UNIVERSITY OF THESSALY
POSTGRADUATE STUDIES PROGRAM

SOFTWARE ENGINEERING FOR DESKTOP AND MOBILE APPLICATIONS

SIMULATION SCENARIOS ON AUTOMATED VEHICLES USING LTE TECHNOLOGY

DIPLOMA THESIS

VASILIKI KYDONA (AM 7419010)

Supervisor: Konstantinos Chaikalis

ΛΑΡΙΣΑ 2021

«Εγώ ο/η <Βασιλική Κυδώνα>, δηλώνω υπεύθυνα ότι η παρούσα Πτυχιακή Εργασία με τίτλο <Σενάρια Εξομοίωσης Αυτοκινούμενων Οχημάτων με την χρήση του LTE> είναι δική μου και βεβαιώνω ότι:

- *Σε όσες περιπτώσεις έχω συμβουλευτεί δημοσιευμένη εργασία τρίτων, αυτό επισημαίνεται με σχετική αναφορά στα επίμαχα σημεία.*
- *Σε όσες περιπτώσεις μεταφέρω λόγια τρίτων, αυτό επισημαίνεται με σχετική αναφορά στα επίμαχα σημεία. Με εξαίρεση τέτοιες περιπτώσεις, το υπόλοιπο κείμενο της πτυχιακής αποτελεί δική μου δουλειά.*
- *Αναφέρω ρητά όλες τις πηγές βοήθειας που χρησιμοποίησα.*
- *Σε περιπτώσεις που τμήματα της παρούσας πτυχιακής έγιναν από κοινού με τρίτους, αναφέρω ρητά ποια είναι η δική μου συνεισφορά και ποια των τρίτων.*
- *Γνωρίζω πως η λογοκλοπή αποτελεί σοβαρότατο παράπτωμα και είμαι ενήμερος(-η) για την επέλευση των νόμιμων συνεπειών»*

«I, <Vasiliki Kydona>, responsibly declare that this Thesis entitled < Simulation Scenarios on Automated Vehicle using LTE technology> is mine and I certify that:

- *In all cases where I have consulted a published work of third parties, this is indicated by a relevant reference to the points in question.*
- *In all cases where I quote third parties, this is indicated by a relevant reference to the points in question. With the exception of such cases, the rest of the text of the degree is my work.*
- *I explicitly list all the help sources I have used.*
- *In cases where parts of this dissertation were done jointly with third parties, I explicitly state what my contribution is and which of the third parties.*
- *I know that plagiarism is a very serious offense and I am aware of the legal consequences. »*

< υπογραφή >

< Βασιλική Κυδώνα>

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Τόπος:	....
Ημερομηνία:	....

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. ....
2. ....
3. ....

Summary

Automated vehicles are vehicles without a driver and their aim is to meliorate the safety at roads and free drivers from their attention at the steering wheel, especially in cities. These vehicles use sensors in order to collect data and make decisions for a better driving. Vehicles must share their data through wireless communication using a specific protocol, like 802.11p or LTE. Vehicular Ad Hoc Networks (VANETs) were conceived for meeting this necessity of more security, and at the same time offering a more efficient traffic management.

The aim of this diploma thesis is to demonstrate how these vehicles exchange data messages when driving in an heterogeneous network using LTE Technology for V2I communication and 802.11p for V2X.

Acknowledgements

Firstly, I would like to thank my supervisor, Professor Chaikalis, for his trust in assigning me this thesis, his help throughout the accomplishment of it and his motivation.

I also would like to thank my family for the encouragement they provided me with and for their continuous support during all this time of studying.

Last but not least, I would like to thank anyone that, directly or indirectly, had lent their hands on this.

Vasiliki Kydona

25.11.2021

ΠΡΟΣΟΧΗ: Ο πίνακας περιεχομένων παράγεται **αυτόματα** με κατάλληλες εντολές. ΜΗΝ κάνετε edit μέσα σε αυτό τον πίνακα γιατί ότι γράψετε θα χαθεί στο πρώτο update. Το update γίνεται με δεξί κλικ πάνω σε κάποιο από τα περιεχόμενα + Update Field (Ενημέρωση Πεδίου) + Πλήρης ενημέρωση πίνακα (όχι μόνο σελίδες). Σβήστε την παρούσα παράγραφο όταν έχετε εξοικειωθεί με τον πίνακα περιεχομένων.

Βασικό: Δεν έχει νόημα να έχετε ενότητα π.χ. 2.4.1 αν δεν σκοπεύετε να έχετε και 2.4.2. Οι ενότητες 1ου ή 2ου επιπέδου έχουν κατά βάση πολύ περιεχόμενο. Αποφύγετε π.χ. ενότητες 2ου επιπέδου μεγέθους μιας ή δύο παραγράφων. Κάτι άσχημο που κάνετε πολλοί είναι να γράφετε Κεφάλαιο 1 μιας σελίδας και να βάζετε μέσα 1.1, 1.2, 1.3 κτλ (μέσα σε μία σελίδα)!!!!

Περιεχόμενα

1 Introduction

Nowadays the road safety is an important matter for all societies worldwide. Vehicular Ad-hoc Networks (VANETs) have attracted attention in the support of safe driving, intelligent navigation, and emergency and entertainment applications. VANET can be viewed as an intelligent component of the Transportation Systems as vehicles communicate with each other as well as with roadside base stations located at critical points of the road, such as intersections or construction sites. [1]

Traditionally, VANET simulators consist of two components. One is the wireless network simulator and the other is the road traffic simulator. The protocol is simulated with the help of veins, that is an open source framework, which can offer unrestricted extensibility. Veins simulator requires OMNeT++ and SUMO framework for simulation. Veins are used because it can be used to solve challenges in VANET [13].

2 VANETS

The Vehicular Ad-Hoc Network, or VANET, is a technology that uses moving cars as nodes in a network to create a mobile network since VANET is a subset of Mobile Ad-Hoc Network (MANET). VANET turns every participating car into a wireless router or node, allowing cars approximately 100 to 300 meters of each other to connect and, in turn, create a wide range of network. Position of the nodes in VANET will be frequently changing due to high mobility and random speed of the vehicles [13].

VANET is a particular case of wireless multihop network, which has the constraint of fast topology changes due to the high node mobility. With the increasing number of vehicles equipped with computing technologies and wireless communication devices, intervehicle communication is becoming a promising field of research, standardization, and development. VANETs enable a wide range of applications, such as prevention of collisions, safety, blind crossing, dynamic route scheduling, real-time traffic condition monitoring, etc. Another important application for VANETs is providing Internet connectivity to vehicular nodes. Figure 1 below shows an example of a VANET. [3]

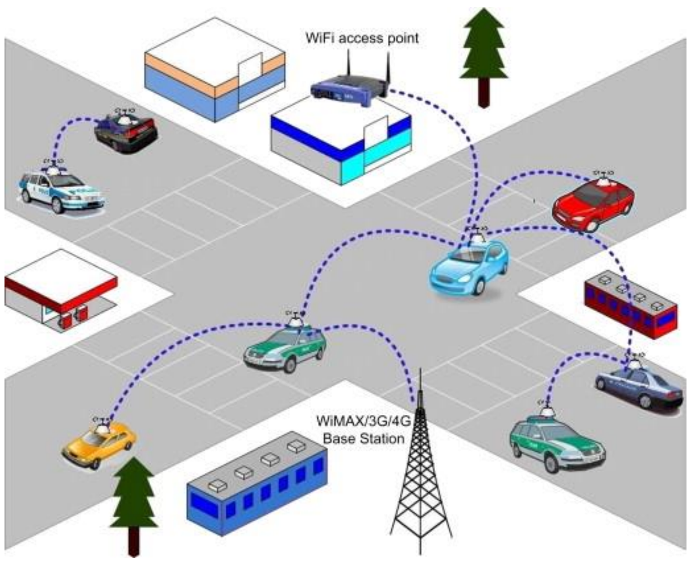


Figure 1 - An example of a VANET

While VANET helps the traveler to have a safe and secure driving experience, by providing them useful mobile information, it implicitly preserves the green environment, e.g. by regularizing fuel consumption rates. A comprehensive well-organized VANET is responsible for extracting, managing and interpreting the information to achieve knowledge, and making it available for travelers. [1]

2.1 Overview of VANETs

The main objective of the VANET is to support vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication modes. The IEEE 802.11p network uses the random-access medium access control protocol carrier-sense multiple access with collision avoidance (CSMA/CA) to support V2V and V2I services. The advantages of the CSMA/CA protocol are in its simplicity, minimum control signaling, and the broadcast nature of transmission. These enable low packet transmission delay at lower teletraffic load. However, due to the lack of coordination among transmitters, packet collisions can occur. And thus, increase the packet transmission delay and reduce the packet delivery ratio. [2]

In Fig. 2, we can see an example of a group of vehicles communicating at an intersection using a V2V system. Since this system only supports V2V communication, this network cannot use any of the capabilities of the infrastructure devices within its range. In Fig. 3, we can see an example of a V2X system in action. Since this is a V2X system, the vehicles can now also use the nearby cellphone tower to send and receive long distance packages. Additionally, they can communicate with infrastructure devices installed in the traffic lights, which may contain detailed information about this intersection. Both these modes of operations are subsets of VANETs, with more modern works focusing on development for V2X capabilities. [4]

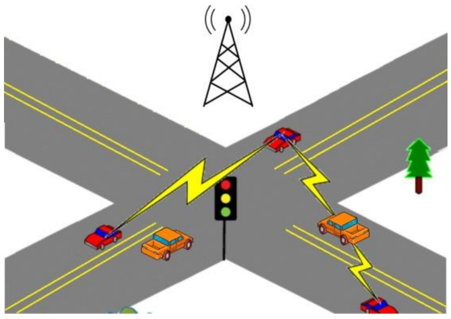


Figure 2 - An example of a VANET with V2V communications only

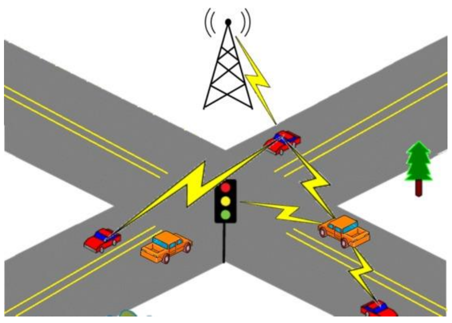


Figure 3 - An example of a VANET with V2X communications

2.2 Architecture

With respect to the IEEE 1471-2000 [5] and ISO/IEC 42010 [6] standards, VANET components are classified into three domains:

i) **Mobile domain** includes the vehicle and the mobile device domains. The former comprises all type of vehicles (e.g., cars, trains, buses). The latter includes all types of portable devices (e.g., smartphones, laptop, smart watches).

ii) **Infrastructure dom**ain incorporates the roadside infrastructure domain (eg. Traffic light, camera, etc.) and the central infrastructure domain (Traffic Management Centres (TMCs), Vehicle Management Centres)

iii) **Generic domain** includes the Internet and the Private infrastructures. [7]

The European architecture standard for VANETs is little different. In fact, it relies on the CAR-2-X communication system pursued by the CAR-2-CAR Communication Consortium [8]. As shown in Figure 4, the reference architecture of the C2C Communication System, comprises the following domains:

- In-vehicle domain is composed of one or multiple application units (AUs) and one Onboard Unit (OBU). An AU is a dedicated device, which can be an integrated part of a vehicle or a separate portable device such as smartphone, laptop, etc. It runs one or many applications that exploit the OBU communication capabilities. The AUs and OBU are permanently connected through a wired or wireless connection.
- Ad-hoc domain is composed of vehicles equipped with OBUs and stationary Road-Side Units (RSUs) deployed in specific locations along the road. OBUs can

communicate each with other, directly or via multi-hop, using wireless short-range communication devices allowing ad-hoc communications between vehicles. An RSU is a stationary device that can be connected to an infrastructure network or to the Internet. It can send, receive or forward data in the ad-hoc domain (i.e., vehicles equipped with OBUs and RSUs), which enables to extend the coverage of the ad-hoc network. An OBU may access to the Internet via an infrastructure connected RSU, public commercial or private wireless Hot Spots (HSs) to communicate with Internet nodes or servers.

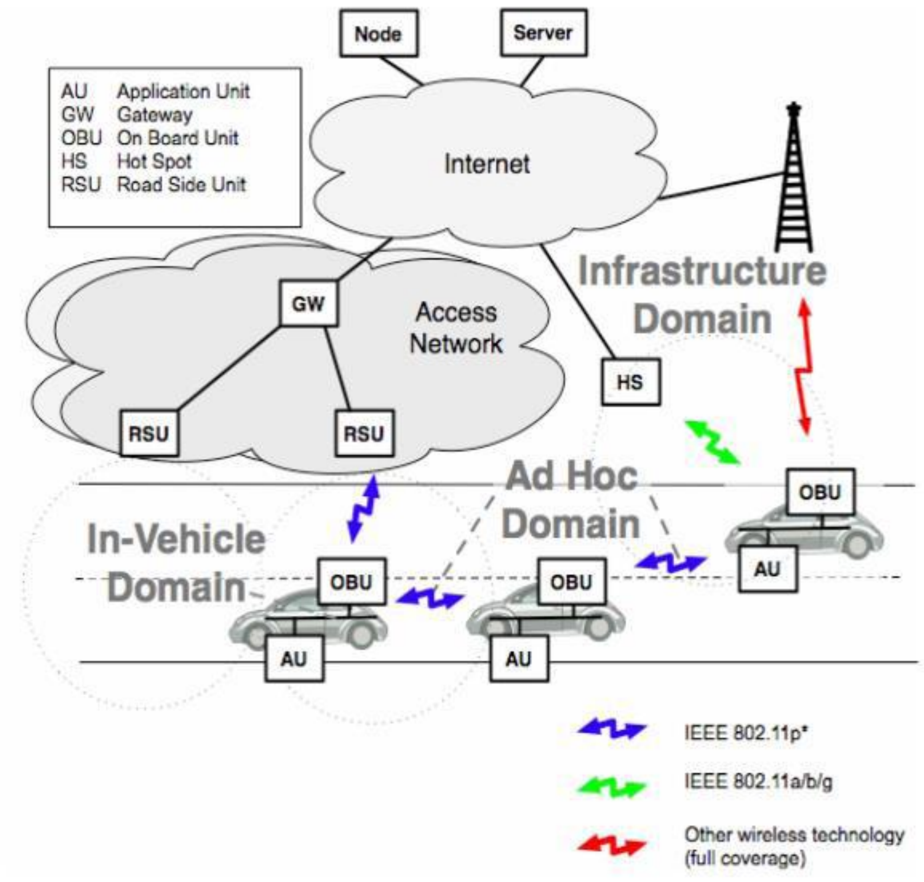


Figure 4 - C2C-CC reference architecture

- Infrastructure domain access consists of HSs and RSUs. In case that neither RSUs nor HSs provide Internet access, OBUs can exploit cellular radio networks for example 4G.

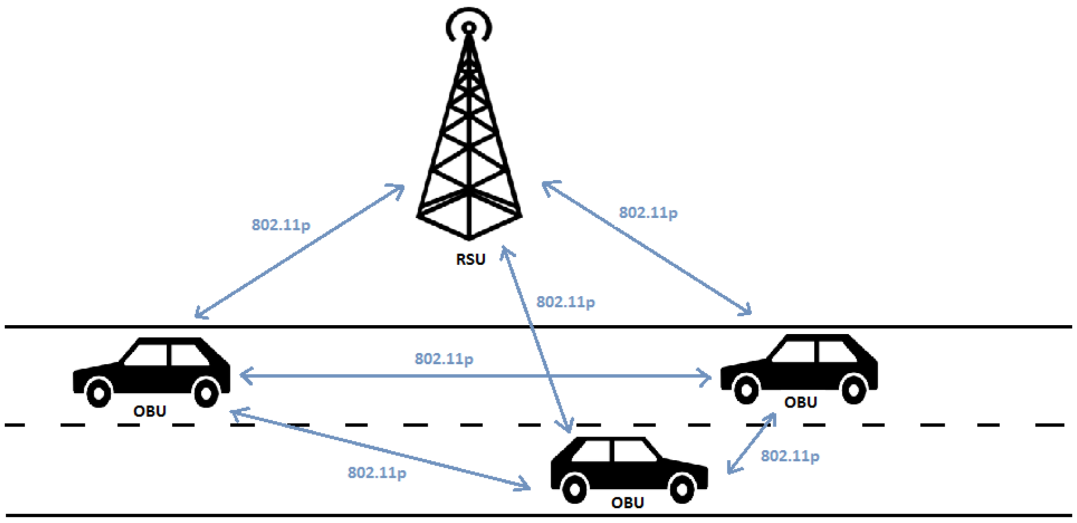


Figure 5 - VANET example with three OBUs and one RSU

In figure 5 above, we can see a road with vehicles equipped with OBUs connecting each other, as well as with the roadside unit. This would be a simple scenario where cars exchange their positions in order to improve this road's traffic efficiency [14].

2.2.1 Communication Architecture

VANET communications are categorized as follows [9]:

- In-vehicle communication between OBU of the vehicle and its AUs.
- Vehicle-to-vehicle (V2V) wireless communications between vehicles via their OBUs.
- Vehicle-to-infrastructure (V2I) refers to bidirectional wireless communications between vehicles and infrastructure-connected RSUs.
- Infrastructure-to-Infrastructure (I2I) communications between RSUs enable extending the coverage of the network.
- Vehicle-to-broadband cloud (V2B) communications between vehicles and broadband cloud via wireless broadband technologies such as 3G/4G.

