



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

AN APPLICATION-DRIVEN FRAMEWORK FOR INTELLIGENT
TRANSPORTATION SYSTEMS USING SOFTWARE-DEFINED VEHICULAR
NETWORKS

Tiago do Vale Saraiva

Orientadores

Carlos Alberto Vieira Campos
Sidney Cunha de Lucena

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 2018

An Application-driven Framework for Intelligent Transportation Systems using
Software-defined Vehicular Networks


TIAGO DO VALE SARAIVA

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO
DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFOR-
MÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO).
APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.


Aprovada por:



Carlos Alberto Vieira Campos, D.Sc. - UNIRIO



Sidney Cunha de Lucena, D.Sc. - UNIRIO



Carlos Eduardo Ribeiro de Mello, D.Sc. - UNIRIO



José Ferreira de Rezende, D.Sc. - UFRJ

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 2018.

Catálogo informatizada pelo(a) autor(a)

S243	<p>Saraiva, Tiago do Vale AN APPLICATION-DRIVEN FRAMEWORK FOR INTELLIGENT TRANSPORTATION SYSTEMS USING SOFTWARE-DEFINED VEHICULAR NETWORKS / Tiago do Vale Saraiva. -- Rio de Janeiro, 2018. 100 f.</p> <p>Orientador: Carlos Alberto Vieira Campos. Coorientador: Sidney Cunha de Lucena. Dissertação (Mestrado) - Universidade Federal do Estado do Rio de Janeiro, Programa de Pós-Graduação em Informática, 2018.</p> <p>1. Sistemas Inteligentes de Transporte. 2. Redes Veiculares. 3. Redes Definidas por Software. 4. Cidades Inteligentes. 5. Emulação. I. Campos, Carlos Alberto Vieira, orient. II. Lucena, Sidney Cunha de, coorient. III. Título.</p>
------	--

Dedico essa dissertação à minha família.

Agradecimentos

Em primeiro lugar, agradeço a minha família, por acreditar e investir em mim. Sem o apoio da minha mãe e do meu pai eu não teria conseguido alcançar esse objetivo.

Aos amigos que me incentivaram na execução desse trabalho e estiveram disponíveis para que eu pudesse compartilhar as diversas emoções durante esse período.

Agradeço também ao professor Carlos Alberto Vieira Campos, que com sua dedicação e competência me orientou com excelência, possibilitando que eu conseguisse produzir esse trabalho e concluir essa importante etapa.

Agradeço ainda aos membros da UNICAMP, professor Christian Esteve Rothenberg pela atenção dada durante a elaboração desse trabalho e ao pesquisador Ramon Fontes, sempre disponível para ajudar e com orientações que foram fundamentais para finalizar o estudo de caso da solução proposta, através do uso do Mininet-wifi.

Saraiva, Tiago do Vale **An Application-driven Framework for Intelligent Transportation Systems using Software-defined Vehicular Networks**. UNIRIO, 2018. 95 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

Diversos problemas urbanos vêm sendo resolvidos com o uso de recursos tecnológicos nas cidades inteligentes. Os sistemas inteligentes de transporte (ITS) visam tornar os meios de transporte mais eficientes, confortáveis e seguros. As redes de comunicação veiculares fornecem a infraestrutura que suporta a comunicação de modernas aplicações ITS. Essas redes são formadas pelos veículos em movimento conectados entre si e com a infraestrutura da cidade, e estão sujeitas a constantes mudanças de topologia, desconexões e congestionamentos, em função da dinâmica de mobilidade natural dos veículos. As diferentes aplicações veiculares possuem diferentes requisitos de comunicação como, por exemplo, atraso e largura de banda, sendo que suportar de forma dinâmica esses requisitos em um ambiente complexo como o das redes veiculares é um grande desafio. Os trabalhos relacionados não fornecem soluções que tratem desse problema considerando os aspectos necessários tanto das aplicações quanto da rede de comunicação veicular. Este trabalho trata do problema e propõe uma solução através de um framework para a implantação de redes veiculares que atendem dinamicamente aos requisitos de diferentes aplicações ITS, através de uma abordagem com o uso de redes definidas por software. A proposta foi validada através de uma estratégia de emulação realística em um cenário de congestionamento urbano com aplicações veiculares com diferentes requisitos de largura de banda e os resultados foram comparados com uma abordagem que faz uso de mecanismos de qualidade de serviço (QoS) e outra que não faz priorização de tráfego. Com o uso da solução proposta foram obtidos resultados superiores de taxa de recepção de dados e perda de pacotes, sem impactar significativamente o tempo de comunicação entre os veículos e os servidores das aplicações.

Palavras-chave: Sistemas Inteligentes de Transporte, Redes Veiculares, Redes Definidas por Software, Cidades Inteligentes, Emulação.

ABSTRACT

Several urban problems have been solved with the use of technological resources in smart cities. Intelligent transportation systems (ITS) aim to make transportation more efficient, comfortable and safe. Vehicular communication networks provide the infrastructure that supports the communication of modern ITS applications. These networks are formed by moving vehicles connected to each other and to the city's infrastructure, and are subject to constant changes of topology, disconnections and congestion, depending on the dynamics of natural mobility of vehicles. Different vehicular applications have different communication requirements, such as delay and bandwidth, and dynamically supporting these requirements in a complex environment such as vehicular networks is a great challenge. The related works do not provide solutions that deal with this problem considering the necessary aspects of both the applications and the vehicular communication networks. This work addresses the problem and proposes a solution through a framework for the deployment of vehicular networks that dynamically meet the requirements of different ITS applications, through an approach with the use of software defined networks. The proposal was validated through a realistic emulation strategy in a scenario of urban congestion with vehicular applications with different bandwidth requirements and the results were compared with an approach that makes use of quality of service mechanisms (QoS) and another approach that does not prioritize any traffic. With the use of the proposed solution, higher data reception rate and packet loss rates were obtained without significantly impacting the communication time between the vehicles and the application servers.

Keywords: Intelligent Transportation Systems, Vehicular Networks, Software Defined Networks, Smart Cities, Emulation.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Formulation of the problem	3
1.3	Justification	3
1.4	Objectives	3
1.5	Open Challenges and Contributions	4
1.6	Structure of the work	5
2	Theoretical basis and related work	6
2.1	Nomenclature and Synonyms	6
2.2	ITS - Intelligent Transportation systems	6
2.2.1	ITS Applications	7
2.3	Vehicular Networks	9
2.4	5G Vehicular Networks	11
2.5	Software Defined Vehicular Networks	15
2.6	Related Works	17
3	A Framework for Application-driven vehicular networks	22
3.1	Application plane	24

3.2	Data plane	26
3.3	Control Plane	27
3.3.1	MANO (Management and Orchestration)	28
3.3.2	Central Network Controller	30
3.3.3	Local Network Controller	35
3.3.4	Global Mobility Algorithm	37
4	Performance evaluation	40
4.1	Implementation	40
4.1.1	The Mininet-wifi emulator	41
4.1.1.1	Implementation of Mininet-wifi emulation components	43
4.1.2	Generation of traffic related to the applications	45
4.1.3	Deploying the shared database	45
4.1.4	Central Controller Implementation	47
4.1.5	Implementation of local controllers	48
4.1.6	Auxiliary codes	51
4.2	Evaluation methodology	51
4.3	Evaluation scenario	52
4.4	Evaluation metrics	56
4.5	Results	57
4.5.1	RSUs balance as function of framework actions	57
4.5.2	Throughput and RTT over time	60
4.5.3	Packet Delivery Ratio	68
4.5.4	Round Trip Time	73
4.5.5	Discussion	76

5	Conclusion and future works	78
	References	80

List of Figures

2.1	A VANET in V2I and V2V communications.	10
2.2	Network slicing in 5G. [6]	13
2.3	An architecture of Vanets using MEC. [6]	14
2.4	SDVN system architecture [28]	16
2.5	The components of an Software Defined Vehicular Network [1]	17
3.1	Architecture of proposed framework	22
3.2	Abstraction layers of proposed framework with correspondent components	23
3.3	Flow chart of the framework in function of changes in vehicles topology, applications or infrastructure	27
3.4	MANO actions enumerated by execution order, where 1 are the feed in- formations, 2 the database query, 3 the heuristics to define new infrastruc- ture, 4 the infrastructure deployment, 5 the update in shared database and 6 the central controller notification.	30
3.5	Informations in a shared database example and respectively slices	34
4.1	Components and connections in a network created with Mininet-WiFi with two hosts and a Access Point	42
4.2	Architecture of three node cars in an example with a RSU node	43
4.3	Tables created in shared database	46

4.4	A summary of the actions executed by local control in RSUs	49
4.5	Evaluation scenario representing a traffic jam with different congestion levels over the time	53
4.6	Throughput and RTT over the time in application E using framework approach	61
4.7	Throughput and RTT over the time in application E using QoS only approach	62
4.8	Throughput and RTT over the time in application E using Best effort approach	63
4.9	Throughput and RTT over the time in application E2 using Framework approach	63
4.10	Throughput and RTT over the time in application E2 using QoS only approach	65
4.11	Throughput and RTT over the time in application E2 using Best effort approach	65
4.12	Throughput and RTT over the time in application G using Framework approach	66
4.13	Throughput and RTT over the time in application G using QoS approach .	66
4.14	Throughput and RTT over the time in application G using Best effort approach	67
4.15	Throughput and RTT over the time in application S using Framework approach	67
4.16	Throughput and RTT over the time in application S using QoS approach .	68
4.17	Throughput and RTT over the time in application S using Best effort approach	68
4.18	Application E - PDR	69
4.19	Application E2 - PDR	70
4.20	Application G - PDR	71

4.21	Application S - PDR	72
4.22	ECDF of Application E - RTT	73
4.23	ECDF of Application E2 - RTT	74
4.24	ECDF of Application G - RTT	75
4.25	ECDF of Application S - RTT	76

List of Tables

2.1	KPI requirements for V2X use cases [8]	9
3.1	MANO algorithm - Inputs and Examples	28
3.2	MANO algorithm - Outputs and Examples	29
4.1	Configurations used to performance evaluation	54
4.2	Applications characteristics	54
4.3	RAN status in evaluated times	55
4.4	Results in balance of RSU1 with Framework solution	58
4.5	Results in balance of RSU2 with Framework solution	59
4.6	Results in balance of RSU3 with Framework solution	60
4.7	App E Packet Delivery Ratio	69
4.8	App E2 Packet Delivery Ratio	70
4.9	App G Packet Delivery Ratio	71
4.10	App S Packet Delivery Ratio	72
4.11	App E Round trip time	73
4.12	App E2 Round trip time	74
4.13	App G Round trip time	75
4.14	App S Round trip time	76

List of Nomenclatures

APP	Application
DPI	Deep Packet Inspection
DPID	Datapath Identifier
DSRC	Dedicated Short Range Communications
eNodeB	enhanced Node B
IEEE	Institute of Electrical and Electronic Engineers
IoV	Internet of Vehicles
IP	Internet Protocol
ITS	Intelligent Transportation System
JSON	Java Script Object Notation
KPI	Key Performance Indicator
LTE	Long Term Evolution
LTE-V	Long Term Evolution for Vehicles
LTE-V2X	Long Term Evolution for V2X
MANET	Mobile Ad Hoc Networks
MEC	Mobile Edge Computing
NBI	Northbound Interface
OBU	On Board Unit
ODN	Optical Distribution Network
OVS	Openvswitch
OVSDB	Open Vswitch Database
PDR	Packet Delivery Ratio
PON	Passive Optical Networks
QoS	Quality of Service
RAN	Radio Access Network
REST	Representational State Transfer
RSU	Roadside Unit

RTT	Round Trip Time
SBI	Southbound Interface
SDN	Software Defined Network
VANET	Vehicular Ad Ho Network
VNF	Virtual Network Function
V2I	Vehicle to infrastructure
V2X	Vehicle to Everything
V2V	Vehicle to Vehicle
VCC	Vehicular Cloud Computing
WAVE	Wireless Access in Vehicular Environment

1. Introduction

Intelligent Transportation Systems (ITS) emerged around the 2000s using technologies to some simple tasks such as plate recognition and traffic light control. Nowadays ITS is a broad term that englobes a wide range of technologies concerning modern vehicular services. There are under study several vehicular safety applications to avoid accidents and the most recent are related to the hot topic of Autonomous Driving. Information and entertainment services (Infotainment) are also emerging in vehicular networks and attracting the attention of service providers. These services include, but are not limited to, advertisements, tourist information, traffic information, and parking, and can attract a considerable mass market in vehicular networks, reaching sales of up to \$131.9 billion in 2019 [1].

Vehicular Networks provide the infrastructure for wireless communication to enable vehicular applications both in urban and highways scenarios. Over the years, these communication technologies were known by different terms. With the rapid development of wireless communications, came the VANETs (Vehicular Adhoc Networks) concept, using DSRC (Dedicated Short-range Communications), defined in IEEE 802.11p standard, to improve road safety and transport efficiency.

For a long time, VANETs were on top researches, but with the evolution of vehicular applications demands and the challenges imposed by the vehicular environment, the reliability of the connections was falling behind [2]. With the emergence of Big Data and Internet of Things came the Internet of Vehicles (IoV) concept [3]. Today, in Vehicle-to-Everything (V2X), vehicles can communicate with everything and this paradigm is seen as an enabler for safer, cheaper, intelligent, connected and autonomous transportation systems [5].

The highly dynamic environment of vehicular networks, in function of vehicles dynamics, causes dramatic changes in the spatial and temporal behavior of the network

topology, resulting in degraded communication quality. At the same time, various vehicular applications need their communication requirements to be addressed dynamically in such a complex environment.

According to [6], traditional mobile communication networks employ the one-size-fits-all approach to provide services to mobile devices, regardless of the communication requirements of vertical services. This traditionally fixed resource distribution mode fails to satisfy the future driving environment [2].

Some frameworks propose the use of Software Defined Networks (SDN) to allow greater flexibility and better deal with these challenges. This paradigm separates the communication devices in the data plane, where the packet switching devices are, and in the control plane, where the devices that define how the data plane should switch the packets are, allowing a centralized, programmable and flexible control [7].

Currently, 5G networks bring some technologies to improve Radio Access Networks (RAN) and the core of mobile communication networks. With the concept of network slicing, 5G aims to meet diversified service requirements under the background of existing technologies [6]. Although it is difficult to determine when exact standardization of 5G will occur with procedures to meet specific vehicle communications requirements [3], the lack of alternatives has motivated the interest in using 5G for vehicular communications.

Despite the automotive industry has experienced advanced transformations due to innovative technologies, such that cited in previous paragraphs, the communication requirements to new applications regarding latency, reliability, and scalability are beyond the capabilities of current 4G (or the evolving 5G) and VANETs [8].

1.1 Motivation

Intelligent Transportation Systems are one of the components to make Smart Cities through which they have sought to improve levels of safety, comfort, and efficiency of transportation systems. Vehicular networks support the exchange of messages by vehicles with the information necessary for proper functioning of these systems.

Because of the vehicular communications complexity, in function of topology changes caused by the natural dynamics of vehicles, the networks have difficult to accomplish the requisites of modern and future ITS applications adequately. So, it is necessary to pursue new solutions that may improve vehicular networks to deal with requirements of ITS applications properly and enable the improvement of the transportation systems.

1.2 Formulation of the problem

The highly dynamic environment of vehicular networks causes dramatic changes in the spatial and temporal behavior of the network topology, resulting in degraded communication quality. Such issue, in conjunction with diverse vehicular applications requirements, forms a problem which is to provide a infrastructure that can dynamically meet vehicular applications communication requirements in such complex environment.

1.3 Justification

ITS applications are increasingly demanding vehicular communication resources, and these communication networks have not been designed for such demands, resulting in degraded experiences that impact the future of modern applications.

Although several works address the problem of how to accomplish the requirements of different applications in a vehicular network, no one proposes a broad and clear detailed solution, considering the necessary aspects of vehicular networks and applications, with proper validation. Even in the works that are already studying the use of 5G Network Slicing in vehicular networks there is a lack in providing details of how it can be used in practical scenarios.

1.4 Objectives

The objective of this work is to study the technologies that can be used in the orchestration of resources in a vehicular communication network and how to integrate them to provide a solution to the problem that is the deployment of a mobile infrastructure that can dynamically fulfill the demands of vehicular applications.

Using the knowledge about state of the art, it is also objective of this work to propose an architecture that integrates some existing technologies and newly proposed algorithms to act in order to manage the resources of the vehicular network to meet vehicular applications requirements dynamically.

Using realistic techniques, it is another objective of this work to apply the proposed solution in an emulated scenario to prove its efficiency.

1.5 Open Challenges and Contributions

As will be shown in Section 2.6, there is in state of the art several open points that need to be further explored to reach the goal of deploying vehicular networks that dynamically meet the requirements of different ITS applications. In this way, we list below some of the main open challenges identified and the corresponding contribution of this work:

1. Definition of how controllers receive the information about the requirements of the application and how the solution acts to prioritizing more than one application, differentiating them to meet specific requirements:
 - (a) This work proposes an architecture to application-driven software-defined vehicular networks, where the requirements of applications and others information associated with each vehicle are stored in a shared database accessible by elements with control functions;
 - (b) The network control algorithms that we propose define how the controllers use the information in the shared database to configure the network prioritizing the limited resources.
2. Definition of how network controllers use in the decision-making process the information related to topology changes due to vehicular traffic dynamics:
 - (a) The network control algorithm of local controllers proposed deal with the problem of identifying the vehicles in each RAN, using the information of vehicles associated with local RSU. According to vehicles movements, local controllers calc the resource balance and act if necessary.
3. Use of a concept proof with realistic SDN components, like existing SDN controllers, switches and OpenFlow features, and taking into account realistic vehicular scenarios with VANETs components such as vehicles and RSUs:
 - (a) The proof of concept conducted to evaluate the proposed solution uses a realistic emulation strategy with a scenario of an urban congestion scenario and software-defined vehicular network with applications with different bandwidth requirements. It is made the use of metrics such as Packet Delivery Ratio and Round Trip Time and the results are compared with other possible approaches found in the literature.

Some solutions deal with these topics partially, but this is the first that addresses them in conjunction with a detailed approach defining the architecture and algorithms with proper validation.

1.6 Structure of the work

The Chapter 1 contains a brief overview of vehicular networks, ITS applications, and software-defined networks. It presents some challenges related to the fulfillment of applications requirements by vehicular networks and defines the problem that this work aims to approach, outlining its objectives and contributions.

The Chapter 2 begins with the presentation of the theoretical foundation that underlies the work. It exposes the concepts of vehicular networks and their evolution over the time, including the standard IEEE 802.11p and LTE-V in 5G networks. The key 5G technologies are presented, such as software-defined networks, mobile edge computing, network function virtualization, and how these technologies can be used to improve vehicular network efficiency. In sequence, related works are presented and shown how they deal with the problem of meeting the requirements of ITS applications, and what are the open research points.

The Chapter 3 presents the proposed Framework with a detailed explanation of its architecture and related algorithms.

The Chapter 4 describes the implementation details, the configurations used in performance evaluation, the evaluated scenario, performance metrics, and a discussion of the results comparing the proposed solution with an approach using only Quality of Service and other with Best Effort.

Chapter 5 presents the conclusions of the work as well as the research suggestions for future work.

2. Theoretical basis and related work

This chapter begins with the presentation of the theoretical foundation that underlies the work. It exposes the concepts of vehicular networks and their evolution over the time, including the IEEE 802.11p standard and LTE-V in 5G. The key 5G technologies are presented, such as software-defined networks, mobile edge computing, network function virtualization, and how these technologies can be used to improve vehicular network efficiency. In sequence, related works are presented and showed how they deal with the problem of meeting the requirements of ITS applications and what are the open research points.

2.1 Nomenclature and Synonyms

Before presenting the concepts and the related works to this research theme, it is necessary to define the nomenclature and some synonyms used throughout the work.

RSU, Base stations and eNodeB's, in our terminology and within the scope of work, have the same meaning. Thus, we shall refer to each of these terms without distinction, bearing in mind that they relate to the same thing.

The App, Application, ITS Application, and vehicular application also have the same meaning in this work. KPI, Applications requirements and demands also have the same sense.

2.2 ITS - Intelligent Transportation systems

There has been a growing demand for more efficient cities, in an attempt to reduce the adverse effects of urbanization, like for example traffic congestion, which have led

governments to adopt technologically-based approaches and handle the adverse effects of urbanization via broadband interconnected cities, known as Smart Cities [9].

A concept closely linked to smart cities is that of ubiquitous or pervasive computing, where day-to-day objects have communication, storage, and processing skills, providing services to users anytime, anywhere [10]. As vehicles are continuously improving regarding processing power and networking capacities, they are one of the most promising building blocks for these smart cities [11].

In this context, Intelligent Transportation System (ITS) represents a new concept that emerges with the objective of providing better levels of safety, comfort, and efficiency to all those involved in a transportation system.

Vehicular Ad-hoc Networks (VANETs) can be used to boost the use of ITS [12], and to provide communication and automation services infrastructure to make cities smarter [11]. According to [19], advancements in mobile communications and protocols for VANETs will allow the emergence of architectural solutions for vehicular networks, in both road and urban environments, to support applications with different requirements, which will allow the delivery of new services related to efficiency, comfort and entertainment.

