University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Master of Science in Computer Science	Spring semester, 2023 [✓ Open] / Confidential
Author: Raymond Essandoh	
Main supervisor: Assoc. Prof. Naeem Khademi Co-supervisor: Aitor Martin Rodriguez	
Title of master's thesis: An Emulation Framework for Evaluating V2X Communications in C-ITS Applications	
Credits: 30	
Keywords: C-ITS Vehicle-to-Vehicle (V2V) Vehicle-to-Infrastructure (V2I) Vehicle-to-Everything (V2X) Interoperability Road Safety	Number of pages: 97 Stavanger, 14 July 2023

University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

An Emulation Framework for Evaluating V2X Communications in C-ITS Applications

Master's Thesis in Computer Science by

Raymond Essandoh

Main Supervisor

Assoc. Prof. Naeem Khademi

Co-Supervisor

Aitor Martin Rodriguez

July 14, 2023

Abstract

By utilizing cutting-edge communication technologies, Cooperative Intelligent Transportation Systems (C-ITS) have become a promising paradigm for transforming transportation systems. Vehicles can interact with one another and with infrastructure with the use of C-ITS, which makes it easier to exchange important data and supports in-the-moment decision-making. By examining the underlying communication protocols, data exchange processes, and system architectures, this thesis investigates the potential of C-ITS in improving road safety and efficiency.

Understanding the core ideas behind C-ITS, including its essential elements and technologies like Dedicated Short-Range Communication (DSRC) and Cellular Vehicle-to-Everything (C-V2X) communication, is the initial goal of the research. A thorough assessment of the literature is done to emphasize the C-ITS-related studies, protocols, and standards that have already been done.

The thesis then explores the development and application of a C-ITS framework which aims to become a platform for researchers and deployers to prototype C-ITS apps and evaluate communication performance in different flexible scenarios. The suggested framework incorporates several elements, such as cooperative decision-making algorithms, message propagation, and data collecting. To assess the usefulness and performance of the framework under different traffic circumstances, real-world tests and simulations are carried out.

The research's conclusions offer insightful information on the potential of C-ITS to transform transportation systems and present chances for legislators, traffic management agencies, and automakers to incorporate C-ITS technologies into current infrastructure. The findings show how C-ITS has a considerable potential to increase traffic safety, increase traffic efficiency, and eventually contribute to a more intelligent and sustainable transportation ecosystem.

Acknowledgments

The several people who helped and contributed to the completion of this thesis have my sincere gratitude. First and foremost, I want to express my sincere gratitude to my supervisor Assoc. Prof. Naeem Khademi and co-supervisor Aitor Martin Rodriguez for their tremendous support and advice during this study process. This thesis has been greatly influenced by their mentoring.

I owe the Department of Electrical Engineering and Computer Science at the Faculty of Science and Technology for creating a stimulating academic atmosphere that has substantially improved the quality of this work.

I also want to express my gratitude for the help and inspiration I have received from my friends and coworkers, whose camaraderie and stimulating discussions have served as a constant source of inspiration.

I will always be grateful to my dear family for their unwavering support, compassion, and faith in my skills.

Finally, I want to express my sincere gratitude to everyone who has shaped and inspired both my academic and personal development. Your support has had a lasting impact on my path. I want to extend my sincere gratitude to everyone named above as well as the many other people who have helped me along the journey for your unshakable believe in me.

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BTP	Basic Transport Protocol
C-ITS	Cooperative Intelligent Transport System
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CPU	Central Processing Unit
DENM	Decentralized Environmental Notification Message
ETSI	European Telecommunications Standards Institute
GLOSA	Green Light Optimal Speed Advisory
GUI	Graphic User Interface
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
NS-3	Network Simulator 3
OSI	Open System Interconnection
RSU	Roadside Unit
SUMO	Simulation of Urban Mobility
V2I	Vehicle-to-Infrastructure
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
WAVE	Wireless Access in Vehicular Environments

Chapter 1

Introduction

1.1 Motivation

Smart cities are already being shaped up today and there have been major efforts to empower the road infrastructure and its users with intelligence. A new trend has appeared where a car is more than a tool, capable of actively aiding its driver to mitigate some risks related to human interaction and increasing road safety for everyone.

By equipping a vehicle with the intelligence and technology to not only make informed decisions based on the data collected through its sensors but also on information that could be gathered by communicating with the road infrastructure and other cars driving on it, we're enhancing its decision capabilities. Tomorrow's car will represent a step change in form and function, combining the intelligence of driver-assisted vehicles with the potential of vehicular communications.

This vision has led to the advent of Cooperative Intelligent Transportation Systems (Cooperative Intelligent Transport System (C-ITS)) which rely on vehicular communication technologies to enable applications that could potentially improve road safety, and traffic efficiency, and introduce new entertainment and business models.

As these systems will be dealing with road safety, thorough validation has to be made, but due to the multidisciplinary areas involved, real-world C-ITS testing is complex and therefore very expensive. This is where computer-generated simulations shine, reducing simulation costs and complexity while offering a realistically simulated controlled environment even though they carry along their peculiar abstractions that enumerate their limitations.

1.2 Research Aims and Objectives

The Faculty of Science and Technology at the University of Stavanger is studying the development of a small-scale and controlled environment where C-ITS applications can be tested in the real world. To support that ambition, this study will use a computer simulation program to establish the groundwork for future research on C-ITS-enabled settings before investing in a real-world project.

The study's research questions are as follows:

1. What fundamental elements and features must a C-ITS simulation system have to faithfully reproduce cooperative traffic scenarios including V2V and V2I communication?

2. How can the C-ITS emulation framework's efficiency and scalability be improved to handle extensive simulations and real-time traffic conditions?

3. What are the main design factors to ensure the C-ITS emulation framework's flexibility, modularity, and extensibility to meet future technology improvements and growing C-ITS standards?

The main objectives of this study are as follows:

1. The creation of a thorough C-ITS emulation framework that includes crucial elements including vehicle models, communication protocols, traffic management algorithms, and road infrastructure models to faithfully replicate cooperative traffic scenarios.

2. By using effective algorithms, parallel processing techniques, and resource utilization strategies to handle large-scale simulations and real-time traffic circumstances, it will be possible to maximize the performance and scalability of the C-ITS emulation framework.

3. By using open standards, flexible interfaces, and modular architecture, the C-ITS emulation framework will be able to adapt to changing C-ITS standards as well as future technological breakthroughs.

Using a methodical development procedure, the research objectives will be met. The development and implementation of the C-ITS emulation framework will require a thorough examination of the C-ITS frameworks and standards now in use, the identification of key elements and functionalities, and the design of the C-ITS framework. Benchmarking, simulation tests, and performance analysis will be used to gauge the framework's effectiveness and scalability. Different C-ITS applications, protocols, and technologies will be incorporated to evaluate the framework's flexibility and extensibility. The framework will be improved through testing, prototyping, and iterative development.

1.3 1.3 Outline

This document is structured as follows:

Chapter 1 serves as an introduction.

Chapter 2 describes the current methodologies, tools, and standards utilized by the C-ITS sector.

Chapter 3 describes the simulation platform architecture, including the chosen simulation stack.

Chapter 4 offers a detailed explanation of the development of the framework.

Chapter 5 discusses the usage of the framework as well as the chosen evaluation scenarios and applications, test case documents for each, gathered results, and scalability analysis of the chosen software.

Finally**, Chapter 6** concludes the thesis by organizing findings related to the simulation stack, discussing how it can contribute to the C-ITS research community, and offering future work ideas based on what has been accomplished.

Chapter 2

Literature Review

To begin with, we will have an overview of the following terms and concepts that are essential for understanding the objectives and implementation of this project:

- What is an emulation framework?
- What is C-ITS

Also, we will cover extensively the related research and findings on C-ITS, the necessary tools that encompass the development of C-ITS solutions, and the potential use cases for our emulation framework. Some frameworks, libraries, and tools for C-ITS emulation and simulation are thoroughly reviewed in this chapter. The goal is to identify the most important innovations and techniques used in this field. The academic articles, conference proceedings, and pertinent industry reports published between 2010 and 2023 were the main targets of the literature search. The review's main goal is to draw attention to the most popular frameworks, libraries, and tools for C-ITS simulation and their suitability for usage in diverse research and development contexts.

2.1 Emulation Framework

A complete platform for the simulation and study of multiple systems, including computer networks, software applications, and hardware designs, is an emulation framework. Researchers, developers, and engineers may test, assess, and improve their designs and algorithms in this virtual environment by simulating the behavior and traits of real-world systems. The framework often comes with tools, libraries, and APIs to make building, configuring, and managing mimicked systems easier. Users may evaluate system performance, scalability, reliability, and security in a controlled and repeatable manner thanks to its capabilities including traffic creation, performance monitoring, and result analysis. Users may greatly decrease the expense and difficulty of physical testing while receiving insightful knowledge about the behavior and performance by utilizing an emulation framework.

Several modeling and simulation frameworks for C-ITS scenarios have been created. Researchers and engineers may simulate complicated interactions between automobiles, infrastructure, and other stakeholders using these frameworks. The iTETRIS framework (Intelligent Transport Systems Test Bed for Rapid Assessment of Innovative Solutions) is one significant framework that has been mentioned in the literature [1]. With support for V2V and V2I communication, cooperative mobility scenarios, and traffic management techniques, iTETRIS provides a thorough environment for testing and assessing C-ITS applications. The Simulation of Urban Mobility (SUMO) [2] is another well-known framework that is frequently used to simulate traffic situations for C-ITS systems. In-depth road network modeling, simulations of traffic flow, and support for V2X communication protocols are all provided by SUMO. In addition, the widely used commercial simulation application VISSIM (Traffic Simulation and Analysis) [3] also supports C-ITS scenarios. For modeling complicated traffic settings and studying the effects of C-ITS technologies on mobility and safety, VISSIM offers a complete range of functions.

2.2 C-ITS Simulation Libraries and Tools

To aid in C-ITS research and development, several simulation libraries and tools have been created in addition to frameworks. These libraries and tools offer specialized features and modules that can be applied to independent tools or incorporated into bigger simulation frameworks. Veins is a well-known C-ITS simulation package that has been discussed in the literature [4].

Veins is an open-source framework for simulating vehicular networks that works with OMNeT++ and SUMO. It supports several wireless communication protocols and allows the simulation of real-world V2V and V2I communication scenarios. C-ITS simulations have made use of OMNeT++ [5], a discrete event simulation framework popular in network modeling. It has been used in conjunction with C-ITS libraries and provides substantial support for modeling and simulating communication networks. Additionally, by adding new C-ITS-specific libraries, the open-source network simulator ns-3 [6] has been used for C-ITS simulation research. To explore C-ITS communication protocols and scenarios, use ns-3, which offers a complete set of modules for simulating communication networks.

The present academic discourse concerns the recommended articles for the development of specific Day 1 Services in the context of CITS applications. These said articles have been previously devised and utilized using a simulation stack that shares several similarities with the ones being emphasized.

The research article titled "A Decentralized Vehicle Re-routing Approach using Vehicular Ad-hoc Networks" authored by S. TRakkesh et al proposes a method for redirecting vehicles through a decentralized system utilizing vehicular ad-hoc networks. The data presented by [7] postulates a novel rerouting tactic that could be integrated into a VANET system to effectively redirect automobiles in the event of an accident or obstruction within the traffic locality.

The work of Noori and Valkama [8] presents a VANET communication approach aimed at reducing travel time in realistic, large-scale urban areas. They propose a novelle technique for determining the most efficient path from the point of departure to the destination, utilizing vehicular communication. The utilization of V2X communication technology facilitates the reduction of emissions, traffic congestion, and fuel consumption en route to a given destination. The work titled "Potentials and Limitations of Green Light Optimal Speed Advisory Systems" authored by David Eckhoff et al. [9] constitutes a research endeavor conducted within the academic context.

2.3 Cooperative Intelligent Transport System

Cooperative Intelligent Transport Systems (C-ITS) are created by equipping ITS components with standardized interaction capabilities. This allows road vehicles to communicate with other vehicles, traffic signals, road infrastructure, and other road users. With increased access to information, C-ITS has the potential to reduce road fatalities, improve road capacity, reduce carbon emissions, and enhance the user experience during travel by allowing road users and traffic managers to coordinate their actions. **Figure 1** represents a typical C-ITS ecosystem.

The EU's C-ITS 2016 report states that due to the expected benefits and relatively moderate costs associated with deployment, there is a strong interest in enabling a quick European-wide move towards market production and early deployment.

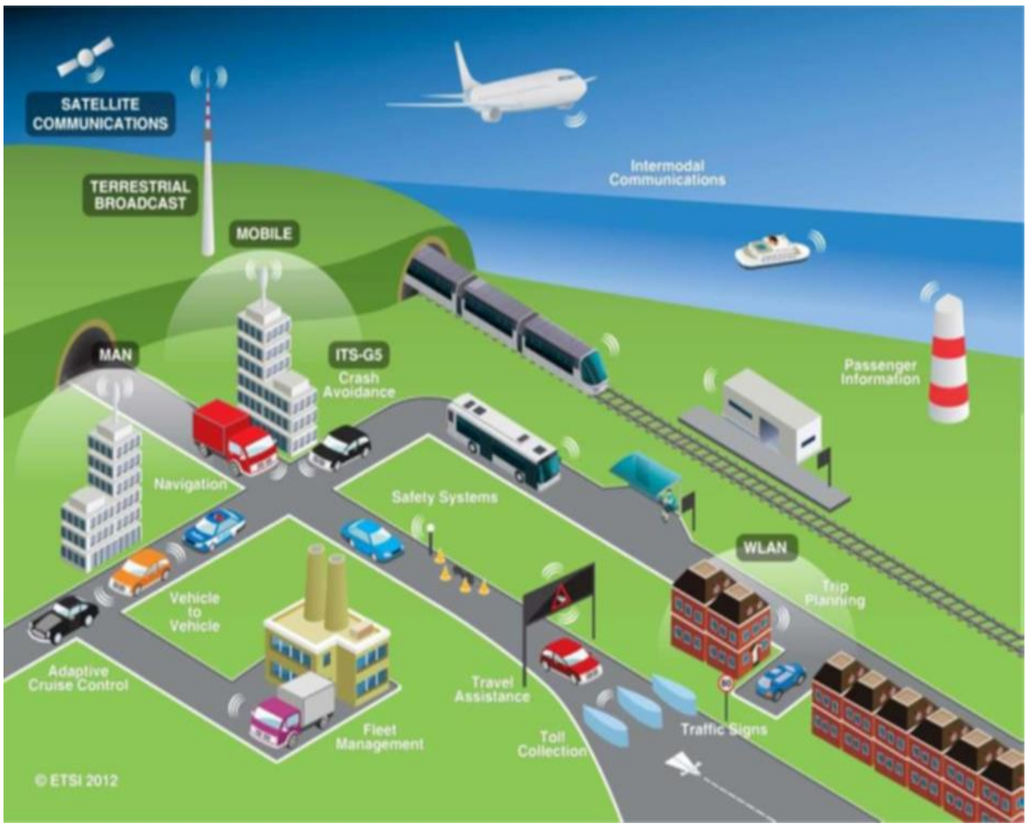


Figure 1: Cooperative Intelligent Transport System Ecosystem [10]

2.4 C-ITS architecture

Each agent in the C-ITS system is an ITS station (ITS-S) that can record its surroundings via built-in sensors. The ability to collect information about the environment paved the way for cooperative applications in which two or more ITS stations could exchange messages about relevant events. A connection between two ITS stations can be achieved using Vehicle-to-Everything (V2X) technology, which allows standardized message exchange. These technologies are categorized as follows:

- **Vehicle-to-vehicle (V2V):** A connection used by two or more vehicles forms a vehicle ad-hoc network (VANET) through which the vehicles communicate.

- **Vehicle to infrastructure (V2I):** An interconnect used by vehicles to communicate with Roadside Units (RSUs). Reverse communication is defined as infrastructure-to-vehicle (I2V).

- **Vehicle to Pedestrian (V2P):** Network between vehicles and pedestrians via personal ITS stations and vice versa.

Due to the diversity of potential business fields for C-ITS systems, many competing stakeholders, such as car manufacturers, telecommunications operators, road infrastructure operators, and owners, are interested in its development. Therefore, a common framework of standards and protocols was needed to connect products from different stakeholders and ensure proper C-ITS scalability.

For this reason, protocol architectures supporting C-ITS applications have been developed in the US, Japan, and the EU [11]. This project focuses on compliance with the European Standards Institute (ETSI) [12] guidelines for the worldwide deployment of C-ITS protocol and architecture standards. This decision was made because this is the most popular protocol within the research community.

2.5 ETSI ITS-G5

In terms of the type of vehicular communications that are the focus of the tools developed within the scope of this thesis, V2V, as previously stated, ETSI ITS-G5 is currently the most widely accepted standard among the largest European industry players in the automotive area. Similarly, in the US, IEEE 1609 WAVE is the region's equivalent accepted standard; it is very similar in terms of what ETSI ITS-G5 specifies, and they both share IEEE 802.11p, as underlying communications.

The goal of this project is to develop tools that will enable the analysis and evaluation of ITS-G5 implementations on various automotive scenarios using the IEEE 802.11p lower layers. Although there is currently discussion about the potential use of LTE technology (LTE-V2X) [13], with 5G in mind, and its advantages compared to ETSI ITS-G5 usage of IEEE 802.11p, this comparison will not be analyzed within the scope of this project.

As was already mentioned, ETSI ITS-G5 appears to be the standard that has the best possibility of being adopted by the automotive industry, at least in Europe. Due to the complexity of all ITS scenarios, communications and information exchanges between ITS subsystems are conducted and should be feasible. These subsystems can have a wide range of topologies, from the most obvious ones, such as vehicles, to RSUs that can provide useful information regarding the status of its surroundings, or even road walk pedestrians carrying a smart phone that can be implied in whatever scenario is currently in play. Figure 8 shows a few of these subsystems and their respective stack design.

The ITS-S host, which is the focus of this project and represents the component responsible for most of the implementation required by ETSI ITS-G5 going from the application layer to the lower layers that are, as well, implemented on the ITS-S router component, is shown under Figure 9, where three major components can be seen. Because only V2V communications are the focus of this thesis, this figure provides a more detailed overview of the stack architecture expected in a vehicle.
In most cases, this gateway will connect the ITS-S components to a CAN Network to get access to and exchange information with the car's ECUs responsible for the vehicle actuators, as well as to gather some useful data. The Vehicle ITS-S gateway is the component responsible for the interface of both previously mentioned pieces with the Proprietary in-vehicle Network, which should vary within different manufacturers.

A typical Vehicular ITS-S Host's stack design is shown in Figure 2, along with some example items that could be implemented within each layer. The Application layer from the OSI model can be merged into the Applications and Facilities layers described by ETSI, following the commonly accepted reference OSI model [14], just as the Access layer developed by ETSI unifies the Physical and data connection levels from the OSI model.
Considering this, most of the work in this part and the ones that follow will be devoted to analyzing the Application and Facilities layers, although the network and transport as well as the access layers will be briefly touched on in the discussion and how they exchange information to confirm any potential co-influences under various implementations.:

The **ITS applications** layer includes ITS applications and use cases for traffic efficiency, road safety, entertainment, and business.

A selection of features is offered by the **ITS facilities layer** to facilitate ITS applications. Data structures are provided by the facilities to store, combine, and retain data of many types and sources (such as data from car sensors and data obtained through communication). When it comes to communication, ITS facilities assist the construction and maintenance of communication sessions, offer message handling that is special to ITS, and enable various forms of addressing to applications. The administration of services, including their discovery and download as software modules and management in the ITS station, is a crucial facility.

Protocols for data transmission between ITS stations and from ITS stations to other network nodes, such as network nodes in the core network (for example, the Internet), are included in the **ITS network & transport layer**. The effective distribution of data across geographical areas and the routing of data from source to destination through intermediary nodes are two features of ITS network protocols. Depending on the needs of ITS facilities and applications, ITS transport protocols offer end-to-end data transmission in addition to extra services like dependable data transfer, flow control, and congestion avoidance. The Internet protocol IP version 6 (IPv6) is one specific protocol used in the ITS network & transport layer. The use of IPv6 involves IPv6 packet transmission across ITS network protocols, dynamic choice of and handover between ITS access technologies, as well as IPv6 and IPv4 compatibility concerns.

For the physical and data connection levels, the **ITS access technologies layer** includes several communication mediums and associated protocols. Although most access technologies are based on wireless communication, they are not limited to a single form of media. Both internal communication (among an ITS station's internal components) and external communication (for instance, with other ITS stations) utilize access technologies. Some ITS access technologies (including GPRS, UMTS, and WiMAX) are full, non-ITS-specific communication networks that are viewed as "logical links" through which ITS data is openly transmitted for external communication.

The setup of an ITS station, cross-layer information sharing between the various levels, and other responsibilities which is out of the scope of this thesis fall within the purview of the **ITS management** organization.

The **ITS security** entity which is also out of scope for this thesis offers security and privacy services, including managing identities and security credentials, sending secure communications at various communication stack stages, and providing components for secure platforms (firewalls, security gateways, and tamper-proof hardware).

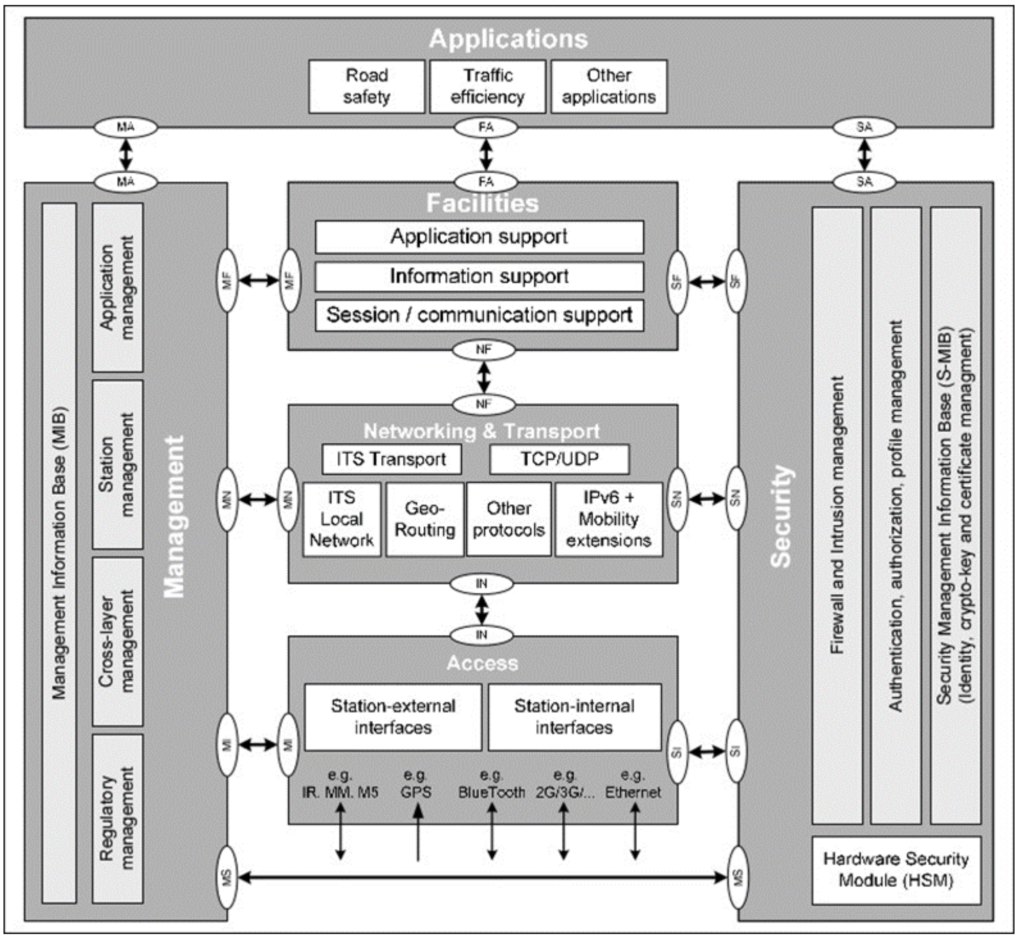


Figure 2: ITS station reference architecture [15]

2.6 The ETSI Message Set

The European Telecommunications Standards Institute (ETSI), as discussed in **section 2.5**, has proposed [12] a middleware layer (referred to as facilities) that oversees supporting the common communication requirements of many ITS services. Periodic Status Exchange (PSE) and Asynchronous Notifications (AN) are these requirements [16]. As a result, the facilities layer specifies two message types that are exchanged within the cooperative environment, and that each responds to a different requirement: the cooperative awareness message (CAM) and the distributed environmental notification message (DENM).