2.3 Characteristics of VANETs

VANET is an application of MANET but it has its own distinct characteristics which can be summarized as [11]:

- High Mobility: The nodes in VANETs usually are moving at high speed. This makes harder to predict a node's position and making protection of node privacy [10].
- Network topology: Due to high node mobility and random speed of vehicles, the position of node changes frequently. As a result of this, network topology in VANETs tends to change frequently.
- Unbounded network size: VANET can be implemented for one city, several cities or for countries. This means that network size in VANET is geographically unbounded.
- Frequent exchange of information: The ad hoc nature of VANET motivates the nodes to gather information from the other vehicles and road side units. Hence, the information exchange among node becomes frequent

2.4 VANET Applications

VANET applications are categorized as safety, traffic efficiency and infotainment applications. Figure 6 illustrates a taxonomy of VANET applications and related use cases.

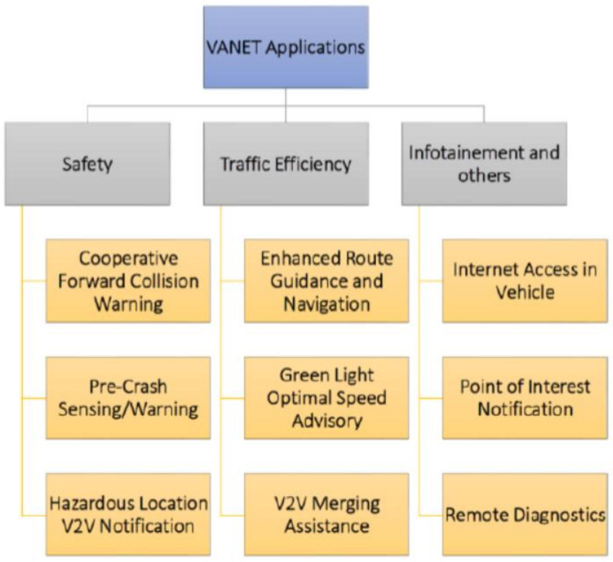

VANET Applications
Safety
Traffic Efficiency
Infotainement and others
Cooperative Forward Collision Warning
Pre-Crash Sensing/Warning
Hazardous Location V2V Notification
Enhanced Route Guidance and Navigation
Green Light Optimal Speed Advisory
V2V Merging Assistance
Internet Access in Vehicle
Point of Interest Notification
Remote Diagnostics

Figure 6 - Taxonomy of VANET applications

2.4.1 Safety Applications

Safety applications aim to warn drivers at the right time about dangerous situations in the road in order to enhance driving safety.

• Cooperative Forward Collision Warning: the goal of this use case is to avoid rear-end collisions with other vehicles by providing assistance to drivers. In fact, rear-end collisions are generally caused by driver disturbance or sudden braking. To avoid a crash, concerned vehicles share relevant information such as position, speed and direction. When a critical situation (e.g., insufficient safety distance) is detected, the vehicle warns its driver [7].

• Pre-Crash Sensing/Warning: Unlike the Cooperative Forward Collision Warning, this use case assumes that a crash is unavoidable and will take place [12]. In such cases, the involved vehicles exchange in an efficient way the related information with neighboring vehicles in order to enable a better usage of vehicle's actuators.

• Hazardous Location V2V Notification: the goal of this use case is to share pertinent information about dangerous locations on the roadway (e.g., potholes, bottleneck, etc.) between vehicles in certain area. To this end, the vehicle detecting a dangerous location uses the information for optimizing its safety systems then broadcasts it to neighbouring vehicles in the surrounding area. Through V2V communications, the information is then progressively shared with other concerned vehicles. Information about dangerous locations on the roadway can also be transmitted from external service providers to RSUs, which in turn send it to some vehicles in their communication ranges. Thereafter, vehicles receiving the information can disseminate it to others via V2V communications [7].

2.4.2 Traffic efficiency applications

Traffic efficiency applications aim to enhance the efficiency of transportation systems by providing traffic related information to drivers or road operators. In order to achieve this goal, traffic information should be exchanged through the VANET. Therefore, road-users and road operators will respectively benefit from shorter travel times as well as from reduced costs of roads construction and maintenance [7].

• Enhanced Route Guidance and Navigation: it enables the infrastructure owner to collect traffic data of a large region to be used later for predicting traffic congestion on roadways. Predicted information will be then transmitted to vehicles via RSUs. Hence, driver will be notified about the current and the expected traffic throughout the region, expected delays to reach his destination and better routes that might exist to avoid some congested roads. This will undoubtedly lead to improve the overall efficiency of the transportation system.

• Green Light Optimal Speed Advisory: it provides information related to the location of a signalized intersection and the signal timing (i.e., time to switch the light signal) to vehicles approaching the intersection, which contributes to smoother driving and avoid stopping. Receiving such information at the right time, the vehicle can calculate the optimal speed to reach the intersection when the traffic signal is green, and therefore the driver will not have to decrease the speed of the vehicle or to stop. This fact will probably bring about a significant increase in the traffic flow and fuel economy.

2.4.3 Infotainment and others

Non safety or traffic efficiency use cases are classified in this category. Some of these use cases provide entertainment or information on a regular basis to drivers. Other ones are transparent to the driver and they play important role for improving the vehicle functions.

• Internet Access in Vehicle: it allows drivers and eventually passengers to access the Internet via the VANET. In this case, RSUs act as internet gateways.

• Point of Interest Notification: it allows traders and advertisement companies to advertise their business promotions to nearby vehicles. To this end, an RSU broadcasts the advertisement information (e.g., location, hours of operation and pricing) to the contacted vehicles. The received advertisements will be filtered by each vehicle with respect to the driver profile and context then appropriate advertisements are presented to the driver [7].

3 Network

Network access in the Internet includes the physical and data link layers of the OSI model [15]. Since wireless communications are inherently broadcast, a MAC layer has to be implemented in practically all IVC systems. In IVC systems fast and slow fading effects can be expected to cause problems as both the transmitter and receiver are moving, potentially toward each other. Also, the antennas of both the transmitter and receiver are closer to the ground than the typical cellular base station.

Today, there are several communication standards that may be used as access networks for IVC systems, such as Bluetooth, IEEE 802.11, and cellular mobile networks (e.g.,

GSM, GPRS, and 3G) [16].

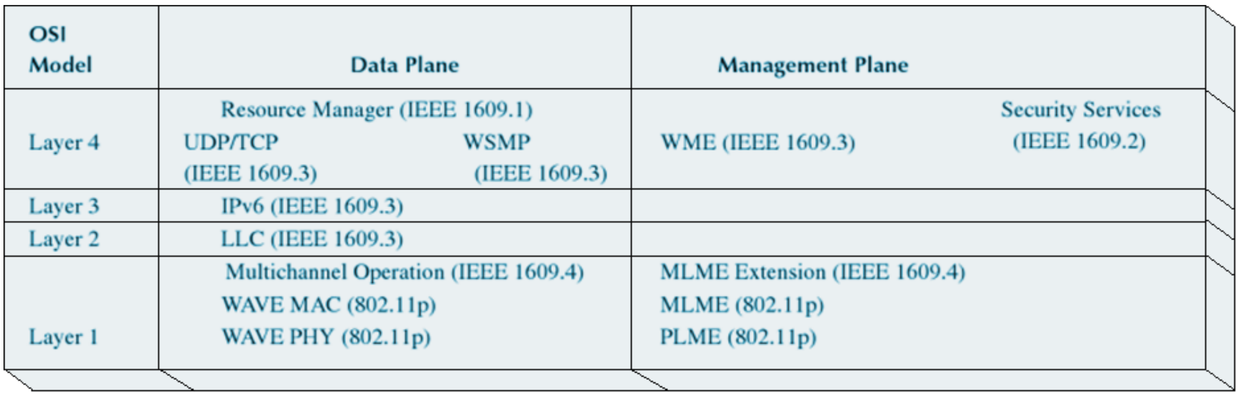
IEEE 802.11p is the standard that supports ITS applications in Vehicular Ad hoc Networks (VANETs). Due to its limited radio range and without a pervasive roadside communication infrastructure, 802.11p can only offer intermittent and short-lived V2I connectivity. LTE and LTE-A are the main candidate access technologies which have different characteristics and can match the vehicular application's requirements, more or less effectively [18].

3.1 802.11p

The de-facto standard for V2x is Direct Short Range Communication (DSRC) wireless technology, which is based on the IEEE 802.11p standard, the 1609 Wireless Access in Vehicular Environment (WAVE) protocol in the U.S., and the European Telecommunications Standards Institute (ETSI)TC-ITS European standards [17]. WAVE is the wireless technology that can carry a best performance in vehicular networks [14].

IEEE 802.11p was designed, from the beginning, to meet every V2x application requirement with the most stringent performance specifications. In 1999, the U.S.Federal Communications Commission (FCC) set aside 75 MHz of bandwidth, in the 5.9 GHz region, for V2x, and the IEEE 802.11p standard operates within this range. The standard was approved in 2009, and since then, there have been a number of field trials. Several semiconductor companies, including Autotalks, NXP Semiconductors, and Renesas, have also designed and tested 802.11p-compliant products [17].

The protocol stack of this WAVE technology is represented in Fig.6. The physical (PHY) layer the Physical Layer Management Entity (PLME) is defined by IEEE 802.11p. The same protocol defines the medium access control (MAC) layer. Also in MAC layer, and as a MAC Sublayer Management Entity (MLME) extension, there is the 1609.4, a protocol that supports multi-channel operations and channel switching procedures by providing frequency band coordination and management. The Logical Link Control (LLC) is defined by IEEE 802.2 protocol to manage and ensure the integrity of data transmissions. Then, for network layer WAVE makes use of IPv6 and for transport layer it uses UDP/TCP protocols. The WAVE Short Message Protocol (WSMP) is defined by two other IEEE 1609 protocols. 1609.2 is the protocol in charge of the security services for applications and management messages. It defines secure message formats and processing and defines the situations for using secure messages. 1609.3 is used for networking services, addressing and routing. This protocol defines short messages for WAVE too, giving an alternative to IPv6. Finally, on the top of the WAVE protocol stack, there is 1609.1, a resource manager for OBUs interaction.



OSI Model	Data Plane	Management Plane
Layer 4	Resource Manager (IEEE 1609.1) UDP/TCP (IEEE 1609.3) WSMP (IEEE 1609.3)	WME (IEEE 1609.3) Security Services (IEEE 1609.2)
Layer 3	IPv6 (IEEE 1609.3)	
Layer 2	LLC (IEEE 1609.3)	
Layer 1	Multichannel Operation (IEEE 1609.4) WAVE MAC (802.11p) WAVE PHY (802.11p)	MLME Extension (IEEE 1609.4) MLME (802.11p) PLME (802.11p)

Figure 6 - WAVE protocol stack

The main modifications in 802.11p, when compared with the traditional 802.11 are [19]:

- In the MAC layer, the overhead to establish a communication was reduced, due to the reduced time of contact between vehicles.

- In traditional 802.11, devices connected through an access point define a group called IBSS (Independent Basic Service Set), which must be identified when a connection is established. On the other hand, 802.11p defines a new type of BSS called WBSS (WAVE BSS), which has a fixed identifier and transmits beacons on demand. A beacon contains the essential information to establish a communication, as well as the list of services offered by the group, eliminating the authentication process.
- in order to eliminate the need of scanning the channels in order to find the desired network, the functions of each channel are fixed.

3.1.1 Channels in DSRC

In DSRC, OFDM with a 10MHz channel is used for data modulation in the physical layer, while in 802.11a, 20 MHz channel bandwidth is the standard. The use of a smaller channel bandwidth has several benefits, such as the possibility to overcome some cases for Root Mean Square (RMS) delay spread, since on a 20 MHz band, inter symbol Interference (ISI) might be present which leads to the necessity of using more advanced adaptive equalization algorithms. While in the MAC layer, Enhanced Distributed Channel Access (EDCA) has been followed. The total spectrum for DSRC is separated into seven channels, which can be distinguished as a control channel (CH) and Service Channels (SH).

Illustration of these channels and their differences is shown on Figure 7 [20].

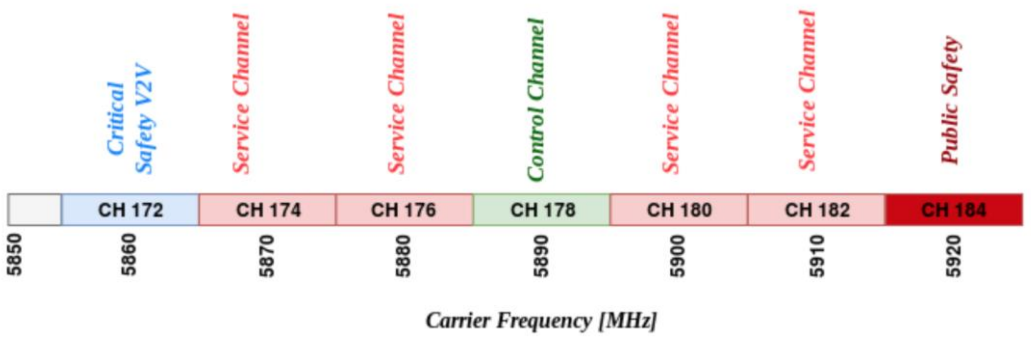
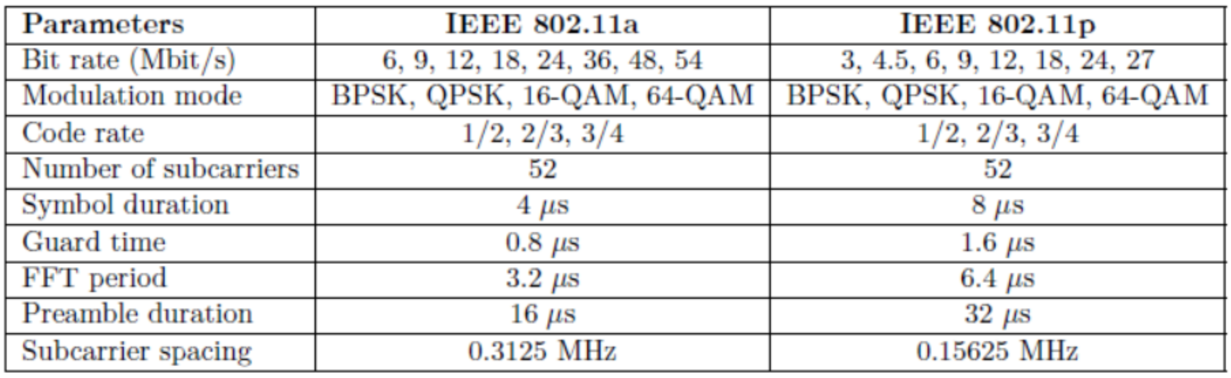


Figure 7 - Channel types and spectrum in DSRC

Channels 180 and 182 are assigned for authorized agencies usage only. Channel 184 is designed for long range communication, which is required in certain cases. On the other hand, for safety applications channel 176 is assigned, and channel 178 would assist for controlled messages. However, there is a strict rule that only a channel of 10 MHz bandwidth can be used and no other combination [20].

3.1.2 802.11p PHY layer

The 802.11p PHY is defined by the 802.11 OFDM PHY. The PHY properties of 802.11p are based on the already widely used IEEE 802.11a.



Parameters	IEEE 802.11a	IEEE 802.11p
Bit rate (Mbit/s)	6, 9, 12, 18, 24, 36, 48, 54	3, 4.5, 6, 9, 12, 18, 24, 27
Modulation mode	BPSK, QPSK, 16-QAM, 64-QAM	BPSK, QPSK, 16-QAM, 64-QAM
Code rate	1/2, 2/3, 3/4	1/2, 2/3, 3/4
Number of subcarriers	52	52
Symbol duration	4 μs	8 μs
Guard time	0.8 μs	1.6 μs
FFT period	3.2 μs	6.4 μs
Preamble duration	16 μs	32 μs
Subcarrier spacing	0.3125 MHz	0.15625 MHz

Figure 8 - Parameters of 802.11a and 802.11p

In the standard, for the 52 subcarriers OFDM has, there are defined the different modulation schemes that OFDM allow: binary phase shift keying (BPSK), quadrature phase shift keying (QPSK), 16 quadrature amplitude modulation (16-QAM) and 64-QAM. These modulations are set according to the data rate and they are unchangeable. The forward error correction (FEC) is implemented with 1/2, 2/3 or 3/4 code rates [14].

This standard doesn't take into consideration the fact that the channel is non-stationary. This means that statistical properties of the channel change over the time. At the working frequency, the channel impulse response main contributions are: line-of-sight (LOS) deterministic scattering, and diffuse scattering [21]. Because of that channel specifications, the protocol has to use estimators, which performance is influenced by the diffuse components strength, and the SNR.

