity. The DENM format can be seen in Fig. 4.3;

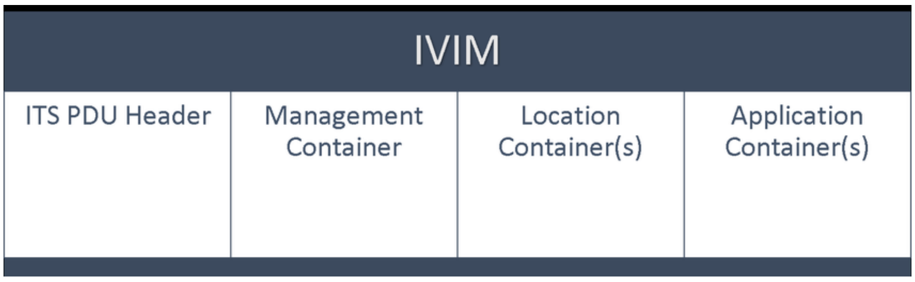


Figure 4.4: Format of an IVIM message.

- the IVI service, as specified in [78], is an instance of the infrastructure services that control the generation, transmission, and receiving of IVIM messages. An IVIM provides support for both obligatory and advisory road signs, such as contextual speed limits and roadwork warnings. Additionally, IVIM gives information on real road signs, such as static or variable road signs, virtual traffic signs or roadworks. Despite its purpose, which is similar to that of DENM, the IVIM message is specifically dedicated to non-critical V2I communications. As for the DEN, the IVI service is also triggered by application levels in an asynchronous way. The IVIM format is shown in Fig. 4.4.

4.4 The AMQP Broker server

The Advanced Message Queuing Protocol (AMQP) [60] is an open standard application protocol for message management in both a point-to-point and publish/-subscribe manner. AMQP provides message orientation, routing, message queuing, reliability, point-to-point security, and a variety of other characteristics.
AMQP is one of the protocols that is well-suited for managing a message Broker (other similar protocols are MQTT, STOMP, XMPP, etc.). A message Broker is a network node that accepts messages from one (or more) sender(s), performs some operation on the messages (e.g., processing, queuing, aggregation, decomposition, validation, etc.) and forwards the messages to one (or more) receiver(s).

One of the most fundamental characteristics of an AMQP Broker is that it operates in a publish/subscribe fashion, with senders and receivers acting as message producers and consumers, respectively. A producer is a node that creates content of interest and publishes it on the Broker server. Every node in the network has the option of subscribing to a specific set of contents made available on the Broker server. As soon as new content is made available for that content set, the Broker notifies the active consumers by sending the information requested to them in real time. *Queues* and *Topics* are the two main types of message structures that are managed by the broker server.

4.4.1 Addresses, queues and topics

Selecting an address is necessary in order to establish an online connection between producers and consumers. As with all other message Broker protocols, AMQP has the feature of a **address hierarchy**, which is described below. Consider a multi-level address such as the following:

$$brokerIP : 5672/This.is.an.example.with.AMQP.Broker$$

where *brokerIP* refers to the IP address of the Broker server (which is often a public IP address), 5672 corresponds to the standard port devoted to AMQP, and the entire string *This.is.an.example.with.AMQP.Broker* refers to the address of the AMQP queue. Within the previously defined address, it is possible to distinguish between several address levels, which are separated by the 'dot' character. The separated-values structure presented here reflects an address hierarchy in which the values that appear first are more important than the values that appear after them. This hierarchical structure is extremely important for the definition of a geographical area. By associating a digit code index with a given area, it is possible to select a subset of that area by adding less relevant digits. This hierarchical representation of a map environment is called *Quadkeys* and its principle is illustrated in Fig. 4.5. The separated-values structure of AMQP addresses fully complements the Quadkeys format, resulting in an optimum solution for the identification of geographical areas.
Aside from that, the AMQP standard allows for the use of wildcard characters in the address specification. Consider the following scenario, where there are two producers and one customer, each having the following addresses:

$$producer1 : \quad brokerIP : 5672/AMQP.address.producer.example.1.2.3$$

$$producer2 : \quad brokerIP : 5672/AMQP.address.producer.example.4.5.6$$

$$consumer : \quad brokerIP : 5672/AMQP.address.producer.example. * . * .*$$

We can see from the address definition that the two producers will publish their messages in distinct queues. However, the consumer will be able to receive and consume messages from both producers. This is due to the fact that the structure of the consumer's address matches both the producers' addresses due to the usage of the wildcard '*' in the address definition on the consumer side.

Once an address has been specified and a link has been formed between the producer and consumer entities, the data structure entitled to the message management must be chosen between queues and topics. From the point of view of the addresses, there is just a small deviation between the definition of queues and the definition of topics. In order to select a topic data structure, the string *topic:/* has

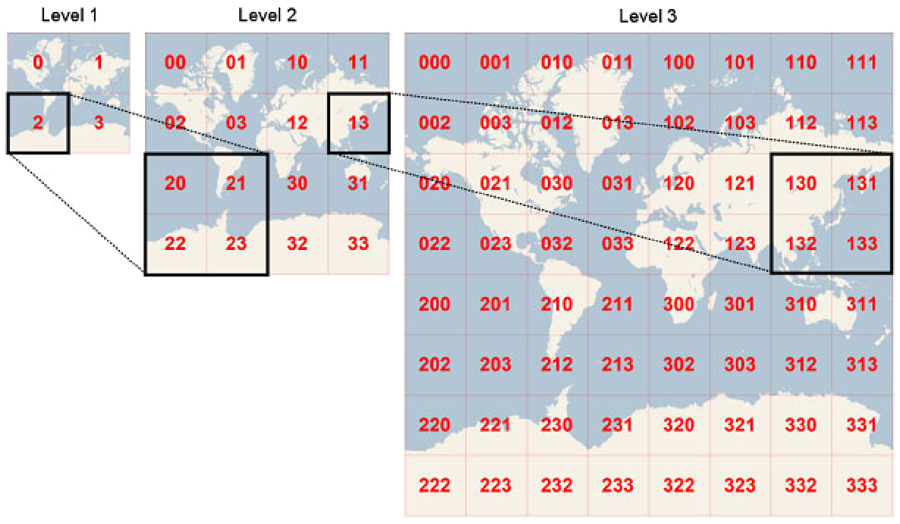


Figure 4.5: Quadkeys hierarchical structure for a geographical area definition.

to be inserted before the AMQP address name. If no data structure is specified, the AMQP protocol assumes a queue data structure, as detailed in the following:

$$queue: \quad brokerIP:5672/This.is.an.example.with.AMQP.Broker$$

$$topic: \quad brokerIP:5672/topic://This.is.an.example.with.AMQP.Broker$$

Despite the fact that they share a similar address definition, queues and topics have significantly distinct characteristics. A first difference is regarding **message availability**. If we publish a flow of messages on an AMQP queue where no consumer has an active subscription, then the messages are marked as "pending messages" for a given time period. As soon as a new consumer subscribes to the queue, all pending messages will be sent to the new consumer, who is now actively listening. On the other hand, topics do not guarantee message availability: if a producer publishes a flow of messages on a topic where there is no active consumer, the messages are no longer available for consumption.
Because of the message availability feature, we may use queues or topics data structures that are appropriate for the time-dependent attribute of the application we are developing. If we are interested in an application that informs drivers of a long-lasting event (e.g., adverse weather or poor road conditions), it may be beneficial to also communicate this information to the new customers who have just subscribed to the queue. The opposite is true when developing a vehicular application that warns drivers of an imminent hazard (e.g., an imminent danger ahead due to an animal crossing the road), since our information will be related to a relatively small time window and, as a consequence, topics will be the most appropriate choice for the data structure.

A second difference between queues and topics is regarding **load balancing**. When a flow of 200 messages is published by a producer on a queue with two active subscribers, the AMQP protocol generates a load balancing mechanism between the two active subscriptions on the queue. Thus, each consumer receives 100 messages, with a single message being received by either the first or second consumer. As a consequence, each consumer receives 100 messages. Topics, on the other hand, do not provide load balancing functionality. If a flow of 200 messages is published on a topic with two active consumers, then the whole flow of 200 messages will be delivered to each of the two active consumers.
When deciding whether to utilize queues or topics for our application, the load balancing property is a crucial factor to consider. If the system we are designing deals with low-priority information (e.g., adverse weather or poor asphalt quality), then a continuous flow of messages ensures that at least one of the messages is received even in a situation with several active customers. On the contrary, if we intend to develop a security application that sends high-priority notification messages (e.g., an imminent hazard ahead due to an animal crossing the road), the time it takes for the message to be received is extremely sensitive information. In such a circumstance, we must ensure that all active consumers receive a rapid response, which means that the load balancing attribute should not be utilized and that the topic data structure should be used instead.

In both the applications we developed, which are explained in the following sections, we provide contextual traffic information in real time or notification messages for an immediate road hazard situation. As a result, we chose to use topics as our data structure for the administration of the AMQP Broker server.

4.4.2 ActiveMQ and Qpid Proton

The ActiveMQ online message broker tool [82] and the Qpid Proton message library [83] were both utilized to manage our AMQP Broker server. Apache ActiveMQ is the most widely used open source multi-protocol message broker that is built on Java. It supports industry-standard protocols, allowing users to make use of a wide range of client options across a broad range of languages and software platforms. This tool enables connections to AMQP consumers written in JavaScript, C, C++, Python,.Net, and other programming languages.

Qpid Proton is a messaging library that is both high-performance and lightweight. It may be used in a broad variety of messaging applications, such as brokers, client libraries, routers, bridges, proxies, and other similar technologies. It is simple to interface with the AMQP ecosystem using Proton, regardless of the platform, environment, or programming language used. The purpose of Proton is to give ubiquitous access to a global-scale inter-operable message bus, which includes implementations of the AMQP protocol in Java and C, as well as multiple language bindings around the latter, such as for Python and C++. These implementations

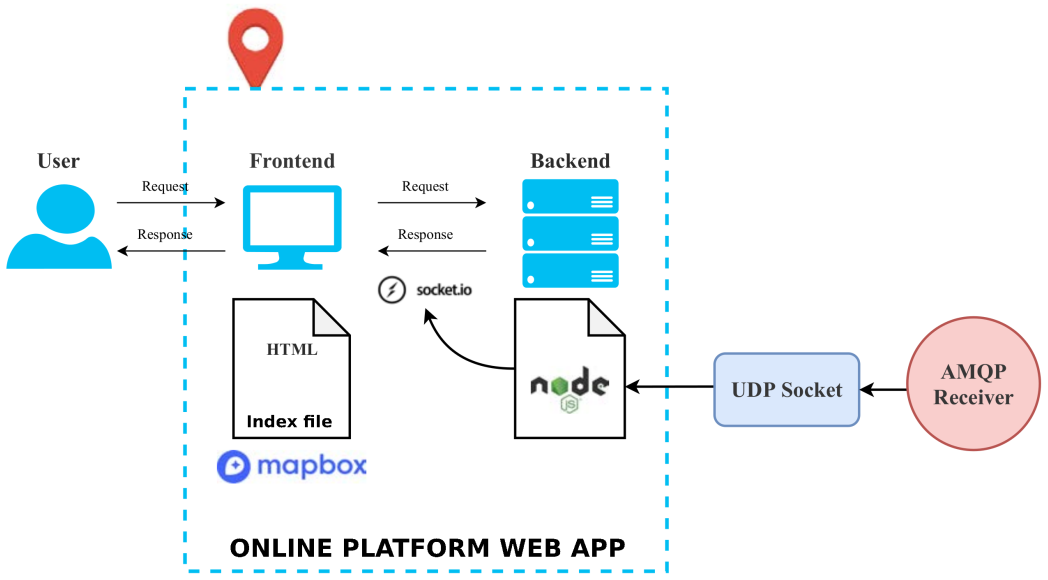


Figure 4.6: Online platform web app architecture.

serve as the basis for offering protocol access to a wide range of environments in a number of settings.

4.5 Development of an online platform service

Both the projects presented in the following sections are focused on automotive applications that rely on V2N communication technology. Those kinds of applications may often benefit from the use of a visualization tool to examine the results of message flows on the server side. In order to do this, an online platform specialized in the depiction of events within a map context was created. In this section, we will look at the methods that might be used to create such an object. The representation of the full architecture described in the following is reported in Fig. 4.6.

The first step is to organize all the setup requirements on the proper hardware. For this purpose, a server machine has to be exposed with a public IP address. In such a way, it will be reachable by any user and from any location. In addition to this, ports 80 for HTTP service and 443 for HTTPS have to be enabled for both incoming and outgoing traffic. In order to ensure a secure connection through HTTPS service, a public key certificate has to be issued by a trusted Certification Authority (CA) and associated with a public domain name. For the scope of the projects described in the following, a public key certificate issued by Politecnico di Torino root CA was associated with the domain *serverfull.polito.it* which corresponds to the public IP address 130.192.30.241 of our server machine. In such a

Figure 4.7: Mapbox geographical tile.

way, we are able to reproduce a so-called backend node that is able to expose online services.

The backend node is entitled to host all the files for the exposure of the online platform service. The first necessary file to compile is the Node.js file [84]. Node.js is an open-source, cross-platform run-time environment in JavaScript for managing a backend web browser. Thanks to that, we are able to set up the HTTP and HTTPS services on dedicated ports. The Node.js file is where you upload the public key certificate files to enable a secure connection. It also represents the very first arrival point of any data on the server. Each message transiting through the Node.js file has to be managed by using sockets. In particular, the messages arriving from the Broker server are delivered to the Node.js file via UDP socket and are then forwarded to the index file.

The index file is usually made up of an HTML or JavaScript index file and it is where the actual online application is built. It constitutes the part of our online application which will be provided to the client, i.e., the frontend. For this purpose, a Mapbox [85] online map tile was developed using our HTML index file. Mapbox is a provider of custom online maps for websites and applications dealing with open-source mapping from OpenStreetMap [24]. Using a Mapbox tile, we are able to create a map environment with geo-localized data, as reported in Fig. 4.7.

Upon a web request made by a user client, an instance of the index file is uploaded on the frontend side. For this reason, every data that has to be transferred from the Node.js file to the index file has to pass through a dedicated web socket. The web socket library we used for this purpose is the socket.IO library [86]. The socket.IO JavaScript library is a dedicated tool for managing web sockets in real-time web applications. It enables real-time, bi-directional communication between

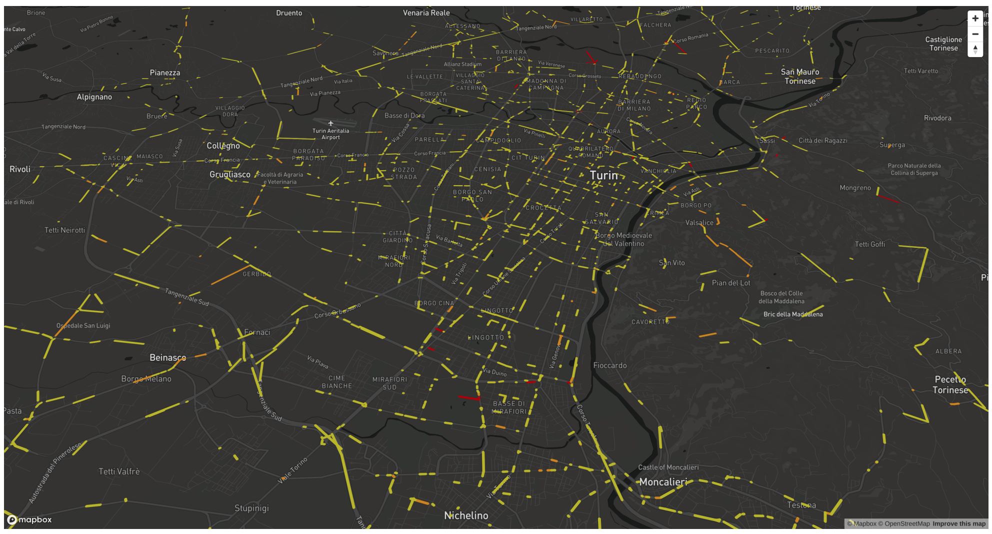


Figure 4.8: Turin map environment with hour-based traffic information from TuST.

web clients and servers.
By developing such an architecture, we are able to create an online platform able to receive and display data in real time from the AMQP Broker server. An example is given in Fig. 4.8, where hour-based traffic congestion indications from the TuST simulator are displayed on a Turin map environment.

4.6 The 5GAA demonstrator

The 5G Automotive Association (5GAA) [87], founded in September 2016, is a global, cross-industry organization of companies from the automotive, technology, and Information and Communications Technology (ICT) industries, who are working together to develop end-to-end solutions for future mobility and transportation services. The 5GAA promotes the idea that 5G will be the ultimate platform for enabling C-ITS and the supply of V2X through live events, on-field test demonstrations, studies, white papers, etc. On November 14th, 2019, the 5G Automotive Association (5GAA) held a live demo event in Turin where they presented use cases that were ready to be deployed in the city's streets. On this occasion, representatives from the Politecnico di Torino, TIM, FCA, Links Foundation, and the City of Turin with 5T presented the Urban Geo-referenced Alert (UGA) use case. With the UGA use case, a traffic efficiency and management application type was developed. Its goal is to recreate a navigation system that provides geo-located mobility and traffic information in real time that is obtained directly from the info-mobility agency. In Fig. 4.9, the complete architecture of the work under consideration is depicted and it is further discussed in the following paragraph.

For the development of our UGA application, an FCA vehicle was equipped with

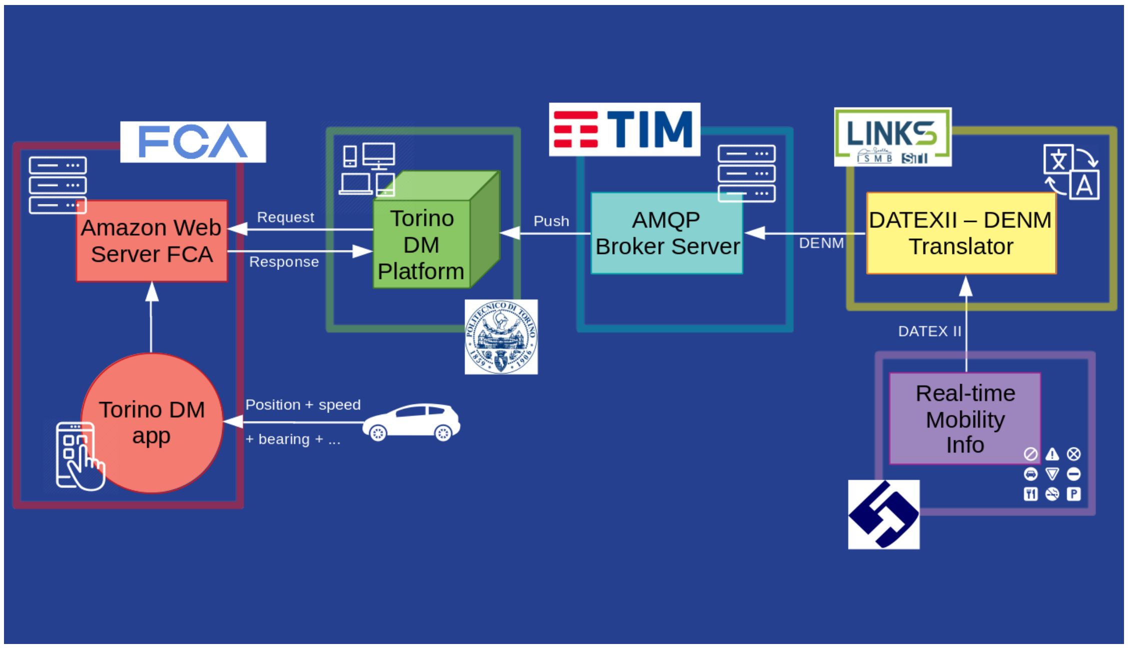
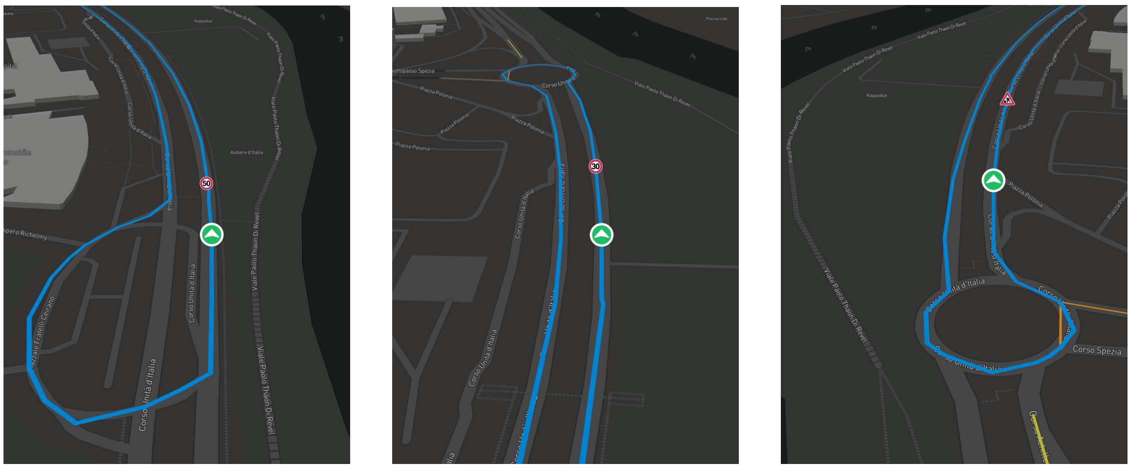


Figure 4.9: 5GAA demonstrator architecture.

an OBU that was running the Torino Digital Mobility application. The Torino Digital Mobility app permitted the car to transmit its location, speed, bearing angle, and any other motion information to an FCA Amazon web server, which was accessible through the internet via the Uu vehicular interface. Due to the fact that we are dealing with a non-critical application, the update rate of the vehicular motion information has been set at 1 Hz.

With a periodic request/response mechanism, we were able to get the live motion of the vehicle and replicate it on a specialized web platform service (similar to the one described in the previous section), which we named the Torino Digital Mobility platform. The Torino Digital Mobility platform was accessible at [63] for the duration of the demonstration event; however, after that, FCA no longer authorized access to the Amazon web server. Nevertheless, an offline reproduction has been constructed and is presently available at [64], where a simulated instance of a car is following the path planned for the demo event on Corso Unità d'Italia. In exactly the same way, on the demo day, we were able to track the real-time movement of the car on our Torino Digital Mobility platform, which operated as a sort of online navigation system.

On top of this online model, we also included a second level of information in our use case, which consisted of real-time traffic data. A roadwork event and two speed reduction alerts were replicated on the planned path by the 5T info-mobility agency. When the infrastructure and the info-mobility companies generate traffic statistics and data, they utilize a European standard electronic language called



(a) (b) (c)

Figure 4.10: Geo-referenced mobility information displayed in real-time on the Torino Digital Mobility online platform.

DATEX II [88]. Every piece of information associated with the traffic event is contained within a DATEX II message in XML format. Because the communication between ITS stations does not use the same standard, the Links Foundation developed a message translator. To be more specific, the created roadwork event was transformed into a DENM message, and the two speed limit signal alerts were converted into IVIM messages.

Eventually, TIM was entitled to manage the AMQP Broker. The DENMs and IVIMs were sent by Links Foundation over the AMQP Broker server, and they were successfully received by our Torino Digital Mobility platform. Upon the reception of DENMs (for the roadwork event) and IVIMs (for the speed limit alerts), we performed a decoding of messages from ASN.1 to plain text and extracted the relevant information from them. We were able to display the events on our map environment, as illustrated in Fig. 4.10. In particular, in Fig. 4.10a , a 50 km/h speed restriction notification was displayed along the demo planned path, which is seen in blue in the pictures. As seen in Fig. 4.10b, a 30Km/h speed limit sign was placed at the end of the same street edge before the roundabout. Upon returning to the starting location, a roadwork signal was signaled to the driver when the DENM was received, and it was reported on the web platform, as seen in Fig. 4.10c. Additionally, DENM and IVIM messages were pushed from the AMQP Broker server directly to the test car, which was listening on a dedicated AMQP topic. FCA created an integrated Human-Machine Interface (HMI) for this purpose, which was capable of displaying the exact same traffic signals indicated above on the OBU of the vehicle as well.

4.7 The Rainbow project

The Internet of Things (IoT), which is powered by Cloud computing, has established itself as the dominant technology paradigm for monitoring, understanding, and interacting with both the physical and digital worlds. Since cloud processing capacity outperforms the resource limitations of mobile and IoT devices, IoT services naturally offload demanding analytic workloads to the cloud. However, as the volume of data created at the network's logical edges grows, and bandwidth does not scale at the same rate as computing capacity, data transmission is becoming a bottleneck, limiting the cloud paradigm [89]. This effectively translates to additional data and traffic on already overloaded Internet arteries. Thus, it seems inevitable that data must be processed at the networks' edge to achieve shorter response times and less network pressure [90].

Fog computing is a network computing technique that allows computation at network extremes, such as on downstream data for cloud services and upstream data for IoT services [91]. Fog computing is based on the idea that computation should take place close to the data source, with the "edge" referring to any compute and network resources located along the path between the data and the cloud. Raw signals are processed into meaningful information in proximity rather than being transmitted back and forth to the cloud for "centralized" processing as IoT hardware evolves. Although the terms fog and edge computing are sometimes used interchangeably, there is a distinction between them in terms of computing location [92]. Edge computing enables analysis where data originates, while fog computing extends the cloud's analytic capabilities to the local network to close the "cloud-to-thing continuum".

Although fog computing puts computation closer to delay-sensitive services, the cloud model continues to face issues as the amount of created data increases. Now, these massive amounts of data must not only be handled quickly, but also on potentially "weaker" technology, with possible fog nodes including vehicles, WiFi access points, sensors, cameras, and other devices [93]. As a result, existing technology assumes that fog nodes are stationary and that only IoT devices are mobile. The impact of fog node mobility on IoT and cloud service QoS, cost, and energy usage receives little attention. However, considering that fog nodes can be mobile and that rapid changes can occur at anytime in dynamic networks (e.g., connectivity failure, bandwidth fluctuation), the orchestration of fog services becomes more challenging. Open research challenges include service discovery, resource provisioning, coping with fog and resource heterogeneity, life-cycle management, and task offloading, which all require a rethinking of current decentralized computing and network communication algorithms, as well as the introduction of new efficient and agile orchestration models.

The goal of the Rainbow project is to provide a fog/edge orchestration framework capable of preserving the appropriate QoS for applications running at the network's

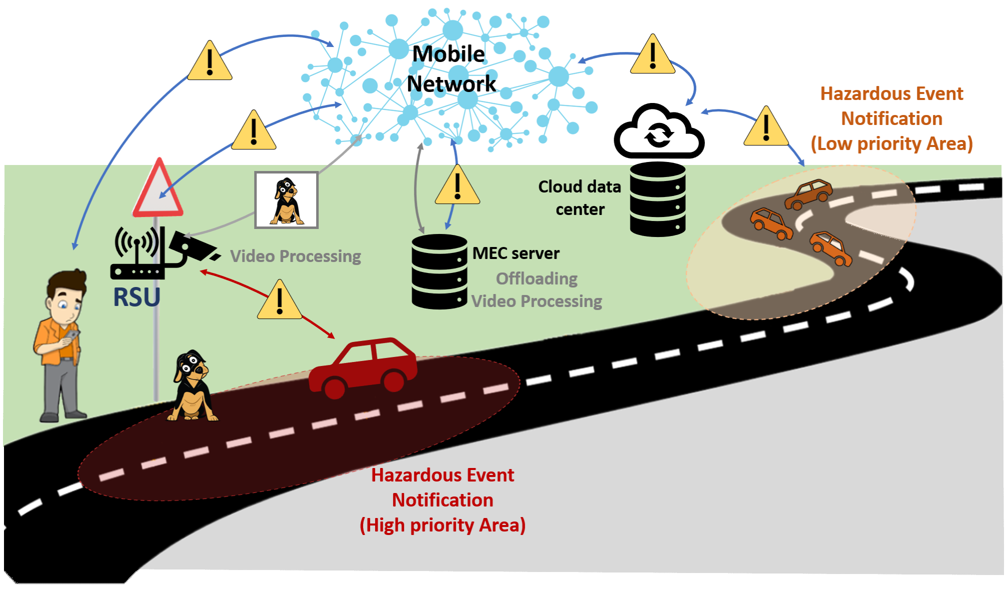

Mobile Network
Hazardous Event Notification (Low priority Area)
Cloud data center
Video Processing
RSU
MEC server
Offloading Video Processing
Hazardous Event Notification (High priority Area)

Figure 4.11: Urban Mobility Demonstrator operative representation.

edge (or fog) level by detecting and optimizing resources in real time while satisfying application-related limitations. The Rainbow platform evaluation is carried out through three use cases that aim to affirm the framework capabilities at the edge/fog level. For this occasion, Politecnico di Torino, Links Foundation and Stellantis Group with the CRF lab (i.e., the FCA Research Center in Turin) were asked to develop a vehicular use case for the Rainbow platform evaluation, called Urban Mobility Demonstrator (UMD).