As an example of how VANETs can serve to support ITS-related applications and also provide communication infrastructure for smart cities, [13] evaluates the taxi mobility in the city of Rome (Italy), to verify the efficiency of these vehicles as “mules” that can provide communication.

2.2.1 ITS Applications

For a long time, the primary practical focus on ITS applications has been in the development of protocols to support road traffic safety through the transmission of messages with information such as speed, position, and direction of vehicles [14]. The official name of the protocol that sends these position messages in Europe is CAM (Cooperative Awareness Messages). In the United States, the equivalent are the BSM (Basic Safety Messages). The applications of BSM messages dissemination are classified by [15] in single hop and multiple hops. The difference is that in the second case it is necessary the routing of messages by intermediate nodes. The implementation of a multi-hop approach is more complicated than the single-hop because it requires a routing algorithm for messages, while the single-hop strategy is less efficient because of the smaller range.

There are also DENM (Decentralized Environmental Notification Messages), which

are event-triggered messages that are transmitted in case of specific events (e.g., accidents). While the event is valid, the DENMs will be transmitted alongside CAMs [14].

[17] provide some other examples of vehicular applications, like vehicular traffic control (intelligent navigation and intelligent signaling), steering safety (detection of road conditions and emergency warnings) and entertainment (advertising and multimedia). [1] cites vehicular applications of security, surveillance, traffic management, network virtualization, infotainment, parking, and signage management (traffic lights). [12] is another work where several applications designed for vehicular networks are presented.

Different applications in VANETs have varying requirements. Forward Collision Warning (FCWS) applications, for example, require fast message dispersal for nearby vehicles, while other messages, such as bottleneck alerts, support greater latency [16]. The applications requirements can be named as KPIs (Key Performance Indicators), that define the conditions concerning the communication that the infrastructure needs to provide to the applications works properly.

The 3rd Generation Partnership Project (3GPP) is a group of telecommunications standard development organizations that produce the reports and specifications that define the 3GPP technologies, which deal with Radio Access Networks (RAN), Services & Systems Aspects (SA), and Core Network & Terminals (CT).

3GPP has started the work on new 5G V2X (Vehicle to Everything) communications enhancements under Release 15 and has completed the analysis of new use cases like autonomous driving, platooning, sensor and map sharing, information sharing for partial/conditional and high/fully automated driving, remote driving, among others [18]. Some of these use cases require the transmission of up to 50 pps (packets per second), a maximum latency between 3 and 10 ms, and up to a 99.99% reliability level (defined regarding PDR), and to support these requirements, some of the LTE-V enhancements (referred to as V2X Phase 2 or eV2X) are under discussion in Release 15 [18].

Some examples of V2X use cases cited by [8] are cooperative collision avoidance system (CCAS), bird's eye view system (BEVS) at the road intersection, augmented reality (AR), intelligent navigation system (INS) based on real-time road conditions, and 4K live video. In Table 2.1 are the KPIs of these examples.

Table 2.1: KPI requirements for V2X use cases [8]

Use case	E&E Latency (ms)	Reliability	Data rate (Mbps)
CCAS	10	10^{-5}	Less than 5
BEVS	50	10^{-3}	40
AR	100	10^{-2}	100
INS	100	No	50
4K live video	500ms	No	40Mbps (per video)

2.3 Vehicular Networks

Vehicular networks, also known as Vehicular Ad Hoc Networks, are networks formed by moving vehicles equipped with wireless communication devices. These networks are a particular case of Mobile Ad Hoc Networks, with the difference that the mobility of nodes (cars) in VANETs is restricted by roads and traffic characteristics (traffic jam, speed limit, signaling, etc.) in each region.

The communications in a vehicular network can be between vehicles, also known as V2V (Vehicle to Vehicle), between vehicles and infrastructure, also known as V2I (Vehicle to Infrastructure) or Hybrid (combination of V2V and V2I). There is also V2X, where vehicles can communicate with "everything," including cities sensors and pedestrians. These communications follow some rules defined in standards like IEEE 802.11p or 3GPP 5G.

When in V2I mode, there is a unit connected to the city's infrastructure, to provide infrastructure resources access to vehicles. In 802.11p this unit is denominated RSU (Roadside Unit) and in 5G is eNodeB (enhanced NodeB). Figure 2.1 shows a VANET in a generic city.

Each vehicle in a VANET has an OBU (On Board Unit), and when the cars are communicating with each other, the transmission/reception happens between these OBUs. In the infra-structured case, the communication takes place between the OBUs in each vehicle and the Road Side Units of the city.

Given the movement of vehicles, the topology in a vehicular network is generally quite dynamic, with frequent disconnections and mobility limited by transit routes. Other general characteristics are low bandwidth and wireless transmission over short distances [19], when using IEEE 802.11p standard.

In urban scenarios, the topology of a VANET may have hundreds of vehicles in a relatively small region. In this case, solutions that deal mainly with collisions or traffic

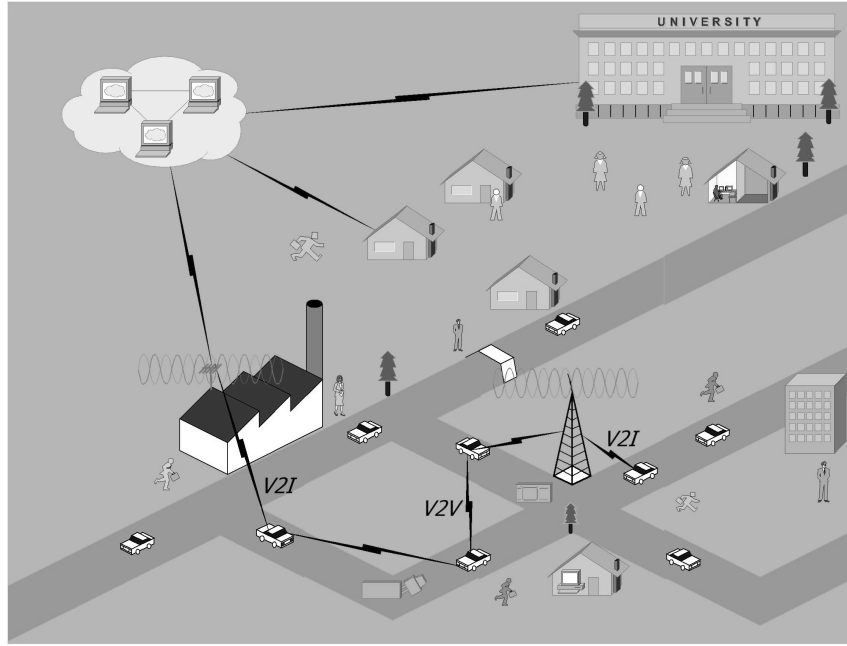


Figure 2.1: A VANET in V2I and V2V communications.

congestion are better. On the other hand, in expressway scenarios, where the topology is sparser, and the connectivity is more intermittent, disconnection tolerant protocols may be more appropriate. Also, vehicles traveling in both situations need to adapt their behavior to variations in network density to provide the best data transfer [19].

The reference architecture of VANETs is called WAVE and is defined by the IEEE 1609 standards [20]. IEEE 1609.4 provides for multi-channel operations using IEEE 802.11p, which is part of the Dedicated Short Range Communications (DSRC), in the 5.9 GHz frequency band, where the spectrum is divided into seven channels of 10 MHz, being three of control, and data rate between 3 and 27 Mbps [21]. ETSI EN 302 665 describes an inter-vehicle communication architecture for Europe [14]. These are the de facto standard for VANETs [5].

DSRC has been proven to be very useful in supporting both safety and non-safety services in V2V communications [22]. However, it should be noted that VANETs have been studied for more than ten years and standardized to at least 6 in the United States and have not yet been widely adopted, due to cost and technology challenges in a dynamic environment with requirements as strict as the vehicular [8].

According [3], since IEEE 802.11p provides wireless transmissions over short distances, support diverse vehicular applications requires widespread deployment of RSUs (Roadside Units) to infrastructure-based communications, to cope with the effect of intermittent connectivity and, once the widespread implementation of RSUs may not be

practical, this standard does not fully meet the application diversity requirements of vehicular communications.

As cited by [18], IEEE 802.11p uses as medium-access-scheme the carrier-sense multiple access with collision-avoidance and can face some challenges when guaranteeing strict reliability levels and ensuring the network's scalability as the load increases. Other problem is that a significant portion of the dedicated spectrum in IEEE 802.11p is utilized by periodic beacons, causing it to reach its capacity quickly, increasing the probability of erroneous transmissions in large-scale network deployments [3], which is not supported by some applications.

Thus, given the limitations presented, regarding IEEE 802.11p, it is clear that there are significant challenges in supporting different ITS applications with varied communication requirements in a vehicular network with this technology.

2.4 5G Vehicular Networks

Fifth-generation (5G) mobile networks are under intensive research with the purpose of improving existing technologies to reach new levels of quality in communications and support new services that aren't possible today. It is important to emphasize that 5G encompasses communication technologies for many use cases, including vehicular scenarios. As cited by [3] 5G is not only a new access technology but also a user-centric network concept that aims to address the application requirements of all the stakeholders in a connected world.

There are two complementary views driving research and industry activity on 5G [23]. One view is looking at novel technologies to improve radio access, resulting in more efficient spectrum utilization, and another view is service-oriented, dealing with questions related to support a wide range of services with different requirements in the network.

As an alternative to IEEE 802.11p, the 3GPP published the first version of its Release 14 standard in September 2016, which includes support for V2X communications in 5G. The standard is commonly referred to as LTE-V, LTE-V2X, or cellular V2X and its physical layer improves the link budget concerning IEEE 802.11p, while increase the reliability, under certain conditions, by adding a redundant transmission per packet [18].

While IEEE 802.11p has a bandwidth limitation of 10Mbps per channel, 5G cellular network with broad spectrum, multiple-input-multiple-output, and ultra-dense network technologies, can realize 7.5Gbps data rate in a stationary environment and 1.2Gbps in a

mobile environment with a vehicle at speed of 100km/h, based on 28GHz spectrum [8].

Based on 3GPP Release 14, [8] argument that, even with advantages of LTE-V in 5G, the most feasible approach is to fuse different access technologies (e.g., IEEE 802.11p and 5G) to maintain the scalability and flexibility. For the authors, 5G lacks support for V2V communications, as the cellular networks were initially designed for mobile broadband traffic.

The study conducted by [18] has shown that LTE-V can represent an alternative to IEEE 802.11p due to its improved link budget, the support for redundant transmissions per packet, different sub-channelization schemes, and the infrastructure assistance under mode 3, where the cellular network selects and manages the radio resources used by vehicles for their direct V2V communications. However, the distributed scheduling designed for LTE-V mode 4, where cars autonomously choose the radio resources for their direct V2V communications and can operate without cellular coverage, is not collision-free and requires a careful configuration of the transmission parameters, in particular for autonomous driving applications that require vehicles to transmit more pps [18].

Thus, it is clear that the 5G, through the LTE-V, brings a significant evolution concerning IEEE 802.11p for vehicular communications and the research indicates that it will be the technology of the future. However, it is possible to verify that since it is a new technology, practical tests are lacking evaluating its performance in vehicular environment.

Until now, the aspects addressed in this work have been about IEEE 802.11p and 5G networks concerning communications at the level of radio technologies. Although these aspects are highly relevant when studying how the network supports the applications, there are other components directly involved in the network core.

5G goes beyond Radio Access Networks (RAN), since it is designed to use several technologies such as Software Defined Networks (SDN), Mobile Edge Computing (MEC), and Network Functions Virtualization (NFV) to attend the requirements of mobile applications. With these technologies, it provides the basis for upgrading existing platforms (e.g., 2G, 3G, 4G, and WiFi), using some LTE building blocks, with greater coverage, availability, and density, to meet the needs of various stakeholders that will benefit from service and applications [3].

To meet the requirements of different applications, 5G brings the concept of Network Slicing, where it splits a single communication network into slices that meet specific requirements. Figure 2.2 illustrates network slicing where the components in a same infrastructure were divided into slices to meet demands of three use cases: vehicles, sensors

and smart-phones. In the same way, an mobile network can be divided to meet the requirements of different vehicular applications.

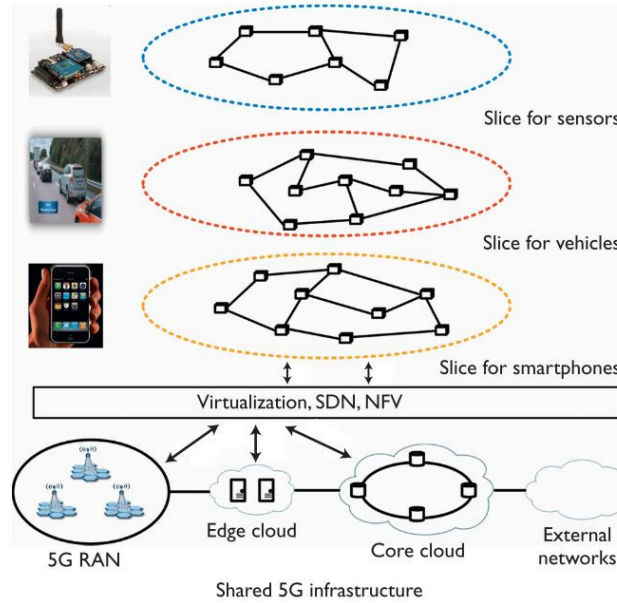


Figure 2.2: Network slicing in 5G. [6]

With network slicing, it is possible to customize the slices for diverse and complex 5G communication scenarios [6], to cope flexibly with different network applications. Thus, this topic has become one of the research focus in the network communication field and can also meet Internet of Vehicles (IoV) requirements on ultra-low delay and high reliability and other specific applications [2].

As cited by [4], with network slicing several network instances, each devoted to a specific set of services, have co-existed in the same infrastructure in order to satisfy the requirements of current and future mobile applications. The authors affirm that the agenda for 5G networks is to achieve this mainly via network virtualization, thus creating on top of the physical infrastructure a set of logical networks to meet the needs of different service providers.

The NFV, MEC, and SDN are known as 5G key technologies that are used to implement network slicing and, in this way, are essential for meeting the requirements of applications supported by a 5G mobile network.

Research in 5G components generally uses a Management and Orchestration element (MANO) that is a component that is responsible for orchestrating the devices in the network to create the slices. As informed by [23], the exact form that the network slicing MANO entity should have is still unclear with different works presenting different ideas.

Due to the limitations of the vehicles processing, the servers in the cloud process part of the data collected from its sensors and the requisitions to some applications. These servers can be Virtual Machines or Containers that host Application Servers. The connection of the remote cloud with the RSUs used by vehicles is through the backbone of the city.

In big cities, the backbone interconnects infrastructure components via optical technologies, such as PON (Passive Optical Networks). Present PON standardization is working on 25 Gbps line rates as an add-on solution for the deployed ODN (Optical Distribution Network) [24].

Despite of 5G LTE-V networks with higher transmission rates and lower delay and the backbone of the cities interconnected with links of the order of Gbps, some applications, such as those intended to avoid accidents or still related to autonomous steering, have ultra-reliable low-latency communication requirements and thus can not be subject to congestion at the core of the network. Security applications, for example, require low latency (100ms), while automotive vehicles require ultra-low latency (1ms) [3].

MEC technology is used to reduce the overall network delay [8] with the MEC cloud servers that are hosted in each RAN, directly connected in the Base Station, providing low delay responses to client requests and facilitate deployment of latency-sensitive services. Figure 2.3 shows as architecture using MEC.

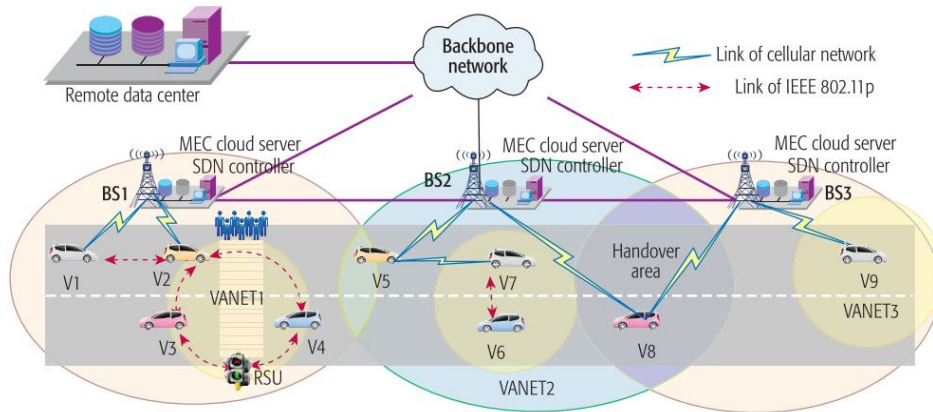


Figure 2.3: An architecture of Vanets using MEC. [6]

The use of containers offers a lightweight virtualization solution that allows a portable runtime of MEC services, which is particularly useful for mobile users [25]. As a real option that MANO can use when orchestrating application servers, Docker¹ is the most prominent container solution to facilitate an edge computing environment [26].

¹<https://www.docker.com/>

Despite the MEC benefits, it is valid to note that, mainly for reasons of cost and scalability, the best approach is not all servers applications in the Edge. Some tasks related to non-delayed applications must to be processed in the remote cloud, since the hybrid approach is the most rational [2].

To interconnect the various elements of the network, 5G can dynamically create new network functions through NFV. NFV decouples specific network functions from dedicated and expensive hardware platforms to general-purpose commodity hardware [6], allowing the collocation of multiple instances of network functions or services in the same hardware, residing within a single VM or distributed across various VMs over a cloud infrastructure [25].

In this way, some recent works proposes the use of 5G to meet the demands of different applications through network slicing, and the control of the network communications via SDN with the required MEC and NFV components in each slice.

2.5 Software Defined Vehicular Networks

As cited in the previous section, SDN is one of the key technologies in 5G networks. Some frameworks propose the use of SDN to allow greater flexibility and better deal with the challenges in a vehicular environment. Vehicular networks that use SDN are commonly named as SDVN (Software Defined Vehicular Network).

In 2014 [27] proposes the first SDN architecture in VANETs. Researchers agree that while SDN provides some level of control over network resources and management, some issues such as heterogeneity, mobility, and flexibility still need to be further explored. These issues need to be adequately addressed to meet the communication demands of modern vehicular applications.

It is worth to note that although 5G brings network slicing as a concept that can offer many benefits in meeting the requirements of applications supported by the network, in 2016 [28] already mentioned the use of network slicing via SDN to meet the needs of different applications. However, to date, there is still a lack of details and studies on how to best apply this technology in a vehicular environment.

In Figure 2.4, where is showed the SDVN architecture proposed by [28], it is possible to observed that in the data plane are data routing devices such as vehicles and RSUs, which can use different wireless technologies, since SDVN solution is independent of the access technologies.

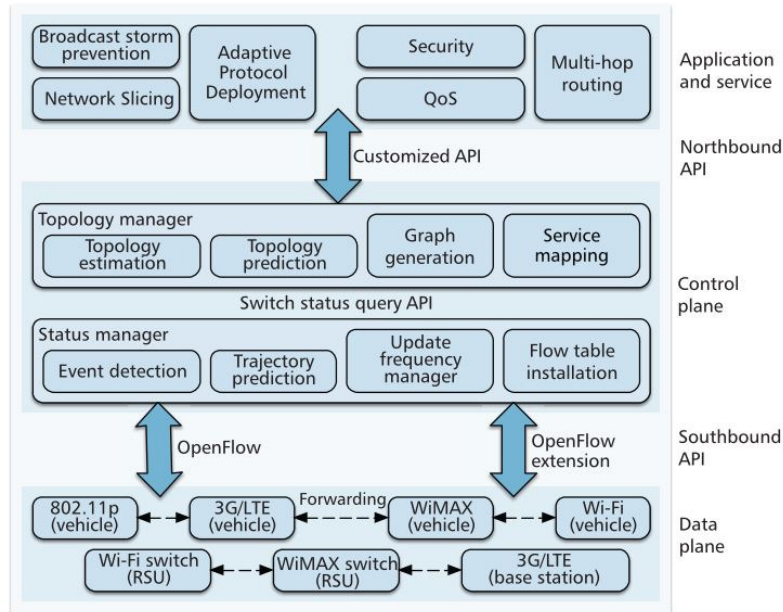


Figure 2.4: SDVN system architecture [28]

In the control plane are the elements that use several algorithms that can take into account informations such as the movement of the vehicles to define how the data should be forwarded by data plane. The communication between controller and the devices in data plane occurs through an Southbound API (SBI).

OpenFlow protocol is a standard SBI widely used. Each device in data plane has tables where can be installed flows that will define the data forwarding. The tables are identified by an 64 bits number known as datapath-id. Some works propose expansions of OpenFlow to deal with specific requirements of the vehicular networks.

Above the control layer, there is the application layer, with applications that can be built to program the network to meet the most varied demands, reaching the benefits offered by the adoption of SDN. An application represents a client entity that can request some services from the SDN controller, which usually involves data or modifications in the data plane, making the network management more flexible [8]. This interaction is made via the Northbound API (NBI). There is not yet a standard NBI. As an example, the Ryu Controller² has a REST (Representational State Transfer) API that is used to receive JSON (JavaScript Object Notation) instructions.

In the Figure 2.5 there is an image extracted from [1], which contains the components of an SDVN. It is possible to note that there are two types of controllers, an SDN controller and other RSU controllers, which are directly connected to the RSUs of the network.