3.1.3 802.11p MAC layer

In a VANET it is harder to deploy a MAC scheme that is relying on a centralized controller, e.g., time division multiple access (TDMA), frequency division multiple access (FDMA), or code division multiple access (CDMA). In a centralized, infrastructure-based network, a base station or an access point is responsible for sharing the resources among the users, thereby enabling guaranteed QoS for time-sensitive data traffic. The idea of having a node that could act as a central control unit in a distributed VANET is not appealing because of the high mobility nodes. The central unit would not remain central for long and constantly changing the central unit would require much information exchange and negotiation among the nodes. The negotiation can be expected to incur excessive delay and once a decision is made it is likely to already be outdated. A MAC scheme that does not require a central control unit is CSMA, where each node starts by listening to the wireless channel and transmits only if the channel is free. This scheme is easily deployed in a distributed network, but has one big disadvantage; the nodes could experience unbounded delays due to constantly sensing a busy channel during high utilization periods. This is not acceptable in real-time systems. Real-time systems such as traffic safety applications, call for a deterministic MAC method. We define a deterministic MAC method to be a scheme for which the time from channel access request to channel access has a finite upper bound [22].

The MAC protocol of 802.11 is a stop-and-wait protocol and therefore the sender awaits an acknowledgment (ACK). If no ACK is received due to e.g., the transmitted packet never reaching the recipient, the packet being incorrect at reception, or the ACK being lost or corrupted, a backoff procedure is invoked before a retransmission is allowed. For every attempt to send a specific packet, the size of the contention window (CW) will be doubled from its initial value (CWstart) until a maximum value (CWend) is reached. This is due to the fact that during high utilization periods, it is convenient to spread the nodes that want to send in time. After a successful transmission or when the packet had to be thrown away because the maximum number of channel access attempts was reached, the contention window will be set to its initial value again. In 802.11p different QoS classes are obtained by prioritizing the data traffic within each node. There are four different priority levels implying that each station maintains four queues. These queues have different AIFS and different backoff parameters, e.g., the higher priority, the shorter AIFS. In a broadcast situation, i.e., when packets destined for all nodes are transmitted, none of the receiving nodes will send ACKs in response. Therefore, a sender never knows if anyone has received the transmitted packet correctly, and it will perform at most one backoff (which occurs when a busy channel is sensed at the initial channel access attempt). Hence, at most one backoff decrement will take place for broadcasted packets.

3.2 LTE Standard

The Long-Term Evolution (LTE) of the UMTS (3GPP-TS 36.300) is the de-facto standard for next-generation cellular access networks. To achieve high throughput performance, in addition to an advanced physical layer design, LTE exploits a combination of sophisticated radio resource management functionalities, such as Channel Quality Indicator (CQI) reporting, link rate adaptation through Adaptive Modulation and Coding (AMC), and Hybrid Automatic Retransmission Request (HARQ) [26]. In an LTE cell, a base station or eNodeB (eNB) allocates radio re-sources to a number of User Equipments (UEs), i.e. handheld devices, laptops or home gateways, using Orthogonal Frequency Division Multiplexing Access (OFDMA) in the downlink, and Single-Carrier Fre-quency Division Multiplexing (SC-FDMA) in the uplink. On each Transmission Time Interval (TTI, 1ms), a time/frequency frame of resource blocks (RBs) is allocated to the UEs, in both directions. Each RB carries a variable amount of bytes to/from an UE, depending on the selected modulation and coding scheme. The latter, in turn, is chosen by the eNB based on the Channel Quality Indicator (CQI) report-ed by the UE, which measures how the UEs perceive the channel. The new Release 10, called LTE-Advanced (LTE-A) (3GPP-TS 36.913), besides promising larger bandwidths due to spectrum aggregation, envisages different transmission paradigms, such as Coordinated Multi-Point (CoMP) Scheduling or device-to-device (d2d) communications [23].

Resource allocation and management in LTE/LTE-A is a key performance enabler: carefully selecting which UEs to target, using which modulation scheme, etc., clearly affects the efficiency of the system. Moreover, inter-cell interference coordination, cooperative transmission, antenna selection, resource partitioning among macro and micro-/pico- cells, energy efficiency and battery saving schemes, etc., enable a wealth of declinations of the classical point-to-multipoint LTE scheduling problem. The performance evaluation of resource management schemes for LTE/LTE-A is normally carried out via simulation [23].

One of the advantages of LTE network infrastructure is that base stations nodes (eNodeB in LTE instead of RSUs) are located in higher altitudes than RSUs of IEEE 802.11p, feature that improves dealing with the non-line of sight (NLOS) problem. Also, LTE has an infrastructure-assisted scheduling and access control which contributes towards achieving the vehicular networks requirements [14].

The LTE Standard has also drawbacks. When there is a high cellular traffic load (because of the presence of big number of mobile phones or to a traffic jam), it is difficult

to cope with the delay requirements, something really important in safety applications. Furthermore, if LTE is used in vehicular networks, other user may experience degraded network performance in their mobile phones due to the amount of vehicles sending messages.

3.2.1 LTE Structure

Development of LTE started with release 8 and has been continuously advanced up until release 14 which was introduced in 2017. Different requirements were defined and met by each release which leads to a quite advanced system architecture and an evolution of the eNodeB. LTE radio access is known as E-UTRAN which consists of eNodeB and User Equipment (UE). The Evolved Packet System (EPS) does not support circuit switching but rather works only with packet switching. The EPS consists of E-UTRAN and Evolved Packet Core (EPC) [20].

The main task of the EPC is supplying IP based connection. The EPC is built from a Mobility Management Component (MME), a Serving Gateway (S-GW) and a Packet Data Network Gateway Component (P-GW). The MME's task, besides a Leading Control Component is also security. Meanwhile, the S-GW and P-GW are mainly responsible for handling data transportation to the User Equipment (UE). The operation and structure of LTE are shown in Figure 9.

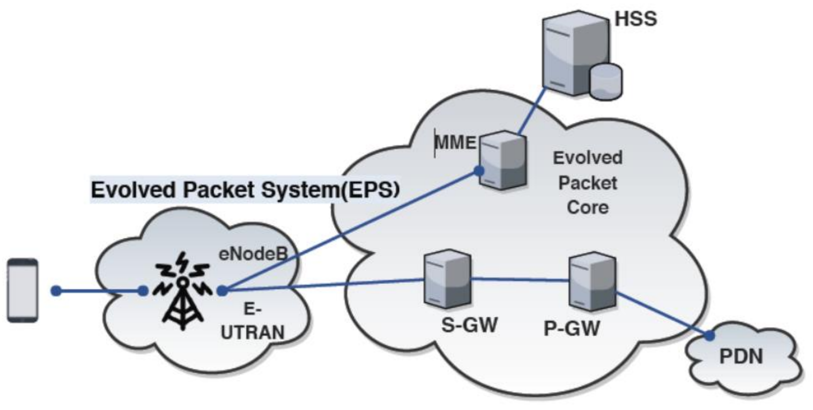


Figure 9 - Structure of LTE

Evolved Node B (eNodeB)

The Evolved Node B (eNodeB) is what the E-UTRAN consists of and handles receiving and sending the radio signal to User Equipment (UE) and includes functionality for packet-switching. The main task is to provide communication between the User Equipment (UE) and the Evolved Packet System(EPS). The ENodeB handles coding and decoding of user protocol information, and manages the duplicated messages reception. It contains few functions and it is managed by Radio Network Controller (RNC).

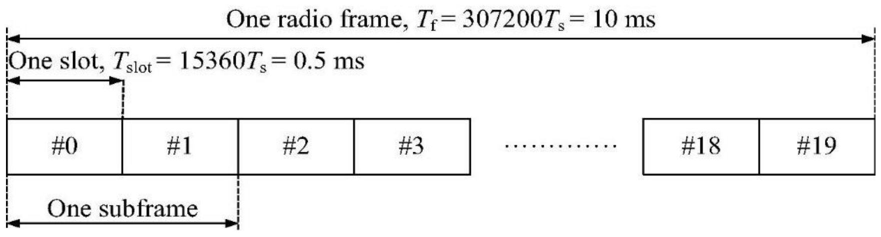
Transmission in Downlink (DL) and Uplink (UL)

Two variants of transmission schemes exist in terms of communication path direction downlink (DL) and uplink (UL). DL represents the link from the base-station towards the UE while the UL is the exact opposite direction of communication, namely from UE towards the base station. The need for high data rate transmission with reliable link quality in DL is met using Orthogonal Frequency Division Multiplexing (OFDM). OFDM is a highly attractive solution that overcomes different issues that were present in the past for cellular networks such as for example Inter-Symbol-Interference (ISI) [20].

3.2.2 LTE Physical Layer

The PHY module resides at the bottom of the LTE protocol stack and implements physical layer related functions, such as channel-feedback reporting and computation, simulation of the air channel, and data transmission and reception. It also stores the node's PHY parameters, like the transmission power and the antenna profile (i.e., whether transmissions are omni-directional or anisotropic). This control over transmission parameters allows one to define so-called heterogeneous scenarios, composed of macro-, micro-, and pico-eNBs, each one having its one radiation profiles [24].

The LTE physical layer supports two types of frame structures as types 1 and 2. The type 1 structure (Fig. 10) is used for Frequency Division Duplex (FDD) mode; however, the type 2 structure (Fig. 11) is applied to Time Division Duplex (TDD) mode maintaining only full duplex operation.



One radio frame, $T_f = 307200T_s = 10$ ms
One slot, $T_{slot} = 15360T_s = 0.5$ ms
#0
#1
#2
#3
#18
#19
One subframe

Figure 10 - Frame structure type 1

Type 1 lasts 10 ms equivalent to 10 subframes (each 1 ms long) or 20 slots (each 0.5 ms long). As in FDD, each frame consists of 10 subframes of 1 ms long and each subframe consists of two concatenated slots of 0.5 ms long. The radio frame used in TDD mode (type 2) also has a length of 10 ms, but it is divided to two half-frames of length 5 ms. Just like the FDD, each subframe of type 2 frame structure also consists of two slots of length 0.5 ms. The special subframe in each half-frame includes three fields; DwPTS (Downlink Pilot Time Slot), GP (Guard Period) and UpPTS (Uplink Pilot Time Slot) [25].

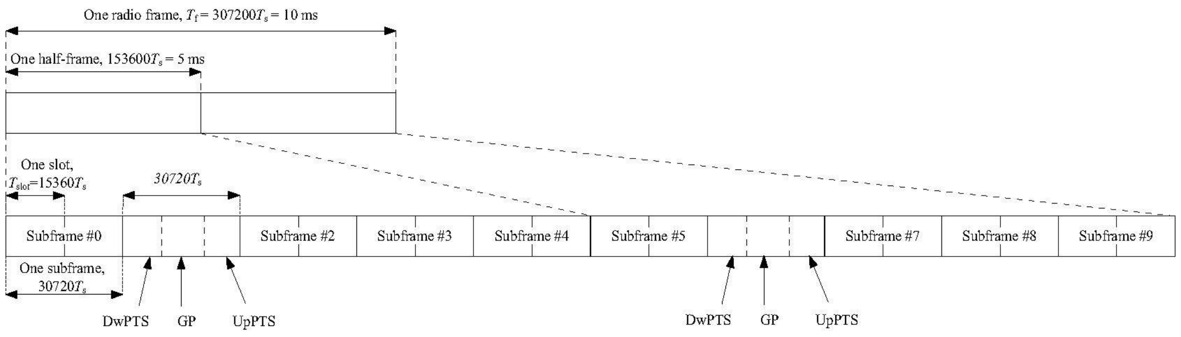


Figure 11 - Frame structure type 2

3.2.3 LTE MAC layer

Most of the intelligence of LTE nodes is implemented in the MAC module. The main tasks of this module are:

- buffering packets coming from lower layers (PHY)
- requesting data for transmission from upper ones (RLC)
- encapsulating MAC SDUs into MAC-PDUs and vice versa
- handling and storing channel feedback
- performing scheduling
- Adaptive Modulation and Coding (AMC)

Most of the operations related to the flow of packets are the same at the UE and the eNB. Scheduling and channel-feedback management, instead, are performed differently on the two nodes.

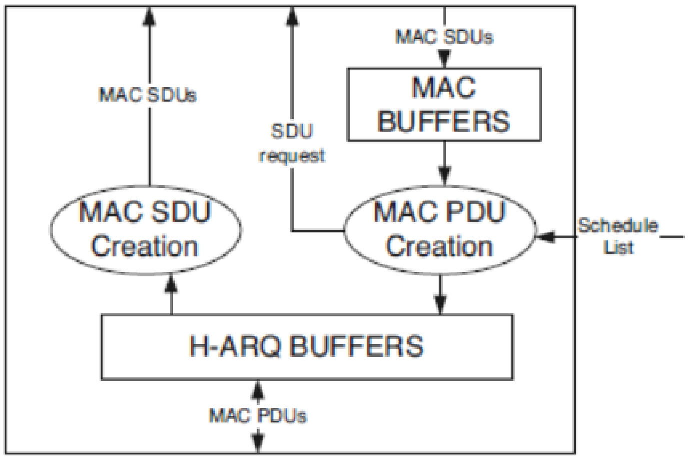


Figure 12 - High-level view of the MAC layer structure

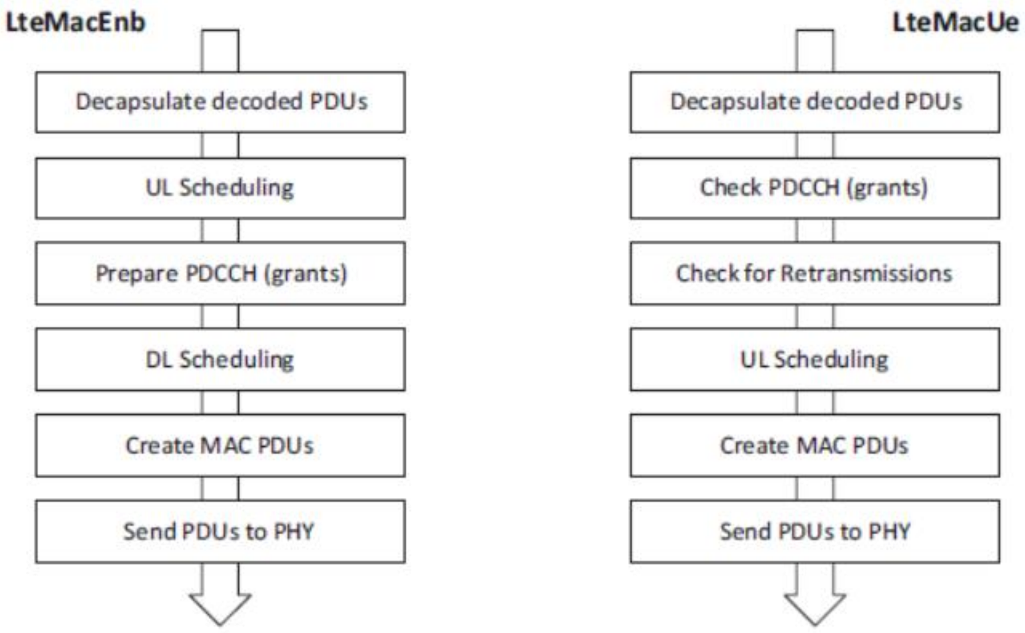


Figure 13 - Main MAC-level operations

On the eNB side, UL connections are scheduled for transmission according to a configurable policy. Scheduling decisions are notified to the UEs via grant messages, i.e., the

PDCCH is not simulated. Similarly, DL scheduling is performed by selecting which connection to serve, and how much data to send. For each scheduled connection, MAC-SDUs are then requested to the RLC layer and encapsulated into MAC-PDUs. The latter are finally stored in the H-ARQ buffers and forwarded to the PHY for transmission. [24]

On the UE side, instead, each UE checks if any grants have been received and decides which local connection will be able to use the granted resources, if any. If no resources are available, it will perform a resource request via the RAC procedure, which is again implemented through messages generated by the MAC module. Then, the UE will check its H-ARQ buffers to see if any transmission is expected for this TTI1 and will proceed with the scheduling of new transmissions otherwise. [24]

H-ARQ buffers are used to store MAC-PDUs that are being sent and received. There is one set of such buffers for transmissions and one for receptions, to which we will refer as TxHbuff and RxHbuff, respectively, and as Hbuff to denote both. [24]

4 Cellular Networks using SimuLTE

Currently, cellular networks are being considered as a livable alternative to other technologies, such as WiFi which traditional mobile applications use, IEEE 802.15.4 and Long Range (LoRa) for sensors and smart things, IEEE 802.11p for vehicular networks, and Asymmetric Digital Subscriber Line (ADSL) for home Internet access. The main advantages of an LTE network are:

- Their nature of being infrastructure based and operator managed which lessens users from the need of deploying and managing a service-specific infrastructure of their own, or making do with the lack of one
- The fact that it operates on licensed spectrum, guaranteeing absence of external interference
- Its built-in features for security, mobility management, and terminal-side power saving
- The progression towards fifth-generation (5G) access and the standardization of Multi-access Edge Computing (MEC)

Evaluating the performance of cellular networks poses several challenges. The fact that LTE includes a whole stack of layered protocols, each one having buffers and timers, which interact with other features, intrinsically defies analytical modeling. On the other hand, building prototypes for live measurements can only scale so much as for number of users and base stations, transmission and computing power, and available bandwidth. So, simulation is the ideal performance evaluation technique. Several simulators of cellular networks have been developed so far. Some are link-level simulators. These do an extensive job of modeling the physical layer of a link, with little or no interest in what is above it. They are good for evaluating signal propagation, spectral efficiency, the impact of transmission techniques such as Multiple Input Multiple Output (MIMO) and interference management, but they are generally unsuitable to understand issues related to resource scheduling, protocol interaction, or application-level performance. Instead, there are some system-level simulators which focus on some modeling details, like signal propagation and communication among protocol layers.