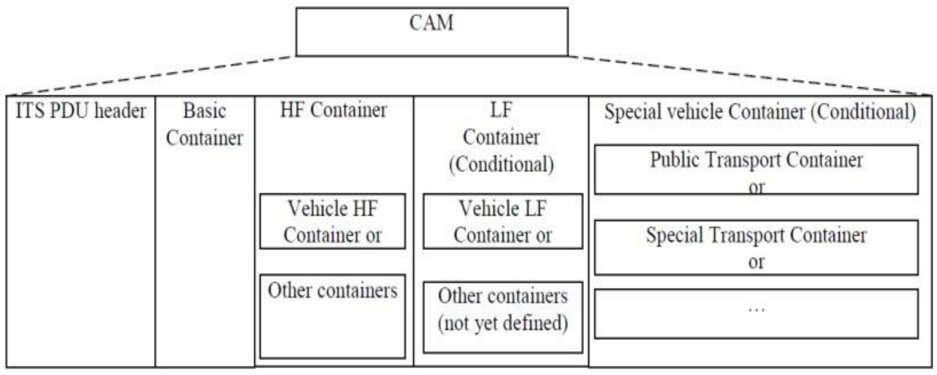

CAM
ITS PDU header
Basic Container
HF Container
Vehicle HF Container or
Other containers
LF Container (Conditional)
Vehicle LF Container or
Other containers (not yet defined)
Special vehicle Container (Conditional)
Public Transport Container or
Special Transport Container or
...

Figure 3: General structure of a CAM [12]

2.6.1 Cooperative Awareness Message (CAM)

The Cooperative Awareness Message (CAM) is a vital component in vehicular communication systems.

CAM represents a form of periodic heartbeat communication transmitted by every Intelligent Transport Systems Station (ITS-S) to its nearby peers, hence facilitating the generation and preservation of cognizance on the ITS-S characteristics within the proximity. With the use of broadcasting strategy, information may be widely disseminated and a cooperative setting where vehicles can exchange pertinent data is made possible.

While CAM communications are generally broadcast, other vehicles within the communication range can also retransmit them. This retransmission system ensures that CAM signals are received by vehicles that are farther distant from the original transmitter and broadens the coverage area. The effectiveness and reach of communication in C-ITS are improved by this collaborative approach. As a standard, ten CAM are sent per second, equivalent to a transmission interval of 100 milliseconds. But the rate at which they do so can change. The precise transmitting frequency is chosen based on several variables, including the flow of traffic, the vehicle's speed, and the dynamic demands of the C-ITS application. The necessity for current information interchange and the circumstances can cause the CAM rate to alter dynamically.

The data encompasses both the status and attribute information related to the originating Intelligent Transportation System Station (ITS-S), comprising key indicators such as its classification, spatial coordinates, velocity, and acceleration parameters as corroborated by cited literature [17].

The details of the message composition are illustrated in **Figure 3** and are detailed accordingly:

- **ITS PDU header** of Intelligent Transport Systems (ITS) comprises a protocol version, a C-ITS station identification, a message identification, and a timestamp indicating the transmission initiation.

- **Basic container** comprises the C-ITS station category, on either a vehicle or Roadside Unit (RSU), in conjunction with its corresponding geographic coordinates.

- **High-Frequency Container** stores dynamic status information associated with the vehicle, such as speed, heading, and other relevant parameters.

- **Low-frequency container** is comprised of station data that is either static or exhibits slow change over time. Examples of data within this container include lighting conditions.

- **Special Vehicle Container**, such as law enforcement patrol cars and medical ambulances, are utilized in times of crisis to provide aid and assistance to those in need.

2.6.2 Decentralized Environmental Notification Message (DENM)

DENM (Decentralized Environmental Notification Messages) are messages that are triggered by critical events and disseminated to road users to alert them of potential hazards. vehicles or infrastructure equipment often send DENM notifications to all adjacent cars within their communication range. This broadcasting strategy enables the widespread distribution of critical environmental alerts and guarantees that pertinent information reaches all impacted vehicles nearby.

DENM signals can also be retransmitted by other vehicles within the communication range, much like CAM can. This cooperative retransmission system ensures that DENM signals are received by vehicles farther away from the original transmitter and aids in expanding the coverage area.

The effectiveness and reach of communication in C-ITS for disseminating important environmental notifications are improved by this strategy.

Depending on the individual event or environmental state being reported, different DENM signals may be sent at different intervals. The transmission of DENM is prompted by certain events or recognized environmental circumstances, unlike CAM, which is transmitted regularly. When DENMs are issued depends on the occurrence of pertinent events or modifications to the environmental conditions.

The communication transmissions encompass illustrative details on the communicated occurrence, which may include its classification, geographical point of origin, and underlying rationale, and may be recurrently disseminated contingent upon the relevance of the event [17].

As depicted in **Figure 4** and **Figure 5**, the composition of the message structure entails the following elements:

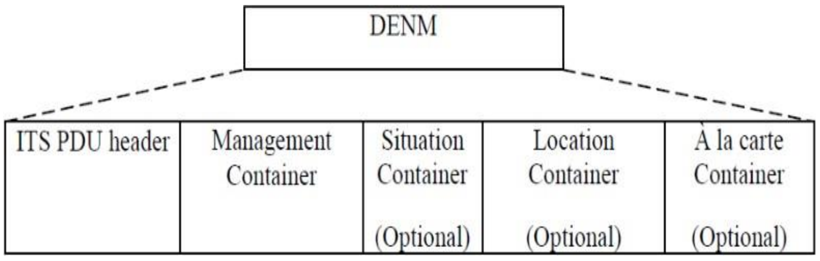
- **ITS (PDU) header** characteristics and functions are the same as for CAM.

- **Management container** comprises fundamental details on the indicated occurrence, such as the location, temporal marking, etc.

- **Situation Container** situation encapsulates additional details on the detected event, which may encompass the type of the event, cause code, and other relevant information.

- The "**Location Container**" encompasses the geographical coordinates of the pertinent occurrence.

- **"A la carte container"** is a container that includes supplementary customizable data that is valuable for specific application logic.



DENM
ITS PDU header
Management Container
Situation Container (Optional)
Location Container (Optional)
À la carte Container (Optional)

Figure 4: General structure of a DENM [12]

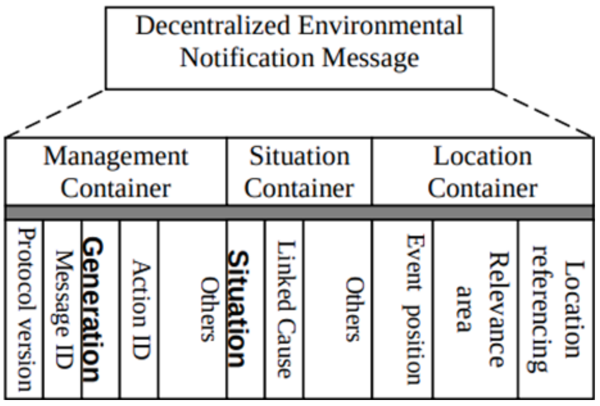


Figure 5: DENM container's properties [12]

Other C-ITS standardized messages as illustrated in **Figure 6** include "In-Vehicle Information" **(IVI)**. It describes the delivery of information and services within a moving vehicle. Dynamic speed restrictions are included in the ISO 19321-standardized coding of traffic signs, which enables moving vehicles to comprehend and show pertinent information to drivers. Additionally, the Signal Phase and Timing/Map **(SPAT/MAP)** protocols, as defined by SAE-J2735 and ISO 19091, provide for intersection signaling and the sharing of information about intersection geometry and signal timing. MAP data contains stop lines and lane information, enabling precise placement and lane direction. IVI and SPAT/MAP both offer internal vehicle transmission as well as external source communication, whether it is through specialized short-range communication or internet access, providing the availability of up-to-date information to improve driving safety and effectiveness.

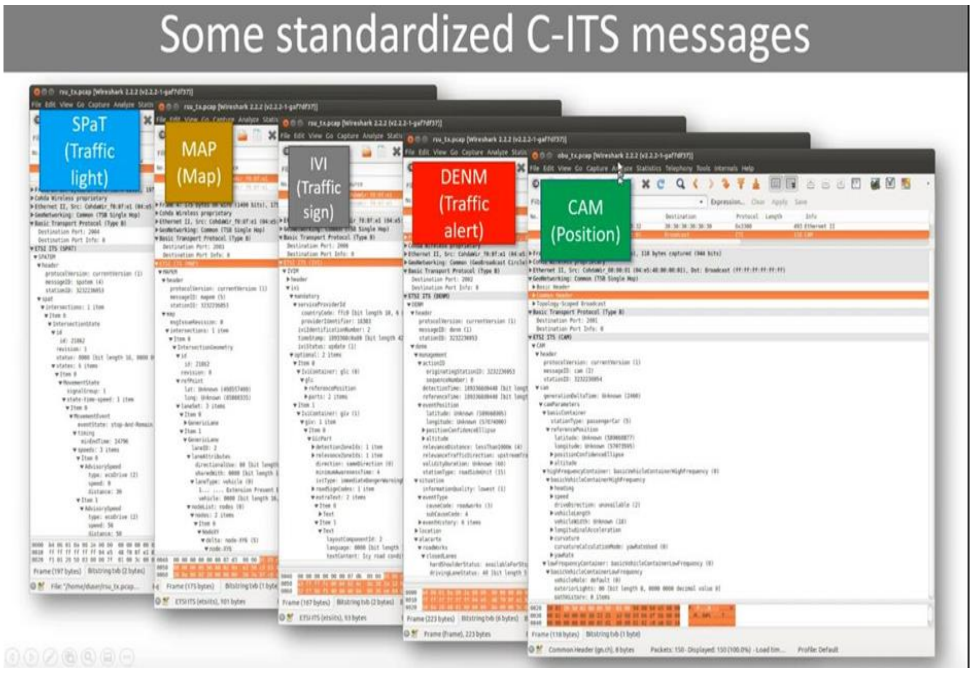


Figure 6: Standardized C-ITS Messages [18]

2.7 Cooperative Environment

Concurring to ETSI rules [12], the agreeable environment lodging a C-ITS framework is made up of the following ITS sub-systems:

- Vehicle sub-system
- Roadside sub-system
- Personal sub-system
- Central sub-system

Each of these sub-systems contains ITS stations of its domain. Some ITS stations, according to their functional requirements, do not employ the full C-ITS architecture [19].

2.7.1 Vehicle sub-system

The present domain consists of motorized road vehicles that possess the capacity to offer the complete C-ITS architecture, thereby facilitating their communication with other ITS stations as depicted in **Figure 7**.

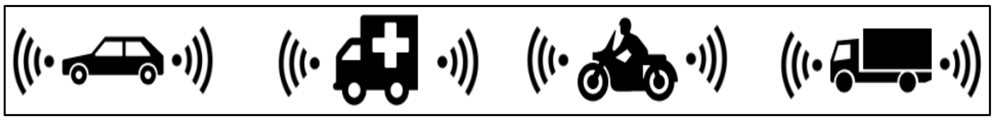


Figure 7: Vehicle Subsystem - regular and emergency vehicles

2.7.2 Roadside sub-system

The roadside sub-system is a component within the transportation infrastructure that is designed to enhance functionality and safety along roadways.

The present domain encompasses RSUs possessing the capacity to implement the comprehensive C-ITS architecture, and as such, facilitating their communication with other Intelligent Transportation System (ITS) stations. As depicted in **Figure 8**, these ITS-Station (ITS-S) are equipped with a border router, which facilitates communication between the local sub-system and the central sub-system through a distinct protocol stack.

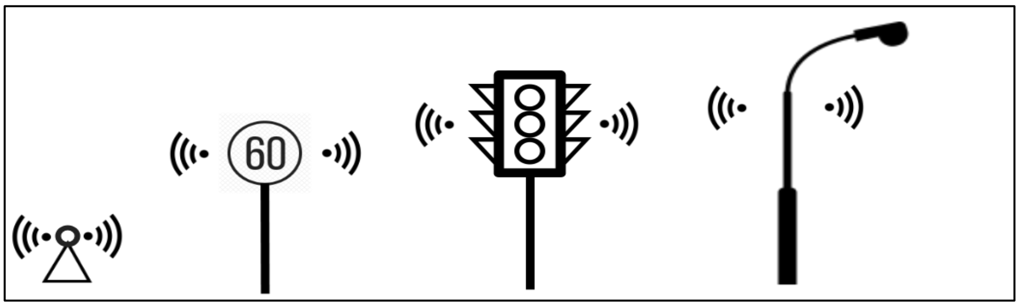


Figure 8: Roadside Subsystem - RSU, Speed Limit Sign RSU, Traffic Light RSU, Lamp Post RSU

2.7.3 Personal sub-system

Figure 9 shows a set of portable electronic devices that possess the capacity to offer essential application and communication features to enable interaction with Intelligent Transport Systems (ITS) stations. The devices are referred to as Personal Intelligent Transportation Systems stations, which may encompass Personal Digital Assistants, mobile phones, and similar handheld electronic gadgets.

Figure 9: Personal Subsystem: A person carrying a mobile phone (personal ITS-S).

2.7.4 Central sub-system

The domain as illustrated in **Figure 10** shows various systems that collate information from the Communication and Information Technology Systems (C-ITS) environment and are specifically intended to be utilized by a central C-ITS entity to enable macro traffic management.

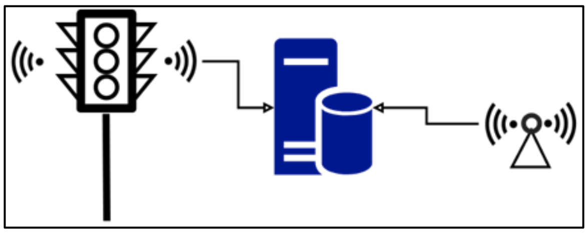


Figure 10: Central Subsystem architecture: RSUs communicating with central traffic management.

2.8 Application Classes

The present discussion pertains to the classification of Cooperative Intelligent Transport Systems (C-ITS) applications into various categories, herein referred to as "C-ITS Application Classes".

The efficacy of a cooperative environment rests upon the extent to which it enables applications to bolster the cognitive capacities of drivers on the road. The establishment of a collaborative setting is facilitated by the implementation of Cooperative Intelligent Transport Systems (C-ITS) applications that operate within each Intelligent Transport System Station (ITS-S) entity.

As explained by the European Telecommunications Standards Institute's (ETSI) guidelines [20], Cooperative Intelligent Transport Systems (C-ITS) applications are categorized into four distinct application classes, namely Road Safety, Traffic Efficiency, Local Services, and Internet Services [21]. The extent to which communication services are utilized by various applications governs the degree of stringency imposed by application classes concerning performance parameters such as reliability, security, and latency. Each application is assigned to a specific application class and subsequently falls under a distinct group within that class, if relevant.

2.8.1 Road Safety Class

There exist driving assistance applications that function to manage emergency scenarios, which can be categorized into two distinct sub-groups:

- **cooperative awareness** which involves collaborative actions on micro traffic management, thus associated with discrete driving choices made by individual drivers.

- **road hazard** warning system that enables drivers to perceive the environmental conditions prevalent in their surroundings and remain cautious of potential threats that may arise.

2.8.2 Traffic Efficiency Class

Traffic management applications are implemented to enhance the capacity and utilization of road infrastructure. This grouping can be further split into two distinct sub-categories.

speed management systems significantly benefit the driver in selecting the most optimal cruising speed.

cooperative navigation is to effectively collaborate in macro traffic management, which pertains to the driving actions undertaken by a large population.

Class	Group	Name
Road Safety	Road Hazard Warning	Electronic emergency brake light
		Weather conditions
		Slow or stationary vehicle(s)
		Traffic jam ahead
		Hazardous location
		Road works
		Signal violation intersection safety
	Cooperative Awareness	Emergency vehicle approaching
		In-vehicle signage
Traffic Efficiency	Speed Management	In-vehicle speed limits
		Green light optimal speed advisory (GLOSA)
		Shock wave damping
	Cooperative Navigation	Traffic signal priority request by designatedvehicles
		Probe vehicle data

Table 1: Day 1 Services Bundle Classification [22]

The C-ITS Platform, initiated in November 2014 by the European Commission services (DG MOVE), serves as a strategic entity that aims to delineate investment plans conducive to the emergence of business models. One of the primary objectives of the said platform is to foster interoperability among stakeholders and deliberate public-private stakeholder cooperation.

The initial stage involved the convergence of both public and private entities, encompassing all major stakeholders throughout the value chain. This cohort collectively comprised public authorities, vehicle manufacturers, suppliers, service providers, as well as telecommunications enterprises, among others.

The conclusive outcome was a report [21], delineating a communal perspective towards facilitating the seamless implementation of Cooperative Intelligent Transport Systems within the European Union. This study presents a comprehensive technical framework that is essential for the successful implementation of Cooperative Intelligent Transport Systems (C-ITS). Within this framework, a compilation of Day 1 Services bundles has been included, which are anticipated to offer significant societal benefits and are expected to be available soon due to the advanced stages of technology development. The categorized consolidated services have been exemplified following the tabular format presented in **Table 2**.

Moreover, ETSI has delineated specific criteria [23] concerning certain applications incorporated within the package of services offered on the initial day, as presented in **Table 1**.

Services	**V2X Technology (V2V, V2I/I2V, V2P)**	**Critical Actuation Time**	**Minimum Message Broadcast Frequency (1Hz = 1 message/second)**	**Messages Supported**
Electronic emergency brake light	V2V	100 ms	10Hz	CAM/ DENM
Slow or stationary vehicle(s)	V2V + V2I + I2V	100 ms	10Hz	CAM
Hazardous location	V2I + I2V	n.d.	10Hz	DENM
Road works	I2V	100 ms	2Hz	DENM
Signal violation intersection safety	I2V	100 ms	10Hz	CAM/ DENM
Emergency vehicle approaching	V2V + V2I + I2V	100 ms	10Hz	DENM
In-vehicle signage	I2V	500 ms	1Hz	CAM
In-vehicle speed limits	I2V	n.d.	1-10Hz	CAM/ DENM

Table 2: Day 1 Services Criteria and requirements [22]

Depicted in **Figure 11** shows several Day 1 services of which a handful of them relating to speed management will be briefly explained alongside their service classification.

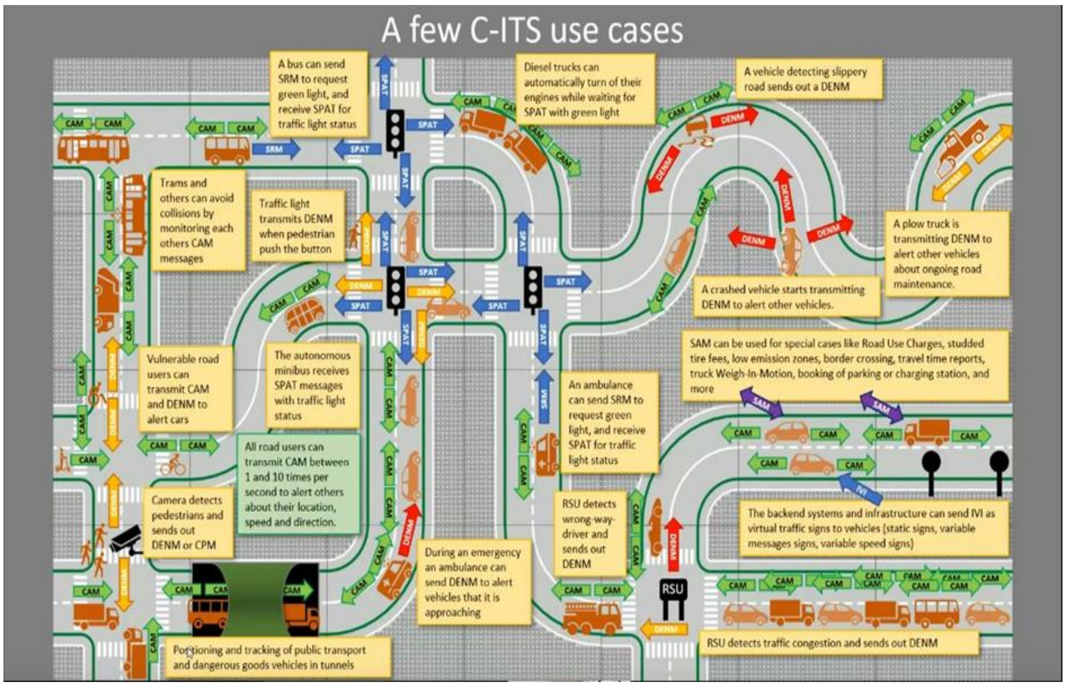

A few C-ITS use cases
A bus can send SRM to request green light, and receive SPAT for traffic light status
Diesel trucks can automatically turn of their engines while waiting for SPAT with green light
A vehicle detecting slippery road sends out a DENM
Trams and others can avoid collisions by monitoring each others CAM messages
Traffic light transmits DENM when pedestrian push the button
A plow truck is transmitting DENM to alert other vehicles about ongoing road maintenance.
A crashed vehicle starts transmitting DENM to alert other vehicles.
SAM can be used for special cases like Road Use Charges, studded tire fees, low emission zones, border crossing, travel time reports, truck Weigh-In-Motion, booking of parking or charging station, and more
Vulnerable road users can transmit CAM and DENM to alert cars
The autonomous minibus receives SPAT messages with traffic light status
An ambulance can send SRM to request green light, and receive SPAT for traffic light status
All road users can transmit CAM between 1 and 10 times per second to alert others about their location, speed and direction.
Camera detects pedestrians and sends out DENM or CPM
RSU detects wrong-way-driver and sends out DENM
The backend systems and infrastructure can send IVI as virtual traffic signs to vehicles (static signs, variable messages signs, variable speed signs)
During an emergency an ambulance can send DENM to alert vehicles that it is approaching
RSU
RSU detects traffic congestion and sends out DENM
Positioning and tracking of public transport and dangerous goods vehicles in tunnels
CAM
SPAT
SRM
DENM
SAM
IVI

Figure 11: C-ITS Use Cases [18]

2.8.3 Road Safety Services - Road Hazard Warning

Electronic Emergency Brake Light

Improves safety within a highly congested driving environment through the provision of timely alerts to the driver regarding an imminent hard-braking incident ahead. The primary objective is to prevent incidents of rear-end collisions. These collisions may arise in instances where a vehicle traveling ahead applies sudden braking, particularly in circumstances characterized by high-density traffic or compromised visibility, such as adverse weather conditions or obstructions resulting from other vehicles. Specifically, the focus centers on implementing measures that mitigate the occurrence of these potentially dangerous circumstances on highways.

As seen in **Figure 12** a driver shall receive prior notification prior to detecting the prompt deceleration of the vehicle in front, particularly if there are other vehicles in motion between the said vehicles. This mechanism is achieved through the dissemination of a self-initiated emergency brake occurrence to proximal automobiles (V2V) and neighboring infrastructure (V2I).

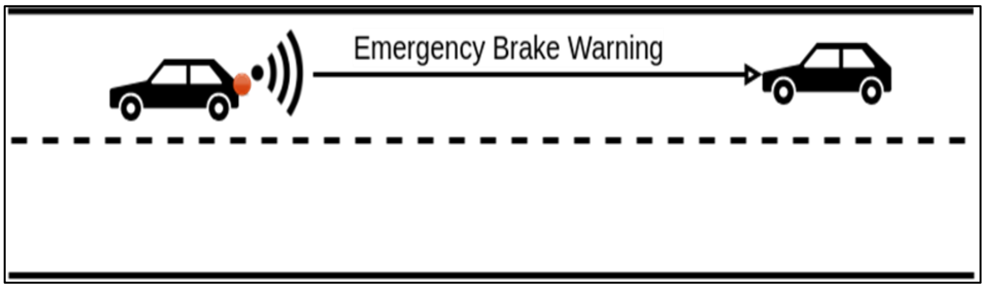


Figure 12: Electronic Emergency Brake Light usage example

Slow of Stationary Vehicle(s)

The I2V system is designed to alert drivers of a slow-moving vehicle's presence on a particular roadway. This feature depicted in **Figure 13** is particularly advantageous in mountainous terrain that comprises winding roads that exhibit obscured corners which severely diminish the driver's line of sight. This application, when integrated with an in-vehicle routing application, has the potential to enhance traffic flow by prompting the consideration of alternate routes.

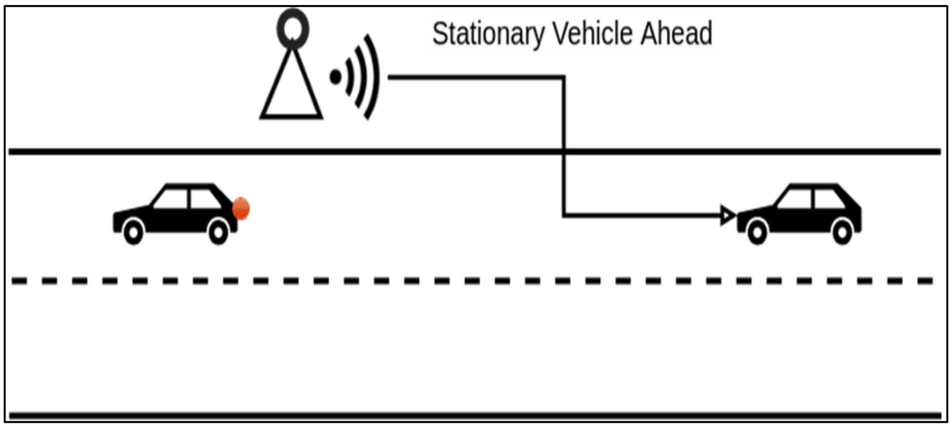


Figure 13: Example stationary vehicle ahead scenario

2.8.4 Traffic Efficiency Services - Speed Management

In-vehicle Speed Limits

A proficient RSU disseminates the current local speed limits (regulatory and contextual) to nearby vehicles through the I2V communication technology. The foremost objective is to enhance road safety. Examples are depicted in two-grouped images below in **Figure 14**.