²<https://osrg.github.io/ryu/>

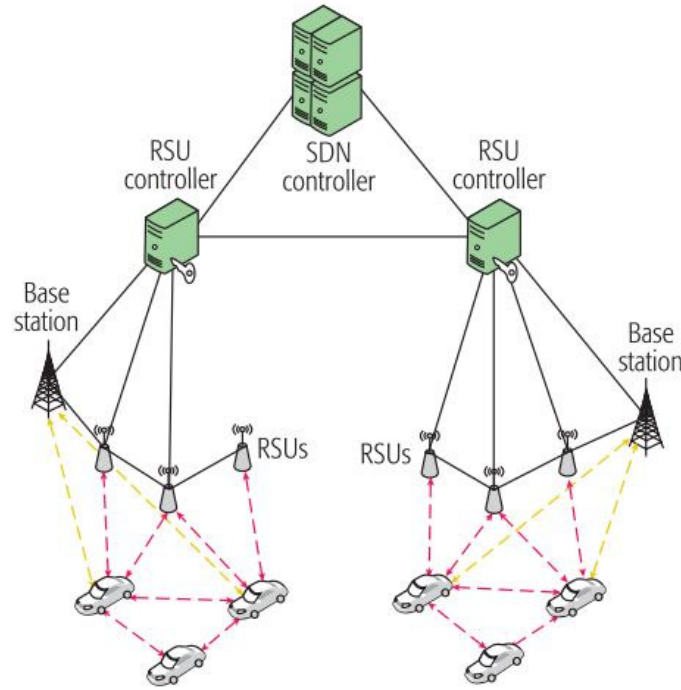


Figure 2.5: The components of an Software Defined Vehicular Network [1]

2.6 Related Works

The authors in [30] cite that the general idea of directing network behavior to meet application demands with efficient resource utilization is not new and has already been considered in the dense literature related to QoS provisioning. However, these works handled to (and hence constrained by) legacy computer networks with distributed control.

They introduce the concept of Application-Driven Network (ADN), which are networks capable of providing data paths to meet application communication requirements dynamically while using network resources efficiently, in a way that changes may occur in demands and the network reorganizes in real time.

The proposed solution can adapt to meet the varying Quality of Service (QoS) needs of applications and uses an idea of getting the application needs explicitly expressed and submitted by an DDS (Data Distribution Service)³ middleware or by an application agent, or even inferred via DPI (Deep Packet Inspection) based traffic classification techniques. In contrast to previous works, this relies on a flow-based forwarding at the SDN substrate that combines layer 2 to layer 4 packet headers.

Although the authors work to meet the requirements of applications using SDN, pro-

³<https://www.omg.org/omg-dds-portal/>

viding an architecture with control algorithms and a proof of concept related to vehicle driver training, the solution does not consider specific aspects of the vehicular environment, such as the vehicles in RSUs.

In evaluation strategy they implement the algorithms on top of the Floodlight SDN controller operating with Mininet emulator⁴ and OFSoftSwitch13⁵ network nodes but did not evaluate the results according to the traffic dynamics in a vehicular network. Additionally, they only make the comparison with a solution that uses Shortest Path heuristics based on the bandwidth of the links and not optimizes the resources according to the different applications demands.

The authors acknowledge that the solution has limitations on scalability and declares that the focus is on real-time targets or business-critical applications in an enterprise or campus networks, where scalability is not the primary concern.

[29] proposes a framework called D2-ITS to dynamically distribute the control in an SDN network and enable the dissemination of messages in ITS. The solution uses a distributed control plane based on a hierarchy of controllers to dynamically adjust environmental and network conditions to meet ITS application requirements.

The benefits of the solution are demonstrated through simulation via NS-3 Network Simulator⁶, using the Libfluid SDN controller⁷ and the traffic of 10 vehicles is generated through SUMO mobility Simulator⁸, resulting in lower latency with minimum overhead when compared to a centralized control approach. In their proof of concept, the proposed solution is more efficient because the nodes do not send the messages directly to the central controller, connected via LTE, with bigger delay.

The solution uses just OpenFlow throughout the network (including experimenter messages), even for the analyzed vehicular traffic service, related to Road Hazard Warning application. The authors do not address how the controllers receive the information about the application requirements and do not inform the criteria to define local controllers.

In [5] the authors analyze the potential of Software-Defined Vehicle Networks, emphasizing the need to rethink the traditional approach from a theoretical and practical point of view in this application context. An emulation approach based on the *node car* of the Mininet-wifi emulator⁹ is presented to show the applicability and some benefits of

⁴<http://mininet.org/>

⁵<https://github.com/CPqD/ofsoftswitch13>

⁶<https://www.nsnam.org/>

⁷<http://opennetworkingfoundation.github.io/libfluid/>

⁸<http://sumo.dlr.de/index.html>

⁹<https://github.com/intrig-unicamp/mininet-wifi>

the SDN in a selected use case scenario.

In the performance evaluation, the controller modifies the vehicle flows as the topology changes to reach the best data routing in a scenario with four cars communicating with a remote client consuming UDP ITS video monitoring application through 3 RSUs.

Despite the paper shows an emulation solution that enables realistic Vanets experiments with SDN, the proposal does not inform in the use case scenario how the controller detects the topology changes. Other point is that in the decision-making process the controller do not considers application requirements.

In [17] the authors present a solution with Fog computing to support some vehicular applications. Fog computing is another solution that like MEC brings the servers closer to client devices. They inform that this technology, especially in vehicular environment, is still in its early stage, with many unresolved and under-explored technical and operational challenges.

The authors provide an example of traffic control service in a vehicular network clearly defining the roles of local and remote nodes. Nearby nodes (RSU Fogs) send statistical data to a central server and receive processed cloud data to assist in controlling local vehicular traffic through the programming of traffic lights. At the central cloud is done the processing for traffic planning throughout the city.

The paper presents results of statistical calculations regarding the detection of Fog nodes committed for forensic security, based on random traffics generated with SUMO and Openstreet Maps. While addressing the importance of Fog nodes and citing state of the art, the proof of concept was not made using the solution to meet specific vehicular applications requirements.

[31] proposes a 5G architecture for VANETs, using SDN and Fog Computing with the network control divided hierarchically. The authors use MATLAB to evaluate scenarios between 10 and 60 vehicles, comparing the results between the traditional architecture, a 5G VANET architecture and their proposal.

The transmission delay, throughput and overhead are analyzed and showed lower delay and control overhead for different vehicle densities with the proposed solution using an adaptive bandwidth scheme.

The paper does not go into evaluated applications details nor vehicles mobility, do not use OpenFlow nor a realistic SDN controller, and do not provide message exchange details or algorithms running on the components.

In [32] the authors propose a solution that addresses the QoS that the applications need to meet the users' requests, according to the network status.

The authors did a systematic survey of related works and proposed a solution that works with the concept of the cluster in a way that the network does not depend on only one controller. It was not used OpenFlow as Southbound API since the authors propose their own API, named HSDV. When the local controller loses connectivity with the central unit, it assumes the central role for its coverage area.

The proof of concept is done via simulation in the NS-3 with traffic generated by SUMO and in the performance analysis the packet delivery rate, throughput, end-to-end delay and routing overhead are compared, showing that the proposed solution has better results than traditional routing protocols (AODV, DSDV, and GPSR). The authors did not show how the modules work and this proof of concept goes far from the proposed solution since it deals only with routing.

[8] proposes an SDN-enabled network architecture assisted by MEC, which integrates different types of access technologies, to provide low-latency and high-reliability communication, using a heterogeneous network with 5G and IEEE 802.11p.

The authors did not show how the network differentiates the applications so that the solution can meet the specific requirements. Despite stating that with the proposal the network configuration and the routing policies can be updated appropriately, according to the fast context variations, they did not show how this is possible in practice.

In [33] the authors present a VCC (Vehicular Cloud Computing) platform with support for 5G communications, to provide effective vehicular cloud services. As a case study a taxi sharing service supported by the proposed solution is shown.

The simulation was poor realistic once, to represent the advantage of future 5G communication technologies, the authors consider the test network in two situations, i.e., the network without any loads, and the network with loads (512kbps), to approximatively indicate the difference between future 5G communications and existing communication technologies. The paper did not show several implementation details (e.g., scenarios, topology, number of vehicles and subscribers).

In [22] the authors aim to address the issue that existing technologies for VANETs are not able to solve the inherent challenges of supporting vehicle applications in such a dynamic environment. They use an approach of heterogeneous networks, proposed through the use of SDN.

In the paper are presented proposals related to the management of resources in the evaluated network to deal with several problems, such as the efficient allocation of radio channels and interference in environments with heterogeneous access networks. There is no proof of concept, and they do not make clear the difference between vehicular applications and network control applications, and even how is the interaction between them.

Given the analysis of related works, it is possible to verify that several open points need to be further explored and, as shown in previous items, some solutions deal with these topics partially, but none of them address them in conjunction with the properly approach.

3. A Framework for Application-driven vehicular networks

In this chapter, we present the solution proposed in this work, which consists of a Framework for the implementation of vehicular networks that dynamically act to meet the communication requirements of different ITS applications. This solution can be classified into the category of SDVN, because of SDN, or even Vehicular Cloud Computing [34], because of MEC and central clouds.

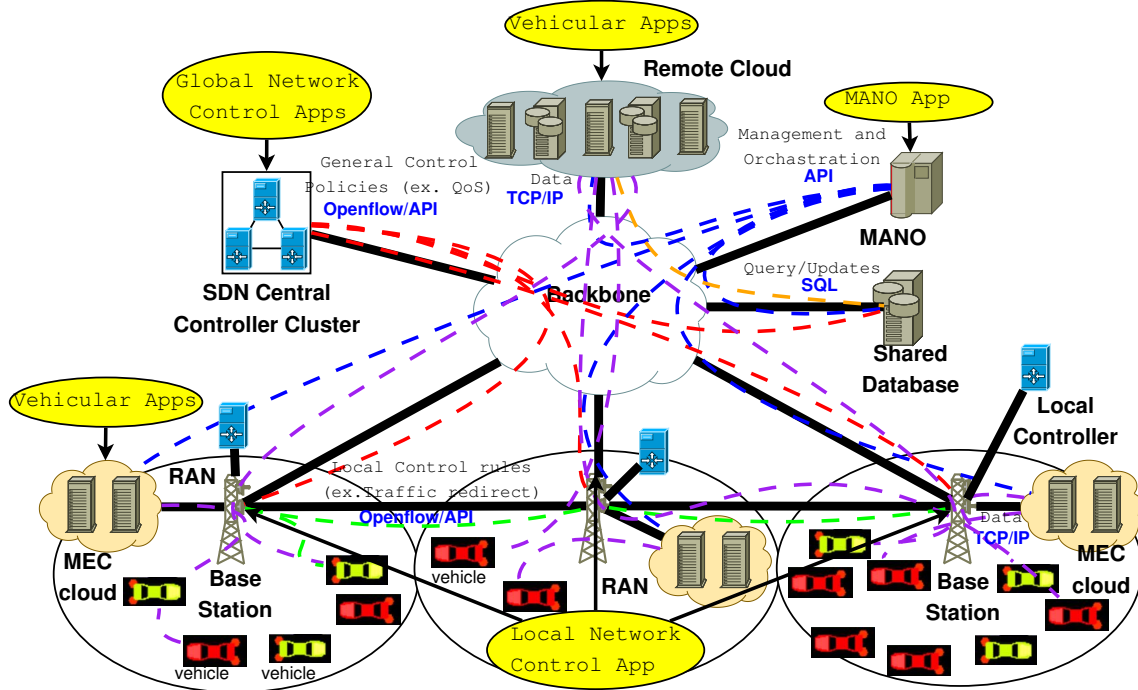


Figure 3.1: Architecture of proposed framework

The proposed architecture is composed of the elements that make up a traditional vehicular network, such as vehicles and RSUs, working together with some key technologies used in 5G network slicing. The radio elements are not part of the solution, despite we note that 5G radio technologies like LTE-V represent the future of vehicular radio communications. The Figure 3.1 contains an image of the components of the architecture and how they are interconnected.

The central and local controllers share the role of controlling SDVN elements. There is also the shared database, which is used by control algorithms to information sharing with vehicular applications and MANO, that orchestrates the infrastructure resources to meet applications demands. Remote cloud hosts some servers, including application servers, while local clouds host other components like MEC servers for applications that are delay sensitive.

Following an organization similar to the used in the SDN model, where there are layers of application, control, and data, the framework's architecture can be represented as shown in Figure 3.2. The details of the purpose, the way of acting and the relationships between the components in each layer are described in the sub-items of this chapter.

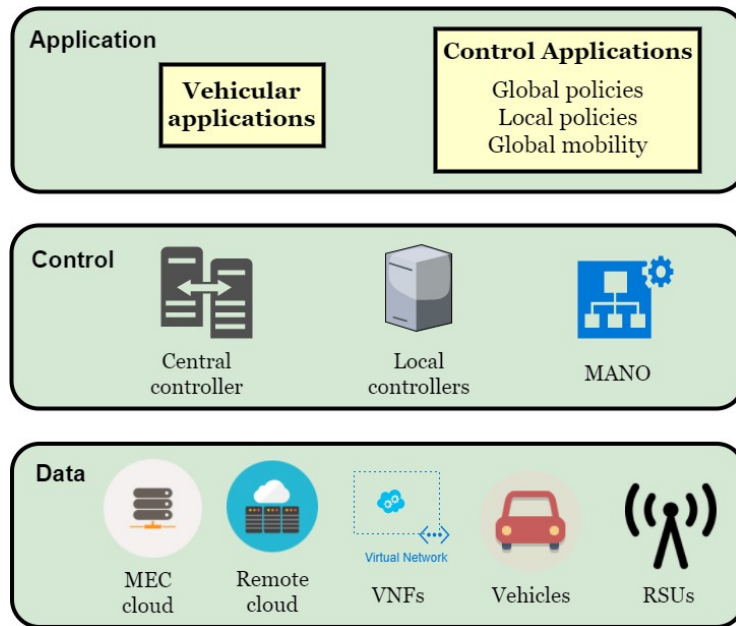


Figure 3.2: Abstraction layers of proposed framework with correspondent components

In the proposed architecture, we adopted from other works the concepts of hybrid (also known as hierarchical) and distributed architectures, and the use of MEC to attend applications with strict delay requirements.

In hierarchical control, the control functions are divided into more than one layer. With this approach it is possible to obtain lower levels of latency overhead [29]. There are yet some other advantages, like an optimized control, based in locations with different scenarios [22].

Because it plays a fundamental role in the network, the controller often becomes a vulnerable point, since if it fails the control features are impacted. In this way, several approaches employ distributed control, where controllers at the same level and with the same assignments act on a redundancy scheme, to ensure high availability.

Regarding technologies for the dynamic allocation of resources across network slices, a variety of solutions have been proposed and are well understood [4]. Despite this, there is a lack in providing it to meet vehicular environment requirements. Thus, this work contributes with this point through the proposed framework, that is composed by the shared database and the algorithms to manage its information and calculate the balance of resources in each radio access network, to orchestrate the resources to meet the requirements of each vehicular application slice.

3.1 Application plane

As described earlier, vehicular ITS applications need a vehicular network infrastructure to support its communication requirements. These applications can be for example of vehicular traffic control, steering safety and entertainment. Each one of those has their communication KPIs as, for example, delay and bandwidth.

Although the purpose of the proposed framework is to support vehicular applications of ITS systems, it is worth to note that in an SDN network there are other types of applications, which are the control applications that define the network behavior. In this way, these two types of applications (vehicular and control) are part of the application layer in the proposed solution.

In some related works, vehicular applications interact directly with network controllers to have their communication demands met. In a large and complex network, supported by several operators and maybe with different controllers with their APIs, this is not the best approach. In our approach, the controller devices and vehicular applications share the relevant information through a shared database using a standard DML (Data Manipulation Language), like SQL (Structured Query Language). So, the control level becomes transparent to vehicular applications that can be agnostic to control functions, focusing on providing their services.

The use of the shared database has an additional advantage, once it allows the persistence of a valuable source of information that can be used to feed several algorithms that optimize city's transportation systems aspects. Besides we use as the example a relational database, there is no limitation to use non-relational ones, to manage non-structured data and implement Big Data solutions.

Each vehicular application in a smart city will have its algorithms to accomplish what they are designed to do. However, each application will have at least one common algo-

rithm to update relevant information, like that about its service subscribers, in the shared database. This generic App feature is in Algorithm 1.

Algorithm 1 Generic App Algorithm - subscription vehicles update

Input: VS, VU, DA, MA, KPI

Output: DU, MN

- 1: Begin
 - 2: DU(DA, VS, KPI, VU)
 - 3: MF \leftarrow 1
 - 4: NM(MA, MF, AID)
 - 5: End
-

To execute the algorithm, application servers will need permission to write and update its information in the shared database. With the database address (DA), application servers, using the database connection and a DML language, proceeds with a database update (DU) to register new vehicle subscriptions (VS), unsubscriptions (VU) or even some KPI change. After this, the application notifies MANO (NM), that receives a flag informing if the notification is or not about a modification (MF) and the ID of related application (AID). Next MANO will proceed to verify the necessary infrastructure re-configuration in function of updates and notify SDN controllers. With these actions, the control components identify the changes in application scope and check if it needs to reflect in infrastructure changes to keep meeting App KPIs.

Related to control applications, we propose four: Global Network Control App, Local Network Control App, Global Mobility App, and Management and Orchestration App.

The Global network control App is responsible for defining the network policies that will be applied by the central controller to the network to forward data related to vehicular applications, prioritizing them according to its KPIs and available resources. Since the vehicles are always moving in each RAN, that has their limited local resources, the local network control App manage the local resources, reinforcing the global policies of central controller, in function of the vehicles in each RAN and the resources in its neighbors, for example, restricting or redirecting some communication flows. As it will be discussed in Section 3.3, this approach of heterogeneous control has some advantages when compared with single centralized control.

Management and orchestration App is responsible for deploying the necessary infrastructure to support each application, based on its requirements and the vehicles associated. In the last analysis, it instantiates the components that are in the network slices and will communicate following the network policies defined by the SDN controllers.

Since vehicles are moving and could pass through different geographic regions with

different RANs and distances of kilometers, the resources deployed by MANO can't be static. Global mobility App receives kinematic data from vehicles and updates its region in the shared database. With the knowledge of the cars in each region, the solution can optimize the resource deployment dynamically.

Despite the global network control, global mobility, local network control, and management and orchestration be applications, their algorithms are detailed in Section 3.3, where the control elements are described, once that these applications that define how must be the execution in the devices in control plane.

3.2 Data plane

As already mentioned, in an SDN network the elements in the data plane are that who forward application data packets, that is a role executed mainly by the switches. In our framework, the switches in network core are part of the data plane and receive switching rules from the controllers, forwarding packets in according to them, to prioritize vehicular applications data properly.

Each RSU can switch packets between the vehicles in its RAN, and the devices in the MEC and remote clouds. To do this, each RSU has an internal switch that also receives instruction flows from the controllers, like any other switch. In this way, the RSUs are also part of the data plane.

Vehicles generate the client traffic related to the vehicular applications. When deciding the best way to route the traffic, for example when in the range of more than one RSU, the best path could be defined by a flow installed by the controllers in the vehicles datapath. So, vehicles also are part of the data layer.

MEC and remote clouds are hosting the applications servers that receive messages from vehicles and provide responses. Also, the VNFs instantiated by MANO will be hosted in these clouds and their behavior when switching packets defined by the controllers. So, these components are also part of the data plane.

Some references, like the study for deployment of 5G in Europe by the 5G-PPP Automotive Working Group¹, do not consider V2V in the provision of services. Despite this, we believe that this type of communication is necessary to extend the coverage of infrastructure access to cover unattended vehicles and in services as, for example, platooning and cooperative driving.

¹https://5g-ppp.eu/wp-content/uploads/2018/02/5G-PPP-Automotive-WG-White-Paper_Feb.2018.pdf

Some works deal with resource allocation at RAN level. As mentioned earlier, our solution doesn't deal with the radio part. Solutions can be incorporated to integrate the elements of radio communication in data plane enhancing the proposal and in this way contributing to the overall goal of meeting the application requirements.

3.3 Control Plane

Since the objective of the proposed framework is to meet applications requirements in a dynamic environment, it is necessary to define when the network leaves a stable state, that is when the infrastructure is attending applications KPIs, and what the components do to keep attending these applications requirements.

As illustrated in Figure 3.3, the network can leave the stable state by five ways, that results in a transitory group of actions executed by the control components, to bring the network to steady state again.