SimuLTE is a system-level simulator based on OMNeT++ for 4G LTE and Long Term Evolution Advanced (LTE-A) networks. SimuLTE focuses particularly on the LTE data plane. It consists of over 40.000 lines of code. [28] Most of the control-plane functions are abstracted by using an oracle, called the *Binder*, which the various modules can challenge to obtain information which would otherwise require control-plane protocols and elements. SimuLTE includes all the protocols of the LTE stack. It models the effects of the physical layer by computing the received Signal-to-Interferenceplus-Noise-Ratio (SINR) at receivers, taking into account all the simultaneous transmitters. It allows both infrastructure-mode communications, where the endpoints of communications are always one User Equipment (UE) and the evolved Node B (eNB), and network-assisted D2D communications, where both endpoints are UEs, and the eNB is in charge of resource scheduling. For the latter, both one-to-one and one-to-many communications are modeled. Within SimuLTE, the LTE functions are confined to a Network Interface Card (NIC), to facilitate the setup of mixed scenarios, where LTE coexists with other layer-2 technologies (like WiFi), or to use LTE as a layer-2 technology in application scenario simulations (e.g., communicating vehicles). [24]

4.1 Architecture of LTE

An LTE network is composed of the Evolved Packet Core (EPC) part and the Radio Access Network (RAN) part, as shown in Figure 12. The EPC is an Internet Protocol (IP)-based network that includes entities performing core functionalities for the network operator, such as Mobile Management Entity (MME), Home Subscriber Server (HSS), and Serving Gateway (SGW).

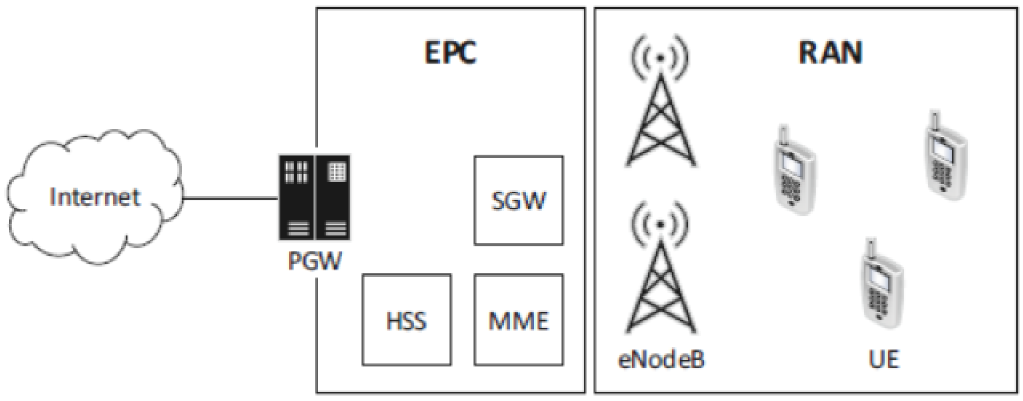


Figure 14 - Architecture of the LTE network

The Packet Data Network Gateway (PGW) allows the EPC to connect to the Internet. The RAN is composed of base stations, called eNBs, which are in charge of resource allocation, and UEs, e.g., smartphones or any device capable of connecting to an LTE network.

Downlink (DL) transmission occurs from the eNB to the UEs, and Uplink (UL) ones in the opposite direction. As for the data-plane transmissions flows, in the Downlink an eNB receives IP packets from the EPC, processes them through the LTE protocol stack, which includes fragmentation and reassembly, and sends them on the air.

The LTE stands in the in the Layer 2 of the Open Systems Interconnection (OSI) stack, while its functions are divided among three protocols (from top to bottom):

- the Packet Data Convergence Protocol (PDCP)
- the Radio Link Control (RLC)
- the MAC

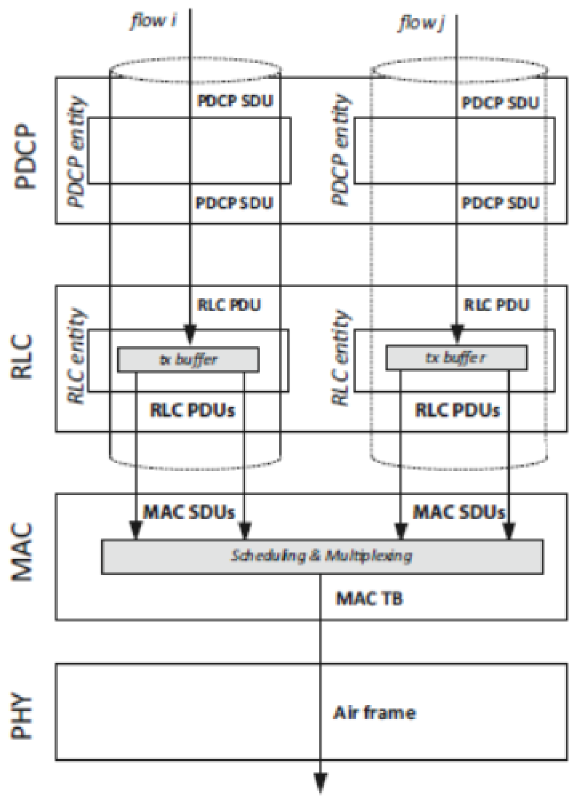


Figure 15 - Top-down traversal of the LTE protocol stack

In figure 13, it is shown the downstream flow of data within the LTE stack, for example the one that an IP packet would undergo on transmission. At each layer, data comes from the upper layer in the form of Service Data Units (SDUs) and goes to the lower layer as Protocol Data Units (PDUs). A PDCP entity is responsible for maintaining numbering and ciphering information. Each PDCP SDU is assigned a PDCP Sequence Number (SN) and ciphered so that only the peering PDCP entity can decode it. Data from the PDCP is sent to the RLC. It is buffered in the RLC transmission buffer, until sent down in the form of RLC PDUs, upon request from the MAC layer. A flow is allocated to a PDCP and a RLC entity at a node, and its PDCP/RLC entities at the transmitter and receiver are synchronized. The RLC can work in one of three modes: Transparent Mode (TM), Unacknowledged Mode (UM), or Acknowledged Mode (AM). The first one does not perform any operation; hence, a RLC SDU corresponds exactly to a RLC PDU. The second one performs segmentation/concatenation of SDUs on transmission, as well as reassembly, duplicate detection, and reordering of PDUs on reception. The third one adds an Automatic Repeat-reQuest (ARQ) mechanism on top of that to ensure reliable delivery of RLC PDUs. [24]

The MAC layer performs scheduling and multiplexing of RLC PDUs coming from different flows and encapsulates them into MAC Transport Blocks (TBs), which are sent over the physical layer. At every Transmission Time Interval (TTI), which is 1 ms in LTE, the MAC scheduler at the eNB assembles the DL and the UL subframes. The latter include a fixed number of Resource Blocks (RBs), e.g., 50 in a 10MHz deployment. In the DL, the MAC scheduler selects which UEs will receive information, how large their TB will be, and which RBs it will occupy. [24]

UL scheduling mirrors the DL one: the eNB allocates transmission grants to UEs having backlogged data, specifying which RBs they can use, using which modulation. However, since data is physically stored at the UEs, a UE needs a signaling method to declare to the eNB its intention to transmit. UEs can append quantized Buffer Status Reports (BSRs) to their scheduled data, to inform the eNB of their residual backlog. UEs can also signal the presence of backlog using an out-of-band Random Access (RAC) procedure. Simultaneous RAC requests from different UEs may collide, in which case UEs undergo a backoff procedure before attempting another RAC. The eNB responds to a RAC request by scheduling the UE in a future TTI. If unanswered, RAC requests are reiterated. The handshake for UL transmissions is summarized in Figure 14.. It consists of up to three parts: a RAC request, usually followed by a (small) grant that the eNB gives to the UE, large enough to contain a BSR, and finally a (larger) grant that the eNB can give to the UE once it knows its backlog.

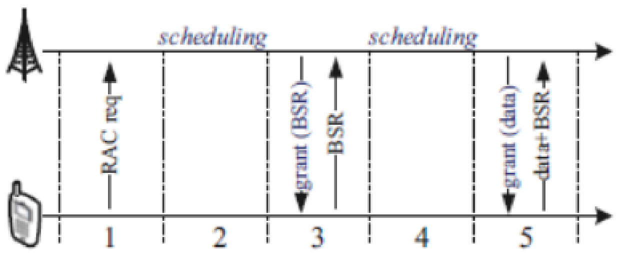


Figure 16 - Handshake for the scheduling of Uplink UE traffic

4.2 Nodes

The two main nodes of a simulated LTE network are eNBs and UEs, which handle all data-plane traffic through the LTE network. SimuLTE adds a third node, namely, the Binder, which acts as the “oracle” of the network, keeping track of communication associations, and which is also used to abstract control-plane operations. LTE nodes, specifically UEs and eNBs, can be created by inserting into them the LTE NIC (Network Interface Card) as a data link interface, and modules from INET for upper-layer protocols. [27] The general structure of a simulated network is shown in Figure 17.

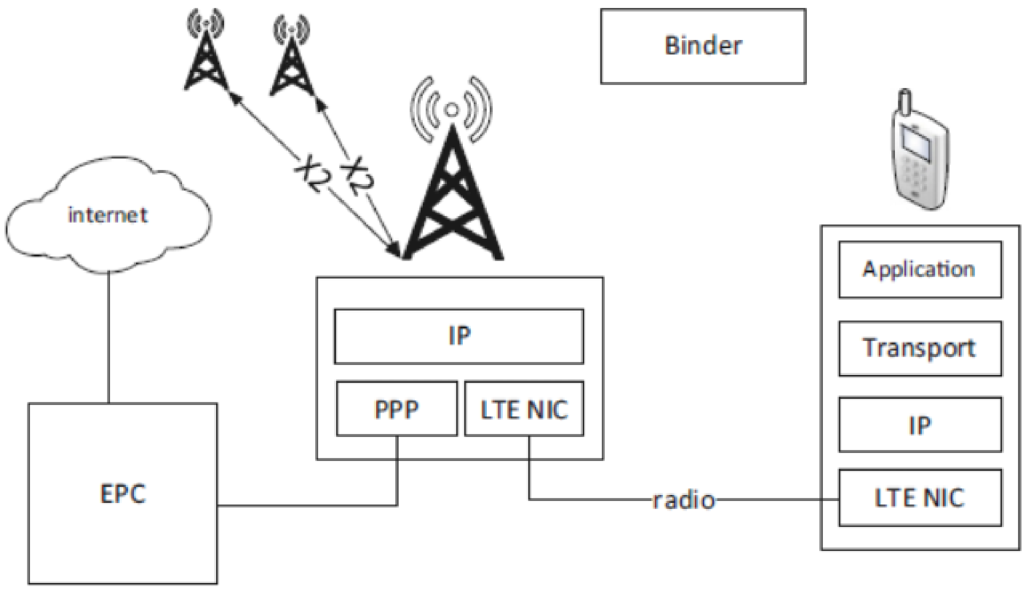


Figure 17 - High-level view of the main simulator nodes

4.3 User Equipment

The UE node represents the behavior of an LTE-based cellphone. Its structure is inspired from INET’s StandardHost, having multiple Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) applications, UDP and TCP modules implementing the respective transport layer protocols. Each TCP/UDP application is one end of a connection, the other end of which may be located within another UE or anywhere else in the simulated network. SimuLTE provides models of reallife applications (e.g., VoIP and VoD), but it can include any TCP/UDP-based OMNeT++ application. The IP module is taken from the INET package as well. Last but not least, an LTE NIC card is used for communication with the eNB or with other UEs in case of D2D communications. [24]

4.4 The Binder

The Binder is a simple module that stores information about every LTE-related node within the system. Its main purpose is to cover all the operations that are not modeled as a message exchange, either to simplify the model or to speed up the simulation process. This includes references to nodes, communication associations among eNBs and UEs, peering associations for D2D pairs, etc.

4.5 LTE NIC

The core SimuLTE module is the LTE Network Interface Card (NIC) and implements the LTE stack within eNBs and UEs. The NIC implementation allows one to develop nodes with multiple connectivity capabilities (e.g., LTE and/or Wi-Fi), and fully embodies the modularity paradigm on which the OMNeT++ framework is based. [27]

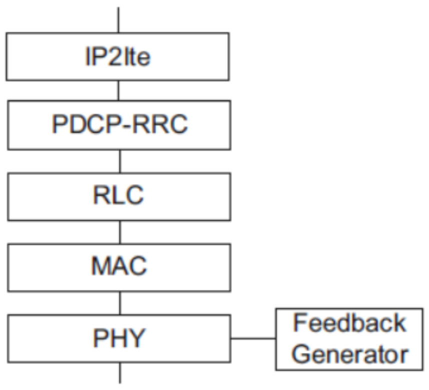


Figure 18 - Internal structure of the LTE NIC module

In figure 18, it is shown the internal structure and connections with other modules, namely, one between the UE and the eNB through an air channel and one with an IP module. The NIC structure is the same for both UE and eNB, the only exception being the FeedbackGenerator, only implemented in the UE. It is responsible for creating channel feedback that is then managed by the Physical Layer (PHY) module. [24]

5 Simulation Tools

In order to simulate scenarios with heterogeneous networks, we need to build a framework with adequate libraries. For the purpose of this diploma thesis, we used the following tools:

- Ubuntu 16.04
- OMNET++ 5.1.1
- INET 3.6
- Sumo v0.30.0
- Veins 4.6
- simuLTE

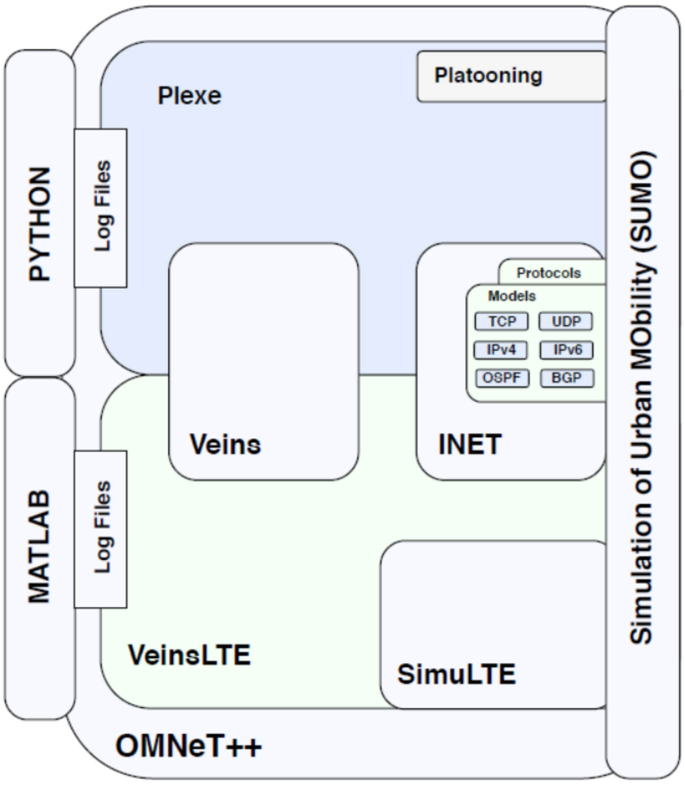


Figure 19 - Relation among Plexe and VeinsLTE. Two independent frameworks Relation among Plexe and VeinsLTE. Two independent frameworks VeinsLTE and Plexe that both use Veins and INET frameworks. The dierence is that Plexe contains Platooning features while VeinsLTE contains the stack of LTE.

5.1 OMNET++ 5.1.1

OMNeT++ is considered to be a discrete network simulator platform, based and written on C++ libraries, with free licensed software for academic use. It provides the user with all the necessary tools and libraries for creating and performing simulations. Modules and ideas can be reused for extending or creating new modules. Modules of OMNeT++ are connected via gates. This simulator platform is available for different operating systems like Linux, Mac OS, and Windows. The GUI offers a great feature especially if the user is interested in debugging or investigating what is happening

behind the scenes. In addition, OMNeT++ with its own tool has the capability to analyze the data, which provides the user with a rich environment for performing various analyses and investigations. [20]

We chose OMNeT++ for several reasons:

- It is a stable, mature and with many features' framework.
- It is modular, which means that one can write all the required code from scratch and extend it.
- It already contains a large amount of simulation models (e.g. INET), all the TCP/IP stack, mobility, wireless technologies etc. This allows users to simulate mixed scenarios, of which LTE/LTE-A constitutes only one part, and get a feeling of the true end-to-end performance of simulated applications.
- It is supported by a large and active community of users, from both academia and networking industries.

5.2 INET 3.6

INET is an open source OMNeT++ model suite for wired, wireless and mobile networks. It contains models for the Internet stack (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.), wired and wireless link layer protocols (Ethernet, PPP, IEEE 802.11, etc), support for mobility, MANET protocols, DiffServ, and MPLS with LDP and RSVP-TE signaling. Vehicular networks, overlay/peer-to-peer networks, or LTE take INET as a base, and extend it into specific directions. [29]

INET is built around the concept of modules that communicate by message passing. Agents and network protocols are represented by components, which can be freely combined to form hosts, routers, switches, and other networking devices. New components can be programmed by the user as well.

INET benefits from the infrastructure provided by OMNeT++. It’s not only making use of the services provided by the OMNeT++ simulation kernel and library (component model, parameterization, result recording, etc.), but also it allows the models to be developed, assembled, parameterized, run, and their results to be evaluated from the OMNeT++ Simulation IDE, or from the command line.