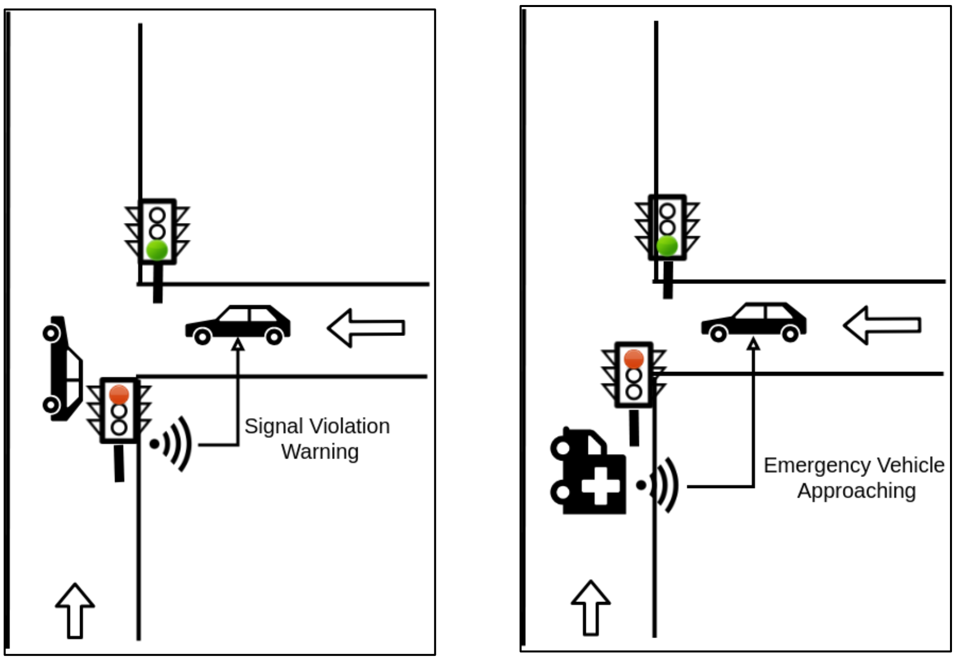


Figure 14*:* Signal Violation Intersection Safety Emergency Vehicle Approaching example scenario.

Green light Optimal Speed Advisory (GLOSA)

The Green Light Optimal Speed Advisory (GLOSA) systems facilitate drivers with optimal speed advisories, enabling them to traverse traffic lights within the green interval. There are two divergent methodologies in this area of study. In a singular-segment approach, the system is designed to furnish vehicles with the most suitable speed for the adjacent segment before the nearest traffic signals.

A multi-segment strategy entails the consideration of numerous signals along the course of a vehicle's trajectory. According to RSU's I2V system, a speed advisory pertinent to a specific area is communicated to drivers immediately before encountering a traffic light when approaching a single segment. In the multi-segment approach, Roadside Units (RSUs) disseminate speed advisories through Infrastructure-to-Vehicle (I2V) communication upon receipt of individual vehicle routes via Vehicle-to-Infrastructure (V2I) communication. This approach considers the specific route of each vehicle.

An instance of Green Light Optimized Speed Advisory (GLOSA) was executed in VSimRTI, as reported in reference [24]. The demonstration presented a noteworthy 80% decline in the duration of stops, referring to instances when the automobile remains still. Furthermore, the trial showed indications of fuel consumption reduction of up to 7% in high-traffic conditions.

Shock Wave Damping

The occurrence of disturbances within heavily congested traffic conditions (e. g the abrupt instances of stopping or acceleration and changing lanes produce far-reaching shock waves that cause sustained interference with the movement of vehicular traffic). The implementation of Shock Wave Damping techniques may be effective in mitigating the deleterious effects associated with the harmonization of traffic flows. By hindering the genesis and progression of shock waves, this approach may be instrumental in reducing traffic congestion and improving overall traffic efficiency. To accomplish this goal, Roadside Units (RSUs) located within a designated zone collect traffic-related data (Vehicle-to-Infrastructure - V2I) that is pertinent to that zone. This information is subsequently transmitted to the traffic management system's central sub-system which utilizes this data to compute an optimal driving speed advisory. The speed advisory that is computed can be disseminated through I2V communication for provision to the driver.

2.8.5 Traffic Efficiency Services - Cooperative Navigation

Traffic Signal Priority Request by Designated Vehicles

An emergency transport vehicle communicates its precise location and desired projected route to the road traffic regulatory Remote Service Units (RSUs), utilizing Vehicle-to-Infrastructure (V2I) technology. The central sub-system of the traffic management infrastructure coordinates the traffic regulatory signage to prioritize the itinerary of emergency vehicles.

Probe Vehicle Data

The transmission of Cooperative Awareness Messages (CAM) is a continuous process by vehicles that renders crucial information concerning their current position, rate of speed, atmospheric conditions, and other pertinent details. The messages are received by Roadside Units (RSUs) that belong to the Vehicle-to-Infrastructure (V2I) communication paradigm. These RSUs can transmit the messages to the traffic management infrastructure, which constitutes the central sub-system. This communication process facilitates the real-time collection of crucial road data statistics. Moreover, it enables the dynamic adaptability of road infrastructure. An illustration of this can be observed in the process of dynamic traffic light control, which necessitates the presence of traffic detection for its operation. The absence of detection systems restricts the possibility of implementing solely predetermined timing mechanisms for the green, yellow, and red traffic signals. Utilizing sustained monitoring of an intersection approach, the C-ITS system remains cognizant of the presence of vehicles as they frequently transmit Cooperative Awareness Messages (CAM), subsequently enabling the system to make necessary adjustments to optimize traffic light synchronization. **Figure 15** shows mostly the flow and interactions encountered within a Traffic Efficiency Service

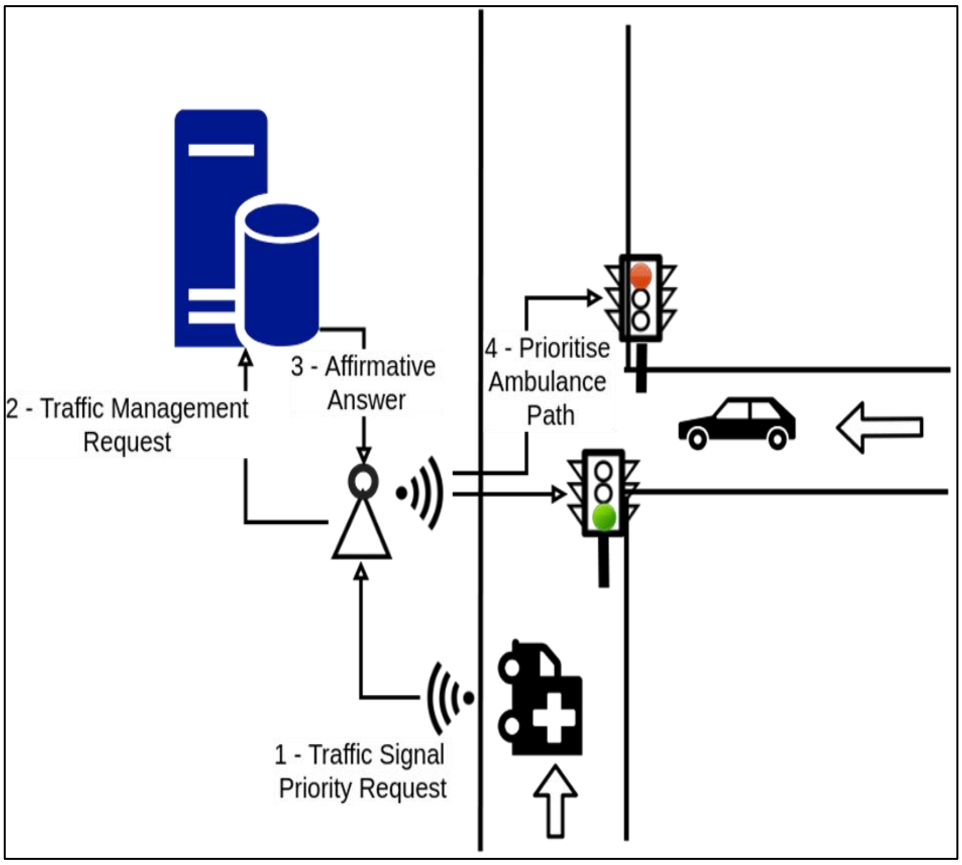

3 - Affirmative Answer
2 - Traffic Management Request
4 - Prioritise Ambulance Path
1 - Traffic Signal Priority Request

Figure 15: Traffic Signal Priority Request by Designated Vehicles example scenario

2.9 Network architecture of ITS stations

Generically, both internal and external networks are included in the network design of every C-ITS framework. External networks connect ITS stations to other instances or interconnect ITS stations among themselves. The subsequent third-party networks are recognized:

- • ITS ad hoc network.
- Access networks (ITS access networks, open access networks, and closed access networks).
- • Core network (e.g., the Internet).

An internal network connecting the ITS station's parts is another option for an ITS station. The multiple networks will offer a range of use cases for commercial applications, entertainment, and traffic efficiency. However, it is assumed that not all applications and use cases would have their communication needs met by a single network. Rather, combinations of networks using various ITS access and networking technologies are envisaged. The ITS network design may be implemented in many scenarios to adjust to certain economic and governmental circumstances and to assist the implementation of ITS gradually. Typical high-level network architecture employed in most cases is as seen in **Figure 16**.

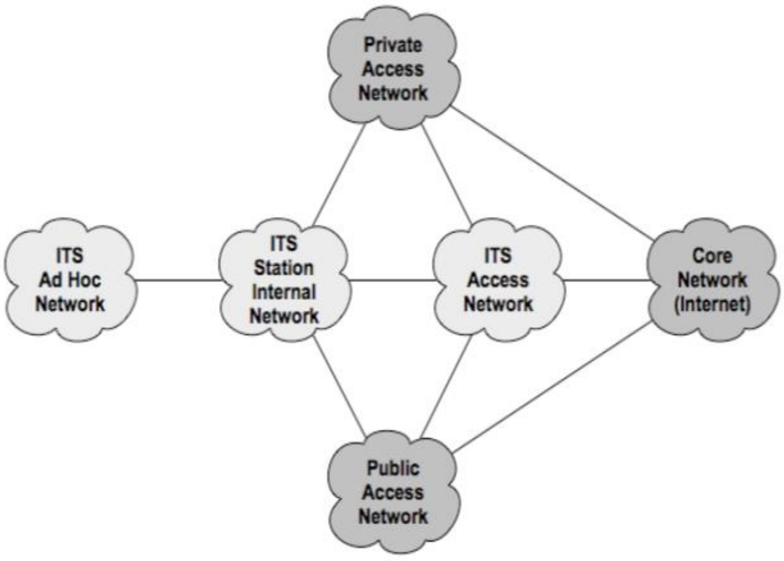


Figure 16: High-Level Network Architecture [45]

With the scope of our implementation geared towards internal networks, and with a few modifications made specifically for Intelligent Transport System Communications (ITSC), the networking and transport layer combines features from the OSI network layer and the OSI transport layer:

These are the Network protocols that are presently supported by ITS-G5 among the several protocols:

- The GeoNetworking protocol will be detailed in the subsequent paragraph.
- IPv6 networking [15] with support for network mobility (NEMO) as stated in RFC 3963 [25] or alternative methods depending on the deployment environment, as specified in RFC 6275 [26].
- Support for the switch to IPv6 from IPv4 [27].

The Transport protocols that ITS-G5 supports are as follows:

- BTP, or Basic Transport Protocol [28].
- UDP (User Datagram Protocol), as described in RFC 768[29].
- The TCP protocol as described in RFC 793[30].

A common option is the widely used UDP/TCP protocols. Although BTP was the protocol used on the most recent implementations mentioned in this thesis, ETSI ITS-G5 primarily employs it as its preferred transport protocol. For instance, BTP is utilized on top of the GeoNetworking protocol, which also incorporates UDP and TCP. An overview of GeoNetworking and IPv6 integrated with an ITS station is shown in **Figure 17**

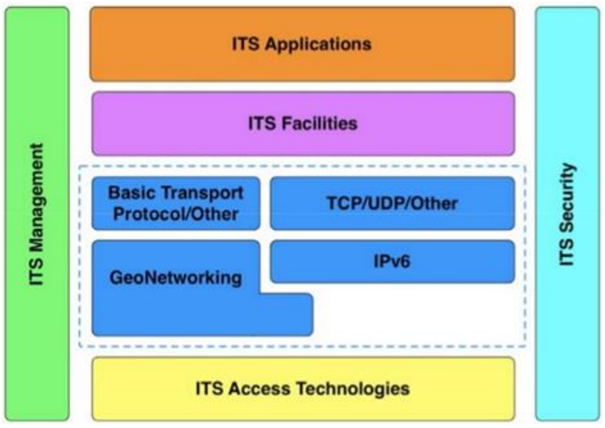


Figure 17: Combined GeoNetworking and IPv6 stack in an ITS station [31]

GeoNetworking: The ETSI GeoNetworking protocol regulates the transmission of data packets from a source node to a destination node, which can be either a single node (GeoUnicast), all nodes, or any node within a geo-area (GeoBroadcast/GeoAnycast), all nodes in a one-hop/n-hop neighborhood (single-hop/topologically scoped broad- cast). Every ad hoc router has packet buffers for location service, store-carry-and-forward, and forwarding algorithms, as well as a location table that keeps track of the positions of its known neighbors and is utilized to make forwarding decisions.

Two packet transport methods are relevant for use cases involving traffic efficiency and safety: single-hop broadcast for the transmission of periodic CAMs, and geo-broadcast for the multi-hop distribution of event-driven messages (DENMs) within a geo-area.

To distribute messages inside a geo-area, the ETSI GeoNetworking protocol offers three primary forwarding algorithms: Simple Geo-Broadcast and contention-based forwarding serve as the basic forwarding schemes, and advanced forwarding combines the two base schemes and includes several protocol techniques to enhance their performance.

A thorough discussion of Geo-Broadcast and the other forwarding algorithms can be found at [32].

Access Layer

The media can be accessed through the access layer. The PHY and Data Link Layer (DLL) are both included in this layer.

the initial person in charge of making a physical connection to the communication medium. The DLL can be divided into two sub-layers, the MAC layer, and the LLC layer. The MAC layer is in charge of controlling access to the media, while the LLC layer is in charge of logical link control.

Despite being based on IEEE 802.11a, this standard was developed with the following goals in mind to focus on and support vehicle scenarios:

- Increasing the maximum operating distance (by around 1 kilometer);
- High network node mobility and speed.
- A method for reducing and controlling the Multipath effect, which is the occurrence of multiple signal echoes.

- Regarding the variety of ad-hoc networks present in these contexts, try to ensure the best QoS.

These objectives are accomplished by several features enforced by this technology. One such feature is the channel distribution (10MHz channel spacing) allocated inside the 5.9GHz spectrum for the ITS-G5 standard [34]. There are also some pertinent channel properties. It is significant to note that CAM, which is these implementations' primary application emphasis, is conveyed via the G5-CCH channel.

Incidentally, the MAC protocol of IEEE 802.11-2012 [33] and the constrained bandwidth of ITS-G5 can cause the data load on the wireless channels to occasionally exceed the capacity. Decentralized Congestion Control (DCC) techniques, as outlined in TS 102 687[35], are thus necessary for ITS-G5 stations to manage channel load and prevent unstable system behavior.

Physical Layer (PHY)

Our framework is based on the 802.11p technology. The fundamental concept is to segment the frequency spectrum into smaller subchannels (subcarriers). The high-rate data stream is divided into several lower-rate data streams that are sent via several narrow-banded subcarriers simultaneously. There are 52 subcarriers, of which 4 are pilot carriers and 48 are used for data. The eight variable transfer rates that the OFDM PHY layer supports are attained by utilizing various modulation techniques and coding rates.

• The physical layer convergence procedure (PLCP) protocol data unit (PPDU), which is the final PHY packet ready for transmission, is constructed as shown in **Figure 18**.

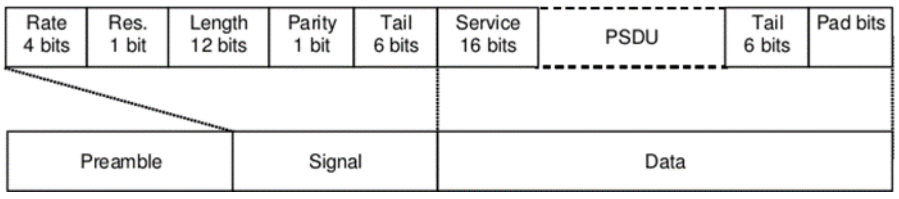


Figure 18: PHY packet - PPDU, ready for transmission [36]

Chapter 3

Framework Architecture

The primary aim of this project is to recreate and examine the functionality of C-ITS (Cooperative Intelligent Transport Systems) applications, which involve the exchange of messages between vehicles. Such message exchange can have a transformative effect on the speed of the involved vehicles in response to a hazardous location warning, or the form of an adjusted itinerary in the event of an impending traffic jam. As a result, the vehicular and network simulator needs to be closely interconnected in VANET simulation due to the potential sudden alterations in traffic dynamics. The framework must be capable of accommodating the outlined necessities:

- The capabilities of realistic simulation of vehicular mobility and V2X communications in both urban and motorway settings.

- The framework must exhibit the ability to engage in autonomous assessment and determination of vehicular actions, relying on the operation of C-ITS applications that operate at each station. This, in turn, allows for the dynamic modification of pertinent parameters such as velocity and trajectory in real-time.

- The present study reports on a highly innovative GUI that features real-time updating and the ability to effectively demonstrate V2X communications. This cutting-edge technology is poised to revolutionize the field of user interface design by providing users with an intuitive and efficient means of accessing and interacting with V2X communication systems.

- The aggregation of statistical data on the simulation, including the tally of messages transmitted by a particular vehicle in communication with a specific RSU, the mean velocity of a designated vehicle, and other relevant metrics.

Moreover, the subsequent characteristics are highly valued:

- The product is designed for utmost ease in utilization and setup.
- The VANET simulation is accompanied by ample documentation and examples.
- The GUI provided offers ample opportunity for the manipulation of simulation parameters.
- The existing implementations of the IEEE 802. 11p standards are readily accessible in the literature [37].
- Various implementations for domain-specific models in the vehicular environment have been made available, including those for RSUs, ITS-s, traffic light management systems, collision events. and speed advisor. In this implementation, the speed advisor application will be the use case for our framework.

3.1 Evaluation of VANET Simulators

Due to the growing interest in V2X simulation software among the scientific community, there has been a proliferation of simulation tools that effectively integrate vehicular and network simulators to enable realistic simulations for Cooperative Intelligent Transport Systems (C-ITS) applications.
After examining multiple surveys and comparative studies on simulators for vehicular ad hoc networks (VANETs) [38] [39] [40] [41][42], we employed the MS-Van3t as the simulator that meets most of our simulation requirements.

As well, we needed a mobility tool to handle the scenario generation and its integration into ns-3 and hence settled on the, widely used SUMO tool that is overall compatible with mostly all network simulators that incorporate V2X implementations.

	VEINS	VSimRTI	VANETsim	Ms-Van3t
Open Source	Yes	Yes	Yes	Yes
Implemented V2X Communication	Yes	Yes	Yes	Yes
Individual Vehicle Decision Making	Yes	Yes	Yes	No
Real-Time GUI	Yes	Yes	Yes	Yes
Statistics Gathering	Yes	No	Yes	Yes
Ease of Use and Setup	Medium	Medium	Medium	Easy
Documentation	Medium	Good	Bad	Good
V2X Examples	Good	Good	Very good	Very Good
GUI Parameter Tweaking	No	No	Yes	No
Scenario Import	Yes	Yes	Yes	Yes
IEEE 802.11p Implemented	Yes	Yes	Yes	Yes
Domain Specific Models in V2X Environment Implemented	Yes	Yes	Yes	Yes
Active Development	Yes	Yes	No	Yes

Table 3: V2X and VANET Simulators and Frameworks Comparative Overview

To assess the simplicity of their utilization and establishment, and the scope of our implementation, we have completed an installation process and sought after a pre-existing vehicular example application for ns-3 simulator, scrupulously recording our outcomes. Furthermore, an in-depth analysis of the official documentation and expeditious introductory manuals brought us to a conclusive end on the comparative overview of the simulators and their features, as described in our research findings, presented in **Table 3** and thus settled on Ms-Van3t.

The selected simulation framework can be divided into the following two components:

3.1.1 Network

The ns-3 platform, a discrete-event network simulator that facilitates the process of designing, modeling, and simulating a diverse range of communication networks is used. It presents a diverse array of capabilities for emulating network protocols, traffic intricacies, and application idiosyncrasies. The integration of ns-3 with Cooperative Intelligent Transport Systems (C-ITS) enables the modeling and assessment of communication technology efficacy within intelligent transportation systems. Researchers and developers can conduct an in-depth analysis of the efficiency and reliability of C-ITS applications such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, cooperative collision avoidance, and traffic management systems by incorporating ns-3 with C-ITS. Through this integration, the various impacts of different communication protocols, network topologies, and vehicular traffic patterns on the C-ITS applications can be examined. The process of integration provides a comprehensive platform for conducting meticulous testing and optimizing Cooperative Intelligent Transport Systems (C-ITS) solutions within a controlled and scalable setting before their implementation in the actual world.

3.1.2 Mobility

The mobility software suite referred to as SUMO [43] which facilitates a simulation of vehicular traffic through a user-friendly graphical user interface (GUI) is chosen. Its design is intended to enable the handling of large-scale road networks and was implemented from the ground up for this purpose. The software exhibits a smooth integration with OpenStreetMap and possesses the capability to import authentic road networks from global road infrastructure. These imported networks are incorporated with specific and discernable roadway features, such as traffic light synchronization, lane quantity, maximum velocity, and more.

As **Figure 19** presents a comprehensive and integrated view of the entirety of the architectural framework **Table 4** shows an overview of software used with their respective up-to-date information ranging from version to their latest release as well as their license classification.

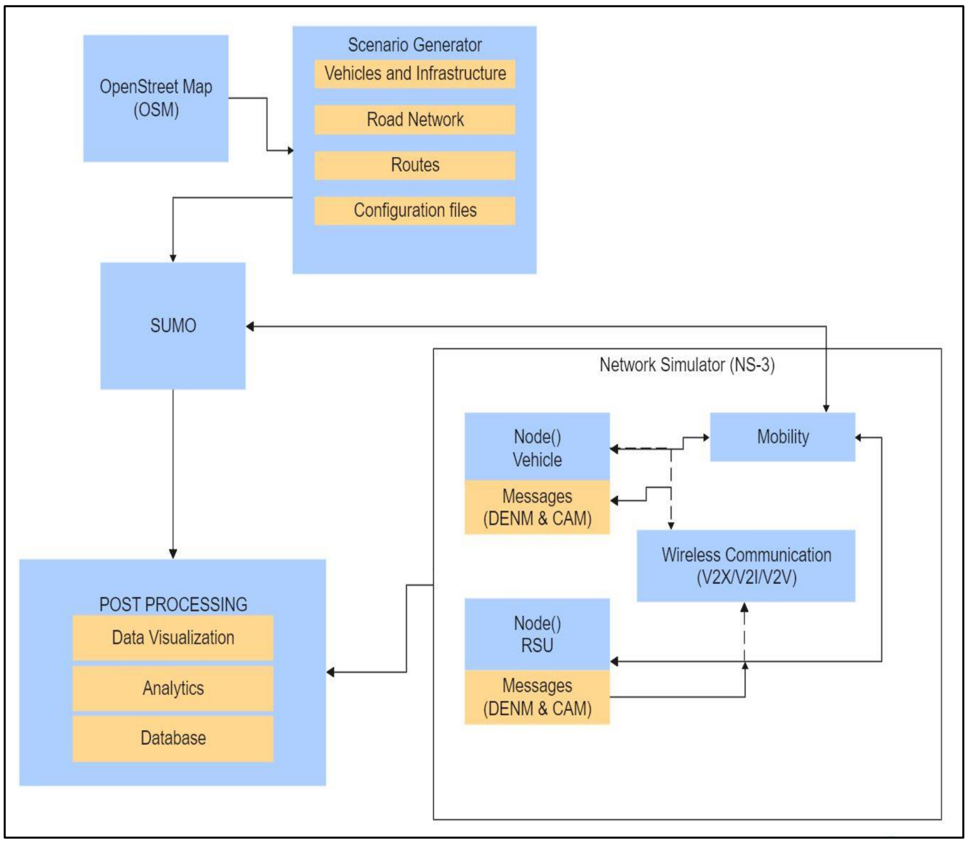


Figure 19: Architecture of the solution showcasing all interconnection of Testbed.

Software Name	Version	Latest Release	License
Ns-3	3.38	March 17, 2023	LGPL 3.0
SUMO	1.17.0	April 25, 2023	GPL 2.0

Table 4: Information on Network and Mobility tools used.

3.2 Development Environment

The entirety of the work was collated and packaged as a virtual appliance, effectively a pre-configured virtual machine image that is readily deployable. The main aim of this approach was to eradicate the expenses related to setting up, arranging, and upkeeping intricate software and hardware stacks.

