



Performance Analysis of Routing Protocols on VANETs

Ricardo de Bettencourt Fontes Preto dos Santos

Thesis to obtain the Master of Science Degree in

Telecommunications & Computer Science Engineering

Supervisors: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Dr. Afonso Mota da Conceição Oliveira

Examination Committee

Chairperson: Prof. Ricardo Jorge Fernandes Chaves
Supervisor: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Member of the Committee: Prof. Fernando Manuel Valente Ramos

December 2021

Acknowledgments

I would like to acknowledge my gratitude to my dissertation supervisor Prof. Teresa Vazão for her insight, experience, support and sharing of knowledge that has made this thesis possible.

I would also like to thank my family for their support, encouragement and caring over all these years, for always being there for me through my academic journey and without them this project would not be possible.

Last but not least, thanks to all my friends and colleagues that helped me and were always there for me.

Thank you.

Abstract

This thesis proposal aims to create a test environment for several routing protocols in Ad Hoc vehicular networks where the type of communications is predominantly of V2X. In order to observe different performance indicators for analysis and comparison of various protocols, these will be tested over three topology-based routing protocols: AODV, OLSR, and DSDV in four different mobility environments including urban, highway, country and a realistic scenario of Lisbon. However, precedently, this thesis will focus on the theoretical concepts in an effort to gain context of the Ad Hoc vehicular networks that are going to be analyzed. Therefore, concepts of how 5G will revolutionize the future of V2X communications and the current prism of Ad Hoc vehicular networks and its future will be addressed in this thesis. In addition, as this thesis aims to create a test environment for multiple protocols, all tools such as SUMO, NS3 and the created tool SUMO&NS3-Coupling will be subjects of a study so that this process can be carried out by someone in the academic world who desired to do a study in their own mobility scenarios.

Keywords

V2X; VANET; WAVE; 5G; AODV; OLSR; DSDV; VANET routing protocols.

Resumo

Esta proposta de tese tem como objetivo criar um ambiente de teste de vários protocolos de roteamento em redes veiculares Ad Hoc, onde o tipo de comunicação predominante é V2X. De forma a poder observar diferentes indicadores de performance para análise e comparação dos vários protocolos, nesta tese serão testados quatro protocolos topology-based diferentes: AODV, OLSR, e DSDV. Mas antes de a tese chegar a esse momento, este tese irá começar com conceitos teóricos de forma a ganhar um contexto à volta do que são as redes veiculares Ad Hoc que se vão analisar. Portanto, esta tese inicialmente vai introduzir os conceitos de como o 5G irá revolucionar o futuro das comunicações V2X e o atual prisma de das redes veiculares Ad Hoc e o seu futuro. Como esta tese tem o objetivo de criar um ambiente de teste de vários protocolos, todas as ferramentas como SUMO, NS3 e a ferramenta criada SUMO&NS3-Coupling vão ser alvos de um estudo para que este processo possa ser efetuado por alguém no mundo académico que queria fazer um estudo nos seus próprios cenários de mobilidade.

Palavras Chave

V2X; VANET; WAVE; 5G; AODV; OLSR; DSDV; protocolos de roteamento em VANETs.

Contents

1	Introduction	1
1.1	Introduction	3
1.2	Goals and Contribution	3
1.3	Organization of the Document	4
2	Related work	5
2.1	VANET	7
2.1.1	Type of communication	7
2.1.2	VANET protocol stack	8
2.1.3	Physical layer	9
2.1.4	MAC layer	10
2.1.5	Network layer	11
2.1.6	Transport and Application layers	12
2.2	Routing Protocols	13
2.2.1	Topology-based Ad-hoc Routing Protocols	13
2.2.1.A	AODV	14
2.2.1.B	OLSR	14
2.2.1.C	DSDV	15
2.2.1.D	DSR	15
2.2.1.E	ZRP	15
2.2.2	Broadcast Routing Protocols	16
2.2.2.A	SRB	16
2.2.2.B	DVCAST	16
2.2.2.C	PBSM	16
2.2.3	Cluster-based Routing Protocols	17
2.2.3.A	CBLR	17
2.2.3.B	CBDRP	17
2.2.4	Position-based Routing Protocols	17

2.2.4.A	GPSR	18
2.2.4.B	DREAM	18
2.2.4.C	LABAR	19
2.2.5	Infrastructure-based Routing Protocols	19
2.2.5.A	RAR	19
2.2.6	Final comparison	19
2.3	NS-3 Simulator	20
2.3.1	NS-3 Introduction	20
2.3.2	NS-3 Architecture	20
2.3.3	NS-3 Installation & Set-up	21
2.3.4	NS-3 Writing and Running Scripts	22
2.3.5	NS-3 Documentation	22
2.4	SUMO Simulator	22
2.4.1	SUMO Introduction	22
2.4.2	SUMO application areas	23
2.4.3	SUMO Installation & Set-up	23
2.4.4	SUMO Creating and Running Mobility Scenarios	24
2.4.5	SUMO Documentation	30
3	SUMO&NS3 Coupling	31
3.1	SUMO&NS3-Coupling Architecture	33
3.2	Vehicular Mobility Scenario Creation	34
3.2.1	Urban Grid Scenario	36
3.2.2	Highway Scenario	37
3.2.3	Country Grid Scenario	38
3.2.4	Realistic Scenario	38
3.3	SUMO&NS3-Coupling Translation	39
3.4	NS3 Simulation	42
3.5	SUMO&NS3-Coupling Results	42
3.5.1	Urban Grid Scenario	45
3.5.2	Highway Scenario	48
3.5.3	Country Scenario	50
3.5.4	Lisbon Scenario	53
4	Conclusion & Future Work	57
4.1	Conclusions	59
4.2	Future Work	59

Bibliography	61
A Code of Project	63
B Installation Guides for SUMO/NS-3	73
B.1 SUMO Installation & Set-up	73
B.2 NS-3 Installation & Set-up	75

List of Figures

2.1	Direct and Network Based communication in V2X	8
2.2	Wireless Access in Vehicular Environments (WAVE) protocol stack	9
2.3	Channels in vehicular networks according to the IEEE 802.11p standards	10
2.4	NS-3 architecture.	21
2.5	New network	24
2.6	Network mode menu	25
2.7	Edge creation	26
2.8	Network example	27
2.9	Demand mode menu	27
2.10	Vehicular settings and Routes	28
2.11	File and Edit menus	29
2.12	Mobility scenario in sumo-gui	30
3.1	SUMO&NS3-Coupling Architecture	34
3.2	Grid map	37
3.3	Highway map	37
3.4	Country map	38
3.5	Lisbon map	39
3.6	Network Animator	43
3.7	Example of AODV packet	44
3.8	Average Speed Grid Scenario	46
3.9	Running Vehicles Grid Scenario	46
3.10	Receive Rate Grid Scenario	47
3.11	Overhead Grid Scenario	47
3.12	Average Speed Highway Scenario	48
3.13	Running Vehicles Highway Scenario	49
3.14	Receive Rate Highway Scenario	49

3.15 Overhead Highway Scenario	50
3.16 Average Speed Country Scenario	51
3.17 Running Vehicles Country Scenario	51
3.18 Receive Rate Country Scenario	52
3.19 Overhead Country Scenario	52
3.20 Average Speed Lisbon Scenario	53
3.21 Running Vehicles Lisbon Scenario	54
3.22 Receive Rate Lisbon Scenario	54
3.23 Overhead Lisbon Scenario	55
B.1 Example of successfully installation	75
B.2 Home folder	76

List of Tables

2.1	Routing protocol characteristics	19
3.1	Grid Map characteristics	37
3.2	Highway Map characteristics	38
3.3	Country Map characteristics	38
3.4	Lisbon Map characteristics	39
3.5	Metric Averages of Grid Scenario	48
3.6	Metric Averages of Highway Scenario	50
3.7	Metric Averages of Country Scenario	53
3.8	Metric Averages of Lisbon Scenario	55

Listings

3.1	Portion of mobilityScenario.sumocfg	34
3.2	Portions of mobilityScenario.net.xml	35
3.3	Portions of mobilityScenario.rou.xml	35
3.4	Portion of mobilityScenario.xml	40
3.5	Portion of mobilityScenario.tcl	41
3.6	mobilityScenario_mobility_stats.csv	43
A.1	Portions of Simulation.sh	63
A.2	StatsParser.py	68
A.3	TclParser.py	69
B.1	SUMO prerequisites	73
B.2	NS-3 prerequisites	75

Acronyms

AODV	ad hoc on-demand distance vector
API	Application Programming Interface
AP	Access Point
BSS	Basic Service Set
CBD RP	cluster based directional routing protocol
CBLR	cluster Based location protocol routing
CCH	control channel
DREAM	distance routing effect algorithm for mobility
DSDV	destination sequence distance vector
DSRC	Dedicated Short-Range Communication
DSR	Dynamic source routing
DVCAST	distributed vehicular broadCAST
ETSI	European Telecommunications Standards Institute
GNU	GNUs not Unix
GPSR	greedy perimeter stateless routing
GPS	Global Positioning System
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ITS	Intelligent Transportation Systems
KPI	Key Performance Indicators
LABAR	location area based ad-hoc routing
LREQ	location request
LTE	Long Term Evolution
MAC	Media Access Control

MANET Mobile Ad Hoc Network
MCTP Mobile Control Transport Protocol
MPR multipoint relays
NS-2 Network Simulator 2
NS-3 Network Simulator 3
NS2 Network Simulator 2
NS3 Network Simulator 3
NetAnim Network Animation
OFDM Orthogonal Frequency Division Multiplexing
OLSR Optimized Link State Routing
OS Operating system
PBSM parameterless broadcast in static to highly mobile
POI Point-of-interest
QoS Quality Of Service
R-DSDV randomized-destination sequence distance vector
RAR roadside-aided routing
RERRs Route Errors
RREPs Route Replies
RREQs Route Requests
RSUs Road Side Units
RSU Road Side Unit
SCH service channels
SRB secure ring broadcasting
SUMO Simulation of Urban Mobility
TCP Transmission Control Protocol
UDP User Datagram Protocol
V2I Vehicle-to-Pedestrian
V2N Vehicle-to-Network
V2P Vehicle-to-Infrastructure
V2V Vehicle-to-Vehicle
V2X Vehicle-to-Everything
VANET Vehicular Ad Hoc Networks
VITP Vehicular Information Transfer Protocol
VTP Vehicular Transport Protocol

WANET Wireless Ad-hoc NETwork

WAVE Wireless Access in Vehicular Environments

Wi-Fi Wireless Fidelity

WiMAX Worldwide Interoperability for Microwave Access

ZRP zone routing protocol

1

Introduction

Contents

1.1 Introduction	3
1.2 Goals and Contribution	3
1.3 Organization of the Document	4

1.1 Introduction

Over the past few decades, due to the rapid growth and evolution of our population in addition to needs that are constantly more demanding without any signs of deceleration, our society has been experiencing the benefits of new and more advanced mobile communications generations. Furthermore, this evolution of communications systems towards the current 5G telecommunication generation is expected to meet various communication requirements of future industrial or commercial fields. Currently, information and communication technology is not only the key driving factor for some of the most important innovations in the automotive industry, but also the future of Intelligent Transportation Systems (ITS). This is motivated by the consumer demand for a vast variety of ITS applications, for instance autonomous driving and, as a result, researchers are exploring new and more efficient network architectures, such as Vehicular Ad Hoc Networks (VANET). VANETs have emerged as a fascinating research and application field. As an increasingly number of vehicles are being equipped with more and more technology such as sensors, processing and wireless capabilities are enabling a new paradigm of possibilities to revolutionise the ITS, more particularly in road safety, efficiency, and comfort. VANETs are a subclass of Mobile Ad Hoc Network (MANET) which belongs to a family of Wireless Ad-hoc NETWORK (WANET). Regarding MANETs, they are fundamentally a self-organizing communication system that is not dependent on any infrastructure and is mostly used in military, however, nowadays it is gaining ground on civilian applications. Moreover, MANETs communications are equal to the basic communication methodology on Bluetooth ad hoc networks used for data sharing between mobile phones. At last, the basic principle of VANETs is the same as MANETs but, applied in vehicular scenarios where nodes are the cars with embedded sensors and communication systems or fixed infrastructure consisting of Road Side Units (RSUs).

1.2 Goals and Contribution

The objective of this thesis is to create a simulation environment for various routing protocols in VANETs networks and in this way to take advantage of Vehicle-to-Everything (V2X) communications. This environment has to be intuitive and easy to use, so that any student or researcher can use this environment to be able to make comparisons between the routing protocols. And for that, the SUMO&NS3-Coupling tool software was developed, this software will enable any student or researcher to seamlessly use the software to compare the routing protocols. Also, this thesis aims to provide a background on the theoretical concepts involved with VANET scenarios as well as the tools used.

1.3 Organization of the Document

This thesis is organized as follows:

- **Chapter 2 - Related Work:** This chapter is dedicated to all the surround work behind the goals of this thesis. Diving into the theoretical aspect of VANETs as well as overview of the most known routing protocols for vehicular networks. In addition, this chapter also gives a background of two key softwares to achieve the goals of this thesis, the Network Simulator 3 (NS-3) and Simulation of Urban Mobility (SUMO).
- **Chapter 3 - SUMO&NS3-Coupling:** This chapter is strictly dedicated to the SUMO&NS3-Coupling program that I have developed to achieve the goal of this thesis. Starting with the architecture of the have a brief introduction to how the program work. Then, subsequently diving into each phase of the program, to full understand how it works. Starting with the creation of vehicular mobility scenarios until the end, then going through how SUMO&NS3-Coupling combines both SUMO and NS-3 until the last phase, with the analysis over the performance of each routing protocol.
- **Chapter 4 - Conclusion & Future work:** The last chapter is dedicated to the conclusion thoughts and future work ideas.

2

Related work

Contents

2.1 VANET	7
2.2 Routing Protocols	13
2.3 NS-3 Simulator	20
2.4 SUMO Simulator	22

2.1 VANET

VANETs or Vehicular Ad Hoc Networks are not the common type of network we are used to see in our daily basis where we have a fixed topology network where only the terminals are dynamically changing position, for instance the example of our phones. Therefore, in contrast, VANETs are a special case of a Mobile Ad Hoc Network (MANET), highly dynamic and intermittent connected typologies networks due to the nature of constant and fast mobility of vehicles which bring new challenges to data communication and its Quality Of Service (QoS) requirements [1]. VANETs will be essential to the new paradigm of autonomous driving and for intelligent transportation systems, and this new paradigm is not far away. Since, nowadays various automotive manufactures are already equipping their vehicles with onboard computing, sensors as well as wireless communications devices and navigation systems such as GPS in preparation to this new paradigm. However, this section will not only, but focus more on the technical aspect of the VANETs, diving into the routing protocols which is the main subject of this thesis.

2.1.1 Type of communication

Currently, the advances in mobile communications allow us different deployments of architectures for vehicular networks in urban areas, highways, and rural environments via ad hoc networks to support different applications and its QoS requirements. Therefore, a VANET regardless of the environment where it is operating, is going to utilize new types of communication between vehicles and fixed road-side equipment and infrastructure. These communications are grouped in what is called Vehicle-to-Everything (V2X) that is based on two types of communication, direct or ad hoc based and network based which have the following communications list of possibilities and represented in the Fig. 2.1 [2,3]:

- **Vehicle-to-Vehicle (V2V):** Direct based communication that allows direct communication between vehicles without relying on the road side or fixed infrastructure.
- **Vehicle-to-Infrastructure (V2I):** Network based communication that allows communication between the vehicles to the infrastructure.
- **Vehicle-to-Pedestrians (V2P):** Direct based communication that allows direct communication between vehicle and pedestrians.
- **Vehicle-to-Network (V2N):** Network based communication that allows communication between the vehicles to the network.

Also, when dealing with a VANET there are some characteristics to consider beyond the type of communication such as:

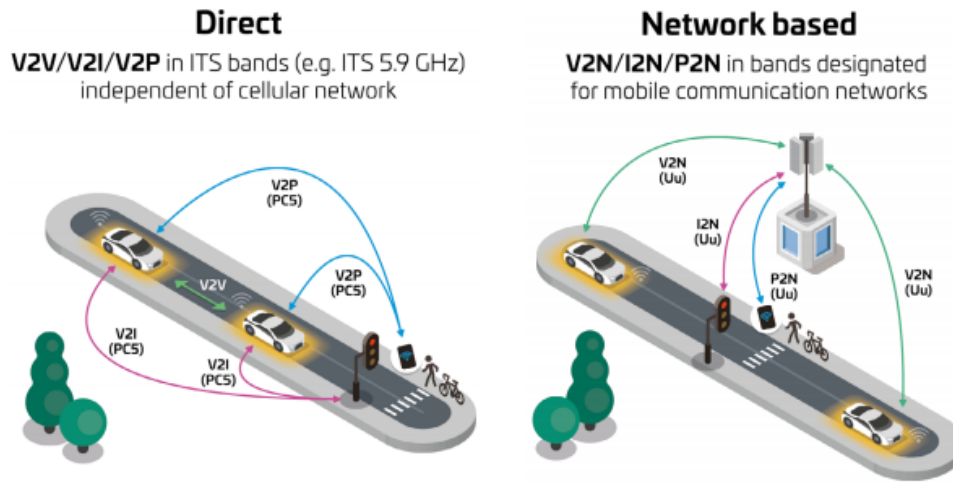


Figure 2.1: Direct and Network Based communication in V2X
[2]

- **Highly dynamic topology:** VANET networks due to the speed of the nodes and range of the radio signal that is mainly dependent on radio wave frequency used.
- **Frequently disconnected:** Since the nodes are highly dynamic, those can be in and out of range in a matter of seconds due to variation of speeds, causing frequently changes on the state of the connection between nodes and updates on the routing tables of each node.
- **Geographical position and patterns:** In case of some routing protocols, in particular the geographic based that which will be explained forwarder, can benefit from geographical information in order to predict mobility pattern for future routing purposes.
- **Propagation model:** For most of cases, VANETs operate in three environments such as urban scenarios, highways and rural. Therefore, a network has to be able to adapt the propagation models between each environment, since it is known that rural areas are not as dense a urban areas. Where the signal can suffer from interference and reflections, or even total loss of signal due to blocking by the buildings.

Therefore, there are spatial and temporal constraints with this kind of network whereas fixed network don't and that need to be taken into account to the design of communication protocols in VANETs.

2.1.2 VANET protocol stack

The protocol stack for vehicular networks has to manage communication with nearby vehicles and between them, pedestrians and roadside equipment as previous mentioned. Therefore there is a protocol

stack designed to handle all the challenges mostly based on the IEEE 802.11 Wireless Access in Vehicular Environments (WAVE) standards [4] as illustrated in the Fig. 2.2 [5].