Some of its helpful features are:

- OSI layers implemented (physical, link-layer, network, transport, application)
- Pluggable protocol implementations for various layers
- IPv4/IPv6 network stack (or user’s own network layer)
- Transport layer protocols: TCP, UDP, SCTP
- Routing protocols (ad-hoc and wired)
- Wired/wireless interfaces (Ethernet, PPP, IEEE 802.11, etc.)
- Physical layer with scalable level of detail (unit disc radio to detailed propagation models, frame level to bit/symbol level representation, etc.)
- Wide range of application models
- Network emulation support
- Mobility support
- Supports the modeling of the physical environment (obstacles for radio propagation, etc.)

For this diploma thesis scenarios, we used the 3.6 version which is more stable and complies with OMNeT++ 5.1.

5.3 Veins 4.6

Veins is a model library for OMNeT++, which supports simulations involving communicating road vehicles. It is distributed as an open-source software; that means, it is free to download, adapt, and use. It is based on two well-established simulators: OMNeT++, an event-based network simulator, and SUMO, a road traffic simulator. Veins has become well-established in the domain of Vehicular Ad Hoc Networks (VANETs) and Intelligent Transportation System (ITS). It is employed by both academia and industry around the globe. It serves as the basis of hundreds of publications and contributed to the standardization process of Inter-Vehicle Communication (IVC). One of its great advantages is that it can include third-party models directly or via the INET Framework, e.g., models for ETSI ITS-G5, 4G/5G cellular networking like 3gpp LTE and C-V2X.

5.4 Sumo 0.30

Sumo is an open source, highly portable, microscopic and continuous multi-modal traffic simulation package designed to handle large networks supported by Eclipse. SUMO can simulate medium to large road networks of cities, urban areas, highways, and freeways. It can simulate the movement of road vehicles like cars and trucks, of scooters and bicycles, of pedestrians, and of trains. SUMO supports a wide range of different mobility models (from idealized, lanediscrete models to sub-lane models of mixed car/scooter traffic), a set of different intersection controllers (from simple right of way to demand-actuated traffic lights), and a wide range of road network input formats (from OpenStreetMap and TIGER to proprietary, specialist Geographic Information System (GIS) formats) [24].

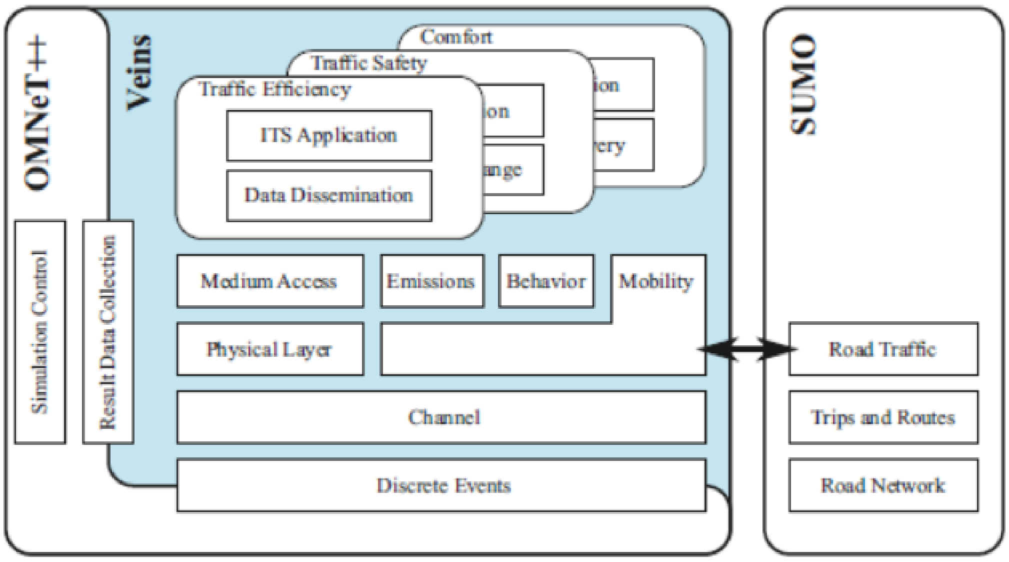


Figure 20 - High-level architecture of Veins

By default, mobility information is polled from SUMO at fixed intervals of, e.g., 100 ms, though adaptive polling is equally well supported by the interface. The execution of the OMNeT++ simulation pauses while SUMO computes any mobility information for the desired point in time. However, the performance impact of this is minimal because SUMO is designed to simulate at least an order of magnitude more mobile nodes than can be afforded in a highly detailed wireless network simulation. As a consequence, in a reasonably complex wireless network simulation, only fractions of percent of simulation time are spent calculating and communicating mobility information.

Whenever SUMO simulates the departure of a mobile node, Veins creates a dedicated simulation module in OMNeT++. Then, as the mobile node moves in SUMO, Veins keeps the corresponding OMNeT++ module updated with its position, heading, and speed (along with some more data). Similarly, when SUMO simulates the mobile node arriving at its destination, Veins removes the corresponding OMNeT++ module from the simulation. This way, Veins couples node mobility in OMNeT++ to that in SUMO.

This coupling is bi-directional: in addition to the OMNeT++ simulation evolving as dictated by the SUMO simulation, the OMNeT++ simulation can influence the simulated road traffic in SUMO, for example, to have cars choose a different route to their destination in response to received traffic information or to have a car perform an emergency brake in response to received warnings.

6 Setting up the Simulation Environment

6.1 Integrating Veins and LTE for heterogeneous networks

In order to simulate scenarios in heterogeneous networks, meaning with vehicles using both communication protocols, 802.11p and LTE, one needs Veins LTE, a simulator for heterogeneous vehicular networks. Another way around is to combine some components together, which will be described in this chapter.

6.1.1 Installing Veins LTE 1.3

The advantage of Veins LTE is that an application can decide to send a message packet via a specific network stack (in case of LTE or DSRC, it sends the packet via the associated technology) or let a dedicated module named *Decision Maker* decide which network technology is better suited for the current transmission even when the decision is made randomly. The source code of the DesicionMaker can be found in [36].

For this environment to be set up, it's needed:

- Ubuntu 14.04
- Veins LTE 1.3
- OMNTeT++ 4.6
- Sumo 0.25

We navigate to http://veins-lte.car2x.org/ and we download the zip file.

In /home, we need to create a master where we will have all the folders for our project. In our case, let's say "Veins_lte_project". We extract the zip file there. We also save in our main folder the extracted omnetpp-4.6 and the sumo-0.25.

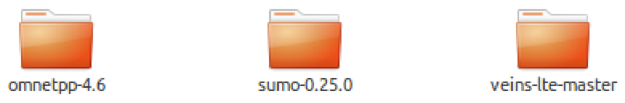


Figure 21 - Structure of main folder

First, we need to install OMNeT. We navigate to: https://doc.omnetpp.org/omnetpp4/InstallGuide.pdf, more specifically to page 15.

Using the command line, we navigate to omnetpp-4.6 folder and we type:

```
Make makefiles
```

Then,

```
make
```

In case we get an error about opp_makemake, we follow the instructions printed in command line to create the write path.

To install SUMO, we just need to type in the command line

```
./configure
```

```
Make
```

In the root folder.

To make sure that sumo runs, we use terminal and we navigate from our main folder (in our case “Veins_lte_project”) to folder “sumo-0.25.0”, then to “bin” and we run the command:

```
Sumo-gui
```

We open the OMNeT++ IDE. In the pop-up window, we need to select our main project folder:

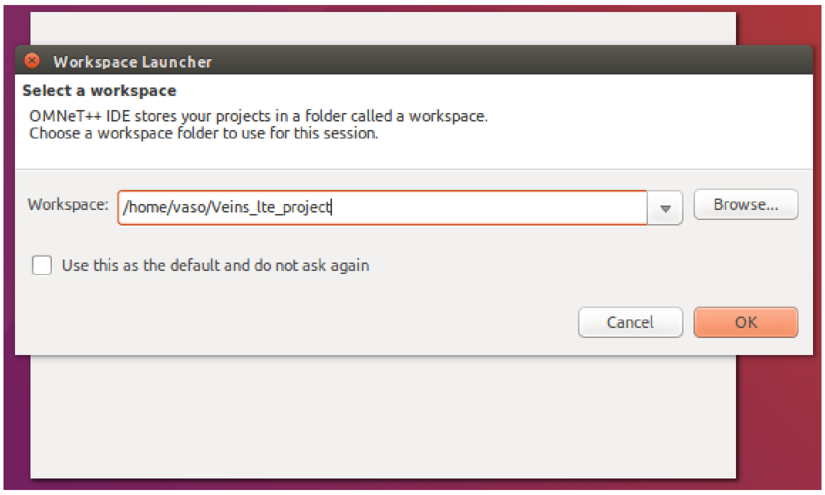

Workspace Launcher
Select a workspace
OMNeT++ IDE stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.
Workspace: /home/vaso/Veins_lte_project
Browse...
Use this as the default and do not ask again
Cancel
OK

Figure 22 - Workspace Lancher

Then we need to import our project as follows:

File → Import →

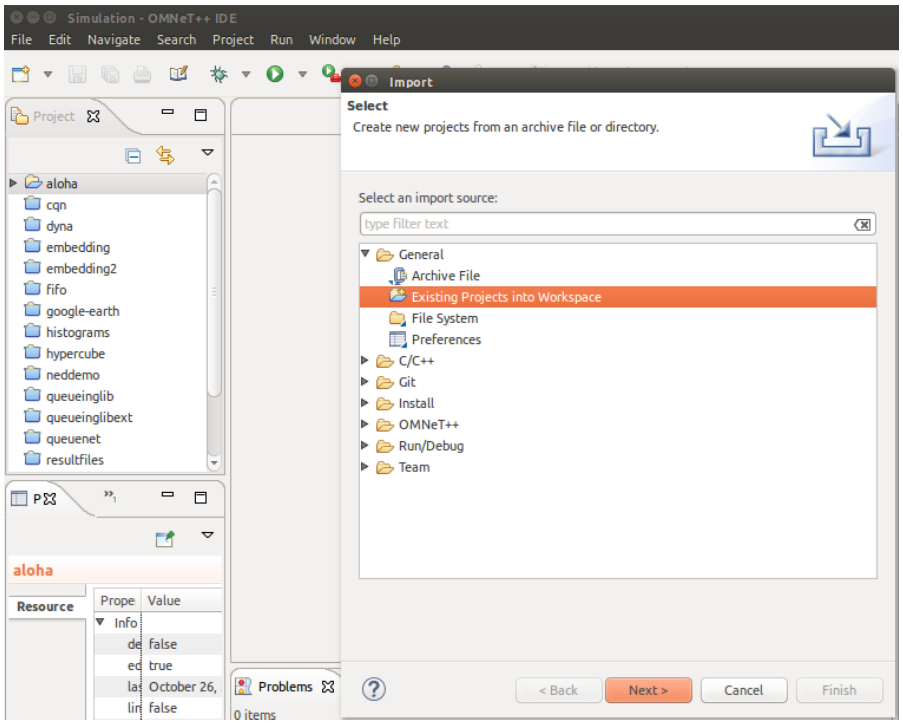
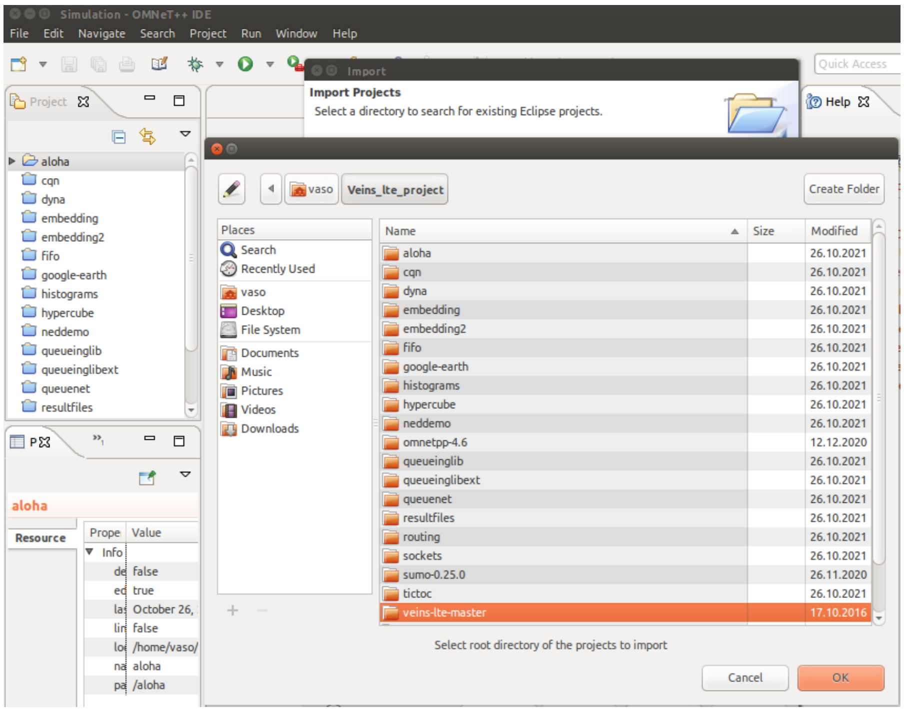


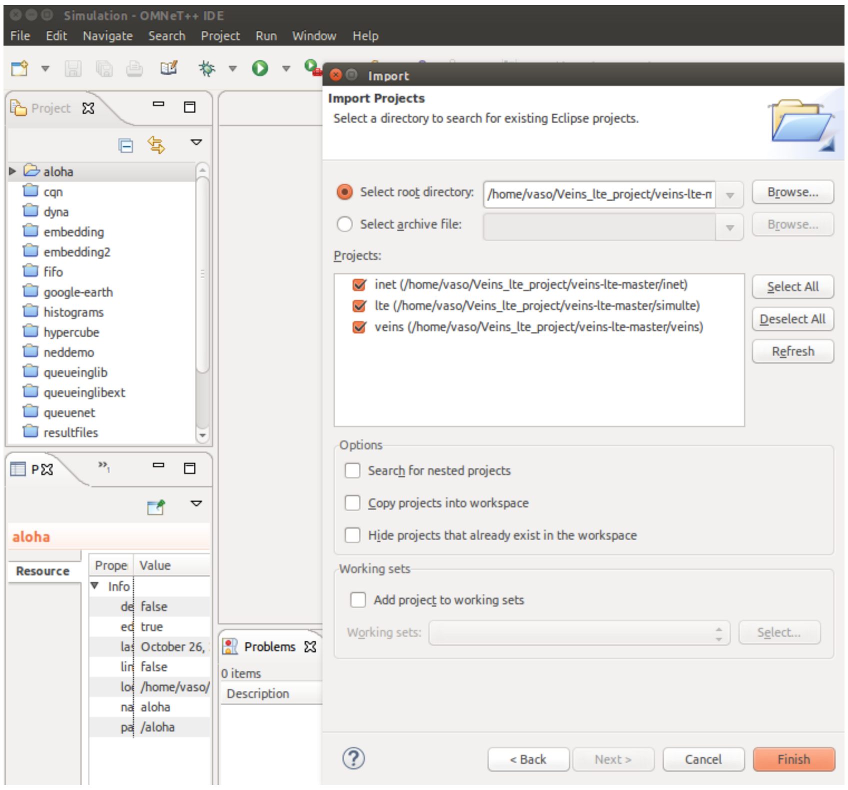
Figure 23 - Importing the existing projects



Name	Size	Modified
aloha		26.10.2021
cqn		26.10.2021
dyna		26.10.2021
embedding		26.10.2021
embedding2		26.10.2021
fifo		26.10.2021
google-earth		26.10.2021
histograms		26.10.2021
hypercube		26.10.2021
neddemo		26.10.2021
omnetpp-4.6		12.12.2020
queueinglib		26.10.2021
queueinglibext		26.10.2021
queuenet		26.10.2021
resultfiles		26.10.2021
routing		26.10.2021
sockets		26.10.2021
sumo-0.25.0		26.11.2020
tictoc		26.10.2021
veins-lte-master		17.10.2016

Select root directory of the projects to import

Figure 24 - Selecting our main project folder



Import Projects
Select a directory to search for existing Eclipse projects.

[✓ Select root directory: /home/vaso/Veins_lte_project/veins-lte-m]
Select archive file:

Projects:
- [✓] inet (/home/vaso/Veins_lte_project/veins-lte-master/inet)
- [✓] lte (/home/vaso/Veins_lte_project/veins-lte-master/simulte)
- [✓] veins (/home/vaso/Veins_lte_project/veins-lte-master/veins)

Options
- [ ] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

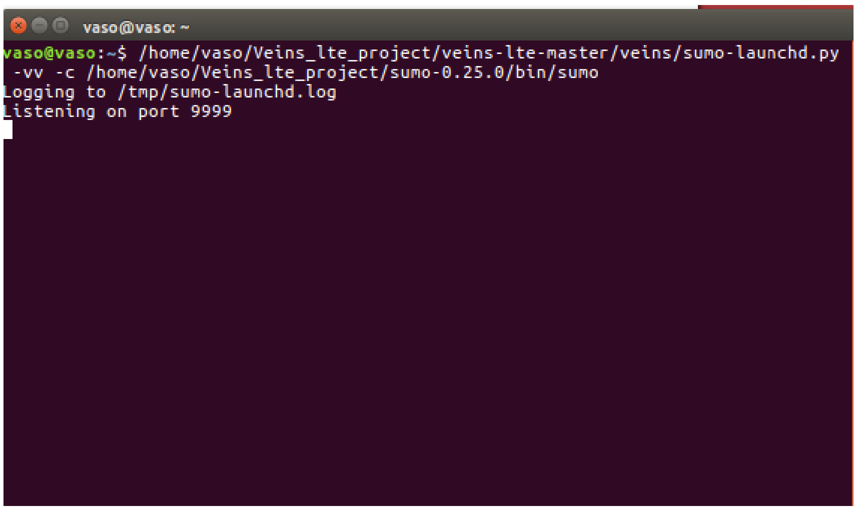
Working sets
- [ ] Add project to working sets

Figure 25 - Selecting which folders to import

Before we run the simulation, we need Sumo to run in parallel.