4.7.1 The UMD use case description

A general description of how the Urban Mobility Demonstrator (UMD) use case operates, given in this section, is reported in Fig. 4.11. The goal of the UMD use case is to show how the Rainbow system will contribute to the development of a real-time geo-referenced notification system for vehicles traveling in urban areas in the event of a hazardous situation. The demonstrator is made up of a road section with an on-site installation of a Road Side Unit (RSU) that is directly linked to several IP cameras. A dedicated server (also referred to as Mobile Edge Computing - MEC - server) that acts as an edge node and connects with the RSU via the Internet can also access the video stream. The Automatic Hazardous Events Detection (AHED) service uses a combination of computer vision and artificial intelligence (AI) algorithms to detect the presence of animals on the road, which may be a dangerous situation for drivers. Based on this, the AHED service can run on either the RSU or the MEC node, with advantages and disadvantages in terms of latency

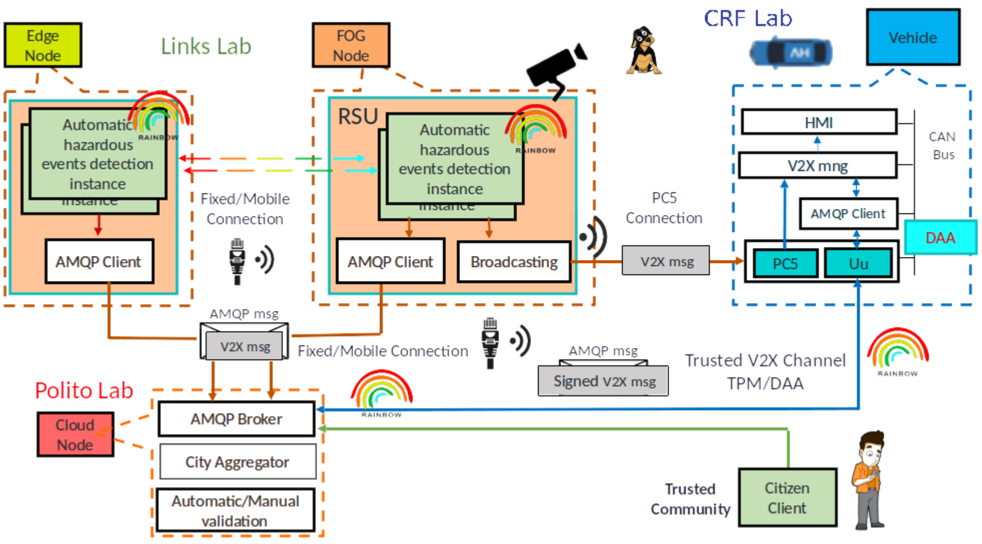


Figure 4.12: Urban Mobility Demonstrator architecture.

and accuracy constraints for each configuration. Indeed, the RSU benefits from the direct connection with IP cameras, which yields a high-resolution video stream at a high frame rate. The higher the resolution, the more accurate the results will be. At the same time, the AHED system requires high computational power to run and the RSU, to achieve this, must increase its power consumption, which may cause the overheating of the system and malfunctions. The MEC node, on the other hand, is robust to overheating and hence capable of running sophisticated algorithms. Unfortunately, because of channel latency and bandwidth constraints, the MEC node is forced to operate with a low-quality video feed at a lower frame rate, resulting in less accurate results. The Rainbow platform's objective is to orchestrate the ideal utilization setup between the MEC and the RSU in order to increase accuracy while lowering power and bandwidth consumption.

When an animal on the road is recognized by the AHED service, the detecting entity generates a hazardous event notification by producing a set of DENM vehicular messages. DENMs generated in this manner are split into two streams: one is broadcast through the PC5 interface for direct communication, while the other is transmitted via the network via the Uu interface to a cloud data center managing an AMQP Broker server. In such a way, we can distinguish between a high-priority area close to the detected animal, where drivers are notified on the fly directly by the RSU, and a low-priority area, where drivers who are far away but approaching the event can subscribe to the AMQP topic on the Broker server and receive notification of the hazardous situation ahead. Finally, for future project advancements, we intend to simulate a so-called Citizen App, in which pedestrians near a traffic

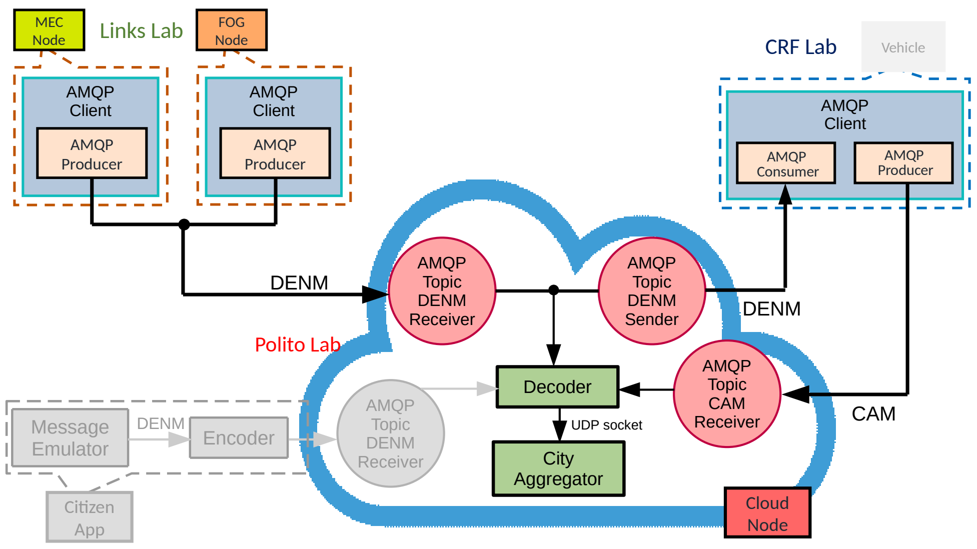


Figure 4.13: Cloud node architecture.

hazard may report the dangerous situation and, as a result, DENMs transmission across the network will be triggered.

4.7.2 The UMD use case architecture

The Polito lab, the Links lab, and the CRF lab collaborated extensively to create such a difficult use case. Fig. 4.12 depicts the completed architecture with all of its components. The Links lab created the Automatic Hazardous Events Detection (AHED) service for the visual detection of a dog, as seen in the image. The RSU node, the edge node, the DENMs sent from them, and the Rainbow orchestration of the AHED service between the edge and RSU nodes are all managed by the Links lab.

Due to COVID-19 constraints, it was not possible to have an actual car on the road for the DENMs reception. As a result, CRF lab created a simulated car using a GPS antenna, a V2X board, an On Board Diagnostic (OBD) Bluetooth dongle, a vehicular Controller Area Network (CAN) bus emulator, an AMQP client emulation on a Raspberry Pi and a tablet running the HMI.

We were in charge of operating the cloud node in this system design as Polito lab. The major objective of this component, as shown in Fig. 4.13 where a complete architecture schema of the cloud node is presented, is to control message flows between the Links lab (MEC node and fog node) and the CRF lab (simulated vehicle). An AMQP Broker server was set up for this purpose, with three active

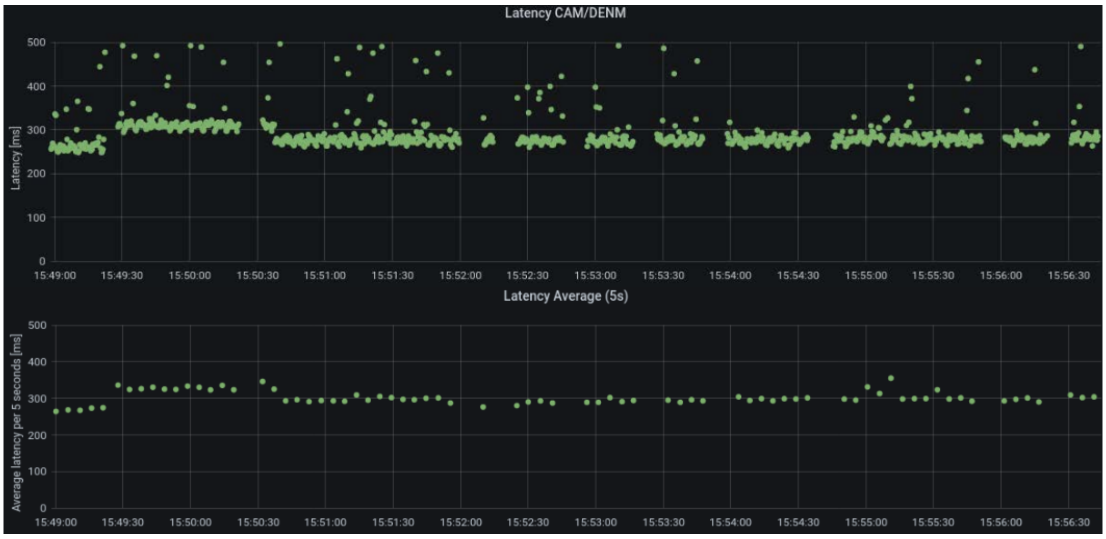


Figure 4.14: Latency measurements on a Grafana dashboard.

topics for handling two message flows. The simulated vehicle generated a flow of CAM messages providing basic motion information for transmitting periodic car location updates to the cloud node. A second flow of DENM notifications is created when the MEC node or the RSU detects a road hazard. DENMs provided in this manner are accepted by the AMQP Broker server and passed to a second topic, where the simulated vehicle listens for the arrival of the alert notification. A third flow of DENMs generated from the Citizen App is deemed suitable for future work.

Messages are delivered to the Decoder module after being received by the AMQP Broker. The goal of the Decoder component is to convert ASN.1 messages to plain text and extract relevant data that can be used for analysis. The *referenceTime* field is critical since it contains the message creation timestamp. An end-to-end latency measurement (i.e., the time it takes for the message to be created, encoded, sent, received, and decoded) can be computed by comparing this field to the actual timestamp on the receiver side. It's important to note that the end-to-end latency is vital information for this, as well as any other vehicle safety applications. As a consequence, as shown in Fig. 4.14, a specific Grafana [94] dashboard was created for the real-time monitoring of punctual values and averages of end-to-end latency.

Eventually, the City Aggregator, an online platform for event visualization on a map environment, was built and is now available online at [65]. When a message is sent from the simulated car instance to the Broker server, the message's most relevant fields are extracted and delivered to the City Aggregator. As demonstrated in Fig. 4.15, it is possible to observe the live movements of cars in a map visualization environment using periodic CAM received every 100 milliseconds. When a DENM message is received and decoded, the relevant information about the hazard's location and the region of interest is retrieved and delivered to the City Aggregator

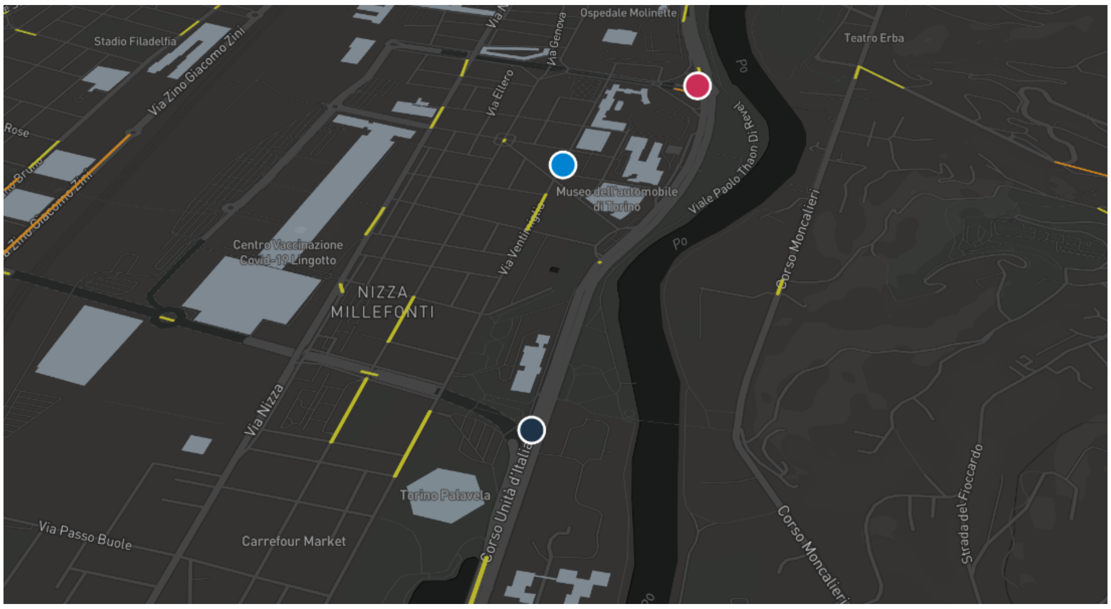


Figure 4.15: Vehicle instances representation on the City Aggregator online platform.

web platform using the same process. DENMs cause the high priority region (the red circle around the dog icon) and the low priority area (the ellipsoid on the road segment approaching the hazard) to be visualized, as illustrated in Fig. 4.16.

4.8 Conclusions

During the last few years, the development of new technologies (such as 5G mobile networks, ADAS systems, and V2X equipment), as well as the efforts of standardization organizations to define appropriate messages and protocols, have made incredible steps forward in vehicular communication research and technology development overall. Technical advancements must, however, contend with the limited penetration rate of V2X equipment on real streets, which prevents a number of cooperative vehicular applications from being successfully implemented in real-life situations. A possible solution relies on V2N communication-based applications, which can guarantee the service even for scenarios with a single vehicle equipped in the whole use case environment.

In this chapter, we examined the most important aspects to consider while developing a V2N-based application. The characteristics of an AMQP Broker server, in particular, were detailed in depth, with all of the aspects of its hierarchical structure for the addresses definition and the properties of the queues and topics data structures being included in the description. Following that, we provided a summary of the protocols and messages that established the standardized entities for

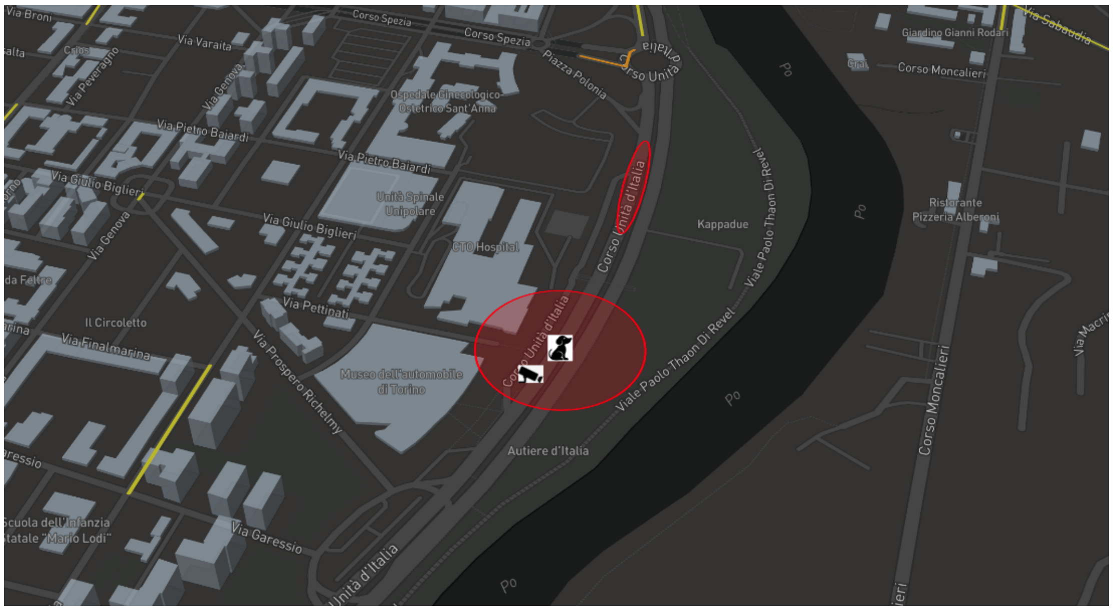


Figure 4.16: Representation on the City Aggregator online platform of high and low priority areas associated to the road hazard.

vehicular communication. We concentrated on the differences between European and American standards in terms of software rules and messaging, as well as the distinctions between direct communication and cellular communication. Finally, a section is devoted to the description of how to build an online platform service, which brings this methodological part to a close.

The two real-world test case scenarios that we developed in the context of V2N communication-based applications were presented at the end of the chapter. We built an Urban Geo-referenced Alert (UGA) use case for the 5GAA demonstrator, which is a navigation system that receives contextual Geo-referenced information in real time from the info-mobility agency, which sends it across the network via AMQP Broker. In the context of the Rainbow project, an Urban Mobility Demonstrator (UMD) was created to allow for the testing of the Rainbow orchestration platform. The UMD use case focuses on a real-time road hazard notification message exploiting V2N communication to inform drivers of the danger ahead. For both the projects presented, we gave a comprehensive description of the methodology and we presented the results and the events visualization on the online platforms developed.

Chapter 5

Monitoring Public Transportation Occupancy

Along with vehicular applications, people-flow analysis is another field of study that has seen a significant increase in interest in recent years. In the context of a city, monitoring the movements of people and passengers on public transportation may be beneficial in the development of services that are tailored to meet the needs of the community. Some examples of this may include better line shaping provided by a public transportation authority or dedicated applications for data exchange in crowded areas, among others. In contrast to vehicular data, obtaining information about pedestrians and commuters on public transit is particularly difficult, both from a methodological and a privacy perspective.
In this chapter, we will discuss the development of an Automatic Passenger Counting System (APCS), which was utilized to provide a real-time monitoring service for the GTT (Gruppo Torinese Trasporti - i.e., Turin Transport Group) local transportation agency.

5.1 Research motivation

The Italian National Institute of Statistics (ISTAT) presented a study of daily commuters and their modes of transportation in November 2018 [95]. The analysis was based on data from the previous year's national census. The findings of this survey revealed that more than 40% of the Italian population regularly chooses not to utilize private transportation for their daily commutes. The 19% of the overall Italian population chooses to travel by foot or bicycle, while the 23% relies on public transportation such as buses, trams, trains, school buses, or metro.

Over all, this constitutes a significant amount of national commuters who rely on non-private modes of transportation for their daily routines. This segment of the urban population is dependent on appropriate infrastructure and services offered by local transportation authorities in order to go to and from job or education on a regular basis. In order to achieve this goal, it is evident that simple data obtained through a census would not be enough. In order to maintain a high quality of service, there is a continual requirement for accurate monitoring of the number of people using public transportation. The local transportation authority would be able to adequately scale the service if they had real-time data on bus passengers collected at each stop. It would be able to determine the appropriate number of public transportation vehicles for each tram and bus line in order to ensure a minimum frequency of service.

Additionally, following the worldwide lockdown enforced by governments in order to contain the spread of the COVID-19 pandemic, capacity limitations in public locations were imposed in order to limit the number of people who gathered. In Italy, a capacity threshold was established for the maximum number of people that might be carried on public transportation vehicles. A pedestrian flow monitoring system is even more critical in light of these restrictions and limitations. On the other hand, those types of systems are incredibly complicated to develop in a trustworthy manner, and they must constantly ensure the privacy of the data of every person engaged.

5.2 Main contributions

This project was initiated by an intention on the part of Turin's municipal transportation agency, GTT (Gruppo Torinese Trasporti - i.e., Turin Transport Group), to install a passenger monitoring system on board their public buses. The goal of this technology is to calculate the number of people that are currently on board each bus stop along the route.

A number of attempts by GTT to calculate this statistic have been made in the past, all of which have been found to be inaccurate. Initially, it was suggested that the driver should be able to count the number of people boarding at each stop and sign them using a specialized tablet that would be located near the steering panel. Obviously, this approach was abandoned rather quickly, since drivers complained that it was distracting them from their jobs. Then, taking advantage of the fact that all bus tickets are in a contact-less format, GTT required that every person entering a bus or tram validate the ticket while getting on the vehicle. As a result of the sound created by the dedicated validation machines on board, this proposal was given the name of BIP technique. GTT, on the other hand, offers many types of tickets for the final user, and tickets valid for a week, a month, or an entire year do not need to be validated on each journey. The BIP technique did not produce the expected result since the validation on boarding for each passenger was more of a request to the user than an actual requirement. As a result, it frequently understated the actual number of passengers on the bus.

As the government imposed a capacity constraint on transportation systems after the general lockdown, GTT started to invest in more accurate hardware strategies for Automatic Passenger Counting Systems (APCSs). Several APCSs were evaluated by GTT over the course of several months, and the limitations of each were documented. Many systems, such as infrared lasers, pressure panels, or cameras installed on the top of the bus doors, are used to count the number of individuals who pass through the doors. All of these approaches, however, cannot guarantee a high degree of accuracy: they are always imprecise when two or more people board at the same moment from the same entrance, or when they stand directly in front of the door due to the presence of other passengers on board. The results of various experiments led GTT to the conclusion that the only way to achieve high accuracy in APCSs was to combine data from multiple sources.

The main objective of this project is to install three Automatic Passenger Counting Systems (APCSs) on a public GTT bus and to collect passenger monitoring data from all of them in order to develop a reliable counting system. The first APCS, known as AESYS, was installed by GTT and makes use of stereoscopic cameras mounted on the tops of bus doors. When passengers stand exactly under the cameras without moving, the system suffers from low accuracy, as do all systems of this sort. When this occurs, the system is unable to determine whether the passenger is entering or exiting the bus. Despite this, GTT says that AESYS is capable of achieving an accuracy of more than 90%. A second APCS was installed on the bus suspensions by a group of students from the Politecnico di Torino's department of transport engineering. It keeps track of the total weight of the bus and uses this information to make a prediction about how many people are on board. The average accuracy declared for this APCS was in the range of 80%. The third APCS is the one that we developed, and it is the only one that we will discuss in this chapter. The technique we used attempts to estimate the number of bus passengers on board by measuring the number of WiFi signals received. Theoretically, assuming that everyone has a smartphone, we should be able to maintain a high level of accuracy in monitoring. It is possible that the presence of individuals with more than one WiFi interface active (e.g., from two smartphones or a smartphone and a tablet) as well as the existence of devices with the WiFi interface switched off will result in a bias in the final results. Despite the fact that our results are still not at the same level of accuracy as those obtained by the other two APCSs, we can assert that our system is far more independent of location. We can certainly imagine our system being used in many public spaces (such as workplaces, libraries, public squares, metro stations, and so on), where the use of the other two APCSs described above would be impractical or impossible.

The tuning phase of any APCS is essential, during which results are regularly compared with a so-called ground truth measurement. The ground truth metric for this project was achieved by manually counting the number of people on board between the two ends of the line. Multiple manual counting sessions were scheduled in order to fine-tune all of the parameters of our system to their optimal settings.

However, because of the immediate lockdown and limitations for the COVID-19 pandemic, we were unable to access all of the data we had planned to process. As a result, the research described in this chapter is still in the early stages of development, and the conclusions reported can be considered preliminary in nature for this specific study. Considering this, the results we will report in the following demonstrate a degree of accuracy that is acceptable given the current stage of the project. A publication on this topic has not yet been created due to the fact that this work is not considered to be completed.

The next section provides an overview of MAC addresses and the WiFi association procedure. Following that is a methodology section in which we detail all of the hardware and software solutions used in this investigation. Due to the importance of privacy problems in this setting, we have included a section dedicated to the discussion of privacy issues after the methodology. The acquired data is then presented, along with conclusions.

5.3 Scanning of the WiFi media

The approach we utilized to estimate the number of passengers on board makes use of WiFi signal reception and processing. In particular, we scanned WiFi media and attempted to capture WiFi packets transmitted by smartphones. When a smartphone's WiFi interface is turned on, it instantly sends out several broadcast messages to discover any nearby Access Points (APs) or smartphones. We check for the MAC (Media Access Control) address information on collected packets to match a captured WiFi signal with a smartphone (and hence with a person on the bus). In the subsections that follow, we will look at the properties and functions of a MAC address, the phases of a WiFi association and their related messages. Finally the MAC randomization mechanism is described as well as the main software tool dedicated to packet sniffing over the wireless channel.

5.3.1 The MAC address

Network interfaces established by the IEEE 802 standard [96], such as Ethernet, WiFi, and Bluetooth, all have a Media Access Control (MAC) address that serves as a unique identification number. While Internet Protocol (IP) addresses are associated with the network layer (i.e., the third layer) of the Open Systems Interconnection (OSI) stack defined by the International Organization for Standardization (ISO), MAC addresses are associated with the second layer, which is dedicated to Logical Link Control and Media Access Control, respectively, LLC and MAC. In contrast to the fact that the same network card might have many IP addresses associated with it in separate connections, the MAC address is unique for the same card and indicates its identifying code. Taking this into consideration, any device capable of connecting to the internet will have a MAC address for each connection interface. For example, a laptop will have a MAC address for its WiFi card, a second MAC address for its Ethernet interface, which is dedicated to cabled connections, and a third MAC address associated with its Bluetooth card.

The MAC address is a 48-bit code that is generally subdivided into 6-octets of bits. The most common representation of a MAC address is an expression with semicolons between each octet and with each octet represented as a pair of hexadecimal values. The Organizationally Unique Identifier (OUI) is expressed by the first three octets of a MAC address, while the Network Interface Controller (NIC) is represented by the second three octets of the address. As an example, the MAC address associated with the WiFi interface of my laptop is

$$68 : 54 : 5a : eb : f7 : 26$$

where $68 : 54 : 5a$ is the OUI and $eb : f7 : 26$ represents the NIC. Furthermore, the initial octet contains two less-significant bits that have a particular function. The least significant bit of this octet distinguishes between addresses that are unicast and those that are multicast or broadcast. If this bit is set to zero, the corresponding hexadecimal expression is an even value, and the MAC address under consideration indicates a unicast address, which is an address that is used for one-to-one communications only. A unicast/multicast bit that is set to one will result in an odd value for the last hexadecimal of the first octet, indicating that the MAC address is being used for one-to-many communications. In this chapter, we will focus in particular on the MAC addresses of the WiFi network card. MAC addresses of WiFi cards are typically expressed as unicast addresses since WiFi is, by its nature, a broadcast medium.

The second-least-significant bit of the first octet of the address is used to distinguish between addresses that are universally administered and addresses that are locally administered. A universal/local bit set to zero will identify a universal address and will result in a ‘0’, ‘1’, ‘4’, ‘5’, ‘8’, ‘9’, ‘c’ or ‘d’ value for the last hexadecimal position in the first octet. If this is the case, the MAC address reflects a unique address assigned to a device by its manufacturer, and the value of the OUI can be universally mapped to identify the organization that provided the address. In the above example, the final hexadecimal of the first octet is an ‘8’, indicating that the address is both unicast and universal. In fact, with a simple online search of the OUI, it will be easy to identify Intel Corporate as the vendor of the WiFi card, while the NIC will uniquely identify the WiFi card installed on my laptop. On the other hand, ‘2’, ‘3’, ‘6’, ‘7’, ‘a’, ‘b’, ‘e’ and ‘f’ values imply a local administered address with the local bit set to one. In this circumstance, the MAC address must be considered random, as detailed in further detail in the next section dedicated to MAC randomization. It is critical to understand how to discern between a unicast/multicast address and a universal/local address, as MAC addresses are contained in every packet transmitted via the network.