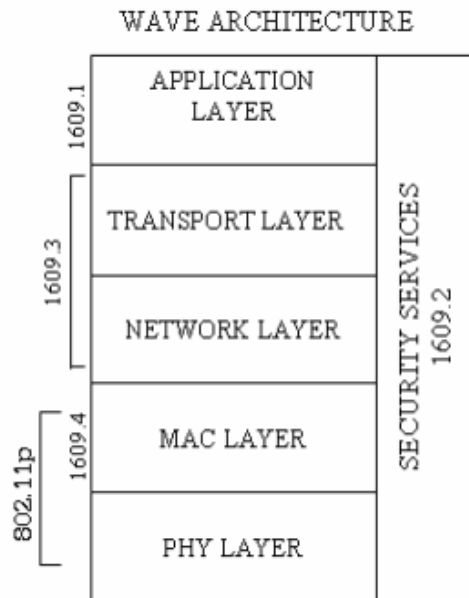


Figure 2.2: WAVE protocol stack [5]

2.1.3 Physical layer

The physical layer in vehicular networks is a challenging one compared with our typical fixed topology networks. For instance, the protocols in this layer must take into account the multipath fading as well Doppler effect in radio wave frequencies shifts caused by the fast movements of the nodes specially in highway scenarios. Vehicle-to-vehicle communication have use radio wave usually on very high frequencies [6], for instance micro and even millimeter waves which are also used in 5G, are used. Note that millimeter waves are only used in line-of-sight communication, whereas the microwaves are used in the broadcast type communications. The frequencies used, were defined in the Dedicated Short-Range Communication (DSRC) system which is dedicated to VANETs. This system is as the name suggest is a short to medium range communication technology that operates around the 5.9GHz band. In which, according to the European Telecommunications Standards Institute (ETSI), 70 MHz where allocated so it operates in the 5.885-5.925 Ghz band. The DSRC system is able to manage speeds up to 200km/h, and a transmission range up to 1000m. In addition, the DSRC is known as the IEEE 802.11p WAVE standards where it also defines the function and services that operate in vehicular networks without the need of a Basic Service Set (BSS), which means that no common Access Point (AP) is needed in the network to provide communication between nodes. The IEEE 802.11p also defines interfaces functions

between the physical layer and the Media Access Control (MAC) layer, sharing the same logical channel as we can see in the Fig. 2.2 as well as the other DSRC channels, each with represent with the IEEE 1609 standards which are divided by the following list from the article [6, 7]:

- **1609.1:** Specifies the services and interfaces of the WAVE Resource Manager application.
- **1609.2:** Defines secure message formats and processing.
- **1609.3:** Defines network and transport layer services including addressing and routing, in support of secure WAVE data exchange.
- **1609.4:** Enables operation of upper layers across multiple channels, without requiring knowledge of PHY parameters.
- **802.11p:** Define the WAVE signaling technique and interface functions that are controlled by the IEEE 802.11 MAC.

Also according to the ETSI and In the Fig. 2.3, the frequency band is divided into six service channels (SCH) and one control channel (CCH), each one with a band of 10 MHz and all of them filling the 70 Mhz band allocated previously mentioned to the DSRC. Each channel is allocated to three types of applications. The ITS non-safety 5.855-5.875 MHz, safety, and traffic efficiency 5.875-5.905 Mhz, Future ITS 5.905-5.925 Mhz [8, 9] .

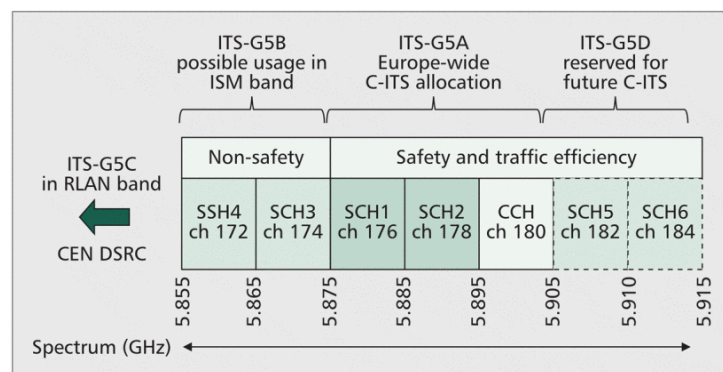


Figure 2.3: Channels in vehicular networks according to the IEEE 802.11p standards [10]

2.1.4 MAC layer

MAC layer protocols are responsible for managing the use of a share medium, therefore these protocols decide which nodes will access the medium at any given time. The MAC layer in case of vehicular networks has to provide a reliable, stable and efficient channel. Also, MAC protocols should consider the different applications for which the communication will occur. For example, messages related with

safety applications have to be sent rapidly, with low failure rate. Therefore, this calls for a resilient medium of communication, which is even more challenging when dealing with VANETs due to the highly node mobility and topology changes. And this is especially important since with the 5G capabilities the trend is to use even more multimedia applications by passengers, demanding more throughput in the VANET network. In addition, in VANETs, the bandwidth has to be shared between vehicles, that being said, the use of Orthogonal Frequency Division Multiplexing (OFDM) technology to control the medium access communication to avoid collisions. Fortunately, the IEEE 802.11p WAVE protocol is designed to fulfill the requirements present in V2X communications, where high reliability and low latency are mandatory. This is done by enabling a very efficient communication setup with little overhead and removing the BSS operations from the IEEE 802.11 in a truly ad hoc environment for vehicles [7].

2.1.5 Network layer

The network layer, the layer that allows for the connection and transfer of data packets between the nodes by using routing protocols that implement viable ways of communication without disruption. With that said, in vehicular networks it supports different communications:

- **Unicast communication:** Type of communication from the source node to target node end-to-end in the network via multi-hopping. Where the target node may be at a known location or within a certain range. Despite this communication usefulness in VANETs, multicast is better suited for applications that require dissemination of messages to different nodes in the network.
- **Multicast/Geocast communication:** Type of communication, where the data transmission is addressed to a group of targets simultaneously. Geocast is based on the Multicast but takes into account the geographical location into the mix. In which a message is sent to a group of targets node based on their geographic position, commonly based by the relative distance to the source of the message.
- **Broadcast communication:** Type of communication, where the source node sends data to everyone on the network at once. However, in vehicle networks the broadcast works a little bit differently from the typical fixed networks. In these networks, the nodes are scattered in the space, which means that. Probably in most of the cases, the nodes may not be within the range transmission of the source node. To prevent this, the target nodes from the first source, relay the data also in broadcast mode repeating the process until no nodes are within range of the source. Forming a chain of broadcast messages that breaks when no vehicles are in range. In addition, it is with broadcast that the nodes discover their neighbours in the discovery phase of the routing protocols in order to find the most efficient route for the unicast communications.

With mention of routing protocols, it is to note that routing protocols are very different from the typical fixed networks and since there are some substantial differences. With that said, the routing protocols will have their own section 2.2 dedicated to them ahead, despite their operation in this section of network layer.

2.1.6 Transport and Application layers

As previous mentioned, vehicular networks are characterized by the highly node mobility, rapid topology changes and intermittent connectivity. And in contrast with other ad hoc networks, VANETs present a more predictable mobility patterns. Since in this environment, vehicles have a somewhat predictable path due to the road network. The patterns can improve the short time window that the nodes have in high mobility to exchange data the by increasing good throughput, or goodput. In VANETs, many unicast applications it is safe to assume that the Transmission Control Protocol (TCP) protocol characteristics regarding the reliability and in-order data. Unfortunately, this protocol in terms of performance is not the greatest in vehicular networks with high mobility and frequent topology changes [11]. In addition, Vehicular Transport Protocol (VTP) is a transport protocol for unicast applications in VANETs that make use of path characteristics to improve performance. Other protocol is the Mobile Control Transport Protocol (MCTP) which is based on similar principles of the Ad Hoc TCP protocol with the aim of trying to provide end-to-end QoS between vehicles and the roadside infrastructure. In the application, protocols should minimize the end-to-end communication delay, which is important when providing emergency messages. In addition, these protocols when dealing with emergency messages have to take into account some factors and comply to them in an appropriate time window to guarantee the vehicle's driver receive all the information. Those factors can be the location where they are generated due an emergency event, velocity of the receiver vehicle. Also, some applications need to be designed with a commercial mindset for business such as restaurants, hotels, gas station among others can broadcast their marketing information in VANETs. Application protocols may also be used for transactions with Vehicular Information Transfer Protocol (VITP) [12]. Which is an application layer communication protocol designed to handle the establishment of a distributed ah hoc service infrastructure in VANETs. The field of application in this space is evolving rapidly due to the evolution of communication such as 5G influencing existing use cases or futures ones such as:

- **Safety Applications:** Aiming to improve the road safety.
- **Efficiency Applications:** Aiming to improve the road efficiency.
- **Comfort Applications & Interactive Entertainment:** Aiming to improve the passengers comfort.
- **Autonomous Driving:** The most ambitious, that aims to revolutionize our ITS, by aiming to switch the all vehicular mobility to autonomous driving.

2.2 Routing Protocols

Due to the high mobility of nodes in a VANET environment, designing routing protocols able to compute and handle efficiently many routing paths among vehicles, represents nowadays a challenging research issue. Until now, several routing protocols have been developed, some of them have been adaptations improving already established algorithms from MANETs. However, this protocols despite having been demonstrated on how they can perform well for MANETs, that does not mean that they are able to guarantee the same level of efficiency into VANETs scenarios. And, that is why new approaches and more sophisticated strategies have been developed where a lot of this approaches manages the routes starting from the information about the node location where other protocols group the nodes into smaller clusters [13].

Being said that , the rest of this section is dedicated to a summary of the most know VANETs routing protocols that can be divided in five different categories [14]: Topology-based, Broadcast, Cluster-based, Position-based and Infrastructure-based.

2.2.1 Topology-based Ad-hoc Routing Protocols

In this category of routing protocols there are some algorithms that have been designed for MANETs and have been adjusted to fit a VANET environment. Topology-based protocols can be divided in categories, proactive, reactive and hybrid.

- **Proactive:** In this type of routing protocol, each node on the network keeps on maintaining regularly the routing table to store the routes information for every other node. Therefore, each table entry contains the information of the next-hop, despite the route being needed or not by using the Bellman Ford Algorithm. Since we are in a VANET environment, this tables must be updates regularly to reflect the topology changes of the network, and to perform that each node has to broadcast regularly to discover its neighbours. However, this has a downside, it produces overhead cost due to maintaining up-to-date information and as a result it may affect the throughput of the network. Upside is that whenever is necessary, it has the availability information of the next-hops. Also, the proactive routing protocols relies on the shortest path algorithm to find out the optimum route, for that there are two kind of strategies, link state strategy and distance vector strategy.
- **Reactive:** In this type of routing protocol, each node on the network keeps on maintaining only the routes in need. Therefore, each node starts a route discovery process when it wants to send data if the path is not already known. This network paths searching, relies on handshake by flooding route request messages and it reaches the destination node, the destination node replies in unicast communication forming a connection. That means that reactive protocols are more suited for

dense networks, high mobility that frequently change typologies due to the reduce overhead of maintaining the routing tables of the proactive protocols.

- **Hybrid:** In this type of routing protocols, as the name suggests it is the middle ground between the above two protocols, the proactive and reactive. This protocol aims to take advantage of the strengths of the other two while reducing the limitation, such as routing network overhead from proactive protocol and delay in the routing discovery process from the reactive protocol. In this hybrid protocol, the whole network is divided into zones. Making it easier to maintain routing tables, and even increasing efficiency in discovery process. Also, in this protocol each node has been labeled as region inside node, or region outside node.

2.2.1.A AODV

AODV protocol stands for ad hoc on-demand distance vector routing protocol. Which is a reactive protocol that enables dynamic, self-starting, multihop routing between nodes wishing to establish and maintain an ad hoc network. Since it is a reactive protocol and allows nodes to obtain routes quickly for new destinations and does not require nodes to maintain routes to destinations that are not in active in communications. Doing all this while dealing with link breakages and changes in network topology in an acceptable time window since when a link breaks, this protocol notifies the affected set of nodes by sending a route error message, so that they can invalidate the routes using the lost link quickly. Also, the operation of AODV protocol is loop-free, and by avoiding the Bellman-Ford "counting to infinity". And does that by using a simple solution and a distinguishing feature of this protocol which is the use of a destination sequence number for each route entry. In addition, the AODV protocol uses the User Datagram Protocol (UDP) transport protocol to send its own messages that are mostly Route Requests (RREQs), Route Replies (RREPs), and Route Errors (RERRs).

2.2.1.B OLSR

OLSR protocol stands for Optimized Link State Routing protocol. Which is a proactive protocol, meaning that is a routing table driven protocol that exchanges and manage topology information regularly between nodes of the network. To do that, each node selects a set of neighbours nodes as multipoint relays (MPR), these nodes are responsible for forwarding the control traffic, intended for diffusion into the whole network. MPRs provides an efficient mechanism for flooding control traffic by reducing the number of transmissions required. Another responsibility of the nodes selected as MPRs, is to declare all the link state information in the network periodically over the control messages in order to OLSR maintain the shortest paths routes updated and for redundancy. In addition, in route calculations, the MPRs are used to form the route from a source target to any destination in the network.

2.2.1.C DSDV

DSDV protocol stands for destination sequence distance vector routing protocol. Which is a proactive protocol, meaning that is a routing table driven protocol based on the distance vector strategy and applies the shortest path algorithm of Bellman-Ford. In this protocol only one optimal route is stored in the routing table for each destination while having the information to all approachable network's nodes with the destination nodes and its costs. Similarly, to the AODV, this protocol also stores as a label the sequence number of its routes in order to avoid the Bellman-Ford "counting to infinity" problem. DSDV maintains the routes by periodically broadcasting the control messages to its neighbours. Since DSDV is a proactive protocol, it is more prone to overhead with the increase number of nodes due to the addition overhead to maintain the routing tables. However, the main limitation of DSDV routing protocol is that lacks congestion control for the network, multiple paths for destinations decreases the DSDV routing efficiency. These limitations were mitigated with a new protocol based on the DSDV, the randomized-destination sequence distance vector (R-DSDV). Which provides support for network congestion control, but with that also having an increase of overhead compared to DSDV.

2.2.1.D DSR

DSR protocol stands for Dynamic source routing protocol. Which is a reactive protocol, with the main objective to provide a very low network overhead yet was able to react very quickly to changes in the network. The DSR protocol provides successful data packet delivery despite of network changes by taking advantage of its tow step mechanism, route discovery and route maintenance that allows respectively to discover and maintain the routing tables. The DSR is a multi-hop routing protocol, that in route discovery or route maintenance operate strictly on demand thence being a reactive protocol. And, unlike other protocols, the DSR protocol does not requires periodic sending of packets of any kind within the network. For instance, the DSR does not use the routing advertisement, link status or even neighbour detection and relies on the package's headers. When some source node, originates a new packet to some destination, then source node places in the header of the packet altered, giving the sequence of hops to the destination. Normally, the sender will obtain a route by searching its cache of routes previously learned. If no route is found, it will initiate the route discovery protocol to find a route. However, this approach can have a significant overhead in early stages, due to the caches being empty.

2.2.1.E ZRP

ZRP protocol stands for zone routing protocol. Which is a hybrid protocol, that according to its name, divides the whole network into multiple zones based on some criteria such as transmission power required, transmission signal strength, mobility of the nodes among others. Since ZRP is a hybrid protocol, ZRP

uses the proactive routing approach for inside nodes of zone and reactive routing approach for outside nodes of zone.

2.2.2 Broadcast Routing Protocols

In general, the role of a protocol is to find a route to connect two nodes. However, routing algorithms based on broadcast protocols have a different aim. This kind of protocols are used whenever the destination node is out of the range of the source node. Mostly these protocols are used with application that are concerned with the safety such as road and weather condition warning, emergency warning messages, road conditions among others, where the information is use full to every node. The positive side of these kind of protocols is the reliability, therefore being used for safety. The downside is that these types of protocols consume more bandwidth, and many duplicate packets reaches the node which is not an efficient use of resources.

2.2.2.A SRB

SRB stands for secure ring broadcasting routing protocol, which is a broadcast routing protocol that divides the nodes into three different classes, inner nodes (IN), outer nodes (ON) and secure ring nodes (SRN). The inner nodes, as the name suggest are the ones closer to the source node, then we have the outer nodes which are far away from the source node and finally the secure ring nodes which are within the maximum distance defined from the source node. So, in the SRB only the nodes within the secure ring can broadcast the packets more than one time to destination.

2.2.2.B DVCAST

DVCAST stands for distributed vehicular broadCAST routing protocol, which is a broadcast routing protocol that operates based on the local information about its neighbours to maintain communication. DVCAST protocol gets information about the network by sending periodic beacon messages. When a source node does not find enough nodes it does not broadcast, the packet will be cached until nodes get into broadcast range, if no node is found the packets will eventually be discarded. Also, this protocol deals with message duplication by using flag parameters.

2.2.2.C PBSM

PBSM stands for parameterless broadcast in static to highly mobile routing protocol, which is a simple broadcast routing protocol. That to eliminate redundant broadcasting operates dividing neighbours nodes into two classes, the nodes that received (R) and the node that did not receive (NR) any packets. Which helps to detect neighbors that already received and that which did not receive the packet.

2.2.3 Cluster-based Routing Protocols

As said previously, in VANETs the topology changes are very frequent over, usually large areas. Therefore, dealing with scalability is usually a big issue. One way of dealing with that issue is by dividing the network in different regions or clusters, which coordinate and communicate with each other to achieve communications between nodes. Moreover, if a vehicle node needs to communicate with another node within the cluster then the communication is a direct path as it is considered to be a local communication. If the vehicle node needs to communicate with another node outside the cluster then it requires the help of its cluster head for reaching the destination. The positive side of cluster-based protocols are the scalability factors as it makes a good choice for complex networks over large areas. However, the drawbacks are traffic delays.

2.2.3.A CBLR

CBLR stands for cluster Based location protocol routing protocol, which is not only a broadcasting protocol but also acts like a reactive protocol or on demand routing protocol. Since a routing table is used by each cluster header, which contains all the addresses and locations of the cluster member nodes. In addition, cluster headers also track information about the neighbour clusters in the cluster neighbour table. Therefore, when a source wants to send a packet to a destination node, it sends it to the closest neighbour if it is in the same cluster. If the destination node is in another cluster, then it caches the packets and broadcasts location request (LREQ) packets to find the destination node.

2.2.3.B CBDRP

CBDRP stands for cluster based directional routing protocol, which is a broadcasting protocol where the nodes are being clustered according to the direction of the vehicles, so the nodes that are moving in the same direction are grouped together in a cluster, and one of the nodes is elected the cluster-head. Then, if a packet has to leave the cluster, the packet needs to be forwarded to the header of the cluster which sends it to the header of the node's cluster header.

2.2.4 Position-based Routing Protocols

In these kinds of protocols, a source node will communicate to the destination node using by using geographical positions as well as with its network address. The geographical position of the nodes, can be obtained naturally through Global Positioning System (GPS) or V2I communication since it is know the location of the Road Side Unit (RSU) infrastructure [15] that can act as redundancy whenever the satellite signals is weak when the vehicle goes in the area like tunnel. In the position-based routing protocols there is a specific category, Geocast routing, in which the nodes are being the nodes are

being divided into predefined geographical positions regions. And, to forward the packets there are three strategies:

- **Greedy forwarding:** In this strategy, protocols do not create a path from source to destination, they forward the packets to the next-hop tanking in consideration the position of other nodes, for instance the closest neighbour to destinations in terms of distance instead of the typical cost in other protocols.
- **Restricted directional flooding:** In this strategy, protocols forward the packets to several nodes like broadcasting, however is a directional one instead of every node.
- **Hierarchical:** In this strategy, protocols create a hierarchy according to position of the vehicle to escalate large number of nodes.

Since position-based protocols use geographical location information of the nodes within the network. In a vehicular scenario such as VANETs, movements are usually restricted in a few directions based on the road network, therefore having an advantage of predictability and performance over other routing protocols designed for VANETs. Hence, position-based protocols being nowadays the most promising protocols for VANETs scenarios.

2.2.4.A GPSR

GPSR stands for greedy perimeter stateless routing protocol, which is a protocol that uses the greedy forward approach to find the shortest paths to its destination nodes and takes advantage of its correspondent graphical position and connectivity in the network. By using the positions of the nodes to make packet forward decisions. Therefore, GPSR uses the greedy forwarding approach to forward packets to nodes that are always progressively closer to the destination node. However, in regions of the network where the greedy path is not found, GPSR recovers by forwarding the packet in perimeter routing strategy mode, in which the connectivity graph switches into a planar subgraph, until reaching closer to destination, where the greedy forwarding resumes.