The equipment was transferred to the prevalently employed format, Virtual Disk Image (VDI). The system has been pre-configured to utilize precisely 12 gigabytes of Random Access Memory (RAM) and execute at maximum capacity, utilizing 100% of the Central Processing Unit (CPU), as well as all available cores of the host machine on which it is currently operating.

The equipment encompasses a comprehensive suite of development components, inclusive of a pre-established Integrated Development Environment (IDE) and an elaborate build architecture.

The entirety of development and testing procedures were all executed on the virtual machine whose technical specifications are outlined in **Table 5**.

Operating System	Graphic Processing Unit	RAM	Solid State Drive	CPU
Ubuntu 22.04 LTS	RTX A3000	12 Gigabytes	150 Gigabytes	12 Gen Intel(R) Core (TM) i9-12950HX 2.30 GHz

Table 5: Technical Specification of Virtual Machine used.

Chapter 4

Implementation of Framework

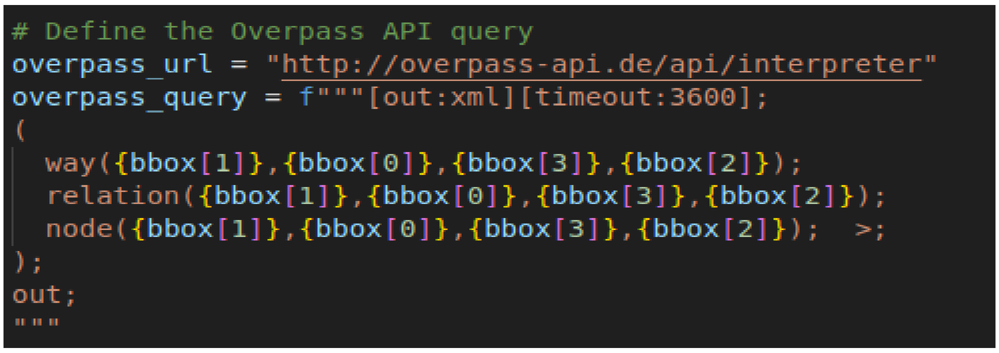
This chapter focuses on the methodology for deploying an actual road map, vehicle routing, and a C-ITS application - area speed advisor utilizing the chosen stack.

4.1 Importing a Road Map and Traffic Flow

As seen in our Testbed architecture in **Figure 19**, we trigger the scenario generator and all its interconnected objects by using the Overpass API provided by OpenStreetMap, a service that enables users to search for and get specific location data from the OSM database by simply providing the coordinates of the map area of interest With. a user interface provided to handle user inputs of the map coordinates and using a unique language called Overpass QL, it makes it simple to obtain information about anything on a map. It can process a large amount of data and provides results in several formats but for our scope, OSM file format is of interest. The procedure is as follows:

- Find a map area and injects their coordinates in the simulation framework and provide the number of vehicles to be used in the simulation.
- Run the scenario generator to generate all configuration and mobility files needed with the help of the Overpass API provided by OpenStreetMap and various SUMO tools.

Below is a snapshot of calling the API and corresponding generated files using SUMO tools and their resulting outcome as seen in **Figure 20**.

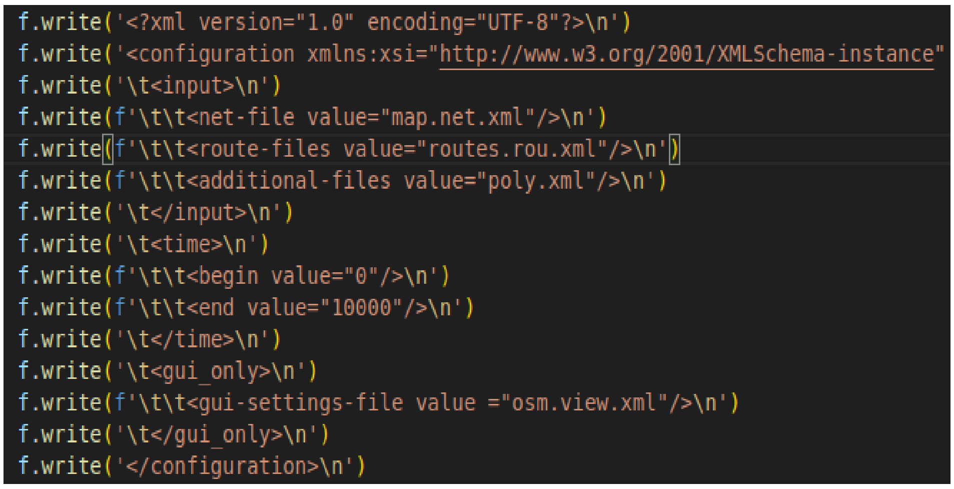


```
# Define the Overpass API query
overpass_url = "http://overpass-api.de/api/interpreter"
overpass_query = f"""[out:xml][timeout:3600];
(
  way({bbox[1]},{bbox[0]},{bbox[3]},{bbox[2]});
  relation({bbox[1]},{bbox[0]},{bbox[3]},{bbox[2]});
  node({bbox[1]},{bbox[0]},{bbox[3]},{bbox[2]});  >;
);
out;
"""
```

Figure 20: Map query using the OpenStreetMap Overpass API

The resulting files generated are consolidated in a configuration file to which the ns-3-based C-ITS application can process.

Figure 21 shows the various file composition that makes up the mobility configuration file.



```
f.write('<?xml version="1.0" encoding="UTF-8"?>\n')
f.write('<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
f.write('\t<input>\n')
f.write(f'\t\t<net-file value="map.net.xml"/>\n')
f.write(f'\t\t<route-files value="routes.rou.xml"/>\n')
f.write(f'\t\t<additional-files value="poly.xml"/>\n')
f.write('\t</input>\n')
f.write('\t<time>\n')
f.write(f'\t\t<begin value="0"/>\n')
f.write(f'\t\t<end value="10000"/>\n')
f.write('\t</time>\n')
f.write('\t<gui_only>\n')
f.write(f'\t\t<gui-settings-file value ="osm.view.xml"/>\n')
f.write('\t</gui_only>\n')
f.write('</configuration>\n')
```

Figure 21: Composition of a mobility Configuration file

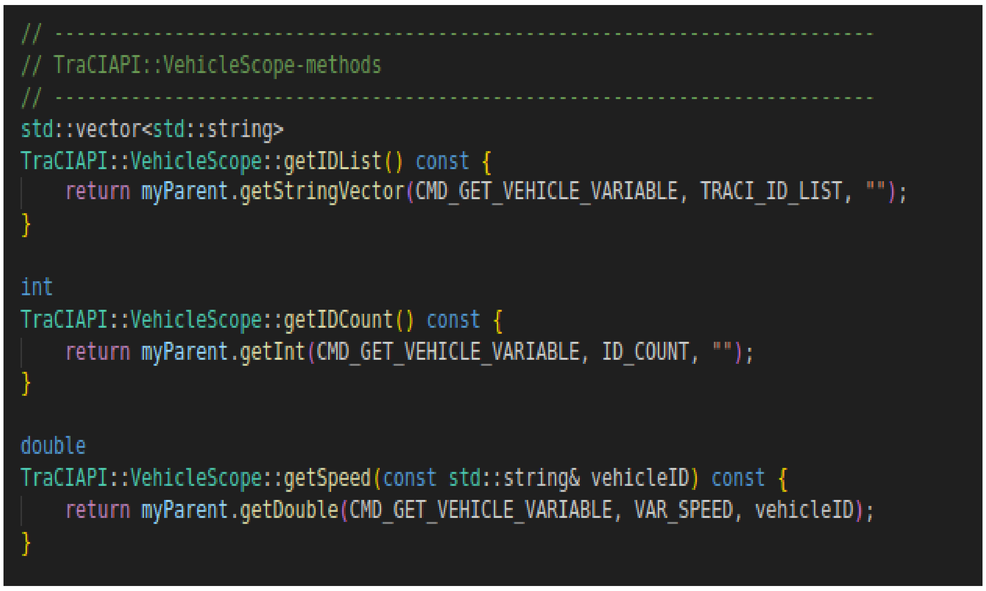
4.2 Defining C-ITS Application

There are several C-ITS applications as part of this project, however, the use case for our scope was tailored to the Area Speed Advisor also referred to as ASA as its application in Day 1 services is crucial as detailed in **Table 1** and **Table 2**.
Depending on whether it is a vehicle or an RSU, each station, represented as a node, already has a facilities layer installed, either Stationary Middleware or Vehicle Middleware. However, stations must have an ItsG5BaseService running in each one to use the available middleware and execute C-ITS programs that exchange messages.

The CAM and DENM services (based on ItsG5BaseService) that support the application need synchronous and asynchronous message dissemination, respectively, hence comply with the ASN.1 standard because it communicates with the outside world.
Instead of using any ns-3 simulated models, this application emulates n-cars, each with its own CA and DEN basic service making it possible for future hardware-in-the-loop testing and evaluation.

4.2.1 Traffic Control Interface

The TraCI Application Programming Interface (API) [43] provides access to a live road traffic simulation, by synchronizing processes between network simulators -ns-3 and mobility framework-SUMO.
As a result, C-ITS Applications have access to two different types of methods namely [43], value retrieval and state change, on which we place special emphasis for C-ITS research purposes. A few methods relevant to the scope of this implementation and information retrieved from the TRACI API are stated and explained below with some corresponding code snippets.



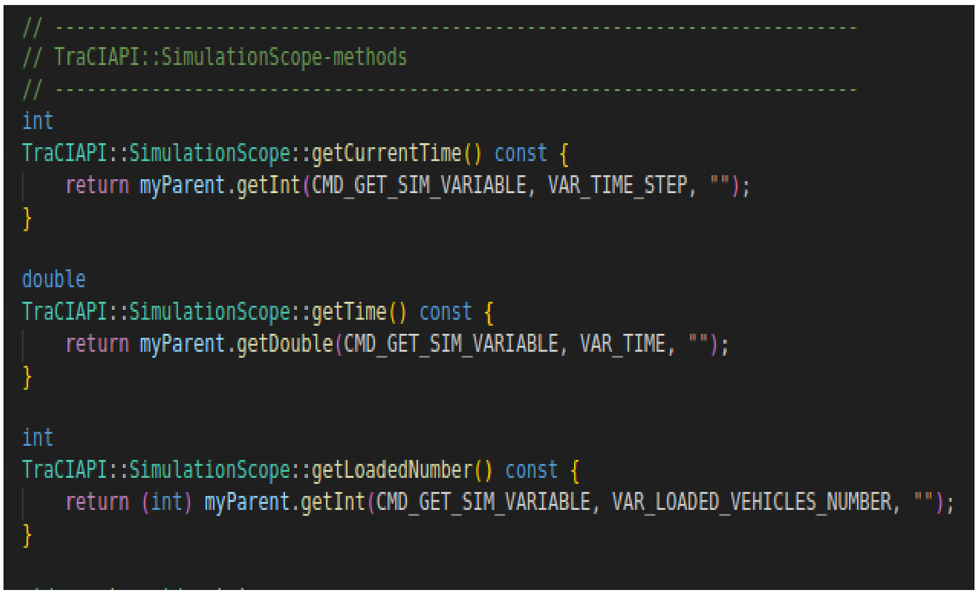
```
// ------------------------------------------------------------------------
// TraCIAPI::VehicleScope-methods
// ------------------------------------------------------------------------
std::vector<std::string>
TraCIAPI::VehicleScope::getIDList() const {
    return myParent.getStringVector(CMD_GET_VEHICLE_VARIABLE, TRACI_ID_LIST, "");
}

int
TraCIAPI::VehicleScope::getIDCount() const {
    return myParent.getInt(CMD_GET_VEHICLE_VARIABLE, ID_COUNT, "");
}

double
TraCIAPI::VehicleScope::getSpeed(const std::string& vehicleID) const {
    return myParent.getDouble(CMD_GET_VEHICLE_VARIABLE, VAR_SPEED, vehicleID);
}
```

Figure 22: TRACI API Vehicle scope sample code [44]

The provided code snippet in **Figure 22** defines methods for the TraCIAPI namespace's VehicleScope class. Here is a quick explanation of each technique:

The list of vehicle IDs for each car in the simulation is retrieved using the **getIDList()** method. The method returns a list of type **std::vector** by calling the **getStringVector()** function on the myParent object and giving pertinent arguments such CMD_GET_VEHICLE_VARIABLE, TRACI_ID_LIST, and an empty string. The **getIDCount()** method is used to calculate the total number of vehicles in the simulation. This method makes use of the myParent object's **getInt()** function and passes the variables ID_COUNT and CMD_GET_VEHICLE_VARIABLE to it. A numerical figure that represents the total number of vehicles is the result of this process. The **getSpeed(const std::string vehicleID)** function is another option that makes it possible to find the speed of a certain vehicle that is identified by its vehicleID. The **getDouble()** function on the myParent object produces a double value that represents the speed of the selected vehicle after being called with the required arguments.

These functions are a part of the VehicleScope class in the SUMO simulator's TRACI interface. They offer the capability for locating vehicle-related data, including IDs, vehicle counts, and the speeds of certain cars in the simulation.



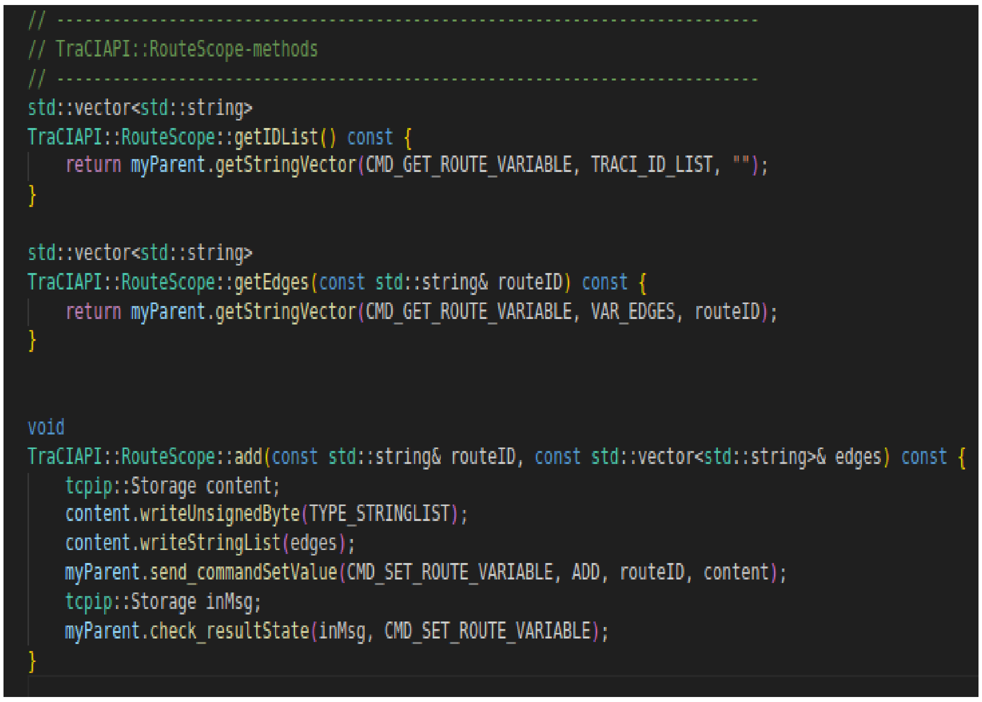
```
// ---------------------------------------------------------------------------
// TraCIAPI::SimulationScope-methods
// ---------------------------------------------------------------------------
int
TraCIAPI::SimulationScope::getCurrentTime() const {
    return myParent.getInt(CMD_GET_SIM_VARIABLE, VAR_TIME_STEP, "");
}

double
TraCIAPI::SimulationScope::getTime() const {
    return myParent.getDouble(CMD_GET_SIM_VARIABLE, VAR_TIME, "");
}

int
TraCIAPI::SimulationScope::getLoadedNumber() const {
    return (int) myParent.getInt(CMD_GET_SIM_VARIABLE, VAR_LOADED_VEHICLES_NUMBER, "");
}
```

Figure 23: TRACI API Simulation Scope sample code [44]

The current simulation time is returned as an integer by the **getCurrentTime()** method as seen in **Figure 23**. The command CMD_GET_SIM_VARIABLE, the variable VAR_TIME_STEP, and an empty string are passed to the myParent object's **getInt()** function as inputs.

Using the myParent object's **getDouble()** function with its arguments, the **getTime()** function returns the current simulation time as a double value. The number of loaded vehicles in the simulation is returned by the **getLoadedNumber()** method. This is accomplished by using the myParent object's **getInt()** function along with its parameters. The **getLoadedIDList()** method further creates a list of IDs for each loaded vehicle included in the simulation. To accomplish this, it uses the myParent object's **getStringVector()** function, passing several arguments as inputs as seen below.

These techniques offer a mechanism to retrieve various simulation-related data using the TRACI interface inside the SUMO simulator, including the time, the number of loaded vehicles, and the IDs of loaded vehicles.



```
// ---------------------------------------------------------------------------
// TraCIAPI::RouteScope-methods
// ---------------------------------------------------------------------------
std::vector<std::string>
TraCIAPI::RouteScope::getIDList() const {
    return myParent.getStringVector(CMD_GET_ROUTE_VARIABLE, TRACI_ID_LIST, "");
}

std::vector<std::string>
TraCIAPI::RouteScope::getEdges(const std::string& routeID) const {
    return myParent.getStringVector(CMD_GET_ROUTE_VARIABLE, VAR_EDGES, routeID);
}


void
TraCIAPI::RouteScope::add(const std::string& routeID, const std::vector<std::string>& edges) const {
    tcpip::Storage content;
    content.writeUnsignedByte(TYPE_STRINGLIST);
    content.writeStringList(edges);
    myParent.send_commandSetValue(CMD_SET_ROUTE_VARIABLE, ADD, routeID, content);
    tcpip::Storage inMsg;
    myParent.check_resultState(inMsg, CMD_SET_ROUTE_VARIABLE);
}
```

Figure 24: TRACI API Route scope sample code [44]

The provided code snippet in **Figure 24** defines methods for the TraCIAPI namespace's RouteScope class. Here is a quick explanation of some implementation:

Using the **getIDList()** method, you may obtain a list of all the routes' IDs. The respective arguments are passed to the myParent object's **getStringVector()** function as inputs.

By using routeID to identify the route, this function retrieves the edges connected to that route. The parameters passed to the myParent's **getStringVector()** function as inputs.

This method creates a new route using the list of edges provided, as specified by routeID, and adds it to the simulation.

The edge list is initially written to a tcpip::Storage object called content as a string list. It then dials myParent.To transmit the command CMD_SET_ROUTE_VARIABLE along with the arguments ADD, routeID, and content, use the **send_commandSetValue()** function. It utilizes myParent to finish. Check the command's result state by calling **check_resultState().**

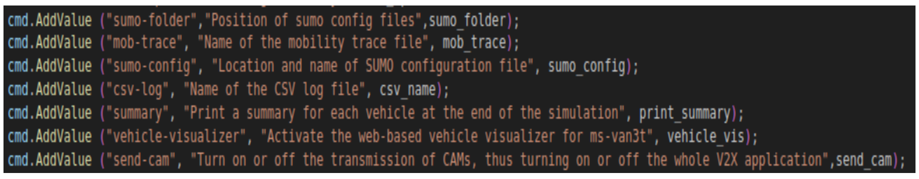
These functions are a part of the RouteScope class in the SUMO simulator's TRACI interface. They offer the ability to add new routes to the simulation as well as retrieve route data such as IDs and associated edges.

It must be stated that the above-explained implementations explained above are a few of the many relevant classes and methods imported into our framework to facilitate simulation.

4.2.2 Use case Application Development

As stated in earlier chapters, the framework was developed to simulate the C-ITS use case Area Speed Advisor. This use is generally focused on basing its implementation using 802.11p. 802.11p was created for use in areas where there is moving traffic. It allows for direct communication between cars (V2V) and between vehicles and roadside infrastructure (V2I) and runs in the 5.9 GHz frequency spectrum. It offers low-latency, high-speed, and long-distance communication for uses like avoiding collisions, enhancing traffic flow, and disseminating real-time information. 802.11p supports intelligent transportation systems and improves safety. The implementation makes it possible for vehicles to periodically broadcast CAM signals as well as RSU to periodically broadcasts DENM to warn moving vehicles to slow down. In this scenario, BTP and GeoNetworking are used to encapsulate CAM and DENM.

Overall, the application demonstrates the integration of network simulations, vehicular mobility, and V2X communication using the ns-3 library. It simulates a scenario with OBUs, a server, and dynamic nodes (vehicles) exchanging messages over an 802.11p wireless channel.



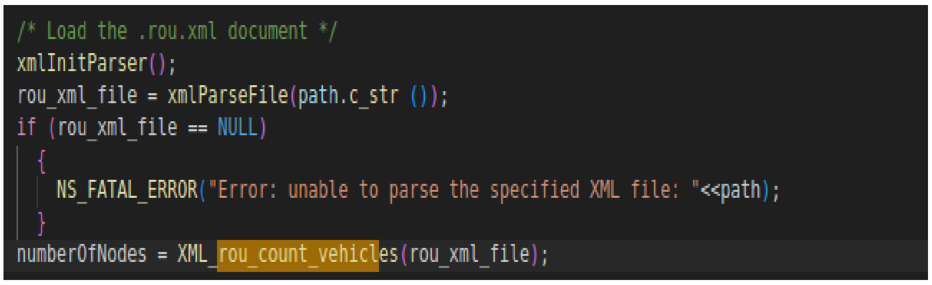
```
cmd.AddValue ("sumo-folder","Position of sumo config files",sumo_folder);
cmd.AddValue ("mob-trace", "Name of the mobility trace file", mob_trace);
cmd.AddValue ("sumo-config", "Location and name of SUMO configuration file", sumo_config);
cmd.AddValue ("csv-log", "Name of the CSV log file", csv_name);
cmd.AddValue ("summary", "Print a summary for each vehicle at the end of the simulation", print_summary);
cmd.AddValue ("vehicle-visualizer", "Activate the web-based vehicle visualizer for ms-van3t", vehicle_vis);
cmd.AddValue ("send-cam", "Turn on or off the transmission of CAMs, thus turning on or off the whole V2X application",send_cam);
```

Figure 25: Command Line Parser [44]

The code parses command line arguments using the CommandLine class as seen in **Figure 25**. Using the AddValue method, it defines many command line parameters such as **realtime**, **sumo-gui**, **sim-time**, **tx-power**, **datarate**, and so on. When launching the software, these options allow users to set different parameters. For example, --realtime enables the usage of the real-time scheduler, --sumo-gui enables the SUMO graphical user interface, --sim-time specifies the simulation length, and --tx-power and --datarate indicate the 802.11p channel transmission power and data rate, respectively.

XML File Loading:

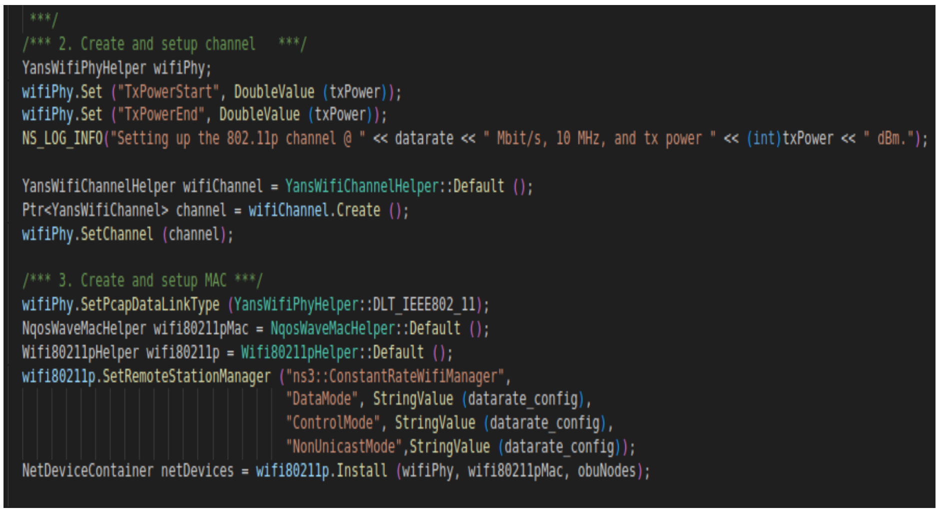
The code parses an XML file containing vehicle mobility information using the **sumo_xml_parser.h** header and related functions. The **xmlParseFile** function is called to load the XML file given by the **mob_trace** command line argument as seen below in **Figure 26**.



```
/* Load the .rou.xml document */
xmlInitParser();
rou_xml_file = xmlParseFile(path.c_str ());
if (rou_xml_file == NULL)
  {
    NS_FATAL_ERROR("Error: unable to parse the specified XML file: "<<path);
  }
numberOfNodes = XML_rou_count_vehicles(rou_xml_file);
```

Figure 26: Loading mobility (trace file) [44]

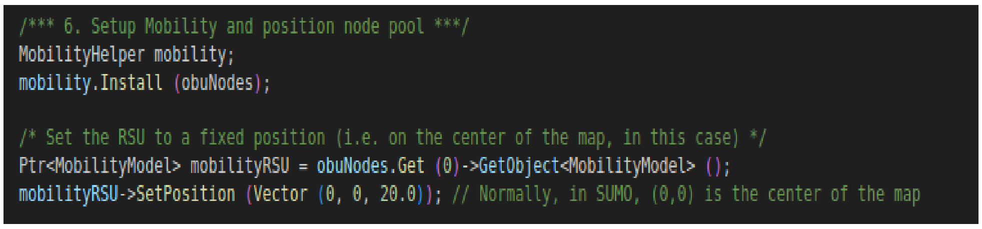
The **XML_rou_count_vehicles** method is then used to retrieve the number of vehicles from the XML file. This data is eventually utilized in the code to establish a container for **OBUs** (On-Board Units) and to configure the simulation properly.



```
 ***/
/*** 2. Create and setup channel   ***/
YansWifiPhyHelper wifiPhy;
wifiPhy.Set ("TxPowerStart", DoubleValue (txPower));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txPower));
NS_LOG_INFO("Setting up the 802.11p channel @ " << datarate << " Mbit/s, 10 MHz, and tx power " << (int)txPower << " dBm.");

YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
Ptr<YansWifiChannel> channel = wifiChannel.Create ();
wifiPhy.SetChannel (channel);

/*** 3. Create and setup MAC ***/
wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11);
NqosWaveMacHelper wifi80211pMac = NqosWaveMacHelper::Default ();
Wifi80211pHelper wifi80211p = Wifi80211pHelper::Default ();
wifi80211p.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                                    "DataMode", StringValue (datarate_config),
                                    "ControlMode", StringValue (datarate_config),
                                    "NonUnicastMode",StringValue (datarate_config));
NetDeviceContainer netDevices = wifi80211p.Install (wifiPhy, wifi80211pMac, obuNodes);
```

Figure 27: Creating and setting up network channel [44].