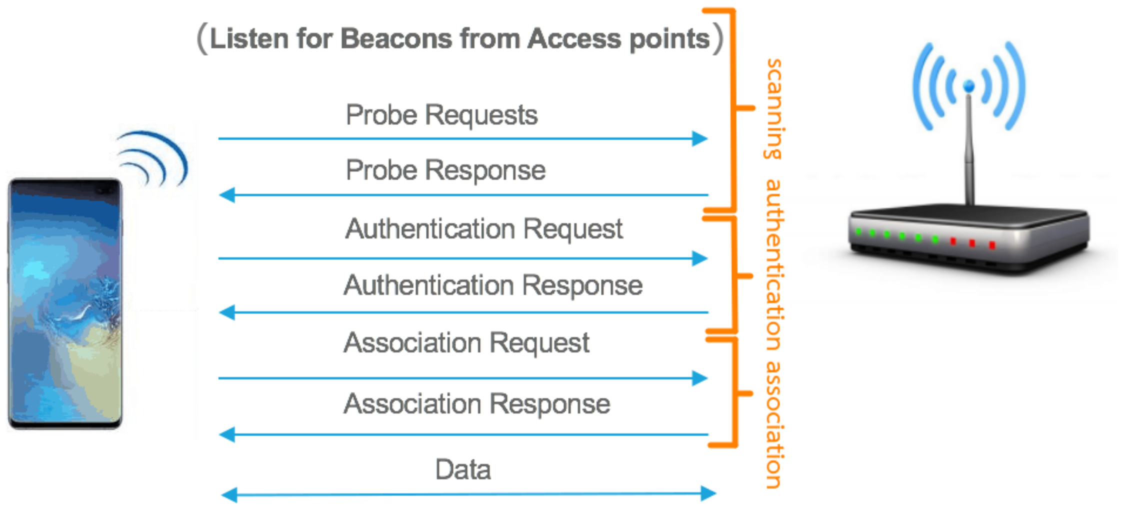


Figure 5.1: Phases and messages for the association procedure to a WiFi Access Point.

5.3.2 WiFi Access Point association procedure

To have a better understanding of the approach we employed for our APCS, we must first define the procedure that network devices must take in order to associate with a WiFi Access Point (AP). According to the standard process [97], the procedure is separated into three parts, as shown in Fig. 5.1. It is necessary to conduct an initial scanning phase in order to alert the actors to their mutual existence. **Beacon** messages are sent in broadcast mode by APs during this phase in order to notify all nearby devices of their proximity. A Beacon is a management frame that contains all of the information regarding the network that has been issued by the AP, such as the Service Set Identifier (SSID) used for identifying the wireless network name, the time remaining until the next beacon broadcast, the data rates supported by the AP and other information about the WiFi network. On the other hand, smartphones and other network devices that have a WiFi card active can send **Probe Request** messages for the same purpose. Similarly to Beacons, Probe Requests are management frames used to alert the APs in the vicinity of the transmitter's presence, as well as about the data rates and other capabilities that the sender device is capable of supporting. On the basis of the entity that initiates the scanning operation, we may distinguish between *active scanning* and *passive scanning*. In particular, APs always perform an active scanning of the WiFi medium because they periodically send Beacons over all of the available channels (from 1 to 14 and with a carrier frequency of 2.4GHz). Smartphones are capable of supporting both active and passive scanning, and they often switch between the two modes based on a variety of variables, which we will discuss in further detail below. A smartphone in active scanning mode will send Probe Request messages over all the channels with a carrier frequency of 2.4 GHz. In contrast, if a smartphone is in passive scanning mode, it does not transmit any information until it detects a Beacon from an AP; only the reception of a Beacon will cause the sending of Probe Requests. Regardless of whether the initial part of this phase was completed by active or passive scanning, the AP will respond to a Probe Request reception with a **Probe Response** message. A Probe Response is a form of acknowledgment that contains the SSID, the supported data rates, the encryption techniques (if required) and any other capabilities of the AP in question. The receipt of the Probe Response message will bring the scanning phase to a close. The scanning phase is an automated procedure that is performed on a regular basis without the requirement for any external participation from the user. The final result of this step, from the perspective of the user device, is to have complete awareness of all accessible wireless networks in its immediate vicinity at all times.

The **Authentication Request** message is the management frame that initiates the authentication phase. As compared to the previous step, the authentication phase is normally activated by the user picking a specific wireless network from among all the WiFi instances that have been detected in the former phase. Exceptions are made for certain circumstances: in the event of a well-known AP (e.g., our home wireless network or the AP at our workplace), this phase might be initiated automatically once the network has been detected during the scanning process. In any case, following the reception of a Probe Response, the device station sends an Authentication Request to the selected AP. The Authentication Request is transmitted at a low level of the ISO/OSI stack (i.e., the MAC layer) in order to make it more resistant to packet sniffing. It comprises the SSID of the wireless network to which the device will connect and a proposal for a shared key to be used for the encryption channel. If the AP accepts the proposed key, it will respond with an **Authentication Response** message over the same low-level channel, which serves as an acknowledgement. Some IEEE 802.11 capabilities allow a mobile station to establish low-level authentication with a number of different APs. When switching between APs, this shortens the time it takes to establish a connection. A mobile device, on the other hand, can only be actively associated with and transmit data through a single access point at a time.

Similarly to the authentication process, the mobile device is the entity that initiates the association phase by sending to the AP a **Association Request** frame, which is subsequently acknowledged by the AP with a **Association Response** frame. While the previous stage was essential for the negotiation of a shared key, this phase is required for the negotiation of all other parameters related to the establishment of the connection. The Association Request and Association Response frames include the chosen encryption technique, the selected data rates, the

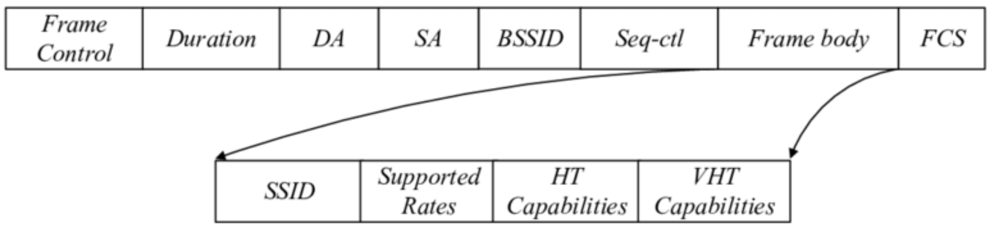


Figure 5.2: Format of a Probe Request frame.

transmission channels, and any other characteristics necessary for a functional connection to be established between two devices. Only once a mobile device has been authenticated and associated with an AP, it begin exchanging its data across the network, making use of all of the agreed settings and encryption techniques.

5.3.3 Features of Probe Request frames

The Probe Requests produced by a mobile device during the WiFi association method, as previously described, are the initial messages sent by the device during the procedure. Because they are part of the first scanning phase, our devices typically send them automatically and on a regular basis. Consequently, they are an essential component of any WiFi scanning session. The structure of a Probe Request, according to the IEEE 802.11ac standard [98], is reported in Fig. 5.2. It is a variable-length frame since it is composed of a fixed-length header and a frame body that contains mandatory and optional fields. In particular, it is composed of:

- **Frame Control**: it is the initial element of the header and contains information such as the protocol version used, the type and sub-type of the message (i.e., type 0 and sub-type 4 will refer to Probe Request frames);
- **Duration**: it indicates how long the frame and its acknowledgment frame are going to occupy the channel. Due to the fact that the AP does not respond to the station with a dedicated acknowledgement frame but rather with a separate message (i.e., the Probe Response frame), the length of the duration value in a Probe Request is always zero;
- **Destination Address (DA)**: since the Probe Request frame is always a broadcast message, the destination MAC address is always set to the broadcast one (i.e., $ff:ff:ff:ff:ff:ff$);
- **Source Address (SA)**: in this field, we are able to read the MAC address of the mobile device's WiFi card that transmits the Probe Request;

- **Basic Service Set Identifier (BSSID)**: it corresponds to the MAC address of the AP that is responsible for providing the SSID. We shall notice that this field is always set to broadcast in this study since the Probe Request is always utilized to identify new APs in the context of this analysis;
- **Sequence number (Seq-ctl)**: it represents the sequence number of the frame. It grows in size with each new Probe Request frame that is transmitted, and it can range from 0 to 4096. The 4097^{th} frame will have a sequence number equal to 0;
- **Frame Check Sequence (FCS)**: even if it is placed at the very end of the message, it is considered to be part of the header. It is a checksum that is used to ensure that the previous portion of the frame was correctly received on the receiver side.
- **Frame body**: it corresponds to the payload of the Probe Request frame. It is divided into multiple mandatory and optional fields, including the following:
  - **SSIDs list**: it is a mandatory field that contains all of the SSIDs identified by the mobile device up until the Probe Request transmission;
  - **Supported Rates**: it is a mandatory field in which the mobile station device specifies which data rates it is capable of supporting for the connection. Usually it includes only the very basic data rates supported in any standard transmission (i.e., 1 Mbit/s, 2 Mbit/s, 5.5 Mbit/s and 11 Mbit/s). If any data rate other than the basic ones is available, it is often included in the optional **Extended Supported Rates** field;
  - **Direct Sequence (DS) Parameter**: it is a parameter that is optional and reports the current channel that is being used to transmit data;
  - **High Throughput (HT) Capabilities**: it is an optional field that describes all of the parameters that are required for a high-throughput connection (i.e., IEEE 802.11n). In particular, it includes the HT Capabilities Info, the Aggregate MAC Protocol Data Unit (A-MPDU) parameters, the Supported Modulation and Coding Scheme Set, the HT Extended Capabilities, the Transmit Beam Forming Capabilities and the Antenna Selection Capabilities;
  - **Very High Throughput (VHT) Capabilities**: it is an optional field with all the parameters described in the case of a very high throughput connection (i.e., IEEE 802.11ac), including VHT Capabilities Info and Supported Modulation Coding Scheme (MCS) Set;
  - **Vendor Specific list**: it is an optional field that contains all of the OUI that has been seen up until the Probe Request transmission.

Probe Requests can be transmitted over the channel by smartphones on their own initiative, or they can be triggered by the reception of a Beacon frame, depending on whether the scanning mode is active or passive. Smartphone devices are normally capable of supporting both scanning modes, and they are capable of switching between the two. Given the fact that we will be performing a passive scan, our APCS will only record Probe Requests that are produced by smartphones on their own initiative during an active scan. Considering this, it is critical to understand the typical behavior of a smartphone's Probe Request sending and the situations under which it shifts from passive to active scanning mode.

In accordance with [99], it is clear that there is no universally applicable condition under which a smartphone switches from a passive scanning mode (i.e., waiting for a Beacon frame) to an active scanning method, in which Probe Requests are sent voluntarily over the channel. The number of Probe Requests generated and their frequency are often determined by the kind of Operating System (OS), its version, the number of well-known SSIDs sensed at the moment, and the smartphone's current usage. The only way to acquire statistics on Probe Requests is through reverse engineering studies because the techniques utilized by the manufacturers are proprietary. In a number of studies, such as [100] and [101], the amount of Probe Requests transmitted over the channel by smartphones running a different operating system was examined. A large number of Probe Requests collected over the WiFi medium is a strong indication that the device has been switched to an active scanning mode. Following a series of experiments, [100] discovered that whether the phone is connected to the charger or not, and whether Bluetooth is turned on or off, has no effect on the transmission of Probe Requests. Having the device unlocked from sleep mode, on the other hand, increases the number of Probe Requests that are sent over WiFi channels. The opposite is true in the case of a device that has already been associated with a WiFi network: it continues to send Probe Requests in order to complete a scanning throughout the channels, but the amount of Probe Request frames sent is extremely minimal in this situation. Furthermore, in [101], the authors conducted a number of human behavior-oriented tests. During their analysis, they discovered that when the smartphone is unlocked or when the user answers a call, both iOS and Android devices enter into an active mode and transmit numerous Probe Request frames. Devices using the iOS operating system continue to transmit Probe Requests when in sleep mode, while Android devices considerably decrease the number of transmissions in this situation. However, when it comes to Android, every application running in the background is accountable for the increase in the number of Probe Request frames. To summarize, we can affirm that Probe Requests are sent in bursts and that, depending on the current use of the phone, we can observe a range from a single Probe Request per minute up to 2000 Probe Requests per hour. On average, there is an 88% chance of seeing at least one Probe Request frame from a device in a 5-minute time span, while 85% of phones have an 80% chance of sending a Probe Request in a 3-minute time span.

5.3.4 The MAC randomization process

Nowadays, almost every environment presents multiple devices within our proximity capable of connecting to a wireless network, such as tablets, laptops and smartphones. As a result, each of those devices is regularly broadcasting Probe Request frames in order to scan the WiFi channel and find neighboring APs. As previously stated, due to the high frequency at which Probe Request frames are transmitted across the channel, it is quite simple to collect a large number of them from the WiFi medium. As will be discussed in further detail below, there are particular software tools available that are capable of sniffing packets received over WiFi networks. When using a sniffer program to gather frames via WiFi, the output is typically an enormously high number of Probe Requests captured from various sources.

Probe Requests, as well as all other frames transmitted to a WiFi Access Point during the association operation, are of the management type. They are needed in order to construct an encrypted channel, which the user will utilize in the succeeding for the transfer of data. As a result, all of the fields included inside any of those management frames are broadcast without any encryption. By collecting a Probe Request from the WiFi channel, it is possible to read in plain text all of the information indicated above, as well as the MAC address of the sender. According to [102], knowing the MAC address of a device is very sensitive information that might be used for a variety of cyber attacks as well as for monitoring people's movements.

A famous example of this is known as the Snowden case. Edward Snowden, a computer intelligence consultant and former employee of the Central Intelligence Agency (CIA), leaked highly classified information about the American National Security Agency (NSA) in 2013. His disclosure revealed numerous global surveillance programs, many of which were run by the NSA itself with the cooperation of telecommunication companies and European governments. Snowden, in particular, confirmed the existence of a system that follows the movements of mobile devices in a city by monitoring the MAC addresses of Probe Requests captured from the WiFi media.

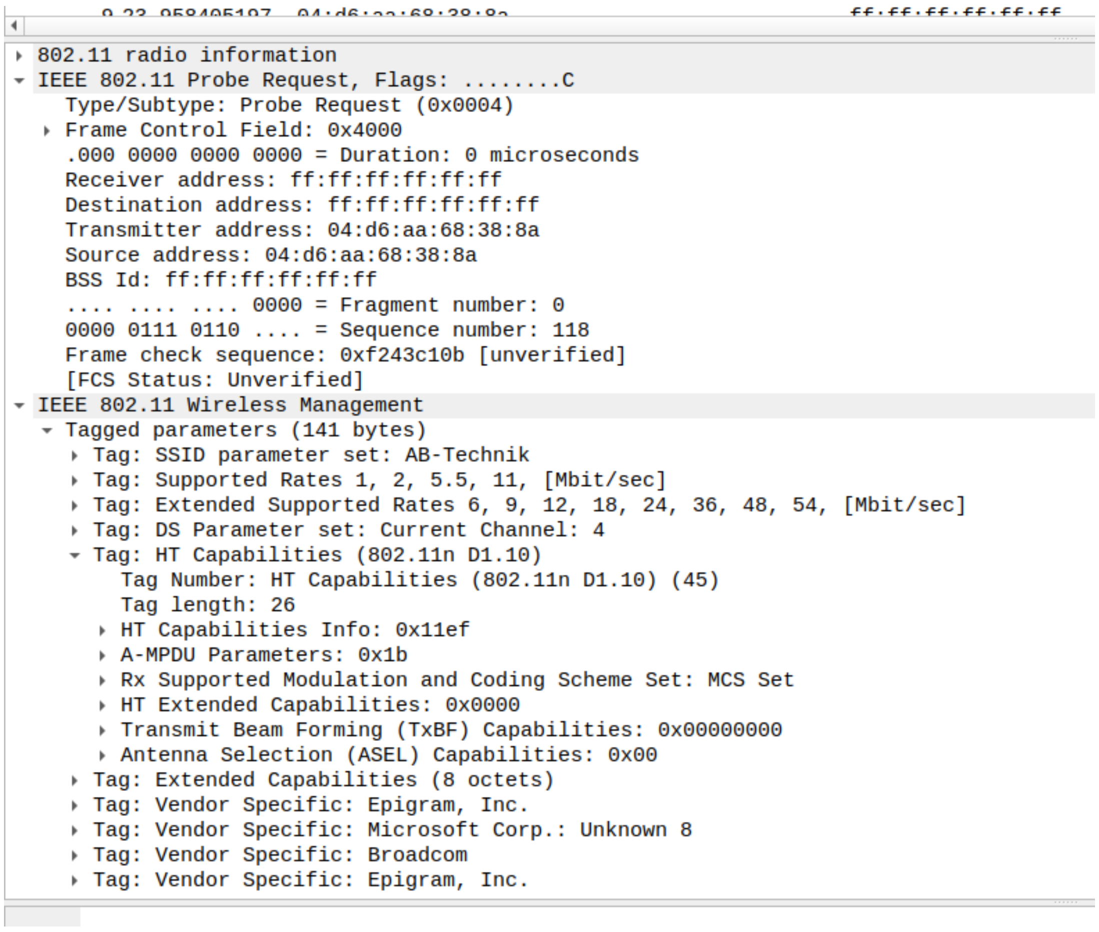
Since the Snowden case, all software providers have become more aware of the potential risks associated with packet sniffing over WiFi channels, and they have become more concerned with how to avoid disclosing sensible information in broadcast without completely changing the AP association procedure mechanism or their associated management frames. The method that they came up with is based on the randomization of MAC addresses. It was observed by software providers that, given the fact that the vast majority of Probe Request frames do not result in an actual connection to an AP but are instead used solely for scanning purposes, it is not necessary at this stage of the process to broadcast the actual MAC address. In this situation, a random MAC address is generated with the universal/local bit set to one. Real MAC addresses can be sent only in situations where it is necessary to establish an actual WiFi connection with a network AP, for example, from the authentication phase on or during a passive scanning session when the Probe Request is triggered by a Beacon frame transmitted by an AP.

The MAC randomization algorithms, like the Probe Request transmission, are proprietary and dependent on the manufacturer. In particular, Apple was the first to implement MAC randomization in 2014 for smartphones running iOS 8 [103], followed by Linux the same year with Linux Kernel 3.18 and Android the subsequent year with Android 6.0 [101]. Despite the fact that it is supported by all operating systems, many devices (particularly those running the Android operating system) do not have the MAC randomization option enabled by default. It was discovered through some reverse engineering experiments that all OSs alter their MAC addresses every time a burst of Probe Requests is issued. Moreover, [99] highlighted how devices with iOS 10.1.1 implement MAC randomization over some behavior from the user. In particular, for smartphones running this software version, a distinct MAC address is assigned each time the device is locked or unlocked, its WiFi interface is enabled or disabled, and also each time a connection with a different AP is attempted.

However, even though MAC randomization techniques help to alleviate the problem of potentially sensitive information being transmitted over the channel, they do not totally eliminate it. A number of flaws and weaknesses in these implementations have been demonstrated [104], including leveraging other elements provided in the Probe Request frame or their timing, which allows for the tracking of devices even if their MAC addresses are randomized. Since the invention of MAC randomization algorithms, de-randomization processes have been implemented in order to nullify the randomization effort. Later in this chapter, when we describe our methodology, we will also go through the decisions we made when developing our own de-randomizer algorithm.

5.3.5 Wireshark and TShark sniffing tools

In order to be able to collect Probe Request frames over the WiFi medium, we must use specific software tools that are capable of capturing and analyzing packets as they travel across the network. Those specific software packages are known as packet sniffers, and they are capable of intercepting and logging communication that travels across any network, including wireless connections. Traffic can only be collected on one channel at a time on wireless networks. Nevertheless, a capture on any one channel will be enough for our purposes because Probe Requests are delivered over every channel on wireless networks. Sniffer tools decode captured frames from their raw digital form into a human-readable format, which is then shown to the user in different forms. Packets not destined for the same SSID as the sniffer are not intercepted unless the sniffer is operating in the monitor mode. It is

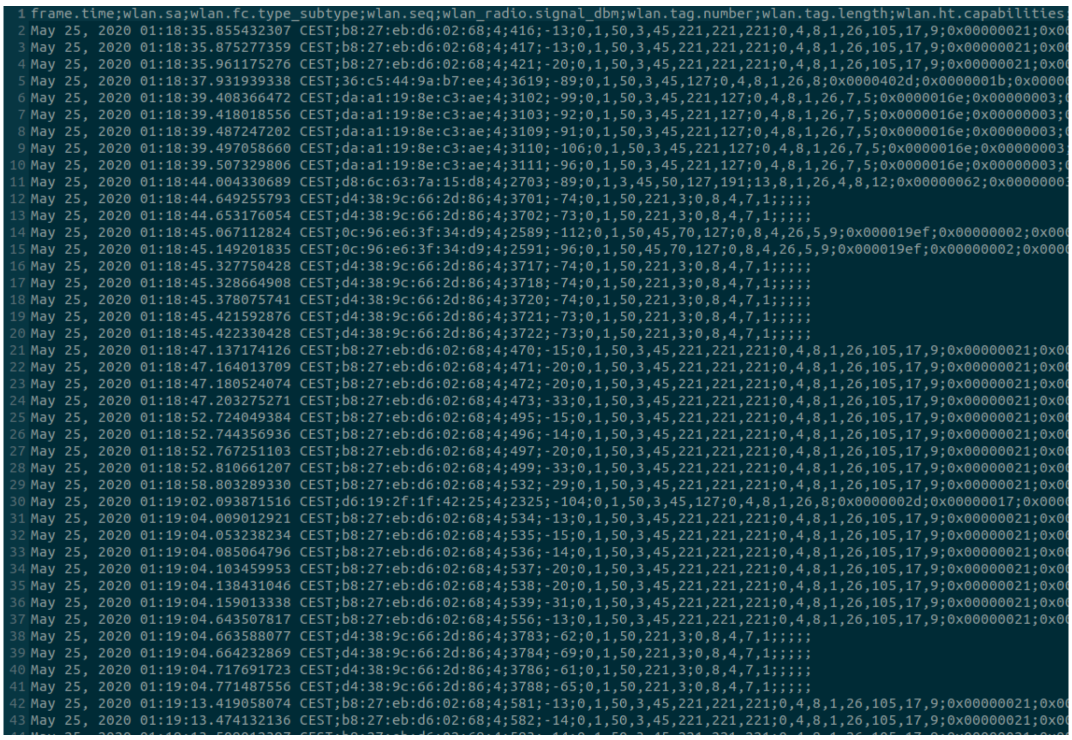


```
▸ 802.11 radio information
▾ IEEE 802.11 Probe Request, Flags: ........C
    Type/Subtype: Probe Request (0x0004)
  ▸ Frame Control Field: 0x4000
    .000 0000 0000 0000 = Duration: 0 microseconds
    Receiver address: ff:ff:ff:ff:ff:ff
    Destination address: ff:ff:ff:ff:ff:ff
    Transmitter address: 04:d6:aa:68:38:8a
    Source address: 04:d6:aa:68:38:8a
    BSS Id: ff:ff:ff:ff:ff:ff
    .... .... .... 0000 = Fragment number: 0
    0000 0111 0110 .... = Sequence number: 118
    Frame check sequence: 0xf243c10b [unverified]
    [FCS Status: Unverified]
▾ IEEE 802.11 Wireless Management
  ▾ Tagged parameters (141 bytes)
    ▸ Tag: SSID parameter set: AB-Technik
    ▸ Tag: Supported Rates 1, 2, 5.5, 11, [Mbit/sec]
    ▸ Tag: Extended Supported Rates 6, 9, 12, 18, 24, 36, 48, 54, [Mbit/sec]
    ▸ Tag: DS Parameter set: Current Channel: 4
    ▾ Tag: HT Capabilities (802.11n D1.10)
        Tag Number: HT Capabilities (802.11n D1.10) (45)
        Tag length: 26
      ▸ HT Capabilities Info: 0x11ef
      ▸ A-MPDU Parameters: 0x1b
      ▸ Rx Supported Modulation and Coding Scheme Set: MCS Set
      ▸ HT Extended Capabilities: 0x0000
      ▸ Transmit Beam Forming (TxBF) Capabilities: 0x00000000
      ▸ Antenna Selection (ASEL) Capabilities: 0x00
    ▸ Tag: Extended Capabilities (8 octets)
    ▸ Tag: Vendor Specific: Epigram, Inc.
    ▸ Tag: Vendor Specific: Microsoft Corp.: Unknown 8
    ▸ Tag: Vendor Specific: Broadcom
    ▸ Tag: Vendor Specific: Epigram, Inc.
```

Figure 5.3: Probe Request frame captured with Wireshark tool.

the only method to capture the management frames sent between network entities during the association procedure, since the monitor mode permits packets to be collected without the need to associate with an AP.