2.2.4.B DREAM

DREAM stands for distance routing effect algorithm for mobility routing protocol, which is a protocol that uses the restricted directional flooding approach. Therefore, the packets are being forwarded to several next-hop nodes to reach the destination. The protocol limits the number of broadcasts by broadcasting the packets to only certain regions. Those regions are calculated using the positions of each node on the network, with that the expected region is calculated is predicted and the protocol forwards and broadcast the packets to those regions.

2.2.4.C LABAR

LABAR stands for location area based ad-hoc routing protocol, which is a protocol that uses the hierarchical position-based approach. LABAR creates location areas using G-nodes and V2I communication. Forming, a virtual back-bone network, where the G-nodes are able to communicate with the local service infrastructure to learn about the S-nodes. To forward a packet LABAR creates areas using the back-bone network and directional routing. So, once the zones are defined with the G-nodes forming the top hierarchical network with the correspondent multiple S-nodes. With this said, if communication takes place within a zone it uses the S-nodes, if it goes outside the zone, G-node back-bone network must be used.

2.2.5 Infrastructure-based Routing Protocols

As said previously in the position-based routing protocols 2.2.1, the use of road side infrastructure can be used as redundancy whenever the satellite signals is weak. However, in infrastructure-based routing protocols instead of being used as redundancy is the main source of information relying on fixed infrastructure bases to assist routing issues.

2.2.5.A RAR

RAR protocol stands for roadside-aided routing protocol. Which is a protocol that takes advantage of road side infrastructures to assist routing. In this protocol, geographical areas are divided into sectors with RSUs, enabling V2I communication between RSUs. This protocol is preferable in urban scenarios since is where most infrastructure is already in place.

2.2.6 Final comparison

This section presents a small summary of the topology-based protocols which are the ones that will be used ahead on the experimental side of this thesis.

Table 2.1: Routing protocol characteristics

Routing Protocol	OLSR	DSDV	AODV
Protocol Category	Topology-based	Topology-based	Topology-based
Protocol Type	Proactive	Proactive	Reactive
Topology Structure	Flat/Hierarchical	Flat/Hierarchical	Mostly Flat
Approach	Link State	Distance vector	Distance vector
Storage Requirements	High	High	Low
Scalability	Overhead increased	Overhead increased	Overhead increased
Advantages	Reduce control overhead	Loop free	Low overhead
Drawback	Throughput	Throughput	Large Delay
Simulation Tools	NS2, NS3, OPNET	NS2, NS3	NS2, NS3, OPNET

2.3 NS-3 Simulator

This section is related to the NS-3 Simulator. It is going to include an overview of what is the NS-3, how is the development process, its architecture, applications as well as how to install it and run the custom scripts for the testing of the various routing protocols for the purpose of this thesis.

2.3.1 NS-3 Introduction

NS-3 or network simulation 3 is a discrete event network simulator successor of NS-2, intend to focus primarily on research and educational use as it is a free software, licensed under the GNU General Public License version 2 [16]. Therefore, license was created in order to guarantee the freedom to share or modify the software which is relevant for any researcher that intends to use and modify the NS-3 for his research.

The NS-3 project is one of few reference projects of open-source simulator for extensible network simulations platform, for networking research and education, as it should be aligned with the simulations needs of current networking research that is developed with the motivation and contribution by the community, mainly establish of students and researchers. These contributors aim to build the NS-3 project to a solid simulation core that is well documented and caters to the need of the entire simulation workflow, from simulations configurations to performance analysis of real-time network emulators that could be interconnected with the real world as it allows many existing real-world protocol implementations to be used in within NS-3 simulations and can be found in the following Gitlab repository <https://gitlab.com/nsnam>.

In brief, NS-3 provides models of how packet data networks work and perform, by providing a simulation engine core supports research of both IP and non-IP based networks. Capable of performing studies that are difficult or even impossible to perform in real networks, by studying the network behavior in a controlled and reproducible environment. However, most of its users focus on wireless/IP simulations which involve models for Wi-Fi, WiMAX or LTE for the first and second layer and a variety of static or dynamic routing protocols such as OLSR and AODV as mentioned in previous sections.

2.3.2 NS-3 Architecture

The NS-3 simulator is library designed in C++ and provides a set of network simulations models implemented in C++ objects and wrapped through python. Therefore, to use this library the application must be coded in C++ or python to instantiate a set of simulations models to set up and run the simulation scenario of interest.

As the NS-3 library is wrapped through python using the pybindgen library, this means that the delegation of parsing of the NS-3 c++ headers to gccxml and pygccxml to generate the corresponding

c++ files which are compiled into the NS-3 python module to allow the users to interact with the NS-3 modules and core through python scripts. As shown in the following Fig. 2.4

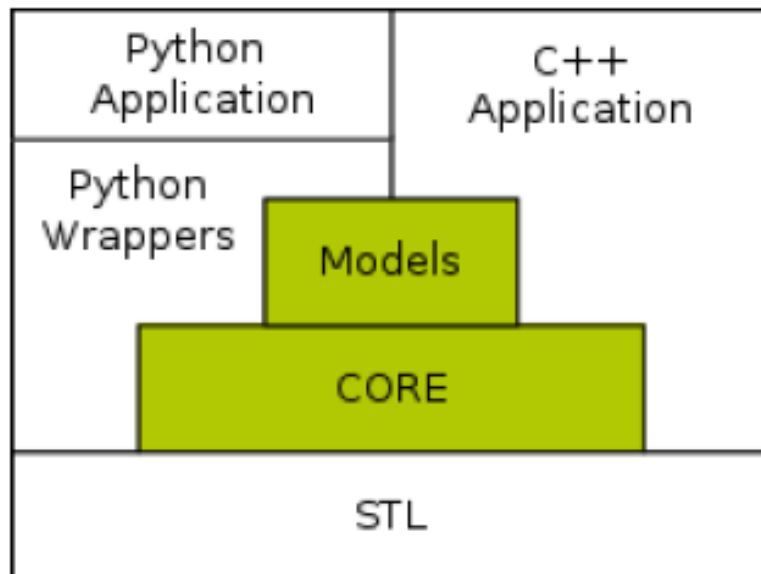


Figure 2.4: NS-3 architecture.
[17]

Unlike most of other discrete-event simulators, NS-3 is distinguished by the high-level design. Simulation events in NS-3 are made by simple function calls that are scheduled to execute at a prescribed simulation time as we are going to see in a more detailed manner in the following sections. With that said, any function can be defined as an event and scheduled by the use of a callback function which means that any event can call invoke other events.

2.3.3 NS-3 Installation & Set-up

The NS-3 simulator is available on Linux, Mac OS and even Microsoft Windows using Cygwin, however it is suggested to use the Linux environment as it is the most stable, since NS-3 is primarily developed on GNU/Linux platforms. Therefore, this section will provide a brief guide to the installation of NS-3 on the Ubuntu distribution of Linux. However, before proceeding with the installation of NS-3, there are some prerequisites that are needed. Those requisites are due to libraries of NS-3 which have several dependencies on third-party libraries such a C++ or python among others. The guide for installation and set-up can be found attached to the appendix B.

2.3.4 NS-3 Writing and Running Scripts

While experienced users of NS-3 often write their own scripts, the less experienced ones start with the examples provided in the NS-3 build. In this section we will run the famous example of hello world scripts, and just outputs to the terminal Hello World. For that, we need a terminal window opened in the NS-3 directory `/ns-allinone-3.34/ns-3.34` by typing the following command:

```
1 ./waf --run hello-simulator
```

Note that, for running the script the `./waf` & `--run` are present. This are indispensable to run any script.

2.3.5 NS-3 Documentation

As the NS-3 is not the primary focus of this thesis, but rather an important tool for the experimental aspect, this section is focused on the available resources for further information about the NS-3.

- Documentation: <https://www.nsnam.org/documentation/>
- User FAQ: https://www.nsnam.org/wiki/User_FAQ
- HOWTOs: <https://www.nsnam.org/wiki/HOWTOs>

2.4 SUMO Simulator

This section is related to the SUMO simulator. It is going to include an overview of what is SUMO, how is the development process, its architecture, applications as well as how to install, and finally how to create custom mobility scenarios for the testing of the various routing protocols for the purpose of this thesis.

2.4.1 SUMO Introduction

SUMO or Simulation of Urban Mobility is a free and open-source vehicular traffic simulation intend to focus primarily on research, and both educational and commercial purposes. SUMO is mainly developed by the employees of the Institute of Transportation Systems at the German Aerospace Center and licensed under the Eclipse public license V2. Which means this license guarantee the freedom to share or modify the software as long as the contributor or distributor provides it as an open source.

It is available since 2001 and it allows modelling of intermodal traffic systems, this means that it allows modelling of road vehicles, public transport, and pedestrians. Also, included with SUMO is an extent of numerous supporting tools which intend to automate core tasks for the creation, the execution and

evaluation of traffic simulations, such as network import, route calculations, visualization, and emission calculation. In addition, SUMO can be enhanced with custom models and provides various Application Programming Interfaces (APIs) and features to enhance the realism and utility of the simulation.

2.4.2 SUMO application areas

SUMO is a software that stands out in its field, due to the sophistication and high realism simulations that have been contributing to numerous areas. SUMO has been used within several projects, research and even in real world scenarios. As we can see with the following list of some applications SUMO has had:

- Evaluate the performance of traffic lights, including the evaluation of modern algorithms up to the evaluation of weekly timing plans.
- Vehicle route choice has been investigated, including the development of new methods, the evaluation of eco-aware routing based on pollutant emission, and investigations on network-wide influences of autonomous route choice.
- SUMO is widely used by the V2X community for both, providing realistic vehicle traces, and for evaluating applications in an on-line loop with a network simulator.
- AI training of traffic light plans.
- Simulation of the traffic effects of autonomous vehicles and platoons.
- Simulation and validation of autonomous driving function in cooperation with other simulators.
- Simulation of parking traffic.
- Traffic safety and risk analysis.
- Calculation of emissions (noise and pollutants).

2.4.3 SUMO Installation & Set-up

The SUMO simulator is available on the three main Operating system (OS) environments, Linux, Windows and even Mac OS. From my experience both Windows and Linux version work well and stable. However, it is suggested to use it on Linux since is where the NS-3 simulator is most stable and where the experimental side of this thesis is done. Therefore, this section will provide a brief guide to the SUMO simulator and its main tools for Linux using the git repositories containing all the binary Linux versions of SUMO.

Similarly, to NS-3, the SUMO simulator also requires some prerequisites that are needed for some of SUMO tools such as cmake, python and g++ among others. The guide for installation and set-up can be found attached to the appendix B.

2.4.4 SUMO Creating and Running Mobility Scenarios

As this title suggest, this section is dedicated to the creation of a mobility scenario which is mandatory to the experimental perspective of this thesis. So, to create the mobility scenario we have to use the netedit tool included in SUMO. Which is a powerful tool that enables us to create a road network for our mobility scenario as well as vehicular traffic.

First step is to use this tool I've implemented and just run the following command from the utility make file on the terminal while in the /tése directory.

```
1 make run_netedit
```

As soon as netedit click in create a new network in the file menu as the figure bellow Fig. 2.5.

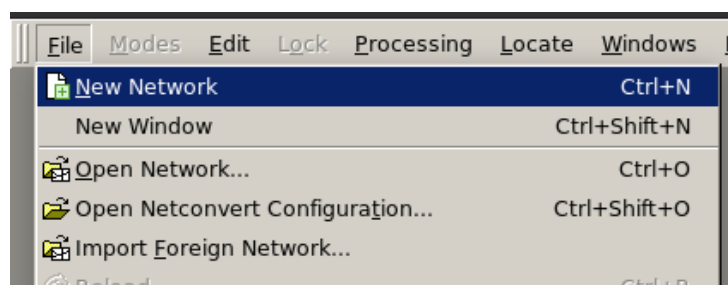


Figure 2.5: New network

Now with the that we have a new blank network assess the Network mode, this mode is where we have the tools to create the network as we can see in the Fig. 2.6 as well as their definition on the list below.

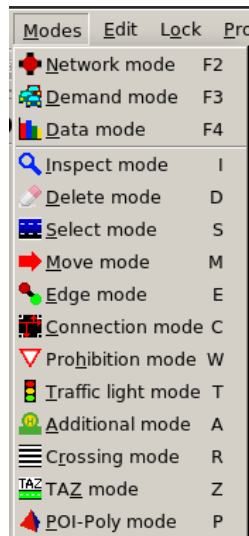


Figure 2.6: Network mode menu

- **Inspect mode:** This tool gives information about the elements of network by clicking them.
- **Delete mode:** This tool is to delete network elements.
- **Select mode:** This tool is to select one network element or multiple.
- **Move mode:** This tool is move or drag already created edges.
- **Edge mode:** This is the main tool, and its purpose is to create roads. Note that edges in SUMO are the roads.
- **Connection mode:** When two edges merge creating a junction this tool helps to define how each lane of a road connects with the other road lanes.
- **Prohibition mode:** This tool applies road signs policies on the edges or junction, such as stop or priority signs among others.
- **Traffic light mode:** When we have junction, this tool serves to apply traffic lights.
- **Additional mode:** This tool helps to add different elements such as bus stops and parking lots among others to the network.
- **Crossing mode:** This tool applies serves to apply cross walks.
- **Traffic assigned mode:** This tool serves to apply policies to a selected zone, for instance speed or type of vehicles permitted on the zone.
- **Point-of-interest mode:** This tool is to create building and points-of-interest.

Now we have all the tools required to create the network for the mobility scenario. To start we should use the edge mode to create the roads, change the settings speed and number of lanes to the desired Fig. 2.7(a), and also while we are in this mode select the edit menu and check the box named create and edge in the opposite direction to create a two way road Fig. 2.7(b).

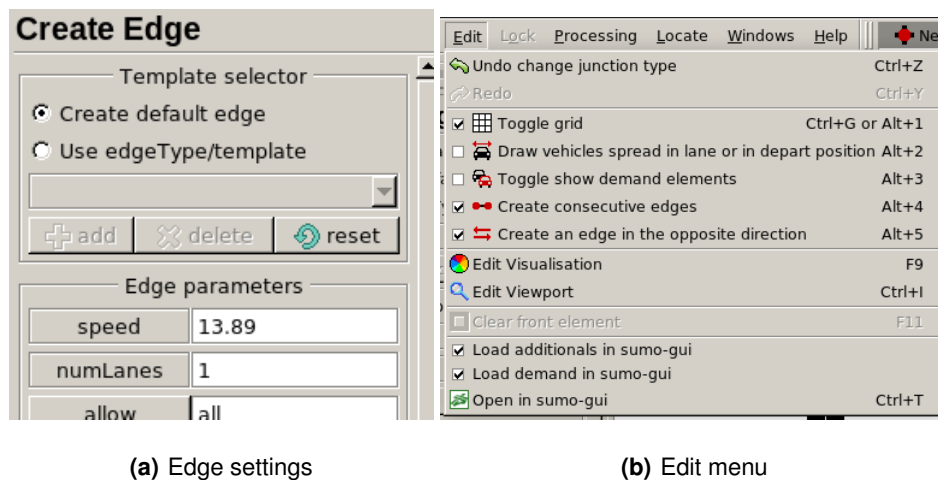


Figure 2.7: Edge creation

Once the edges are created they should look similar to the Fig. 2.8(a). Then, select the Connection mode to check if the connections of the junctions were successfully created and if not correct them to the how it is desired, and looking similar to the Fig. 2.8(b), were the red zones have the white lines representing the connections.

This are the bare minimum tasks to create a network a proceed to the demand mode which is where the traffic is created. However, it is possible to expand the complexity of the network with more elements using the tools describe above in the list. For instance the traffic lights, for that select the Traffic light mode and select the junctions where the lights are desired and click create to have something similar to the Fig. 2.8(c).

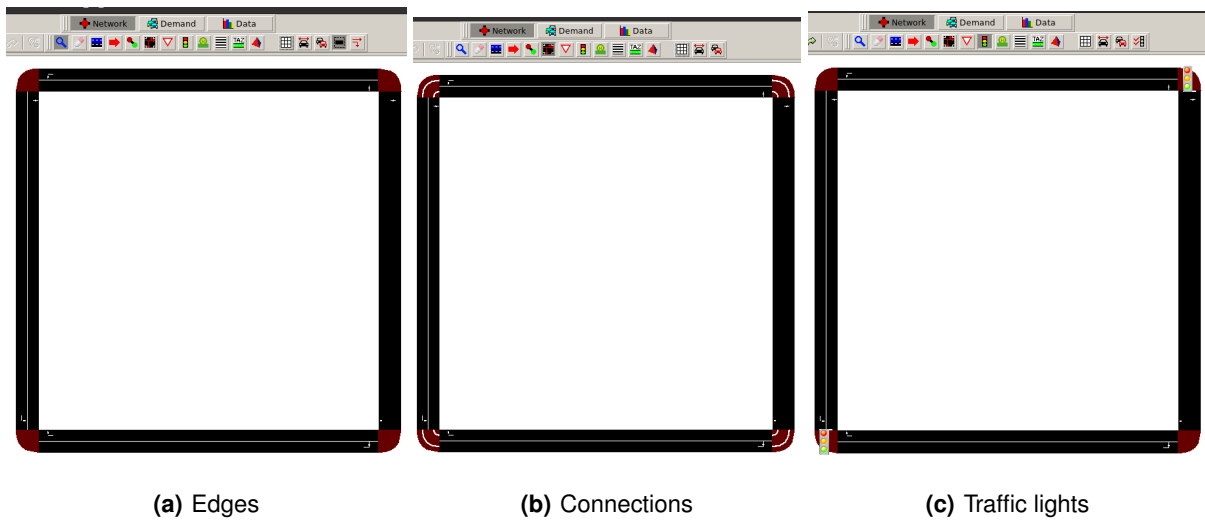


Figure 2.8: Network example

Now that the road network is done, save the network. Then it is time to fill the road network with the vehicular traffic. For that select the demand mode 2.9, which is similar to the network mode tools, this mode has a list of tools to create the traffic demand as we can see in the as well as a list of each new tool definition that the network mode did not have.

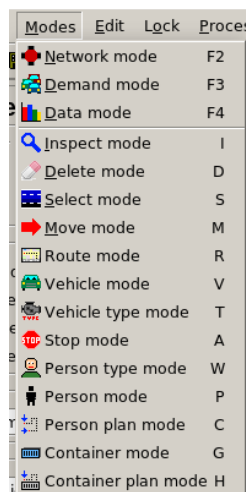


Figure 2.9: Demand mode menu

- **Route / Vehicle mode:** These tools are the main ones, the purpose of this tools are to create routes with traffic.
- **Vehicle type mode:** This tool serves to create different types of vehicles, for instance we can create different cars with different attributes such as acceleration, deceleration, max speed and even length.

- **Stop mode:** This tool serves to create stops along the routes, for instance the case of public transportation, where a bus needs to stop multiple times from the point of departure to the arrival.
- **Person / Person plan mode:** These tools serve to create pedestrian traffic and routes.
- **Container / Container plan mode:** These tools serve to create cargo traffic and routes.

Now we have all the tools required to create the vehicular traffic for the mobility scenario. To start we should use the vehicle mode, while in this mode we can see that it is possible to choose the type of route according to the Fig. 2.10(a) which is a trip route type. This one type and the flow (from-to), cover most of the use cases. The trip is as simple as simple as a single vehicle with a departure edge and arrival edge where we can define the departure time. The flow is the same as the trip, but for more than a single vehicle, so it has not only the departure edge and arrival edge but also the number of vehicles, begin and end time. This means that if we choose 10 vehicles with begin a 0s and end at 100s, we will have a car departing each 10 seconds until the simulation clock is at 100s. After creating new routes desired, and after clicking in the show all trips in the edit menu we should get something similar to the Fig. 2.10(b). Where the car icons are at the departure point of the route, with an orange line showing the path they will take to the arrival point. Note that, for this mobility scenario example, I've chosen the blue to represent the trip route, and yellow the flow route.