So, in the command line we type:

```
/home/vaso/Veins_lte_project/veins-lte-master/veins/sumo-launchd.py -vv -c
/home/vaso/Veins_lte_project/sumo-0.25.0/bin/sumo
```



```
vaso@vaso:~$ /home/vaso/Veins_lte_project/veins-lte-master/veins/sumo-launchd.py
 -vv -c /home/vaso/Veins_lte_project/sumo-0.25.0/bin/sumo
Logging to /tmp/sumo-launchd.log
Listening on port 9999
```

Figure 26 - command line

We navigate in the left side bar in the IDE, to the omnetpp.ini file for the heterogenous simulation

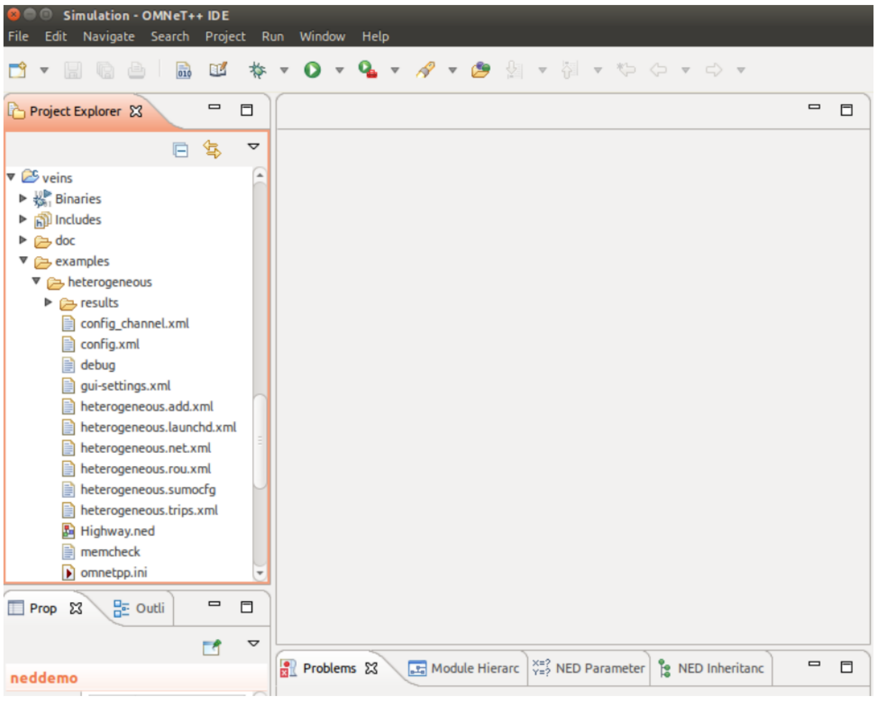


Figure 27 - omnetpp.ini

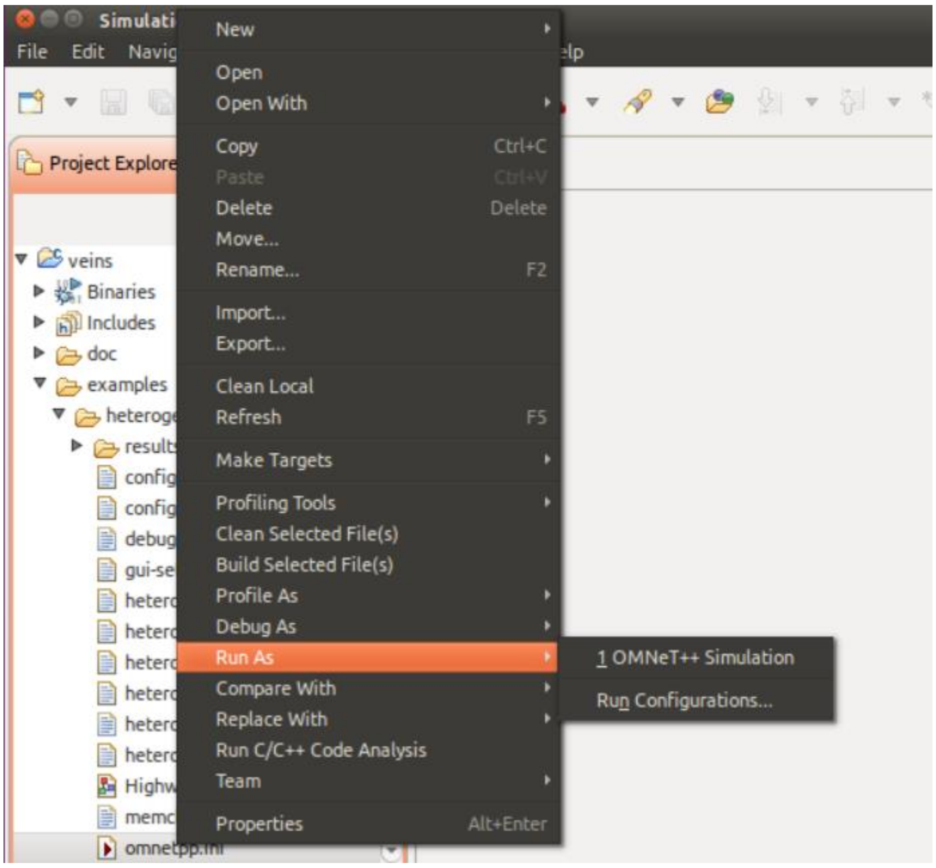
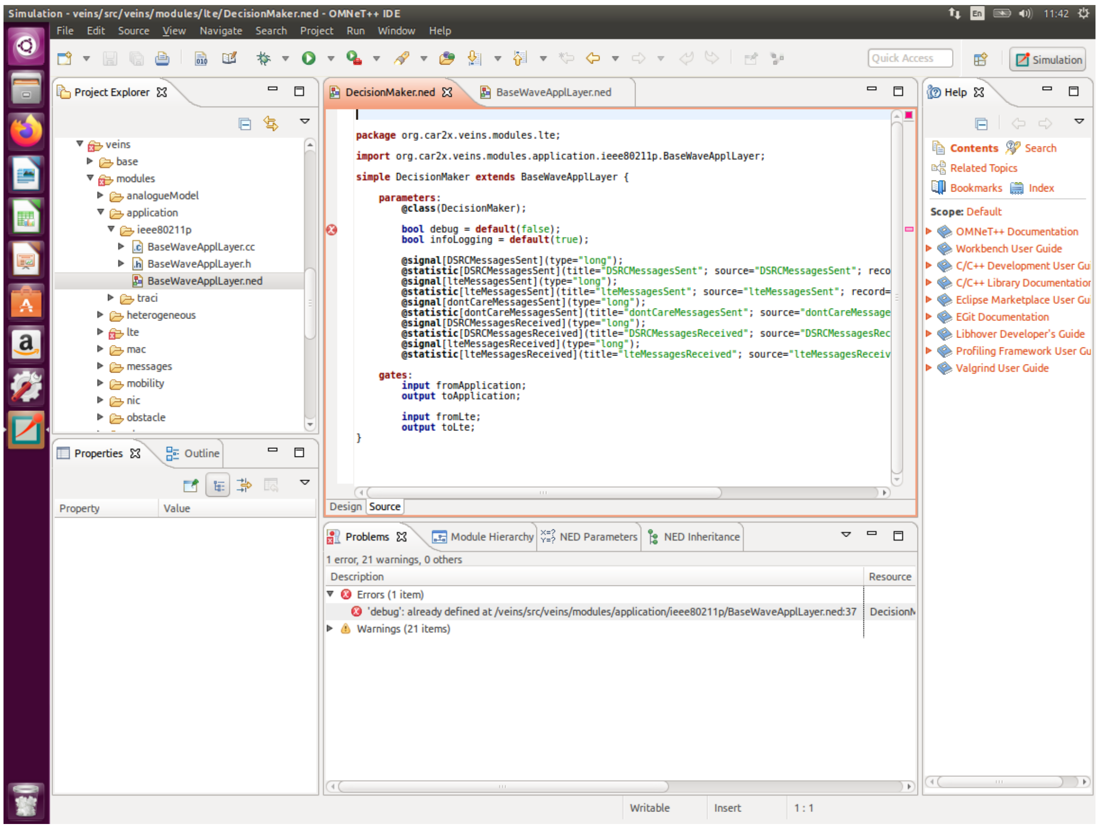


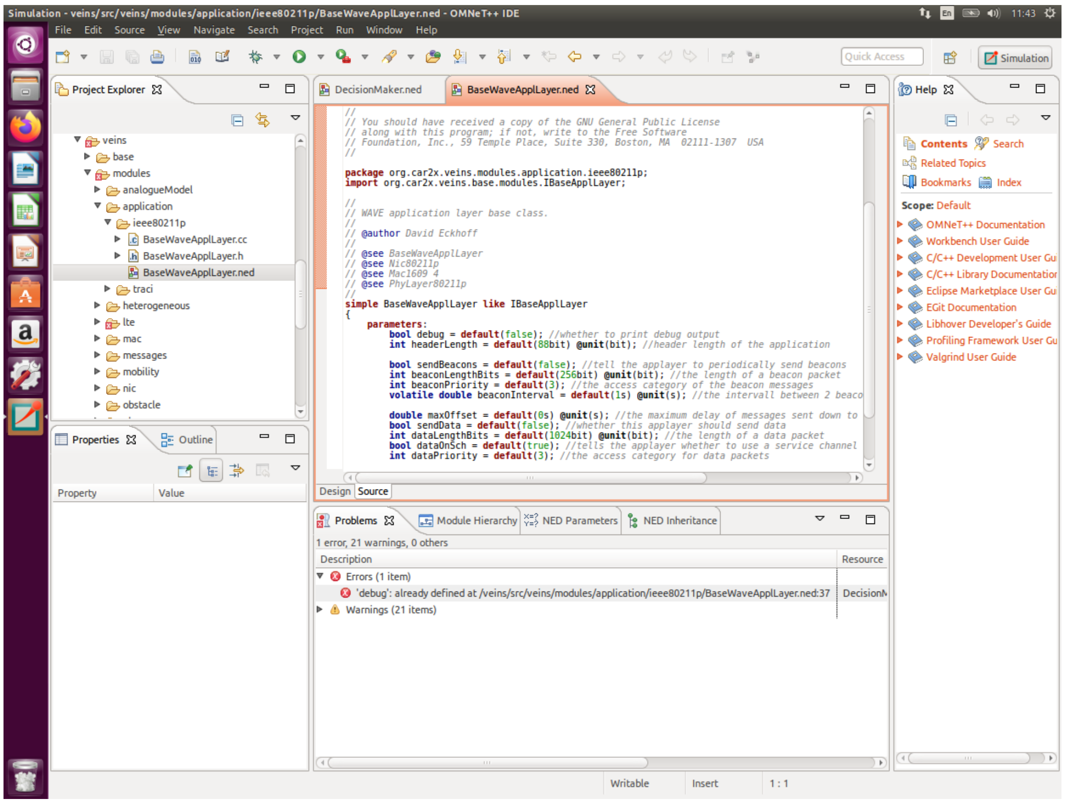
Figure 28 - Run the simulation


Simulation - veins/src/veins/modules/lte/DecisionMaker.ned - OMNeT++ IDE
DecisionMaker.ned
BaseWaveApplLayer.ned
Project Explorer
Help

```
package org.car2x.veins.modules.lte;

import org.car2x.veins.modules.application.ieee80211p.BaseWaveApplLayer;

simple DecisionMaker extends BaseWaveApplLayer {

    parameters:
        @class(DecisionMaker);

        bool debug = default(false);
        bool infoLogging = default(true);

        @signal[DSRCMessagesSent](type="long");
        @statistic[DSRCMessagesSent](title="DSRCMessagesSent"; source="DSRCMessagesSent"; reco
        @signal[lteMessagesSent](type="long");
        @statistic[lteMessagesSent](title="lteMessagesSent"; source="lteMessagesSent"; record=
        @signal[dontCareMessagesSent](type="long");
        @statistic[dontCareMessagesSent](title="dontCareMessagesSent"; source="dontCareMessage
        @signal[DSRCMessagesReceived](type="long");
        @statistic[DSRCMessagesReceived](title="DSRCMessagesReceived"; source="DSRCMessagesRec
        @signal[lteMessagesReceived](type="long");
        @statistic[lteMessagesReceived](title="lteMessagesReceived"; source="lteMessagesReceiv

    gates:
        input fromApplication;
        output toApplication;

        input fromLte;
        output toLte;
}
```

```
1 error, 21 warnings, 0 others
Description                                                                         Resource
Errors (1 item)
  'debug': already defined at /veins/src/veins/modules/application/ieee80211p/BaseWaveApplLayer.ned:37   DecisionM
Warnings (21 items)
```

Figure 29 - decisionMaker.ned



```
//
// You should have received a copy of the GNU General Public License
// along with this program; if not, write to the Free Software
// Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
//

package org.car2x.veins.modules.application.ieee80211p;
import org.car2x.veins.base.modules.IBaseApplLayer;

//
// WAVE application layer base class.
//
// @author David Eckhoff
//
// @see BaseWaveApplLayer
// @see Nic80211p
// @see Mac1609 4
// @see PhyLayer80211p
//
simple BaseWaveApplLayer like IBaseApplLayer
{
    parameters:
        bool debug = default(false); //whether to print debug output
        int headerLength = default(88bit) @unit(bit); //header length of the application

        bool sendBeacons = default(false); //tell the applayer to periodically send beacons
        int beaconLengthBits = default(256bit) @unit(bit); //the length of a beacon packet
        int beaconPriority = default(3); //the access category of the beacon messages
        volatile double beaconInterval = default(1s) @unit(s); //the intervall between 2 beaco

        double maxOffset = default(0s) @unit(s); //the maximum delay of messages sent down to
        bool sendData = default(false); //whether this applayer should send data
        int dataLengthBits = default(1024bit) @unit(bit); //the length of a data packet
        bool dataOnSch = default(true); //tells the applayer whether to use a service channel
        int dataPriority = default(3); //the access category for data packets
```

```
1 error, 21 warnings, 0 others
Description                                                                                   Resource
▼ Errors (1 item)
    'debug': already defined at /veins/src/veins/modules/application/ieee80211p/BaseWaveApplLayer.ned:37   DecisionM
▶ Warnings (21 items)
```

Figure 30 - BaseWaveAppLayer.ned

Either running the simulation for the IDE or from the command line, the debug error remains.

So, it is not possible to run the heterogeneous simulation from VeinsLTE. [30]

6.1.2 OpenCV2X with Veins

The second option to integrate Veins with LTE is to use the OpenCV2X implementation of UCC. [31]

We set up our environment again in Ubuntu 16.04.

Then we need to install SUMO 1.2.0, which we can find in [32]. This is the file we will need: “sumo-all-1.2.0.tar.gz”.

We follow the instructions in the *Read me* file to install SUMO in the main folder of SUMO.

Next step is to install OMNeT++ 5.6.2 according to [33]

When we are about to type the configure command in the main folder of OMNeT, we do as follows:

```
./configure WITH_OSGEARTH = no WITH-OSG = no
```

During configure, we get an error:

“Configure:error : No C++ complier found – one is required to build OMNET++/OMNSET, and also for model development”

In terminal, we type:

```
Gcc –version
```

Or

```
Gcc -v
```

To find out the current version of our system.

It is important to install GNU GCC 7.3. We will download this dependency from Jonathon F’s ppa because Ubuntu ppa has only the 7.2 version.

```
Sudoapt-get install ppa-purge
Sudo add-apt-repository -y .ppa:jonathonf/gcc
Sudo-apt-get update
Sudo apt-get install gcc-7
```

After

```
Gcc –version
```

The version of GCC isn’t updated to 7.3. We can try the following:

```
Sudo add-apt-repository ppa:jonathonf/gcc
Sudo apt-get update
Sudo apt-get install gcc-7
Sudo apt update
Sudo apt install build-essential
```

Again, the GCC version isn’t updated to 7.3. We will try to install the 7.5 version.

```
Sudo apt-get install -y software-properties-common
Sudo apt-repository ppa:ubuntu-toolchain-r/test

Sudo apt update
Sudo apt install g++-7 -y

Sudo update-alternatives –install /usr/bin/gcc gcc/usr/bin/gcc-7 60 \ --slave
/usr/bin/g++ g++ /usr/bin/g++-7
Sudo update-alternatives –config gcc
```

We try to do the ./configure command in the source folder of OMNeT but the same error remains, that no gcc compiler was found.

6.1.3 SimuLTE

The last approach to install Veins with LTE is the following:

We navigate to [34]. We download the tar.gz file ("*Download the complete package*"). Then, in the main folder of our project, let's say "VeinsLTE", we need to install OMNTeT++ 5.1.1. We follow the instructions in [33]. In paragraph "4.4 Environmental Variables", when we are about to edit .bashrc,

```
$ gedit ~/.bashrc
```

At the end of the file, we add the following line and then we save.

```
Export PATH=$PATH:/home/vaso/VeinsLTE/omnetpp-5.1.1/bin
```

Instead of "vaso", one can use it's one user name.