The most well-known and extensively used tool for this purpose is called Wireshark [105]. Wireshark is a free and open-source packet analyzer that can be run on practically any operating system. It offers a very efficient and user-friendly visual interface in which recorded packets are displayed and grouped according to their protocol or packet type using color-coding principles. A further feature is the ability to choose individual recorded packets for further inspection. In this case, we can see an examination of every field of the sniffed frame. As an example, Fig. 5.3 depicts a detail of the fields from a Probe Request frame acquired using Wireshark. The use of a terminal-based capture tool is necessary in the case of a packet capture done by a sensor or a device that does not have display capabilities. For this reason, the Wireshark team created TShark [106], a terminal-based version of Wireshark that has all of the functionality included in its predecessor. TShark is perfectly suited



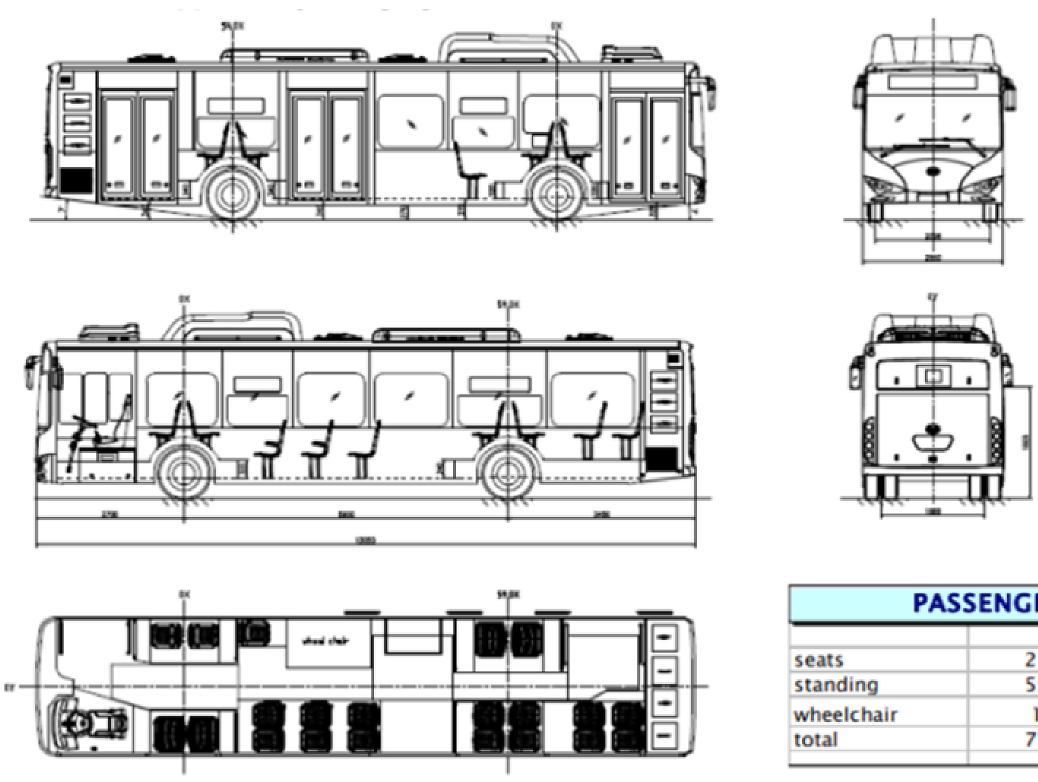
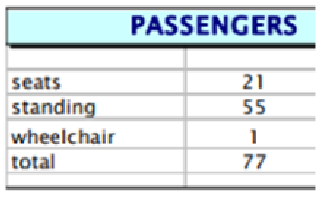
```
frame.time;wlan.sa;wlan.fc.type_subtype;wlan.seq;wlan_radio.signal_dbm;wlan.tag.number;wlan.tag.length;wlan.ht.capabilities;
May 25, 2020 01:18:35.855432307 CEST;b8:27:eb:d6:02:68;4;416;-13;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:35.875277359 CEST;b8:27:eb:d6:02:68;4;417;-13;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:35.961175276 CEST;b8:27:eb:d6:02:68;4;421;-20;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:37.931939338 CEST;36:c5:44:9a:b7:ee;4;3619;-89;0,1,50,3,45,127;0,4,8,1,26,8;0x0000402d;0x0000001b;0x00000
May 25, 2020 01:18:39.408366472 CEST;da:a1:19:8e:c3:ae;4;3102;-99;0,1,50,3,45,221,127;0,4,8,1,26,7,5;0x0000016e;0x00000003;
May 25, 2020 01:18:39.418018556 CEST;da:a1:19:8e:c3:ae;4;3103;-92;0,1,50,3,45,221,127;0,4,8,1,26,7,5;0x0000016e;0x00000003;
May 25, 2020 01:18:39.487247202 CEST;da:a1:19:8e:c3:ae;4;3109;-91;0,1,50,3,45,221,127;0,4,8,1,26,7,5;0x0000016e;0x00000003;
May 25, 2020 01:18:39.497058660 CEST;da:a1:19:8e:c3:ae;4;3110;-106;0,1,50,3,45,221,127;0,4,8,1,26,7,5;0x0000016e;0x00000003
May 25, 2020 01:18:39.507329806 CEST;da:a1:19:8e:c3:ae;4;3111;-96;0,1,50,3,45,221,127;0,4,8,1,26,7,5;0x0000016e;0x00000003;
May 25, 2020 01:18:44.004330689 CEST;d8:6c:63:7a:15:d8;4;2703;-89;0,1,3,45,50,127,191;13,8,1,26,4,8,12;0x00000062;0x0000000
May 25, 2020 01:18:44.649255793 CEST;d4:38:9c:66:2d:86;4;3701;-74;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:18:44.653176054 CEST;d4:38:9c:66:2d:86;4;3702;-73;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:18:45.067112824 CEST;0c:96:e6:3f:34:d9;4;2589;-112;0,1,50,45,70,127;0,8,4,26,5,9;0x000019ef;0x00000002;0x000
May 25, 2020 01:18:45.149201835 CEST;0c:96:e6:3f:34:d9;4;2591;-96;0,1,50,45,70,127;0,8,4,26,5,9;0x000019ef;0x00000002;0x0000
May 25, 2020 01:18:45.327750428 CEST;d4:38:9c:66:2d:86;4;3717;-74;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:18:45.328664908 CEST;d4:38:9c:66:2d:86;4;3718;-74;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:18:45.378075741 CEST;d4:38:9c:66:2d:86;4;3720;-74;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:18:45.421592876 CEST;d4:38:9c:66:2d:86;4;3721;-73;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:18:45.422330428 CEST;d4:38:9c:66:2d:86;4;3722;-73;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:18:47.137174126 CEST;b8:27:eb:d6:02:68;4;470;-15;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:47.164013709 CEST;b8:27:eb:d6:02:68;4;471;-20;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:47.180524074 CEST;b8:27:eb:d6:02:68;4;472;-20;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:47.203275271 CEST;b8:27:eb:d6:02:68;4;473;-33;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:52.724049384 CEST;b8:27:eb:d6:02:68;4;495;-15;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:52.744356936 CEST;b8:27:eb:d6:02:68;4;496;-14;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:52.767251103 CEST;b8:27:eb:d6:02:68;4;497;-20;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:52.810661207 CEST;b8:27:eb:d6:02:68;4;499;-33;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:18:58.803289330 CEST;b8:27:eb:d6:02:68;4;532;-29;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:02.093871516 CEST;d6:19:2f:1f:42:25;4;2325;-104;0,1,50,3,45,127;0,4,8,1,26,8;0x0000002d;0x00000017;0x0000
May 25, 2020 01:19:04.009012921 CEST;b8:27:eb:d6:02:68;4;534;-13;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:04.053238234 CEST;b8:27:eb:d6:02:68;4;535;-15;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:04.085064796 CEST;b8:27:eb:d6:02:68;4;536;-14;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:04.103459953 CEST;b8:27:eb:d6:02:68;4;537;-20;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:04.138431046 CEST;b8:27:eb:d6:02:68;4;538;-20;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:04.159013338 CEST;b8:27:eb:d6:02:68;4;539;-31;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:04.643507817 CEST;b8:27:eb:d6:02:68;4;556;-13;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:04.663588077 CEST;d4:38:9c:66:2d:86;4;3783;-62;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:19:04.664232869 CEST;d4:38:9c:66:2d:86;4;3784;-69;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:19:04.717691723 CEST;d4:38:9c:66:2d:86;4;3786;-61;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:19:04.771487556 CEST;d4:38:9c:66:2d:86;4;3788;-65;0,1,50,221,3;0,8,4,7,1;;;;;
May 25, 2020 01:19:13.419058074 CEST;b8:27:eb:d6:02:68;4;581;-13;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
May 25, 2020 01:19:13.474132136 CEST;b8:27:eb:d6:02:68;4;582;-14;0,1,50,3,45,221,221,221;0,4,8,1,26,105,17,9;0x00000021;0x00
```

Figure 5.4: Probe Request frame captured with TShark tool.

for on-the-fly packet analysis that has to take place immediately after the capture and has to be conducted in an automated manner. In comparison to Wireshark, it can gather the same information in a more compact way, as illustrated in Fig. 5.4, where it is possible to observe a log of a capture of Probe Request frames done with TShark. In the image, we can see a large number of Probe Requests, each of which is represented by a single line. It should be noted that some of them have values for all of the optional fields, but others are shorter since they only include the obligatory data. Because we will be developing an Automatic Passenger Counting System (APCS) in this project, all of the packets recorded from the WiFi medium will need to be examined on the fly by a sensor as they are received. As a result, TShark is the only tool that meets all of our needs properly.

5.4 Methodology

This project stems from the request of GTT, the local transportation agency operating in the province of Turin, to be able to reliably count the number of people on board their public buses through the use of an online real-time service. Our solution was to develop an Automatic Passenger Counting System (APCS), which consisted of installing a sensor with specific hardware and software capabilities

PASSENGERS	
seats	21
standing	55
wheelchair	1
total	77

Figure 5.5: The GTT bus schema.

on a public GTT bus. With this sensor, we will be able to record and evaluate the number of Probe Request frames transmitted over the WiFi medium by each smartphone. To cope with MAC randomization, our APCS is able to detect the MAC addresses that are most likely to be randomly generated. In this circumstance, by utilizing a custom de-randomizer script, it attempts to correlate several random MAC addresses with a single unique device that has a high probability of generating them. The final result of all of this is a simple integer that represents the number of people currently on board the bus with high frequency and reliability. Specifically, in this section, we will go through the methodological procedure that we used to develop our APCS. In particular, in the following subsections, we will detail the bus test given by GTT on which we conducted our tests, as well as the network that was placed on every GTT bus to which our APCS was connected. In the next subsection, we'll go through our hardware solutions and the reasoning behind our decisions. Eventually, we will examine the software scripts that were created for the purpose of developing our APCS, which is capable of capturing Probe Requests and analyzing them on the fly.

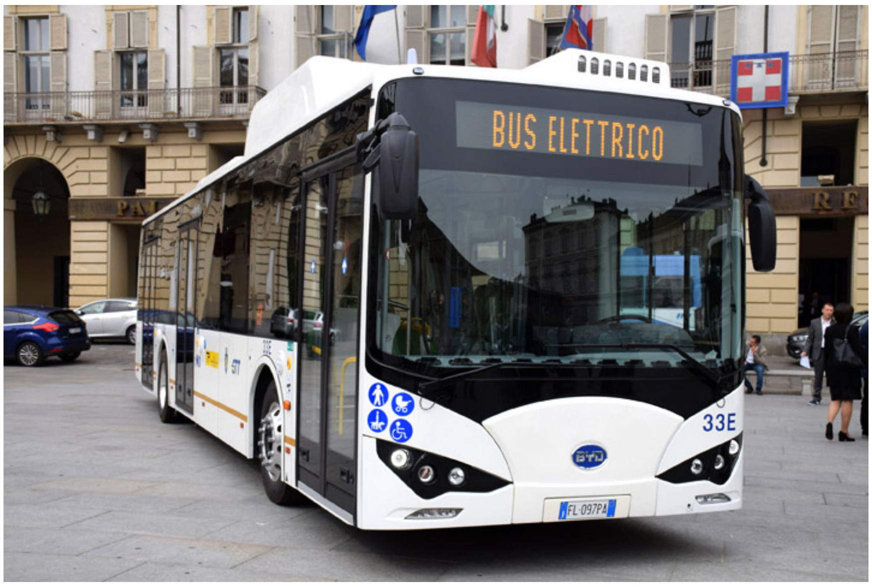
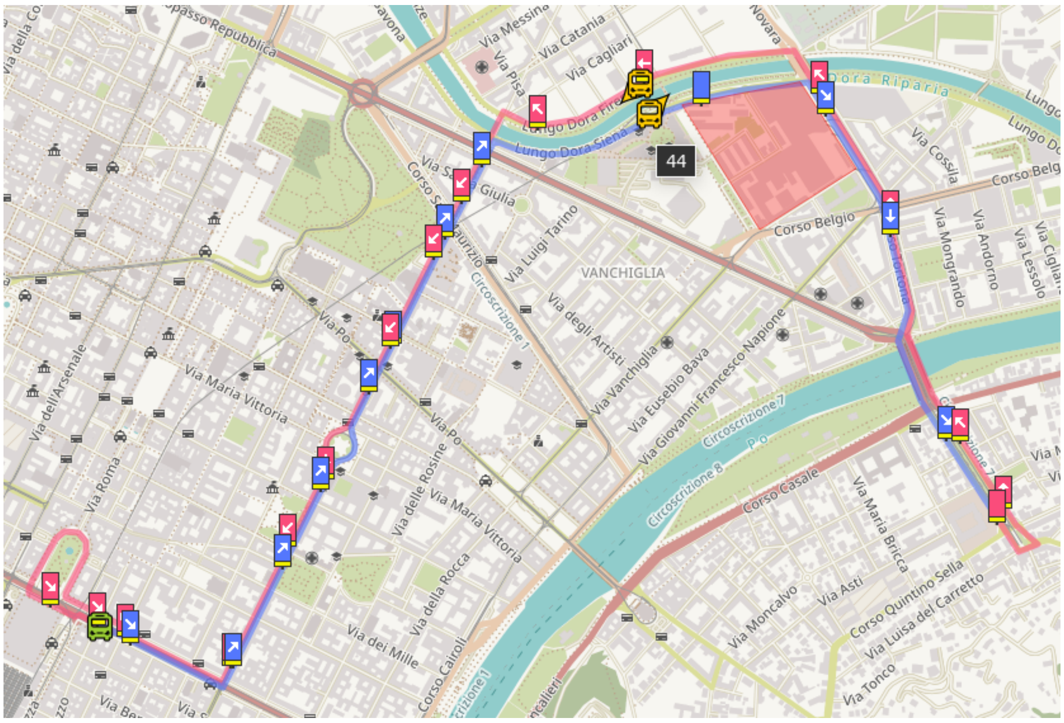


Figure 5.6: The GTT electric bus 33E.

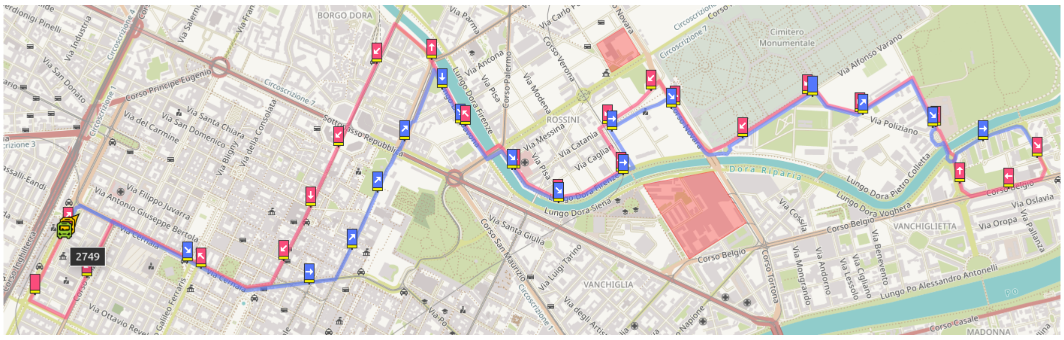
5.4.1 The GTT bus

The bus provided by GTT for our experiments is a BYD K9UB model, and its schema is depicted in Fig. 5.5. It is a 12-meter-long electric bus that can accommodate a total of 77 passengers, including the driver. According to GTT rules, each bus has its own unique identification number that is printed on the front of the vehicle, on both sides, and on the back of the bus. It differs from the line number, which is also shown on a monitor on the front of the bus as well as on small screens on both the bus's sides and on the back of the vehicle. The identification number refers to the vehicle itself, whilst the line number indicates which bus line is being served by that specific bus. During the span of its operating day, a single bus vehicle will typically serve more than one line. To be more specific, the bus that GTT provided to us is the electric bus 33E, which is depicted in Fig. 5.6, and which typically serves lines 6 and 19, which pass through the city center.

Every public GTT bus is equipped with a dedicated network, which is composed of Power over Ethernet (PoE) cables placed in dedicated vans under the bus roof. All messages produced on the bus network are controlled by a router, which then transmits the relevant data to a public IP server of GTT through a mobile LTE interface. Our bus, in particular, is equipped with two security cameras that record a live stream and may transfer it to the GTT server upon request. On top of that, there are signals arriving on the bus network from BIP machines and from the vehicle diagnostic system (e.g., current position, speed, bus line, bus ID, door status, next bus stop, etc.). Since all of the messages carried on the bus network are UDP messages, connecting our sensor to the internal bus network through a



(a)



(b)

Figure 5.7: Bus lines from FALCO online platform. (a) Line number 6. (b) Line number 19.

PoE connection allows us to read them and utilize them in our analysis.

Considering that different buses might serve a single line in a day and that a single bus can normally serve more than one line, the vehicle on which we put our sensor was not always in service. Furthermore, according to GTT's line plans, a bus vehicle may be out of service for several days or even weeks if it is scheduled for routine maintenance. As a result, we were required to verify the current state of the 33E bus on a regular basis, and we achieved this through the use of the FALCO [107] online platform developed by 5T. The FALCO platform, which can be accessed at

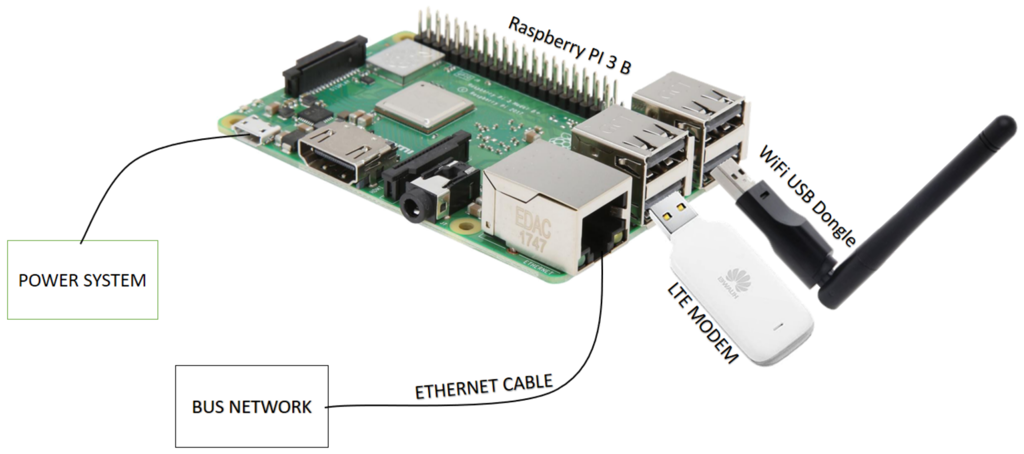


Figure 5.8: Hardware solution composed by a Raspberry Pi 3 Model B, a USB WiFi dongle and a USB 3G and 4G/LTE modem.

https://falco.5t.torino.it/, displays the current position of GTT buses after receiving the bus line as an input parameter. The bus lines 6 (Fig. 5.7a) and 19 (Fig. 5.7b), both of which were served by the bus 33E, are depicted in Fig. 5.7. When you hover the cursor over a bus icon, the bus's identification number is shown, as seen in the images for buses 44 on line 6 and 2749 on line 19.

5.4.2 The hardware choices

Considering that our goal is to design a sensor-based system that could be put on a number of public buses (or other public areas), we needed to keep the prices of our hardware low while still providing adequate processing power to allow for on-the-fly analysis. Using Raspberry Pi (RP) hardware offered the best compromise. The RP is a single-board computer with cheap prices, significant versatility, and an open architecture. We utilized a Raspberry Pi 3 Model B, which has a 1.2 GHz 64-bit quad-core ARM Cortex-A53 CPU, on-board 802.11n WiFi, Bluetooth, 1 GB RAM, an Ethernet connection, four USB ports, and runs the Linux OS. The WiFi interface of the Raspberry Pi 3 Model B does not support monitor mode by default, so we bought a simple USB dongle antenna for recording frames on the WiFi channel using monitor mode as well. To connect to our RP from other networks, we utilized a mobile USB modem that can connect to 3G and 4G/LTE networks through a SIM card. Finally, a PoE splitter was utilized to connect our RP to the PoE cable for both power and Ethernet. Our complete hardware system is shown in Fig. 5.8 and has a total cost of less than 50€ for all the components.

Since our RP is linked to a PoE cable from the bus network, which supplies both Ethernet and power, when the bus shuts off, the entire system will power down. This generally occurs at the end of a bus line, when the bus pauses for a few

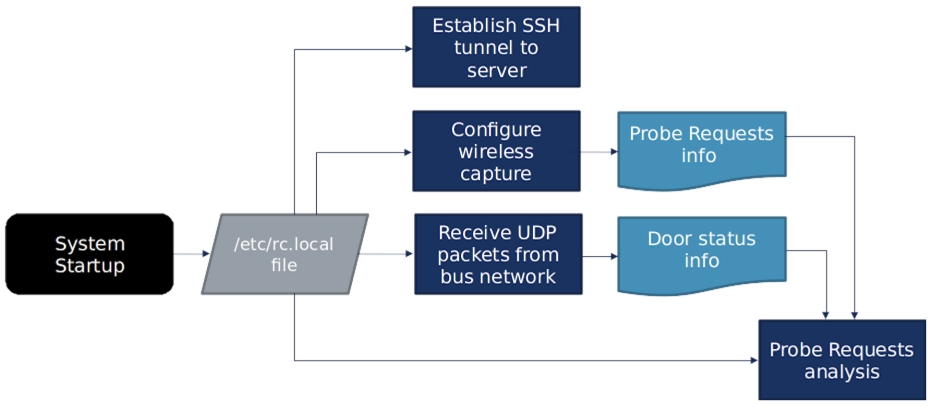

Establish SSH tunnel to server
Configure wireless capture
Probe Requests info
System Startup
/etc/rc.local file
Receive UDP packets from bus network
Door status info
Probe Requests analysis

Figure 5.9: Flowchart of the software process of our APCS.

minutes before departing again, or at the central GTT deposit at the conclusion of the bus service day. Because our system has no way of knowing when the bus will switch off in advance, our whole hardware and software solution must be robust against sudden power cuts. From this perspective, the RP is the best option since it is designed to be used as an external sensor and can tolerate sudden shutdowns and restarts without causing any hardware damage. Additionally, after an abrupt disconnect, the WiFi dongle and LTE modem, like any other USB device, can immediately restart.

5.4.3 The software processes

As described before, when an unexpected shutdown happens, we must ensure that all of our system's software processes are properly restarted. In order to do this, all software processes are called on the RP from a particular `rc.local` file located in the `/ect/` directory. `/ect/rc.local` is a file that is automatically run on system boot. Because every script launched from this file must be run in the background, it is necessary to include the command '&' after the launch in order to prevent the entire system from becoming unproductive while waiting for a script to finish. We run four processes in parallel from the `/ect/rc.local` file: a first process is responsible for establishing a remote connection toward a public IP server in order to form a remote connection for accessing to our RP, a second process reads the messages from the bus network and retrieves the information regarding the door status, a third process is responsible for capturing WiFi frames over the channel, and a fourth process is responsible for analyzing captured Probe Requests in order to estimate the number of passengers currently on board. In the following, we will look at each process that is contained within the `/ect/rc.local` file, composing the total software flow shown in Fig. 5.9.

The SSH tunnel

The installation of our RP sensor on the bus necessitated the development of a method for establishing a remote connection with it. The quickest and most direct method would be to pass through the GTT router that is already present on the bus network and to reach it from the GTT server with a public IP address. However, we were unable to implement this approach since GTT refused to grant us access to their server. In light of this, we determined that a USB modem with a SIM card would be appropriate for connecting our RP via 3G and 4G/LTE mobile networks. The USB modem enables our RP to communicate with external devices, but not the other way around. In order to resolve this issue, we used the `autossh` command to establish a long-lasting connection with a Polito server that had a public IP address. The `autossh` command allows the user to establish an SSH tunnel that is both durable and bidirectional. It works by creating two connections to a computer with a public IP address: the first connection is used for keep-alive messages and management frames, while the second connection is devoted to the user in order to allow for remote control. By using two basic SSH commands, we were able to connect to our remote sensor and collect data from it. The first SSH is used to connect to the Polito server using its public IP address from our laptop. The second SSH command is run with the line `sudo ssh -p 6000 pi@localhost`, where `-p 6000` specifies the port number determined by `autossh`, and `pi@localhost` associates the username of the RP (i.e., `pi`) with the remote control connection from our server to the RP itself.

Considering that our SSH tunnel is the only point of connection to our remote site without requesting that GTT halt the bus for a manual intervention, we added additional log check procedures to make our SSH tunnel more robust to network problems. In this way, we can be sure that the SSH connection will be reestablished in the event of a failure. The pseudo-code for our SSH tunnel is in Algorithm 3.

Algorithm 3 SSH tunnel pseudo-code.

```
1: while true do
2:     Kill any preexisting autossh process.
3:     Launch in background the autossh towards our Polito server with public IP
       and with a log file for any error.
4:     while $size$ is 0 do
5:         $size$ = bytes dimension of the log file reporting any autossh error.
6:         Sleep for 3 seconds.
7:     end while
8:     Sleep for 5 minutes.
9: end while
```

The door status reader

Because our goal is to estimate the current number of passengers on the bus at a particular time, the occasions on which the bus is halted at a bus stop while passengers board represent an unstable condition. During those periods, we can see individuals getting on and off the bus, resulting in a non-constant number of passengers on board. As a consequence, we chose to only have our APCS working while the bus doors are closed, which means that the bus is not currently stationed at a bus stop. In order to accomplish this, we instruct our RP to listen on a UDP socket for messages arriving from the bus network. Among the signals received from the bus's internal network, the information on the current door state is retrieved and used in the following steps.

The capturing procedure

This is the software mechanism for initiating the capture of the Probe Request across the WiFi network medium. First, we discovered the interface of our USB antenna that was responsible for collecting WiFi packets, and we deactivated the WiFi interface of our RP that would have otherwise caused interference. Then we configured the WiFi interface of the USB antenna to operate in monitor mode in order to record all of the management frames that were broadcast over the wireless network. At long last, we started the TShark command to begin capturing the Probe Requests using the following command:

```
sudo tshark -i WIFI_INTERFACE -n -t e -T fields -e frame.time \
    -e wlan.sa -e wlan.fc.type_subtype -e wlan.seq \
    -e wlan_radio.signal_dbm -e frame.len -e wlan.ht.capabilities \
    -e wlan.ht.ampduparam -e wlan.htex.capabilities \
    -e wlan.txbf -e wlan.asel -E header=y -E separator=";" \
    subtype probe-req > FILENAME
```

With this instruction, we are able to run the TShark capture with the following parameters:

- `-i WIFI_INTERFACE`: this option allows us to choose the WiFi interface from which we wish to take data. In our situation, we will substitute the name of our WiFi USB antenna's interface for the `WIFI_INTERFACE` variable;
- `-n`: by selecting this option, we were able to deactivate name resolution for MAC addresses. Using this method, it is possible to see the complete MAC address in plain text;
- `-t e`: this option sets the timestamp format to epoch time (i.e., the time in milliseconds from midnight of January 1^{st}, 1970 to the present date);

- `-E header=y`: when we selected this option, we indicated that we desired a header line in our output file;
- `-E separator=";"`: this field defines the separator value for each of the fields in the output file;
- `subtype probe-req`: it is a TShark filter that captures just frames of a specific kind. In this instance, we simply recorded Probe Request frames;
- `> FILENAME`: the result of this command is a file named FILENAME, which contains all of the captured information;
- `-T fields -e`: when we used the `-T fields` option, we stated that we wanted our collected frames to be divided into fields. In particular, the `-e` option allows us to specify the fields we are interested in, which is really convenient for a compact representation. For our purposes, we recorded the following:
  - `frame.time`: the timestamp at when the Probe Request communication was sent;
  - `wlan.sa`: the MAC address of the transmitter;
  - `wlan.fc.type_subtype`: it is a unique identification number that is used to distinguish between different frame types. Type '4' is used for Probe Requests;
  - `wlan.seq`: this field contains the sequence number of the Probe Request frame;
  - `wlan_radio.signal_dbm`: it is the signal power sensed at the receiver side expressed in dBm;
  - `frame.len`: it is the total length of the Probe Request frame in bytes;
  - `wlan.ht.capabilities`: the high throughput capabilities of the transmitter device are reported in this field;
  - `wlan.ht.ampduparam`: it represents the A-MPDU parameters;
  - `wlan.htex.capabilities`: this field reports the extended high throughput capabilities, if any;
  - `wlan.txbf`: it is the transmit beam forming of the frame;
  - `wlan.asel`: this field represents the antenna selection capabilities.

It should be noted that the vast majority of the collected fields correspond to technical properties of the transmitter (e.g., the high throughput capabilities, the A-MPDU parameters, the very high throughput capabilities, the beam forming and antenna selection). Because they are all specific to the same smartphone, all of this information could be crucial in identifying a device.

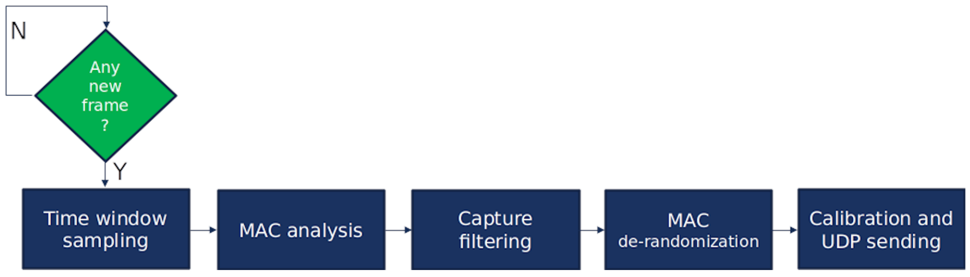


Figure 5.10: Flowchart of the Probe Request analysis.

Probe Request analysis

The examination of the Probe Requests that have been recorded is carried out in real time. The analysis method continually monitors the file created by the capture command, in which all of the Probe Requests are documented, as well as the information on the status of the door. The analysis is divided into five phases, each with its own script file. The five scripts for the analysis are run in a pipeline, one after the other. The pipeline is a programming process that allows developers to execute consecutive programs in a chain, with the output of one script being utilized as input for the script that follows it in the pipeline. In this manner, we are able to perform numerous sophisticated operations on the same input instance in a modular manner. The complete pipeline with all the scripts is described below and it is represented in Fig. 5.10.