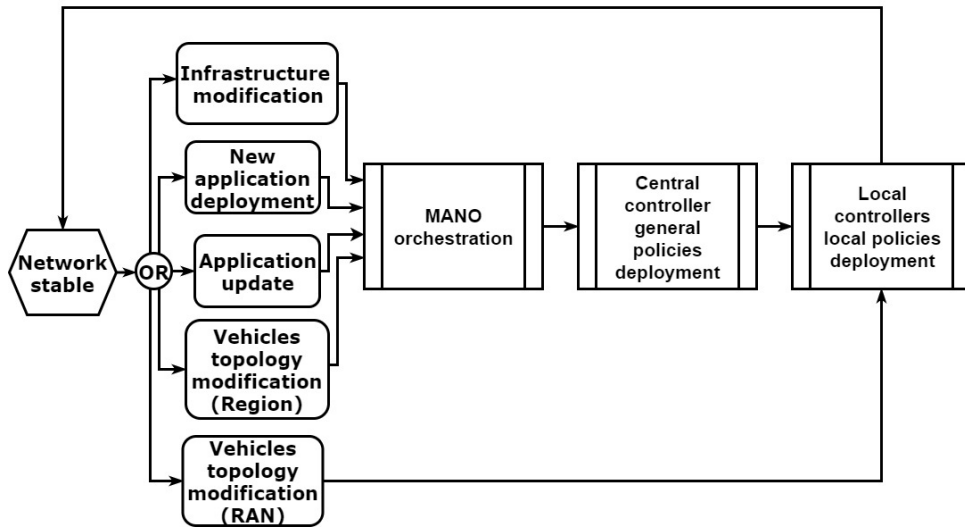


Figure 3.3: Flow chart of the framework in function of changes in vehicles topology, applications or infrastructure

Some of the possible occurrences require that MANO verifies the infrastructure to analyze if it is necessary to optimize the resources again. When this occurs, MANO notifies central controller to check if it is necessary to compute and apply new global policies. Once central controller finalizes its actions, it informs local controllers about the last policies, and they implement new local policies, if necessary, to reinforce the global ones.

This sequence of activities occurs when there is some modification at the infrastructure level, as, for example, the deployment of new base stations in the city, the implementation

of a new vehicular application, applications updates or also when there are modifications at topology in regions level with, for example, vehicles moving to another city.

There is another situation that occurs, generally with more frequency, and also requires actions to the network to get back to the stable state. This situation derives from the natural dynamics of the vehicular environment, where vehicles can move through the range of different base stations, modifying the network topology at the RANs and in some cases resulting in high utilization resources. In this case, local controllers detect the modifications and act reconfiguring local policies, for example, redirecting excess traffic to a non-congested neighbor RSU or, when it is not possible to redirect traffic, send it through the predefined path, limited in according with QoS applied via global policies.

The way that MANO and controllers act is defined by algorithms that will be shown in the next subsections.

3.3.1 MANO (Management and Orchestration)

MANO provides management and orchestration of physical and virtual resources in MEC and remote clouds, to dynamically instantiates or migrates the necessary resources for each vehicular application, including application servers and virtual network functions. The items in Tables 3.1 and 3.2 contains respectively inputs and outputs examples of Algorithm 2, that defines how MANO acts in our proposal. Figure 3.4 illustrates MANO actions when configuring a new application.

Table 3.1: MANO algorithm - Inputs and Examples

Input	Examples
Application Infrastructure Resources (AIR)	Network Resources (e.g., Network Interfaces, DNS register, NTP definition), Servers (e.g., Container, HTTP, FTP)
Infrastructure Configuration Informations (ICI)	Container image URL, IP Address allocation
Vehicles Subscription Info (VSI)	Vehicle_ID, Application, Region
Application Requirements (KPI)	E&E Latency, Reliability, Data Rate
Application Identifier Code (AID)	Unique code (#CCAS1, #4KLV1)
Modify Flag (MF)	0 or 1
Database Address (DA)	URL or IP Address
Central Controller Address (CCA)	URL or IP Address

The MANO algorithm uses the MF control variable to differentiate if the parameters received as input are related to a new application ($MF = 0$) or not ($MF \neq 0$). When deploying a new application, the stakeholders of the city that offers services need to submit

Table 3.2: MANO algorithm - Outputs and Examples

Output	Examples
Central Controller Notification (CCN)	Generic Notification with MF and Applications related (AID)
Deploy Infrastructure(DI)	Container and VNFs deployment and configuration
Database Querys (DQ)	SQL Shared Database All Tables Query Updates
Database Updates (DU)	SQL Shared Database Servers Table Update

to MANO some fundamental information, besides $MF = 0$, that are AIR, ICI, VSI, AID, and KPI. MANO also needs to verify the current status of the elements in infrastructure under its domain, so it defines a DS (Database Status) variable to store the values that it will query from the shared database.

MANO uses the data returned in the database query in conjunction with AIR, ICI, VSI, and KPI to compute the necessary resources to meet application KPIs, using the CR function. As stated in subsection 3.2, MANO can use MEC solution, for example instantiating a Docker container, to meet requirements of delay sensitive applications, if the KPI of E&E latency is lower than a specific threshold.

After the resources are computed, MANO will proceed with the necessary actions to deploy them at infrastructure, using DI function. After that, it updates the information related to the new application in KPI, vehicle and servers tables in the shared database. Next, MANO notifies the central controller, sending the value of MF and the ID of the App deployed (AID), to controller proceed as described in subsection 3.3.2, to implement necessary global network policies. Once finished the deployment, MANO query from the shared database the more updated data in all tables and store locally in DS variable, to use in future cases where $MF \neq 0$.

If MANO receives $MF \neq 0$, this is the case of an update that can be triggered by an application that update in shared database its information or the detection of a topology modification at region level, detected by global mobility application, as defined by Algorithm 5, or also related to some modifications at infrastructure level as already discussed in this chapter. In these cases, MANO defines a variable named NDS (New Database Status), to store new data values, and other variable DC to save data changes. After query the new status and store in NDS, MANO compute the changes, using $CC(NDS, DS)$ function, and store in DC. With previously and difference data, MANO computes resources to the new scenario and after this deploys them. Once the resources were deployed/opti-

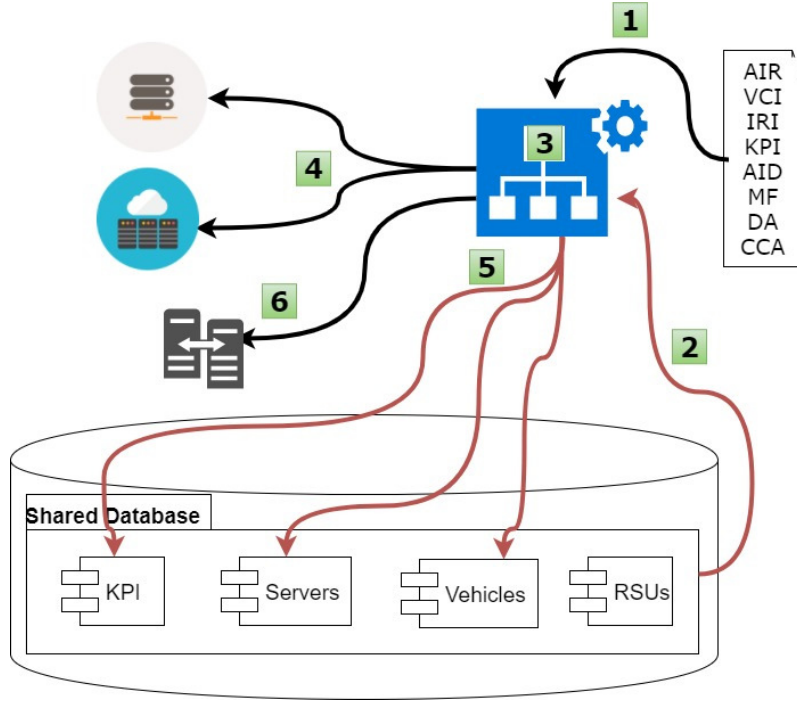


Figure 3.4: MANO actions enumerated by execution order, where 1 are the feed informations, 2 the database query, 3 the heuristics to define new infrastructure, 4 the infrastructure deployment, 5 the update in shared database and 6 the central controller notification.

mized, MANO updates this information at the shared database, update DS variable with the most updated data and notify central controller, with $MF = 1$ and the AID of application related to the updates, that will proceed with the necessary network policies optimizations.

It is valid to note that the heuristics used in functions of MANO ($CR, DI, CC(NDS, DS)$) are not in the scope of this paper and can be implemented using solutions by different technology providers. The best way to implement these functions in general solutions is in the scope of several recent articles. Our purpose here is to show how to integrate these functionalities in the proposed solution to reach the listed contributions. In the framework evaluation, we implement a simplified version of these functions, that are enough to show the benefits of this proposal.

3.3.2 Central Network Controller

Some SDN architectures rely on a logically centralized control plane or make use of a distributed control approach, both of which are not compatible with ITS scalability, geographic decentralization, and latency requirements [29]. In the proposed framework, the control layer is composed of two levels of SDN Control (central and local), in a heterogeneous (also known as hierarchical) approach, to reach more precise and agile

Algorithm 2 MANO (Management and Orchestration)

Input: AID, AIR, ICI, VSI, KPI, MF, DA, CCA**Output:** CCN, DI(), DQ, DU

```

1: Begin
2: if MF = 0 then
3:   DS, RTI  $\leftarrow \square\square$   $\triangleright$  Define variable Database Status and Resources to be
   Implemented
4:   DQ(DA, Alltables)  $\rightarrow$  DS  $\triangleright$  Query All tables
5:   CR(AIR, DS, ICI, VSI, KPI)  $\rightarrow$  RTI  $\triangleright$  Compute resources and store in RTI,
   using MEC if KPI( E&E Latency ) < threshold  $\triangleright$  Use MEC if KPI( E&E Latency ) <
   threshold
6:   DI(RTI)  $\triangleright$  Deploy Infrastructure using RTI computed
7:   DU(DA, KPI, Vehicle, Servers, AID)  $\triangleright$  Update KPI, Vehicle and Servers
   tables in shared database
8:   CCN(CCA, MF, AID)  $\triangleright$  Notify Central Controller
9:   DQ(DA, Alltables)  $\rightarrow$  DS  $\triangleright$  Store locally in DS the last version of all data
10: else
11:   NDS, DC  $\leftarrow \square\square$   $\triangleright$  Define New Database Status and Database Changes variables
12:   DQ(DA, Alltables)  $\rightarrow$  NDS  $\triangleright$  Query All tables
13:   CC(NDS, DS)  $\rightarrow$  DC  $\triangleright$  Compute changes
14:   CR(DC, NDS)  $\rightarrow$  RTI  $\triangleright$  Compute resources based on changes
15:   DI(RTI)  $\triangleright$  Deploy infrastructure using RTI computed
16:   DU(KPI, Vehicle, Servers)
17:   DQ(DA, Alltables)  $\rightarrow$  DS  $\triangleright$  Store locally in DS the last version on all data
18:   CCN(CCA, MF, AID)  $\triangleright$  Notify Central Controller
19:   MF = 1
20: End

```

Algorithm 3 General Policies Central Controller

Input: DA, MF, AID**Output:** DQ, AGP, LCN

```

1: Begin
2: if MF = 0 then
3:   DS, GP, ARL, AVL, BSI, IAL  $\leftarrow \square\square$   $\triangleright$  Define database status, general policies,
   ARL, AVL, BSI and IAL variables
4:   DQ(DA, Alltables)  $\rightarrow$  DS  $\triangleright$  Query all tables in shared database and store in
   DS
5:   CL(DS)  $\rightarrow$  ARL, AVL, BSI, IAL  $\triangleright$  Create ARL, AVL, BSI and IAL lists
6:   for all App in ARL do
7:     DS(App, AVL, IAL, BSI)  $\rightarrow$  SDL(AID)  $\triangleright$  Create slices and store data in
     SDL
8:     Compute_GP(SDL(App), ARL)  $\rightarrow$  GP  $\triangleright$  Compute global policies to meet
     KPIs in ARL
9:     AGP(SDL(AID), GP)  $\triangleright$  Apply general policies on slice
10:    LCN(BSI(Address), SDL(AID), AID, ARL)  $\triangleright$  Send to local controllers
    (in scope regions) slices and ARL.
11: else  $\triangleright$  It is an Update
12:   NDS, DC  $\leftarrow \square\square$   $\triangleright$  Define database changes and new database status variables
13:   DQ(DA, Alltables)  $\rightarrow$  NDS  $\triangleright$  Query All tables in shared database
14:   CD(NDS, DS)  $\rightarrow$  DC  $\triangleright$  Compute changes and store in DC
15:   UL(DC, ARL, AVL, BSI, IAL)  $\rightarrow$  ARL, AVL, BSI, IAL  $\triangleright$  Update lists based on
   changes
16:   for all App in ARL do
17:     US(AID, AVL, BSI, IAL)  $\rightarrow$  SDL_n(AID)  $\triangleright$  Update App slice according
     to new lists
18:     CNGP(SDL_n(AID), ARL)  $\rightarrow$  GP  $\triangleright$  Compute new General Network
     Policies
19:     AGP(SDL_n(AID), GP)  $\triangleright$  Apply general policies
20:     SDL(App)  $\leftarrow$  SDL_n(App)  $\triangleright$  Update Slice App variable to new version
21:     LCN(BSI, SDL(App), App_id, ARL, GP)  $\triangleright$  Send to controllers in affected
     areas
22: End

```

control actions. With a hierarchical network control approach, it is possible to obtain lower levels of latency overhead in the network [29]. There are yet some other advantages of the hierarchical approach, like an optimized control based in locations with different scenarios [22].

The central controller is a cluster composed by some controllers at the same level, in a distributed approach, to provide high availability to the overall solution. Since this cluster is a logical entity formed by several devices, it is necessary an algorithm to coordination among cluster components to achieve high availability. Once several papers acutely analyze this topic (cluster and high availability), it is not in the scope of this work and it will not be proposed a specific algorithm to this end.

As the flexibility of the SDN comes from the separation of the data and control planes, the central controller cluster is centralized precisely because in this way it achieves greater flexibility due to a single point of configuration that can receive information from several sources to take decisions [22].

General policies algorithm (Algorithm 3) has as input DA (Database Address), MF (Modify flag), and AID (Application ID). As output, there is DQ (Database Query) and yet $AGP(SDL(AID), GP)$ that applies in network configurations the general policies as, for example, QoS, flows, and priorities. Another output is $LCN(BSI(Address), SDL(AID), AID, ARL)$, that is a general notification with the content of each device in APP slice ($SDL(AID)$) and the necessary KPIs, organized in ARL (Application Resources List), to local controllers in each Base Station.

The general policies algorithm also uses the control variable MF to differentiate if the parameters received as input are related to a new application ($MF = 0$) or not ($MF \neq 0$). In case $MF = 0$, its because MANO finished the deployment of a new application and central controller needs to optimize networks resources, applying necessary policies meeting applications requirements. So, central controller defines a DS (Database Status) variable to store the values that it will query from the shared database.

The values returned in database query are used to create the list of applications KPIs (ARL), vehicles associated to each application (AVL), the list that contains IP or URL address and the region associated to local controllers in each RAN (BSI) and the list with the infrastructure deployed by MANO to the application (IAL). These lists are created using the function $CL(DS)$ and the content of respectively KPI, Vehicles, Base Station and Servers tables.

For each application in ARL, central controller defines a new network slice, using

DS(App, AVL, IAL, BSI) function, considering all data of AVL, BSI, and IAL. Figure 3.5 shows an example of some shared database tables with information being used to creating three slices for three applications, with the respective resources. There is a table to store the applications' KPIs, other to the associations of vehicles with the applications and the geographic region where they belong, one with information about the base stations in each region with its respectively network address and another table with the data of infrastructure deployed by MANO.

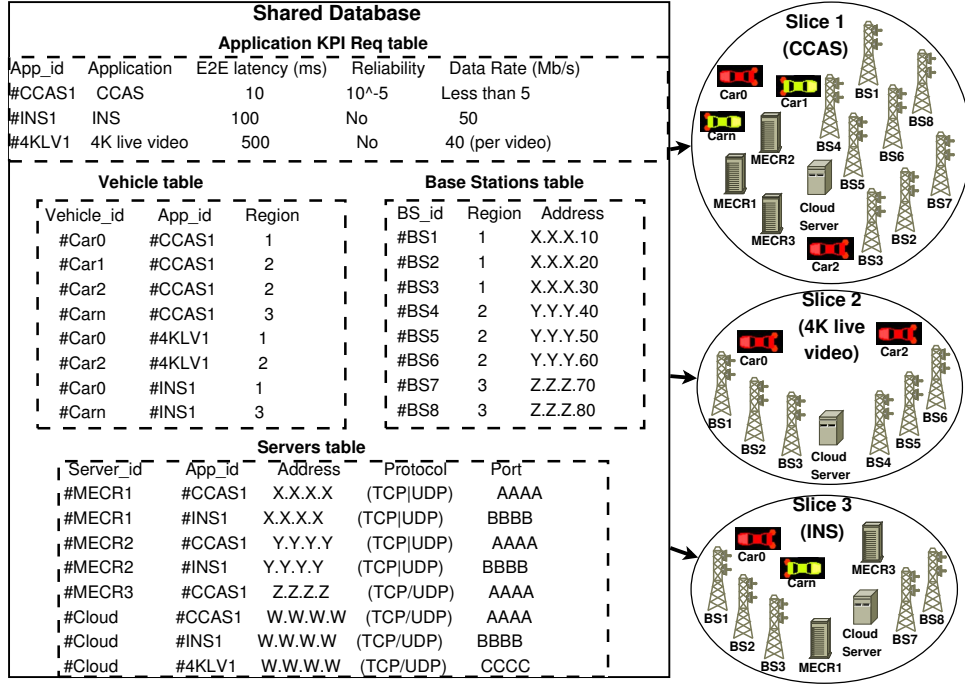


Figure 3.5: Informations in a shared database example and respectively slices

After slice creation, the central controller computes general policies for each slice, using CGP(SDL(App), ARL) function to meet applications KPIs. Once the policies were computed, the central controller applies them to the network, using AGP(SDL(App)) function and some SBI (e.g., OpenFlow). After this, local controllers in regions where there are vehicle subscribers are notified through LCN(BSI(Address), SDL(AID), AID, ARL) function, with the slice data and the respectively KPIs.

An alternative that would be evaluated, instead of send all informations to local controllers, is the central notifying the local controllers with just the AID of the related application, and local controllers query from the database the necessary additional information. The best approach depends on some factor like the throughput between local controllers and shared database and the overhead control between local and central controllers.

If central controller receives $MF \neq 0$, it needs to query the database to identify the changes, using CC(NDS, DS) function, and with these changes it update ARL, AVL,

BSI and IAL lists and the slices, compute the necessary policies to apply, and next notify the local controllers in affected regions.

The same way that was not defined the heuristics in some MANO functions, this work not deals with the implementation of $DS(App, AVL, IAL, BSI)$, $CGP(SDL(App), ARL)$, $AGP(SDL(App))$, $LCN(BSI(Address), SDL(App), ARL)$ and $CD(NDS, DS)$ functions, since each technology provider can do that of the way it thinks is better. In the section of performance evaluation simplified version of some functions are implemented to verify the feasibility of the proposed solution.

3.3.3 Local Network Controller

The local controllers work to prioritize in each RAN the traffic of applications following the global policies defined by the central controller. Due to the heterogeneity of control, in case of unavailability with the central controller, only the new global policies will be impacted and what is already working will continue.

An east-west API makes the communication between two controllers in SDN. In our framework, local controllers in each base station receive information about the idleness of neighbors' resources (for example available bandwidth in links that connect with backbone) through east-west communication. This information is used in the local control algorithm to evaluate the possibility of redirect some traffic to a neighbor in case of the local RSU can't deal with it.

The algorithm of local controllers (Algorithm 4) has as input the slice configurations for each slice in its domain ($SDL(AID)$), the list of KPIs per application (ARL), the global policies implemented by central controller, the information about KPIs that neighbors can meet ($NAKPI$), the maximum total KPI that the current BS can meet by default (TKC), the address of BS neighbors (NAL), a variable that define the interval time that local controllers do its verifications (IV), and database address (DA).

The algorithm also define some variables, that are responsible by storing the total KPI needed per application in RAN ($TKNA$), the minimum and maximum application index to applications prioritization (API , $MPAI$), the KPIs available at current BS after calc the balance in local RAN ($AKPI$), the local policy rules to redirect (LPR) and apply QoS (LPQ), the total of vehicles in RAN (RV), the identifiers of Base Stations neighbors (BS_Id), the id of applications (AID), and the value to store values related to demands treated or passed on to some neighbor (AV).

First, the controller identifies the vehicles in its RAN, using IV_RAN function and

stores in a local file the information about the resources available in neighbors. With the list of cars in RAN, the configuration of application slices and the list of requirements per application, local controllers can compute, using $CKPI(RV, SDL(App), ARL)$ function, the necessary KPIs to attend vehicles Apps in its RAN.

The value that represents the maximum of resources some RSU can deal is defined when this RSUs is deployed and is updated when it is upgraded. This value is TKC (Total KPI supported), which is used with the total necessary KPI in RAN (TKNA) and AV, to local controller verify if it can attend the requirements of vehicles in RAN ($AKPI = TKC - TKNA + AV$) or is necessary an action. AV is a negative value if there is additional requirements that the RSU is attending for some neighbor or otherwise, if it is redirecting some demand to neighbor that it doesn't can deal with this value is positive.

When the local RSU cannot accomplish the KPIs ($AKPI < 0$), the local controller verifies if the neighbors can deal with part of the demand, through $VKN(TKNA, NKPI.file, API)$ function, computing which RSU can deal with the APIs in some priority index. This function will return the AID of analyzed application and the value of the BS_id of the neighbor that can attend. So, the local controller installs local flows to redirect the traffic of the applications through $LPR(BS_Id, AID)$.