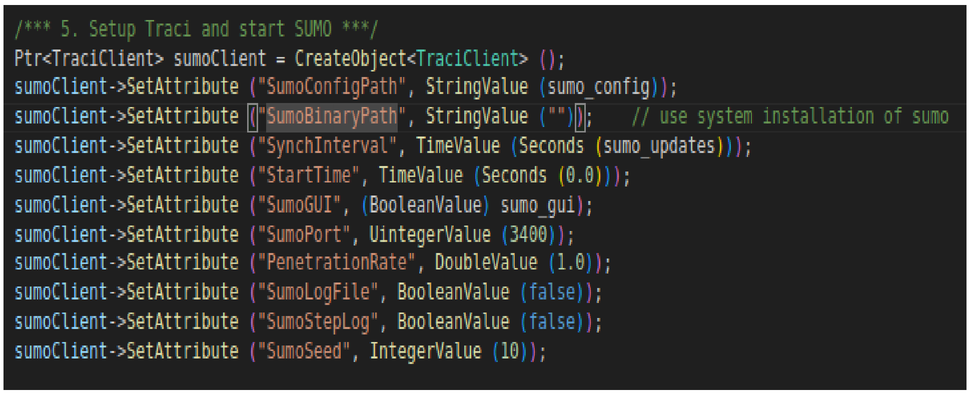
Figure 27 shows how the 802.11p channel is configured for OBUs. It creates a **YansWifiPhyHelper** object and uses the configure function to configure the transmission power. It also configures the channel properties with **YansWifiChannelHelper** and establishes a channel with the default parameters. Using the **SetChannel** method, the newly generated channel is associated with the **WiFi PHY** object.



```
/*** 6. Setup Mobility and position node pool ***/
MobilityHelper mobility;
mobility.Install (obuNodes);

/* Set the RSU to a fixed position (i.e. on the center of the map, in this case) */
Ptr<MobilityModel> mobilityRSU = obuNodes.Get (0)->GetObject<MobilityModel> ();
mobilityRSU->SetPosition (Vector (0, 0, 20.0)); // Normally, in SUMO, (0,0) is the center of the map
```

Figure 28: Mobility Setup [44]

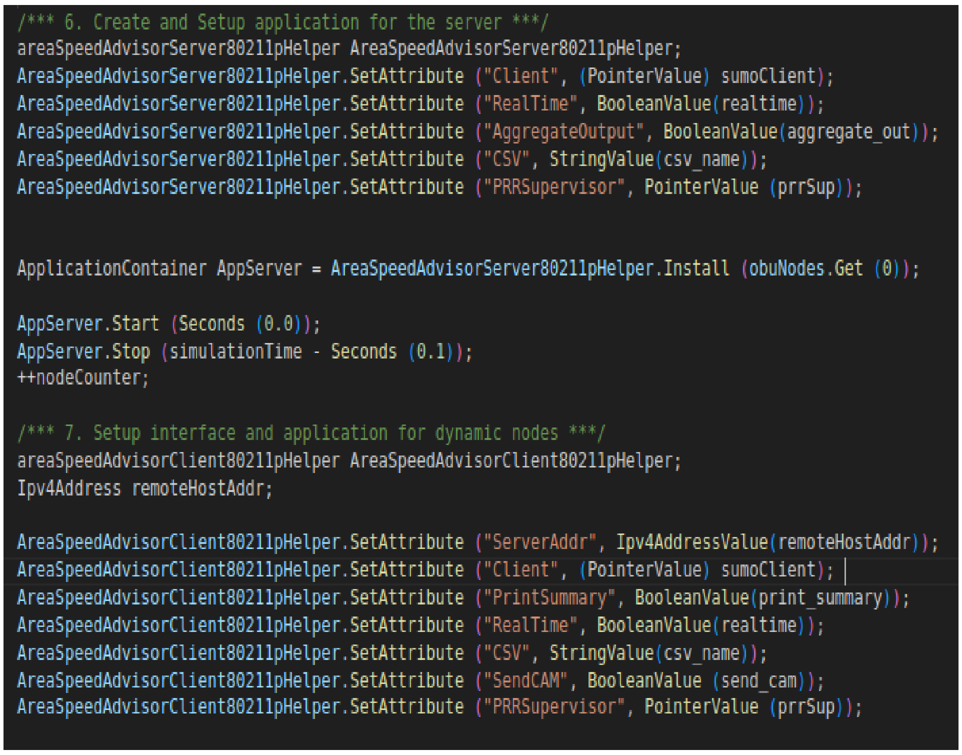
The MobilityHelper class is used by the code to install mobility models for OBUs. Illustrated in **Figure 28**, it uses the Install method and the obuNodes container to configure mobility for all nodes. It also specifically sets the RSU's (Roadside Unit's) position by retrieving the mobility model of the first node **(obuNodes.Get(0))** and passing the desired coordinates to **SetPosition**.



```
/*** 5. Setup Traci and start SUMO ***/
Ptr<TraciClient> sumoClient = CreateObject<TraciClient> ();
sumoClient->SetAttribute ("SumoConfigPath", StringValue (sumo_config));
sumoClient->SetAttribute ("SumoBinaryPath", StringValue (""));    // use system installation of sumo
sumoClient->SetAttribute ("SynchInterval", TimeValue (Seconds (sumo_updates)));
sumoClient->SetAttribute ("StartTime", TimeValue (Seconds (0.0)));
sumoClient->SetAttribute ("SumoGUI", (BooleanValue) sumo_gui);
sumoClient->SetAttribute ("SumoPort", UintegerValue (3400));
sumoClient->SetAttribute ("PenetrationRate", DoubleValue (1.0));
sumoClient->SetAttribute ("SumoLogFile", BooleanValue (false));
sumoClient->SetAttribute ("SumoStepLog", BooleanValue (false));
sumoClient->SetAttribute ("SumoSeed", IntegerValue (10));
```

Figure 29: Traci Setup [44]

Here, in **Figure 29**, the Traci (Traffic Control Interface) module and the SUMO (Simulation of Urban Mobility) traffic simulator are used to create a TraciClient object and configures many properties such as the SUMO configuration file, the SUMO binary location, the SUMO GUI option, and the connection parameters.



```
/*** 6. Create and Setup application for the server ***/
areaSpeedAdvisorServer80211pHelper AreaSpeedAdvisorServer80211pHelper;
AreaSpeedAdvisorServer80211pHelper.SetAttribute ("Client", (PointerValue) sumoClient);
AreaSpeedAdvisorServer80211pHelper.SetAttribute ("RealTime", BooleanValue(realtime));
AreaSpeedAdvisorServer80211pHelper.SetAttribute ("AggregateOutput", BooleanValue(aggregate_out));
AreaSpeedAdvisorServer80211pHelper.SetAttribute ("CSV", StringValue(csv_name));
AreaSpeedAdvisorServer80211pHelper.SetAttribute ("PRRSupervisor", PointerValue (prrSup));


ApplicationContainer AppServer = AreaSpeedAdvisorServer80211pHelper.Install (obuNodes.Get (0));

AppServer.Start (Seconds (0.0));
AppServer.Stop (simulationTime - Seconds (0.1));
++nodeCounter;

/*** 7. Setup interface and application for dynamic nodes ***/
areaSpeedAdvisorClient80211pHelper AreaSpeedAdvisorClient80211pHelper;
Ipv4Address remoteHostAddr;

AreaSpeedAdvisorClient80211pHelper.SetAttribute ("ServerAddr", Ipv4AddressValue(remoteHostAddr));
AreaSpeedAdvisorClient80211pHelper.SetAttribute ("Client", (PointerValue) sumoClient);
AreaSpeedAdvisorClient80211pHelper.SetAttribute ("PrintSummary", BooleanValue(print_summary));
AreaSpeedAdvisorClient80211pHelper.SetAttribute ("RealTime", BooleanValue(realtime));
AreaSpeedAdvisorClient80211pHelper.SetAttribute ("CSV", StringValue(csv_name));
AreaSpeedAdvisorClient80211pHelper.SetAttribute ("SendCAM", BooleanValue (send_cam));
AreaSpeedAdvisorClient80211pHelper.SetAttribute ("PRRSupervisor", PointerValue (prrSup));
```

Figure 30: Application Configuration [44]

Figure 30 shows several application configurations for the server and dynamic nodes (vehicles). To configure the server and client applications, it generates instances of **AreaSpeedAdvisorServer80211p** and **AreaSpeedAdvisorClient80211p Helpers.** The server application is installed using the **Install** method on the RSU **(obuNodes.Get(0)),** whereas the client application is installed using a loop on each dynamic node. These applications help the server and dynamic nodes exchange messages (CAMs and DENMs) over the 802.11p channel.

4.3 Post Processing

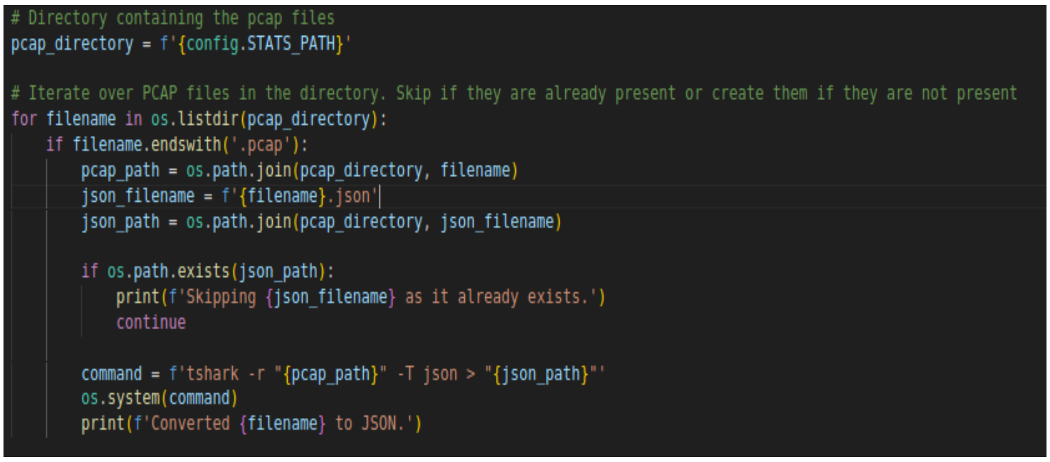
In the context of the Cooperative Intelligent Transport Systems (C-ITS) emulation framework, post-processing is essential since it permits the analysis and improvement of data produced during the emulation. The C-ITS emulation framework intends to model and assess the performance of cooperative systems in a controlled setting, helping the creation and application of cutting-edge transportation technology. Following the emulation runs, post-processing techniques are used to glean insightful information, examine system behavior, and judge the viability of C-ITS applications. Researchers and engineers may better comprehend the intricate connections within the C-ITS ecosystem by utilizing post-processing, which will improve system design and optimization and, ultimately, result in safer and more effective transportation networks.
Considering its importance as stated above, implementation was carried out to satisfy our test cases in **chapter 5.**

4.3.1 Post-processing network (PCAP) files

As PCAP files are generally examined by network tools such as Wireshark, it is quite strenuous to post-process. To ensure a seamlessly feasible way of interacting with its content and subsequently performing operations on them to be able to carry out analysis and thus generate plots and graphs from them, **Tshark** was used. As part of the Wireshark suite and its integrated modules and libraries in Python, it serves as a command-line network analyzer with powerful filtering and protocol analysis features for capturing, reviewing, and debugging network data.

One of its capabilities includes converting PCAP files to JSON files to have fields and parameters that can be directly accessed using Python for data analysis.

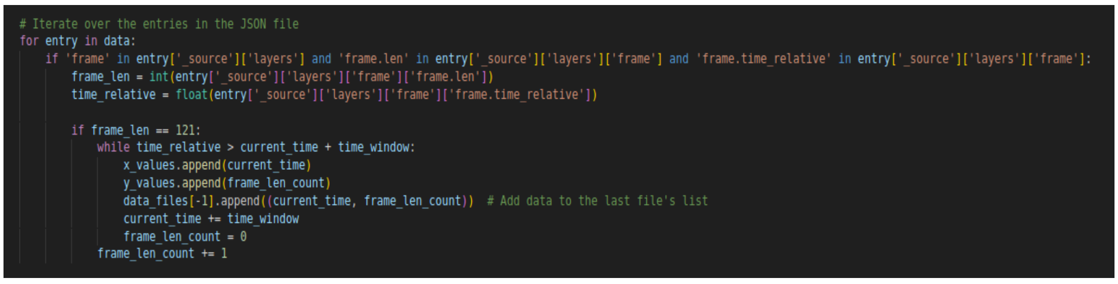
Figure 31 shows its use where PCAP files in residing directory are converted to JSON files. It skips if they are already present or creates them if they are not present.



```python
# Directory containing the pcap files
pcap_directory = f'{config.STATS_PATH}'

# Iterate over PCAP files in the directory. Skip if they are already present or create them if they are not present
for filename in os.listdir(pcap_directory):
    if filename.endswith('.pcap'):
        pcap_path = os.path.join(pcap_directory, filename)
        json_filename = f'{filename}.json'
        json_path = os.path.join(pcap_directory, json_filename)

        if os.path.exists(json_path):
            print(f'Skipping {json_filename} as it already exists.')
            continue

        command = f'tshark -r "{pcap_path}" -T json > "{json_path}"'
        os.system(command)
        print(f'Converted {filename} to JSON.')
```

Figure 31: Converting PCAP files into JSON files with Tshark.

Also, data filtering from the JSON files was needed to carry out data analysis hence filtering was carried out on every JSON file derived from the PCAP conversion concerning their **frame.len** and **frame.time.relative** fields as seen in **Figure 32** to help analyze CAM rate over time. Results derived from the implementation are analyzed and elaborated in **Sub-section 5.3.2.**



```python
# Iterate over the entries in the JSON file
for entry in data:
    if 'frame' in entry['_source']['layers'] and 'frame.len' in entry['_source']['layers']['frame'] and 'frame.time_relative' in entry['_source']['layers']['frame']:
        frame_len = int(entry['_source']['layers']['frame']['frame.len'])
        time_relative = float(entry['_source']['layers']['frame']['frame.time_relative'])

        if frame_len == 121:
            while time_relative > current_time + time_window:
                x_values.append(current_time)
                y_values.append(frame_len_count)
                data_files[-1].append((current_time, frame_len_count))  # Add data to the last file's list
                current_time += time_window
                frame_len_count = 0
            frame_len_count += 1
```

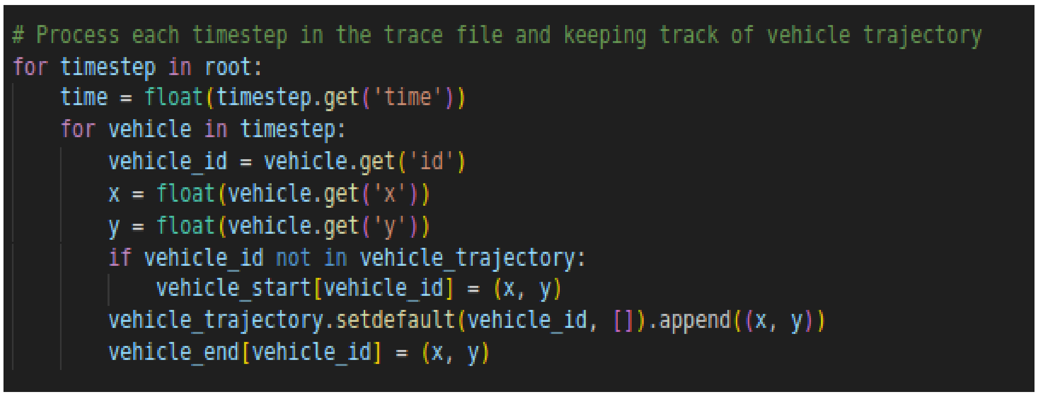
Figure 32: Iterating over field data in JSON file

4.3.2 Post-processing Mobility (Trace) files

Other post-processing carried out includes accessing the lanes field from the FCD-output (trace) file generated by SUMO to track the trajectory of every vehicle during the simulation and subsequently use that processed data to plot some relevant graphs which are marked with the origin and destination of each vehicle.

This was a thought idea if a user intends not to enable the SUMO GUI in parallel when running the ASA application. With this, the user still gets to have a graph plot that nearly satisfies the same intent. **Figure 33** shows a sample code snippet of the implementation.

The results derived from the implementation are analyzed and elaborated in **Sub-section 5.3.3**



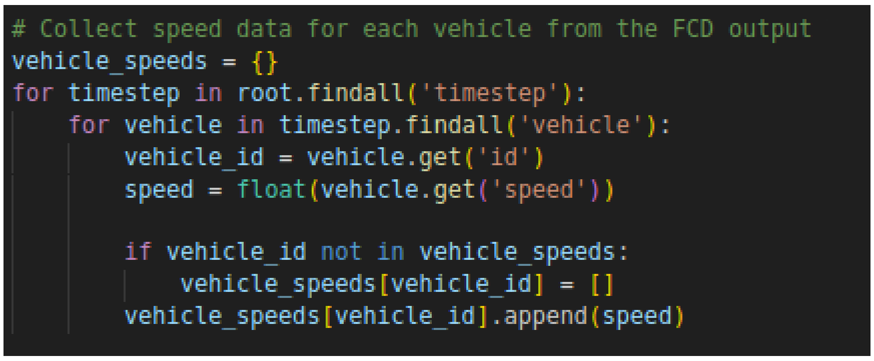
```python
# Process each timestep in the trace file and keeping track of vehicle trajectory
for timestep in root:
    time = float(timestep.get('time'))
    for vehicle in timestep:
        vehicle_id = vehicle.get('id')
        x = float(vehicle.get('x'))
        y = float(vehicle.get('y'))
        if vehicle_id not in vehicle_trajectory:
            vehicle_start[vehicle_id] = (x, y)
        vehicle_trajectory.setdefault(vehicle_id, []).append((x, y))
        vehicle_end[vehicle_id] = (x, y)
```

Figure 33: Filtering trajectory of each vehicle from origin to destination

Figure 34 shows another post-processing carried out to filter all speed data on every trajectory to time to achieve the goal of conducting speed distribution analytics from box plots and Kernel Density Estimation (KDE)

The C-ITS Area Speed Advisor application's box plot analysis offers insights into the speed distribution at the urban intersection, assisting traffic engineers and transportation authorities in making defensible decisions about traffic management, safety enhancements, and enforcement strategies to improve traffic flow, relieve congestion, and ensure adherence to speed limits.

When there are many data points and a smooth, continuous representation of the distribution is sought, the KDE plot is especially helpful. It aids in seeing the salient characteristics of the speed distribution and can be applied to several tasks, including spotting speed peaks, contrasting the distributions of different vehicles, or spotting odd trends or outliers in the speed behavior.



```
# Collect speed data for each vehicle from the FCD output
vehicle_speeds = {}
for timestep in root.findall('timestep'):
    for vehicle in timestep.findall('vehicle'):
        vehicle_id = vehicle.get('id')
        speed = float(vehicle.get('speed'))

        if vehicle_id not in vehicle_speeds:
            vehicle_speeds[vehicle_id] = []
        vehicle_speeds[vehicle_id].append(speed)
```

Figure 34: Filtering speed data from every vehicle in the simulation

Chapter 5

Tool Utilization and Results Evaluation

5.1 Implementing Scenarios

This section focuses on the workflow using the tool, specifically on how to deploy a real road scenario, run the C-ITS application – **areaspeedadvisor** uses V2V/V2I communications with **802.11p** technology, and analyze results.

Referring to the User Interface, let's get accustomed to the various controls available to us for use. In **Figure 35**, the field entries as seen denoted by **A** represents the field where coordinates of any geospatial area of interest will be inserted. **B** entry denotes the number of vehicles you require for your simulation The **Build Scenario** button generates all needed mobility files readily available for the C-ITS application. The **Run App** button when triggered starts the C-ITS application and the **Generate Plot** button launches several plot windows from a mobility and network perspective. The plots are interactive and can be modified to suit the kind of data you require analysis of. Directories to mobility and network files are made readily available with the click of a button as seen in label **C.** The **Generate Plots** button launches a new window labeled **D** and offers functionality to make choices on relevant plots to be displayed for analysis after simulation.

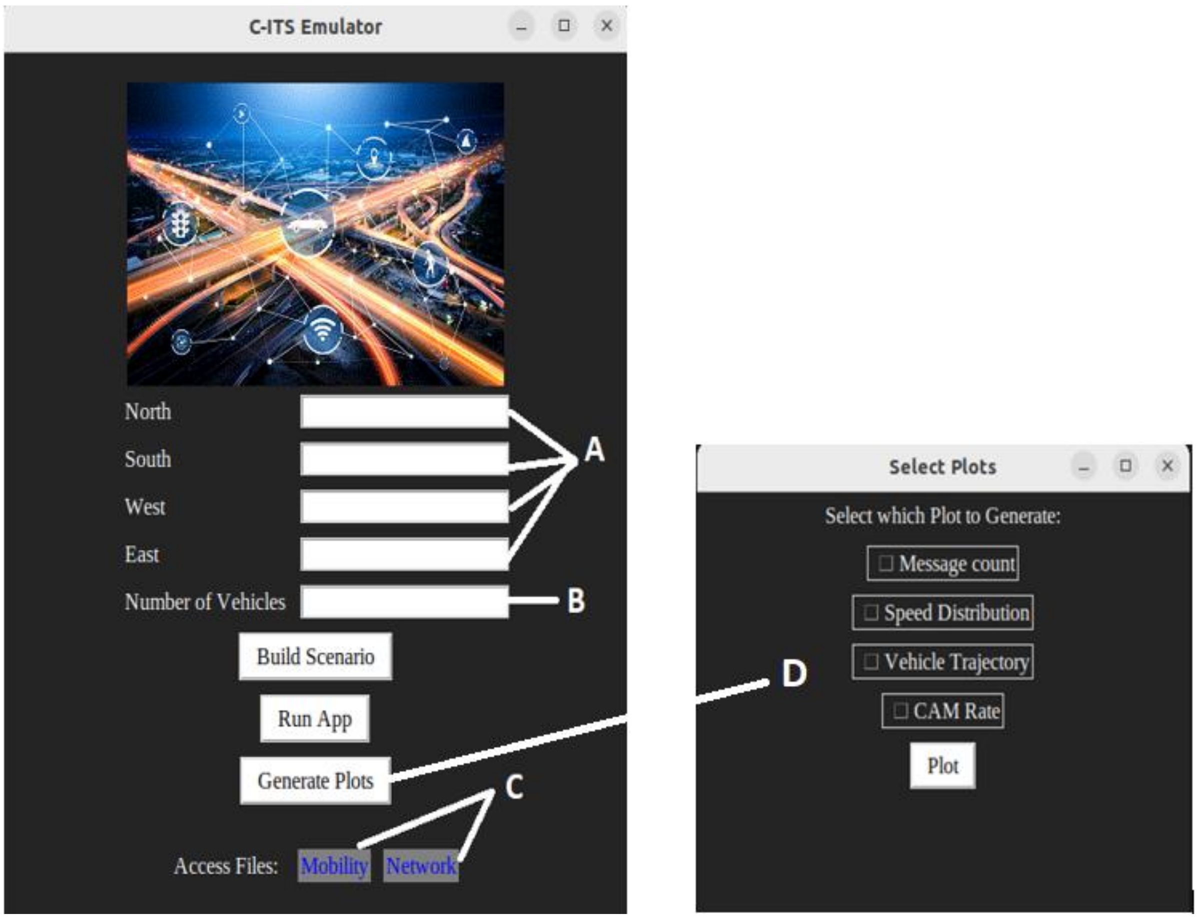


Figure 35: Framework User Interface

Three distinct situations with radically varied characteristics of the road network were selected for examination to enhance the simulation's realism. They are:

The highway scenario consists of long, low-density roadways with sporadic highway intersections. This road topology ensures a more straightforward environment for C-ITS communications since it allows for faster and more consistent vehicle speeds. The speed advisor implementation will be put to the test in this scenario.