1) Time window sampling: given the fact that Probe Requests generated by cellphones are transmitted in bursts, it is not practical to perform sophisticated analysis on a single frame of information. Performing an analysis on a consistent set of frames is essential for obtaining a clear picture of what is taking place aboard a bus at any point in time. The purpose of the first script in our pipeline is to sample the Probe Requests into time windows, which will form the basis for

Algorithm 4 Time window sampling pseudo-code.

```
1: while there is a new Probe Request frame do
2:     Check the current door status.
3:     if doors are closed then
4:         Append the new Probe Request frame in the time window.
5:         Send the time window down the pipeline.
6:     else
7:         Delete the current time window and create a new empty one.
8:     end if
9: end while
```

the subsequent inspections and investigations. To this extent, the sampling script continually monitors the door status, obtained via UDP from the bus network, as well as the Probe Requests recorded in the capturing output file. When the bus doors are closed, the new Probe Request frame is added into the final position of the time window, and the sample formed in this manner is sent further down the pipeline for additional analysis. Every time the door status indicates a door opening at a bus stop, the sampled window is deleted and regenerated from the beginning as soon as the doors close once more. As a result of this strategy, we can be certain of creating an evaluation sample that includes all of the time between two consecutive door openings, which is roughly equivalent to the period between two consecutive bus stops. The pseudo-code of the time window sampling is reported in Algorithm 4.

2) MAC address analysis: since Probe Request frames are broadcast in bursts, we will detect several occurrences of Probe Request frames with the same MAC address in our sampling window. The purpose of this script is to collect all of the

Algorithm 5 MAC address analyzer pseudo-code.

```
while true do
    Read the new input from the pipeline.
    for all the records in the new time window do
        if it is the first instance of this MAC address in the time window then
            if the MAC has the universal/local bit set to one then
                Mark the MAC address as random.
            else
                Mark the MAC address as universal.
                Check the OUI list for knowing the manufacturer.
            end if
            Save the timestamp as the first time this MAC address was seen.
            Save the sequence number as the first for this MAC address burst.
            Set to one the occurrences for this MAC address.
            Save the power level at receiver side for the average.
            Save all the technical characteristics for this MAC address.
        else
            Save the timestamp as the last time this MAC address was seen.
            Save the sequence number as the last for this MAC address burst.
            Update the number of occurrences for this MAC address.
            Update the average power sensed at the receiver side.
        end if
    end for
    Send the new sample down the pipeline.
end while
```

frames from a single MAC address and save them in a more compact format. This is derived from the fact that Probe Requests with the same MAC value for the source address field were certainly sent by the same device. This script receives as input the sampling time window generated by the preceding step in the pipeline and scans all the recorded frames in order to carry out its analysis. The program specifically makes a separate record for each distinct MAC address that occurs inside the sampling time window, along with some additional information for each record that is generated. To be more specific, for each distinct MAC, it analyzes the universal/local bit and determines if the MAC is universal or random; if the MAC is not random, it checks which manufacturer it is from an OUI list stored on the RP. Furthermore, it computes the number of occurrences for that specific MAC address and stores the earliest and last timestamps at which that particular address was seen in the input sample. In a similar manner, the script detects the first and final sequence numbers for a MAC address and computes the average power perceived at the receiver. All of the technical information (i.e., high throughput capabilities, A-MPDU parameters, very high throughput capabilities, beam forming and antenna selection), which is nominally constant for one device, is eventually gathered into a single string. At the end of the process, a single record will contain all of the information linked to the burst of an identical MAC address, making our time window sample a significantly more compact representation of the original data set. The new sample that has been generated in this manner is then sent down the pipeline for the next phase of the analysis. The pseudo-code for the MAC analyzer is reported in Algorithm 5.

3) Power and occurrences filter: during our investigation, we intend to analyze the Probe Requests transmitted by the phones of people who are present on board. This necessitates the use of filters to limit our examination range to only the bus. It is probable that some of the Probe Request frames collected by our sensor were

Algorithm 6 Power and occurrences filter pseudo-code.

```
 1: while true do
 2:     Read the new input from the pipeline.
 3:     for all the records in the input sample do
 4:         if the record has an average power below $P_l$ or the record has less frame
            occurrences than $O_l$ then
 5:             Discard the record.
 6:         else
 7:             Save the record in a new sample.
 8:         end if
 9:     end for
10:     Send the new sample down the pipeline.
11: end while
```

not transmitted by people present on board, but rather by people on the outside of the vehicle. This is especially common when the bus is stopped at a traffic light and is surrounded by other cars and people. As a solution to this issue, we designed a script that reads in input from the pipeline and filters the sample of Probe Requests based on the average power level and the number of occurrences in the burst. To be more specific, we establish a power level threshold P_l as well as an occurrence level threshold O_l. The average power level may be used to determine the distance between the transmitter and the sensor, while a limited number of frames in a burst can indicate that a device is passing close to the bus for a brief amount of time. A record in input with all the data of a burst of Probe Requests is discarded if that burst has an average power below P_l or if it is formed by less than O_l frames. The records that are not eliminated in this manner are extremely likely to belong to devices on board, and they are passed down the pipeline for the de-randomization procedure. The pseudo-code of the power and occurrences filter is shown in Algorithm 6. According to what we will see in the following, P_l and O_l are two parameters that must be appropriately tuned in order to be correctly configured for our specific environment. It was necessary to conduct multiple trials and compare the results with the real number of people on board, as determined by manual counting sessions, in order to achieve this.

4) MAC de-randomization: it is the process at the center of our Probe Request analysis. Based on the information received from the pipeline, it attempts to assess if two bursts of Probe Requests with different randomized MAC addresses are likely to belong to the same device. The data obtained in input is a set of bursts made of Probe Request frames, where each burst has the same MAC address, referred to in the following as MAC$_{current}$. Our de-randomization procedure is based on the iABACUS method [108], where the authors established a recursive mechanism to reverse engineer the MAC randomization technique. When using the iABACUS algorithm, a collection of lists is generated in which different random MAC addresses are associated with a single device. The MAC$_{current}$ address in the input sample is compared to every other MAC address in the existing lists to see if they are compatible (if there is no list present, then a first list with only the MAC$_{current}$ address in it is created). A couple of conditions must be met in order for the MAC$_{current}$ address to be inserted into the list: i) the Probe Requests being inspected must have the same technical characteristic fields as all the other frames in the list; ii) the temporal window of the current burst of Probe Requests must not overlap the temporal windows of other bursts in the list. The first requirement is derived from the nature of the technical characteristic fields of a Probe Request frame: because they provide information that is required for establishing a connection with an AP, they are nominally constant for the same device. The second requirement is related to the fact that the MAC randomization mechanism selects random values for producing a new MAC address, so it is extremely improbable that a device would utilize a random MAC address that has previously been used by another device.

As a result, two bursts of Probe Requests with distinct random MAC addresses that were created by the same device would not coexist at the same time. The burst of frames inspected must meet both requirements for one or more existing lists to be considered valid for a match; otherwise, we have most likely identified a new device, and a new list is created with the MAC$_{current}$ address of the burst inspected. On the contrary, if one or more lists match the two initial requirements, those lists are identified as potential candidates for a match, and the de-randomization algorithm proceeds focused just on those lists.

The second part of the de-randomization algorithm is focused on the computation of a match score between MAC addresses. In particular, the Score$_{i,j}$ can be defined as the match score computed between a MAC$_i$ and a MAC$_j$ in the following way:

$$Score_{i,j} = \left|\frac{1}{\Delta TimeWindow_{i,j}}\right| \cdot \left|\frac{1}{\Delta SequenceNumber_{i,j}}\right| \cdot \left|\frac{1}{\Delta AvgPower_{i,j}}\right|$$

where $\Delta TimeWindow_{i,j}$ is the number of milliseconds elapsed between the last frame seen from the MAC$_i$ burst and the first frame seen from the MAC$_j$ burst; $\Delta SequenceNumber_{i,j}$ is the difference between the final sequence number used in the MAC$_i$ burst and the first sequence number used in the MAC$_j$ burst; the difference between the two average powers of the two bursts is represented by $\Delta AvgPower_{i,j}$. After computing the Score$_{i,j}$ in this manner, a positive number larger than zero is obtained, and it shows how likely it is that the two random MAC addresses, MAC$_i$ and MAC$_j$, are associated with the same device. The greater the score value, the greater the probability that two MAC addresses are attributable to the same source device. The core principle behind this formula is that two successive bursts of Probe Requests with different random MAC addresses are most likely two instances of the same device if they are almost consecutive in time and sequence numbers and if they have almost the same average power. By computing the match score value between MAC$_{current}$ and all the MAC addresses in the candidate lists, we can now determine which candidate for MAC$_{current}$ is the best. We will concentrate on the MAC address MAC$_m$, which has the highest Score$_{m,current}$ value of any other MAC address. If MAC$_m$ is the last address in its list, we can append MAC$_{current}$ to the end of that list to associate it with that specific device. If MAC$_m$ is not at the end of its own list, we will compute Score$_{m,f}$ between MAC$_m$ and the address immediately following it in the same list, namely MAC$_f$. We can now determine which of MAC$_{current}$ and MAC$_f$ is the address that has to be inserted into the list after MAC$_m$, indicating a higher probability of being generated by the same device, by comparing Score$_{m,current}$ and Score$_{m,f}$. Specifically, if Score$_{m,f}$ has a higher value, it means that it is more likely that MAC$_m$ and MAC$_f$ belong to the same device, with respect to MAC$_m$ and MAC$_{current}$. Therefore, the process of finding the correct list for MAC$_{current}$ begins all over again, ignoring MAC$_m$. On the opposite, if the value of Score$_{m,current}$ is

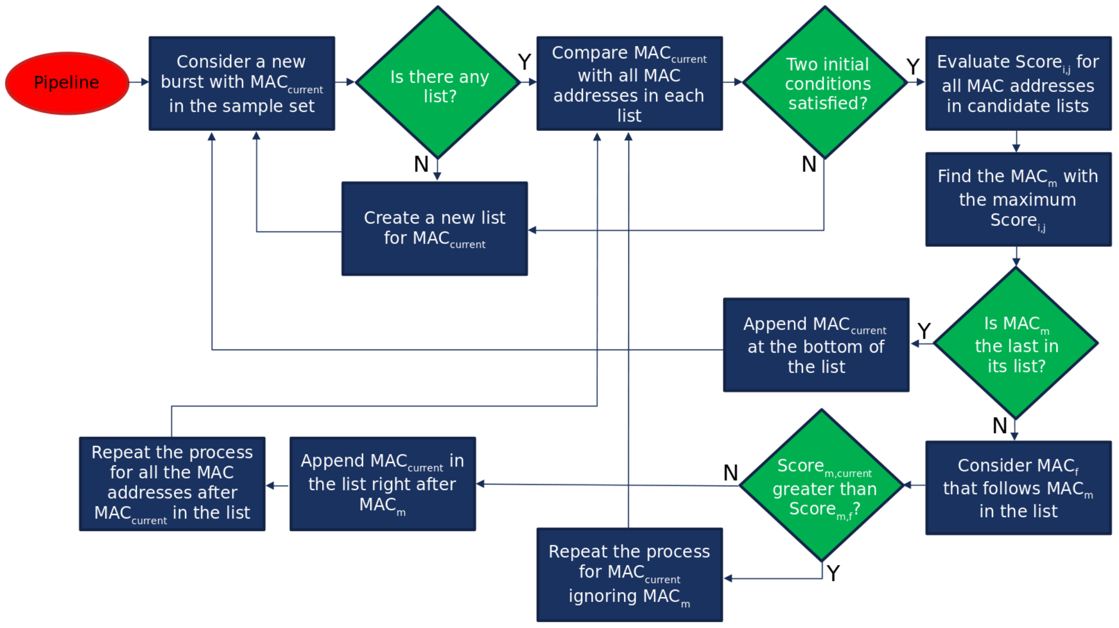

Pipeline
Consider a new burst with MAC$_{current}$ in the sample set
Is there any list?
Y
N
Create a new list for MAC$_{current}$
Compare MAC$_{current}$ with all MAC addresses in each list
Two initial conditions satisfied?
Y
N
Evaluate Score$_{i,j}$ for all MAC addresses in candidate lists
Find the MAC$_m$ with the maximum Score$_{i,j}$
Is MAC$_m$ the last in its list?
Y
Append MAC$_{current}$ at the bottom of the list
N
Consider MAC$_f$ that follows MAC$_m$ in the list
Score$_{m,current}$ greater than Score$_{m,f}$?
N
Append MAC$_{current}$ in the list right after MAC$_m$
Repeat the process for all the MAC addresses after MAC$_{current}$ in the list
Y
Repeat the process for MAC$_{current}$ ignoring MAC$_m$

Figure 5.11: Flowchart of the De-randomization algorithm.

greater than Score$_{m,f}$, the likelihood that the MAC$_m$ and MAC$_{current}$ belong to the same device is greater than the probability that the MAC$_m$ and MAC$_f$ do. As a result, MAC$_{current}$ is put right after MAC$_m$ in the list. Furthermore, the following MAC addresses in the list are no longer certain to be associated with the same device. As a consequence, the procedure is repeated for MAC$_f$ and all subsequent MAC addresses in the list. At the end of the recursive method, all the lists are those that have a higher possibility of having MAC addresses that can be directly attributed to the same device.

This complicated process generates a series of lists, each containing different random MAC addresses. All the MAC addresses in a list are very likely transmitted by the same device, which is attempting to hide its true MAC address by sending multiple random ones in its Probe Request frames. Finally, by counting the number of distinct lists created following the de-randomization algorithm, we can now count the number of distinct devices discovered on board. The whole set of lists prepared in this manner is then sent down the pipeline for our final examination step. We recognize that the de-randomization approach is quite complicated and that certain aspects of it may be difficult to comprehend on first reading. In an attempt to make it more understandable for the reader, its flowchart is shown in Fig. 5.11 and its pseudo-code is reported in Algorithm 7.

5) Parameters calibration and UDP sending: the final phase of our passenger prediction procedure is represented by this script. It takes the set of lists created by the de-randomization algorithm as input and counts the number of lists received to estimate the number of distinct devices on board. We are aware that some passengers on the bus may have their device's WiFi interface switched off, preventing us from counting them. In the same way, someone may be carrying more than one active device (e.g., a laptop, a tablet, or an additional smartphone). All of these actions cause bias in our calculations. As a result, the final count was subjected to an extra exponential filter with the smoothing value α. We can make the counting

Algorithm 7 De-randomizer pseudo-code.

```
1: while true do
2:    Read the new input from the pipeline.
3:    for all the records in the input sample do
4:       Identify the MAC address of the burst, $\text{MAC}_{current}$
5:       if There is no list existing then
6:          Create a new empty list and insert $\text{MAC}_{current}$ in it.
7:       else
8:          Verify the two initial conditions for every MAC in every list.
9:          if the two initial condition are not satisfied for any list then
10:            Create a new empty list and insert $\text{MAC}_{current}$ in it.
11:         else
12:            Save the lists that satisfy the two requirements as potential candidates
               for a match.
13:            Compute $\text{Score}_{i,j}$ between $\text{MAC}_{current}$ and all MAC addresses in the
               candidate lists.
14:            Select $\text{MAC}_m$ with the max score among all the $\text{Score}_{i,j}$ computed.
15:            if $\text{MAC}_m$ is the last one in its own list then
16:               Append $\text{MAC}_{current}$ at the bottom of that list.
17:            else
18:               Select $\text{MAC}_f$ which follows $\text{MAC}_m$ in the list.
19:               Compute $\text{Score}_{m,f}$ between $\text{MAC}_m$ and $\text{MAC}_f$.
20:               if $\text{Score}_{m,current} > \text{Score}_{m,f}$ then
21:                  Recursive call for $\text{MAC}_{current}$ ignoring $\text{MAC}_m$.
22:               else
23:                  Append $\text{MAC}_{current}$ in the list right after $\text{MAC}_m$.
24:                  Recursive call for all the MAC after $\text{MAC}_{current}$ in the list.
25:               end if
26:            end if
27:         end if
28:      end if
29:   end for
30:   Send the device lists down the pipeline.
31: end while
```

curve appropriately reflect the real number of passengers by properly adjusting α. To this extent, all of the parameters discussed in this section were related to the actual number of people on board, which was manually verified. After a long period of trial and error, we arrived at a fair estimate with a small margin of error.

Finally, the number of individuals on board is estimated in this manner, along with the detection timestamp, and transmitted with a UDP packet. The packet is sent across the bus network to provide GTT with the service they demand, as well as over our LTE dongle to a Grafana dashboard for an enhanced visualization service.

5.5 Discussion of privacy issues

Due to the general nature of our experiment, we believe it is necessary to hold a discussion about the potential privacy problems that may arise and how we dealt with them. It is logical to assume that a system that captures packets sent by a smartphone will not be trusted by the user, who may be concerned about disclosing private information from his or her phone to other parties. Even if it is possible to know that the information may be beneficial for a public service, such as the real-time counting of bus passengers in this situation, this will not help to alleviate the problem. Indeed, when data broadcast by cellphones is analyzed by a system, this may be considered a people-monitoring mechanism, which has the potential to cause important privacy dilemmas. In the Snowden case in 2013, the American National Security Agency (NSA) was discovered to monitor people's movements by gathering and analyzing the MAC addresses from smartphone communications in a manner that is very similar to what we are doing during this study. Despite the fact that the NSA claimed it was all done for national security reasons in the event of a terrorist attack, the entire debate reached its conclusion on September 2, 2020, when an American federal court ruled that the NSA's mass surveillance program was illegal and possibly unconstitutional. In this section, we will outline the reasons why we believe that our passenger counting system does not violate the privacy of its users. Even if the approach we utilize is similar to the one that was used by the NSA before 2013, we will see why the final outcome is radically different.

The first important consideration is that we will always execute a passive scanning procedure throughout this project. We made no attempt to decipher any data, nor did we take any active steps that would stimulate or influence regular network activity at any point throughout the investigation. The only messages we are interested in capturing are Probe Request frames, which are transmitted by any smartphone on its own initiative and without the user's involvement. Furthermore, as we saw in the previous sections, Probe Requests are management frames that are used at the beginning of the association process to establish a connection with an AP. As a consequence, it is vital to note that, because of their very nature, they do not include any personal information about the user, but rather merely information about the phone's capabilities and its MAC address. In spite of this, the MAC address of the transmitter phone is typically regarded as potentially sensitive information because it might be correlated with a specific individual. As a result, the legitimacy of packet sniffing activity is frequently determined by the type of analysis that is conducted on the data after it has been captured.

The second point of this discussion focuses specifically on the analysis that was conducted on the packets that were acquired. The fact that we focused exclusively on MAC addresses means that we are only able to collect information "about what" and not "about who." As explained previously, each device's network card has its own MAC address, which serves as a unique identification number for that device's interface. Applying our analysis, we were able to distinguish between different cellphone models on the bus at any given time. Nonetheless, this knowledge does not reflect any personally identifiable information, as we are unable to link the detection of a phone model to the identity of its owner. This is perhaps the most significant distinction between our investigation and the Snowden case, which was previously mentioned. The only way to link a phone's MAC address to a specific person is to have information about who purchased that particular smartphone. As a result, it is evident that none of the actors participating in this research has the authorization to possess or request this type of information from anybody else. On the contrary, a government body might easily access the records for the annual sales of phones, where each model is connected to its MAC address as well as the name of the individual who purchased it.

The final part of this debate will deal with the management of the information gathered throughout this research. As was frequently mentioned in the preceding section, our system performs an analysis on the fly. Upon completion of the entire software procedure, the sole output delivered by our sensor is an integer value corresponding to the number of people on board at any given point in time. Furthermore, information about Probe Request frames is not saved on our sensor, but is instead collected, evaluated, and then discarded at the conclusion of the procedure. Evidence of this is the fact that the file produced during the capture script is overwritten each time the capture process is restarted (i.e., every time the bus is turned off). Consequently, we are unable to retrieve any of the information that was acquired through our system. In light of this, we feel that our research does not raise any ethical concerns and will not in any way compromise the rights of the user.

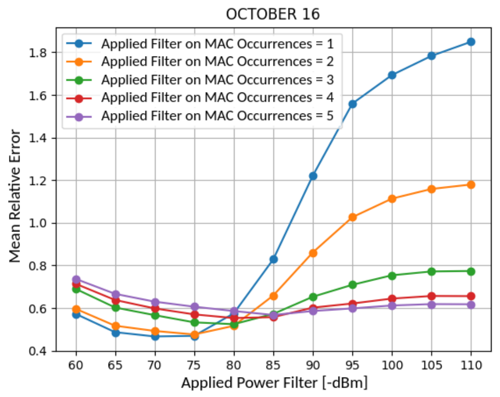
5.6 Results

We needed to periodically compare the results of the counting process with a ground truth comparison in order to appropriately establish all of the parameters that were critical to the effectiveness of our system. Ground truth is a term used in remote sensing systems to refer to information gathered on site that is used to calibrate acquired data and to aid in the interpretation and analysis of what is being detected. Due to the fact that ground truth is normally performed on site, we organized some manual counting sessions to determine the precise number of people on board the GTT bus 33E while our system was in operation. The only way for us to correctly configure the system in accordance with the capturing environment is to compare the manual counting results with the values observed by the sensor at the same time.

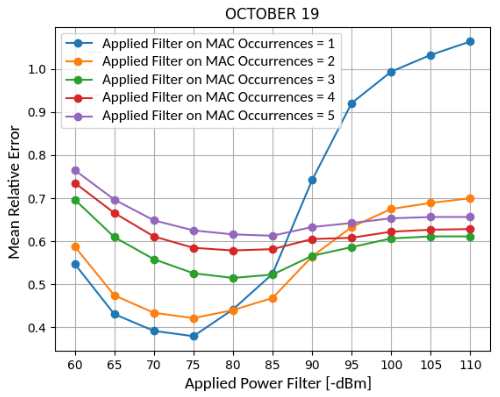
Unfortunately, due to the national lockdown enforced by the government, our manual counting operation was abruptly halted. We also noted a general decrease in the use of public transportation in the months after the conclusion of the lockdown and the relaxation of the limitations that had been imposed. Many commuters in Turin, as well as in most other Italian cities, may have opted to decrease their use of public transportation since they perceived it as a potentially congested area after the lockdown and restrictions, or even to work from home if they had the opportunity. As a result of these actions, the number of people using public transit was significantly lower than usual, and we were unable to continue our manual counting effort in the following months as planned. As a consequence, we may depend on four manual counting sessions that, in our opinion, are reasonable representations of the regular affluence of a public GTT bus service. We have ground truth counting on the 16^{th} of October 2020, the 19^{th} of October 2020, the 30^{th} of October 2020, and the 2^{nd} of November 2020. We are aware that this is a limited sample size for ground truth, yet we were able to fine-tune our parameters in accordance with the data. Furthermore, GTT decided in recent months to continue our tests on another bus that would be fitted with cameras on the top of each of its entrances. They are able to trigger an automatic counting system based on image recognition, known as AESYS. By comparing our APCS data with the data from AESYS, we will be able to obtain a reliable ground truth comparison and complete the setup of this currently ongoing project.

5.6.1 Tuning of system parameters

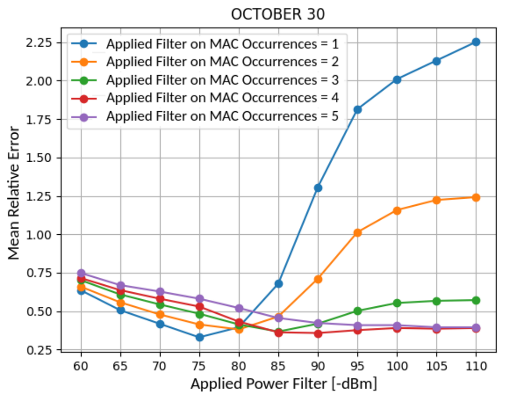
We were able to fine-tune the system settings previously stated by comparing the number of passengers evaluated by our sensor with the number indicated by ground truth counting. To be more specific, we calculated the difference between the actual ground truth counts at each bus stop and the corresponding values from our software system. We calculated the mean relative error for a given set of parameters by averaging all of the errors over the period of the whole manual counting session. As a result of our efforts, the power level threshold P_l and the occurrence level threshold O_l used in the analysis for the power and occurrence filters were both tuned. The mean relative errors for different combinations of power and occurrence thresholds are shown in Fig. 5.12, which includes all of our



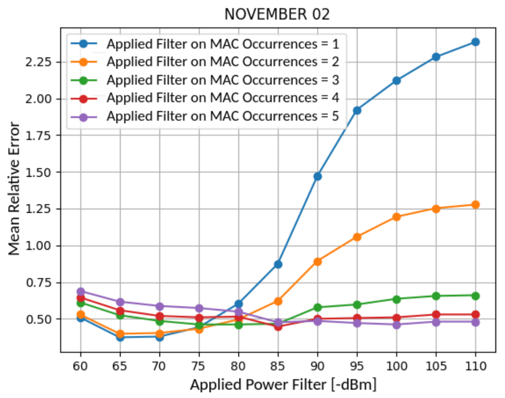
(a)



(b)



(c)

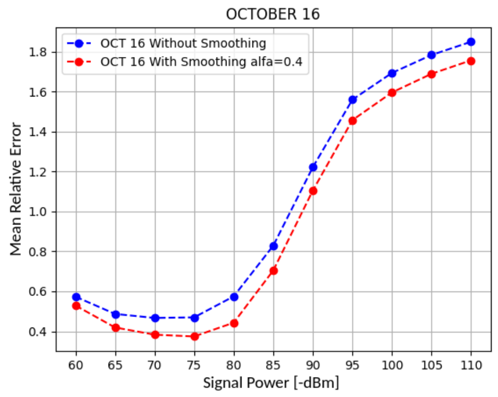


(d)

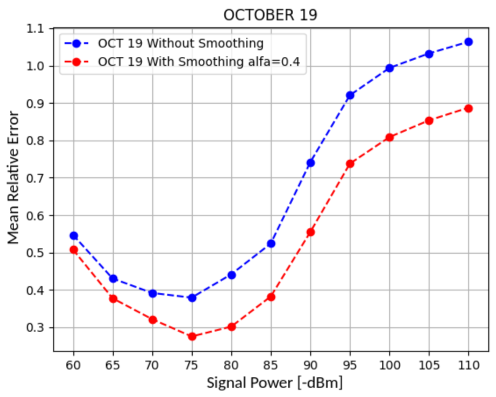
Figure 5.12: Mean relative errors for the power filter and the MAC occurrence filter. (a) October 16. (b) October 19. (c) October 30. (d) November 2.

manual counting days. By looking at the plots, it's evident that using the values of $O_l = 1$ and $P_l = -75$ dBm as thresholds, we can create a filter that reduces the mean relative error in almost all our comparisons.

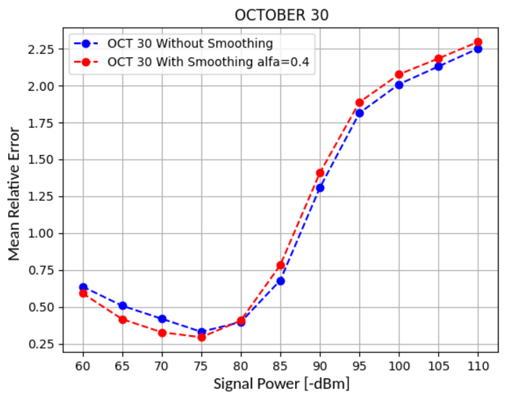
Besides that, we applied an exponential smoothing filter with a smoothing factor of α in order to ensure that our counting curves were as close as possible to the real ground truth values. In Fig. 5.13, we can see the favorable effect of the smoothing filter that has been applied, which has resulted in a further reduction of the minimum mean relative error for all the counting sessions. In particular, we used the smoothing factor $\alpha = 0.4$ in our system to achieve the desired results. More ground truth comparisons will be available as the project progresses in the future, allowing us to validate or enhance the parameters of our filters.



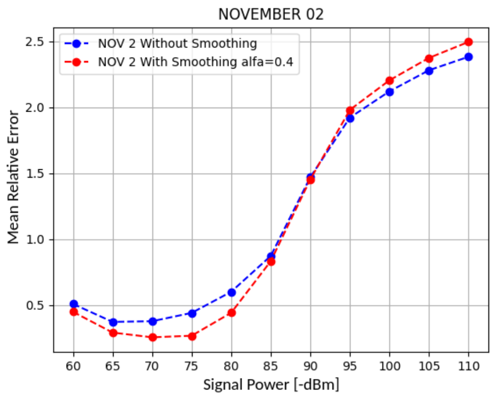
(a)



(b)



(c)



(d)

Figure 5.13: Mean relative errors for the exponential smoothing filter. (a) October 16. (b) October 19. (c) October 30. (d) November 2.