If no one neighbor can attend the additional demand, BS_id will be null, and local controller limits the traffic of this application via QoS according to Global Policies, using $LPQ(AID)$. The controller analyzes the applications by a priority order to limit the applications with lower priority possible.

It is worth to note that despite the algorithm synthesize some steps, seeming that the actions affect all flows of some application, in practice, this action is unfolded in others to controller proceed with the execution in the communication flows of individual vehicles so that first only one vehicle has it flow treated and if the situation persist the local controller proceed with the action in the flow of other vehicle in RSU, an so on, until the situation is stable or no more flows of the application exists and hence other actions will be necessary. In the implementation of the frameworks evaluation, an algorithm to do this was implemented and will be detailed. We don't write this in the local control algorithm because, like in other control algorithms, the intention is not to define the details of all functions.

If there are enough resources locally, local RSU announces extra resources to neighbors, via $SN_AKPI(NA, AKPI)$, and wait the period defined by the IV variable to recheck the RAN status. The value of IV can be determined using heuristics that considers several

variables of transportation in the city (e.g., holidays, accidents).

Just as it was with MANO and central controller algorithms, the main functions of local controller are not in the scope of this work and can be an object of future studies.

Algorithm 4 Local Controllers Algorithm

Input: SDL(App), ARL, GP, NAKPI, TKC, NAL, IV, DA

Output: LPR, AKPI, DQ

```

1: Begin
2: TKNA, API, MAPI, AKPI, LPR, LPQ, RV, BS_ID, AID, AV  $\leftarrow \square$   $\triangleright$  Define
   Variables
3: IV_RAN  $\leftarrow$  RV  $\triangleright$  Identify vehicles in local RAN
4: DQ(DA, BS_ID, redirect_table)  $\rightarrow$  AV  $\triangleright$  Verify additional values of redirect or
   traffic limitations to RSU
5: for all NAKPI do
6:   Update(NAKPI, BS_id)  $\rightarrow$  NAKPI.file  $\triangleright$  Stores locally in a file the resources
   available in each BS neighbor)
7: CKPI(RV, SDL(App), ARL)  $\rightarrow$  TKNA  $\triangleright$  Compute necessary KPIs by App in RAN
8: AKPI = TKC - TKNA + AV  $\triangleright$  Resources balance
9: if AKPI < 0 then  $\triangleright$  IF BS can't meet KPIs
10:   DQ(DA, appkpi_table)  $\rightarrow$  AI  $\triangleright$  Query informations of applications from
   from database
11:   define_API(AI)  $\rightarrow$  MAPI  $\triangleright$  Define API Max level of priority class, based on
   applications information
12:   API = 1  $\triangleright$  Prioritize first applications with lower API
13:   while API < MAPI do
14:     VKN(TKNA, NKPI.file, API)  $\rightarrow$  BS_Id, AID  $\triangleright$  verify if neighbors can
   meet API requirements.
15:     if then BS_id  $\neq$  null  $\triangleright$  Return AID and BS_ID  $\neq$  null if neighbor can deal
16:       LPR(BS_Id, AID)
17:       API+ = 1
18:     else
19:       LPQ(AID)  $\triangleright$  If no neighbor can meet application KPI, limit traffic
   according GP to AID
20:   else
21:     SN_AKPI(NA, AKPI)  $\triangleright$  If there are left over resources, send information to
   neighbors
22:   wait(IV)  $\triangleright$  Wait the time to a new verification
23: Back to Begin

```

3.3.4 Global Mobility Algorithm

The last control algorithm presented is shown in Algorithm 5 and addresses global mobility of vehicles. The function of this algorithm is to identify, based on vehicle beacons, if they are in a geographic region different from the previous one. This application

needs to be hosted on some server with direct communication with MANO.

In some situations, such as for example holidays where a lot of people go to touristic area, for example, vehicles leave their regions and goes to another region. If the infrastructure does not detect this movement, resources may not be sufficient to handle the application demands in the new region. Thus, based on the vehicles kinematic data, the global mobility control application updates vehicle regions and notifies MANO to analyze the deployment of resources and take necessary actions.

The algorithm inputs are vehicles kinematic data (VKD), vehicle identifier (VID), MANO Address (MA), and the database map (DM), that is a map with the relation of coordinates of each region. As output, there are DQ, DU, and MN (Mano Notification).

Algorithm 5 Mobility Central Controller Algorithm

Input: VKD, VID, DA, DM, MA, LV

Output: DQ, AGP, DU, MN, LCN

```

1: Begin
2: NVR, AVL, MF  $\leftarrow \square\square$   $\triangleright$  Define variables, New vehicle region, AVL and MF
3: for all VKD do
4:   CR(VKD, VID, DM)  $\rightarrow$  NVR  $\triangleright$  compute vehicle region based in update
   beacons
5:   if LV  $\neq$  NVR then  $\triangleright$  new VR is different
6:     DQ(DA, Vehicles, VID)  $\rightarrow$  AVL, VR  $\triangleright$  Query applications related to
     vehicles
7:     MF = 1
8:     DU(VID, Region, NVR, DA)  $\triangleright$  Update new VR in shared database
9:     for all App in AVL do
10:      MN(MA, MF, AID)  $\triangleright$  Notify MANO
11: End

```

The algorithm uses the periodic beacons received from vehicles to verifies, through CR(VKD, VID, DM) function, the region of each car based on its ID, the coordinates in VKD messages and the DM. If vehicle region calculated is different from that stored in the shared database, the controller updates the new information in the shared database and notifies MANO about changes in applications associated with affected vehicles.

Since the beacon messages arrive all the time, it is not correct to query the database for each VKD that the algorithm receives, because the region modifications usually aren't frequent for a specific vehicle and because it can overload database and network. So, the algorithm has the last version of the vehicle region stored locally (LV), and it just proceeds with the additional steps if the new region is different from the previous one.

Specific heuristics can be incorporated to improve this mobility algorithm, like vehicle

prediction used in some related works. Our contribution here is to show that is necessary to integrate this feature into the solution, to the infrastructure react in topology changes at regions level.

4. Performance evaluation

This chapter is divided into five parts. The first is a description of the implementation details of the components used in performance evaluation. Next, we explain the evaluation methodology used in this work, followed by the description of the scenario used with its parameters and after are presented the metrics with an explanation of how they were computed. At last, the results obtained are shown with their respective analyzes.

In order to evaluate the proposed framework, it was used an environment that allowed the emulation of a vehicular network with the vehicles moving according to the dynamics of congested traffic and associated with different vehicular applications. This way it was possible to analyze the influence on the chosen metrics as a function of the proposed framework performance to support the communication requirements of the different applications under the same mobile infrastructure. These results were then compared with the obtained with the use of two other possible alternative network control approaches in the same environment.

4.1 Implementation

The implementation is composed of five codes, using Python, Shell script and SQL script codifications, in the environment of Linux Ubuntu 16.04.1. The choice of use shell script was because the emulator software used (Mininet-wifi) uses the Linux operating system features, and some functions of emulated components can be easily accessed with shell commands. The use of Python was since it is the default language to implement Mininet-wifi script configurations, while SQL script was used to automatizes the creation of the shared database. Thus, the main codes that were implemented for this work and will be detailed in the next subsections are:

1. Python script for building the topology in Mininet-wifi, with mobility and general

emulation parameters;

2. Shell script to vehicles generate traffic related to vehicular applications;
3. Database SQL script that implements the shared database in Mysql server;
4. Shell script that implements the central controller application, interacting with Ryu SDN controller through its REST API to apply the general QoS network policies;
5. Shell script that acts on the RSUs, in the role of local controllers application, redirecting traffic and enforcing the policies defined by the central controller.

4.1.1 The Mininet-wifi emulator

To evaluate the proposed framework, the main components with its related algorithms were implemented and tested using an emulation strategy with the Mininet-wifi tool [35], which is a fork of the SDN Mininet emulator [36]. Several works have already been published using the Mininet-wifi for various software-defined wireless network implementations, including vehicular networks [5].

With Mininet-wifi it is possible to build wireless networks that support the functionality of the Openflow SDN protocol. To make this possible, the authors of the tool integrated the well-known Mininet emulation software with the wireless networking components available in the Linux kernel, which implement the functionality of IEEE 802.11 networks.

In Figure 4.1, extracted from Mininet-wifi manual¹, it is possible to identify the main components in an example of the communication between two hosts (stations) through an Access Point in a network created in Mininet-wifi.

The Access Points are devices that manage associated stations and are virtualized through hostapd daemon, using virtual wireless interfaces. Hostapd is an IEEE 802.11 AP and IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator that generally comes by default in Linux distributions. This implementation permits the configuration of some features like SSID, channel, mode, password and cryptography.

In the kernel-space, the module `mac80211_hwsim` is responsible for creating the virtual wireless interfaces used by stations and access points. The Media Access Control Sublayer Management Entity is realized in the stations side in the kernel-space, while in the user-space the hostapd is responsible for this task in the AP side.

¹<https://github.com/ramonfontes/manual-mininet-wifi/raw/master/mininet-wifi-draft-manual.pdf>

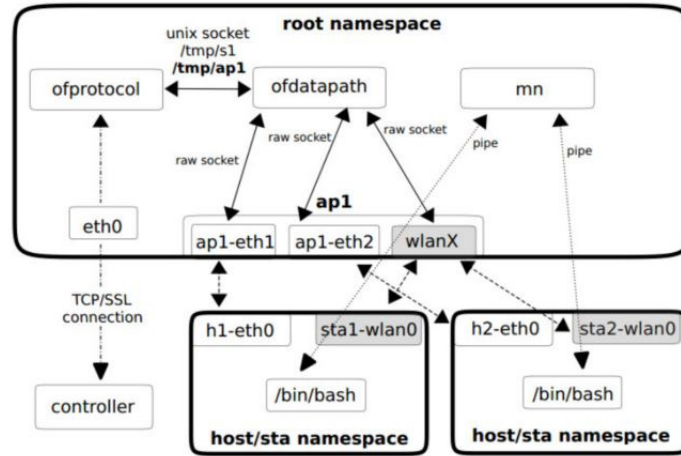


Figure 4.1: Components and connections in a network created with Mininet-WiFi with two hosts and a Access Point

Mininet-WiFi also uses Linux utilities such as `iw` and `iwconfig` for getting information and configure the wireless interfaces. Another utility is Traffic Control, that is an user-space utility program used to configure in the virtual interfaces some parameters such as data rate, delay, latency and loss. To simulate the wireless medium can be used the Traffic Control or Wmediumd tools.

The SDN switches are implemented by Mininet-wifi using the Open vSwitch² [37], that is an open source multilayer virtual switch widely used in academy and industry. In this work, it was used Open vSwitch version 2.5.4.

One of the latest innovations of the tool was the possibility of use hosts that behave as vehicles, allowing experiments with vehicular networks. In Figure 4.2, also extracted from the manual, shows the architecture of three node cars in an example with a RSU node.

The vehicles in the emulation environment have two wireless network interfaces being one to provide communication between cars and RSUs (Vehicle-to-Infrastructure) and the other to provide communication between vehicles (Vehicle-to-Vehicle). With an internal switch in each car, the communication flows passing by it can be manipulated by the SDN controllers. The RSUs in vehicular networks are like the Access Points in traditional wireless 802.11 networks, and in Mininet-wifi they have an internal switch that permits to modify the data communication flows that passes by it too.

A point that is relevant about the Mininet-wifi is that it does not implement either 5G nor 802.11p radio technologies. However, it was chosen because the focus of this work

²<https://www.openvswitch.org/>

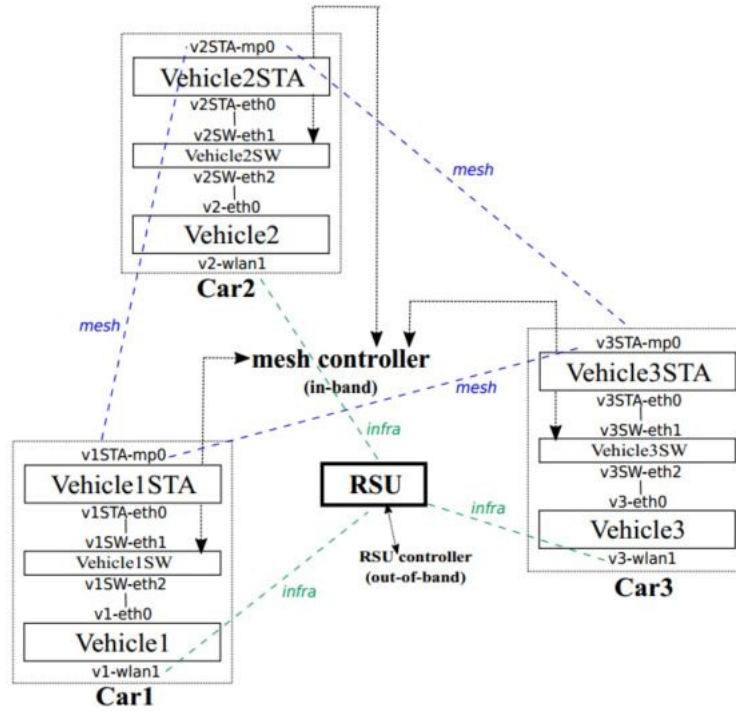


Figure 4.2: Architecture of three node cars in an example with a RSU node

was to orchestrate the resources in the network independent of the radio technologies and, in summary, the use of Mininet-wifi has advantages like the fact of allowing a realistic emulation of SDN wireless networks. Specifically related to vehicular networks it has some interesting characteristics such as the possibility of manually defining the mobility of vehicles, importing mobility traces or even integrating it with the mobility simulator SUMO³.

4.1.1.1 Implementation of Mininet-wifi emulation components

The parameters of time, vehicles mobility, wireless propagation model, and bandwidth of the links were defined in the Mininet-wifi Python code.

Since via a Python script is possible to execute some actions in Linux components, we put in this script some instructions to configure the first switches communication flows, to be possible that the traffic from vehicles arrives in application servers.

When vehicles move to outside from one RSU range and arrive in the range of other RSU, its communication flows need to be adjusted in the network switches. In telecommunications, this is known as handover. As the movement of vehicles is defined in this main Mininet-wifi script, the configuration flows necessary in backbone switch to keep

³<http://sumo.dlr.de/index.html>

cars communicating in the handover process are set in this script also.

Once it starts the emulation, this main script was configured to make the calls to execute the shellcodes of central and local controllers and the SQL script to create the shared database before vehicles start to communicate with application servers.

The application traffic generated in vehicles was made in the direction from cars to servers and, since a response was not expected in the evaluated scenario, the respective UDP ports in servers were not initiated in listen mode. This configuration results in ICMP packet errors when the servers receive packets to ports that are not in listen mode. To avoid the noise of these error messages in the network, we use this code to configure in the firewall of the emulated servers (IPTABLES) some rules to drop these specific ICMP messages.

Another configuration made in this code was related to the ARP table of each one of vehicles and servers. As the congestion increases, the entries in the ARP table of some nodes expire, making necessary to retransmit ARP messages. Besides generating noise in the results, these messages bring additional complexity to manipulate them. The Mininet-wifi itself has a feature named "staticArp" to deal with this, and when it is enabled the nodes start already with the ARP entries loaded without expire. This feature was unstable in the experiments conducted, and we opt to register the entries manually in each node, using external files with MAC entries that are loaded by Mininet-wifi script when the emulation starts.

Another action defined in this script is related to TCPDUMP⁴, that is a tool to collect packets in network interfaces. This tool was enabled to gather information about the input packets in application servers and save them in text files that are used to generate the results to analysis.

This code has options to initialize the emulation with different approaches to manipulate the network communication flows. If the script is executed with the option "-f", the emulation will start all components implemented to use the approach of the proposed framework, including central and local controllers and the shared database. With "-q" option, the code starts the emulation with QoS only approach, where the database is instantiated with the central controller but only the initial QoS related to applications flows are configured and there are no more control actions during execution. With "-b" option the emulation is executed using the Best Effort approach, where there is no prioritization in traffic of applications.

⁴<http://www.tcpdump.org/#documentation>

This code is publicly available at the Github repository of this work⁵.

4.1.2 Generation of traffic related to the applications

Each vehicle emulated by Mininet-wifi is a host in Linux environment that permits the execution of commands in its shells. The code that generates the traffic related to the vehicular applications is executed by each one of the vehicles. The logic was implemented in a way that the cars only initiate the transmission after detecting its connection with some RSU. When they detect a wireless disconnection (for example during a handover), stops the communication and restarts when connected again.

HPING3 tool was used to generate traffic from cars to the servers. This tool allows creating traffic patterns with a certain amount of flexibility. All flows are UDP, in the ports assigned to each application tested: 5002, 5003, 5004 and 5005, and with the data rate of 500 Kbps or 1 Mbps. Each car generates four communication flows that result in 2 Mbps per vehicles, as will be detailed in the evaluation scenarios.

In addition to the traffic generated via HPING3, each vehicle also sends every 1 second a "PING" message and save the result in text files to compute next the round time trip metric (RTT). The TCPDUMP is also enabled in each vehicle to collect the output data in network interfaces to be used to generate results.

This code is publicly available at the Github repository of this work⁶

4.1.3 Deploying the shared database

The database SQL script is responsible for creating the shared database in Mysql server⁷ version 5.7.23. The Mininet-wifi code calls this script and when executed it first erase the database named "framework" if it exists, to clean old data, and creates a new database with the tables showed in Figure 4.3.

The appkpi table has as the primary key an unique integer that represents the identifier code of each application. It also contains a field with the information of applications priority class. Other fields are the name of the vehicular application, the KPIs of end-to-end latency, reliability and data rate, the transport protocol (TCP or UDP), and the transport protocol port. These fields are used by control algorithms to create the flows to prioritize the applications and meet their requirements.

⁵https://github.com/saraivacode/framework_its_sdn/blob/master/testef_14.py

⁶https://github.com/saraivacode/framework_its_sdn/blob/master/carcon.sh

⁷<https://dev.mysql.com/doc/>

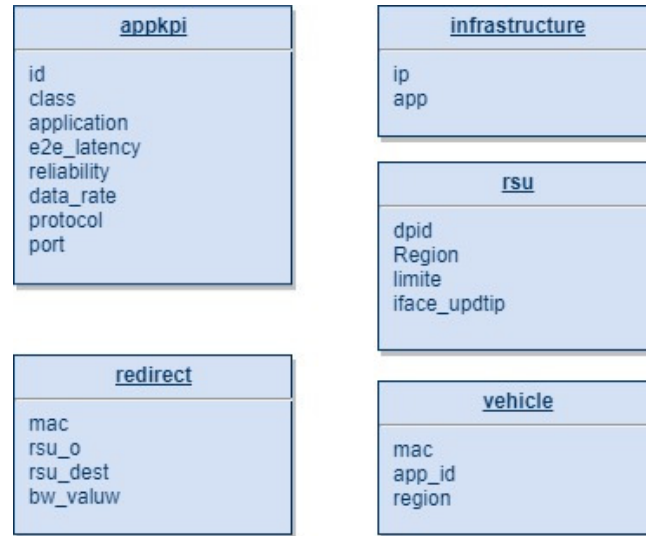


Figure 4.3: Tables created in shared database

The infrastructure table contains the IP address of the server associated with each application. This address is the primary key of this table, while the application identifier is the foreign key that comes from the appkpi table. Controllers use these information when assembling QoS rules and queues to prioritize traffic flows with the destination to these application servers.

The rsu table contains the DPID of the switches in each RSU, the geographic region where they belong, the data rate limit that the upload links of these RSUs supports, and the interface number of the interface that is linked with the backbone upload connection. The control algorithms use this information to apply the QoS rules in the correct interfaces and switches. Despite the limit of the upload link is in this table, the local controllers' script uses an entry defined manually, but it can be easily adjusted to query this information from the database.

The vehicle table contains the relation between the cars, applications and the geographic region where they are. So, based on the information in this table, the control scripts identify the applications associated with each vehicle.

The information about the applications, vehicles, infrastructure, and RSUs related to the scenario used in the evaluation are inserted in database tables by the SQL script. Although our framework defines that MANO initially enters this information during the deployment of new applications in the city, we use this approach to optimize the time of evaluation.

The redirect table is the only that starts the emulation empty. The information is inserted in this table by local controllers when they redirect, limit, or block some traffic,

as will be shown in Subsection 4.1.5. The field `rsu_o` stores the identification of the RSU that takes the action, `rsu_dest` contains the identifier of the RSU that receives the traffic if the action is a redirection, or receives "X", identifying that is a register about a traffic blocked or limited. The `bw_value` receives the data rate related to the action, and the `mac` field contains the MAC address of the vehicle affected by the action. These informations are also used by the local controllers to calculate the balance of the RSUs, as will be detailed in Subsection 4.1.5.

The code related to the deployment of the shared database and the registration of the associated information is also publicly in the Github repository of this work⁸

4.1.4 Central Controller Implementation

The role of central controller in the network was implemented using Ryu SDN controller⁹ Version 4.27. As mentioned by [38], this is a top-rated software that supports the latest versions of the OpenFlow protocol and has a very active community.

Ryu has a REST (Representational State Transfer) API that accepts commands in the JSON (JavaScript Object Notation) format, allowing integration with applications that use this technology. Thus, the script that performs the role of a central controller algorithm can be executed on any machine that can remotely send messages to the central controller via its API.