We complete the installation with:

```
./configure
Make -j8   //To make use of multiple processor cores
```

Before, installing SUMO 0.30, we need to install some libraries first.

```
$ sudo apt-grt install cmake python g++ liberces-c-dev libfox-1.6-dev libgdal-dev lib-
proj-dev libgl2ps-dev
```

```
$sudo apt-getinstalllibavformat-dev libswscale-dev libopenscenegraph-dev python3-dev
swig libgtest-dev libeigen3-dev python3-pip python3-setuptools default-jll
```

```
$sudo pip3 install texttest
```

Then, in the .bashrc file, we add in the last line:

```
Export SUMO_HOME = "/home/vaso/VeinsLTE/sumo-0.30"
```

And save.

In the terminal, we navigate to the sumo-0.30 folder, and we need to declare the following variable:

```
Echo $SUMO_HOME
```

Finally, in sumo main folder, we do:

```
./confige
Make
```

To make sure that SUMO is running properly, we open the terminal and type:

```
/home/vaso/VeinsLTE/simulte_veins/veins-veins-4.6/sumo-launchd.py -vv -c sumo-gui
```

We start the OMNeT IDE. We navigate from File → Import →

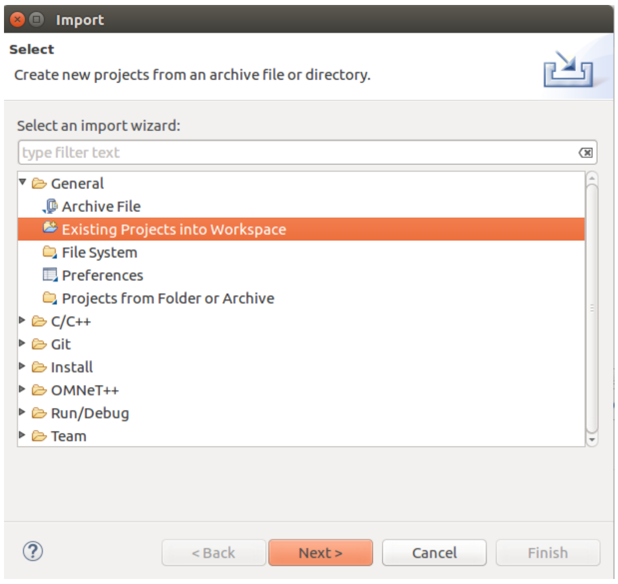
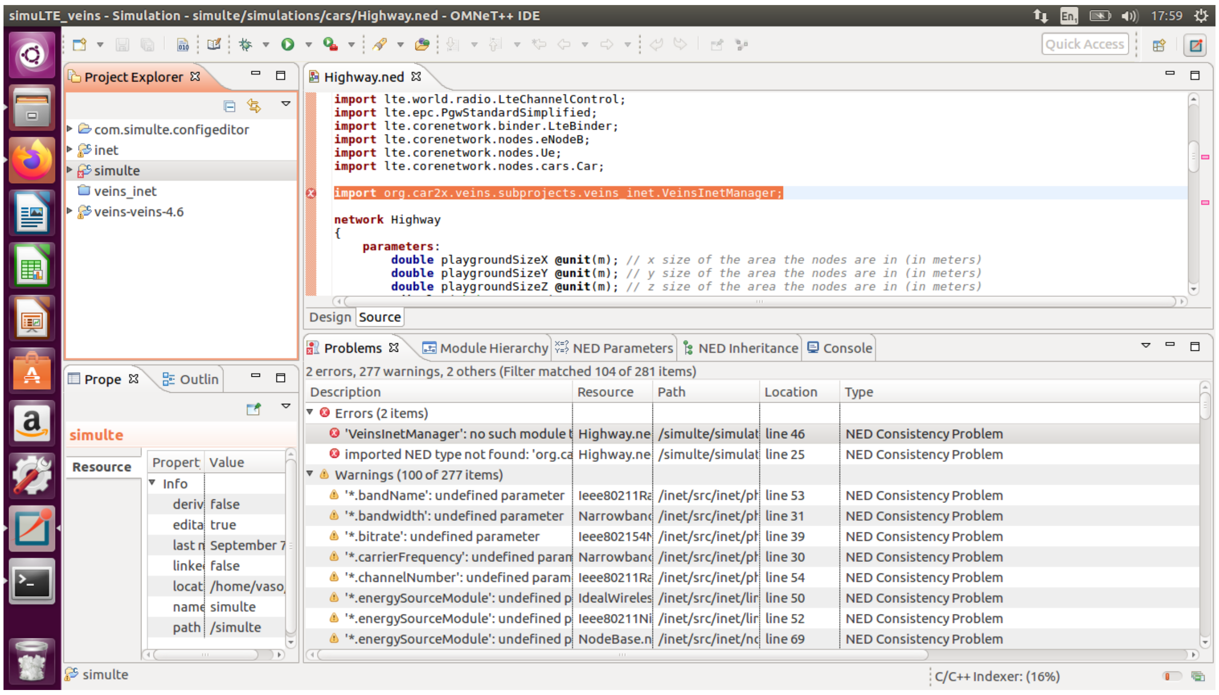


Figure 31 - Import Existing Projects into Workspace

And then, we select the folder *simulte_veins*.

Last step, is *Project → Import*

It is possible that we get the following error:



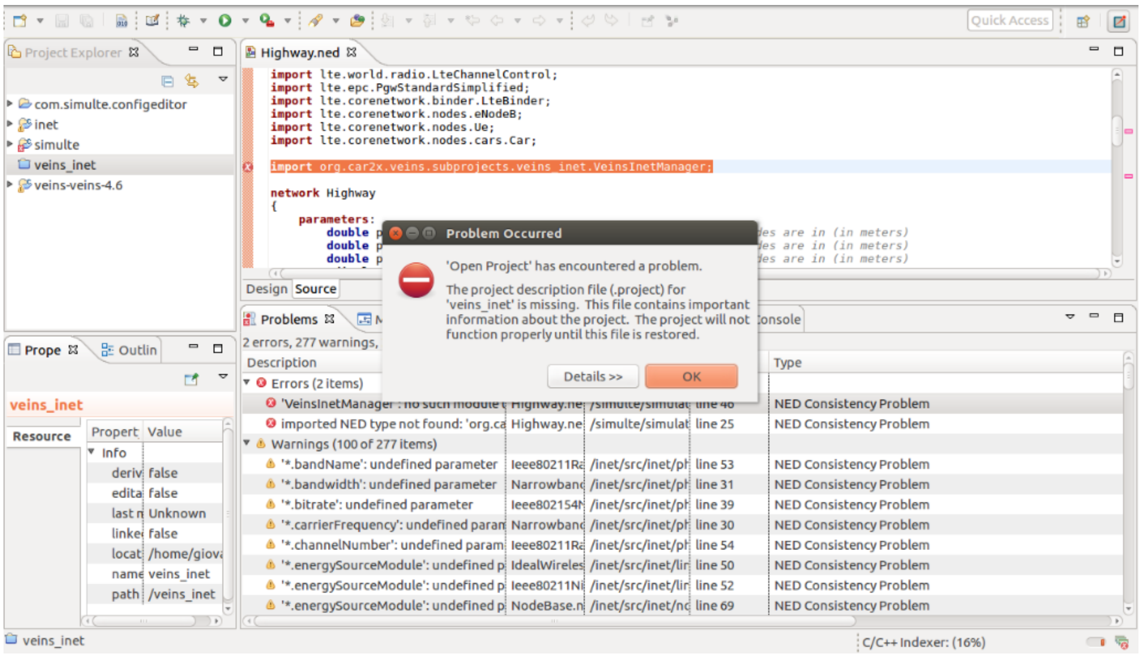
simuLTE_veins - Simulation - simulte/simulations/cars/Highway.ned - OMNeT++ IDE

```
import lte.world.radio.LteChannelControl;
import lte.epc.PgwStandardSimplified;
import lte.corenetwork.binder.LteBinder;
import lte.corenetwork.nodes.eNodeB;
import lte.corenetwork.nodes.Ue;
import lte.corenetwork.nodes.cars.Car;

import org.car2x.veins.subprojects.veins_inet.VeinsInetManager;

network Highway
{
    parameters:
        double playgroundSizeX @unit(m); // x size of the area the nodes are in (in meters)
        double playgroundSizeY @unit(m); // y size of the area the nodes are in (in meters)
        double playgroundSizeZ @unit(m); // z size of the area the nodes are in (in meters)
```

2 errors, 277 warnings, 2 others (Filter matched 104 of 281 items)

Description	Resource	Path	Location	Type
Errors (2 items)				
'VeinsInetManager': no such module t	Highway.ne	/simulte/simulat	line 46	NED Consistency Problem
imported NED type not found: 'org.ca	Highway.ne	/simulte/simulat	line 25	NED Consistency Problem
Warnings (100 of 277 items)				
'*.bandName': undefined parameter	Ieee80211Ra	/inet/src/inet/ph	line 53	NED Consistency Problem
'*.bandwidth': undefined parameter	Narrowband	/inet/src/inet/ph	line 31	NED Consistency Problem
'*.bitrate': undefined parameter	Ieee802154N	/inet/src/inet/ph	line 39	NED Consistency Problem
'*.carrierFrequency': undefined param	Narrowband	/inet/src/inet/ph	line 30	NED Consistency Problem
'*.channelNumber': undefined param	Ieee80211Ra	/inet/src/inet/ph	line 54	NED Consistency Problem
'*.energySourceModule': undefined p	IdealWireles	/inet/src/inet/lir	line 50	NED Consistency Problem
'*.energySourceModule': undefined p	Ieee80211Ni	/inet/src/inet/lir	line 52	NED Consistency Problem
'*.energySourceModule': undefined p	NodeBase.n	/inet/src/inet/nc	line 69	NED Consistency Problem

Figure 32 - No such module error



Problem Occurred

'Open Project' has encountered a problem.

The project description file (.project) for 'veins_inet' is missing. This file contains important information about the project. The project will not function properly until this file is restored.

Figure 33 - Problem occurred

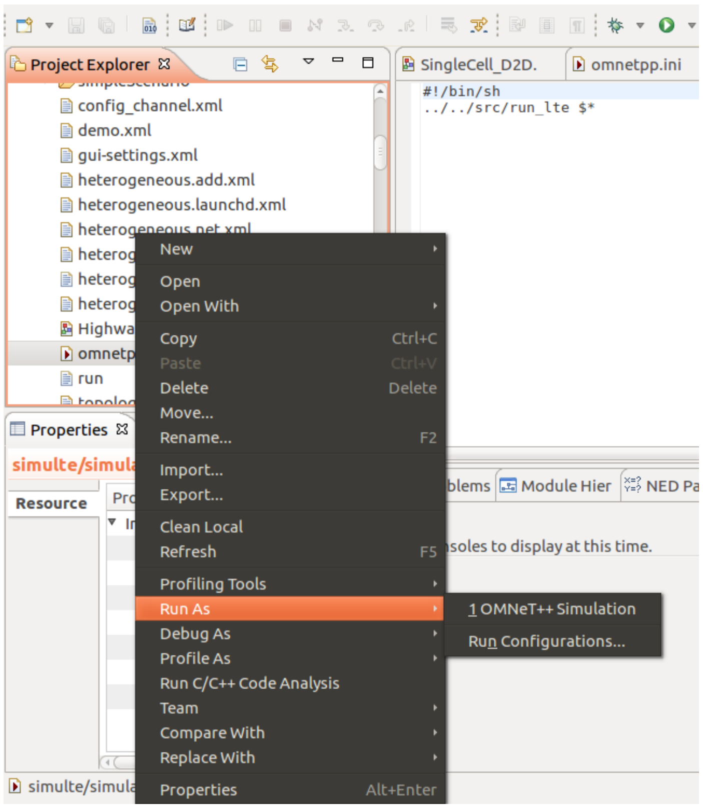
To solve this problem, we need to delete the folder *veins_inet* manually from our work space and import it again.

Before we run the heterogenous simulation, we need to type in the terminal:

```
/home/vaso/VeinsLTE/simulte_veins/veins-veins-4.6/sumo-launchd.py -vv -c
/home/vaso/VeinsLTE/sumo-0.30.0/bin/sumo
```

By this command, we manage to run two simulations in parallel. A sunclass of TraCLScenarioManager called TraCLScenarioManagerLaunch is instantiated and sumo-launchd.py waits for incoming network connections from an OMNeT++ simulation and launches one instance of SUMO for each simulation and proxies connection.

In the project explorer of OMNeT ++, we find the folder simulte → simulations → cars. We run the omnet.ini file as OMNeT simulation.



```
#!/bin/sh
../../src/run_lte $*
```

Figure 34 - Run omnet.ini as OMNeT simulation

6.1.4 Integrating Veins_hetvet with Instant Veins 5.1-i2

Since VeinsLTE is not supported anymore and the final product returns a debug, we could combine Veins, LTE and INET using the veins_hetvnet simulation [35]. We need to install Instant Veins 5.1-i2 which is a virtual machine and contains these components pre-installed:

- Simulation modules
  - Veins 5.1
  - INET Framework 4.2.2
  - SimuLTE 1.2.0 (plus a backported patch, 23c0936e31)
  - Veins_INET included with Veins 5.1
- Software
  - OMNeT++ 5.6.2
  - SUMO 1.8.0
  - Cookiecutter 1.6.0 for cookiecutter-veins-project
- Operating system
  - Debian 10, Linux 4, GNOME 3

‘veins_hetvnet` uses all wireless networking simulation modules that are instantiated from the INET and SimuLTE module libraries. Because both wireless interface modules (one modeling an LTE modem, one modeling a WLAN card) are INET modules, choosing the output interface is as simple as choosing which INET socket to use.

```
void HetVNetDemoApp::sendHetVNetDemoPacket()
{
    {
        HetVNetDemoPacket* packet = new HetVNetDemoPacket("Lte Demo Packet");
        packet->setIsWlan(0);
        packet->setSequenceNo(nextSequenceNumber);
        packet->setCreationTime(simTime());
        packet->setByteLength(packetSizeBytes);
        EV_TRACE << "Sending message [" << nextSequenceNumber << "] via LTE!" <<
std::endl;

        socketLte.sendTo(packet, destAddressLte_, localPortLte);

        emit(sigDemoPktSent, (long) 1);
    }

    {
        HetVNetDemoPacket* packet = new HetVNetDemoPacket("Wlan Demo Packet");
        packet->setIsWlan(1);
        packet->setSequenceNo(nextSequenceNumber);
        packet->setCreationTime(simTime());
        packet->setByteLength(packetSizeBytes);
        EV_TRACE << "Sending message [" << nextSequenceNumber << "] via WLAN" <<
std::endl;

        socketWlan.sendTo(packet, destAddressWlan, localPortWlan);

        emit(sigDemoPktSent, (long) 1);
    }

    nextSequenceNumber++;

    scheduleAt(simTime() + packetInterval, sendPacket);
}
```

And this is what it's implemented. This application sends a packet first using one socket (i.e., interface), then using the other one.

Before we run a simulation in OMNeT++ IDE, we need to type this command in the terminal:

```
/home/veins/src/veins/sumo-launchd.py --v -c sumo-gui
```

6.2 Simulation Scenario

In the first simulation scenario, we set the number of vehicles to 15, so that we observe if the packet loss increases when we increase the number of the vehicles.

The vehicles must be declared in heterogeneous.rou.xml.

```
<routes>
  <!--Namespace(N=75, filecount=None, input='routes.txt', launch=False, output='sin-
gleIntersection.rou.xml', quarter=100)-->
  <vType id="car" accel="0.8" decel="4.5" sigma="1.0" length="4" minGap="2.5"
maxSpeed="13.89" guiShape="passenger" />
  <route id="1" edges="1/0to1/1 1/1to1/2"/>
  <route id="2" edges="1/2to1/1 1/1to1/0"/>
  <route id="3" edges="2/1to1/1 1/1to0/1"/>
  <route id="4" edges="0/1to1/1 1/1to2/1"/>
  <vehicle id="0" type="car" route="2" depart="0"/>
  <vehicle id="1" type="car" route="3" depart="0"/>
  <vehicle id="2" type="car" route="2" depart="2"/>
  <vehicle id="3" type="car" route="3" depart="3"/>
  <vehicle id="4" type="car" route="2" depart="5"/>
  <vehicle id="5" type="car" route="3" depart="5"/>
  <vehicle id="7" type="car" route="3" depart="7"/>
  <vehicle id="6" type="car" route="2" depart="10"/>
        <vehicle id="8" type="car" route="2" depart="12"/>
        <vehicle id="9" type="car" route="3" depart="14"/>
        <vehicle id="10" type="car" route="2" depart="16"/>
        <vehicle id="11" type="car" route="3" depart="17"/>
        <vehicle id="12" type="car" route="1" depart="19"/>
        <vehicle id="13" type="car" route="3" depart="21"/>
        <vehicle id="14" type="car" route="4" depart="23"/>
      </routes>
```

The parameters of the simulation can be found in Appendix A. Also, when we run the simulation, we select the option D2D-Multicast which means that each vehicle can send and receive in the same time and also to everyone.