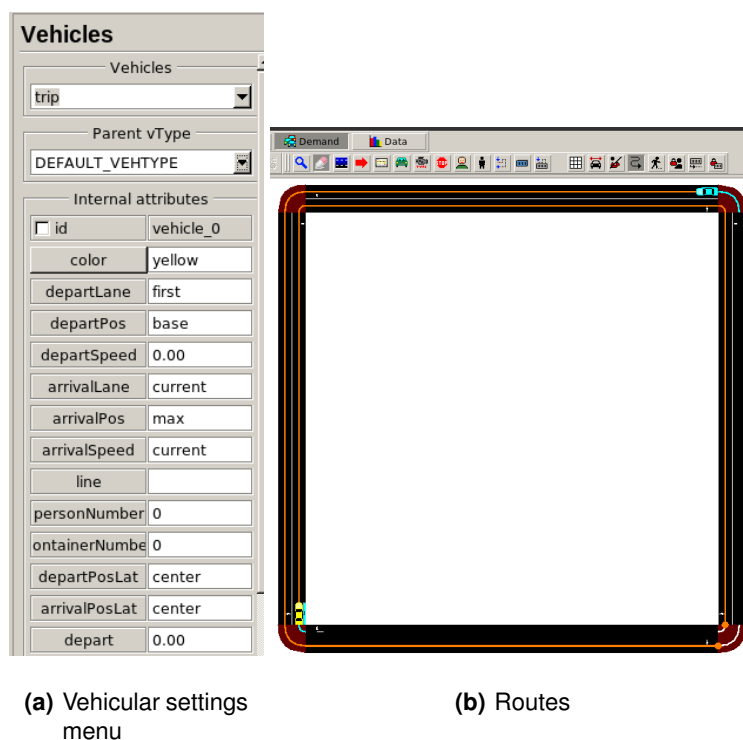


Figure 2.10: Vehicular settings and Routes

Now that the vehicular traffic demand is done, save it by clicking on the Demand Elements in the file menu, Fig. 2.11(a). Once that is done, we want to open both the vehicular traffic demand as well as the network in the sumo-gui which is the graphic user interface for the SUMO. to do that click on pen in sumo-gui, Fig. 2.11(b).

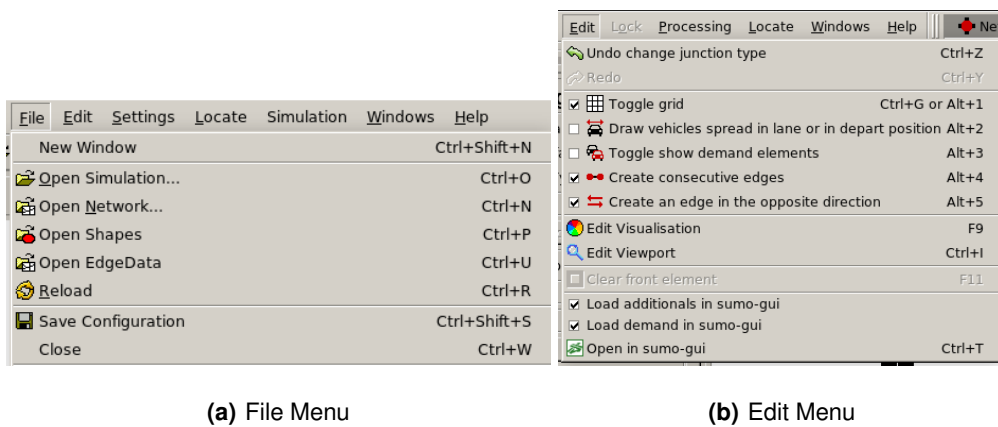


Figure 2.11: File and Edit menus

Now in the sumo-gui, we can play the scenario we have created, including the road network and the vehicular traffic. For that, increase the delay to about 100ms and press play to see the mobility scenario in real time similarly to Fig. 2.12

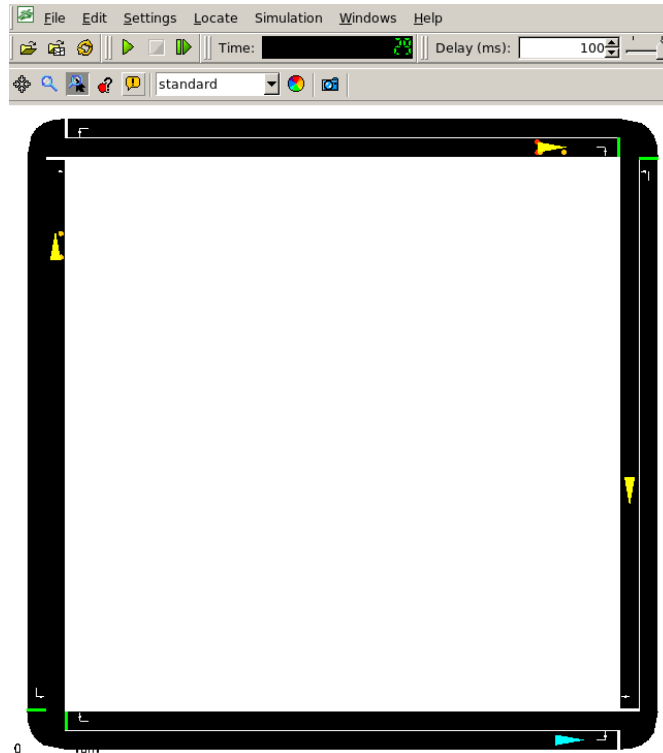


Figure 2.12: Mobility scenario in sumo-gui

Finally, the last step is to save the simulation and with that we will obtain important file the `mobilityScenario.sumocfg` which in itself will invoke the files created for the road network with the `mobilityScenario.net.xml` and the vehicular traffic with the `mobilityScenario.rou.xml`.

2.4.5 SUMO Documentation

As the SUMO simulator is not the primary focus of this thesis, but rather an important tool for the experimental aspect, this section is focused on the available resources for further information about the SUMO simulator.

- Documentation: <https://sumo.dlr.de/docs/index.html>
- User FAQ: <https://sumo.dlr.de/docs/FAQ.html>
- Mailing list: <https://www.eclipse.org/sumo/contact/>

3

SUMO&NS3 Coupling

Contents

3.1 SUMO&NS3-Coupling Architecture	33
3.2 Vehicular Mobility Scenario Creation	34
3.3 SUMO&NS3-Coupling Translation	39
3.4 NS3 Simulation	42
3.5 SUMO&NS3-Coupling Results	42

This chapter is related to the contribution of this thesis, the SUMO&NS3-Coupling tool software that is the main tool that couples this SUMO and NS3 software in order to produce useful data for study routing protocols such as the ones talked in chapter 2. So, in diving into the process of how to use the tool starting with the creation of a vehicular mobility scenario with the SUMO simulator until the moment the simulation output data is generated. All the material develop and generated with the SUMO&NS3-Coupling are provided through a GitLab repository for thesis in the following link: <https://gitlab.com/ist-ricardo-santos/performance-analysis-of-routing-protocols-on-vanets> since, most of the material used for this thesis are very extensible and cannot be attached directly to this document. However, portions of the materials used are going to be referenced in this section and some even attached to the appendix A.

3.1 SUMO&NS3-Coupling Architecture

The SUMO&NS3-Coupling tool allows anyone to combine two programs SUMO and NS3 in order to achieve realistic mobility scenarios for VANET routing protocols study. And does it by making a process of various steps done sequentially and seamlessly to the user. The process can be divided in four phases, listed below and as we can see in Fig. 3.1.

- **Vehicular Mobility Scenario Creation:** This is the initial phase, where we create the mobility scenarios. The SUMO simulator so that we have the mobility files needed to start with the SUMO&NS3-Coupling.
- **SUMO&NS3-Coupling Translation:** This is the Second phase, where the user input the mobility files to the SUMO&NS3-Coupling tool so that the program can do its job until outputting all the result. In this phase, the program takes the mobility files input, parses them, and automatically creates new files that the NS3 can ingest, it is like a translation process from SUMO to NS3.
- **NS3 Simulation:** This is the third phase, where the NS3 simulator takes the translated mobility files and starts the network traffic simulation with the different routing protocols such as AODV, OLSR and DSDV running in a WAVE environment. After all the runs with the different simulators output data is generated, and that data goes back to the SUMO&NS3-Coupling.
- **SUMO&NS3-Coupling Results:** This is the final phase, and this phase is where the SUMO&NS3-Coupling takes the output of the NS3 simulator which are mostly .csv data and automatically, does a statistical work on the data, also generates graphs to visualize better the Key Performance Indicatorss (KPIs) of the simulated scenarios.

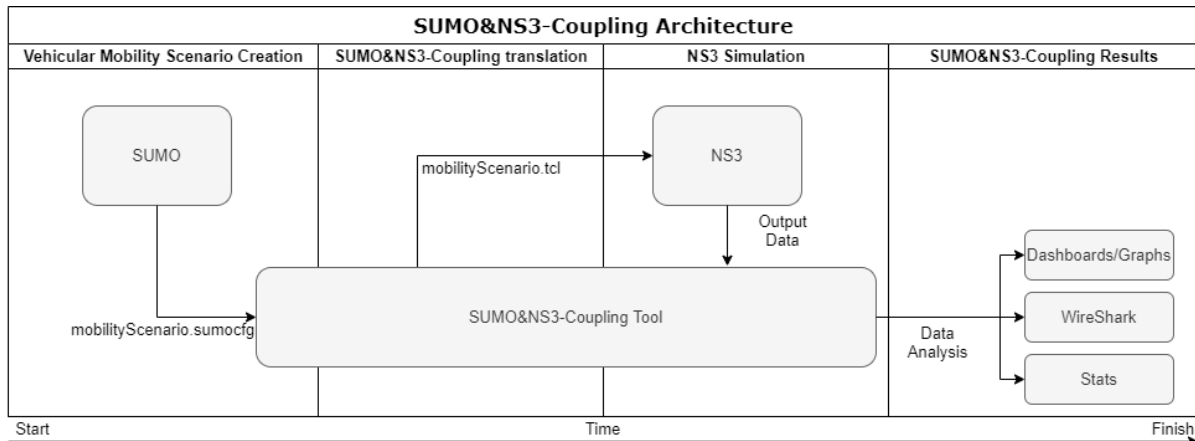


Figure 3.1: SUMO&NS3-Coupling Architecture

3.2 Vehicular Mobility Scenario Creation

As previous mention in the previous chapter, SUMO is an urban mobility simulation program which provides tools for creating vehicular mobility scenarios. Therefore, for the experimental environment we will use it to create the mobility scenarios using the Nedit tool included in SUMO.

From this point onwards, just follow the instructions from the section:2.4.4 - SUMO Creating and Running Mobility Scenarios on how to create a mobility scenario.

Once the mobility scenarios are completed, we can analyze the output files which are mainly .xml files and the most important one the .sumocfg, this is the file that will be served as input for the SUMO&NS3-Coupling tool. These files contain the information about the scenario, containing all the information about edges which are roads in SUMO, lanes, junctions, connections, traffic intensity, traffic type and more.

That being said, in order to continue the rest of the experimental process we need to have at least the following files listed below:

- **mobilityScenario.sumocfg**: This is the main file that will be used for the rest of the process. Is the configuration file which invokes the .xml files detaining all the detail about the scenario as we can see below.

Listing 3.1: Portion of mobilityScenario.sumocfg

```

1     ....
2     <input>
3         <net-file value="osm.net.xml"/>
4         <route-files value="{mobilityScenario.rou.xml}"/>

```



```

5     <additional-files value="osm.poly.xml"/>
6     </input>
7     ....

```

- **mobilityScenario.net.xml** This file has the information about vehicular road network, it contains all the details such as Identifiers (IDs), coordinates, speeds and lengths of the edges, lanes, junctions, and connections. As we can see bellow on the .xml code, the edge: gneE1 has a speed of 13.89m/s which is about 50km/h and the lane length which is 89.60m. We can also see an example of a junction and connection where the edge is involved:

Listing 3.2: Portions of mobilityScenario.net.xml

```

1     ....
2     <edge id="gneE1" from="gneJ1" to="gneJ2" priority="-1">
3         <lane id="gneE1_0" index="0" speed="13.89" length="89.60" shape="107.20,98.40
4             196.80,98.40"/>
5     </edge>
6     ....
7     <junction id="gneJ2" type="priority" x="200.00" y="100.00" incLanes="-gneE2_0 gneE1_0"
8         intLanes=":gneJ2_0_0 :gneJ2_1_0" shape="203.20,96.80 196.80,96.80 196.80,103.20
9         200.36,102.49 201.60,101.60 202.49,100.36 203.02,98.76">
10    </junction>
11    ....
12    <connection from="gneE1" to="gneE2" fromLane="0" toLane="0" via=":gneJ2_1_0" dir="r"
13        state="M"/>
14    ....

```

- **mobilityScenario.rou.xml or mobilityScenario.trips.xml:** This file has the information about the vehicular traffic on the road network, it contains the routes of each means of transport, such as cars, buses, trains, taxis, trucks, bicycles, motorcycles among others. In this file, we can have two types of routes, trips, and flows. The trips represent only one vehicle, and the flows represent multiple ones depending on the number attribute. For instance, on the listing bellow, we can see those two types, the trip, and the flow. Note that on the flow we have the number at 180, begin at 0, and end at 1800. This means that for 1800 seconds we will have 180 vehicles, which is the departure rate of 1 vehicle per 10 seconds.

Listing 3.3: Portions of mobilityScenario.rou.xml

```
1     ....
2     <routes>
3         <trip id="vehicle_0" depart="0.00" from="gneE0" to="gneE7" via="gneE8 -gneE11"/>
4         <flow id="flow_1" begin="0.00" from="gneE11" to="gneE10" via="gneE1 gneE5" end="1800.00"
           number="180"/>
5     </routes>
6     ....
```

Again, note that the scenario can have more files than the ones motioned depending on the complexity of the mobility scenario, for instance if it includes Point-of-interest (POI), buildings, public transportation among others and still proceed with experimental process. But again, only the listed above are mandatory since these are the ones that the SUMO&NS3-Coupling tool needs to generate the file that is needed to proceed to the NS3 phase with the purpose to utilize the vehicular mobility to study the various routing protocols.

For this thesis study, four different scenarios listed below were created with different characteristics examine the vehicular network routing protocols and V2X communications.

- **Urban Grid Scenario:** Aims to represent a scenario which is commonly has more node density with lower vehicular speeds.
- **Highway Scenario:** Aims to represent a scenario with higher vehicular speeds.
- **Country Grid Scenario:** Aims to represent a scenario usually has less node density with lower vehicular speeds.
- **Realistic Scenario:** Aims to represent a more realistic scenario, with a snapshot of Lisbon map similarly to a GPS application commonly used in smartphones.

3.2.1 Urban Grid Scenario

Aims to represent a scenario which is commonly has more node density with lower vehicular speeds.

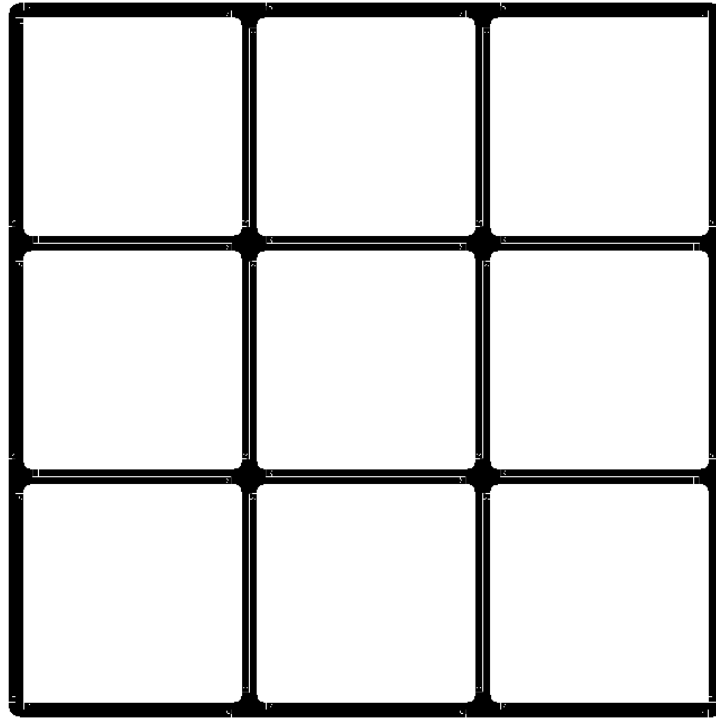


Figure 3.2: Grid map

Table 3.1: Grid Map characteristics

Characteristics	Unit
Node number	50
Road Length	4.17 km
Average Speed	33.2 Km/h
Simulation time	219s

3.2.2 Highway Scenario

Aims to represent a scenario with higher vehicular speeds.



Figure 3.3: Highway map

Table 3.2: Highway Map characteristics

Characteristics	Unit
Node number	40
Road Length	6 km
Average Speed	90 Km/h
Simulation time	100s

3.2.3 Country Grid Scenario

Aims to represent a scenario usually has less node density with lower vehicular speeds.

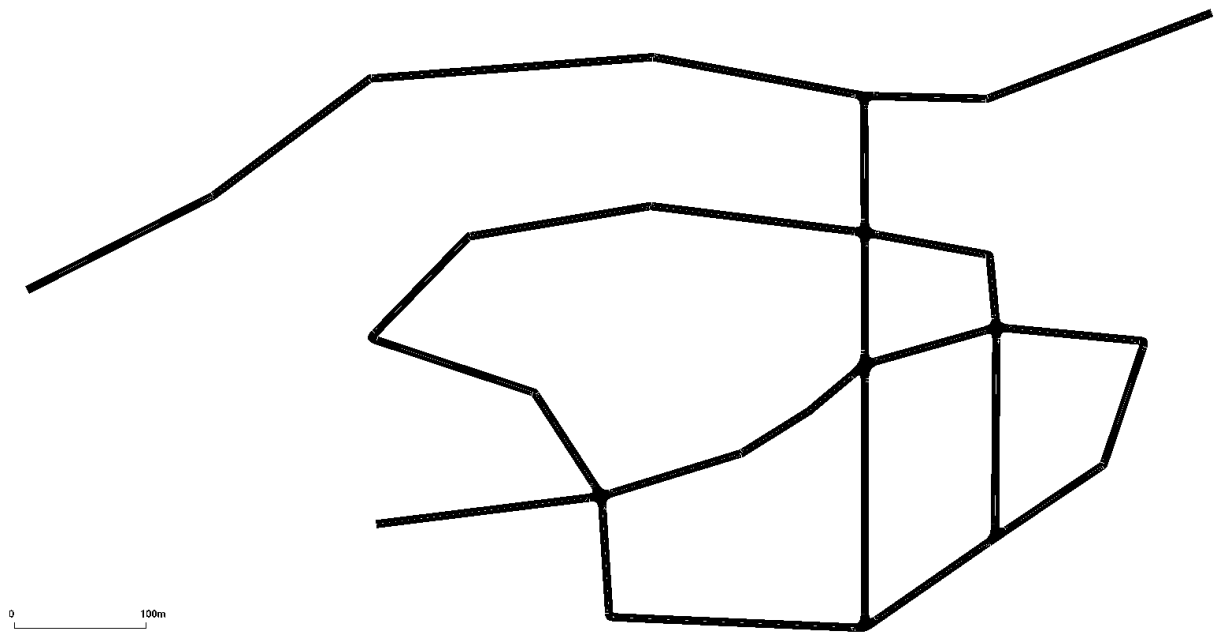


Figure 3.4: Country map

Table 3.3: Country Map characteristics

Characteristics	Unit
Node number	21
Road Length	7 km
Average Speed	49 Km/h
Simulation time	174s

3.2.4 Realistic Scenario

Aims to represent a more realistic scenario, with a snapshot of Lisbon map similarly to a GPS application commonly used in smartphones.