In practice, the central controller script runs on the same Ryu controller machine, and REST instructions were sent to the loopback address on the TCP port where Ryu listens by default (8080). Once the instructions are received, the controller configures the switches through the OpenFlow 1.3 protocol.

To start Ryu to allow the application of QoS in the network and the interaction via REST API, the standard applications `ryu.app.rest_qos`, `ryu.app.qos_simple_switch_13`, `ryu.app.rest_conf_switch` and `ryu.app.ofctl_rest` must be started before.

This script identifies the switches in RSUs by its DPIDs in `rsu` shared database table and configures in its upload links the queues to prioritize the traffic of applications based on the KPIs information in `appkpi` table. The queues are created defining the minimum data rate necessary to meet the data rate KPI of each application. Despite there are other KPIs in the table, the evaluated scenario considers only data rate, as will be shown in the next section.

⁸https://github.com/saraivacode/framework_its_sdn/blob/master/initialdb.sql

⁹<https://osrg.github.io/ryu/>

After defining and applying the queues, the central control script queries the shared database to obtain the IP address and port related to applications. Based on this information, the QoS rules are created to associate the traffic with respective queues on the RSUs. Besides the main application flows, the QoS rules also configure ICMP messages destined to the servers to be prioritized in the same way that the application on the respective server. This configuration was necessary to the messages of PING application be handled as application traffic and the metric derived from this packets (RTT) be realistic.

Code related to central controller deployment is publicly available at Github repository of this project¹⁰.

4.1.5 Implementation of local controllers

The local controllers' code is responsible for calculating the balance available in each RSU and decides what are the necessary actions to do in order to keep attending applications KPIs when some RSU is congested. It is valid to note that the implementation considers three RSUs in experiments and this script executes the actions of local control for them all. The Figure 4.4 shows the flowchart with a summary of the operations performed by local controllers via this script.

Periodically the RSU identifies the vehicles in its RAN and query the shared database to determine the applications associated with these vehicles. This information is locally stored in files `b_list` and `c_list`, with the MAC address of the vehicles related with applications with priority classes B and C. The MAC of cars can be on both lists if it is a subscriber of the two applications.

After the identification of vehicles in RAN and its applications, the balance of resources in the RSU is calculated to verify if the RSU is congested. Local controller considers an RSU congested when there is no resources enough locally to deal with the sum of the applications demand in all vehicles connected in its RAN.

To calculate the balance, first is necessary to compute the demand associated with each vehicle in RAN, according to Equation 4.1, where $D(V_i)$ is the demand related to the car i , resulting from the sum of the needs for all the n applications to which this vehicle is subscriber ($D(Appx)$). As mentioned before, the demands of applications are stored in `appkpi` table of the shared database.

¹⁰https://github.com/saraivacode/framework_its_sdn/blob/master/central_controller2.sh

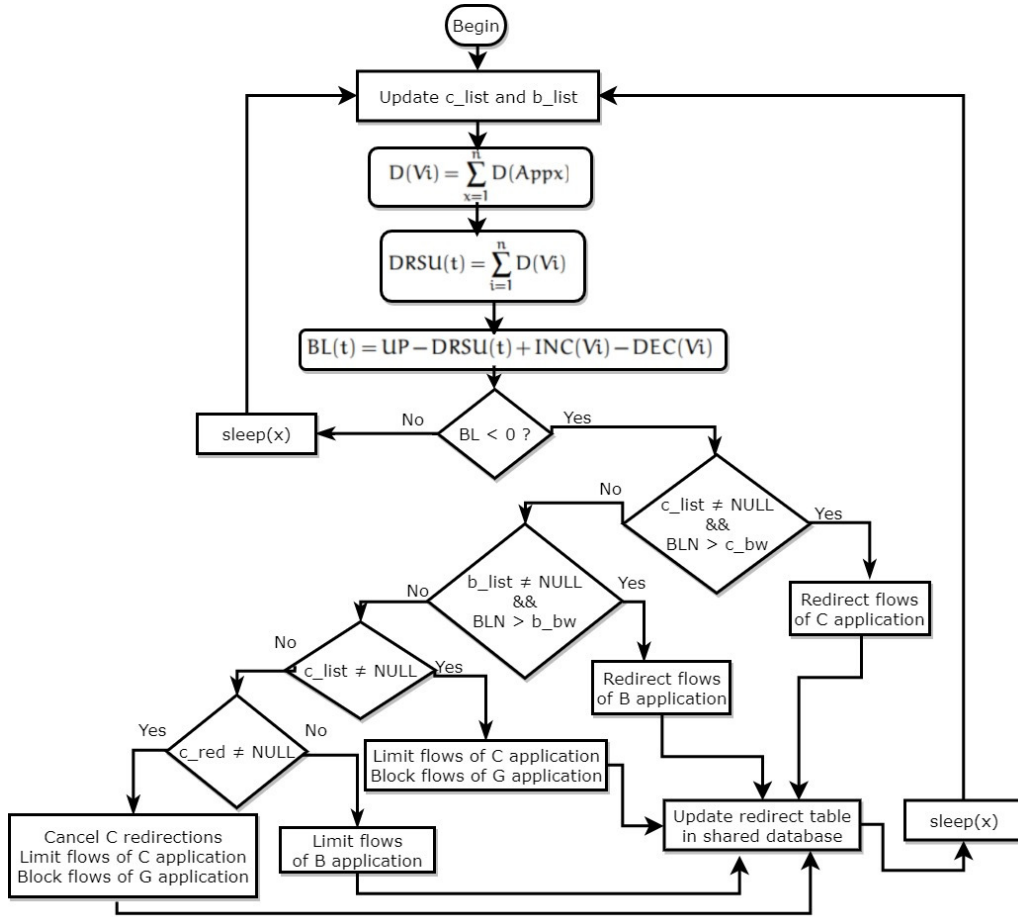


Figure 4.4: A summary of the actions executed by local control in RSUs

$$D(V_i) = \sum_{x=1}^n D(Appx) \quad (4.1)$$

Based on the demand calculated for each vehicle V_i , the local controller can compute the total requirements in the RSU at a given moment t , adding the demand required by all n vehicles in the RSU, as stated in Equation 4.2.

$$DRSU(t) = \sum_{i=1}^n D(V_i) \quad (4.2)$$

Given the value of $DRSU(t)$, the local control queries the redirection table to check for redirected traffic by the RSU in question or to it and limited or blocked in its RAN. If the RSU redirect or limit some traffic in the past, the value of related data rate needs to be added, and if it receives some redirection, the value must be subtracted from the final balance. Thus, the calculation of the RSU balance at a moment t is given by the Equation 4.3, where INC and DEC are respectively the values to add and decrease of the balance

due to the registers in the redirect table and UP is the value referring to the capacity of the upload backbone link.

$$BL(t) = UP - DRSU(t) + INC(V_i) - DEC(V_i) \quad (4.3)$$

Applications class B have more priority and if the RSU is congested ($BL < 0$), the local controller initiates the actions with the prioritized application with lower priority, that in our implementation is the application class C. So, local controller verifies if its local list of vehicles with C application is not empty ($c_list \neq NULL$) and some neighbor has a balance to deal with related data rate ($BLN > c_bw$). If both verifications are true, the flows of the vehicle analyzed will be redirected to the neighbor and related information updated in the shared database. It is valid to note that periodically RSUs updates the redirect table and flows in switches to eliminate the registers related to vehicles that are no more connected to RAN, to avoid distortions in balance results. When updating b_list and c_list files, controllers erase entries related to applications that already have a register in redirect table, since it is not possible block or limit a flow that already is in this situation.

Once the same script calc the balance of all RSUs, the information of balance to use when verifying if some neighbor has a balance to deal with some application is always available. If the executions were in separated environments and with each RSU controller executing an instance of the code, a mechanism to exchange messages, for example using REST/JSON, would need to be implemented.

If the RSU is congested and is not possible to redirect C flows, the controller tries to redirect B application flows. If this is not possible it limit the C applications flows locally, which consists of install in local switch RSU, flows to associate the traffic of affected vehicles to a default queue of non-priority traffic. The non-priority queue already contains traffic of others non-priority applications and the local controller block it to not compete with priority traffic being limited. This default non-priority traffic is named in our implementation as G (General) traffic.

In conjunction with wireless simulation drivers, Mininet-wifi uses the HOSTAPD to create the RSUs. HOSTAPD is a user space daemon for access point and authentication servers that implements IEEE 802.11 access point management and other features like IEEE 802.1X/WPA/WPA2/EAP authenticators, RADIUS client, EAP server, and RADIUS authentication server.

In summary, the RSUs in the environment are access points instantiated via HOSTAPD

and a problem faced in this work implementation, that was causing inconsistency in the RSUs calculation balance, was that even though a vehicle had left the coverage area of an RSU, it appeared as still associated with the respective access point on HOSTADP, resulting in incorrect demand calculations. We notice that this behavior was because Access Point uses a timeout to consider a disassociated node. This timeout is by default 300 seconds.

To correct the inconsistencies, the parameter that instantiates the access points was changed in the Mininet-wifi source code, changing the timeout values. After the adjusts, we have obtained consistent results, but it is yet necessary that local controller waits 20 seconds to confirm if the RSU is really congested, which solves the question of precision, but with an impact on the performance of the local controller, since it takes more time to act and prioritize the applications in the traffic jam. Nevertheless, as will be seen in the section 4.5, the results were superior when compared to others approaches.

Code related to local control deployment is also publicly available at Github¹¹

4.1.6 Auxiliary codes

Besides the mentioned codes, were implemented some others auxiliary to support the evaluation scenarios and results analysis. One of these codes is responsible for inserting and updating information in the shared database, as would be made by MANO. Since the shared database already has registered the necessary data to evaluated scenarios, this code was not used.

Other code used deals with the files generated by TCPDUMP and PING to create the files used as input in R software to compute the data for performance analysis. The R scripts that calculate metrics also are auxiliary codes. All these files are in the Github repository of this work¹².

4.2 Evaluation methodology

As the proposal of the present work deals with a solution to be applied in a vehicular network environment, it is necessary that the evaluation scenario represents a typical vehicular traffic situation. Therefore it was chosen an urban congestion scenario, where the vehicles move slowly due to the traffic being congested, resulting in an increase in the

¹¹https://github.com/saraivacode/framework_its_sdn/blob/master/local_controllers.sh

¹²https://github.com/saraivacode/framework_its_sdn/

number of vehicles for m^2 over time.

Since the ultimate goal of the infrastructure implemented with the use of the proposed framework is to meet the requirements of vehicular applications, different applications have been defined in evaluation scenarios and, due to their intrinsic characteristics, they have different communication requirements. Vehicles in the evaluation environment generate applications data communication flows that consume the limited mobile communication infrastructure resources.

Three rounds of emulation were performed, and each one uses a different network resource control approach. Besides to the approach implemented with the proposed framework, one used applies QoS in the network, in order to prioritize the traffic of the applications according to their demands of bandwidth. In the third approach, called Best Effort, the traffic of the applications is treated equally with any other traffic, regardless of its requirements.

With the number of vehicles increasing in the area of each one of the three RSUs of evaluation scenario, the consumption of resources of the infrastructure by the applications communication flows will increase also, and may result in packet loss, increase in latency or an rate limit lower than the necessary by the applications KPI of data rate. Thus, with the items described, we intend to evaluate the influence, as a function of the control approach used, in the Round-Trip Time, in the bandwidth occupied, and in the packet loss of the different communication flows of the applications, with the vehicles moving according to the dynamics of the configured congested traffic.

4.3 Evaluation scenario

To represent a common situation that occurs in the traffic of most cities, we configure an evaluation scenario with 15 vehicles (car0 to car14) representing a traffic jam, where the density of cars per m^2 tends to accumulate with the time.

During the 300s of each emulation of the scenario with traffic jam, the cars move assuming four different position configurations and stay there for at least 75 seconds. So, the emulation time is divided into four equal periods of 75 seconds, with the congestion in RSUs upload links gradually increasing.

It is possible to assert that in each one of the 75 seconds period (T1 to T4) there is a different level of congestion in the overall network (C1 to C4). Therefore, when we refer in this chapter to the interval times T1, T2, T3 and T4, it implies also about the congestion

levels C1, C2, C3, and C4, like shown in Figure 4.5.

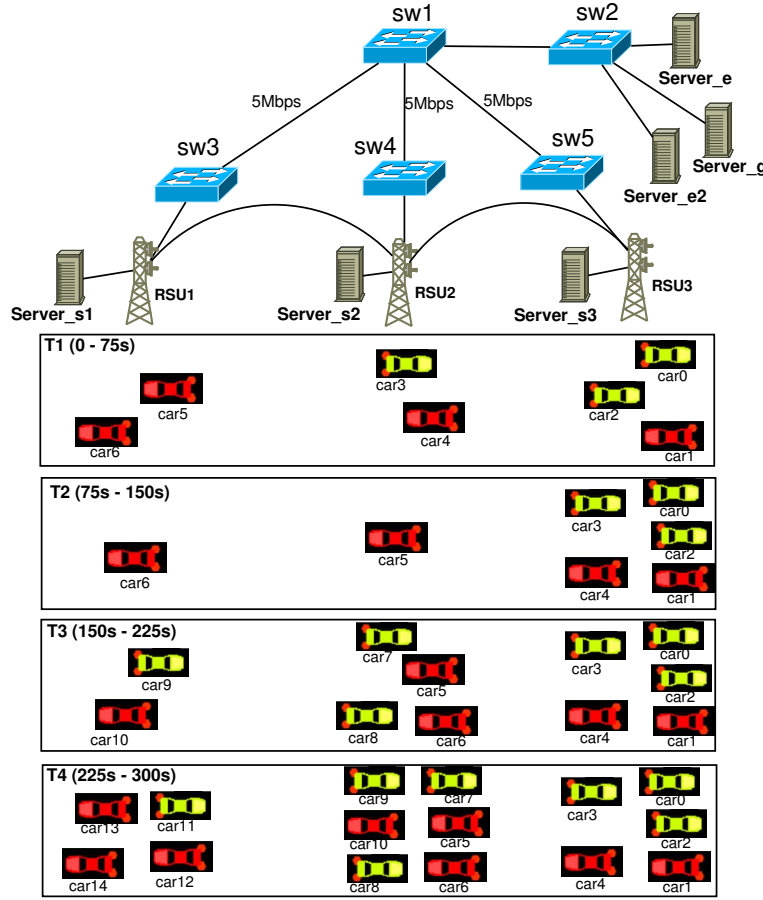


Figure 4.5: Evaluation scenario representing a traffic jam with different congestion levels over the time

The three RSUs were distributed in an area so that the distance between them is equal to 500m and the coverage radius corresponds to 250m. The value of 250m was chosen based on [39], which used RSUs ranging from 50m to 250m in their experiments. The Table 4.1 contains the emulation parameters configured.

In evaluation scenario, all the 15 vehicles are subscribers of four hypothetical vehicular applications that were configured as detailed in the Table 4.2. The application named S represents a vehicular safety application and is the most critical to delays and losses and this way its related servers (S1, S2, and S3) are connected directly to the RAN of each RSU, following a MEC approach. This application has a data rate KPI of 500 Kbps and uses the UDP 5002 transport port.

The applications E and E2 are respectively of efficiency, such for example fuel economy or reduce time travel, and of entertainment, such as online video streams. These applications aren't delay-sensitive, and its traffic goes to its remote servers through the backbone link of the RSUs. Since the backbone link of RSUs was configured with a limit

Table 4.1: Configurations used to performance evaluation

Parameter	Value
Number of vehicles	15
Number of RSUs	3
RSUs range	250m
Backbone switches	5
Application servers	6
Propagation Model	Log Distance
RAN	IEEE 802.11g
Data rate per vehicle	2 Mbps
Number of applications	4
RSUs Upload link BW	5 Mbps
Emulation time	300 seconds
Number of emulations	1 for each approach

Table 4.2: Applications characteristics

Applications	Use	Data rate	Protocol	Port	Priority
S	Safety	500 Kbps	UDP	5002	A
E	Efficiency	500 Kbps	UDP	5003	B
E2	Entertainment	1 Mbps	UDP	5004	C
G	Generic	500 Kbps	UDP	5005	–

of 5 Mbps, in some situation during evaluation time the links are congested and the applications must be prioritized properly. Application E has a data rate KPI of 500 Kbps to UDP port 5003 while E2 consume higher network bandwidth and needs a 1Mbps to UDP 5004. Application G represents a generic traffic of 500 Kbps to UDP port 5005 that has no priority.

As each vehicle in the environment is associated with all applications, each one generates a total of traffic corresponding to the sum of the bandwidth used by each of the four applications, resulting in a total of 2Mbps traffic per vehicle.

The five backbone switches (sw1 to sw5) have the role of interconnecting the RSUs with the cloud servers of the E, E2 and G applications. It is worth noting that in each RSU there is also an internal switch responsible for interconnecting the RSUs with their neighbors, with the respective MEC servers, and with the backbone switches.

Although Mininet-wifi implements emulation strategy to use SDN components, there is also simulation to perform other elements of the wireless environment. As described in [40] and [41], the parameters of a simulation in vehicular networks must be appropriately defined to avoid non-realist results.

It was enabled in the emulation environment the use of `wmediumd`, which is, as quoted by [42], an advanced solution to emulate the characteristics of wireless communication, such as backoff algorithm and isolation between the wireless interfaces. In conjunction with `wmediumd`, the Log Distance propagation model was enabled. According to [43], the propagation model of Log Distance is widely accepted for simulations of wireless communications.

Since Mininet-wifi does not support the IEEE 802.11p or 5G vehicular communication standards and the focus of this work was not on radio technologies, and also the proposed solution is independent of the radio technologies used, it was employed in communication between the vehicles and the RSUs the IEEE 802.11g standard.

Related to the congestion levels during evaluation time, Table 4.3 contains the number of vehicles in each RSU for each congestion level, as well as the calculated bandwidth balance. T is the interval time, CL is the congestion level, NV is the number of vehicles in RSU, $E + E2$ is the sum of priority data rate in Mbps, G is the data rate of non priority G application and BL the balance of RSU.

Table 4.3: RAN status in evaluated times

Time (CL)	RSU1				RSU2				RSU3			
	NC	E+E2	G	BL	NC	E+E2	G	BL	NC	E+E2	G	BL
0 - 75s (C1)	2	3	1	1	2	3	1	1	3	4,5	1,5	-1
75 - 150s (C2)	1	1,5	0,5	3	1	1,5	0,5	3	5	7,5	2,5	-5
150 - 225s (C3)	2	3	1	1	4	6	2	-3	5	7,5	2,5	-5
225 - 300s (C4)	4	6	2	-3	6	9	3	-7	5	7,5	2,5	-5

T is the interval time, CL is the congestion level, NC is the number of vehicles in RSU, $E+E2$ is the sum of priority data in Mbps, G is the data rate of non priority G application in RAN and BL the balance of RSU

It can be observed that at the level of congestion $C1$, that is, between 0 and 75 seconds of the experiment, there is a negative balance in the $RSU3$, with three vehicles. As the upload link is 5Mbps and the combined demand of the three cars equals 6Mbps, there is a 1Mbps deficit. At $C2$ level, the congestion increases at $RSU3$, as two more cars arrive. During the third period, in $C3$ the bottleneck already involves the $RSU 2$ and 3 . Finally, the most intense congestion level is reached $C4$ (between 225 and 300 seconds) with all $RSUs$ congested.

To evaluate the proposed solution were conducted three emulations with the explained scenario. In each emulation, a different approach was used to deal with the traffic of applications. One of them was the proposed framework, where local and central controllers act to prioritize the applications to meet its KPI of data rate. The second approach uses only

QoS configuration in the network. This approach is named as QoS only and it configures the QoS rules and queues as central SDN controller do but don't use the functionalities of local controllers like traffic redirection. The last approach was named Best effort, and it consists of no one configuration in the network to prioritize traffic.

4.4 Evaluation metrics

Were computed the metrics of round trip time (RTT), packet delivery ratio (PDR) and throughput the over time, to evaluate the impact on applications communication in function of the three approaches used.

We chose RTT metric based in some works like [44], which states that it can be defined as the time taken by a VANET application starting from the initiator node (source vehicle) sending a message until receiving a response from the core network.

According to [45], that also uses RTT to evaluate its proposal of a software-defined vehicular network, knowing the value of RTT is important because RTT it is proportional to how much load in a path, what is proportional of congestion. With RTT we can also verify the impact of the approaches in S application data, that uses MEC servers, even with its data not going to backbone it is a valid information, as will be discussed in next section.

The values of RTT calculated by the "PING" ICMP messages in log files were summed each second for all vehicles and divided by the number of cars transmitting. Equation 4.4 shows the calc, where $RTT(s)$ is the value of Average RTT calculated every second, $RttV_x$ the n RTT values received in the second s and NC the number of vehicles that transmitted in the congestion period to which the second s is part.

$$RTT(s) = \frac{\sum_{x=1}^n RttV_x}{NC} \quad (4.4)$$

The PDR metric is widely used to evaluate the performance of vehicular networks. [46], for example, use PDR as one of the metrics to evaluate its proposal that aims to compute feasible routes between the vehicles in a network with multiple QoS constraints. The authors define PDR as a metric that represents the average ratio of the number of successfully received data packets at the destination to the number of data packets sent. With PDR it is possible to evaluate the level of data loss in function of each approach used.