5.6.2 System performance evaluation

With the help of the appropriate parameters that we discussed in the previous section, we are now able to accurately estimate the number of passengers actually on board. The number of passengers on the bus was divided into three groups in order to allow for a more accurate evaluation of our system: a "green zone" implies a low number of passengers on board, precisely less than 16 people (i.e., less than 20% of the total bus volume); a number of passengers on board ranging from 16 to 28 (i.e., from 20% to almost 40% of the max bus capacity) will be seen in the "yellow zone," while the "red zone" with 28 or more passengers (i.e., more than 40% of overall capacity) implies a situation in which interpersonal distances on the bus cannot be guaranteed anymore. The passenger numbers used to define the three capacity models are just indicative and do not correspond to the actual restrictions

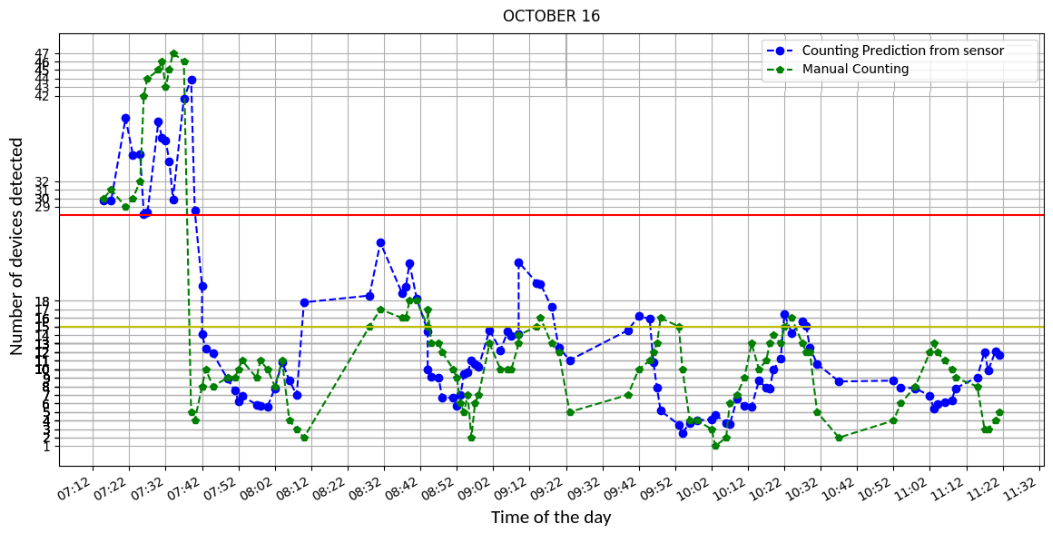


Figure 5.14: Performance evaluation for the counting session on October 16.

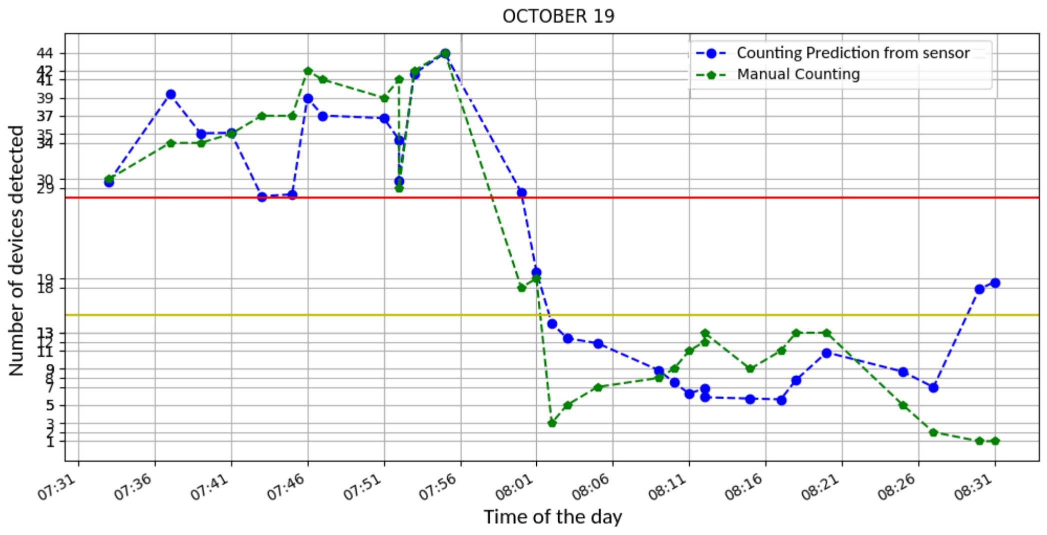


Figure 5.15: Performance evaluation for the counting session on October 19.

imposed by governments as the maximum capacity on public transport. To be more specific, we identified the thresholds that were the most closely aligned with the ground truth values that we observed throughout our manual counting sessions.

Generally, our findings indicate that we are able to estimate the actual number of passengers on board with a high degree of accuracy for all of the ground truth scenarios that we have considered. The comparison between our counting forecast and the manual counting that took place on the 16th of October is seen in Fig. 5.14. When we estimate the overall relative error for this scenario, we can predict an

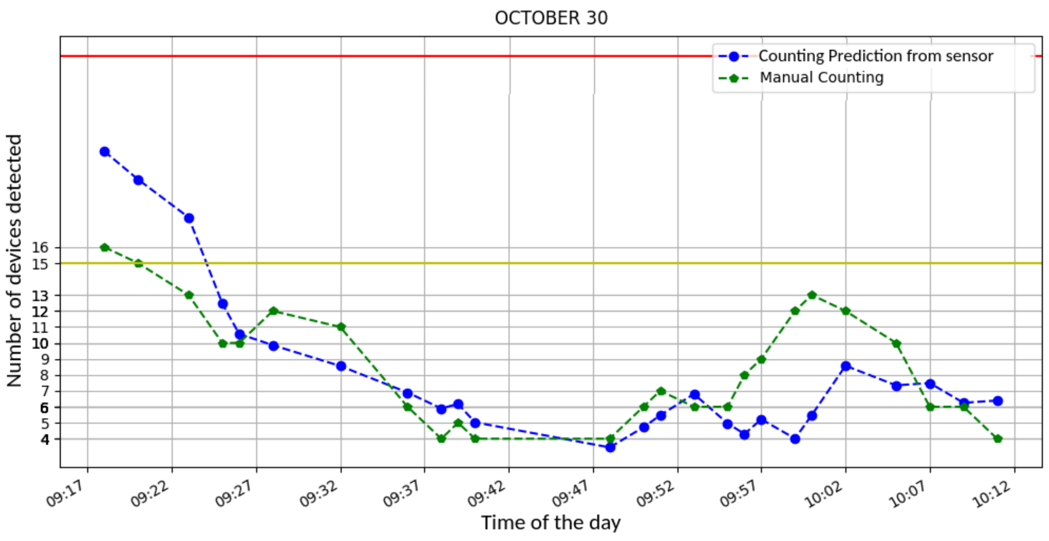


Figure 5.16: Performance evaluation for the counting session on October 30.

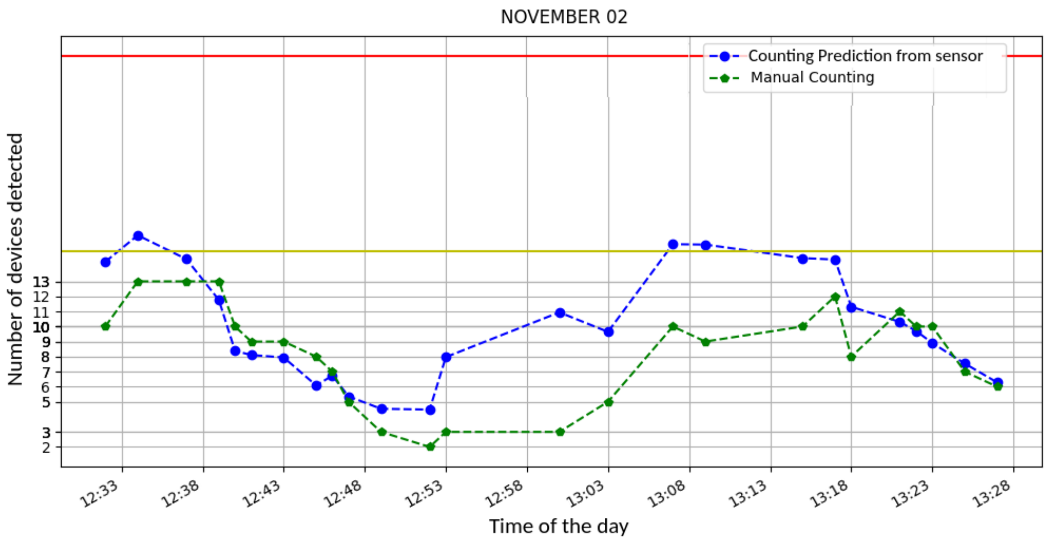


Figure 5.17: Performance evaluation for the counting session on November 2.

accuracy level of 85% for the entire day. The difference between the ground truth and the manual counting on the 19^{th} of October is depicted in Fig. 5.15. In this circumstance, the total accuracy computed is extraordinarily high and reaches 90% throughout the whole measurement. Compared to the other two measures, the manual counting activities on the 30^{th} of October and the 2^{nd} of November revealed that there were fewer individuals on the bus on average over the two sessions. This is due to an increase in the number of daily COVID-19 cases, which resulted in the city of Turin being forced on lockdown in the weeks after our studies. Nevertheless,

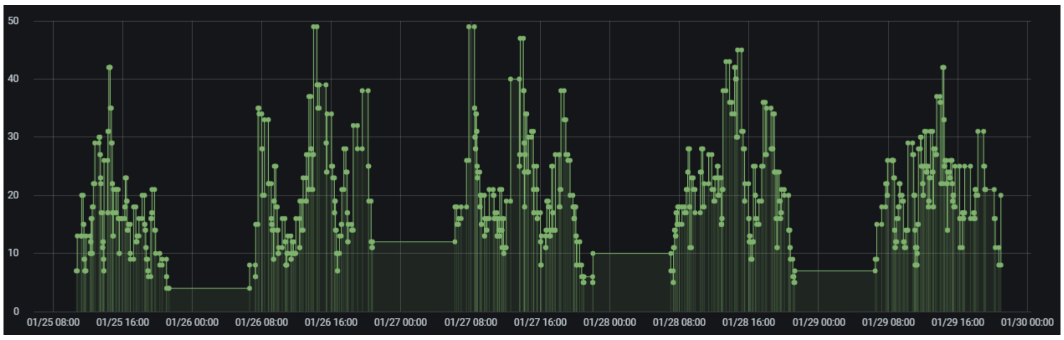


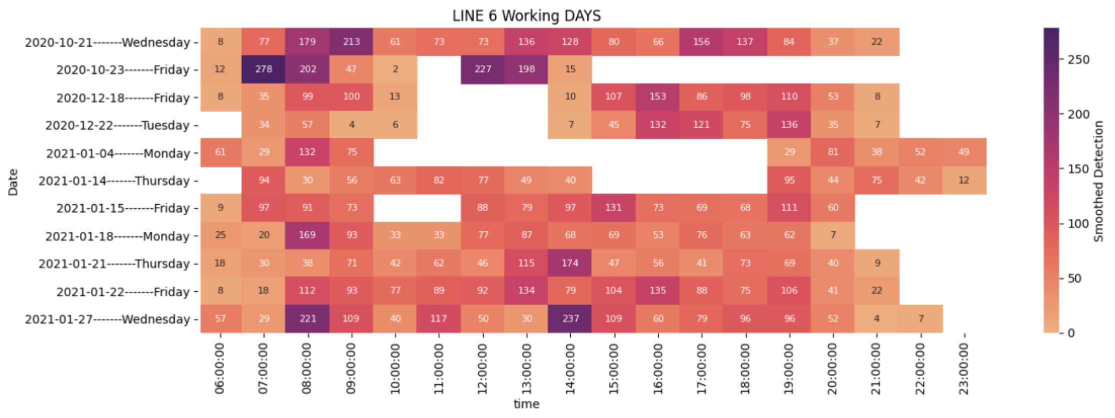
Figure 5.18: Real time counting visualized in the Grafana dashboard.

they both showed a very good estimation of the situation on the bus. In particular, Fig. 5.16 depicts the differences between the ground truth determined on the 30th of October with an overall accuracy of 91%, whereas Fig. 5.17 depicts the differences between the ground truth determined on the 2nd of November with an overall accuracy of 88% for this scenario.

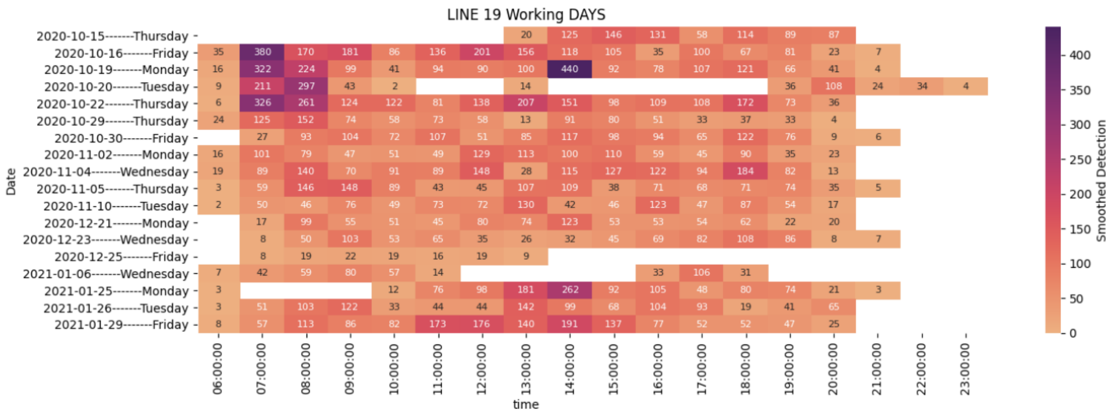
5.6.3 Online dashboard and patterns recognition

The ultimate output of our system is the information on the real number of people on board and the time and date of the detection. That data is encapsulated in a UDP packet and transferred to GTT over the bus network, as well as to a Polito server via an LTE dongle. Thus, as seen in Fig. 5.18, we were able to construct a Grafana dashboard for the purpose of seeing the number of passengers in real time. In the figure above, it is possible to observe five consecutive days of recordings from the GTT bus 33E. The green dots show the measurements of the number of passengers, while the horizontal bars represent the periods during which the bus was not in service.

According to what we previously said, we were only able to plan a limited number of manual counting sessions for the purpose of obtaining our ground truth comparisons. In spite of this, our sensor remained operational and functioning at all times when bus 33E was in service for the duration of the project. Even if those metrics cannot be validated by comparison with a ground truth counting procedure, we can nevertheless use them for the purpose of gathering statistical information about the traffic in Turin. Fig. 5.19 illustrates two heatmaps, where each cell represents an hour of counting and each line represents a day of the week. To be more specific, Fig. 5.19a is associated with bus line number 6, whereas Fig. 5.19b is related to bus line number 19 (i.e., the two lines served the most by the bus 33E). The color of each cell is determined by the total number of bus passengers that were detected during that hour's entire duration (the actual number of passengers is also shown in every cell). The darker the color of an hour slot, the greater the number of



(a)



(b)

Figure 5.19: Hour-based heatmap for pattern recognition of bus passengers. (a) Working days for the bus line 6. (b) Working days for the bus line 19.

passengers we counted during that particular hour. The white slots show a period during which our sensor was not operational due to the fact that bus 33E was not in service. By analyzing the two heatmaps, we may see certain repeated patterns that are consistent with our previous observations of the regular peak traffic in Turin and the major events that occurred over the corresponding period of time. First and foremost, it is clear that the restrictions imposed by the government on the Piemonte region from the 26th of October 2020 until the beginning of January 2020 as a result of the COVID-19 epidemic had a remarkable impact. It is evident that there are a limited number of dark slots throughout the course of a day during all of this time. The time periods preceding and after the limitations follow a different pattern than the rest of the measurements. On several days, we can clearly distinguish the peak hour traffic in the morning (from 7am to 9am), the afternoon rush hour (from 12am to 2pm), and a slight increase in the number of passengers during the evening peak period (from 5pm to 7pm). Because the two heatmaps are based on weekdays, when the vast majority of commuters travel from their homes to their workplaces and vice versa, we can easily recognize the patterns that have been detected as a very reasonable representation of the actual traffic in Turin.

5.7 Conclusions

For their daily commutes, a significant portion of the metropolitan population chooses to take advantage of public transportation. These individuals rely on adequate infrastructure and services provided by local transportation authorities, which require real-time data on bus passengers collected at each stop in order to shape bus frequencies in the right manner. Furthermore, as a result of the COVID-19 pandemic, governments imposed capacity limitations on the number of people permitted to enter public transportation vehicles, making the real-time counting of persons on board a mandatory requirement for public transportation.

As part of this project, we developed an Automatic Passenger Counting System (APCS) that could be used on a public bus in order to meet the needs of GTT, the Turin public transportation agency. In order to complete this work, we used a Raspberry Pi module that was coupled with a special antenna capable of capturing WiFi frames transferred over the air. We are specifically interested in capturing Probe Request frames, which are packets broadcast by smartphones in order to scan the wireless medium in search of a potential Access Point to connect to. Every phone sends Probe Request frames in a periodic and spontaneous manner, without the need for a user to initiate the process. Every Probe Request has a field dedicated to the MAC address, which is a unique identifier connected to the phone's WiFi network card and is used to identify the device. Because the MAC address reflects the unique number assigned to a WiFi card, it may be used to identify a certain phone and, with some further information, a specific person. To protect the confidentiality of this information, several phone manufacturers have developed MAC randomization algorithms that prevent this information from being disclosed in Probe Request frames. On the other hand, MAC de-randomizer methods are currently being developed in a number of research projects, and we used one of them for this project.

In this context, it is critical to have a discussion about ethical and privacy concerns. We are certain that the technique we have provided does not compromise the privacy of users in any way. We can make this claim because we are always doing a passive scanning of the WiFi channel, capturing only management frames that are broadcast by phones on their own initiative. Furthermore, since there is no way to link the information of a MAC address to a single person, our experiments collect only information about phones rather than information about individuals' identities. Additionally, the data acquired by our Raspberry Pi sensor is examined in real time, and the information about a phone's MAC address is destroyed immediately after the examination.

Because of COVID-19 limitations and the resulting general lockdown, we were unable to organize a large number of manual counting sessions on the GTT bus that was specifically devoted to our project. Only four manual counting measures were used as a ground truth evaluation, which was necessary for validating the output of our system and fine-tuning all of the parameters that compose our system's architecture. Although we only had a small number of ground truth measurements, we were still able to appropriately establish all of the variables in our process, and we were able to get a count of people on board that was reasonably close to the real value with an acceptable degree of precision. In particular, we saw a total accuracy level greater than 80% in every validation procedure, demonstrating an impressive performance in light of the difficulties encountered as a result of the COVID-19 lockdown. The data that had not been validated was also utilized for pattern examination, thanks to a Grafana dashboard that was specifically designed for this purpose. We were able to identify the key traffic peak hours in Turin by developing heatmaps of the number of passengers on the GTT bus over the course of many days.

In the future, we want to expand our manual counting efforts, as well as increase our ground truth set, in order to provide a more comprehensive validation procedure. Also on the horizon is the installation of our sensor on a bus that will be equipped with cameras mounted on the top of the door entrances that will be used for passenger counting via image recognition. By comparing our results with the data provided by the cameras, we will be able to obtain a far greater degree of accuracy for our APCS.

Chapter 6

Content Sharing among Pedestrians in a Micro Cloud Environment

In the context of urban mobility, studies that concentrate on pedestrian trace simulations might generate scenarios that are particularly important. Indeed, pedestrians account for a significant portion of the overall number of commuters, owing to the fact that they can interact with public transportation for part of their travels. Applications of content sharing in a pedestrian environment can be immensely useful for both the final customer and the companies that provide the service. In particular, micro cloud concepts, which are commonly employed in vehicular simulations for floating car data investigations, might be easily transferred to the pedestrian environment, resulting in the same great outcomes as in the vehicular context.
In this chapter, we will discuss the architecture of a pedestrian-based micro cloud simulation for a content-sharing application that makes use of people traces in a metro station. The application we built is centered on the sharing between users of content items that are relevant to the surrounding environment.

6.1 Research motivation

UNICEF data confirms the current exponential growth of the urbanization process. According to [109], today more than half of the Earth's population lives in big cities and this rate is expected to increase up to more than 70% by the half of this century. The continuous growth of the urban population requires a reinforcement of infrastructure systems to cover the growing demand. Nevertheless, centralized infrastructure-based methods suffer from high up-front and maintenance costs.

This explains the popularity of distributed approaches, which have the potential of turning the large demand into an advantage. Distributed systems applied to models of sharing economies can benefit from large user demands by spreading content items in wider areas. Nowadays, an increasing number of applications successfully implement sharing economy methods through gamification and app engagement techniques: the more users are willing to share content items with other users, the more rewards they may obtain in terms of price discounts or even cashback. Waze is an example of an app where users share information on online platforms and receive a reward proportional to the amount of items shared.

Much in the same vein, floating data is viewed as an excellent solution to spread information in distributed environments. In particular, micro cloud-based applications exploiting a sharing approach represent prime building blocks for the next generation of Intelligent Transport System (ITS), as detailed in the next section of literature review. This trend is embraced not only by vehicular communication models, but also by other kinds of hybrid mobility, including inter-pedestrian communication environments [110, 111, 112]. Mobility models are now available for selected scenarios and even allow first-edge computing approached in the presence of pedestrians [113, 114].

6.2 Main contributions

In this study, we present a content sharing application suited for a pedestrian-based micro cloud environment. The main contribution of this work can be found in our conference paper [7], which is currently under review. Our work improves the existing literature by several aspects, namely:

- by applying the micro cloud concept, widely used in a vehicular context, to a pedestrian environment;
- by proposing a content item sharing application with rewards suited to the proposed use case;
- by developing scenarios with and without data sharing, with data sharing and congestion control and by comparing them with a benchmark scenario.

The chapter is organized as follows: Section 6.3 is dedicated to related works in literature, while Section 6.4 presents the use case we focused on for our application. In Section 6.5 the methodology used for instantiating the content creation procedure and the content sharing methods are described. The same section also includes a description of the considered scenarios. Results are presented in Section 6.6, while conclusions and future work are in Section 6.7.

6.3 Related Work

Multi-access edge computing [115, 116, 117] is one of the key network architecture concepts in 5G networks. The main idea is to bring computational, storage, and communication infrastructure closer to end users. As communication with back-end cloud services always comes with added network delay problems, having popular or frequently used data cached in the edge servers helps reducing latency [118]. Generally, edge-supported services can be classified as computational offloading, content delivery, aggregation, local connectivity, content scaling, and augmentation [119].

Performance gains from an edge computing architecture are possible only when edge servers are densely deployed. In [120], the authors found out that the location of deployed edge servers is very critical for supporting services, e.g., vehicle-to-everything services, which demand high link capacity.

The lack of edge computing infrastructure in the vicinity can be complemented by virtual edge servers [121]. The idea is to virtualize the physical edge computing infrastructure using on-board computational and storage resources of the cars, whose communication capabilities may be leveraged to form small clusters called *vehicular micro clouds*. These clusters can be formed based on parameters like driving direction, location, speed, etc [122, 123]. Vehicular micro clouds can then share their rich resource pool to provide edge computing services.

Recently, protocols designed for vehicular edge computing [124] have been utilized and studied for content dissemination via pedestrian mobility [114]. In this study, we used pedestrian mobility traces obtained using the ONE simulator [113] and focused on making content items available within certain geographic regions through pedestrians, optimizing for network and storage usage on mobile devices. We followed the concepts of vehicular micro clouds to study content sharing via pedestrians using opportunistic communication in a subway station scenario generated in [125].

6.4 Case Study and System Description

The aim of the present work is to analyze the performance of a content sharing system in a distributed environment. In particular, we aim to reproduce an application for data exchange between pedestrians. The simplest way to represent such a model is by means of simulation with a realistic pedestrian mobility model input to the system. It is extremely difficult to retrieve online a pedestrian trace covering a wide area with a good level of realism. For this reason, the choice of the mobility environment usually affects the use case selection.

The mobility environment chosen for our scope was taken from [125], where the authors aimed to reproduce a realistic mobility scenario for evaluation of opportunistic wireless communication. In order to replicate an ad-hoc pedestrian network with a strong resemblance to reality, they calibrated and validated a mobility model

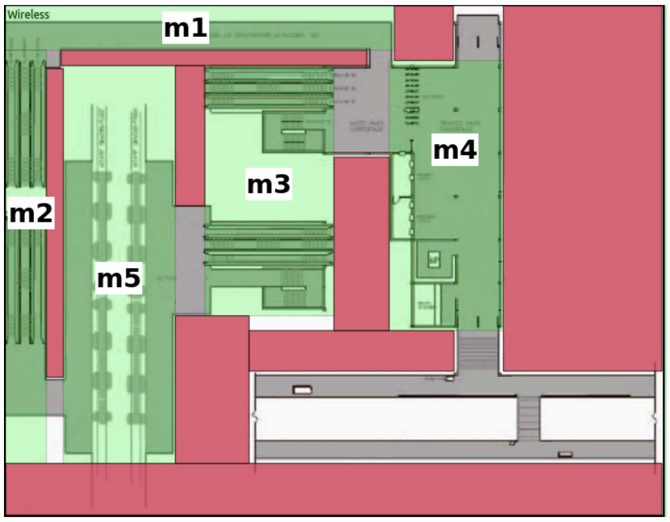


Figure 6.1: Ostermalm subway station map with micro clouds.

by measuring and monitoring pedestrian behaviour, both in controlled settings and with real pedestrian crowds in a real environment. The final outcome is a complete set of synthetic mobility traces from a subway station in the Ostermalm area of Stockholm.

We believe that the selected use case perfectly suites the application we want to target. The coffee shop and the stores in the station main hall can conceivably represent the places where content is generated, in the form of advertisements and special offers on a mobile app. Users are then enticed to spread such information through the subway station by receiving a reward (cashback or discounts) proportional to the quantity of data shared with other users. Such an app can also be leveraged by the transportation authority so that passengers roaming the station can be used as a vehicle to disseminate useful information on train schedules and service disruptions. Such information can either be generated on the fly by users waiting for the train arrival on the platform area or, if information is previously known (e.g., trains being canceled due to planned strikes or line maintenance), it can be provided by the local transportation authority to shop owners for dissemination through the app. Data dissemination services can also be rewarded with special discounts on transportation ticket prices.

In view of the mechanism described above, we propose and evaluate a data dissemination application in a micro cloud distributed environment. Micro clouds are the optimum choice for floating data contexts that do not rely on infrastructure. Being infrastructure-independent significantly reduces costs, but it shifts the burden to the users. In particular, we assume that every user carries a smartphone that can perform indoor localization correctly (beside being an actively researched topic [126] [127], indoor localization is nowadays supported by most smartphones [128]) and, through the content sharing app, can recognize the micro cloud it is crossing.

The map of the subway station used in [125], as well as in our study, is depicted in Fig. 6.1, where green zones highlight micro cloud areas of interest. In particular, the micro cloud areas are the north corridor (m1), the west escalator (m2), the east escalator (m3), the entry hall (m4) and the platforms area (m5). Red regions represent walls hindering radio communication.

Data is assumed to be exchanged through packets broadcast via WiFi Direct. Packets can be of two kinds: beacons and data packets. Through beacons, sent at a frequency of 10 Hz, hosts can share basic information such as:

- Transmission timestamp;
- Host IP address;
- Host motion information: current host position, heading, speed and the current micro cloud the host is crossing (if any);
- List of owned items of data content;
- List of creation times of items of data content;
- List of missing items of data content.

Data packets are simple packets containing the data content ID, a random byte-stream sequence mimicking 2 KB worth of data content and its corresponding creation time.