Figure 3.5: Lisbon map

Table 3.4: Lisbon Map characteristics

Characteristics	Unit
Node number	62
Road Length	67km
Average Speed	19 Km/h
Simulation time	656s

3.3 SUMO&NS3-Coupling Translation

This is the Second phase, where the user input the mobility files to the SUMO&NS3-Coupling tool so that the program can do its job until outputting all the result. In this phase, the program takes the mobility files input, parses them, and automatically creates new files that the NS3 can ingest, it is like a translation process from SUMO to NS3.

It is in this phase where we run SUMO&NS3-Coupling program. However, as previously said to continue with the simulation process is necessary to have the right files, more in particular the `mobility Scenario.sumocfg` which is the input file.

Being said that, to start the SUMO&NS3-Coupling program, type the following command on `/tese` directory of the material provided:

```
1 source Simulation.sh SUMO/mobilityScenario/ mobilityScenario.sumocfg
```

After typing the command, the simulation can take a while depending on the complexity of the mobility scenario, for instance the Realistic Scenario took four hours on a modern high-end laptop. Again, it could not be simpler, it just requires typing a single command and then the user can leave the computer for hours to come back with all the results.

Then the SUMO&NS3-Coupling program will initiate the first action, converting the `mobilityScenario.sumocfg` into a trace file `mobilityScenario.xml`. This file is a file sorted by time in seconds with the of the vehicles' information or flows in the given second as we can see below:

Listing 3.4: Portion of `mobilityScenario.xml`

```
1 ....
2 <timestep time="0.00">
3   <vehicle id="flow_0.0" x="-1.60" y="91.70" angle="180.00" type="DEFAULT_VEHTYPE" speed="0.00"
4     pos="5.10" lane="gneE7_0" slope="0.00"/>
5   <vehicle id="flow_1.0" x="12.30" y="-1.60" angle="90.00" type="DEFAULT_VEHTYPE" speed="0.00"
6     pos="5.10" lane="gneE11_0" slope="0.00"/>
7   <vehicle id="vehicle_0" x="8.30" y="98.40" angle="90.00" type="DEFAULT_VEHTYPE" speed="0.00"
8     pos="5.10" lane="gneE0_0" slope="0.00"/>
9 </timestep>
10 <timestep time="1.00">
11   <vehicle id="flow_0.0" x="-1.60" y="89.72" angle="180.00" type="DEFAULT_VEHTYPE" speed="1.98"
12     pos="7.08" lane="gneE7_0" slope="0.00"/>
13   <vehicle id="flow_1.0" x="14.90" y="-1.60" angle="90.00" type="DEFAULT_VEHTYPE" speed="2.60"
14     pos="7.70" lane="gneE11_0" slope="0.00"/>
15   <vehicle id="vehicle_0" x="10.05" y="98.40" angle="90.00" type="DEFAULT_VEHTYPE" speed="1.75"
16     pos="6.85" lane="gneE0_0" slope="0.00"/>
17 </timestep>
18 ...
```

However, this `mobilityScenario.xml` file is not yet compatible with the NS3. So, the second action of the SUMO&NS3-Coupling program is to convert the `mobilityScenario.xml` into a `mobilityScenario.tcl`. This file narrows down the `mobilityScenario.xml` to the essential information about the mobility of the vehicles for the NS3 simulator, by running the `traceExporter.py` python code resulting in a file sorted again by time in seconds with x,y,z axis information of each vehicle in the given second as we can see below. That way the NS3 can parse the file in order while changing the node positions in the

VANET simulation.

Listing 3.5: Portion of mobilityScenario.tcl

```
1  $node_(0) set X_ -1.6
2  $node_(0) set Y_ 91.7
3  $node_(0) set Z_ 0
4  $ns_ at 0.0 "$node_(0) setdest -1.6 91.7 0.00"
5  $node_(1) set X_ 12.3
6  $node_(1) set Y_ -1.6
7  $node_(1) set Z_ 0
8  $ns_ at 0.0 "$node_(1) setdest 12.3 -1.6 0.00"
9  $node_(2) set X_ 8.3
10 $node_(2) set Y_ 98.4
11 $node_(2) set Z_ 0
12 $ns_ at 0.0 "$node_(2) setdest 8.3 98.4 0.00"
13 $ns_ at 1.0 "$node_(0) setdest -1.6 89.72 1.98"
14 $ns_ at 1.0 "$node_(1) setdest 14.9 -1.6 2.60"
15 $ns_ at 1.0 "$node_(2) setdest 10.05 98.4 1.75"
16 $ns_ at 2.0 "$node_(0) setdest -1.6 86.05 3.67"
17 $ns_ at 2.0 "$node_(1) setdest 19.69 -1.6 4.79"
18 $ns_ at 2.0 "$node_(2) setdest 14.02 98.4 3.97"
19 $ns_ at 3.0 "$node_(0) setdest -1.6 81.06 5.00"
20 $ns_ at 3.0 "$node_(1) setdest 26.16 -1.6 6.47"
21 $ns_ at 3.0 "$node_(2) setdest 19.67 98.4 5.65"
22 $ns_ at 4.0 "$node_(0) setdest -1.6 73.55 7.51"
23 $ns_ at 4.0 "$node_(1) setdest 34.38 -1.6 8.22"
24 $ns_ at 4.0 "$node_(2) setdest 27.38 98.4 7.71"
25 ...
```

Now that the `mobilityScenario.tcl` file is ready to enter the NS3 simulation, but to run a NS3 simulation there are other input parameters that are needed such as time of simulation, number of nodes, protocols, among others. So, to avoid making the user do all that manually, which can be complex and time consuming depending on the complexity of the `mobilityScenario.tcl`, the `TclParser.py` will handle the time of simulation and number of nodes. Then the SUMO&NS3-Coupling program will start to invoke multiple runs of the mobility scenario in NS3, which will be explained better in the next section.

3.4 NS3 Simulation

This is the third phase, where the SUMO&NS3-Coupling tool sets up NS3 simulation runs for the mobility scenario chosen. The way the SUMO&NS3-Coupling is programmed makes it run three simulations for each protocol of the `vanet-routing-compare.cc` which is a code inspired in the `manet-routing-compare.cc` with that takes advantage of multiple modules, and one of them is the Ns2Mobility helper, imported from the predecessor of the NS3, the NS2. This module is responsible for taking the mobility trace `mobilityScenario.tcl`, returning the position of each node every second of the simulation. Another module used which and the main one is the WAVE helper, which is responsible for performing the WAVE protocol. Also, many other modules were used, for instance the routing protocols helpers for the OLSR, AODV and DSDV. Since this code has a considerable number of lines, all these modules can be seen in the git repository provided in the introduction of this chapter.

With that said, in the three different simulations runs, each will have a different VANET routing protocols in the following order and sequentially, OLSR, AODV and DSDV. After each run the SUMO&NS3-Coupling will create a new directory on the mobility scenario directory called `Stats`, the Again, the user do not need to worry about setting up any of this, the tool will handle everything, and this process can take a while depending on the complexity, particularly with the increase of the number of nodes is the one that impacts that aspect the heaviest so we need to be careful setting it up according to our machine's computational power.

3.5 SUMO&NS3-Coupling Results

As said previously on the NS3 Simulation section, after each simulation the SUMO&NS3-Coupling creates a new directory called `Stats`. This is the directory where every possible outcome of the SUMO&NS3-Coupling tool is stored after each simulation run of every routing protocol, starting with the OLSR simulations first, then AODV and finally DSDV. Therefore, after each simulation is completed a new directory with the name of the protocol of whose simulation run has just finished where all the data files associated to that run are stored. Those files can be the following:

- **mobilityScenario.mobility_NetAnim.xml**: This file, contains the information about each node ID, IP in the network, geographical position as well as the messages sent and received. And this file that can be used by a NS3 tool which is the NetAnim tool, which is a tool that provides a graphical view of the behavior of the VANET mobility scenario. It is possible to see the movement of each node, and the messages being sent or received in that moment of the simulation as we can see below in the Fig. 3.6, as a reminder this is only a snapshot of the simulation in a particular second of the entire simulation, since is the only way to show it this thesis work.


```

9      +15s,8.192,16,10,protocol,7.5,80,665,8.3125,6,5,0.833333,0.925,0.882353,0.5,0.45,0.45,0.375 ...
10     ...
11     +90s,17.92,35,10,protocol,7.5,400,6031,15.0775,3060,2910,0.95098,0.886259,0.879155, ...
12     +91s,18.432,36,10,protocol,7.5,400,6071,15.1775,3060,2938,0.960131,0.887688,0.881388, ...
13     +92s,18.432,36,10,protocol,7.5,400,6210,15.525,3102,2974,0.958736,0.904505,0.884867, ...
14     ...

```

- **mobilityScenario.mobility_routing_table:** As the name suggests this file contains the routing tables of each node.
- **mobilityScenario.mobility.FlowMonior.xml:** This file contains information on the flows of the messages between nodes, such as delays, packets sent and received, bytes sent and received and jitter among others.
- **mobilityScenario.log & mobilityScenario.mob:** These files contains information about the geographical position of the nodes as well as their IDs and velocity. At the moment theses files are not used, but eventually they can be in the case of future work on the SUMO&NS3-Coupling tool.
- **mobilityScenario.pcaps:** These files contains information about every packet sent or received about each node in the VANET network simulations. The .pcaps can be opened with the Wireshark software which is already included with the installation of the NS-3. In the Fig. 3.7 below there is a example of the packet 64178 from a node, preforming a route request using the AODV routing protocol.

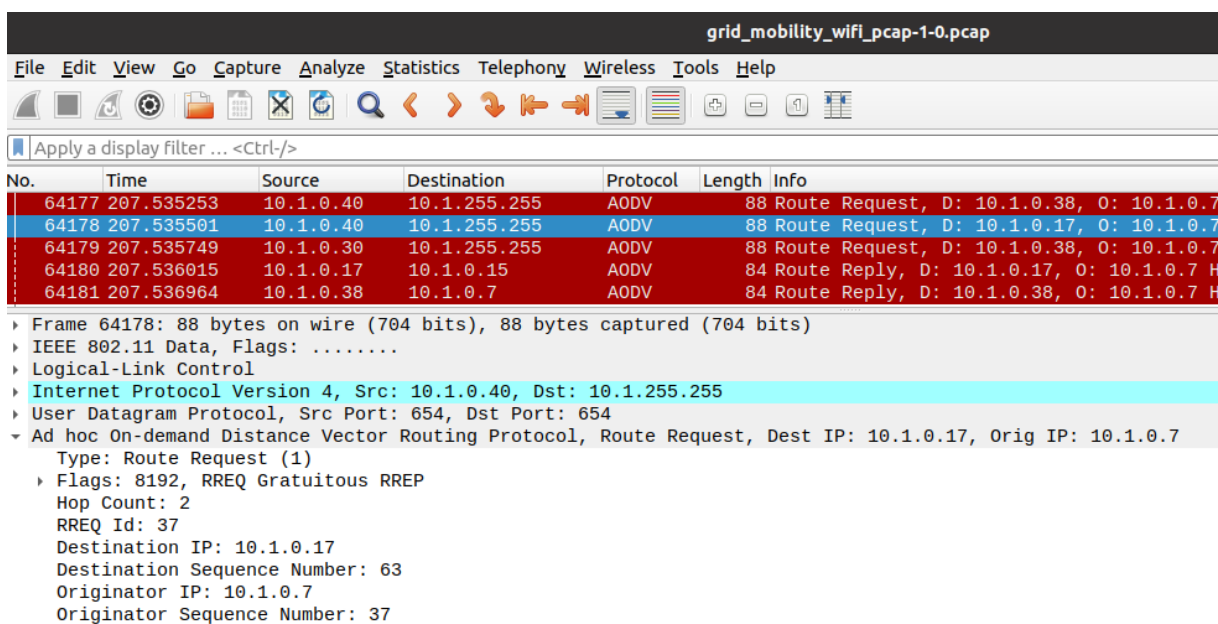


Figure 3.7: Example of AODV packet

After the SUMO&NS3-Coupling tool has run the three runs of each routing protocol, it will parse each one of the `mobilityScenario_mobility_stats.csv` files and plot graphs in order to visualize better the performance comparison between protocols. And does it by using the `gnuplot` which is a command-line driven graphing utility for Linux. In addition, there is a python code `flowmon-parse-results.py` used to extract statistical information from the `mobilityScenario_mobility_FlowMonior.xml`.

The SUMO&NS3-Coupling tool plots 20 graphs with different metrics per routing protocol used in each mobility scenario. However only a portion of those metrics are useful for performance comparison [18–20], those are the receive rate resembling the goodput in packets per second and the overhead caused by each protocol represented by the portion of every packet from the routing protocol sent by the total of packets sent in the network. As we can see in from the line of code in the `vanet-routing-compare.cc` code below. The overhead metric is the main metric of comparison since this metric unveils the extra bandwidth consumed by overhead to deliver data traffic.

```
1      mac-phy-oh = (total-phy-bytes - total-app-bytes) / total-phy-bytes
```

Also, to provide more context to the analysis, the average speed in meters per second and the number of running vehicles in a particular second of the simulation. With that said, the following sections are dedicated to each mobility scenario where we can see the output with the comparisons of the SUMO&NS3-Coupling tool to each routing protocol.

3.5.1 Urban Grid Scenario

The urban scenario, as said previously aims to represent a scenario which commonly has more node density in this case reaching a peak of 34 nodes for a couple of seconds as we can see in the Fig. 3.8, also every node is within a square kilometer since every edge of this grid is 100m, therefore making it 300x300m Fig. 3.2. In this grid mobility scenario, the nodes are circulating at speeds ranging between 6-12 meters per second which is about 20-40 kilometers per hour Fig. 3.9. In these circumstances, we can see that in from the graph in the figures 3.10,3.11 and table 3.5 that the receive rate is similar between protocols, however the OLSR protocol has a slightly advantage of 2.6% over the AODV and 4.4% over the DSDV. On top of that advantage, the overhead caused by the OLSR protocol is lower compared to AODV and DSDV with a difference of -33.7% and -10.2% respectively. This means that, for the OLSR protocol uses less packets, 39.6%, to maintain all the communication necessary between nodes. Having said that, is clear that the OLSR routing protocol performs better than the other protocols in the urban mobility scenario.