The PDR was calculated through the ratio between the sum of packages received by servers and the number of packages sent by vehicles in each congestion level, as showed in Equation 4.5, where PS_r is the sum of servers received packets and PS_t is the sum of cars sent packets.

$$PDR = \frac{PS_r}{PS_t} \quad (4.5)$$

Several works use throughput as an evaluation metric in vehicular networks, as [47] that use it to evaluate its proposal in an SDN Enabled 5G-VANET. Since in the evaluated scenario the KPI of prioritized applications was the data rate, we collected the throughput over the time to verify if the application servers received the data rate expected according to vehicles transmission.

To compute the Throughput, since the data with the size of the packets sent by vehicles and received by servers collected was in bytes, they were summed in each second and multiplied by 8 to obtain the result in bits per second. The values obtained were divided by the number of vehicles that were effectively transmitting in each period, as shown in Equations 4.6 and 4.7. $TTC(s)$ and $TRS(s)$ are respectively the throughput of packets sent by vehicles and received by cars in the second s , while PS_i and PR_i respectively the n packages sent by the NC vehicles and received by the servers in the second s . NC equals to 7 for the period between 0 and 150 seconds (congestion levels 1 and 2), 11 between 150 and 225 seconds (C3) and 15 for the last period of 225 to 300s (C4).

$$TTC(s) = \frac{(\sum_{i=0}^n PS_i) * 8}{NC} \quad (4.6)$$

$$TRS(s) = \frac{(\sum_{i=0}^n PR_i) * 8}{NC} \quad (4.7)$$

4.5 Results

4.5.1 RSUs balance as function of framework actions

Before analyzing the evaluation metrics, this section shows the results in the balance of RSUs in function of the actions of the proposed Framework. Local controllers were expected to prioritize applications data flows per its data rate KPIs and priorities. These results are in Tables 4.4, 4.5 and 4.6, where T is the interval time, NC is the number of

vehicles in RSU, BLB and BLA are respectively the balance before and after framework actions, RDO and RDD are respectively the totals of traffic in Mbps redirected from and to the RSU, and LM is the total traffic limited in Mbps, while BK is the traffic blocked. The codes in parenthesis relate to the type of traffic, e.g., 1E2+1E means that the value results of actions in one flow of the E2 application in one vehicle and one flow of the E application in one vehicle (could be the same vehicle or other).

Table 4.4: Results in balance of RSU1 with Framework solution

T (CL)	RSU 1						
	NC	BLB	BLA	RDO	RDD	LM	BK
T1 (0 - 75s)	2	1	1	0	0	0	0
T2 (75 - 150s)	1	3	3	0	0	0	0
T3 (150 - 225s)	2	-1	0	0	2 (2E2)	1 (2G)	0
T4 (225 - 300s)	4	-3	0.5	0	1,5 (1E2+1E)	3 (3E2)	2 (4G)

T is the interval time, **CL** is the congestion level, **NC** is the number of vehicles in RSU, **BLB** is the RSU balance before actions of the Framework, **BLA** is the RSU balance after actions of the Framework, **RDO** is the total of traffic in Mbps redirected from RSU, **RDD** is the total of traffic in Mbps redirected to the RSU, **LM** is the total traffic limited in Mbps, **BK** is the traffic Blocked in RSU. The codes in parenthesis relates to the type of traffic, e.g., 1E2+1E means that the value results of actions in one flow of the E2 application in one vehicle and one flow of the E application in one vehicle.

During the first 75 seconds the balance of RSUs 1 and 2 was positive and there were no actions. In the RSU3 there was a negative balance of 1Mbps, so the traffic G in three vehicles was limited resulting in a positive balance of 500 Kbps that is shared between the G applications limited flows.

Between 75s and 150s the same seven vehicles persist in the area of coverage of the RSUs but with a different distribution, which increases the congestion in the RSU3, reducing their balance. In RSUs 1 and 2, the balance is higher, since they are with only one vehicle each. In RSU3, the balance before the local control action is negative at 5 Mbps. Thus, G traffic is limited, releasing 2.5 Mbps, and the flows of E2 application of three vehicles are redirected to the RSU2, releasing more 3Mbps and resulting in 500Kbps of positive balance in RSU3, that is shared by the G application of the 5 vehicles, and RSU2 that receive the flows stays with a balance of zero.

When in the congestion level C3 (150s to 225s) there are 11 vehicles in the coverage area of RSUs. The RSU2 that was zeroed reach a negative balance of -6 Mbps, with the arrival of more three cars, totaling 4. In this case, the RSU2 redirects 2.5Mbps, being

Table 4.5: Results in balance of RSU2 with Framework solution

	RSU 2						
T (CL)	NC	BLB	BLA	RDO	RDD	LM	BK
T1 (0 - 75s)	2	1	1	0	0	0	0
T2 (75 - 150s)	1	1	0	0	3 (3E2)	0	0
T3 (150 - 225s)	4	-6	0.5	2,5 (2E2+1E)	3 (3E2)	2 (2E2)	2 (4G)
T4 (225 - 300s)	6	-3.5	0	2 (1E2+2E)	3 (3E2)	5 (5E2)	3 (6G)

T is the interval time, **CL** is the congestion level, **NC** is the number of vehicles in RSU, **BLB** is the RSU balance before actions of the Framework, **BLA** is the RSU balance after actions of the Framework, **RDO** is the total of traffic in Mbps redirected from RSU, **RDD** is the total of traffic in Mbps redirected to the RSU, **LM** is the total traffic limited in Mbps, **BK** is the traffic Blocked in RSU. The codes in parenthesis relates to the type of traffic, e.g., 3E2 means that the value results of actions in three flows of the application E2 in three vehicles.

2Mbps to the RSU1, referring to the flows of applications E2 of two cars and 500 Kbps to RSU 3, related to B application of one vehicle. Thus, RSU1 that had a 1 Mbps balance reaches zero after receiving 2 Mbps from RSU2 and limiting traffic of G application locally automatically to the two vehicles in its RAN. The RSU3 stays with zero balance after receiving the redirection of B application from RSU2. It is worth noting, however, that the traffic for G applications of the five vehicles in the RSU3 at that moment should be close to zero since they are limited and there is no balance. The RSU2, also limits the traffic flows of E2 application in two vehicles to use the 500Kbps free and block the G traffic for the four cars in its RAN.

In C4 (225s to 300s) there are 15 vehicles in the coverage area of RSUs and the most intense level of congestion in the performance evaluation. In that period the situation in the RSU3 no changes, but RSUs 1 and 2 receive new cars, causing congestion in both. The RSU2, which had a balance of 500Kbps shared by the limited applications, stays with a negative balance of 3.5 Mbps with the arrival of two new vehicles. To stabilize the situation in this RSU the controller acted to limit 2Mbps related to the E2application in two cars, blocked the traffic G also of the new vehicles, obtaining a further 1Mbps. Local controller also cancels the redirection of an E2 flow that was directed to the RSU1, and limit this flow locally. After this it redirects the traffic of a stream of the E application to the RSU1, obtaining more 500 Kbps and resulting in a balance of 0. It also blocks G application in RAN and this way the E2 flows from five of the six cars results limited to close to zero since there is no balance to share.

Table 4.6: Results in balance of RSU3 with Framework solution

T (CL)	RSU 3						
	NC	BLB	BLA	RDO	RDD	LM	BL
T1 (0 - 75s)	3	-1	0.5	0	0	1,5 (3G)	0
T2 (75 - 150s)	5	-5	0.5	3 (3E2)	0	2.5 (5G)	0
T3 (150 - 225s)	5	-5	0	3 (3E2)	0.5 (1E)	2.5 (5G)	0
T4 (1225 - 300s)	5	-5	0	3 (3E2)	0.5 (1E)	2.5 (5G)	0

T is the interval time, **CL** is the congestion level, **NC** is the number of vehicles in RSU, **BLB** is the RSU balance before actions of the Framework, **BLA** is the RSU balance after actions of the Framework, **RDO** is the total of traffic in Mbps redirected from RSU, **RDD** is the total of traffic in Mbps redirected to the RSU, **LM** is the total traffic limited in Mbps, **BK** is the traffic Blocked in RSU. The codes in parenthesis relates to the type of traffic, e.g., 5G means that the value results of actions in five flows G application in five vehicles

Still in C4, in the RSU1, that was with zero stays with negative -3M balance, after the arrival of two other vehicles. Local controller acted limiting the traffic of application E2 in two cars and blocking G flows in RAN. Since the redirection of traffic from an E2 application from RSU2 was canceled and sent an E flow with 500 Kbps, the balance finalizes with positive 500 Kbps to share between the E2 streams of three vehicles.

In the next sub-items, the evaluation metrics as a result of these actions were analyzed and compared with QoS only and Best effort approaches.

4.5.2 Throughput and RTT over time

In Figures 4.6, 4.7, and 4.8 are the results of throughput and RTT, related to E application, over the time, using the Framework, QoS only and Best effort approaches, respectively. As mentioned before, this application is the most critical among those that are prioritized in backbone links.

Figure 4.6 shows that with the use of the proposed framework the data transmitted by the vehicles during the first 75s arrive practically with the same rate on the server. In the transition period around 75s, there is a natural decrease in the rates in function of the handover process of some vehicles.

Between 75s and 150s it is observed that initially there is a discrepancy in the received data rate in the server, that normalizes around 110s. This behavior is because RSU3 was

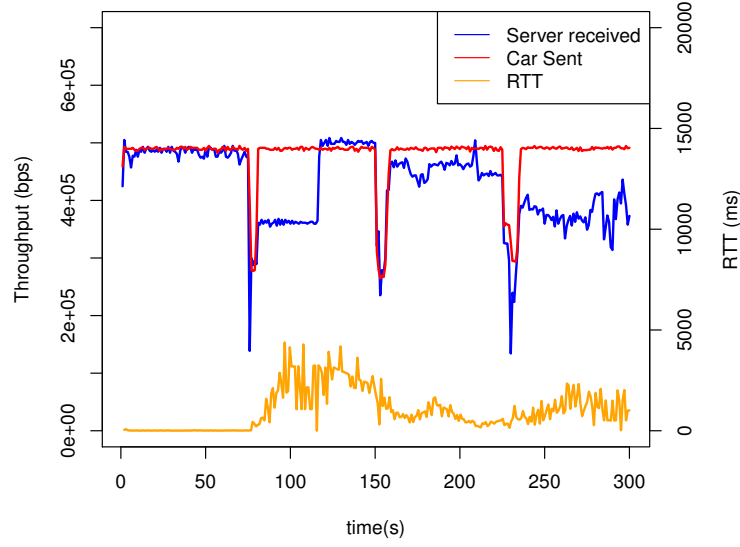


Figure 4.6: Throughput and RTT over the time in **application E** using framework approach

wholly congested and the controller acted redirecting 3 Mbps to RSU2, as previously explained, however, because the controller has to wait at least 20 seconds to initiate the actions to confirm if the RSU is congested, there is a delay in normalization. In this period there is also a significant increase in delay, which reflect in a longer RTT that reaches around 5000 ms, since congestion while the local controller does not act causes the communication to become slower. In function of this longer RTT, the received data rate in server seems subtly above the data rate sent by vehicles, since they are added delayed.

During 150s and 225s there is a small difference concerning what is transmitted and what is received. Local controller redirected the E application traffic of one vehicle from RSU2 to RSU3, which is operating at the limit.

In the final evaluated period, between 225s and 300s, there is a more significant decrease in the received server data rate with an increase in the RTT, because in this period the RSU2 is redirecting the traffic related to the application E in two vehicles to the RSUs 1 and 3, which are both already operating at the limit. Also, since all RSUs have all 15 vehicles transmitting, there is a higher probability of collisions, resulting in packet loss and consequently a decrease in data rate reception.

It is important to note that, although local controller always waits for at least 20s to act, it is worth noting that the difference in traffic received before and after the local controller performance is much more discrepant during 150s and 225s because the result of the action was more significant, since was necessary the redirect of 3Mbps.

In terms of the results, with QoS only approach in the E application traffic (Figure 4.7), it is verified that during T1, where congestion was only in RSU3 and with traffic of type G surplus limited in non priority queue, the result was almost equal to that achieved with the use of the proposed Framework. However, for the other levels of congestion, the losses were significant, with RTT values higher, therefore worse.

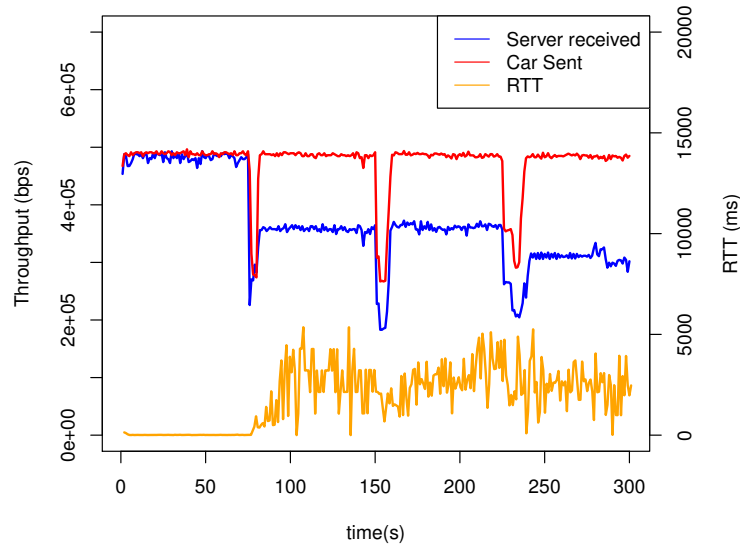


Figure 4.7: Throughput and RTT over the time in application E using QoS only approach

With the application of QoS only, during T2 and T3 the available traffic was divided to meet the priority E and E2 applications in each RSU, since there was no available balance in RANs and this strategy does not use redirection, In T4, as expected, the situation is aggravated, due to the greater congestion of the network.

In the Best effort approach, where there is no traffic prioritization, the results are the worst for the E application in all congestion situations. It is worth noting that the RTT values are much smaller with the Best effort approach, which indicates that the use of QoS and redirects brings a penalty concerning this metric. Otherwise, the KPI used as the reference to guide controllers actions in evaluated scenario was the data rate.

In summary, with the proposed framework were obtained the best results regarding data rate over the time to E application, proving that this approach was the best in meeting the data rate KPI of this application. Even with the impact observed in results in function of the QoS use, our approach presented best results of RTT than QoS only approach, showing that the actions necessary to meet data rate KPI doesn't impacted negatively the RTT metric.

About the E2 application, which was the second to be prioritized, being less priority

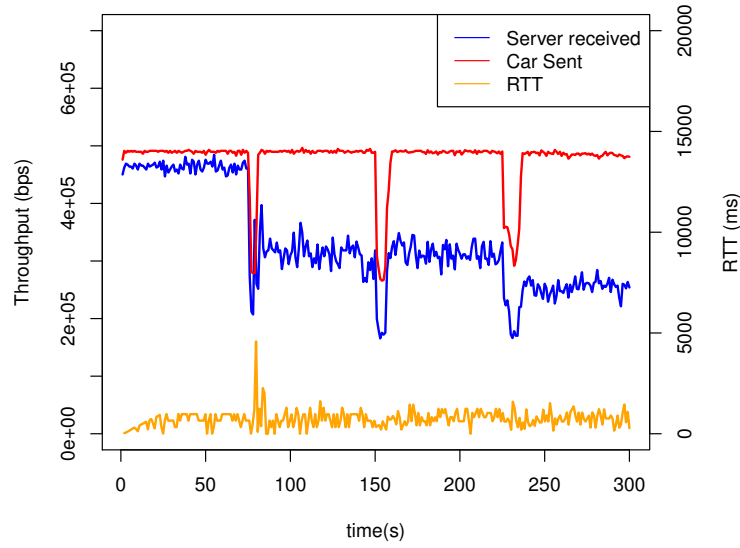


Figure 4.8: Throughput and RTT over the time in application **E** using Best effort approach

than the **E** application but with more priority than **G**. The results are shown in Figures 4.9, 4.10, and 4.11 respectively for the approaches using the proposed Framework, QoS and Best effort.

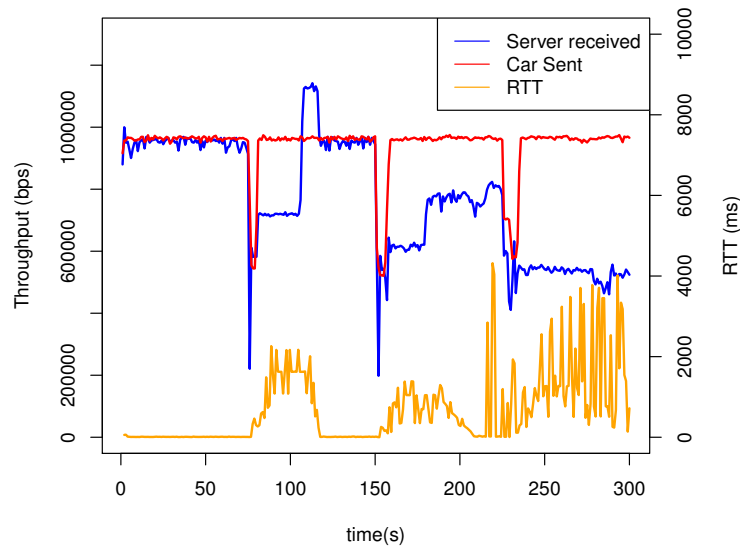


Figure 4.9: Throughput and RTT over the time in application **E2** using Framework approach

With the use of the proposed framework, during **T1** the traffic is practically not impacted. After this, there is a natural loss due to the vehicles movement and the handover, and in **T2** there is a period of loss that local controller acts to normalize. The simultaneous increase in RTT results in a distortion during some seconds with the received server data rate being higher than data rate sent by vehicles. This behavior, as explained to **E**

application, occurs as a function of the time that the local controller needs to wait before confirm a congestion. The RSU3 redirects E2 application flows of three vehicles to RSU2 in T2.

Between 150s and 225s the difference in rates is more significant, reducing with controller actions, although it persists relatively high. The RSU2 redirects the E2 traffic of two vehicles to RSU1, the redirections of the traffic of 3 vehicles from RSU3 to the RSU2 continues, and communication of two vehicles on the RSU2 is limited to use the available bandwidth of 500 Kbps for the required 2 Mbps.

In the final seconds of the emulation with the proposed framework, it is observed that the E2 application is severely impacted, due to congestion and also because it was limited when necessary to prioritize the E application that has higher priority. During T4 a significant increase of RTT is observed, reaching around 4 seconds. At this moment, besides the redirects, the traffic of the E2 application of 3 vehicles in the RSU1 and four vehicles in the RSU2 are limited to almost zero.

With the approach using only QoS (Figure 4.10), it is verified that except during the initial seconds, there is a significant difference between the transmission rates of the vehicles and reception of the server. The result is proportional to what happened with the use of this same strategy with the traffic of the E application (Figure 4.7). The rate values are higher because the E2 application transmits at 1 Mbps, but the similar proportionally can be explained by the fact that in the congestion the available bandwidth is divided among the priority applications. RTT levels similar to those occurring with the Framework approach are observed, except for T4.

By analyzing the results for the Best effort approach (Figure 4.11), it can be seen that, just as occurred with E application, RTT values are better and data rate at reception worst.

In summary, with the proposed framework the results obtained to E2 application were similar to that using QoS only approach, with worst results in T4, with was expected, since the solution prioritized E application. RTT values were proportional with the congestion level and naturally worst when the traffic was limited.

Concerning the G application, the Figures 4.12, 4.13, and 4.14 show the results collected along the emulation time with the data obtained when using respectively the approaches with the proposed Framework, QoS only and Best effort. It is possible to see that with both the proposed framework and QoS only this application is severely impacted, since the congestion increases over time and the data is not prioritized. It is also possible to observe that the RTT values are high, reaching around 10 seconds, because of

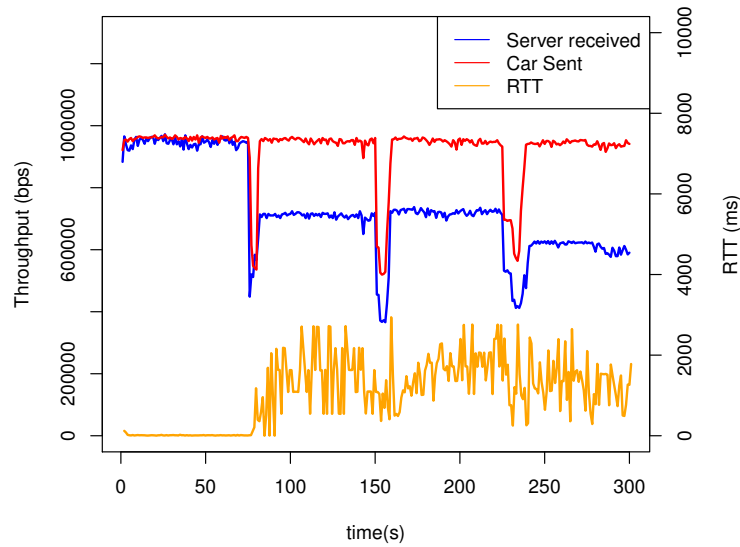


Figure 4.10: Throughput and RTT over the time in application **E2** using QoS only approach

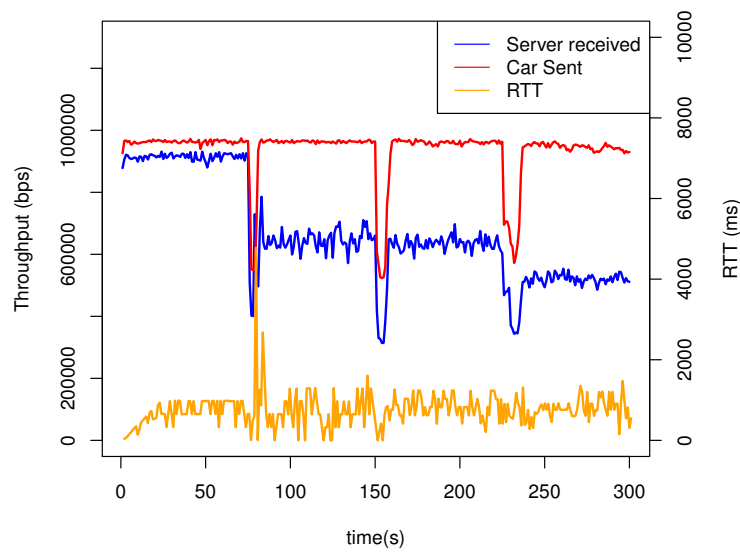


Figure 4.11: Throughput and RTT over the time in application **E2** using Best effort approach

congestion.