Rural scenarios are characterized by very low traffic density and dimly lit roads with multiple junctions and blind turns. This road topology makes it possible for various vehicles to go at vastly different speeds, which when combined with poor visibility is a formula for disaster.

A high-traffic **urban environment** with plenty of road junctions, pedestrian crossings, and traffic lights. This road topology makes it difficult for C-ITS communications since vehicle speed is constantly varying and inconsistent.

In this section, the implementation process of the previously mentioned scenarios is presented. The chosen locations for the scenarios were specifically tailored to meet the requirements mentioned in **section 5.1.**

5.1.1 Highway Scenario

Ryfylketunnelen as seen below in **Figure 36** was chosen as it represents an uninterrupted stretch of highway In the Norwegian county of Rogaland is an underwater road tunnel called the Ryfylke Tunnel. It is a section of Norwegian National Road 13, which passes beneath the Horgefjord between Ryfylke and Stavanger. It is now the longest and deepest undersea road tunnel in the world, measuring 14.4 kilometers.

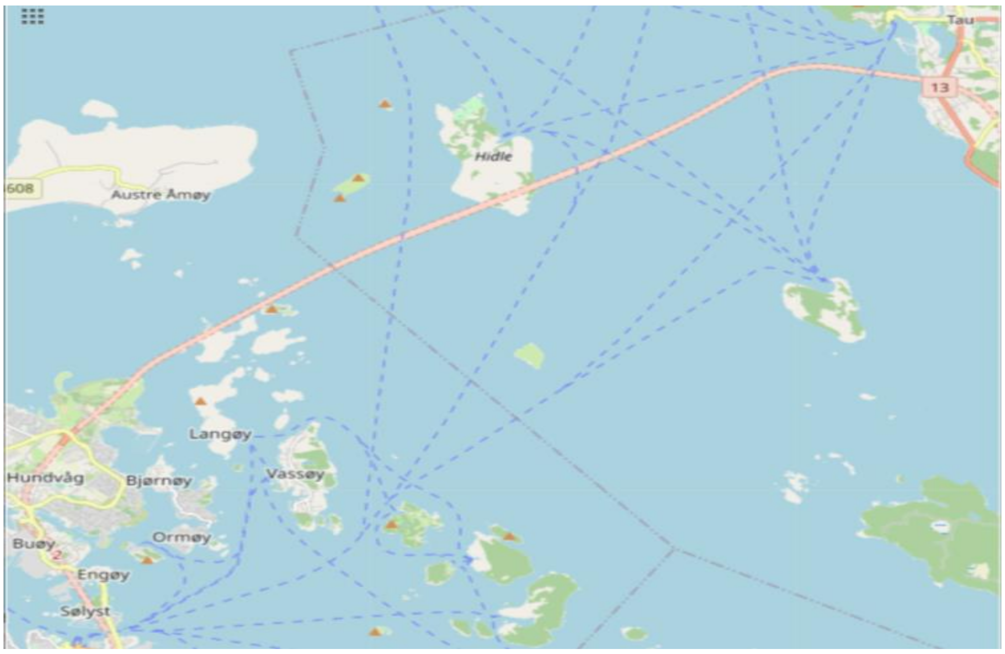


Figure 36: Highway Map Sample [46]

5.1.2 Rural Scenario

In **Figure 37** is Askje, a coastal farming and fishing village in Norway’s Rogaland County which typifies a typical rural setting. It is in the Stavanger municipality. About 578 people are living in the 0.25 square kilometer (62 acres) community, with a population density of 2,312 people per square kilometer (5,990/sq mi).

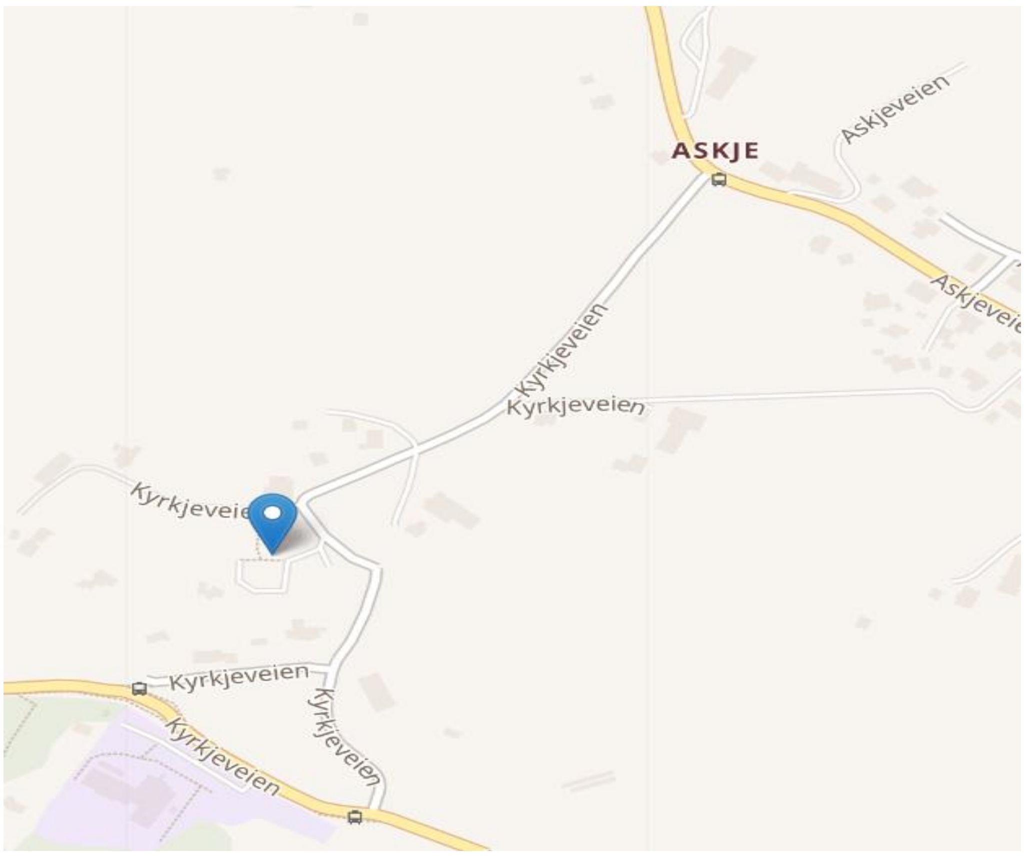


Figure 37: Rural Map Sample [46]

5.1.3 Urban Scenario

For this scenario, Downtown Stavanger was chosen; an area is a very popular tourist zone, flooded with traffic at any given hour of the day, resulting in one of the busiest zones in the city.

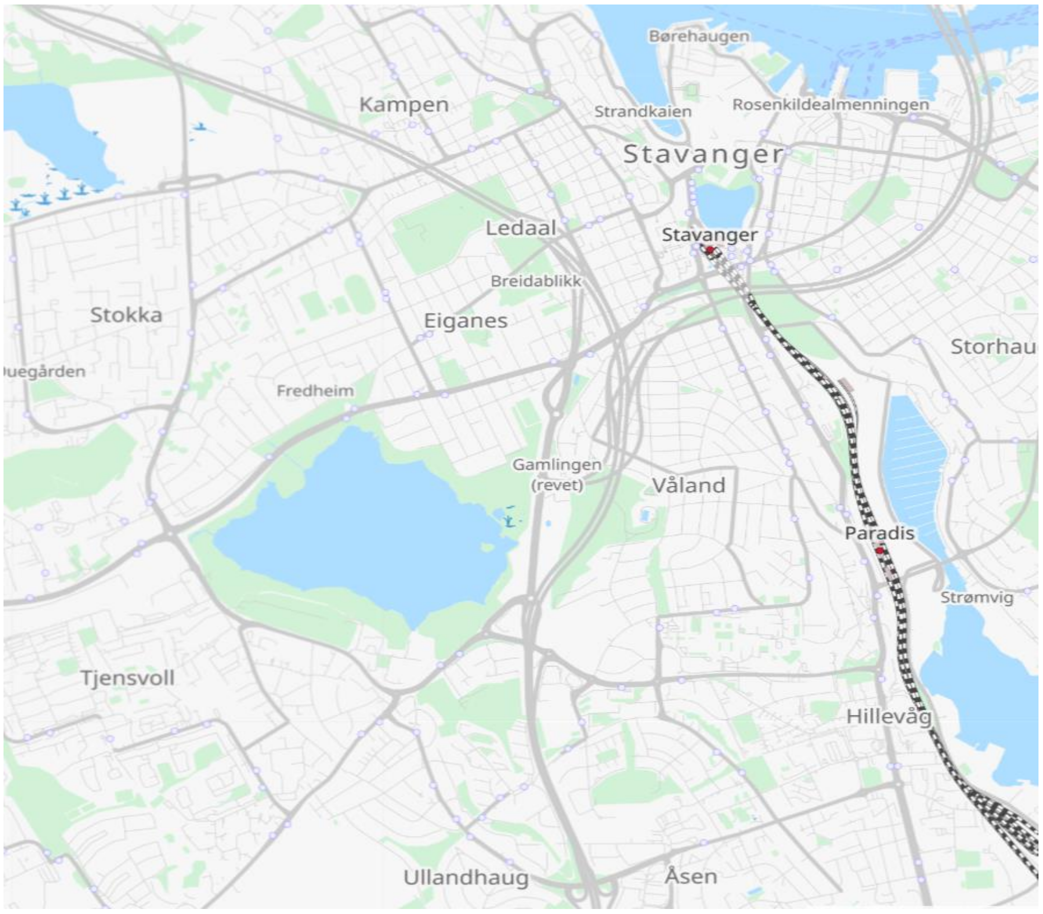


Figure 38: Urban Map Sample [46]

5.2 Test Cases

5.2.1 Test Case 1: Verification of CA AND DEN Message Count

Description:

Verify how many CAMs were sent and how many DENMs were received after the simulation.

Scenario: Run the ASA application Count the number of CAM that were transmitted over a specific period and compare it to the anticipated amount.

Steps for the test:

1. Configure the required parameters and set up the ASA application.
2. Open the program and give it a set amount of time to execute.
3. Take note of and examine the CAM sent over the specified period.
4. Determine how many CAM have been sent and how many DENMs have been received by referring to log files.

Outcome: Within a reasonable tolerance, the actual number of CAM must tally with the graph plot generated afterward.

5.2.2 Test Case 2: Creation of PCAP Files

Description:

Objective: To confirm that the Area Speed Advisor (ASA) application generates PCAP files while it is running. Run the ASA program on a test car and ensure that PCAP files are appropriately generated, containing the essential data and the pertinent CA and DEN messages.

Steps for the test:

1. PCAP file creation feature is enabled by default and can be activated and deactivated based on user preference.
2. Generate a scenario and run the ASA application.
3. Check to see if the ASA program creates PCAP files for each test run.
4. Verify the resulting PCAP files contain the anticipated CA and DEN messages and related data by analyzing them with the **Wireshark** application.
5. Generate plots relating to cam rate over time per every vehicle with each vehicle being able to be filtered out depending on what vehicle data the user expects to analyze.

Expected Outcome:

The ASA application correctly creates PCAP files, and the information in the files matches the anticipated CAM for each test run.

5.2.3 Test Case 3: Vehicle Trajectory

Description:

Considering the high computing resources in running ns-3 and SUMO in parallel, this test case helps as an option to have a fair view of each vehicle's trajectory thus a graph plot is generated that marks the origin and destination of every vehicle during the entire simulation.

Steps for the test:

1. Build a scenario by providing coordinates of the map area of interest.
2. Run the C-ITS application.
3. Generate plots after the simulation is ended.
4. View the graph plot and determine how each vehicle's trajectory over time.

Outcome: Each vehicle labeled should be labeled with a distinct color and a mark showing its origin and destination during the entire simulation message frequency varies with speed and driving circumstances by analyzing the data that was recorded.

5.2.4 Test Case 4: Speed Distribution and Plots Analysis

Description:

Vehicle speed data gathered through this framework can be analyzed to learn more about average speeds, adherence to speed regulations, and differences in speed on various types of roads. This technique can be used to evaluate the effectiveness of speed advisory messaging or identify places that may have potential speed-related safety hazards.

Steps for the test:

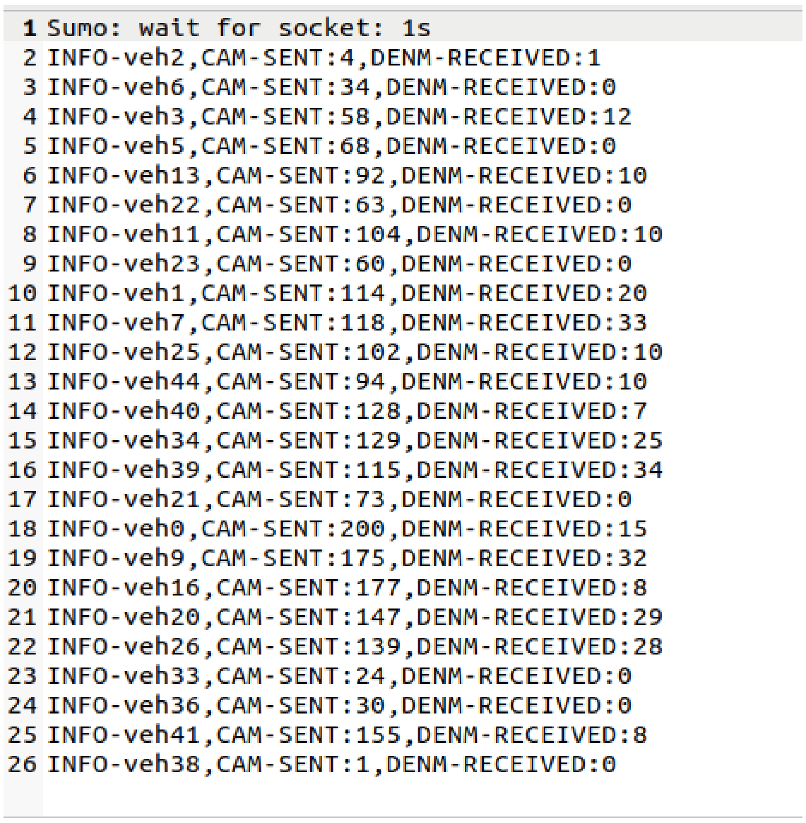
1. Build a scenario by providing coordinates of the map area of interest.
2. Run the C-ITS application.
3. Generate plots after the simulation is ended.
4. View the graph plot (box plot and KDE) and analyze how each vehicle's speed impacts the simulation.

Outcome: Each vehicle in the simulation should have data on its speed represented in a graph plot for analysis. The speed distribution was carried out based on highway, rural and urban scenarios discussed in **Sub-section 5.3.4.**

5.3 Results and Analysis

5.3.1 Results and Analysis of Test Case 1

Following the step-by-step implementation of the test case, the log file responsible for holding the message count was created and retrieved after the simulation. **Figure 39** shows a sample log containing logs for all messages counted for each vehicle.



```
Sumo: wait for socket: 1s
INFO-veh2,CAM-SENT:4,DENM-RECEIVED:1
INFO-veh6,CAM-SENT:34,DENM-RECEIVED:0
INFO-veh3,CAM-SENT:58,DENM-RECEIVED:12
INFO-veh5,CAM-SENT:68,DENM-RECEIVED:0
INFO-veh13,CAM-SENT:92,DENM-RECEIVED:10
INFO-veh22,CAM-SENT:63,DENM-RECEIVED:0
INFO-veh11,CAM-SENT:104,DENM-RECEIVED:10
INFO-veh23,CAM-SENT:60,DENM-RECEIVED:0
INFO-veh1,CAM-SENT:114,DENM-RECEIVED:20
INFO-veh7,CAM-SENT:118,DENM-RECEIVED:33
INFO-veh25,CAM-SENT:102,DENM-RECEIVED:10
INFO-veh44,CAM-SENT:94,DENM-RECEIVED:10
INFO-veh40,CAM-SENT:128,DENM-RECEIVED:7
INFO-veh34,CAM-SENT:129,DENM-RECEIVED:25
INFO-veh39,CAM-SENT:115,DENM-RECEIVED:34
INFO-veh21,CAM-SENT:73,DENM-RECEIVED:0
INFO-veh0,CAM-SENT:200,DENM-RECEIVED:15
INFO-veh9,CAM-SENT:175,DENM-RECEIVED:32
INFO-veh16,CAM-SENT:177,DENM-RECEIVED:8
INFO-veh20,CAM-SENT:147,DENM-RECEIVED:29
INFO-veh26,CAM-SENT:139,DENM-RECEIVED:28
INFO-veh33,CAM-SENT:24,DENM-RECEIVED:0
INFO-veh36,CAM-SENT:30,DENM-RECEIVED:0
INFO-veh41,CAM-SENT:155,DENM-RECEIVED:8
INFO-veh38,CAM-SENT:1,DENM-RECEIVED:0
```

Figure 39: Log showing each vehicle and respective CAM sent and DENM received.

A group bar plot can be used to analyze and visualize the data more thoroughly. Hence as seen in **Figure 40,** the plot would allow for an easier understanding of each vehicle's communication and environmental awareness within the simulation by allowing a side-by-side comparison of the quantity of CAM issued and DENM received. There is also a dedicated interactive filter button that helps you view plots for selected vehicles as seen in **Figure 41** as well. This gives users the flexibility on which vehicle data they require to view concerning their message count data plotted.

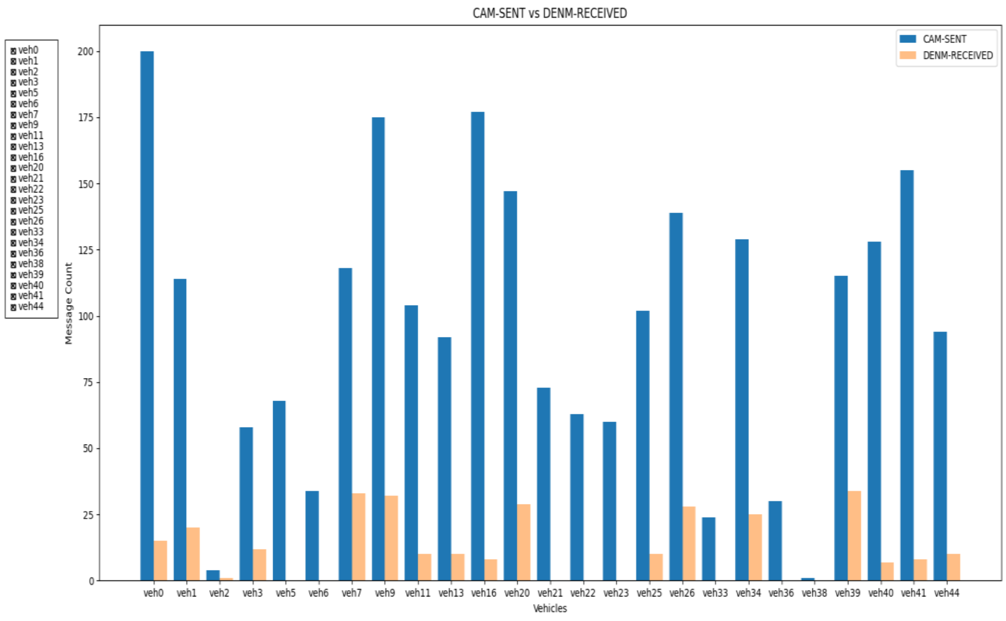


Figure 40: CAM and DENM message count.

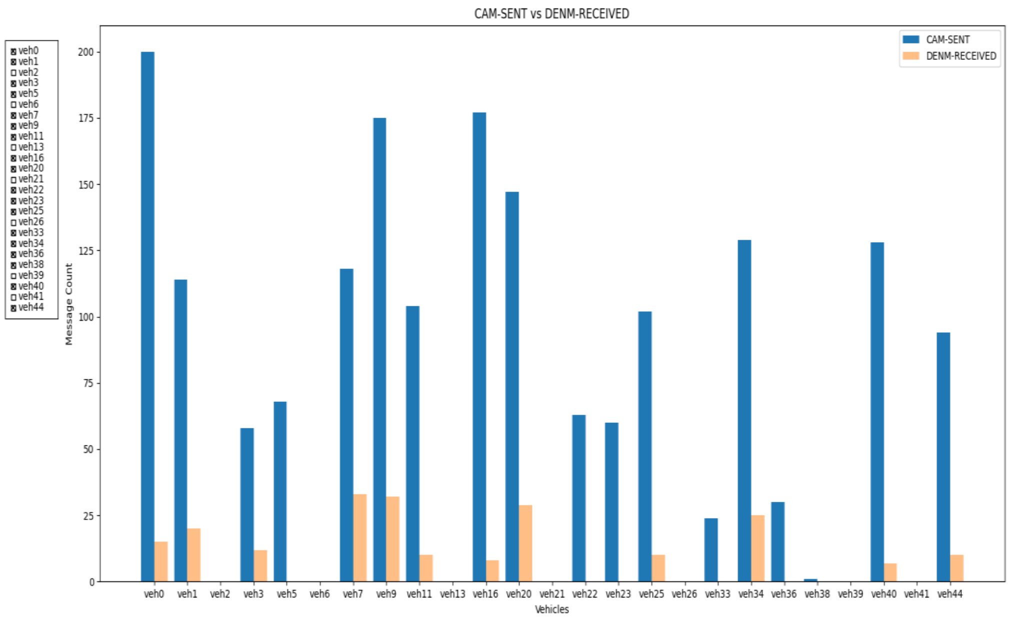


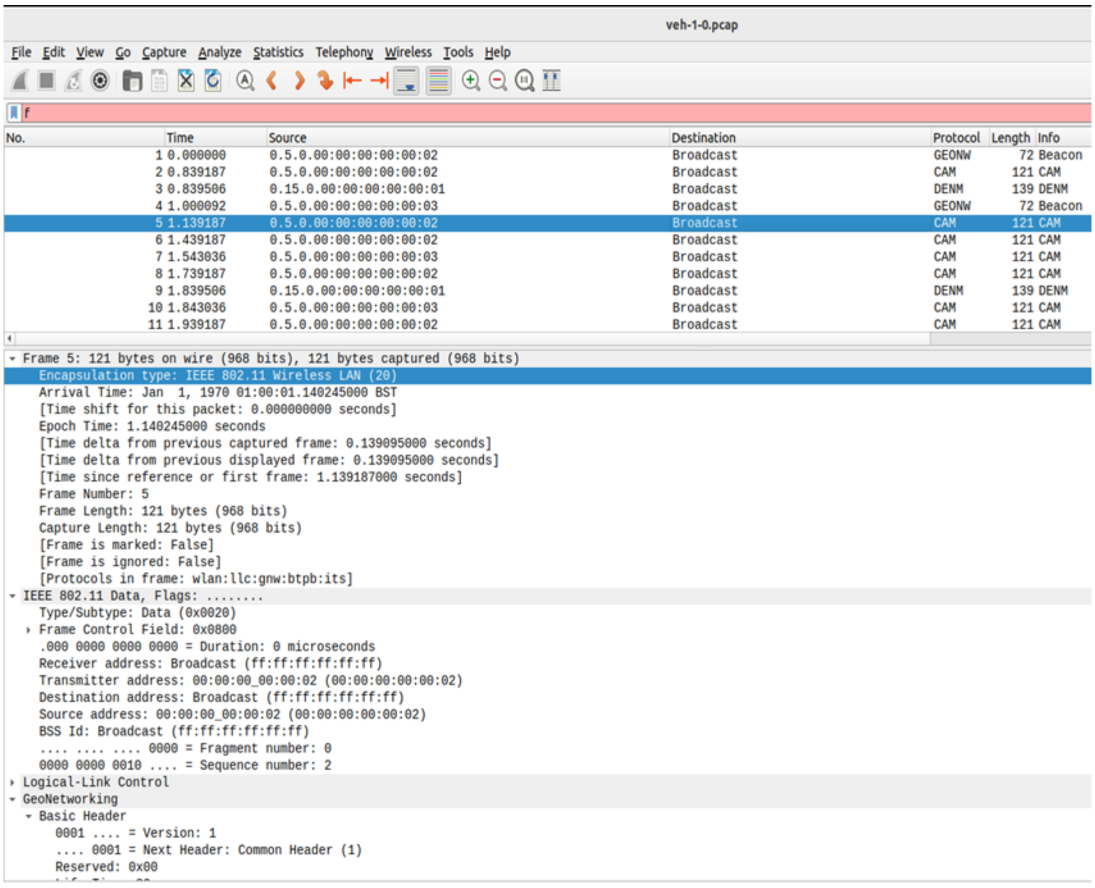
Figure 41: Filtered CAM and DENM message count

First, there are between 1 and 200 CAM delivered, with "veh0" having the highest count of 200 and vehicle "veh38" having the lowest count of 1. This suggests that activity and communication among the vehicles are at various levels. Notably, "veh0", "veh9", and "veh16" have sent more than 150 CAM apiece, demonstrating their active participation in the simulation.