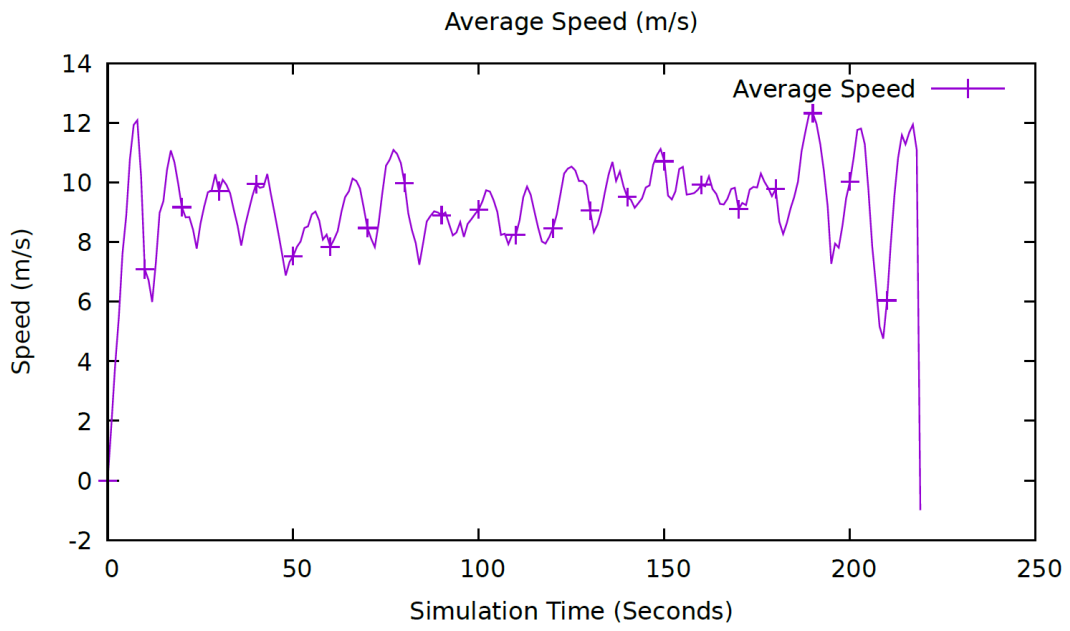


Figure 3.8: Average Speed Grid Scenario

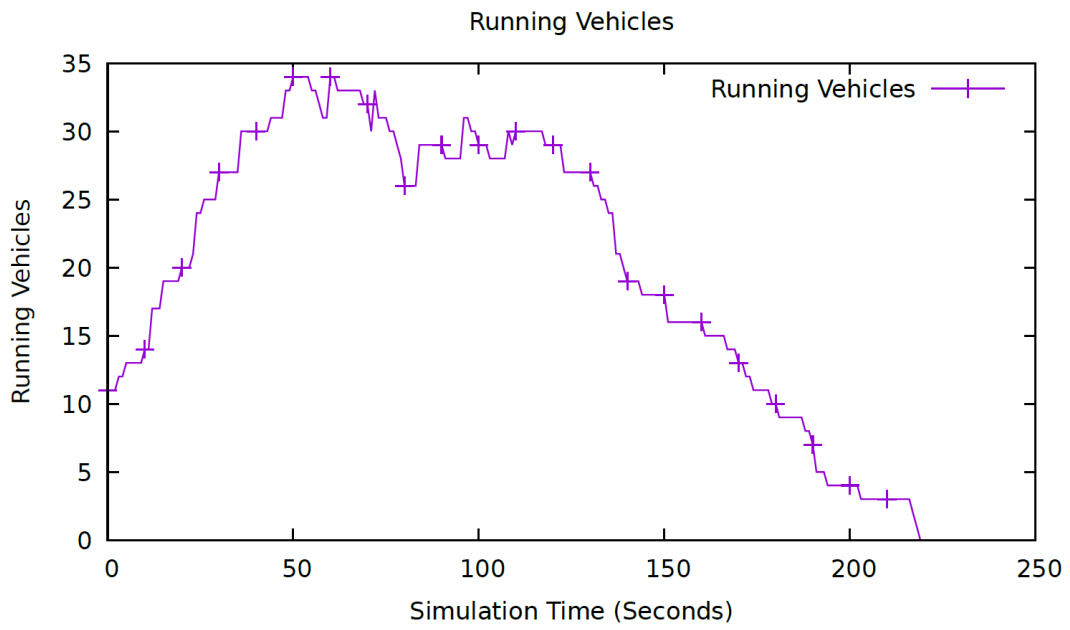


Figure 3.9: Running Vehicles Grid Scenario

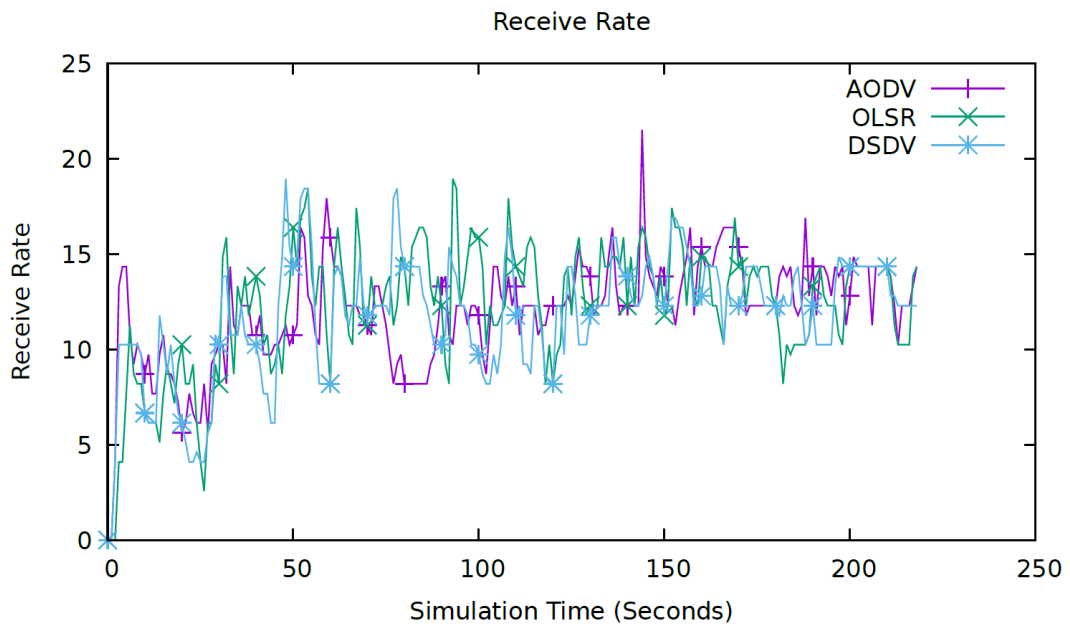


Figure 3.10: Receive Rate Grid Scenario

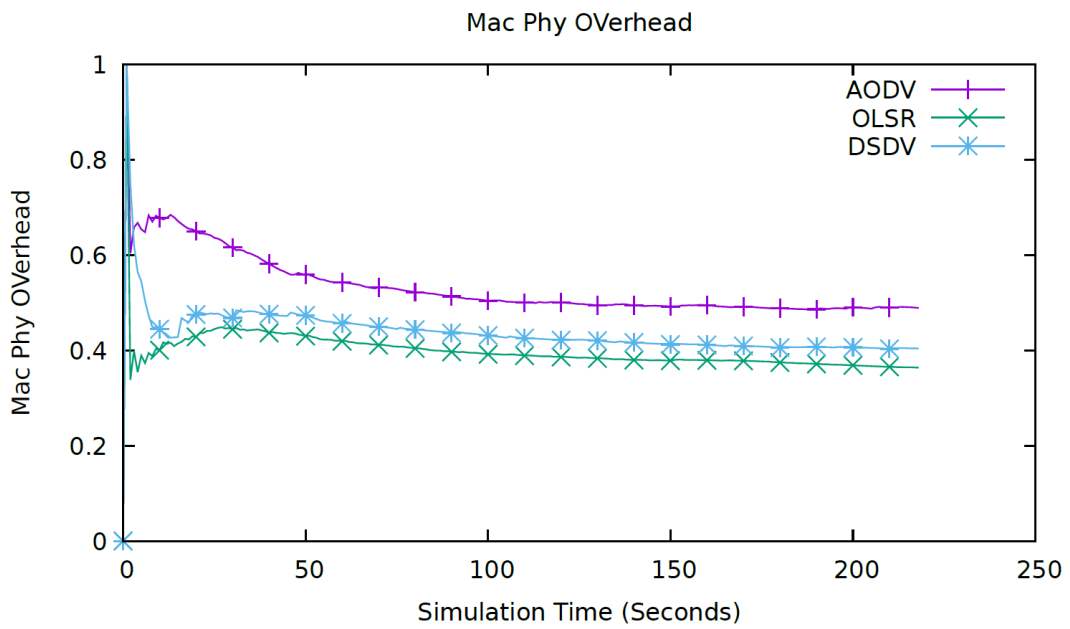


Figure 3.11: Overhead Grid Scenario

Table 3.5: Metric Averages of Grid Scenario

Protocol	Receive Rate	Overhead
AODV	12.109	0.530
OLSR	12.433	0.396
DSDV	11.906	0.437

3.5.2 Highway Scenario

The highway scenario, as said previously aims to represent a scenario which commonly has high node mobility speeds, in this case they are circulating with an average speed of 25 m/s which is about 90 kilometers per hour as we can see in the Fig. 3.12 and table 3.2. Also, the number of vehicles running at a certain second in the simulation reaches a peak of 34 nodes at the second 57 as we can see in the Fig. 3.13, also every node is within an area of 1000x200m Fig. 3.3. In these circumstances, we can see that in from the graph in the figures 3.14,3.15 and table 3.6 that the receive rate is similar between protocols. Also, we can see an interesting phenomenon, in the receiving rate graph, that during the time period between the 20 seconds and 50 seconds the rate is very low when comparing the same time period in the running vehicles graph which is high. This can be counter intuitive, however, the reason is that many vehicles are far away from each other and not making as many communications. In addition, we can verify that the DSDV protocol has an advantage of 2.3% over the AODV and 9.3% over the OLSR. On top of that, the overhead caused by the DSDV protocol is slightly lower compared to AODV and higher compared to OLSR with a difference of -1.0% and 1.9% respectively. Having said that, it is clear that the DSDV routing protocol performs better than the other protocols in the highway mobility scenario.

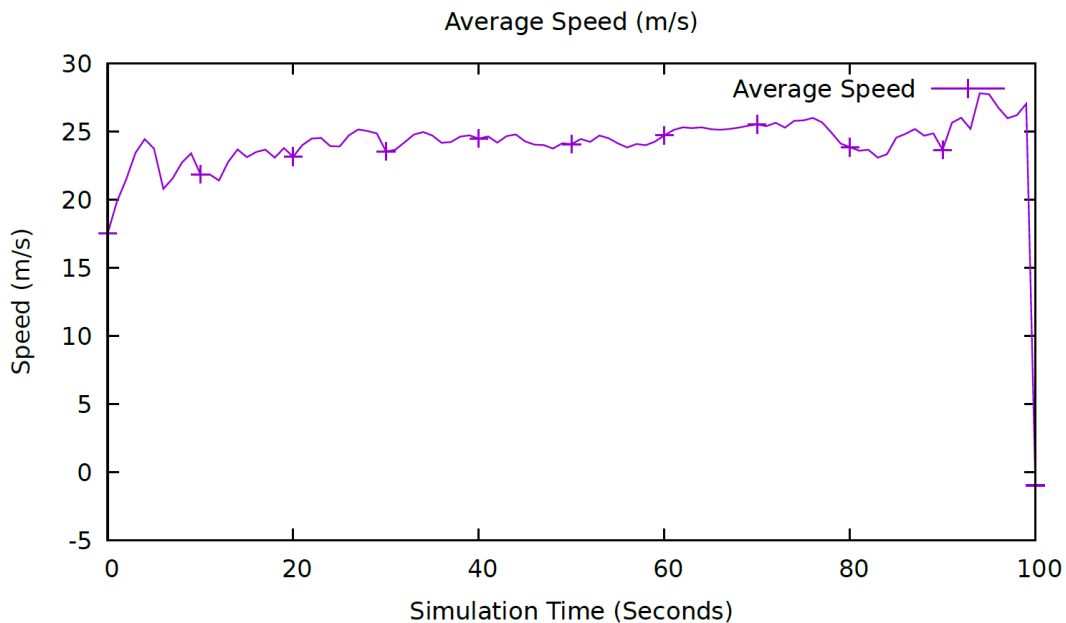


Figure 3.12: Average Speed Highway Scenario

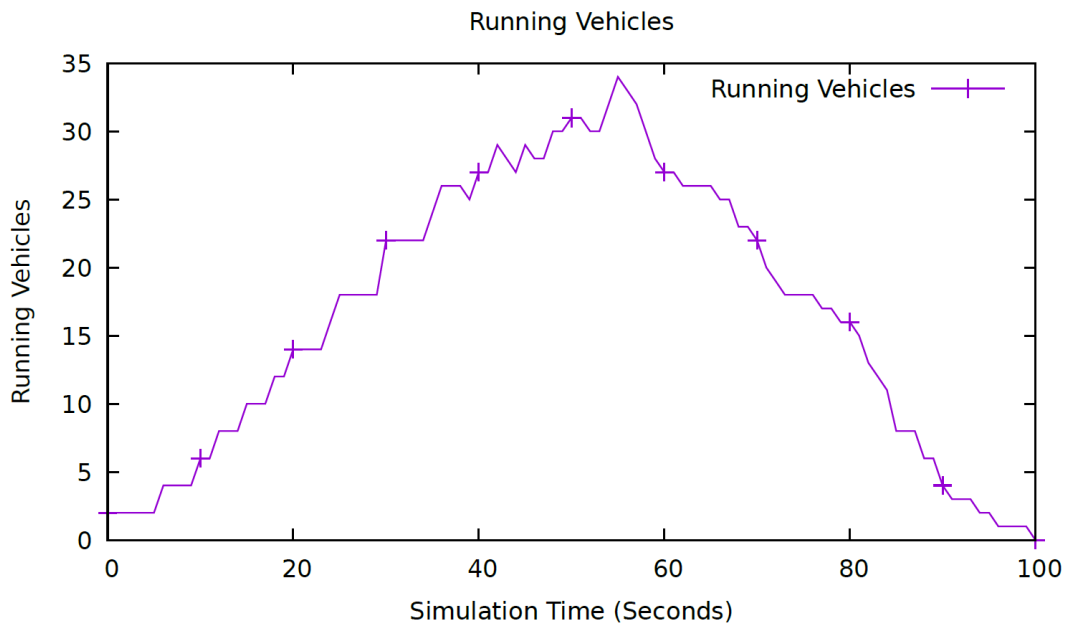


Figure 3.13: Running Vehicles Highway Scenario

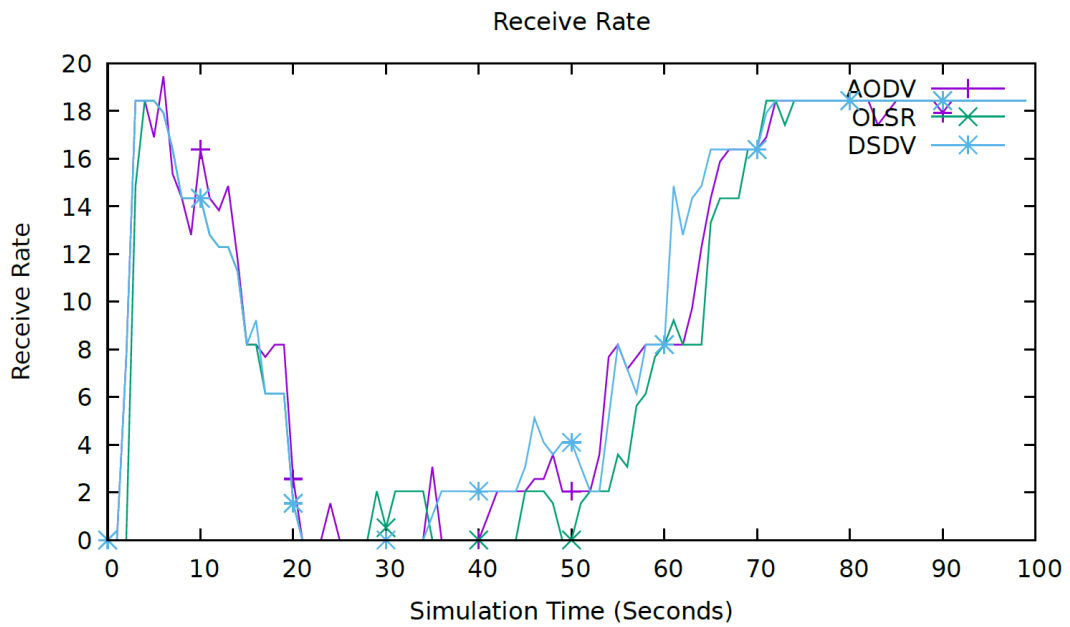


Figure 3.14: Receive Rate Highway Scenario

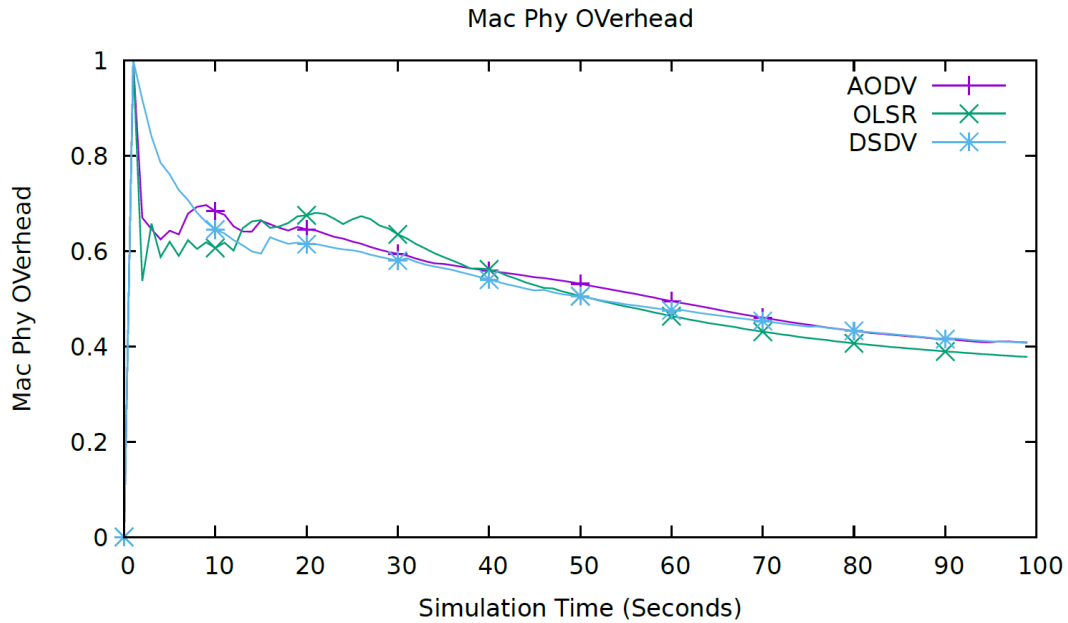


Figure 3.15: Overhead Highway Scenario

Table 3.6: Metric Averages of Highway Scenario

Protocol	Receive Rate	Overhead
AODV	9.923	0.530
OLSR	9.308	0.516
DSDV	10.173	0.525

3.5.3 Country Scenario

The country scenario, as said previously aims to represent a scenario which commonly has low node density in this case reaching a peak of 34 nodes at the second 54 in the simulation as we can see in the Fig. 3.17. However, unlike the grid scenario the country scenario has an area much bigger of 900x300m Fig. 3.4, therefore less dense. In this country mobility scenario, the nodes are circulating at speeds ranging around the 15 meters per second which is about 54 kilometers per hour Fig. 3.16. In these circumstances, we can see that in from the graph in the figures 3.18,3.19 and table 3.7 that the receive rate of the DSDV protocol has a slightly advantage of 5.2% over the AODV and 19.2% over the OLSR. However, the overhead caused by the DSDV protocol is the highest of the three. If we consider the AODV protocol in this metric, the protocol compared to OLSR and DSDV has a difference of -6.6% and -23.3% respectively. A difference of 23.3% is a very considerable one, and since the advantage of DSDV over the AODV in the metric rate is only 5.2%, it is safe to assume that the AODV protocol is the most suited protocol in the country scenario for most use cases. However, if the goodput is extremely

needed over other metrics, the DSDV is the best option.