Also, it is important to mention that, in line:

```
*.veinsManager.launchConfig = xmldoc("heterogeneous.launchd.xml")
```

We declare the configuration that SUMO will “read” in order to run.

In the following line:

```
**.deployer.positionUpdateInterval = 0.001s
```

We specify the time in which we expect to receive the velocity and the position of each node. The Lte network elements use a dynamic deployer. There is one deplayer for each eNB (thus one for each cell).

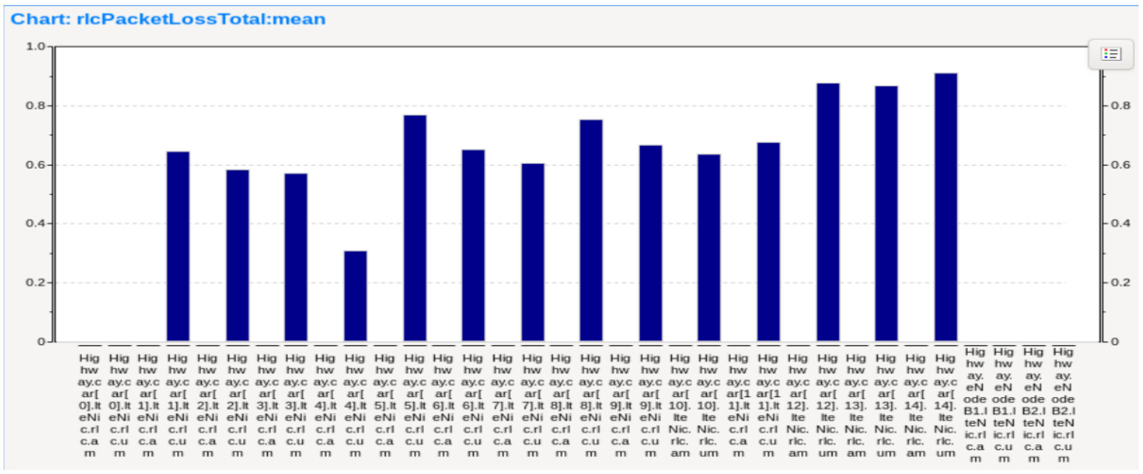


Figure 35 - Packet Loss of each vehicle or eNodeB per time

7 Conclusion

This thesis described the current situation in automated vehicles (in academic available sources) in regards of communication protocols. Technically, it was described how to set up an environment in which the 2 technologies (LTE, 802.11p) can be used, so that when both combined (it is proven theoretically in literature), the security between the packages exchanged is higher and the packet loss lesser. It is important to note, though, that LTE offers higher network capacity and greater support mobility in relation to the IEEE 802.11p standard. Also, the use of heterogeneous networks of vehicles can prevent chain collisions in the roads.

There could be a direction in which this work could be extended. That is, how a communication standard would be selected based on one/more parameter/s of one of the layers. Not in the physical layer because the parameters in PHY are already optimized in this framework. Most likely in a layer above, the MAC layer.

References

[1] Kamel Mohamed et al, "Autonomous and Intelligent Systems", Third International Conference, AIS 2012, pg 34, Aveiro, Portugal, June 25–27, 2012

[2] Shashank Kumar Gupta, Jamil Yusuf Khan and Duy Trong Ngo, “An LTE-Direct-Based Communication System for Safety Services in Vehicular Networks”, May 26th 2020

[3] Hakim Badis, Abderrezak Rachedi, in Modeling and Simulation of Computer Networks and Systems, 2015

[4] Michael Lee, Travis Atkinson, VANET applications: Past, present, and future, Department of Computer Science, The University of Alabama, Tuscaloosa, AL, United States of America

[5] I. A. W. Group, “Ieee std 1471-2000, recommended practice for architectural description of software-intensive systems,” IEEE, Tech. Rep., 2000

[6] ISO/IEC/(IEEE), “ISO/IEC 42010 (IEEE Std) 1471-2000: Systems and Software engineering - Recomended practice for architectural description of software-intensive systems,” p. 23, 07 2007

[7] Yeferny Taoufik, Hamad Sofian, Vehicular Ad-hoc Networks: Architecture, Applications and Challenges, College of Science, Northern Border University, Arar, Saudi Arabia

[8] R. Baldessari, B. Bodekker, M. Deegener, A. Festag, W. Franz, C. C.¨ Kellum, T. Kosch, A. Kovacs, M. Lenardi, C. Menig et al., “Car-2-car communication consortium-manifesto,” 2007

[9] A. Awang, K. Husain, N. Kamel, and S. A¨ıssa, “Routing in vehicular ad-hoc networks: A survey on single-and cross-layer design techniques, and perspectives,” IEEE Access, vol. 5, pp. 9497–9517, 2017.

[10] S. R. Kolte and M. S. Madankar, “Adaptive congestion control for transmission of safety messages in VANET,” 2014 Int. Conf. Converg. Technol. I2CT 2014, pp. 1–5, 2014

[11] Saini Majyot, Singh Harjit, “VANET, its Characteristics, Attacks and Routing Techniques: A Survey”, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064

[12] [22] B. Bako and M. Weber, “Efficient Information Dissemination in VANETs,” in Adv. Veh. Netw. Technol., 2011, p. 20

[13] Abinaya. M ,Archana. M ,Arul Arasi. M, Bragadeeswari. K, “An Efficient Vehicular Ad-Hoc Network to Reduce The Loss of Packets”, International Journal of Engineering Research & Technology (IJERT), Special Issue - 2018

[14] Blaya Vilardebó Pedro Luis, “Investigation on vehicular ad hoc networks: 802.11p vs LTE architecture”, Bachelor Thesis, Faculty of Electrical Engineering and Information Technology of Rheinisch-Westfälischen Technischen Hochschule Aachen, Jan 2016

[15] A. S. Tanenbaum, Computer Networks, Prentice-Hall, 4th ed., 2002

[16] Sichitiu, Mihail; Kihl, Maria, “Inter-Vehicle Communication Systems: A Survey”, IEEE Communications Surveys and Tutorials, 2008

[17] Filippi A, Moerman K, Daalderop G, Alexander P, Schober F, Pfliegl W., “Ready to roll: Why 802.11p beatsLTE and 5G for V2x”, Awhite paper byNXPSemiconductors, Cohda Wireless, and Siemens

[18] Suad Kasapovic, Samra Mujacic, Lejla Banjanovic-Mehmedovic, Izet Jagodic, “Simulation of Heterogeneous Network Vehicles”, Journal of software, 2016

[19] Fernando A.Teixeira, Vinicius F.e Silva, Jesse L.Leoni, Daniel F.Macedo, José M.S.Nogueira, “Vehicular networks using the IEEE 802.11p standard: An experimental analysis”, Vehicular Communications, Volume 1, Issue 2, April 2014, Pages 91-96

[20] Haliti A., “Update and Evaluate Vehicular Simulation Framework for LTE and 802.11p in OMNeT++”, Master’s Thesis, Department of Electrical and Information Technology, Lund University, October 2018

[21] Laura Bernadó, Nicolai Czink, Thomas Zemen, and Pavle Belanovic. Physical layer simulation results for ieee 802.11 p using vehicular non-stationary channel model. In Communications Workshops (ICC), 2010 IEEE International Conference on, pages 1–5. IEEE, 2010.

[22] Bilstrup K., Uhlemann E., Strömand E., Bilstrup U., “Evaluation of the IEEE 802.11p MAC method for vehicle-to-vehicle communication” in: 68th IEEE Vehicular Technology Conference, 2008

[23] Virdis A., Stea G., Nardini G., “SimuLTE – A Modular System-level Simulator for LTE/LTE-A Networks based on OMNeT++”, Dipartimento di Ingegneria dell’Informazione, University of Pisa, Pisa, Italy

[24] Virdis A., Kirsche M., “Recent Advances in Network Simulation -The OMNeT++ Environment and its Ecosystem”, Springer, 2019

[25] H.Mousavi, Iraj S.Amiri, M.A.Mostafavi, C.Y.Choona, “LTE physical layer: Performance analysis and evaluation”, Applied Computing and Informatics, Volume 15, Issue 1, January 2019, Pages 34-44

[26] Masaracchia A., Busacca A., Mangione S., Passarella A., “Opportunistic traffic Offloadings Mechanisms for Mobile/4G Networks”, Universita degli Studi di Palermo, 2016

[27] Nardini G., Virdis A., Stea G., “Modeling Network-Controlled Device-to-Device Communications in SimuLTE”, October 2018

[28] Virdis A., Stea G., Nardini G., “SimuLTE – A Modular System-level Simulator for LTE/LTE-A Networks based on OMNET++

[29] https://inet.omnetpp.org/Introduction.html

[30] https://github.com/floxyz/veins-lte/issues/7

[31] http://www.cs.ucc.ie/cv2x/media/OpenCV2X_Documentation.pdf

[32] https://sourceforge.net/projects/sumo/files/sumo/

[33] https://doc.omnetpp.org/omnetpp4/InstallGuide.pdf

[34] https://simulte.com/add_veins.html

[35] https://github.com/veins/veins_hetvnet

[36] https://github.com/floxyz/veins-lte/blob/master/veins/src/veins/modules/lte/BasicDecisionMaker.cc

Appendix A

```
[General]
cmdenv-express-mode = true
cmdenv-autoflush = true

network = lte.simulations.scenario15vehicles.Highway

#########################################################
#            Simulation parameters                      #
#########################################################
debug-on-errors = false
print-undisposed = false

sim-time-limit = 80s

**.sctp.**.scalar-recording = true
**.sctp.**.vector-recording = true
**.cellBlockUtilization.result-recording-modes = default;

**.coreDebug = false
**.routingRecorder.enabled = false

*.playgroundSizeX = 20000m
*.playgroundSizeY = 20000m
*.playgroundSizeZ = 50m

#########################################################
#            VeinsManager parameters                    #
#########################################################
*.veinsManager.moduleType = "lte.corenetwork.nodes.cars.Car"
*.veinsManager.moduleName = "car"
*.veinsManager.moduleDisplayString = ""
*.veinsManager.launchConfig = xmldoc("heterogeneous.launchd.xml")
*.veinsManager.updateInterval = 0.1s

#########################################################
#                      Mobility                         #
#########################################################
*.car[*].mobilityType = "VeinsInetMobility"

**.deployer.positionUpdateInterval = 0.001s

#########################################################
#      IP configuration for dynamic-created UEs         #
#########################################################
# enrolled multicast groups must be set in the HostAutoConfigurator (instead of
demo.xml),
#seperated by a single space character
*.car[*].configurator.mcastGroups = "224.0.0.10"

#########################################################
#                 channel parameters                 #
#########################################################
**.channelControl.pMax = 10W
**.channelControl.alpha = 1.0
**.channelControl.carrierFrequency = 2100e+6Hz
```

```
# PhyLayer & Feedback parameters
**.lteNic.phy.channelModel=xmldoc("config_channel.xml")
**.feedbackComputation = xmldoc("config_channel.xml")

##########################################################
#              LTE specific parameters                  #
##########################################################

# Enable dynamic association of UEs (based on best SINR)
**.dynamicCellAssociation = true

**.car[*].masterId = 1     # useless if dynamic association is disabled
**.car[*].macCellId = 1    # useless if dynamic association is disabled
**.eNodeB1.macCellId = 1
**.eNodeB1.macNodeId = 1
**.eNodeB2.macCellId = 2
**.eNodeB2.macNodeId = 2
**.eNodeBCount = 2



# AMC module parameters
**.rbAllocationType = "localized"
**.feedbackType = "ALLBANDS"
**.feedbackGeneratorType = "IDEAL"
**.maxHarqRtx = 3
**.numUe = ${numUEs=15}

# RUs
**.deployer.ruRange = 50
**.deployer.ruTxPower = "50,50,50;"
**.deployer.antennaCws = "2;" # !!MACRO + RUS (numRus + 1)
**.deployer.numRbDl = 25
**.deployer.numRbUl = 25
**.numBands = 25
**.fbDelay = 1

# Enable handover
**.enableHandover = true
**.broadcastMessageInterval = 0.5s

# X2 and SCTP configuration
*.eNodeB*.numX2Apps = 1    # one x2App per peering eNodeB
*.eNodeB*.x2App[*].server.localPort = 5000 + ancestorIndex(1) # Server ports
(x2App[0]=5000, x2App[1]=5001, ...)
*.eNodeB1.x2App[0].client.connectAddress = "eNodeB2%x2ppp0"
*.eNodeB2.x2App[0].client.connectAddress = "eNodeB1%x2ppp0"
**.sctp.nagleEnabled = false         # if true, transmission of small packets will be
delayed on the X2
**.sctp.enableHeartbeats = false



# ---------------------------------------------------------------------------- #
# Config "VoIP-Uplink"
#
[Config VoIP-UL]

##########################################################
#                     App Layer                         #
##########################################################
```

```
*.server.numUdpApps = 10
*.server.udpApp[*].typename = "VoIPReceiver"
*.server.udpApp[*].localPort = 3000 + ancestorIndex(0)

*.car[*].numUdpApps = 1
*.car[*].udpApp[0].typename = "VoIPSender"
*.car[*].udpApp[0].destAddress = "server"
*.car[*].udpApp[0].destPort = 3000 + ancestorIndex(1)


# ----------------------------------------------------------------------------- #
# Config "VoIP-Downlink"
#
[Config VoIP-DL]

##########################################################
#                    App Layer                           #
##########################################################
*.server.numUdpApps = 10
*.server.udpApp[*].typename = "VoIPSender"
*.server.udpApp[*].localPort = 3000 + ancestorIndex(0)
*.server.udpApp[*].destAddress = "car[" + string(ancestorIndex(0)) + "]"
*.server.udpApp[*].startingTime = 0.05s


*.car[*].numUdpApps = 1
*.car[*].udpApp[0].typename = "VoIPReceiver"


# ----------------------------------------------------------------------------- #
# Config "VoIP-D2D"
#
# In this configuration, UEs run a VoIP application (using UDP as transport layer proto-
col)
# They communicate using the D2D link, if they are under the same cell
#
[Config VoIP-D2D]

# Enable D2D for the eNodeB and the UEs involved in direct commulteNications
*.eNodeB*.d2dCapable = true
*.car[*].d2dCapable = true
**.amcMode = "D2D"

# --- Set the D2D peering capabilities ---#
#
# For each D2D-capable UE, write a list of UEs (separated by blank spaces)
# representing the possible peering UEs. Note that this relationship is unidirectional.
# Here, car[0] --> car[5], car[1] --> car[6], etc.
*.car[0..4].lteNic.d2dPeerAddresses = "car[" + string(ancestorIndex(1)+5) + "]"

# --- Select CQI for D2D transmissions --- #
#
# To enable the reporting of CQIs for each D2D link, set the parameter
*.eNodeB.lteNic.phy.enableD2DCqiReporting
# To use fixed CQI, set the parameter **.usePreconfiguredTxParams and select the desired
CQI using the parameter **.d2dCqi
*.eNodeB*.lteNic.phy.enableD2DCqiReporting = true
**.usePreconfiguredTxParams = false

*.car[*].numUdpApps = 1
*.car[0..4].udpApp[0].typename = "VoIPSender"
```

```
*.car[0..4].udpApp[0].destAddress = "car[" + string(ancestorIndex(1)+5) + "]"

*.car[5..9].udpApp[0].typename = "VoIPReceiver"




# ----------------------------------------------------------------------------- #
# Config "D2DMulticast"
#
# In this configuration, a transmitting car sends periodic alert messages to neighboring
vehicles
#
[Config D2DMulticast]

### Enable D2D for the eNodeB and the UEs involved in direct communications ###
*.eNodeB*.d2dCapable = true
*.car[*].d2dCapable = true
**.amcMode = "D2D"

### Select CQI for D2D transmissions ###
# One-to-Many communications work with fixed CQI values only.
# Set the parameter **.usePreconfiguredTxParams and select the desired CQI using the pa-
rameter **.d2dCqi
**.usePreconfiguredTxParams = true
**.d2dCqi = ${cqi=7}

### Traffic configuration: one-to-many traffic between UEs (car[0] --> car[1..9]) ###
*.car[*].numUdpApps = 1

# Transmitter
*.car[0].udpApp[*].typename = "AlertSender"
*.car[0].udpApp[*].localPort = 3088+ancestorIndex(0)
*.car[0].udpApp[*].startTime = uniform(0s,0.02s)
*.car[0].udpApp[*].destAddress = "224.0.0.10"      # IP address of the multicast group
*.car[0].udpApp[*].destPort = 1000

# Receivers (they must belong to the above multicast group)
*.car[1..14].udpApp[*].typename = "AlertReceiver"
*.car[1..14].udpApp[*].localPort = 1000

# enrolled multicast groups must be set in the HostAutoConfigurator (instead of
demo.xml), seperated by a single space character
*.car[*].configurator.mcastGroups = "224.0.0.10"

[Config extCells_D2DMulticast]
extends=D2DMulticast

#external eNodeBs

*.numExtCells = 2

#============= Configuration ============
*.extCell[*].txPower = 20
*.extCell[*].txDirection = "ANISOTROPIC"
*.extCell[*].bandAllocationType = "RANDOM_ALLOC"
*.extCell[*].bandUtilization = 0.5

#============= Positioning ============
*.extCell[0].position_x = 100m
```

```
*.extCell[0].position_y = 600m
*.extCell[0].txAngle = 315
*.extCell[1].position_x = 600m
*.extCell[1].position_y = 600m
*.extCell[1].txAngle = 225
```