Second, there are between 0 and 34 DENM that have been received. The maximum number of DENM, 34, was sent to "veh39", whereas none were sent to other vehicles. With more than 30 DENM received, "veh7," "veh9," and "veh39" stand out as having a stronger reactivity and awareness of environmental notifications.

These observations show that the vehicles' communication and attentiveness levels vary. While some vehicles display a higher level of environmental responsiveness by receiving a big number of DENM, others actively engage by sending a significant number of CAM. On the other hand, vehicles that did not get any DENM may not have had much network engagement or awareness of their surroundings hence by indicating potential areas for improvement in terms of a fair message exchange, engaged involvement, and effective communication among the vehicles, these insights help us better understand the communication dynamics. The findings can be utilized to improve the simulation environment's overall efficacy and dependability, as well as to fix communication gaps and optimize the cooperative communication system.

5.3.2 Results and Analysis of Test Case 2


veh-1-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
f

```
No.   Time        Source                            Destination   Protocol  Length  Info
    1 0.000000    0.5.0.00:00:00:00:00:02           Broadcast     GEONW        72 Beacon
    2 0.839187    0.5.0.00:00:00:00:00:02           Broadcast     CAM         121 CAM
    3 0.839506    0.15.0.00:00:00:00:00:01          Broadcast     DENM        139 DENM
    4 1.000092    0.5.0.00:00:00:00:00:03           Broadcast     GEONW        72 Beacon
    5 1.139187    0.5.0.00:00:00:00:00:02           Broadcast     CAM         121 CAM
    6 1.439187    0.5.0.00:00:00:00:00:02           Broadcast     CAM         121 CAM
    7 1.543036    0.5.0.00:00:00:00:00:03           Broadcast     CAM         121 CAM
    8 1.739187    0.5.0.00:00:00:00:00:02           Broadcast     CAM         121 CAM
    9 1.839506    0.15.0.00:00:00:00:00:01          Broadcast     DENM        139 DENM
   10 1.843036    0.5.0.00:00:00:00:00:03           Broadcast     CAM         121 CAM
   11 1.939187    0.5.0.00:00:00:00:00:02           Broadcast     CAM         121 CAM
```

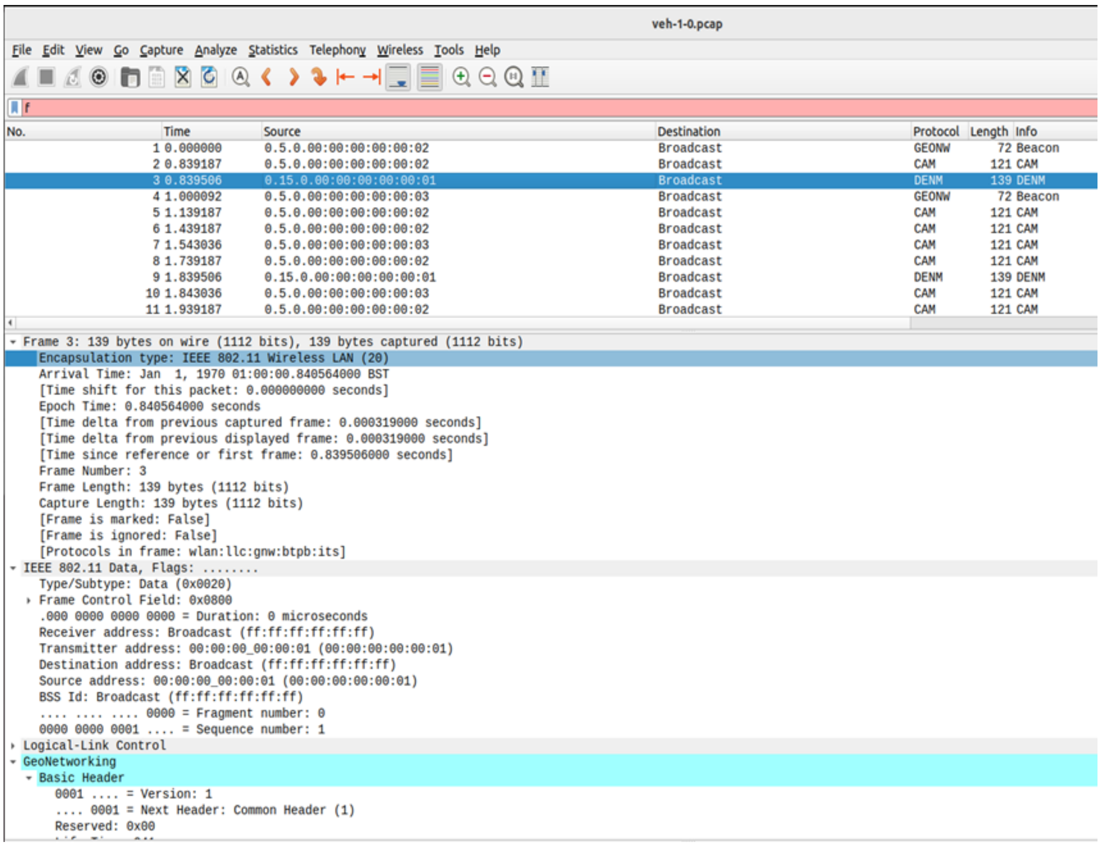
```
▾ Frame 5: 121 bytes on wire (968 bits), 121 bytes captured (968 bits)
    Encapsulation type: IEEE 802.11 Wireless LAN (20)
    Arrival Time: Jan  1, 1970 01:00:01.140245000 BST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1.140245000 seconds
    [Time delta from previous captured frame: 0.139095000 seconds]
    [Time delta from previous displayed frame: 0.139095000 seconds]
    [Time since reference or first frame: 1.139187000 seconds]
    Frame Number: 5
    Frame Length: 121 bytes (968 bits)
    Capture Length: 121 bytes (968 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: wlan:llc:gnw:btpb:its]
▾ IEEE 802.11 Data, Flags: ........
    Type/Subtype: Data (0x0020)
  ▸ Frame Control Field: 0x0800
    .000 0000 0000 0000 = Duration: 0 microseconds
    Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
    Transmitter address: 00:00:00_00:00:02 (00:00:00:00:00:02)
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Source address: 00:00:00_00:00:02 (00:00:00:00:00:02)
    BSS Id: Broadcast (ff:ff:ff:ff:ff:ff)
    .... .... .... 0000 = Fragment number: 0
    0000 0000 0010 .... = Sequence number: 2
▸ Logical-Link Control
▾ GeoNetworking
  ▾ Basic Header
      0001 .... = Version: 1
      .... 0001 = Next Header: Common Header (1)
      Reserved: 0x00
```

Figure 42: Sample Vehicle PCAP file showing details on CAM.

Analyzing the PCAP file for veh-1-0 at timeframe 1.139187 as depicted in **Figure 42**, it is seen that at the frame level, we can see information like the frame size (121 bytes) which is a frame size representation for CAM and the related protocols (wlan:llc:gnw:btpb:its). The wireless communication's source and destination addresses are disclosed by the WLAN layer. The logical link control protocol employed, along with additional control information, is indicated by the LLC layer. Geographic networking is related to the GNW layer, which holds position-related data including latitude, longitude, altitude, and speed. Finally, Basic Transport Protocol (BTP) and its layers are related. These layers include additional cooperative awareness messaging (CAM) headers, payload, and contextual data.

The CAM payload underwent further processing, which revealed numerous measurements and parameters about the transmitting vehicle. The station type (5), reference position (latitude, longitude, and confidence), altitude information (value, confidence), heading, speed, drive direction, vehicle dimensions (length, width), longitudinal acceleration, curvature, curvature calculation mode, yaw rate, and lane position are among them.

We may learn more about the information that is transmitted throughout the network by looking at these specifics. The behavior of the cars, their locations, their movement patterns, and different environmental aspects may all be understood using this data. Such analysis offers a thorough comprehension of the dynamics of the network and supports the creation and improvement of intelligent transportation systems.


veh-1-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
f

```
No.        Time        Source                          Destination    Protocol  Length  Info
         1 0.000000    0.5.0.00:00:00:00:00:02         Broadcast      GEONW       72 Beacon
         2 0.839187    0.5.0.00:00:00:00:00:02         Broadcast      CAM        121 CAM
         3 0.839506    0.15.0.00:00:00:00:00:01        Broadcast      DENM       139 DENM
         4 1.000092    0.5.0.00:00:00:00:00:03         Broadcast      GEONW       72 Beacon
         5 1.139187    0.5.0.00:00:00:00:00:02         Broadcast      CAM        121 CAM
         6 1.439187    0.5.0.00:00:00:00:00:02         Broadcast      CAM        121 CAM
         7 1.543036    0.5.0.00:00:00:00:00:03         Broadcast      CAM        121 CAM
         8 1.739187    0.5.0.00:00:00:00:00:02         Broadcast      CAM        121 CAM
         9 1.839506    0.15.0.00:00:00:00:00:01        Broadcast      DENM       139 DENM
        10 1.843036    0.5.0.00:00:00:00:00:03         Broadcast      CAM        121 CAM
        11 1.939187    0.5.0.00:00:00:00:00:02         Broadcast      CAM        121 CAM
```

```
▾ Frame 3: 139 bytes on wire (1112 bits), 139 bytes captured (1112 bits)
    Encapsulation type: IEEE 802.11 Wireless LAN (20)
    Arrival Time: Jan  1, 1970 01:00:00.840564000 BST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 0.840564000 seconds
    [Time delta from previous captured frame: 0.000319000 seconds]
    [Time delta from previous displayed frame: 0.000319000 seconds]
    [Time since reference or first frame: 0.839506000 seconds]
    Frame Number: 3
    Frame Length: 139 bytes (1112 bits)
    Capture Length: 139 bytes (1112 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: wlan:llc:gnw:btpb:its]
▾ IEEE 802.11 Data, Flags: ........
    Type/Subtype: Data (0x0020)
  ▸ Frame Control Field: 0x0800
    .000 0000 0000 0000 = Duration: 0 microseconds
    Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
    Transmitter address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Source address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    BSS Id: Broadcast (ff:ff:ff:ff:ff:ff)
    .... .... .... 0000 = Fragment number: 0
    0000 0000 0001 .... = Sequence number: 1
▸ Logical-Link Control
▾ GeoNetworking
  ▾ Basic Header
      0001 .... = Version: 1
      .... 0001 = Next Header: Common Header (1)
      Reserved: 0x00
```

Figure 43: Sample Vehicle PCAP file showing details on DENM.

Referencing the same PCAP file for **veh-1-0** as seen in **Figure 43,** by analyzing the DENM packet entry at timeframe 0.839506000, we can see details like the frame length (139 bytes) and the related protocols (wlan:llc:gnw:btpb: its) at the frame level. The wireless communication's source and destination addresses are disclosed by the WLAN layer. The logical link control protocol, which includes control data and SAP (Service Access Point) values, is related to the LLC layer. Geographic networking is related to the GNW layer, which has distinct data for location, speed, and direction.

The presence of BTPB (Basic Transport Protocol) and ITS (Intelligent Transport Systems) layers in the payload is discovered through further research. An **ItsPduHeader_element** of the ITS layer comprises details on the protocol version and message ID. A **DecentralizedEnvironmentalNotificationMessage_element** that contains information about decentralized environmental notifications is also visible within the ITS layer. Included in this are management data such as action ID, detection time, reference time, event position (latitude, longitude, and altitude), and station type.

Along with position information (latitude, longitude, altitude) and confidence values, the GNW layer also offers information about the geographic environment, including speed and heading. In the **alacarte_element** of the DENM (Decentralized Environmental Notification Message), we also find a **roadWorks_element**. This component contains details on roadwork, like the speed limit.

With the generated PCAP file, they were further processed as part of post-processing activity by converting them into JSON files to run some statistical analysis and generate relevant plots from them. With each PCAP file representing network details for each vehicle, we computed the CAM rate every 2 seconds and generated a compound plot for them as seen in **Figure 44.** On the other hand, depending on the preference of the CAM rate, the interval can be modified in the codebase to suit the context of your plots.

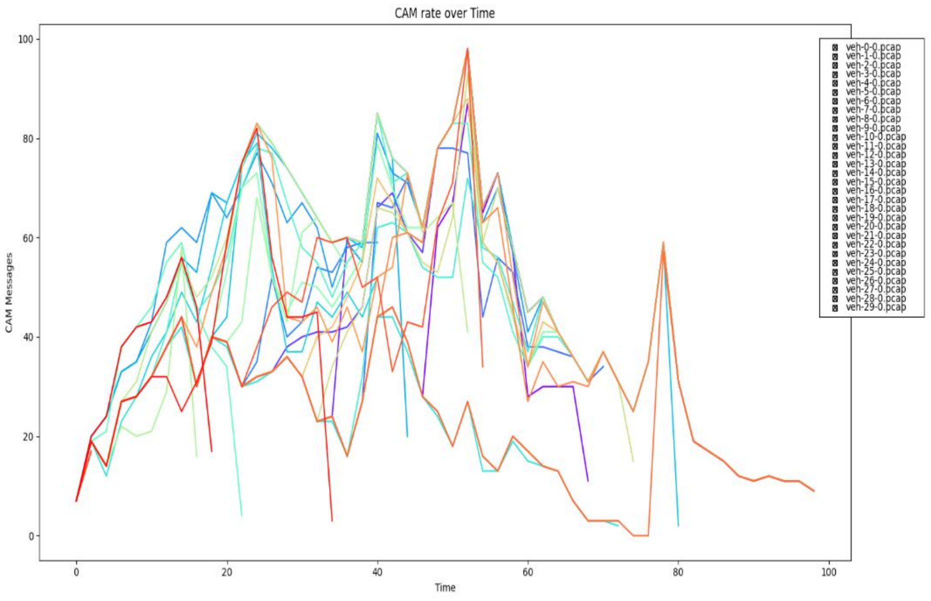


Figure 44: CAM rate over time from PCAP files of all vehicles in simulation.

Again, there was the introduction of dedicated interactive filter buttons that helps you view plots for selected PCAP cam rate over time as seen in **Figure 45.** This gives users the flexibility to filter which PCAP plots they require to view.

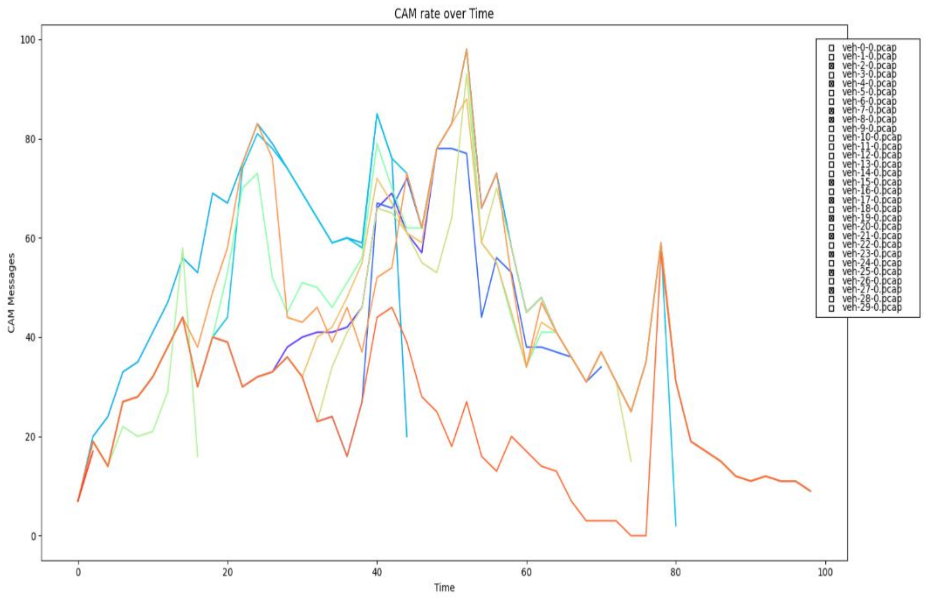


Figure 45: Filtered CAM rate over time from selected vehicle PCAP files

5.3.3 Results and Analysis of Test Case 3

Performing simulations with multiple components and/or infrastructure running in parallel can be resource intensive. Owing to this, aside from the GUI SUMO offers during the simulation as part of running ASA, there was the idea to introduce an alternative to viewing the trajectory of vehicles by providing a graph plot that shows the trajectory of every vehicle and makes it available for view after simulation and to add to it there were indications that points to the origin and destination of every vehicle during the entire simulation period as seen in **Figure 46**.

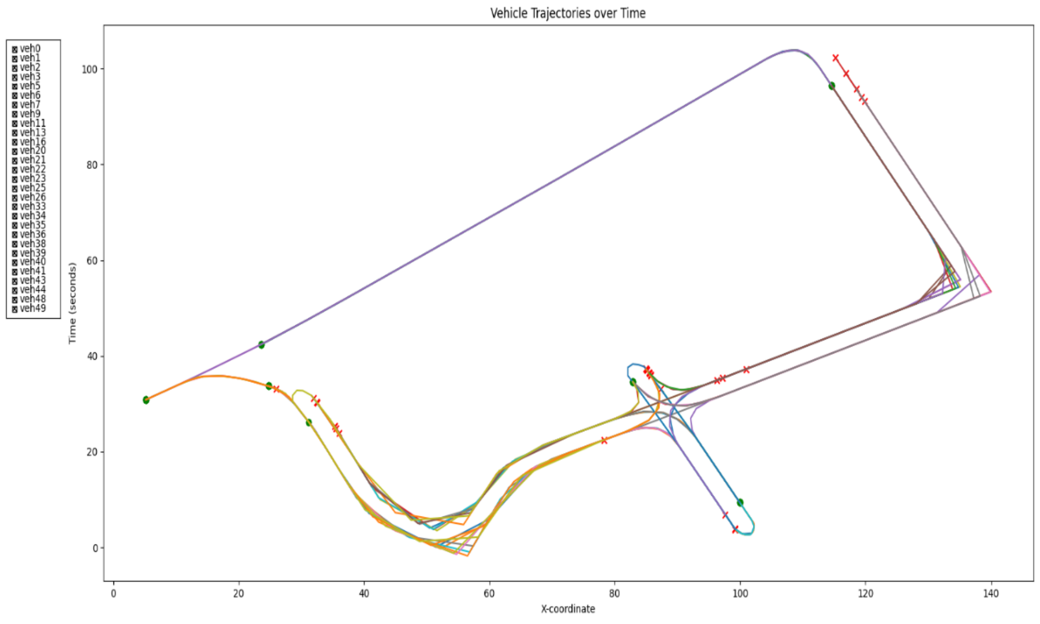


Figure 46: Vehicle trajectory during the entire simulation marked origin and destination.

And to make the visualization much more dynamic, there was the implementation of dedicated interactive filter buttons that helps users view plots for selected vehicles as depicted in **Figure 47** hence giving users the flexibility to filter which vehicle trajectory they require to view.

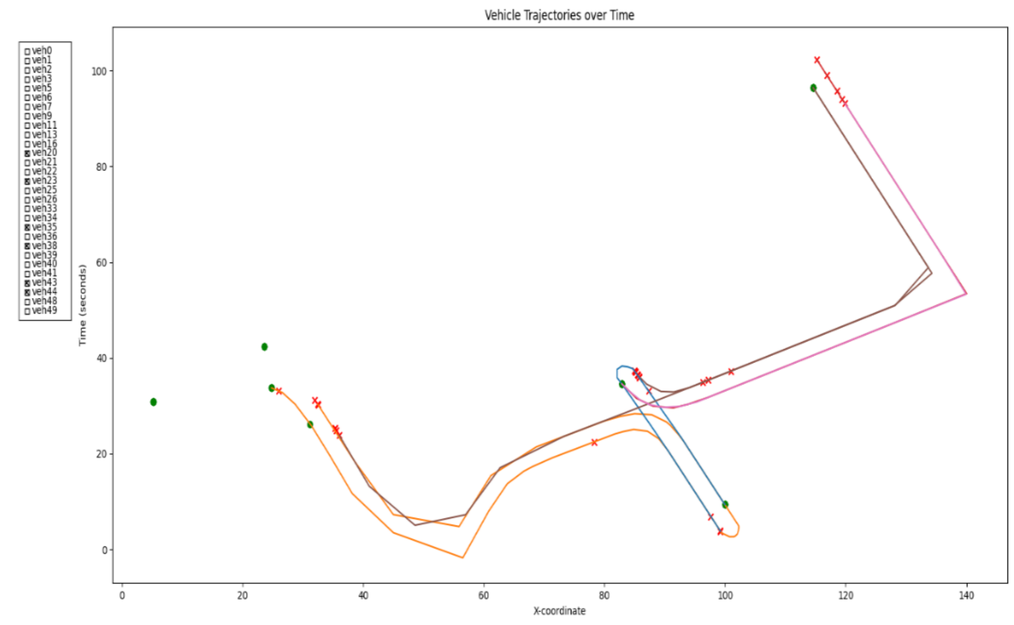


Figure 47: Selected vehicle trajectory during simulation marked with origin and destination.

5.3.4 Results and Analysis of Test Case 4

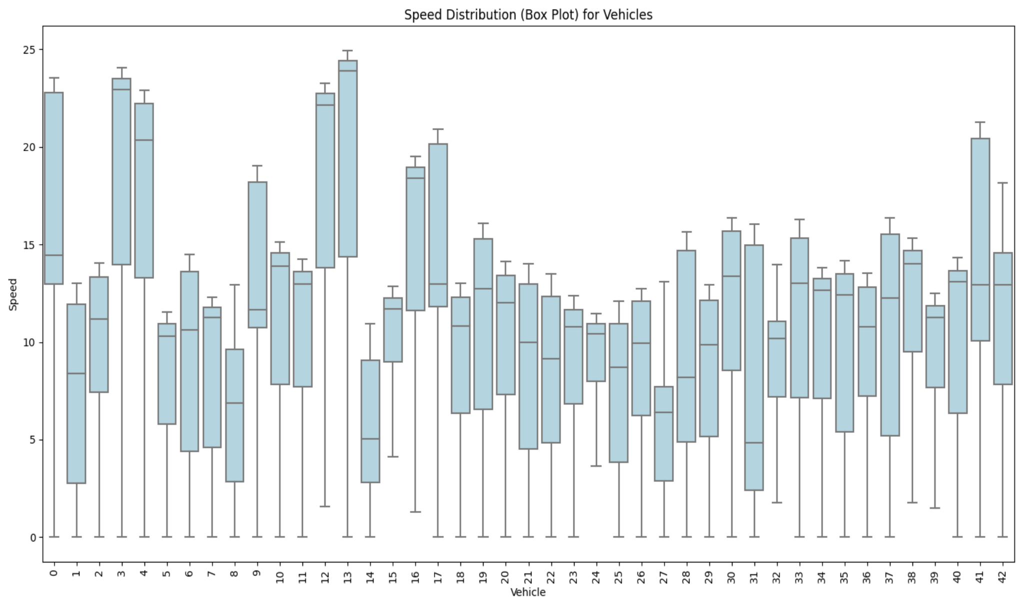


Figure 48: Highway Speed analysis of vehicles using Box plot.

Analysis of the concentration of speeds inside the tunnel is provided by the box plot as seen in **Figure 48**. Vehicles appear to travel through the tunnel at a fairly constant speed, according to a tiny interquartile range (IQR) and a narrow box. This shows that traffic is moving steadily and that any tunnel-specific speed limits or regulations are being followed.

The box plot's box width and whisker length indicate the variation in the speed distribution. A larger box or longer whiskers indicate a wider range of speeds, reflecting variability in driving patterns. Driver traits, traffic circumstances, and the existence of any unique features or restrictions inside the tunnel could all have an impact on this unpredictability.

The influence of tunnel parameters on speed distribution can be considered in the box plot analysis. Driver behavior and speed decisions within the tunnel can be influenced by elements including tunnel length, lighting, visibility, the presence of curves or inclines, and the layout of the entrance and exit points.

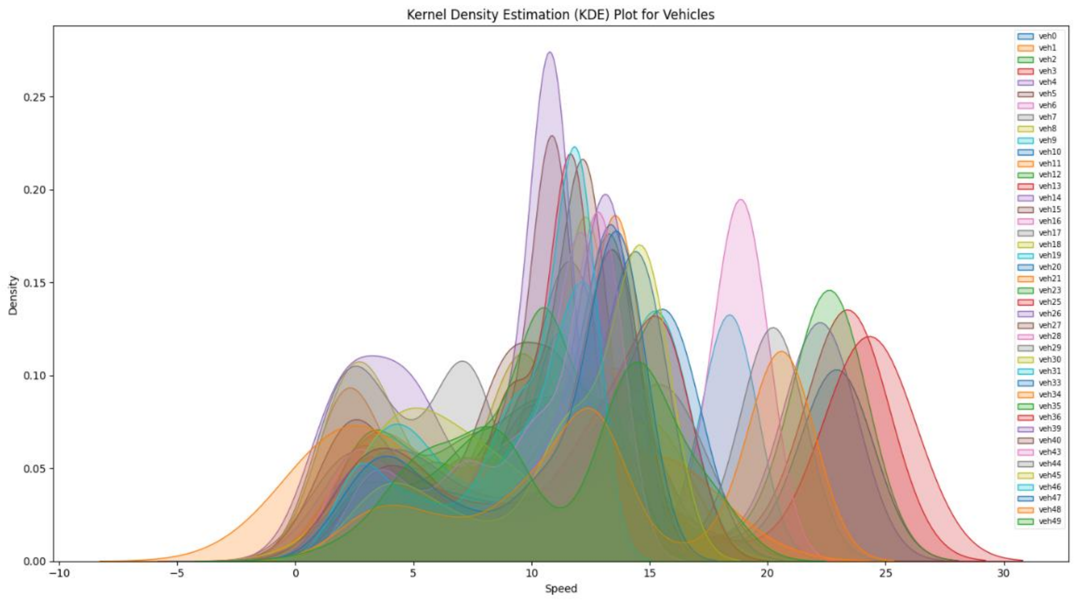


Figure 49: Highway speed distribution of all vehicles using KDE.