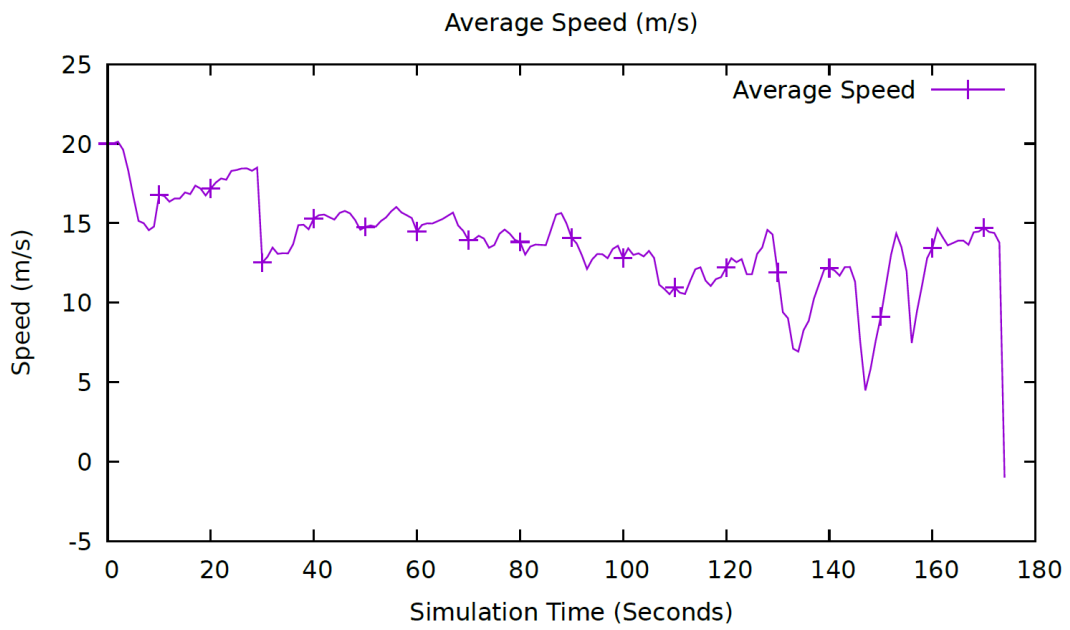


Figure 3.16: Average Speed Country Scenario

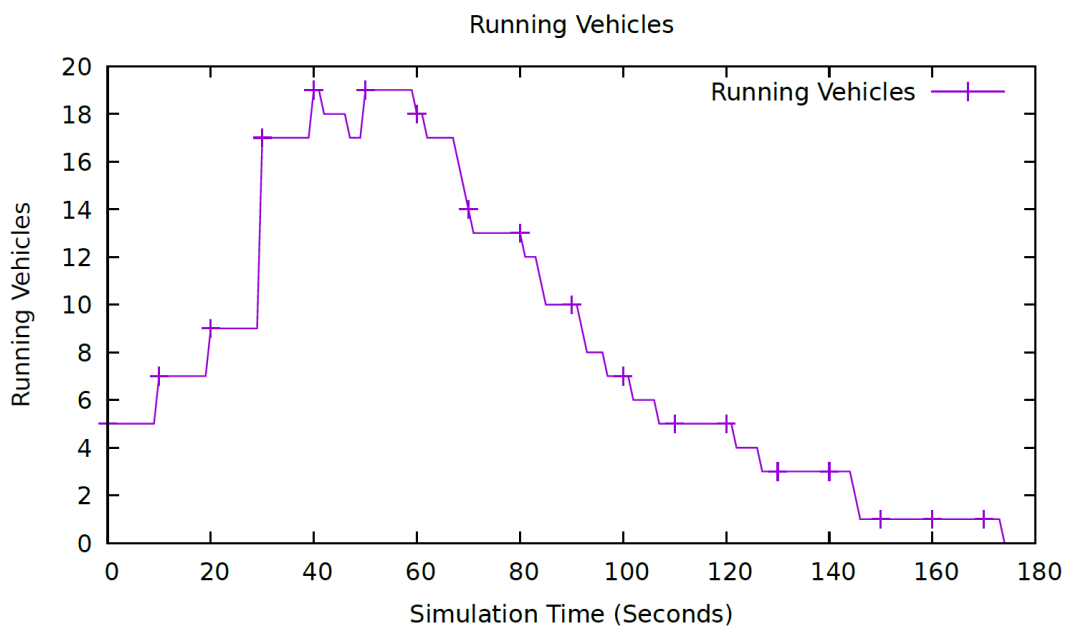


Figure 3.17: Running Vehicles Country Scenario

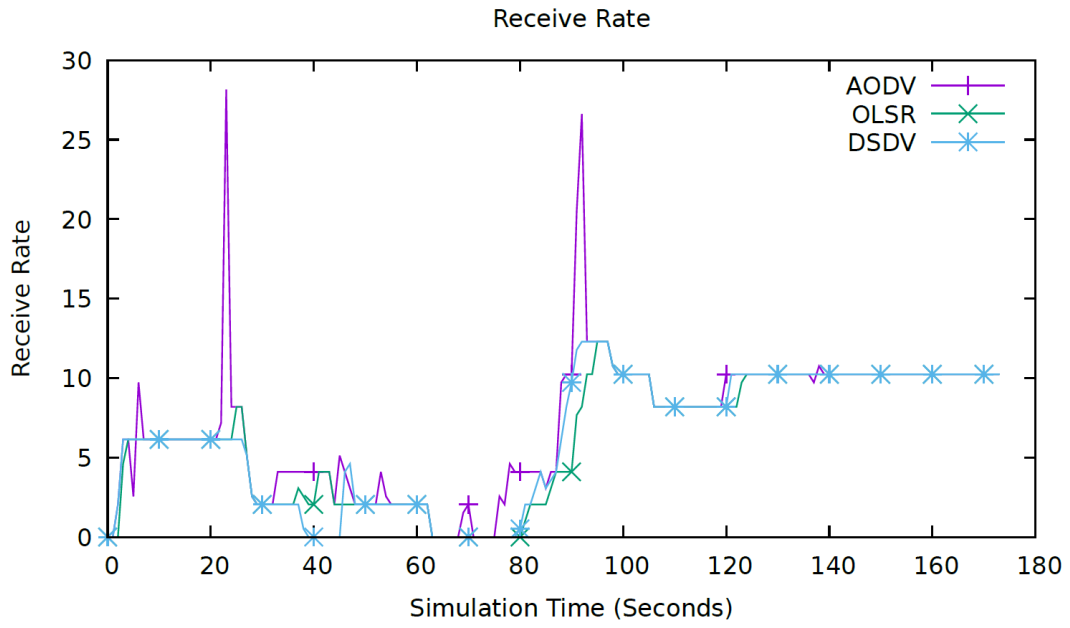


Figure 3.18: Receive Rate Country Scenario

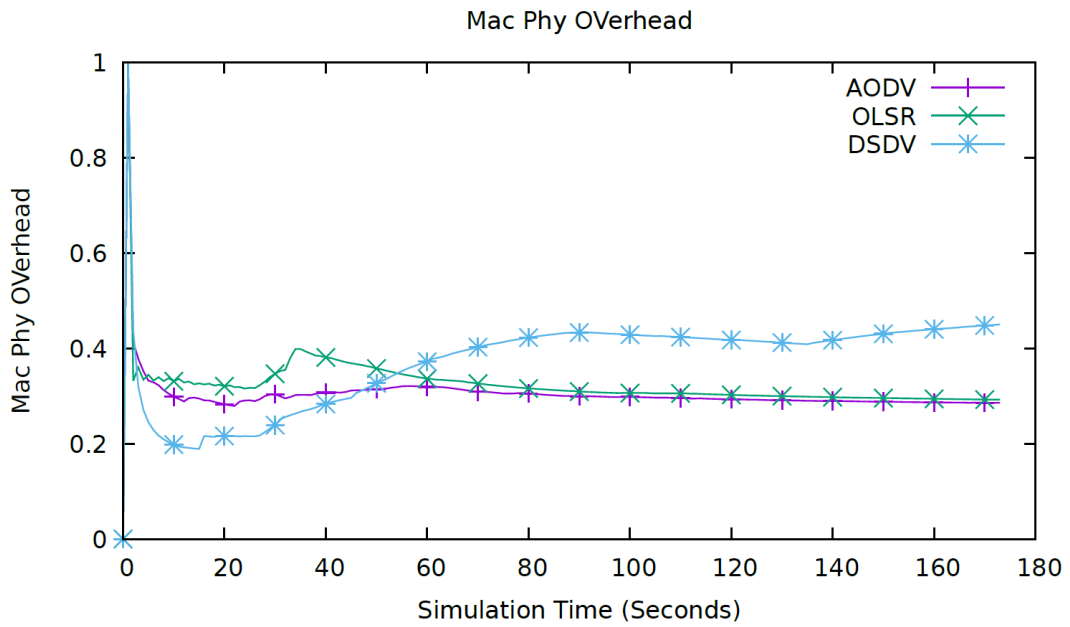


Figure 3.19: Overhead Country Scenario

Table 3.7: Metric Averages of Country Scenario

Protocol	Receive Rate	Overhead
AODV	7.039	0.300
OLSR	6.309	0.320
DSDV	7.403	0.370

3.5.4 Lisbon Scenario

The Lisbon scenario, as said previously aims to represent a a more realistic scenario, with a snapshot of Lisbon map. This portion of the Lisbon city is called the "Baixa de Lisboa", meaning the lowest zone of Lisbon. And, in this scenario the number of vehicles running at a certain second in the simulation reaches a peak of 58 nodes for a couple of seconds as we can see in the Fig. 3.22, also every node is within an area of 1500x700m Fig. 3.5. In this grid mobility scenario, the nodes are circulating at speeds ranging between 0-16 meters per second with an average of 5 m/s which is about 19 kilometers per hour as we can see in the graph of the Fig. 3.9 and in the table 3.4. In these circumstances, we can see that in from the graph in the figures 3.22,3.23 and table 3.8 that the receive rate is similar between protocols, however the AODV protocol has a slightly advantage of 6.5% over the OLSR and 4.9% over the DSDV. In terms of overhead, the OLSR protocol has the lowest when compared to AODV and DSDV with a difference of -14.9% and -31.1% respectively. Having said that, despite the OLSR performing slightly worse in terms of goodput, overall is clear that the OLSR routing protocol performs better than the other protocols when considering the overhead caused by the other protocols in the Lisbon mobility scenario. Making the OLSR the best option for this scenario.

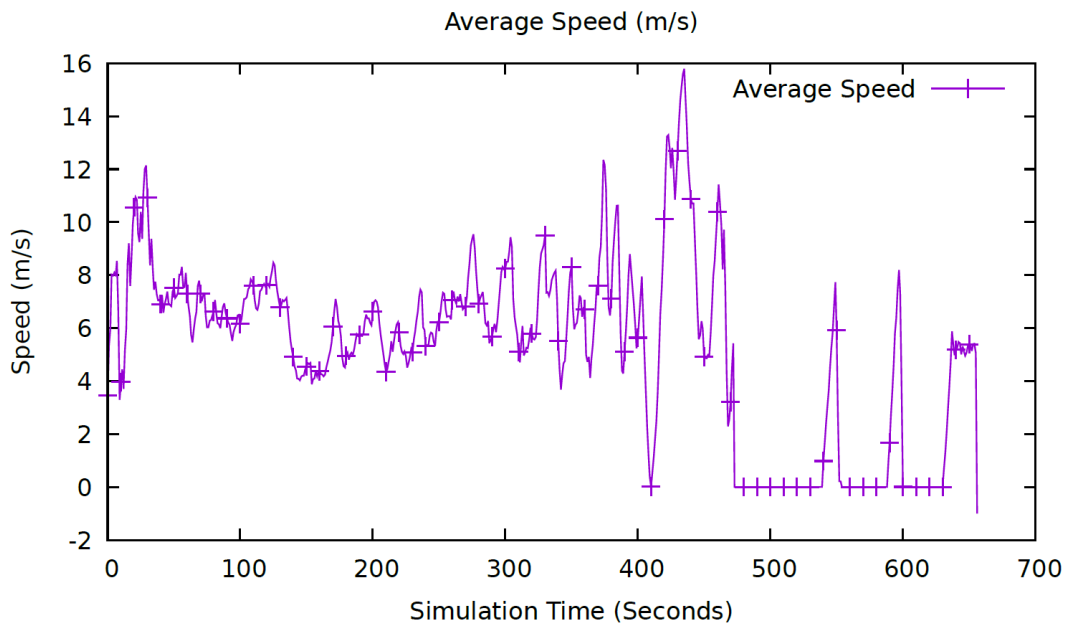


Figure 3.20: Average Speed Lisbon Scenario

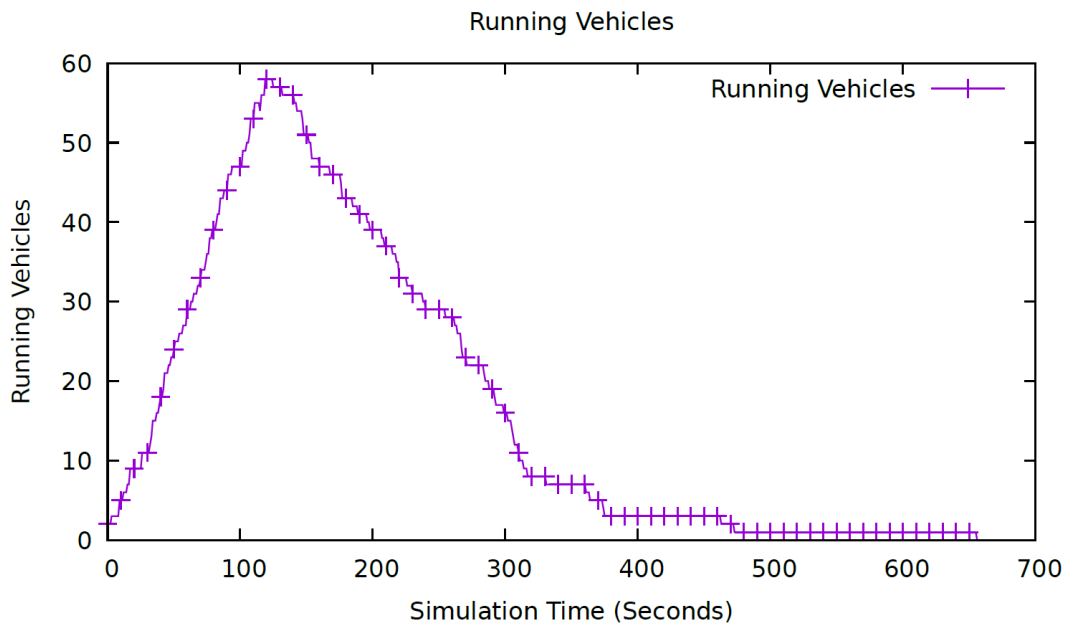


Figure 3.21: Running Vehicles Lisbon Scenario

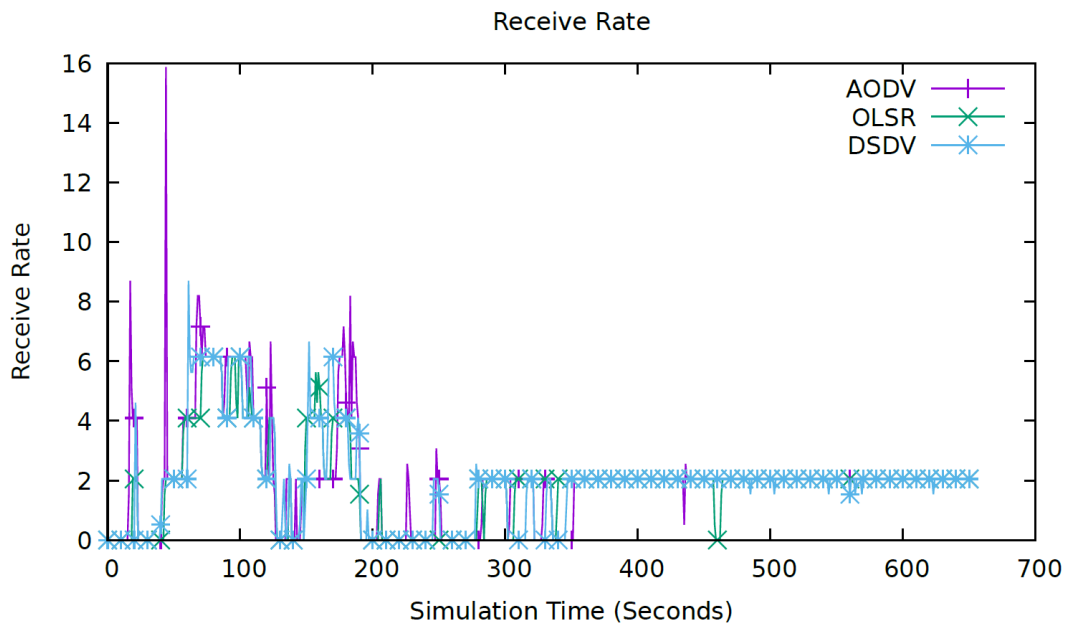


Figure 3.22: Receive Rate Lisbon Scenario

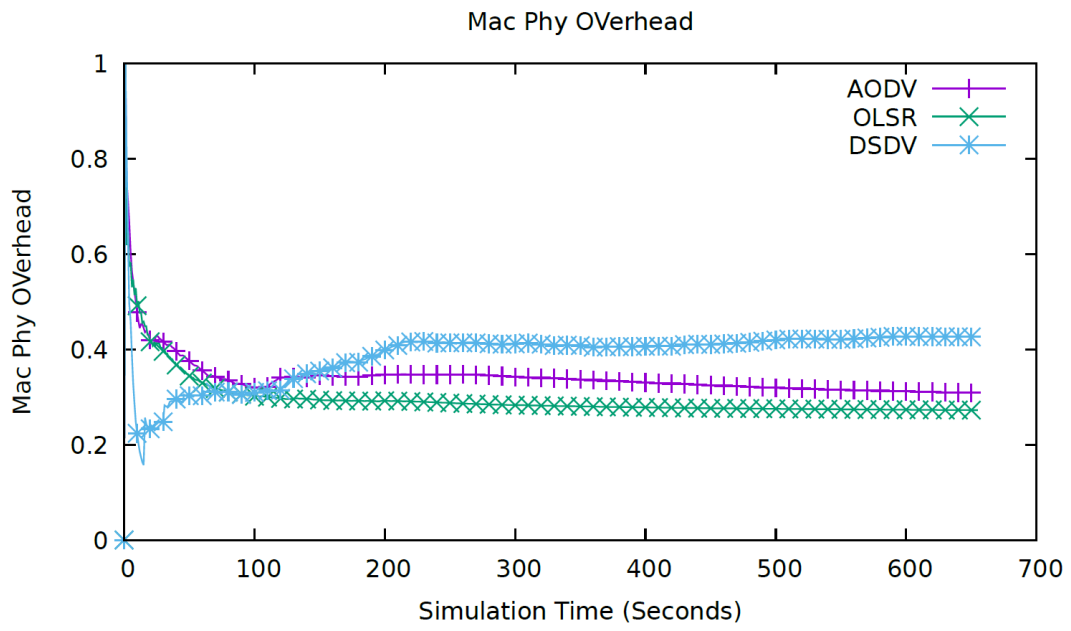


Figure 3.23: Overhead Lisbon Scenario

Table 3.8: Metric Averages of Lisbon Scenario

Protocol	Receive Rate	Overhead
AODV	2.034	0.340
OLSR	1.910	0.296
DSDV	1.939	0.388

4

Conclusion & Future Work

Contents

4.1 Conclusions	59
4.2 Future Work	59

4.1 Conclusions

This thesis addressed an emerging field in the future of Intelligent Transportation Systems (ITS), that is VANETs. More precisely, the routing protocol side of VANETs by testing the performance of three topology-based routing protocols in VANETs which are AODV, OLSR and DSDV. The testing was executed with the aid of the SUMO&NS3-Coupling tool developed in this thesis, enabling us to test four different mobility scenarios settings such as urban, highway, countryside and finally a realistic scenario resembling a famous area of Lisbon. Utilizing the SUMO&NS3-Coupling tool for the testing and judging it mostly with the overhead and goodput metrics of each protocol, in these circumstances, the urban and Lisbon scenarios were those where the OLSR was the clear winner over the others making it more suited for high node density scenarios. Furthermore, in settings with high speeds of mobility such as highway scenarios the DSDV routing protocol outperforms AODV and DSDV having the best ratio between the goodput and the overhead caused between the three protocols. Finally, in the countryside scenario, aimed to test the protocols in a low node density ambient, the AODV outperformed the DSDV, however, the same cannot be said for the OLSR that clearly struggled in this type of scenario. All things considered, this thesis concludes that the OLSR routing protocol is the most adequate for the majority of the scenarios, specially the ones with high node density, performing better in tow out of four scenarios, not trailing too much behind in the highway scenario, and lacking in the rural scenario where the low node density has affected negatively the OLSR when comparing the goodput with the overhead caused against the other protocols.

4.2 Future Work

The SUMO&NS3-Coupling very flexible, therefore making it very easily modifiable, so anything that improves or adds new functionalities to the SUMO&NS3-Coupling could be done. For instance the addition of new routing protocols such as position-based routing protocols with are probably the most adequate protocols in a near future for this kind of networks. Also, the addition of new and improved metrics for the current simulations. And for more ambitious ideas, the inclusion of autonomous vehicles data or video streaming data into the packets of the simulations. That being said, the SUMO&NS3-Coupling tool is the perfect foundation for future ideas and work related with VANETs and enables anyone who desires to work with VANETs to build on it.

Bibliography

- [1] Y. Toor, P. Muhlethaler, A. Laouiti, and A. de La Fortelle, "Vehicle ad hoc networks: Applications and related technical issues. iee communications surveys & tutorials, 10(3), 74-88," *Communications Surveys & Tutorials, IEEE*, vol. 10, 10 2008.
- [2] T. Linget, "A visionary roadmap for advanced driving use cases connectivity technologies and radio spectrum needs," *5GAA*, 2020.
- [3] K. Wevers and M. Lu, "V2x communication for its-from iee 802.11 p towards 5g," *IEEE 5G Tech Focus*, vol. 1, no. 2, 2017.
- [4] D. Jiang and L. Delgrossi, "Iee 802.11p: Towards an international standard for wireless access in vehicular environments," 06 2008.
- [5] S. Gilani and M. Jinnah, "Vehicular ad hoc network (vanet): Enabling secure and efficient transportation system," 10 2021.
- [6] P. Papadimitratos, A. de La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *Communications Magazine, IEEE*, vol. 47, 12 2009.
- [7] "Iee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021.
- [8] "Radiocommunications equipment operating in the 5 855 mhz to 5 925 mhz frequency band," *European Telecommunications Standards Institute*, 2013.
- [9] A. Jafari, S. Al-Khayatt, and A. Dogman, "Performance evaluation of iee 802.11p for vehicular communication networks," 07 2012.
- [10] A. Festag, "Cooperative intelligent transport systems standards in europe," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 166–172, 2014.