Through beacons and their own localization, hosts can collect data from neighbors and are capable of recording motion and content details in basic data structures. In particular, every host leverages two internal data structures: the *Host Storage*, in charge of collecting motion data (i.e., timestamp, host IP, position, speed, heading, micro cloud) about itself and its neighbors and the *Content Manager*, collecting information regarding content owned by hosts (i.e., timestamp, host IP, micro cloud, owned data list, creation times list and missing data list).

6.5 Content creation and sharing procedures

The model presented in this study aims to reproduce a realistic indoor pedestrian mobility environment with the sharing of contextual information in a subway station when M micro clouds are present. Therefore, content items created in each scenario are supposed to be meaningful for the micro cloud they belongs to, e.g., an early/late train departure information will be linked to the platforms area, while the announcement of a new item in the store or in the coffee shop is an example of data content likely created in the main hall micro cloud. To model the creation of micro cloud-relevant content, a generation process was defined as follows: as soon as a host finds itself in a micro cloud, and as long as it remains there, it generates a piece of data content according to a Poisson process with a rate λ. The system allows for a total of N_c different content items, out of which up to $N_m = N_c/M$ can be associated with a single micro cloud. Every item has a limited lifetime of T_l, after which it is deleted from the host storage.

Hosts inside a specific micro cloud can own every kind of content, but can only create content items for the micro cloud where they are transiting. As an example, hosts in the north corridor cannot have information on train delays in a direct way. Indeed, this information has to be shared by users coming from train platforms. Thus, no content on train delays can be generated outside the platforms.
Multiple scenarios were analyzed in order to compare the performance of the model at different levels of complexity.

Algorithm 8 Content sharing procedure.

```
Data: On beacon received
 1: Update the Host Storage with sender's motion info
 2: Update the Content Manager with sender's owned and missing data lists
 3: for every item owned by the sender do
 4:    if there is a content item missing by host then
 5:       Add that item in host's missing list
 6:    end if
 7: end for
 8: for every item in the sender's missing list do
 9:    if there are items owned by the host then
10:       if host is entering in a new micro cloud or host is leaving the current one
          then
11:          Select the owned item with the longest remaining lifetime belonging to
             a micro cloud different from the current one
12:       else
13:          Select the owned item with the longest remaining lifetime belonging to
             the current micro cloud
14:       end if
15:    end if
16: end for
17: if there is a content item to send then
18:    if my collision rate is below $C_r$ then
19:       Send the data packet
20:    end if
21: end if
```

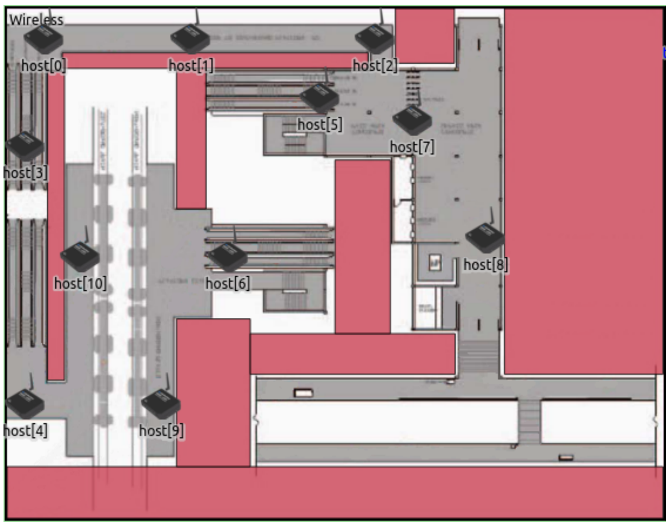


Figure 6.2: Access Points locations in map environment.

6.5.1 Baseline scenario

In the first scenario, used for benchmark purposes, hosts are not allowed to share data content, thus only beacons and not data packets are exchanged. There, hosts only read broadcast beacons and they update their own Host Storage and Content Manager with neighbors' motion and content information. Upon a beacon reception, the receiving host looks up the list of advertised content items sent by the other host and inserts any content items that are missing from their own missing data list. As a consequence, in this baseline scenario, creating content is the only way to own a specific content item. The procedure adopted by hosts at this stage of the model is reported in Algorithm 8 from line 1 up to line 7.

6.5.2 Data sharing scenario

In a second scenario, hosts share content items via data packets. When a beacon is received with some items marked as missing by the sender, the recipient host checks whether it owns them or not. If so, the item with the most recent creation time is chosen, i.e., the one that has the longest remaining lifetime. The selected item will be broadcast encapsulated in a data packet and will be received by all the sender's neighbors. Hosts correctly receive these data packets and add them to their Content Manager. The procedure described in this subsection is reported in Algorithm 8, except for lines 10 to 14 and without the "if" condition on line 18.

6.5.3 Data sharing scenario with congestion control

The purpose of the third scenario is to reduce the number of channel collisions. To this end, unlike the second scenario, content items are broadcast only if the rate of collisions on the channel detected by the host is below a given threshold C_r. Furthermore, in this scenario, a procedure for spreading content items through different micro clouds is introduced. Based on its own motion information, when a host has just entered or is about to exit a micro cloud region, it is allowed to share only data belonging to other micro clouds. Conversely, if motion data suggests that the host is lingering in the same micro cloud region, the host is forced to share content items belonging to the current micro cloud. In such a way, the sharing of content throughout different regions of the map is promoted. The complete Algorithm 8 represents the whole procedure at this stage of the model.

6.5.4 Access Points benchmark scenario

Eventually, a benchmark scenario with Access Points (APs) is also analyzed. In this scenario, 11 APs were located in fixed positions on the map , each one in radio visibility of one or more other APs. The resulting model is depicted in Fig. 6.2. In this scenario, APs are the only ones entitled to create and broadcast content items, while hosts can only share motion information through their beacons. This last scenario represents an infrastructure-based model, where items are managed by a few entities (i.e., the APs) with a different sharing model than the hosts. This scenario has been developed as a benchmark model, where better performance is offset by high equipment costs.

Table 6.1: Simulation parameters.

Description	**Value**
Simulation time limit	600 s
Number of micro clouds M	5
Total number of content items N_c	50
Content item lifetime T_l	50 s
Content item size	2 KB
Beacon frequency	10 Hz
Content items generation rate λ	0.2 s^{-1}
Channel collision threshold C_r	100 collisions/s
Host transmission bit-rate	1 Mbps

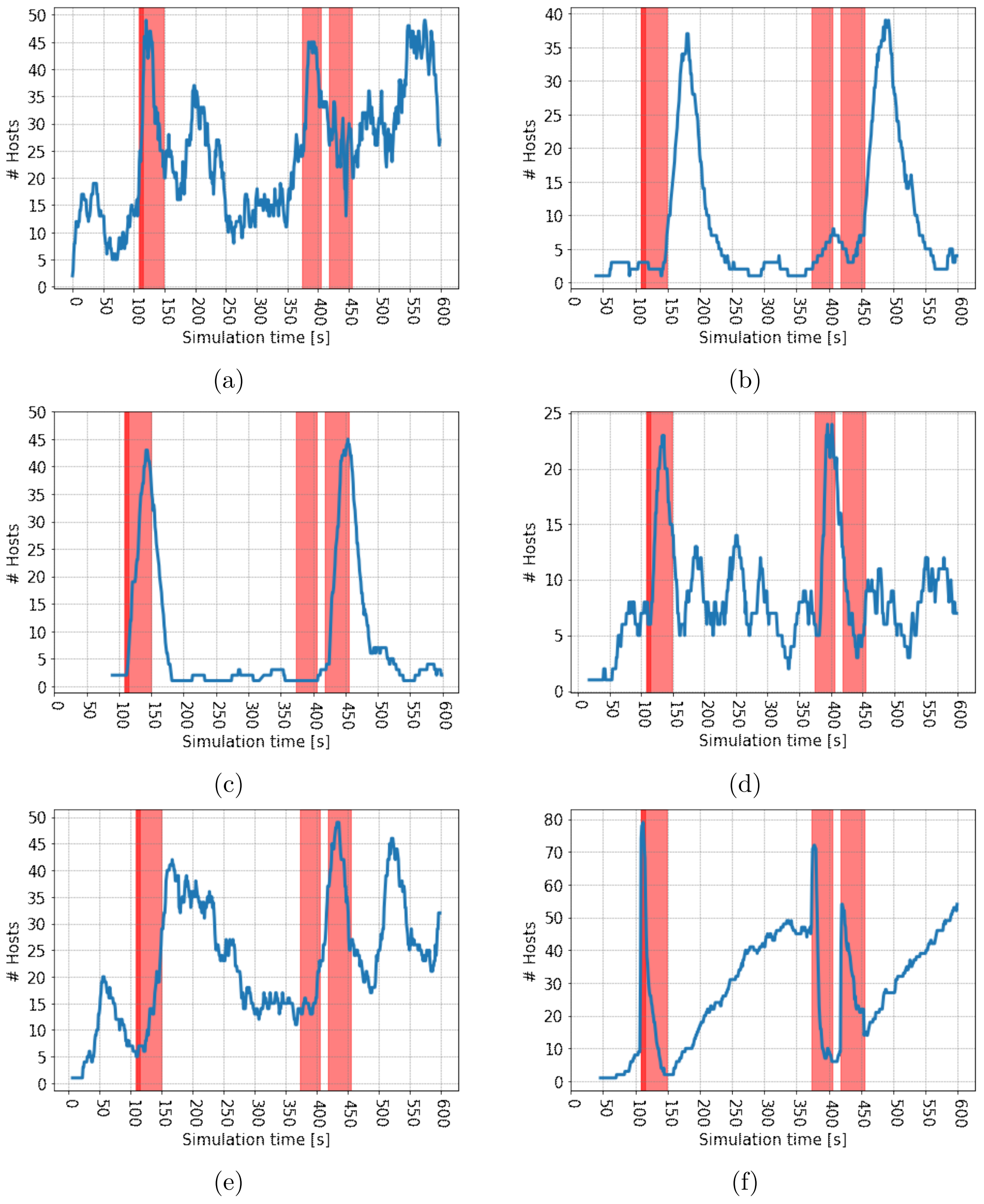


Figure 6.3: Number of hosts throughout simulation for every micro cloud area. Red bars represent train arrivals. (a) Hosts outside micro clouds. (b) Hosts in the north corridor. (c) Hosts in the west escalator. (d) Hosts in the east escalator. (e) Hosts in the main hall. (f) Hosts in the platforms area.

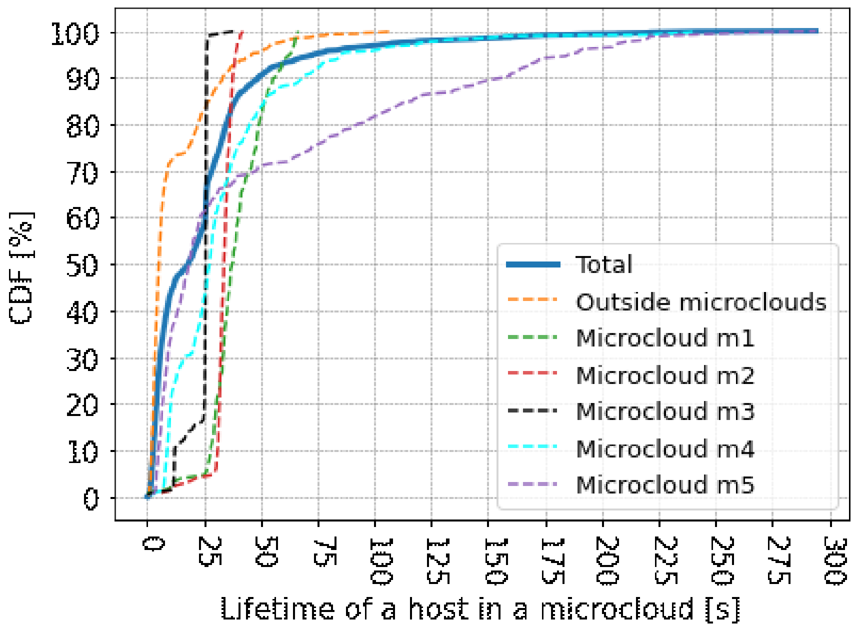
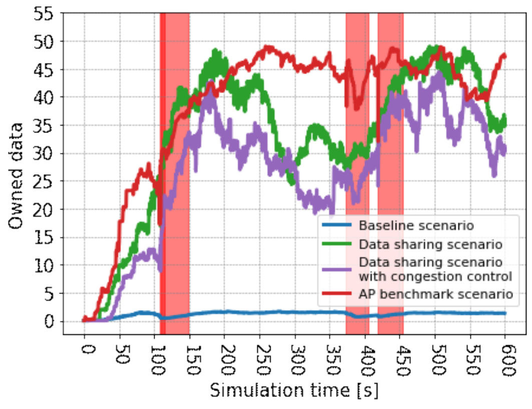


Figure 6.4: Time spent by hosts in micro cloud areas.

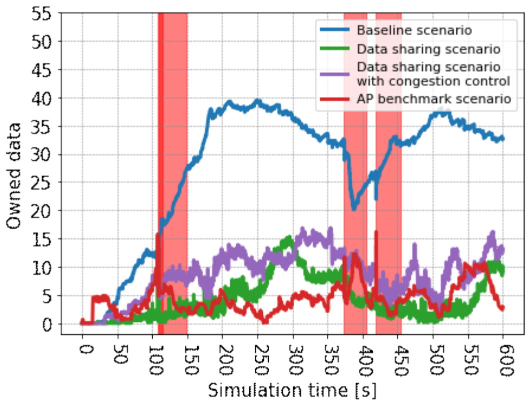
6.6 Results

The simulation environment used for this project is the INET Framework [129], an open-source communication network simulation package, written for the OMNeT++ [130] simulation system. The INET model used as a baseline is a UDP basic application. Hosts can move following a motion model and can exchange basic UDP packets, leveraging WiFi Direct communications. Starting from this, a BonnMotion [131] model of mobility traces from [125] was created and introduced in the simulator. The radio channel was modeled using the “APSK scalar radio medium” INET package, with yields ideal obstacle loss and simple path loss models with 50 m communication range and 90% of packets received at a distance of 40 m, mimicking an indoor WiFi communication environment using APSK modulation.

Hosts were also supplemented with energy storage and consumption models with a 3.1 Ah maximum capacity (i.e., a good-quality battery capacity for a smartphone). A random energy level from 0% to 5% was assigned to every node at the beginning of the simulation. In this way, we were able to deal with sudden deactivation of hosts due to poor battery levels. Other simulation parameters are reported in Table 6.1.



(a)



(b)

Figure 6.5: Content items comparison between scenarios. (a) Owned content. (b) Missing content.

6.6.1 Mobility Analysis

A first and necessary consideration regarding results derived from our model is their high dependency on the chosen mobility scenario. Mobility traces used in our scope define a model with peculiar characteristics, which affects interactions between hosts and, consequently, the whole system. A clear example of this is the event of a train approaching a platform area: in a few seconds, we can observe a discrete number of new hosts popping out on the map and, a few moments later, hosts with longer lifetimes and a large number of owned contents boarding the train and eventually leaving the model. This behavior is confirmed in the mobility analysis reported in Fig. 6.3, where the number of hosts passing through micro cloud areas is reported for every micro cloud throughout the simulation. Red vertical bars represent incoming trains at a platform. It is possible to notice how, for every area of study, trains approaching affect the mobility environment with a spike and/or a depression in the number of hosts.

A more deep mobility study highlighting differences between the mobility of each area was performed. Fig. 6.4 reports the average time a host spends in every area. In particular, the thick blue line represents the overall average time a host experiences in the simulation map: 90% of hosts exit from simulation within 50 s from their initialization. A more detailed view is provided by the dashed lines, which constitute insights into the average time spent by hosts in every micro cloud area. We can observe how some areas are solely transit zones, like the north corridor or the escalators. Differently, the main hall and platform areas see hosts pause for longer periods of time. As a consequence, not all micro clouds will experience the same content sharing environment: areas observing a much greater number of hosts for a longer time are more likely to be the places for wider data distribution.

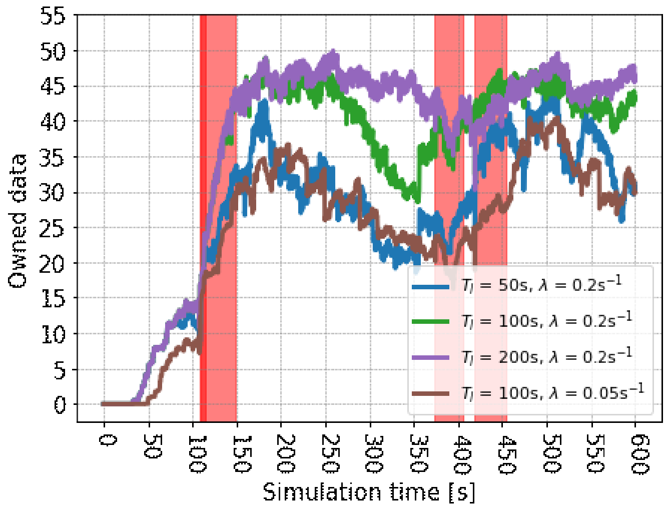


Figure 6.6: Content lifetime and frequency creation analysis.

6.6.2 Micro Cloud Performance

Regarding content items, Fig. 6.5 depicts the number of owned and missing items during simulation time for all the scenarios described in the previous section. The blue line in the figures represents the first scenario where sharing data between hosts was not allowed. Since the only chance for an host to own a content item in this stage is to create it, the number of overall owned data is extremely low, while on the opposite we see the number of missing items growing up to 40, i.e., the 80% of total data in the model. Results for the second scenario correspond to green lines and show how sharing data packets between hosts improves the number of owned content items by up to 100% (50 items out of 50), while the number of missing ones does not overcome 15. The purple line highlights similar performances for the third scenario with congestion control with respect to the sharing one. The main difference between the latter two scenarios is regarding channel collisions, as described in the next subsection. Eventually, red lines represent data owned and requested by hosts in the scenario with APs. Having a few fixed stations with all the data significantly improves results, since the scenario is much less distributed. Nevertheless, high costs of installation and maintenance have to be considered in this latter case.

An analysis regarding content lifetime and data creation frequency is also reported in Fig. 6.6. It is clear how it is possible to improve the spread of items over the map by increasing the content item lifetime T_l in the same scenario. Moreover, the brown line in the figure reports the number of owned content items in a scenario with a less frequent content generation rate λ with respect to others. Comparing the brown line with the green one with a similar T_l, it is possible to notice how content creation frequency has also an impact on content items spreading.

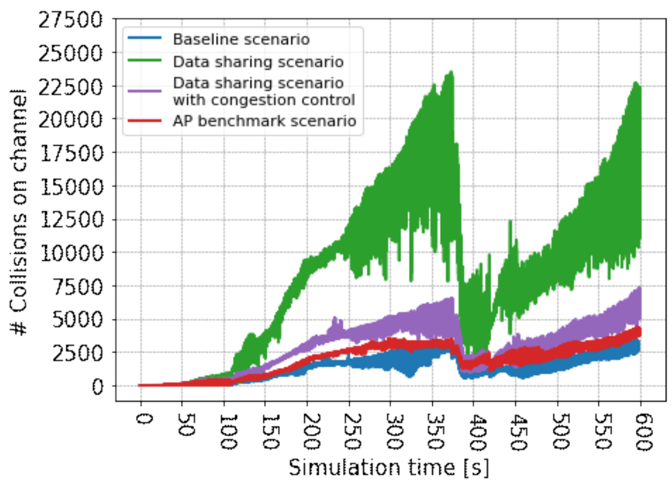


Figure 6.7: Broadcast collisions on channel.

6.6.3 Wireless Channel Load

Eventually, an analysis of wireless channel load was performed and the average number of collisions over the channel media is reported in Fig. 6.7. In this plot, we can see the improvement introduced by the congestion control mechanism, i.e., the purple line, with respect to the green line of the second scenario. The blue line shows a small number of collisions on the channel due to the only beacon sending leveraged in the first scenario. For the last, the red line reports the number of collisions for the APs scenario: only APs are entitled to share content items in this scenario, resulting in a minimum number of channel load.

6.7 Conclusions

The continuous growth of the urban population and the high development and maintenance costs of infrastructure-based approaches make it necessary the utilization of distributed schemes. Such distributed systems can turn the large demand into an advantage by exploiting vast populated areas for the purpose of spreading content items the most. Distributed applications relying on micro clouds are objects of study in both vehicular and pedestrian environments. In the present work, we propose a distributed application based on micro clouds for spreading content items in a sharing economy environment with cashback rewards for users. We thus implemented a content sharing procedure with evaluation over different scenarios. In particular, we considered a baseline scenario where content items sharing between hosts is not allowed, a scenario with items shared among users and a third scenario where channel congestion control is also taken into account. Eventually, a benchmark scenario with Access Points was also analyzed, where better performance is offset by high equipment costs.

Results presented show how it is possible to achieve up to a 100% spread of content items with the described procedure. Furthermore, according to the methodology described that takes into account channel congestion, a 66% reduction in collisions over the channel is achieved. Eventually, a deep mobility study and a content lifetime and frequency creation analysis were also conducted.

As future work, the micro cloud model could be further expanded into a much more large-scale scenario. We are planning to consider a mixed micro cloud scenario where vehicular and pedestrian micro clouds can share content items between each other thanks to Vehicle-to-Pedestrian (V2P) communication. A more complex study could also introduce data from Road Side Units (RSUs) in a full Vehicle-to-Everything (V2X) fashion.

Chapter 7

Conclusions

Urban Mobility studies are essential for the investigation of metropolitan traffic patterns, the evaluation of new traffic policies, the testing of upcoming vehicular communication technologies, and the planning of efficient public transportation systems. Concepts like self-driving cars, vehicular safety applications based on Vehicle-to-Vehicle communication, and real-time mobility services for drivers and pedestrians are becoming a reality as a result of the development of new technologies in the digital communication field. All such concepts, as well as others, necessitate the development of proper urban mobility studies in order to be successfully developed. The main objective of this thesis is to present the methodology and performance evaluation methods that have been applied in different projects covering urban mobility aspects.

In Chapter 2 we have presented an urban-scale traffic simulator named TuST (Turin SUMO Traffic). TuST covers 600^2 Km in and around the Metropolitan City of Turin, and it was modeled using the SUMO simulator. It was created with the help of real-time traffic data given by 5T, the info-mobility authority that supervises all aspects of mobility in the whole Piemonte region. From a set of Traffic Assignment Zones (TAZs), defined by the Turin Metropolitan Mobility Agency and used for several sorts of analysis, and from the Origin/Destination matrices, which describe all of the vehicular daily trips that begin and end in a TAZ, we built our model from scratch. In addition, we modeled the phases of more than 700 traffic lights using data collected from actual traffic light management systems provided by 5T. As a result, we were able to run an extensive Traffic Assignment study on many algorithms as a result of these inputs, and we were able to determine which algorithm provided the best solution for our model. We compared our output findings with authentic traffic data from 5T, which was collected from sensors installed beneath the surface of Turin's major roads. The results of this validation procedure showed that our simulator had acceptable discrepancy levels and confirmed the accuracy of our model. The final result showed how such a large model is capable of absorbing a full-day traffic demand in input at the cost of minor approximations.

The development and testing of a Vehicle-to-Vehicle Virtual Traffic Light (V^{3}TL) application is the subject that we have presented in Chapter 3. Thanks to the Veins simulator, we were able to replicate an environment where vehicles can communicate with each other and share basic motion information. Using this knowledge, each vehicle can recreate a four-step procedure for scheduling priorities at an uncontrolled junction. The complexity and performance of the entire V^{3}TL technique, as well as the scheduling algorithm we created, were evaluated in this study. We also provided a detailed description of our scheduling algorithm, including how decision trees were generated for each iteration in order to identify local optima in a heuristic approach. In order to meet our expectations, we tested our system at a variety of junction types and with a wide range of vehicle generation rates. The final results were compared to realistic situations in which crossings were unregulated or were governed by a traffic light modeled using the Webster approach. The system's fairness was further examined in a complex two-intersection scenario with multiple lanes per direction. Simulations demonstrate that the proposed method leads to considerable improvements in both unregulated junctions and traffic-light-regulated intersections. Improvements in our evaluated metrics have been noticed in all circumstances and with a variety of traffic generation rates.

The description of real-world test cases implemented in the context of automotive applications that rely on Vehicle-to-Network (V2N) communication is the focus of Chapter 4. The Advanced Message Queuing Protocol (AMQP) is the most extensively used protocol for controlling message flows across multiple entities, and it is the optimal option for maintaining a message Broker server. The AMQP Broker has been described in detail in this chapter, including the distinctions between the various data structure options that are accessible to the developer. Also included in this chapter was a brief overview of the main standardized protocols and messages proposed by both the American and European main standardization entities, namely the Society of Automotive Engineers (SAE) and the European Telecommunication Standards Institute (ETSI). When dealing with V2N communication, it is also necessary to consider a possible visualization approach that will be capable of displaying the events that will be triggered as a result of the communication. Specifically, we presented a solution for an online platform service for real-time event visualization on a map environment that is accessible over the internet. Finally, we have outlined two use cases on which we have worked, in which vehicular services utilizing V2N communication were deployed in real-world testing environments. For the 5G Automotive Association (5GAA) project, the demonstration included an Urban Geo-referenced Alert with real-time road traffic signal information transmitted across the network. The Rainbow project focused its efforts on the construction of an Urban Mobility Demonstrator, which would allow drivers to be notified in real time about possible road hazards that they could encounter on the road ahead of them.

We have reported the development and performance evaluation of an Automatic

Passenger Counting System (APCS) for the project detailed in Chapter 5. The service provided by the APCS was required by the local public transportation agency in Turin, GTT, which had a requirement to know the number of individuals that boarded their public buses at each bus stop in real time. A precise capacity indicator was required by the GTT in order to shape bus frequencies across the lines and to comply with occupancy limitations established by governments during the COVID-19 pandemic. In order to meet this need, we installed a Raspberry Pi sensor on a public GTT bus and programmed a software process on it that was capable of gathering WiFi signals emitted by cellphones. In particular, we are interested in studying Probe Request frames, which are specific WiFi signals issued by every network device in order to scan their immediate vicinity for available Access Points to connect to. The fact that our procedure is done on the fly and does not gather any information about the identity of any person involved ensures that the integrity of users' privacy is not compromised in any way. Through an analysis of the traffic patterns discovered on the bus over the course of many months of measurements, the suggested findings revealed an accuracy level of more than 80% for the counting of passengers, demonstrating that the proposed approach is valid for this purpose.

In Chapter 6 we have presented a content-sharing application focused on a pedestrian-based micro cloud environment. The concept of micro cloud analysis, which is commonly employed in the vehicular area for floating car data studies, was specifically applied to a scenario involving indoor pedestrian mobility. The pedestrian traces from a previous study, in which authors recreated in a synthetic way a realistic picture of people's movements in a metro station in Stockholm, served as the basis for the mobility model that we used in this study. The outcomes we obtained were generated from simulations in which each user could share their content items with others in a WiFi Direct communication fashion. The application we intended to recreate leverages a sharing economy model, in which users may choose to share information on train timetables and service disruptions, as well as advertisements and special deals from the coffee shop and store at the metro station's main hall. When users share this information through the subway station, they are rewarded with cashback or discounts from the transportation authority or from store owners, which provides an incentive for them to do so. The results demonstrated that using the proposed procedure, it is feasible to obtain a content item spread of up to 100%. Using the approach provided, which takes into account channel congestion, a 66% decrease in collisions over the channel was obtained.

All the applications and solutions presented in this thesis demonstrate the significance that urban mobility studies will have in the future for the development of new and emerging technologies. A significant increase in the number of services enabled by new mobile network technologies will be seen in the next generation of vehicles and network devices. As a result, studies of mobility, such as the ones discussed in this thesis, are crucially vital in order to support and expand the innovation of all of the investigations that are related to the urban environment.