A smooth approximation of the speed distribution in the tunnel highway is provided by the KDE graphic in **Figure 49**. The prevalent patterns or modes in the speed behavior can be determined by looking at the curve's form. Most cars prefer to maintain a particular speed range inside the tunnel, according to a unimodal distribution with a noticeable peak. Alternately, a multimodal distribution with numerous peaks may indicate various speed ranges or types of driving habits impacted by things like tunnel sections, deteriorating road conditions, or backed-up traffic.

The highest peaks on the KDE plot correspond to the speeds at which a lot of cars typically pass through the tunnel. Understanding the typical driving habits and preferred speeds of drivers in the tunnel environment can be gained by analyzing these peak speeds.

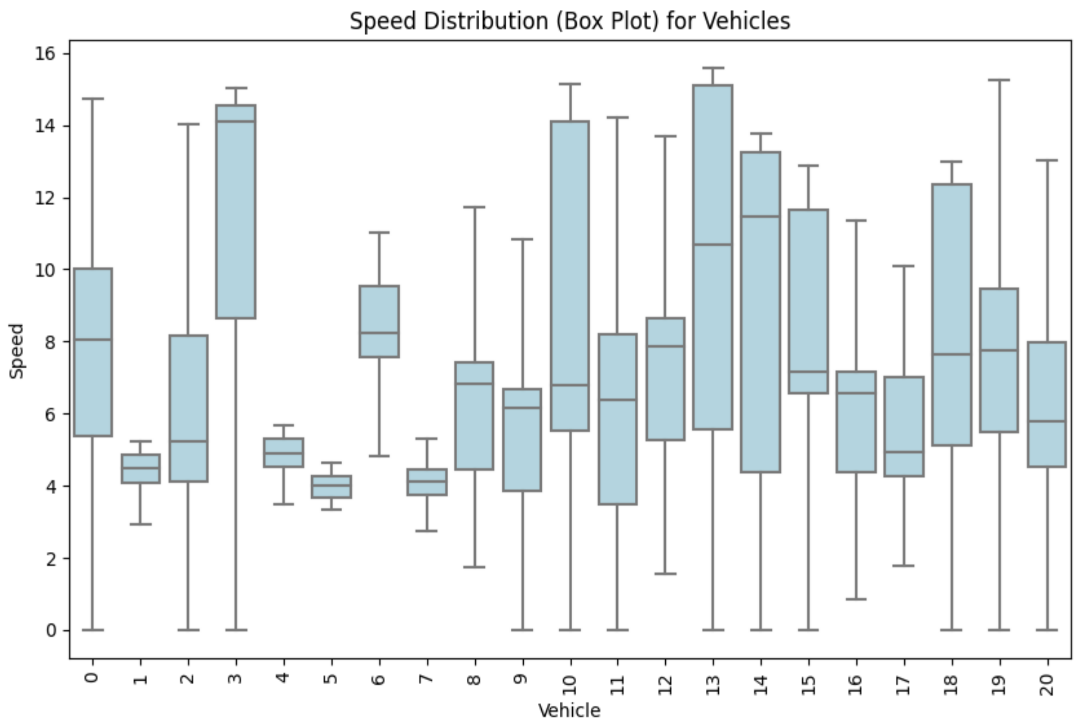


Figure 50: Rural Speed analysis of vehicles using Box plot.

In **Figure 50**, we gain detailed insights into speed behavior by analyzing the box plot for various road segments in rural areas, such as straight stretches, curves, or intersections. With the use of this study, potential safety hazards can be located, such as crossings with strict speed restrictions or abrupt curves where drivers may need to modify their speed.

A bigger speed disparity between cars is indicated by a wider IQR, which points to a variety of driving styles. Road conditions and traffic volume are just two examples of variables that can affect this fluctuation.

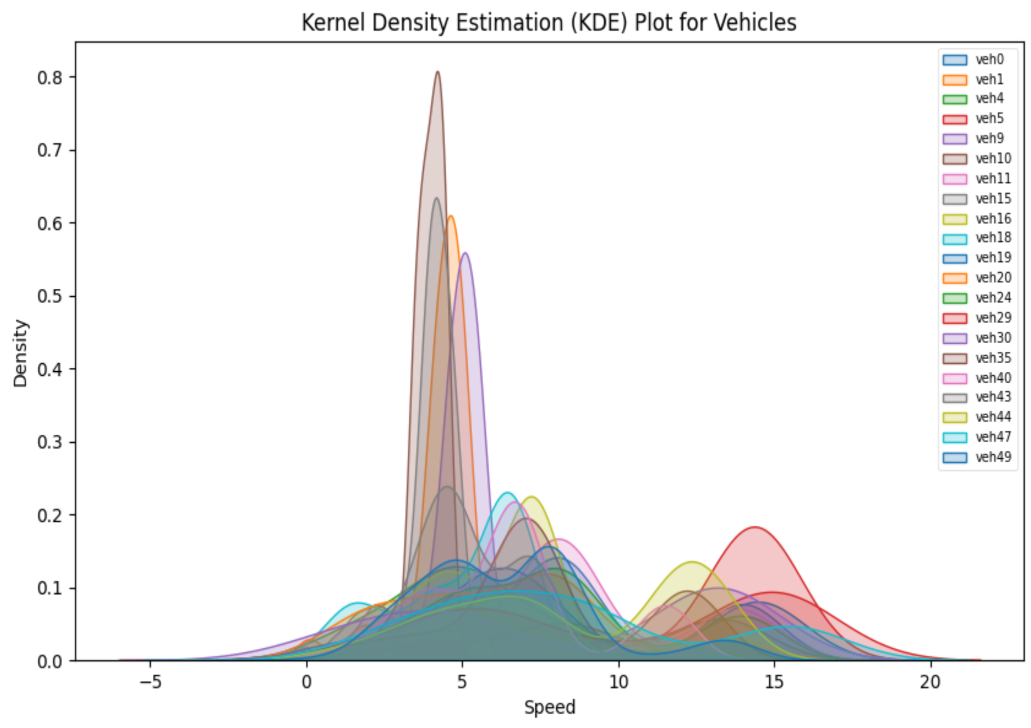


Figure 51: Rural speed distribution of all vehicles using KDE.

The shape, peaks, and spread of the KDE plot in **Figure 51** can provide key details about driving behavior in rural areas by analyzing the general speed distribution. For instance, a single peak in the KDE plot represents a dominant speed range, indicating reliable driving practices on rural roads.

The width of the figure highlights places with various driving habits or parts with different speed limit enforcement, indicating the heterogeneity in speed distribution.

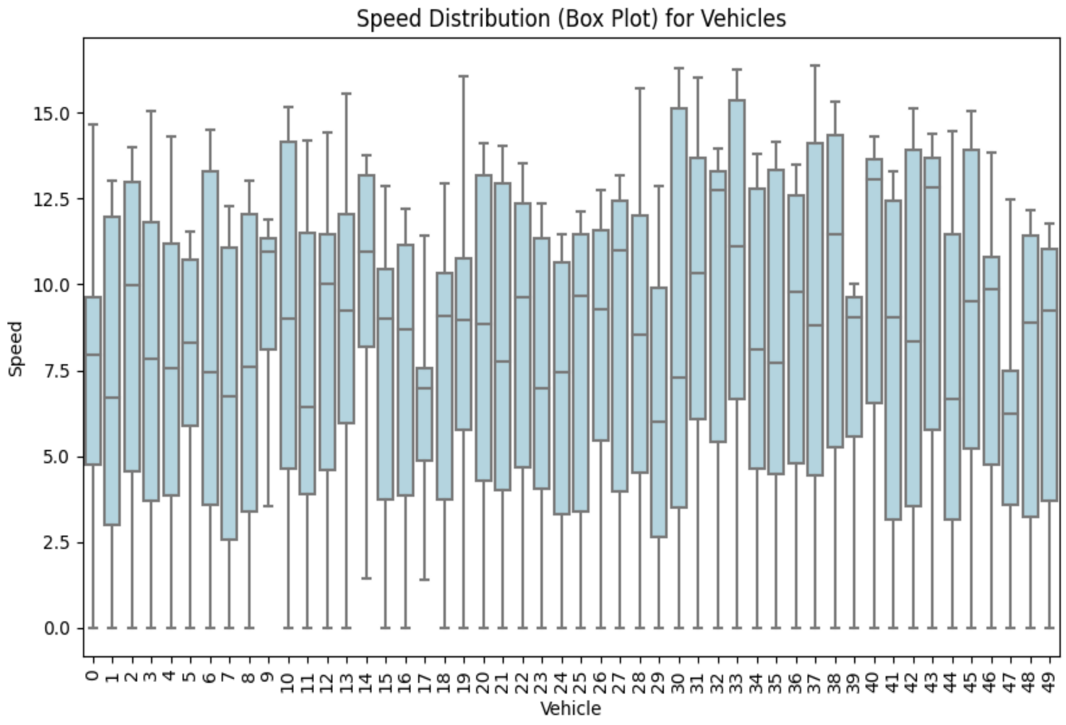


Figure 52: Urban Speed analysis of vehicles using Box plot.

In **Figure 52**, an estimation of the typical speed of cars in the area is given by the horizontal line inside the box, which represents the median speed. A lower median speed may indicate that the location the vehicle is in has greater congestion, whereas a higher median speed for a few vehicles indicates that the area the vehicle is located in has relatively less congestion.

The height of the IQR conveys details about the range of speeds in the vicinity. A smaller IQR might suggest that some cars' speeds are quite steady, whereas a bigger IQR might suggest that there is more speed variety.

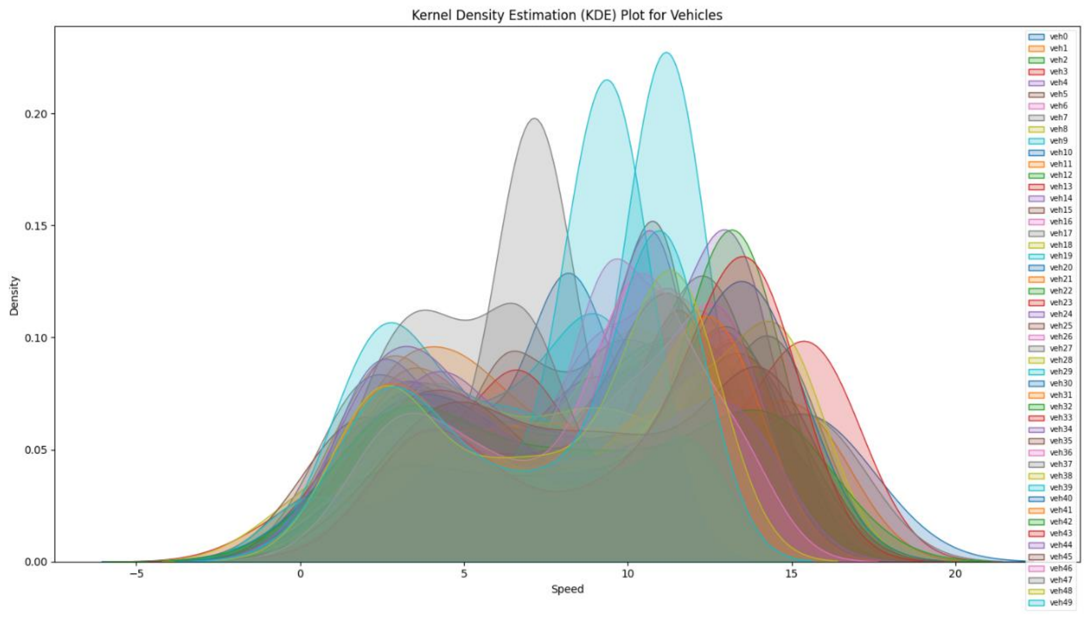


Figure 53: Urban speed distribution of all vehicles using KDE.

The KDE plot in **Figure 53** displays numerous separate peaks or modes, which may indicate the presence of various road types or speeding habits in the urban region. An example of a bimodal distribution is when there is a mix of arterial roads with greater speeds and residential streets with lower speeds. Based on the unique characteristics of each road type, this information can direct focused interventions and improvements to road design.

The excessive speed values seen in the urban region are represented by the KDE plot's tails. Finding locations or times where excessive speeding or unexpectedly low speeds occur can be made easier by analyzing the tail regions. To provide safer driving conditions, certain places may need targeted enforcement actions, traffic calming techniques, or infrastructural upgrades.

Chapter 6

Conclusion

This study commenced by conducting a thorough analysis of the status of Cooperative Intelligent Transport Systems. This study scrutinized European standards on the regulation of communication aspects and application organization and analyzed the present development and testing methodologies used for Cooperative Intelligent Transport Systems (C-ITS). The simulation framework that was selected stands out due to its foundation of realism, which is essential for conducting research that yields significant results, thereby distinguishing it from other available alternatives. Subsequently, an analysis was conducted to delineate the primary real-world situations that exist on the roadways. This process led to the development of three distinct test case documents, each of which accurately reflects the attributes of anticipated intelligent roads. The present investigation selected and imported real-world locations, based on their capacity to best meet the specifications of each scenario.

Based on the assessment outcomes, the developed system manifests as a fitting instrument for the research of C-ITS, with a plethora of adjustable parameters available to proficient users, while simultaneously sustaining a gradual learning curve accomplished through the concurrently packaged frameworks. The scalability of the system is limited by the single-threaded nature of its network and road simulator, resulting in extended simulation times Despite the efficient memory utilization of the stack, which is designed to cater to large networks from the ground up, the execution time required for simulating huge networks on thousands of C-ITS stations would face intense bottlenecking. It is noteworthy that active development is ongoing for all bundled frameworks in the realm of C-ITS.

Accordingly, one may anticipate future advancements in C-ITS research. Notably, the developed solution architecture exhibits modularity and compartmentalization, thus indicating the potential for it to obtain benefits from forthcoming advancements. A noteworthy enhancement for ns-3 and SUMO would involve facilitating multi-threaded support.

Finally, owing to its impressive simulation realism, this technology presents a compelling alternative solution for studying, developing, and testing intricate C-ITS systems and applications without the need for costly test site infrastructure. Regrettably, discernible from the workforce implicated in the production of the entire simulation suite, the development, and testing of C-ITS systems is predominantly a matter for automobile manufacturers and research institutions. When examining the current curricula offered by universities, it becomes apparent that C-ITS-related studies are not yet prevalent. This is likely due to the multi-disciplinary nature of the field, which requires a blend of expertise in Computer, Electronic, and Civil Engineering, in addition to consideration of the social implications of cooperative applications. The topic of mobility is a crucial element of contemporary society, and its multifaceted nature is vast and intricate. To effectively address this field, it is imperative to garner the expertise of disparate specialists. Our driving force has consistently been to bridge the knowledge gap and unify individuals with a shared objective, namely, the development of an innovative road system for the future. Due to these reasons, the proposed solution also aims to establish a simulation platform intended to function as an instrument capable of facilitating instruction and learning on topics on C-ITS. Given the significant number of fatalities that continue to plague our roadways, we hope that our efforts and contributions may enhance the expediency of addressing this pressing public safety concern.

The progression of upcoming technologies is paramount in achieving the intended advantages they offer.

Future Works

The modularity of the simulation package enables the autonomous development of distinct components, facilitating collaboration among diverse domains in future endeavors.
In light of inadequacies within the RSU support framework, a subset of the Day 1 Services applications was not adequately developed.
The primary objective of this research was not to analyze the advantages associated with the implementation of intricate Cooperative Intelligent Transport Systems (C-ITS) within our road networks. Rather, the focus was geared toward establishing the groundwork for future investigations in this regard. Consequently, the utilization of the bundled simulation tool enables the undertaking of more intricate investigations. It is conceivable that a comprehensive investigation could be undertaken, utilizing the refined urban setting to fully employ and assess GLOSA (Green Light Optimized Speed Advisory) concerning the reduction of gas emissions, minimization of travel time, and enhancement of traffic flow efficiency.
Based on the pertinent literature, we suggest undertaking the following projects that are relevant to the existing simulation stack.

There are several potential ways to improve and broaden the C-ITS emulation framework based on the area speed advisor in future work. First off, adding real-time data feeds from multiple sources, like traffic sensors, linked cars, and infrastructure systems, will greatly improve the speed advisory system's accuracy and dependability. As a result, the framework would be able to dynamically adjust to shifting traffic conditions and give drivers more accurate speed suggestions.

Additionally, the framework's prediction skills might be enhanced by incorporating sophisticated machine learning and artificial intelligence algorithms. The system may proactively anticipate traffic congestion, incidents, or unfavorable weather conditions by utilizing historical traffic data, weather conditions, and other pertinent criteria. This would enable it to deliver more proactive and context-aware speed warnings.

Enhancing parameterization for simulation would be another important area to concentrate on in future studies in addition to the aforementioned.

To enable more intricate and adaptable simulations, this entails improving and extending the set of variable parameters inside the C-ITS emulation framework.

Researchers and practitioners would have more freedom to define diverse simulation scenarios, such as variable road layouts, traffic demand patterns, driver behaviors, and environmental conditions, by improving parameterization. This would make it possible to conduct a more thorough examination of how these elements would affect the effectiveness of the area speed advisor and how it will integrate with other C-ITS technologies.

References

[1] Quddus, M. A., Ochieng, W. Y., & Noland, R. B. (2010). "Current map-based and map-less approaches to vehicle navigation systems. Transportation Research Part C: Emerging Technologies", 18(4), 536-547.

[2] Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). "Recent development and applications of SUMO—Simulation of urban mobility. International Journal on Advances in Systems and Measurements", 5(3&4), 128-138.

[3] PTV Group. (2021). PTV Vissim - Traffic Simulation Software. Retrieved from https://www.ptvgroup.com/en/solutions/products/ptv-vissim/.

[4] Sommer, C., Eckhoff, D., & Dressler, F. (2011). IVC in cities: "Signal-based signal processing for improved congestion control". *IEEE Transactions on Mobile Computing,* 10(3), 353-365.

[5] Varga, A., & Hornig, R. (2008). "An overview of the OMNeT++ simulation environment. In Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops" (pp. 1-10).

[6] ns-3 Consortium. (2023). ns-3: A discrete-event network simulator for Internet systems. Retrieved from https://www.nsnam.org/.

[7] R. S.T.Rakkesh, A.R.Weerasinghe. (2016)"A Decentralized Vehicle Re-routing Approach using Vehicular Ad-hoc Networks,*"*.

[8] M. V. Hamed Noori. (2013) "Impact of VANET Based V2V/V2I Communication Using IEEE 802.11p on Reducing Traveling Time in Realistic Large Scale Urban Area,".

[9] R. G. David Eckhoff, Bastian Halmos. (2013) "Potentials and Limitations of Green Light Optimal Speed Advisory Systems,".

[10] European Telecommunications Standards Institute, "Cooperative intelligent transport system," https://ec.europa.eu/transport/themes/its/c-its.

[11] Figueiredo, L., Jesus, I., Machado, J. T., Ferreira, J. R., & De Carvalho, J. M. (2001). "Towards the development of intelligent transportation systems." ITSC 2001. 2001 *IEEE intelligent transportation systems. Proceedings* (Cat. No. 01TH8585), 1206-1211.

[12] ETSI, “ETSI EN 302 665 (2010) "Intelligent Transport Systems (ITS); Communications Architecture,” *ETSI*, vol. 1, pp. 1–44.

[13] A. Festag, (2015). “Standards for vehicular communication—from IEEE 802.11p to 5G,” *e & i Elektrotechnik und Informationstechnik* 132, 409–416.

[14] H. Zimmermann, (1980). “OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection,*” IEEE Transactions on Communications* 28, 425–432.

[15] B. Hinden and D. S. E. Deering, (1998) “Internet Protocol, Version 6 (IPv6) Specification,”, RFC 2460.

[16] J. Santa, F. Pereniguez, A. Moragon, and A. F. Skarmeta, (2013) “Vehicle-to-infrastructure messaging proposal based on cam/denm specifications,” 2013 IFIP Wireless Days (WD).

[17] *V. Communications and A. B. Service,* “Vehicular Communications; Basic Set of Applications"; (2010) Part 2: *Specification of Cooperative,” History*, vol. 1, pp. 1–22,

[18] ITS Norway ‘’https://its-norway.no’’

[19] J. Santa, F. Peren˜´ıguez, A. Morago´n, and A. F. Skarmeta, (2013) “Vehicle-to-Infrastructure Messaging Proposal Based on CAM / DENM Specifications,”.

[20] ETSI, “ETSI TR 102 638 V1.1.1 (2009-06): "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions,” *ETSI, Sophia Antipolis Cedex, France*, vol. 1, pp. 1–81, 2009.

[21] European Commission, (2016) “C - ITS Platform: Final Report,” no. January, p. 140.

[22] Asselin-Miller, N., Biedka, M., Gibson, G., Kirsch, F., Hill, N., White, B., & Uddin, K. (2016). "Study on the deployment of C-ITS in Europe: final report framework contract on impact assessment and evaluation studies in the field of transport study on the deployment of cooperative intelligent transport systems (C-ITS) in Europe": final report (No. 5). A3/119-2013-Lot.

[23] European Commission, (2016) “C - ITS Platform Specifics Portuguese,” no. January, p. 94.

[24] K. Katsaros, (2011) “Performance study of a Green Light Optimal Speed Advisory (GLOSA) Application Using an Integrated Cooperative ITS Simulation Platform,” *Proceedings of Wireless Communications and Mobile Computing Conference* (IWCMC), pp. 918–923.

[25] P. Thubert, A. Petrescu, R. Wakikawa, and V. Devarapalli, (2005) “Network Mobility (NEMO) Basic Support Protocol,”, RFC 3963,

[26] D. B. Johnson, J. Arkko, and C. E. Perkins, (2011) “Mobility Support in IPv6,”, RFC 6275.

[27] "Internet Protocol,”, RFC 791, 1981.

[28] E. T. S. I. (ETSI), (2017) "Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol".

[29] “User Datagram Protocol,”, RFC 768, 1980.

[30] “Transmission Control Protocol,”, RFC 793, 1981.

[31] (ETSI), (2014) "Intelligent Transport Systems (ITS)"; *Vehicular Communications; GeoNetworking; Part 3: Network Architecture*.

[32] S. Kuhlmorgen, I. Llatser, A. Festag, and G. Fettweis, ((2015).) “Performance Evaluation of ETSI GeoNetworking for Vehicular Ad Hoc Networks,” *In 2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pp. 1–6.

[33] “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,” IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012) pp. 1–3534 (2016).*

[34] (ETSI), (2010) Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band.

[35] (ETSI), Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part, 2010.

[36] (ETSI), Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band, 2012.

[37] D. Jiang and L. Delgrossi, (2008) “IEEE 802.11p: Towards an international standard for wireless access in vehicular environments,” *IEEE Vehicular Technology Conference, no. June 2008*, pp. 2036–2040.

[38] C. K. T. Francisco J. Martinez, (2009) “A survey and comparative study of simulators for vehicular ad hoc networks (VANETs),” Wiley InterScience, no. 12, pp. 32 056–32 078.

[39] Harri, J., Filali, F., & Bonnet, C. (2009). Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE communications surveys & tutorials*, 11(4), 19-41.

[40] A. Hassan, (2009) “VANET Simulation,” Electrical Engineering, no. May, p.43.

[41] A. Dahiya, A. Noonia, and B. Singh Jangra, (2014) “Vehicular Ad hoc Networks (VANETS): Simulation and Simulators,” *International Journal of Research in Management, Science & Technology*, vol. 2, no. 1, pp. 2321–3264.

[42] Malinverno, M., Raviglione, F., Casetti, C., Chiasserini, C. F., Mangues-Bafalluy, J., & Requena-Esteso, M. (2020, November). "A multi-stack simulation framework for vehicular applications testing". *In Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications* (pp. 17-24).

[43] SUMO Contributors. (2023). SUMO: Simulation of Urban Mobility. Retrieved from https://sumo.dlr.de/

[44] MS-Van3t,” https://github.com/ms-van3t-devs/ms-van3t.git

[45] ETSI, T. 102 636-3 V1. 1.1 (2010-03) Technical Specification. Intelligent Transport Systems (ITS).

[46] OpenStreetMap Contributors. (2023). Map of Stavanger. OpenStreetMap. Retrieved from https://www.openstreetmap.org.

Appendix

To know more about the Ms-Van3t Project and the entire framework on which this implementation is based on, refer to [42] and [44].