- [11] Z. Fu, X. Meng, and S. Lu, "How bad tcp can perform in mobile ad hoc networks," in *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, ser. ISCC '02. USA: IEEE Computer Society, 2002, p. 298.
- [12] M. D. Dikaiakos, S. Iqbal, T. Nadeem, and L. Iftode, "Vitp: an information transfer protocol for vehicular computing," in *VANET '05*, 2005.
- [13] J. Kakarla, S. Sathya, B. Laxmi, and B. Babu, "A survey on routing protocols and its issues in vanet," *International Journal of Computer Applications*, vol. 28, 08 2011.
- [14] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey," *IEEE Vehicular Technology Magazine*, vol. 2, no. 2, pp. 12–22, 2007.
- [15] M. L. Mat Kiah, L. Qabajeh, and M. Qabajeh, "A qualitative comparison of position-based routing protocols for ad-hoc networks," *International Journal of Computer Science and Network*, vol. 9, 01 2009.
- [16] GNU. (1991, May) GNU General Public License, version 2. Accessed 22-Aug-2021. [Online]. Available: <https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html>
- [17] S. Gupta, M. Ghonge, P. Thakare, and P. Jawandhiya, "Open-source network simulation tools an overview," *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 2, 04 2013.
- [18] A. Al Maashri and M. Ould-Khaoua, "Performance analysis of manet routing protocols in the presence of self-similar traffic," 11 2006.
- [19] A. Ulvan, V. Andriik, and R. Bestak, "The analysis of ieee802.16e mac layer overhead and efficiency in pmp topology," in *2008 5th IFIP International Conference on Wireless and Optical Communications Networks (WOCN '08)*, 2008.
- [20] N. Rani, "Performance comparison of various routing protocols in different mobility models," *International Journal of Ad hoc, Sensor & Ubiquitous Computing*, vol. 3, 08 2012.



Code of Project

This section has some of the material provided in with GitLab repository containing all the material used in this thesis.

Listing A.1: Portions of Simulation.sh

```
1 #echo "Usage:"
2 #echo "Script para converter simulacao sumo (file.sumocfg) para um ficheiro (file.mobility.tcl) de forma a ser
   usado no NS3"
3 #echo
4
5 SUMOCFG_FOLDER=$1
6 SUMOCFG_FILE=$2
7
8 #echo "$SUMOCFG_FOLDER"
9 #echo "$SUMOCFG_FILE"
10 echo
```

```

11
12 WORKDIR=$(pwd)
13 NS3_PATH=/home/ricardo/ns-allinone-3.34/ns-3.34
14
15 if [[ "$SUMOCFG_FOLDER" == "" ]]; then
16     echo "You need to define the folder which contains the .sumocfg file."
17     echo "Also, this script needs to be run as root user in ../datasets folder."
18     echo
19     return
20 fi
21
22 if [[ "$SUMOCFG_FILE" == "" ]]; then
23     echo "You need to define the .sumocfg file."
24     echo "Also, this script needs to be run as root user in ../datasets folder."
25     echo
26     return
27 fi
28
29 #-----
30
31 #First command (file.sumocfg -> file_mobility.xml)
32 sumo -c $WORKDIR/$SUMOCFG_FOLDER$SUMOCFG_FILE --fcd-output
    $WORKDIR/$SUMOCFG_FOLDER${SUMOCFG_FILE::-8}_trace.xml
33 echo
34
35 #-----
36
37 #Second command (file_trace.xml -> grid_mobility.tcl)
38 python3 $SUMO_HOME/tools/traceExporter.py -i
    $WORKDIR/$SUMOCFG_FOLDER${SUMOCFG_FILE::-8}_trace.xml
    --ns2mobility-out=$WORKDIR/$SUMOCFG_FOLDER${SUMOCFG_FILE::-8}_mobility.tcl
39 #echo "$WORKDIR/$SUMOCFG_FOLDER${SUMOCFG_FILE::-8}_mobility.tcl"
40 echo
41
42 #-----
43
44 #Script to check node number and simulation's duration
45 #x=$(py TclParser.py) #Windows

```

```

46 x=$(python3 TclParser.py $SUMOCFG_FOLDER ${SUMOCFG_FILE::-8}_mobility.tcl) #Linux
47
48 #echo ${x[*]}
49 #echo
50
51 #-----
52
53 #Third command (grid_mobility.tcl -> ns3 simulation) (Not Needed)
54 #cd $NS3_PATH
55 #./waf --run "scratch/ns2-mobility-trace
      --traceFile=$WORKDIR/$SUMOCFG_FOLDER${SUMOCFG_FILE::-8}_mobility.tcl --nodeNum=${x[3]}
      --duration=${x[4]} --logFile=$WORKDIR/$SUMOCFG_FOLDER${SUMOCFG_FILE::-8}_ns2-mob.log"
56 #cd $WORKDIR
57 #echo
58
59 #-----
60
61 ...
62
63 #-----
64
65 cd $NS3_PATH
66
67 #1=OLSR - Optimized Link State Routing Protocol
68 ./waf --run "scratch/vanet-routing-compare.cc
      --traceFile=$WORKDIR/$SUMOCFG_FOLDER/${SUMOCFG_FILE::-8}_mobility.tcl --nodes=${x[3]}
      --totaltime=${x[4]} --protocol=1 --scenario=2 --routingTables=1 --pcap=1"
69 mkdir $WORKDIR/$SUMOCFG_FOLDER/Stats
70 cp $WORKDIR/AllStats.plt $WORKDIR/$SUMOCFG_FOLDER/Stats
71 cp $WORKDIR/ComparisonStats.plt $WORKDIR/$SUMOCFG_FOLDER/Stats
72
73 mkdir $WORKDIR/$SUMOCFG_FOLDER/Stats/OLSR
74 mv routing_table $WORKDIR/$SUMOCFG_FOLDER/Stats/OLSR/${SUMOCFG_FILE::-8}_routing_table
75 cd $WORKDIR/$SUMOCFG_FOLDER
76 mv *_mobility* $WORKDIR/$SUMOCFG_FOLDER/Stats/OLSR
77 mv $WORKDIR/$SUMOCFG_FOLDER/Stats/OLSR/${SUMOCFG_FILE::-8}_mobility.tcl
      $WORKDIR/$SUMOCFG_FOLDER
78 cd $NS3_PATH

```

```

79
80 #2=AODV - Ad-hoc On Demand Distance Vector
81 ./waf --run "scratch/vanet-routing-compare.cc
      --traceFile=$WORKDIR/$SUMOCFG_FOLDER/${SUMOCFG_FILE::-8}_mobility.tcl --nodes=${x[3]}
      --totaltime=${x[4]} --protocol=2 --scenario=2 --routingTables=1 --pcap=1"
82 mkdir $WORKDIR/$SUMOCFG_FOLDER/Stats/AODV
83 mv routing_table $WORKDIR/$SUMOCFG_FOLDER/Stats/AODV/${SUMOCFG_FILE::-8}_routing_table
84 cd $WORKDIR/$SUMOCFG_FOLDER
85 mv *_mobility* $WORKDIR/$SUMOCFG_FOLDER/Stats/AODV
86 mv $WORKDIR/$SUMOCFG_FOLDER/Stats/AODV/${SUMOCFG_FILE::-8}_mobility.tcl
      $WORKDIR/$SUMOCFG_FOLDER
87 cd $NS3_PATH
88
89 #3=DSDV - Destination-Sequenced Distance Vector routing
90 ./waf --run "scratch/vanet-routing-compare.cc
      --traceFile=$WORKDIR/$SUMOCFG_FOLDER/${SUMOCFG_FILE::-8}_mobility.tcl --nodes=${x[3]}
      --totaltime=${x[4]} --protocol=3 --scenario=2 --routingTables=1 --pcap=1"
91 mkdir $WORKDIR/$SUMOCFG_FOLDER/Stats/DSDV
92 mv routing_table $WORKDIR/$SUMOCFG_FOLDER/Stats/DSDV/${SUMOCFG_FILE::-8}_routing_table
93 cd $WORKDIR/$SUMOCFG_FOLDER
94 mv *_mobility* $WORKDIR/$SUMOCFG_FOLDER/Stats/DSDV
95 mv $WORKDIR/$SUMOCFG_FOLDER/Stats/DSDV/${SUMOCFG_FILE::-8}_mobility.tcl
      $WORKDIR/$SUMOCFG_FOLDER
96 cd $NS3_PATH
97
98 #4=DSR - Dynamic Source Routing
99 #./waf --run "scratch/vanet-routing-compare.cc
      --traceFile=$WORKDIR/$SUMOCFG_FOLDER/${SUMOCFG_FILE::-8}_mobility.tcl --nodes=${x[3]}
      --totaltime=${x[4]} --protocol=4 --scenario=2 --routingTables=1 --pcap=1"
100 #mkdir $WORKDIR/$SUMOCFG_FOLDER/Stats/DSR
101 #mv routing_table $WORKDIR/$SUMOCFG_FOLDER/Stats/DSR/${SUMOCFG_FILE::-8}_routing_table
102 #cd $WORKDIR/$SUMOCFG_FOLDER
103 #mv *_mobility* $WORKDIR/$SUMOCFG_FOLDER/Stats/DSR
104 #mv $WORKDIR/$SUMOCFG_FOLDER/Stats/DSR/${SUMOCFG_FILE::-8}_mobility.tcl
      $WORKDIR/$SUMOCFG_FOLDER
105
106 #-----
107

```



```

108 cd $WORKDIR
109
110 #python3 flowmon-parse-results.py
    $WORKDIR/$SUMOCFG_FOLDER/Stats/OLSR/${SUMOCFG_FILE::-8}_mobility_FlowMonitor.xml
111 python3 flowmon-parse-results.py
    $WORKDIR/$SUMOCFG_FOLDER/Stats/AODV/${SUMOCFG_FILE::-8}_mobility_FlowMonitor.xml
112 python3 flowmon-parse-results.py
    $WORKDIR/$SUMOCFG_FOLDER/Stats/DSDV/${SUMOCFG_FILE::-8}_mobility_FlowMonitor.xml
113 #python3 flowmon-parse-results.py
    $WORKDIR/$SUMOCFG_FOLDER/Stats/DSR/${SUMOCFG_FILE::-8}_mobility_FlowMonitor.xml
114
115
116 #python3 StatsParser.py $WORKDIR/$SUMOCFG_FOLDER/Stats/OLSR/
    ${SUMOCFG_FILE::-8}_mobility_stats.csv
117 python3 StatsParser.py $WORKDIR/$SUMOCFG_FOLDER/Stats/AODV/
    ${SUMOCFG_FILE::-8}_mobility_stats.csv
118 python3 StatsParser.py $WORKDIR/$SUMOCFG_FOLDER/Stats/DSDV/
    ${SUMOCFG_FILE::-8}_mobility_stats.csv
119 #python3 StatsParser.py $WORKDIR/$SUMOCFG_FOLDER/Stats/DSR/
    ${SUMOCFG_FILE::-8}_mobility_stats.csv
120
121 #-----
122
123 mv $WORKDIR/$SUMOCFG_FOLDER/Stats/OLSR/*OLSR* $WORKDIR/$SUMOCFG_FOLDER/Stats
124 mv $WORKDIR/$SUMOCFG_FOLDER/Stats/AODV/*AODV* $WORKDIR/$SUMOCFG_FOLDER/Stats
125 mv $WORKDIR/$SUMOCFG_FOLDER/Stats/DSDV/*DSDV* $WORKDIR/$SUMOCFG_FOLDER/Stats
126 #mv $WORKDIR/$SUMOCFG_FOLDER/Stats/DSR/*DSR* $WORKDIR/$SUMOCFG_FOLDER/Stats
127
128 cd $WORKDIR/$SUMOCFG_FOLDER/Stats
129
130 #gnuplot AllStats.plt
131
132 gnuplot ComparisonStats.plt
133
134 #-----
135
136 cd $WORKDIR
137 echo

```

```
138 echo "Done"
```

Listing A.2: StatsParser.py

```
1
2 import os
3 import sys
4
5 #print(sys.argv[0]) # prints python_script.py
6 #print(sys.argv[1]) # prints .tcl file path
7 #print(sys.argv[2]) # prints .tcl file name
8
9 # specifying the stats.csv file path
10 #path = 'SUMO/Grid'
11 path = sys.argv[1]
12
13 # specifying the stats.csv file names
14 file_name = sys.argv[2]
15 file_name2 = sys.argv[2][:-4] + "_2.csv"
16 protocol = path.replace("/", "").split("Stats")[-1]
17
18 #print(protocol)
19
20 # changing dir to the .tcl file
21 try:
22     os.chdir(path)
23     #print("Current working directory: {0}".format(os.getcwd()))
24 except FileNotFoundError:
25     print("Directory: {0} does not exist. Aborting".format(path))
26     sys.exit(1)
27
28 # opening the files in READ mode
29 try:
30     file = open(file_name, "r")
31 except FileNotFoundError:
32     print("File: {0} not found. Aborting".format(file_name))
33     sys.exit(1)
34
```

```

35 try:
36     file2 = open(file_name2, "r")
37 except FileNotFoundError:
38     print("File: {0} not found. Aborting".format(file_name2))
39     sys.exit(1)
40
41
42 with file:
43     with open(protocol + ".csv", "w") as temp:
44         for line in file:
45             temp.write(line.replace(",", " "))
46
47 with file2:
48     with open(protocol + "_2.csv", "w") as temp2:
49         for line in file2:
50             temp2.write(line.replace(",", " "))
51
52
53 file.close()

```

Listing A.3: TclParser.py

```

1
2 import os
3 import sys
4
5 print(sys.argv[0]) # prints python_script.py
6 print(sys.argv[1]) # prints .tcl file path
7 print(sys.argv[2]) # prints .tcl file name
8
9 # specifying the .tcl file path
10 #path = 'SUMO/Grid'
11 path = sys.argv[1]
12
13 # specifying the .tcl file name
14 #file_name = 'grid_mobility.tcl'
15 file_name = sys.argv[2]
16

```

```

17 # changing dir to the .tcl file
18 try:
19     os.chdir(path)
20     #print("Current working directory: {0}".format(os.getcwd()))
21 except FileNotFoundError:
22     print("Directory: {0} does not exist. Aborting".format(path))
23     sys.exit(1)
24
25 # opening the file in READ mode
26 try:
27     file = open(file_name, "r")
28 except FileNotFoundError:
29     print("File: {0} not found. Aborting".format(file_name))
30     sys.exit(1)
31
32 last_node, duration = 0, 0
33
34 for line in file:
35
36     if line.startswith("$node"):
37         strnum = ""
38         for char in line[7:]:
39             if char == ")": break
40             strnum += char
41
42         last_node = int(strnum)
43
44     if line.startswith("$ns"):
45         strnum = ""
46         for char in line[8:]:
47             if char == " ": break
48             strnum += char
49
50         duration = float(strnum)
51
52
53 file.close()
54

```

```
55 print(last_node + 1, duration + 0.01)
```


B

Installation Guides for SUMO/NS-3

This section has the guides for the installation and for both SUMO and NS-3 software necessary to run the SUMO&NS3-Coupling tool.

B.1 SUMO Installation & Set-up

The following commands are needed, starting with a update of the environment packages by opening the terminal in home directory and typing:

```
1 sudo apt update
```

Then, the following command that will install all the dependencies needed for SUMO and some other useful libraries to complement SUMO:

Listing B.1: SUMO prerequisites

```
1 sudo apt-get install cmake python g++ libxerces-c-dev libfox-1.6-dev
libgdal-dev libproj-dev libgl2ps-dev swig
```

As this installation gets the Linux binaries directly from the SUMO repository, the git is needed to download them. So, to install git the following command is needed:

```
1 sudo apt install git
```

Now with git installed, it is possible to clone the SUMO from the repository by typing the following command:

```
1 git clone --recursive https://github.com/eclipse/sumo
```

With the SUMO downloaded, is necessary to set-up the environment variable by typing the following command:

```
1 export SUMO_HOME="$PWD/sumo"
```

Then, type the following command to create a new directory:

```
1 mkdir sumo/build/cmake-build && cd sumo/build/cmake-build
```

Then, type the following command:

```
1 cmake ../..
```

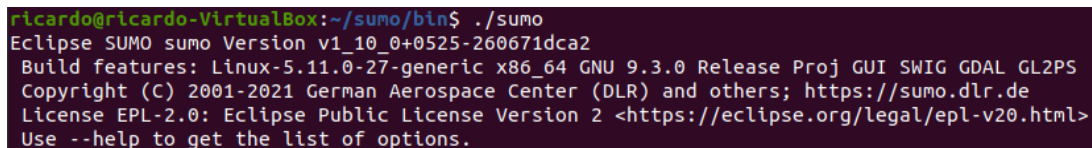
Once that is done, the final command for the instalation and set-up is the following:

```
1 make -j8
```

This command builds the SUMO and some other tools such as netedit and libtraci among others, which will be helpful for the following sections, where we are going to set up the experimental environment of this thesis.

With that said, it is time to check for any failures, crashes, or other errors with the SUMO build. So, if everything went well and we type the following command on `/sumo/bin` directory, we should obtain the same as Fig. B.1

```
1 ./sumo
```



```
ricardo@ricardo-VirtualBox:~/sumo/bin$ ./sumo
Eclipse SUMO sumo Version v1_10_0+0525-260671dca2
Build features: Linux-5.11.0-27-generic x86_64 GNU 9.3.0 Release Proj GUI SWIG GDAL GL2PS
Copyright (C) 2001-2021 German Aerospace Center (DLR) and others; https://sumo.dlr.de
License EPL-2.0: Eclipse Public License Version 2 <https://eclipse.org/legal/epl-v20.html>
Use --help to get the list of options.
```

Figure B.1: Example of successfully installation

B.2 NS-3 Installation & Set-up

Consequently, the following commands are needed, starting with an update of the environment packages by opening the terminal and typing:

```
1 sudo apt update
```

Then, the following command that will install all the dependencies needed for NS-3 and some other useful tools for studying simulation scenarios:

Listing B.2: NS-3 prerequisites

```
1 sudo apt install g++ python3 python3-dev python-dev pkg-config sqlite3
python3-setuptools git qt5-default gir1.2-goocanvas-2.0 python3-gi
python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 openmpi-bin
openmpi-common openmpi-doc libopenmpi-dev autoconf cvs bzip2 unrar
openmpi-bin openmpi-common openmpi-doc libopenmpi-dev tcpdump wireshark
libxml2 libxml2-dev
```

After the installation of the prerequisites, the environment is ready to install the NS-3 and the first step is to download one of the releases of NS-3 from the following link <https://www.nsnam.org/releases/>. The next step is to extract the archive downloaded to the home folder to look like the following Fig. B.2

Once that is done, open the terminal in the `ns-allinone-3.34` type the following command:

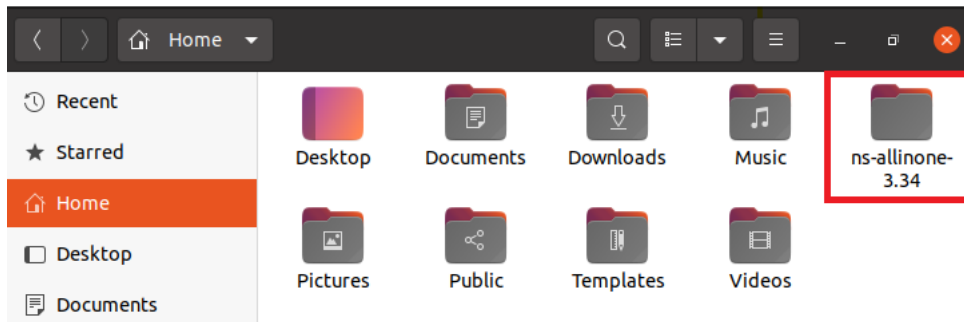


Figure B.2: Home folder

```
1 ./build.py --enable-examples --enable-tests
```

This command will run the python script called `build.py`. This script will get the NS-3 installed and configured for the most commonly useful way. However, for more advanced configurations, the build process typically involves using the native NS-3 build system, Waf. But, for simplicity purposes of this dissertation only the python script was covered.

Now that all the steps for the installation are done, we can run the following command in the following directory `/ns-allinone-3.34/ns-3.34`:

```
1 ./test.py
```

This command runs another python script called `test.py`. This script will check for any failures, crashes or other errors with the NS-3 build. If everything went well all the tests should be marked as passed.