List of acronyms

3G	Third Generation Mobile Networks
3GPP	Third Generation Partnership Project
4G	Fourth Generation Mobile Networks
5G	Fifth Generation Mobile Networks
5GAA	5G Automotive Association
5T	Telematic Technologies for Transports and Traffic in Turin
A&M	Agricultural and Mechanical
A-MPDU	Aggregate MAC Protocol Data Unit
ADAS	Advanced Driver Assistance Systems
AHED	Automatic Hazardous Events Detection
AI	Artificial Intelligence
AMQP	Advanced Message Queuing Protocol
AP	Access Point
APCS	Automatic Passenger Counting System
APSK	Amplitude and Phase-Shift Keying
ASN.1	Abstract Syntax Notation revision One
BSM	Basic Safety Message
BSSID	Basic Service Set Identifier
C-ITS	Cooperative-ITS
C-V2X	Cellular Vehicle-to-Everything
CA	Certification Authority
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CIA	Central Intelligence Agency
CO_2	Carbon Dioxide

COVID-19	Corona Virus Disease 2019
CRF	Centro Ricerche FCA
DA	Destination Address
DENM	Decentralized Environmental Notification Message
DS	Direct Sequence
DSRC	Dedicated Short-Range Communications
eNB	E-UTRAN NodeB
EPA	Environmental Protection Agency
ETSI	European Telecommunication Standards Institute
EVA	Emergency Vehicle Alert
FCA	Fiat Chrysler Automobiles
FCS	Frame Check Sequence
GHG	Greenhouse Gas
GLOSA	Green Light Optimal Speed Advisory
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GTT	Gruppo Torinese Trasporti
HMI	Human-Machine Interface
HS	Heading Set
HT	High Throughput
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IARC	International Agency for Research on Cancer
ICA	Intersection Collision Avoidance
ICT	Information and Communications Technology
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
ISO	International Organization for Standardization
ISTAT	Istituto Nazionale di Statistica
ITA	Incremental Traffic Assignment

ITS	Intelligent Transportation System
IVIM	Infrastructure to Vehicle Information Message
JD	Junction Dataset
LD	Leader Dataset
LDM	Local Dynamic Map
LiDAR	Light Detection And Ranging
LLC	Logical Link Control
LTE	Long Term Evolution
LuST	Luxemburg SUMO Traffic
MAC	Media Access Control
MANET	Mobile Ad-Hoc Network
MCS	Modulation Coding Scheme
MEC	Mobile Edge Computing
MoST	Monaco SUMO Traffic
MQTT	Message Queue Telemetry Transport
NIC	Network Interface Controller
NSA	National Security Agency
O/D	Origin/Destination
OBD	On Board Diagnostic
OBU	On Board Unit
OMNeT++	Objective Modular Network Testbed in C++
OS	Operating System
OSI	Open Systems Interconnection
OSM	OpenStreetMap
OUI	Organizationally Unique Identifier
$\mathbf{PM}_{2.5}$	Fine Particular Matters
$\mathbf{PM}_{10}$	Coarse Particular Matters
PoE	Power over Ethernet
QoS	Quality of Service
Radar	Radio Detecting And Ranging
RHS	Road Hazard Signaling
RP	Raspberry Pi

RSU	Road Side Unit
SA	Source Address
SAE	Society of Automotive Engineers
SD	Solution Dataset
SSID	Service Set Identifier
STOMP	Streaming Text Oriented Messaging Protocol
SUE	Stochastic User Equilibrium
SUMO	Simulation of Urban Mobility
SV	SuperVision
TAPASCologne	Travel And Patterns Simulation in Cologne
TAZ	Traffic Assignment Zone
TCP	Transmission Control Protocol
TOC	Traffic Operation Center
TuST	Turin SUMO Traffic
UDP	User Datagram Protocol
UE	User Equipment
UGA	Urban Geo-referenced Alert
UMD	Urban Mobility Demonstrator
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
V^{3}TL	Vehicle-to-Vehicle Virtual Traffic Light
VANET	Vehicular Ad-Hoc Network
Veh/h	vehicles per hour
Veins	Vehicles In Network Simulator
VHT	Very High Throughput
VRU	Vulnerable Road User
VTL	Virtual Traffic Light
WAVE	Wireless Access for Vehicular Environment
WHO	World Health Organization

WLAN	Wireless Local Area Network
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
ZTL	Zona a Traffico Limitato

Bibliography

[1] Marco Rapelli. *TuSTScenario*. Politecnico di Torino - FULL Lab. URL: https://github.com/marcorapelli/TuSTScenario.

[2] United Nations Framework Convention on Climate Change. “Paris Declaration on Electro-Mobility and Climate Change and Call to Action”. In: *epub* (2015).

[3] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. “Vehicular Traffic Simulation in the City of Turin from Raw Data”. In: *IEEE Transactions on Mobile Computing* (2021).

[4] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. “TuST: from raw data to vehicular traffic simulation in Turin”. In: *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE. 2019, pp. 1–8.

[5] Ahmadreza Jame, Marco Rapelli, and Claudio Casetti. “Reducing Pollutant Emissions through Virtual Traffic Lights”. In: *Computer Communications* (2022).

[6] Marco Rapelli, Claudio Casetti, and Marcello Sgarbi. “A Distributed V2V-Based Virtual Traffic Light System”. In: *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE. 2020, pp. 122–128. DOI: 10.1109/COMSNETS48256.2020.9027339.

[7] Marco Rapelli, Gurjashan Singh Pannu, Falko Dressler, and Claudio Casetti. “Content Sharing in Pedestrian-based Micro Clouds”. In: *2022 IEEE 95th Vehicular Technology Conference (VTC2022)*. IEEE. 2022, pp. 1–7.

[8] David L Schrank and Timothy J Lomax. *Urban mobility report*. Texas Transportation Institute, The Texas A&M University System, 2009.

[9] Shen Wang, Soufiene Djahel, and Jennifer McManis. “An adaptive and vanets-based next road re-routing system for unexpected urban traffic congestion avoidance”. In: *2015 IEEE vehicular networking conference (VNC)*. IEEE. 2015, pp. 196–203.

[10] Muoversi a Torino (MATO) and Città di Torino (Municipality of Turin). *Torino Centro Aperto*. 2021. URL: https://www.muoversiatorino.it/torinocentroaperto/.

[11] Ademar T Akabane, Rafael L Gomes, Richard W Pazzi, Edmundo RM Madeira, and Leandro A Villas. "Apolo: A mobility pattern analysis approach to improve urban mobility". In: *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE. 2017, pp. 1–6.

[12] Gaetano Manzo, Marco Ajmone Marsan, and Gianluca Rizzo. "Performance modeling of vehicular floating content in urban settings". In: *2017 29th International Teletraffic Congress (ITC 29)*. Vol. 1. IEEE. 2017, pp. 99–107.

[13] Agon Memedi, Christoph Sommer, and Falko Dressler. "On the need for coordinated access control for vehicular visible light communication". In: *2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*. IEEE. 2018, pp. 121–124.

[14] 5T S.r.l. *5T Torino, 5t.torino.it*. URL: http://www.5t.torino.it.

[15] Sandesh Uppoor, Oscar Trullols-Cruces, Marco Fiore, and Jose M Barcelo-Ordinas. "Generation and analysis of a large-scale urban vehicular mobility dataset". In: *IEEE Transactions on Mobile Computing* 13.5 (2013), pp. 1061–1075.

[16] Valeria Caiati, Luca Bedogni, Luciano Bononi, Francesco Ferrero, Marco Fiore, and Andrea Vesco. "Estimating urban mobility with open data: A case study in Bologna". In: *2016 IEEE International Smart Cities Conference (ISC2)*. IEEE. 2016, pp. 1–8.

[17] Lara Codecá, Raphaël Frank, Sébastien Faye, and Thomas Engel. "Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation". In: *IEEE Intelligent Transportation Systems Magazine* 9.2 (2017), pp. 52–63.

[18] Lara Codecá, Jakob Erdmann, and Jérôme Härri. "A sumo-based parking management framework for large-scale smart cities simulations". In: *2018 IEEE Vehicular Networking Conference (VNC)*. IEEE. 2018, pp. 1–8.

[19] Karl-Heinz Kastner and Petru Pau. "Experiences with sumo in a real-life traffic monitoring system". In: *SUMO 2015–Intermodal Simulation for Intermodal Transport* 1 (2015).

[20] Roozbeh Ketabi, Babak Alipour, and Ahmed Helmy. "En route: Towards vehicular mobility scenario generation at scale". In: *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2017, pp. 839–844.

[21] Feng Xia, Azizur Rahim, Xiangjie Kong, Meng Wang, Yinqiong Cai, and Jinzhong Wang. “Modeling and analysis of large-scale urban mobility for green transportation”. In: *IEEE Transactions on Industrial Informatics* 14.4 (2017), pp. 1469–1481.

[22] Eclipse Foundation. *Turin SUMO traffic (TuST)*. 2021. URL: https://sumo.dlr.de/docs/Data/Scenarios.html.

[23] *JOSM*. 2014. URL: https://josm.openstreetmap.de.

[24] OpenStreetMap Community. *OpenStreetMap*. 2006. URL: https://www.openstreetmap.org.

[25] Eclipse Foundation. *SUMO - Simulation of Urban MObility, (version 1.1.0)*. 2018. URL: http://sumo.sourceforge.net/userdoc.

[26] Eclipse Foundation. *NETCONVERT, (version 1.1.0)*. 2018. URL: http://sumo.sourceforge.net/userdoc/NETCONVERT.html.

[27] Google Maps. *Distance matrix API*. 2021. URL: https://developers.google.com/maps/documentation/distance-matrix.

[28] MIZAR Automazione SpA. “UTOPIA–Urban Traffic Control System Architecture”. In: *MIZAR, Verona* (2012).

[29] John Glen Wardrop and James Ivor Whitehead. “Correspondence. some theoretical aspects of road traffic research.” In: *Proceedings of the Institution of Civil Engineers* 1.5 (1952), pp. 767–768.

[30] Liu Shihsien and Jon D Fricker. “Estimation of a trip table and the θ parameter in a stochastic network”. In: *Transportation Research Part A: Policy and Practice* 30.4 (1996), pp. 287–305.

[31] Christian Gawron. “Simulation-Based Traffic Assignment. Computing user equilibria in large street networks”. PhD thesis. Universität zu Köln, 1998.

[32] Tom V Mathew and KV Krishna Rao. “Fundamental parameters of traffic flow”. In: *Introduction to Transportation Engineering; National Program on Technical Education and Learning (NPTEL): Mumbay, India* (2007), pp. 1–8.

[33] Michael Behrisch, Daniel Krajzewicz, and Yun-Pang Wang. “Comparing performance and quality of traffic assignment techniques for microscopic road traffic simulations”. In: *Proceedings of DTA2008* (2008).

[34] B. Stabler, H. Bar-Gera, and E. Sall. *Transportation Networks for Research*. 2020. URL: https://github.com/bstabler/TransportationNetworks.

[35] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[36] Eclipse Foundation. *DUAROUTER, (version 1.1.0)*. 2018. URL: https://sumo.dlr.de/docs/duarouter.html.

[37] Eclipse Foundation. *MAROUTER, (version 1.1.0)*. 2018. URL: https://sumo.dlr.de/docs/marouter.html.

[38] Eclipse Foundation. *Simulation/Meso, (versio 1.1.0)*. 2018. URL: https://sumo.dlr.de/docs/Simulation/Meso.html.

[39] World Health Organization WHO. "Burden of disease from household air pollution for 2016". In: (2018).

[40] J Cloke, G Harris, S Latham, A Quimby, L Smith, and C Baughan. "Reducing the environmental impact of driving: a review of training and in-vehicle technologies". In: *TRL REPORT 384* (1999).

[41] Michel André and Ulf Hammarström. "Driving speeds in Europe for pollutant emissions estimation". In: *Transportation Research Part D: Transport and Environment* 5.5 (2000), pp. 321–335.

[42] B Bradaï, A Garnault, V Picron, and P Gougeon. "A Green Light Optimal Speed Advisor for Reduced CO 2 Emissions". In: *Energy Consumption and Autonomous Driving*. Springer, 2016, pp. 141–151.

[43] Robert Braun, C Kemper, C Menig, F Busch, R Hildebrandt, I Paulus, R Pre ss lein-Lehle, and F Weichenmeier. "TRAVOLUTION network-wide optimization of the light signal control and lsa vehicle communication". In: *Road ss enverkehrstechnik* 6 (2009), S–365.

[44] Tessa Tielert, Moritz Killat, Hannes Hartenstein, Raphael Luz, Stefan Hausberger, and Thomas Benz. "The impact of traffic-light-to-vehicle communication on fuel consumption and emissions". In: *2010 Internet of Things (IOT)*. IEEE. 2010, pp. 1–8.

[45] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and guidance on standardization*. Standard. European Telecommunication Standard Institute (ETSI), June 2011.

[46] J Alexander Fax and Richard M Murray. "Information flow and cooperative control of vehicle formations". In: *IEEE transactions on automatic control* 49.9 (2004), pp. 1465–1476.

[47] Michel Ferreira, Ricardo Fernandes, Hugo Conceição, Wantanee Viriyasitavat, and Ozan K Tonguz. "Self-organized traffic control". In: *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*. 2010, pp. 85–90.

[48] Michel Ferreira and Pedro M d'Orey. "On the impact of virtual traffic lights on carbon emissions mitigation". In: *IEEE Transactions on Intelligent Transportation Systems* 13.1 (2011), pp. 284–295.

[49] Florian Hagenauer, Patrick Baldemaier, Falko Dressler, Christoph Sommer, et al. "Advanced leader election for virtual traffic lights". In: *ZTE Communications, Special Issue on VANET* 12.1 (2014), pp. 11–16.

[50] Christoph Sommer, Florian Hagenauer, and Falko Dressler. "A networking perspective on self-organizing intersection management". In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE. 2014, pp. 230–234.

[51] Alessandro Bazzi, Alberto Zanella, and Barbara M Masini. "A distributed virtual traffic light algorithm exploiting short range V2V communications". In: *Ad Hoc Networks* 49 (2016), pp. 42–57.

[52] Antonio Casimiro, Emelie Ekenstedt, and Elad Michael Schiller. "Membership-based manoeuvre negotiation in autonomous and safety-critical vehicular systems". In: *arXiv preprint arXiv:1906.04703* (2019).

[53] *Dedicated Short Range Communications (DSRC) Message Set Dictionary.* Standard. Society of Automotive Engineers, Sept. 2015.

[54] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. "Microscopic traffic simulation using sumo". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2575–2582.

[55] Andras Varga. "OMNeT++". In: *Modeling and tools for network simulation.* Springer, 2010, pp. 35–59.

[56] Christoph Sommer, Reinhard German, and Falko Dressler. "Bidirectionally coupled network and road traffic simulation for improved IVC analysis". In: *IEEE Transactions on mobile computing* 10.1 (2010), pp. 3–15.

[57] F. V. Webster. *Traffic signal settings.* Tech. rep. 1958.

[58] Rajendra K Jain, Dah-Ming W Chiu, William R Hawe, et al. "A quantitative measure of fairness and discrimination". In: *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA* (1984).

[59] European Telecommunication Standards Institute (ETSI). "TR 103 562 V2.1.1". In: *Intelligent transport systems (ITS)* (2019).

[60] OASIS Standard. "OASIS advanced message queuing protocol (AMQP) version 1.0". In: *International Journal of Aerospace Engineering Hindawi www. hindawi. com* 2018 (2012).

[61] *3GPP TR 21.916 V0.6.0 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 16 Description; Summary of Rel-16 Work Items (Release 16).* Technical Requirement. 3rd Generation Partnership Project, 2020.

[62] G. Avino, M. Malinverno, C. Casetti, C. F. Chiasserini, F. Malandrino, M. Rapelli, and G. Zennaro. "Support of Safety Services through Vehicular Communications: The Intersection Collision Avoidance Use Case". In: *2018 International Conference of Electrical and Electronic Technologies for Automotive*. 2018, pp. 1–6. DOI: 10.23919/EETA.2018.8493191.

[63] Marco Rapelli. *Torino Digital Mobility Application, online version*. 2019. URL: https://serverfull.polito.it/TorinoDMapp_online.html.

[64] Marco Rapelli. *Torino Digital Mobility Application, offline version*. 2019. URL: https://serverfull.polito.it/TorinoDMapp_offline.html?imei=866221031860344.

[65] Marco Rapelli. *City Aggregator online platform*. 2021. URL: https://serverfull.polito.it:7007/.

[66] Georgios Karagiannis, Onur Altintas, Eylem Ekici, Geert Heijenk, Boangoat Jarupan, Kenneth Lin, and Timothy Weil. "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions". In: *IEEE communications surveys & tutorials* 13.4 (2011), pp. 584–616.

[67] Yunxin Jeff Li. "An overview of the DSRC/WAVE technology". In: *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*. Springer. 2010, pp. 544–558.

[68] *IEEE 1609.0-2013 - IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture*. Standard. Institute of Electrical and Electronics Engineers, 2014.

[69] *IEEE 1609.2-2016 (Revision of IEEE Std 1609.2-2013) - IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages*. Standard. Institute of Electrical and Electronics Engineers, 2016.

[70] *IEEE 1609.3-2016 (Revision of IEEE Std 1609.3-2010) - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services*. Standard. Institute of Electrical and Electronics Engineers, 2016.

[71] *IEEE 1609.4-2016 (Revision of IEEE Std 1609.4-2010) - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation*. Standard. Institute of Electrical and Electronics Engineers, 2016.

[72] *ETSI EN 302 665 V1.1.1 - Intelligent Transport Systems (ITS); Communications Architecture*. Standard. European Telecommunication Standard Institute, 2010.

[73] *3GPP TR 21.914 V14.0.0 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 14 Description; Summary of Rel-14 Work Items (Release 14).* Technical Requirement. 3rd Generation Partnership Project, 2018.

[74] *SAE J2945 - On-Board System Requirements for V2V Safety Communications.* Standard. Society of Automotive Engineers, 2016.

[75] *ETSI TS 101 539-1 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 1: Road Hazard Signalling (RHS) application requirements specification.* Technical Specification. European Telecommunication Standard Institute, 2013.

[76] *ETSI TS 101 539-2 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 2: Intersection Collision Risk Warning (ICRW) application requirements specification.* Technical Specification. European Telecommunication Standard Institute, 2018.

[77] *ETSI TS 101 539-3 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 3: Longitudinal Collision Risk Warning (LCRW) application requirements specification.* Technical Specification. European Telecommunication Standard Institute, 2013.

[78] *ETSI TS 103 301 V1.2.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Facilities layer protocols and communication requirements for infrastructure services.* Technical Specification. European Telecommunication Standard Institute, 2018.

[79] *ETSI TS 102 894-2 V1.2.1 - Intelligent Transport Systems (ITS); Users and applications requirements; Part 2: Applications and facilities layer common data dictionary.* Technical Specification. European Telecommunication Standard Institute, 2014.

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

[81] *ETSI EN 302 637-3 V1.3.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service.* Standard. European Telecommunication Standard Institute, 2019.

[82] Binildas Christudas. “Activemq”. In: *Practical Microservices Architectural Patterns.* Springer, 2019, pp. 861–867.

[83] The Apache Software Foundation. *Qpid Proton.* 2015. URL: https://qpid.apache.org/proton/index.html.

[84] OpenJS Foundation. *Node JS.* URL: https://nodejs.org/it/.

[85] Mapbox Studios. *Mapbox*. URL: https://www.mapbox.com/.

[86] Guillermo Rauch. *Socket.IO*. URL: https://socket.io/.

[87] 5GAA. *5G Automotive Association*. URL: https://5gaa.org/.

[88] Intelligent transport systems of the European Committee for Standardization. *DATEX II*. URL: https://docs.datex2.eu/.

[89] Ben Zhang, Nitesh Mor, John Kolb, Douglas S Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. "The cloud is not enough: Saving iot from the cloud". In: *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*. 2015.

[90] Weisong Shi and Schahram Dustdar. "The promise of edge computing". In: *Computer* 49.5 (2016), pp. 78–81.

[91] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. "Edge computing: Vision and challenges". In: *IEEE internet of things journal* 3.5 (2016), pp. 637–646.

[92] Xiang Sun and Nirwan Ansari. "EdgeIoT: Mobile edge computing for the Internet of Things". In: *IEEE Communications Magazine* 54.12 (2016), pp. 22–29.

[93] Tuan Nguyen Gia, Amir M Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. "Fog computing approach for mobility support in internet-of-things systems". In: *IEEE Access* 6 (2018), pp. 36064–36082.

[94] Grafana Labs. *Grafana: The open observability platform*. URL: https://grafana.com/.

[95] *Spostamenti Quotidiani e Nuove Forme di Mobilità*. Statistical Document. ISTAT, Istituto Nazionale di Statistica, 2018.

[96] IEEE Computer Society LAN MAN Standards Committee et al. "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications". In: *ANSI/IEEE Std. 802.11-1999* (1999).

[97] *IEEE 802.11-2016 - IEEE Standard for Information technology — Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. Standard. Institute of Electrical and Electronics Engineers, 2016.

[98] Matthew S Gast. *802.11 ac: a survival guide: Wi-Fi at gigabit and beyond*. O'Reilly Media, Inc., 2013.

[99] Luiz Oliveira, Daniel Schneider, Jano De Souza, and Weiming Shen. "Mobile device detection through WiFi probe request analysis". In: *IEEE Access* 7 (2019), pp. 98579–98588.

[100] Julien Freudiger. “How talkative is your mobile device? An experimental study of Wi-Fi probe requests”. In: *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 2015, pp. 1–6.

[101] Célestin Matte. “Wi-Fi tracking: Fingerprinting attacks and counter-measures”. PhD thesis. Université de Lyon, 2017.

[102] Mathieu Cunche. “I know your MAC address: targeted tracking of individual using Wi-Fi”. In: *Journal of Computer Virology and Hacking Techniques* 10.4 (2014), pp. 219–227.

[103] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. “A study of MAC address randomization in mobile devices and when it fails”. In: *arXiv preprint arXiv:1703.02874* (2017).

[104] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. “Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms”. In: *Proceedings of the 11th ACM on Asia conference on computer and communications security*. 2016, pp. 413–424.

[105] Laura Chappell and Gerald Combs. *Wireshark 101: Essential skills for network analysis*. Protocol Analysis Institute, Chapell University, 2013.

[106] Gerald Combs. *TShark*. URL: https://www.wireshark.org/docs/man-pages/tshark.html.

[107] 5T SRL. *FALCO*. URL: https://falco.5t.torino.it/.

[108] Michele Nitti, Francesca Pinna, Lucia Pintor, Virginia Pilloni, and Benedetto Barabino. “iABACUS: A wi-fi-based automatic bus passenger counting system”. In: *Energies* 13.6 (2020), p. 1446.

[109] Unicef et al. “La condizione dell’infanzia nel mondo 2012–Figli delle città”. In: (2012), pp. 1–7. URL: http://www.unicef.it/Allegati/SOWC%5C_2012%5C_ITA.pdf.

[110] Parag Sewalkar and Jochen Seitz. “Vehicle-to-Pedestrian Communication for Vulnerable Road Users: Survey, Design Considerations, and Challenges”. In: *Sensors* 19.2 (Jan. 2019), p. 358. DOI: 10.3390/s19020358.

[111] Quang-Huy Nguyen, Michel Morold, Klaus David, and Falko Dressler. “Adaptive Safety Context Information for Vulnerable Road Users with MEC Support”. In: *15th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2019)*. Wengen, Switzerland: IEEE, Jan. 2019, pp. 28–35. DOI: 10.23919/WONS.2019.8795475.

[112] Quang-Huy Nguyen, Michel Morold, Klaus David, and Falko Dressler. “Car-to-Pedestrian Communication with MEC-Support for Adaptive Safety of Vulnerable Road Users”. In: *Elsevier Computer Communications* 150 (Jan. 2020), pp. 83–93. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2019.10.033.

[113] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. “The ONE simulator for DTN protocol evaluation”. In: *Proceedings of the 2nd international conference on simulation tools and techniques*. 2009, pp. 1–10.

[114] Noelia Pérez Palma, Falko Dressler, and Vincenzo Mancuso. “Precise: Predictive Content Dissemination Scheme Exploiting Realistic Mobility Patterns”. In: *Elsevier Computer Networks (COMNET)* 202 (Dec. 2021), p. 108556. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2021.108556.

[115] ETSI. *Mobile Edge Computing (MEC), Framework and Reference Architecture*. GS MEC 003 V1.1.1. ETSI, Mar. 2016.

[116] P. Mach and Z. Becvar. “Mobile Edge Computing: A Survey on Architecture and Computation Offloading”. In: *IEEE Communications Surveys & Tutorials* 19.3 (Mar. 2017), pp. 1628–1656. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2682318.

[117] Congfeng Jiang, Xiaolan Cheng, Honghao Gao, Xin Zhou, and Jian Wan. “Toward Computation Offloading in Edge Computing: A Survey”. In: *IEEE Access* 7 (Jan. 2019), pp. 131543–131558. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2938660.

[118] Mustafa Emara, Miltiades C Filippou, and Dario Sabella. “MEC-assisted end-to-end latency evaluations for C-V2X communications”. In: *2018 European conference on networks and communications (EuCNC)*. IEEE. 2018, pp. 1–9.

[119] Michael Till Beck, Martin Werner, Sebastian Feld, and S Schimper. “Mobile edge computing: A taxonomy”. In: *Proc. of the Sixth International Conference on Advances in Future Internet*. Citeseer. 2014, pp. 48–55.

[120] Baldomero Coll-Perales, M Carmen Lucas-Estañ, Chang-Heng Wang, Javier Gozalvez, Takayuki Shimizu, Sergei Avedisov, Miguel Sepulcre, Takamasa Higuchi, Bin Cheng, Akihiko Yamamuro, et al. “Impact of the MEC Location in Transport Networks on the Capacity of 5G to Support V2X Services”. In: *2021 16th Annual Conference on Wireless On-demand Network Systems and Services Conference (WONS)*. IEEE. 2021, pp. 1–8.

[121] Falko Dressler, Carla Fabiana Chiasserini, Frank H. P. Fitzek, Holger Karl, Renato Lo Cigno, Antonio Capone, Claudio Ettore Casetti, Francesco Malandrino, Vincenzo Mancuso, Florian Klingler, and Gianluca A. Rizzo. *V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing*. cs.NI 2106.10063. arXiv, June 2021.

[122] Iftikhar Ahmad, Rafidah Md Noor, Muhammad Reza Zaba, Muhammad Ahsan Qureshi, Muhammad Imran, and Muhammad Shoaib. “A cooperative heterogeneous vehicular clustering mechanism for road traffic management”. In: *International Journal of Parallel Programming* 48.5 (2020), pp. 870–889.

[123] Craig Cooper, Daniel Franklin, Montserrat Ros, Farzad Safaei, and Mehran Abolhasan. “A comparative survey of VANET clustering techniques”. In: *IEEE Communications Surveys & Tutorials* 19.1 (2016), pp. 657–681.

[124] Gurjashan Singh Pannu, Florian Hagenauer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. “Keeping Data Alive: Communication Across Vehicular Micro Clouds”. In: *20th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2019)*. Washington, D.C.: IEEE, June 2019. ISBN: 978-1-7281-0270-2. DOI: 10.1109/WoWMoM.2019.8792973.

[125] Ólafur Helgason, Sylvia T Kouyoumdjieva, and Gunnar Karlsson. “Opportunistic communication and human mobility”. In: *IEEE Transactions on Mobile Computing* 13.7 (2013), pp. 1597–1610.

[126] Kyuwon Han, Seung Min Yu, and Seong-Lyun Kim. “Smartphone-based indoor localization using Wi-Fi fine timing measurement”. In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2019, pp. 1–5.

[127] Eduardo Sánchez Morales, Michael Botsch, Bertold Huber, and Andrés García Higuera. “High Precision Indoor Navigation for Autonomous Vehicles”. In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2019, pp. 1–8.

[128] Google. *Go Inside With Indoor Maps*. URL: https://www.google.com/maps/about/partners/indoormaps/.

[129] Levente Mészáros, Andras Varga, and Michael Kirsche. “Inet framework”. In: *Recent Advances in Network Simulation*. Springer, 2019, pp. 55–106.

[130] Klaus Wehrle, M Güneş, and James Gross, eds. *Modeling and tools for network simulation*. Springer, 2010. ISBN: 978-3-642-12330-6. DOI: 10.1007/978-3-642-12331-3.

[131] Matthias Schwamborn and Nils Aschenbruck. “On modeling and impact of geographic restrictions for human mobility in opportunistic networks”. In: *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE. 2015, pp. 178–187.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.