With the Best effort approach, the G application has a very similar impact to that achieved with the use of this same approach in the E and E2 applications, because this approach treats all traffic equally and consequently at least not limit G traffic.

In summary, G application has the best results with Best effort approach, and this was expected since both Framework and QoS only approaches limit the communication of this

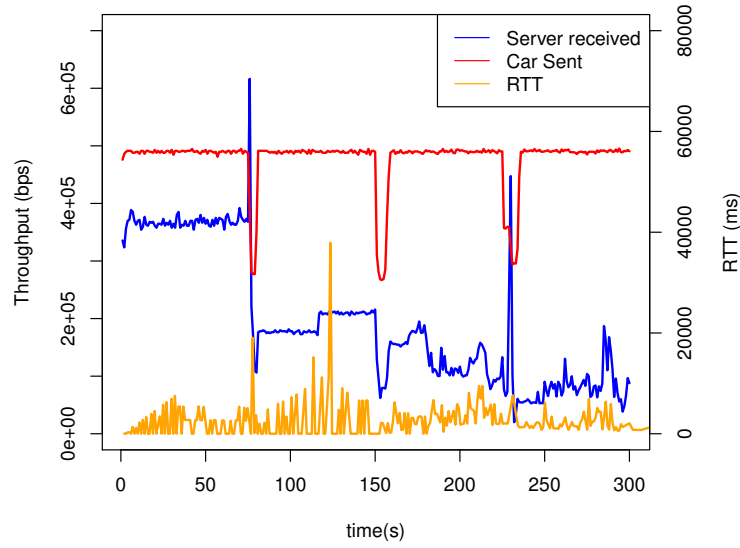


Figure 4.12: Throughput and RTT over the time in application **G** using Framework approach

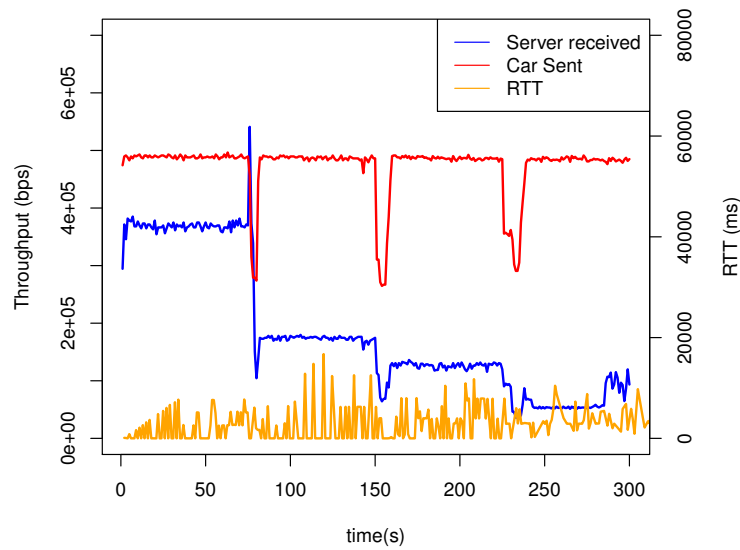


Figure 4.13: Throughput and RTT over the time in application **G** using QoS approach

application to prioritize others.

In the Figures 4.15, 4.16, and 4.17 are shown the RTT results and transmission and reception rates collected over time on the **S** application MEC servers. Since the data of these applications do not pass through the network backbone, the expected result is independent of the approach used.

The results are very similar, which is consistent with the theory, except for some outliers in the transition around 75s in Best effort approach. It is also observed that, although

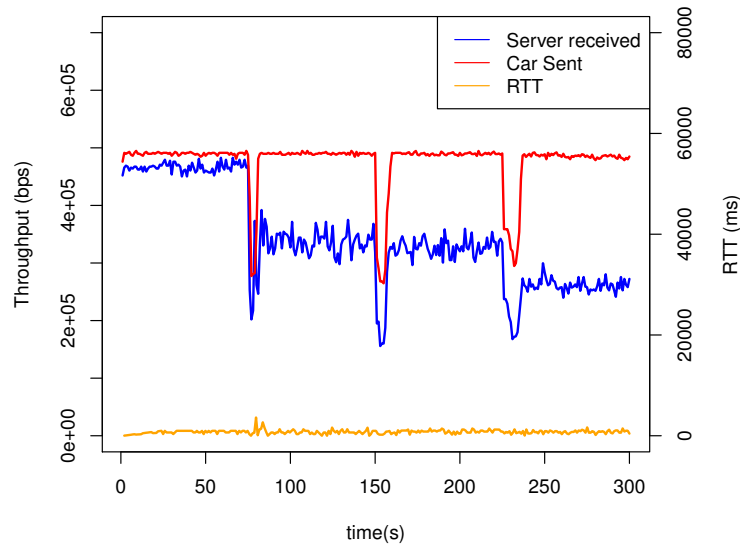


Figure 4.14: Throughput and RTT over the time in application **G** using Best effort approach

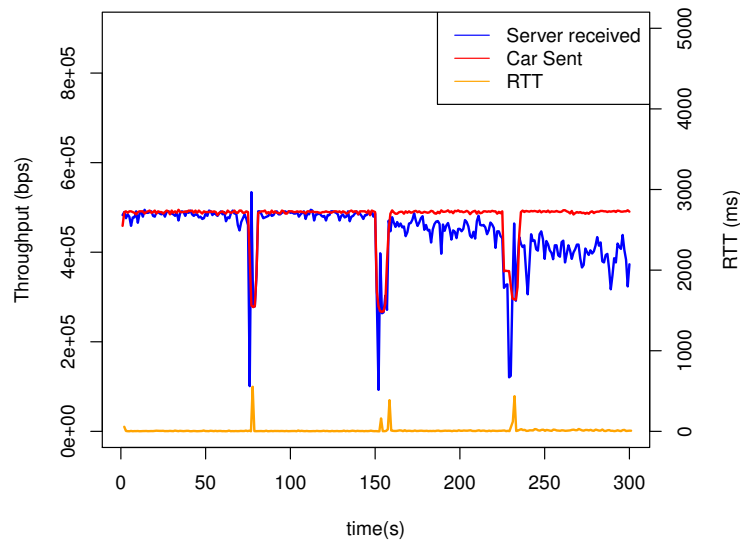


Figure 4.15: Throughput and RTT over the time in application **S** using Framework approach

congestion in the backbone does not impact the traffic of these applications, it is not immune to congestion at the RAN level and, therefore, it is possible to observe a decrease in reception rate in T4 in all graphs of S application. The RTT values were shallow, as expected in function of the use of MEC servers. A relevant result is that although the decrease in data rate during T4 in function of RAN congestion, there is not a significant increase in RTT, as with other applications that doesn't use MEC and face congestion in backbone.

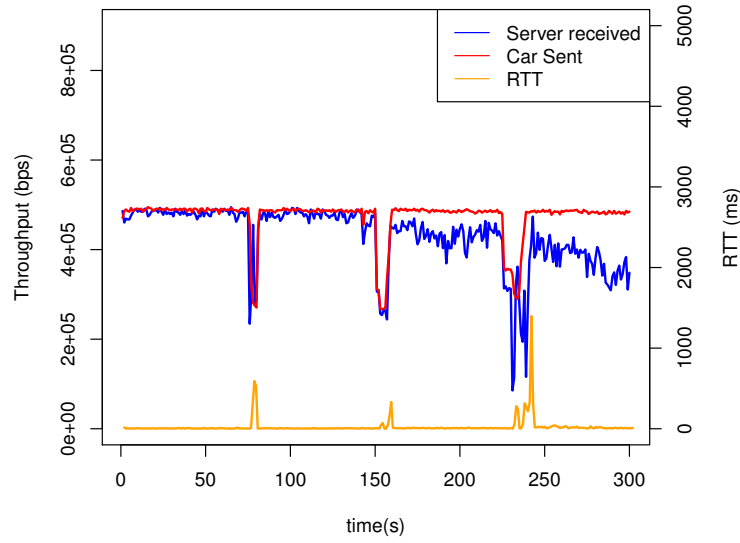


Figure 4.16: Throughput and RTT over the time in application **S** using QoS approach

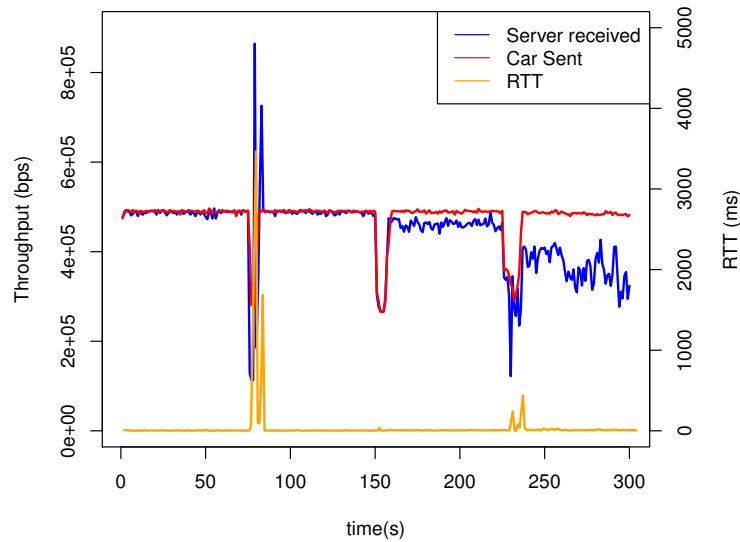


Figure 4.17: Throughput and RTT over the time in application **S** using Best effort approach

4.5.3 Packet Delivery Ratio

In this section, are analyzed the PDR values obtained according to the use of the Framework, QoS only, and Best effort approaches. The Figure 4.18 shows the result of the mean PDR for the E application calculated in each one of the four periods of congestion. In the Table 4.7 the results of the PDR during the emulation for E application using each approach are also summarized.

As already mentioned, during T1, RSUs 1 and 2 were with a positive balance, while in

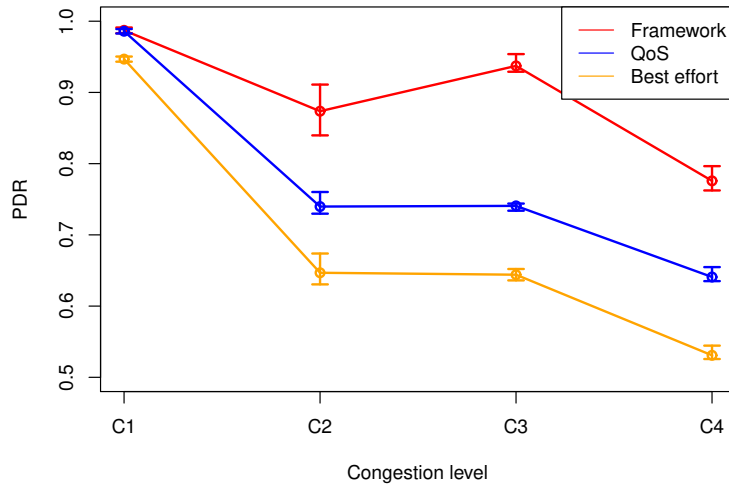


Figure 4.18: Application E - PDR

Table 4.7: App E Packet Delivery Ratio

Approach	1st Qu.	Median	Mean	3rd Qu.
Framework	0.7734	0.9361	0.8954	0.9904
QoS	0.7087	0.7388	0.7783	0.9614
Best effort	0.5760	0.6444	0.6940	0.9102

RSU3 was with a little congestion solved with the QoS rules associating G application to default non-priority queue. So, in this time frame, there is no adverse effect on PDR when using the Framework or QoS only approaches. With the Best effort approach, there is a little decrease since the non-priority traffic G compete with the priority ones. In the others time intervals, with the proposed solution the values of PDR are significantly higher, in function of the action of the local controller to prioritize E application.

It is valid to note that in T3 the value of PDR increases in the relation of T2, despite the higher congestion. This happening is a reflect of the losses in T2 during the time that local controller wait to confirm the congestion.

Considering all the 300 seconds of emulation, as shown in Table 4.7, the mean PDR with the proposed solution was 89.54%, with QoS only 77.83%, and with Best effort 69.40%. These results confirm that with the proposed solution the priority E application had fewer loss packets in the network. It is possible to realize, based in the quartiles in the table, that the results to Framework approach were less dispersed and in 50% of the time, the PDR was superior to 93.61% with our solution, while in QoS only and Best effort these values where respectively 73.88% and 64.44%.

In summary, even with the problems caused in the efficiency of proposed solution implementation due the time necessary to confirm congestion, all results of PDR were superior for the prioritized E application.

The Figure 4.19 shows the results of the PDR for the E2 application in congestion periods and in Table 4.8 the summarization considering all 300 seconds of emulation.

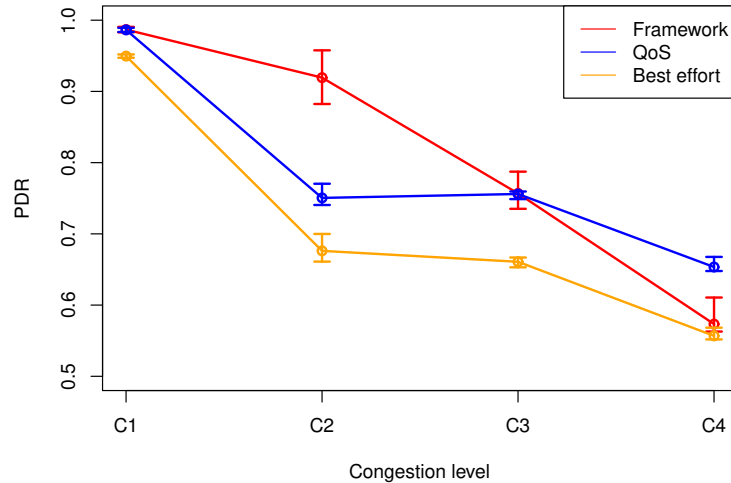


Figure 4.19: Application E2 - PDR

Table 4.8: App E2 Packet Delivery Ratio

Approach	1st Qu.	Median	Mean	3rd Qu.
Framework	0.6341	0.8117	0.8129	0.9894
QoS	0.7246	0.7516	0.7881	0.9595
Best effort	0.6004	0.6680	0.7123	0.9248

The PDR of E2 application shows a PDR with over than 90% for all approaches in T1 since the congestion was little and just Best effort cannot deal with it, but the effect was minimal. In T2 there is more congestion, and the traffic of the application is penalized in both QoS and Best effort approaches. Our solution presents a superior result because RSU3 redirects to RSU2 the flows of E2 application in three vehicles, normalizing the congestion. As the others approaches can't redirect traffic they present the worst result.

In T3 the worst result again with Best effort and our solution presents almost the same result as QoS only, even with the E2 application flows of two vehicles being limited to not to impact in the E application flows. In T4 the results are nearby, with an advantage to QoS only approach, because framework limits the E2 flows of 8 vehicles in RSUs 1 and 2 to not impact in E application.

Considering all the emulation time, even with the E2 being penalized when necessary to not impact in E application, the results were superior with the proposed solution, with a mean of 81.29% against 78.81% and 71.23% of respectively QoS only and Best effort approaches.

The Figure 4.20 shows the results of PDR related to G application using all approaches and in Table 4.9 the results were summarized considering all emulation time.

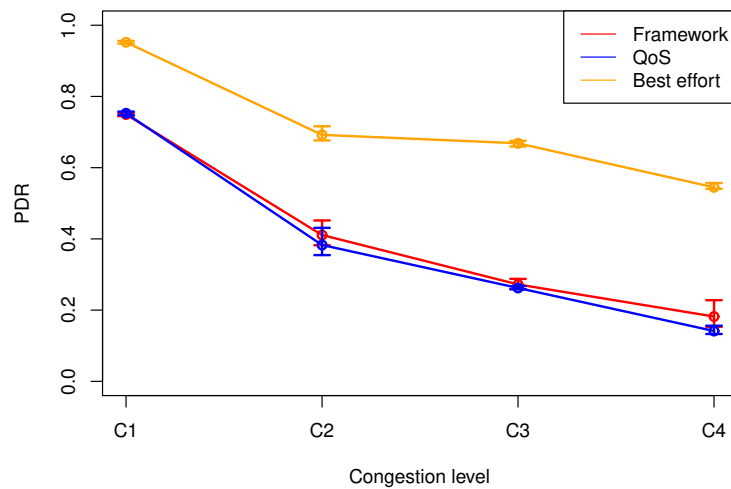


Figure 4.20: Application G - PDR

Table 4.9: App G Packet Delivery Ratio

Approach	1st Qu.	Median	Mean	3rd Qu.
Framework	0.20670	0.35890	0.40730	0.72190
QoS	0.2428	0.3369	0.3879	0.7247
Best effort	0.6069	0.6775	0.7157	0.9164

As have been seen with throughput, the results of PDR to G application were superior with Best effort approach, once this application was not a priority and the approaches of QoS only and Best effort penalizes it to prioritize the other applications, while Best effort deals with all applications in the same way.

Considering all emulation time, the PDR with the proposed solution and QoS only stays around 40%, while with the Best effort the result was superior to 70%. It is valid to note that for all applications the Best effort approach results in the PDR is around 70%, with is in coherence with the theory since it deals with all applications with no difference.

The Figure 4.21 shows the PDR result for the approaches used in S application and the Table 4.10 summarizes the results considering all 300s emulation time.

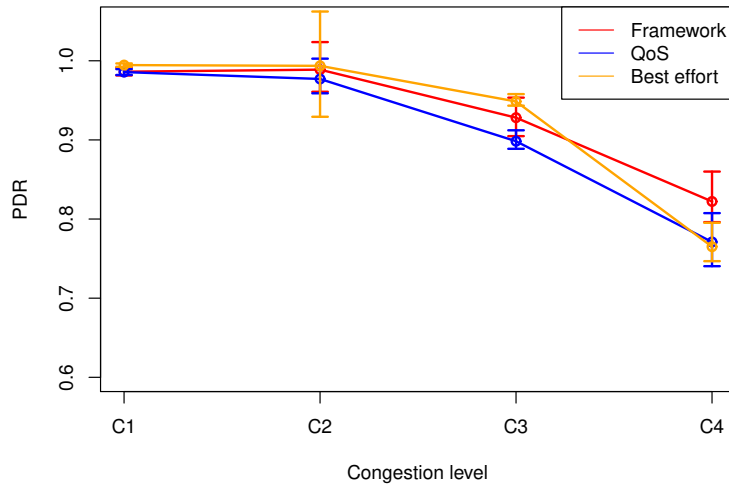


Figure 4.21: Application S - PDR

Table 4.10: App S Packet Delivery Ratio

Approach	1st Qu.	Median	Mean	3rd Qu.
Framework	0.8915	0.9676	0.9338	0.9930
QoS	0.8613	0.9520	0.9099	0.9860
Best effort	0.9029	0.9789	0.9274	0.9965

The results show the PDRs near for all approaches, which was expected since this application uses MEC servers and this way the traffic does not go to the backbone. These results present two points that require additional explanation. The first is related to T2, where the values of PDR were higher than 100% considering the 95% confidence interval, which is contra-intuitive since it is not possible that the number of packets received is higher than those sent by vehicles. This behavior occurs because in best effort, during the first handover the high delay causes an anomaly resulting in outliers in received data rate being more prominent than the sent data rate. The other point is related to T4, where can be observed that even with the traffic does not facing congestion in RAN, there is a decrease in PDR in function of the collisions and hence the probability, because the use of the Medium Access control protocol with collision avoidance in 802.11g.

It is possible to verify that, considering the values of all emulation time, the Mean for all approaches in S application is higher than 90%, which shows the benefits of use MEC solution related to this metric. It is also important to observe that the use of proposed solution doesn't interfere negatively in these results.

4.5.4 Round Trip Time

In this section, we analyze the impact on the RTT in the applications as a function of the approaches used. The Figure 4.22 shows the results through an empirical CDF for E application, and the summarized values collected during emulation time are in Table 4.11.

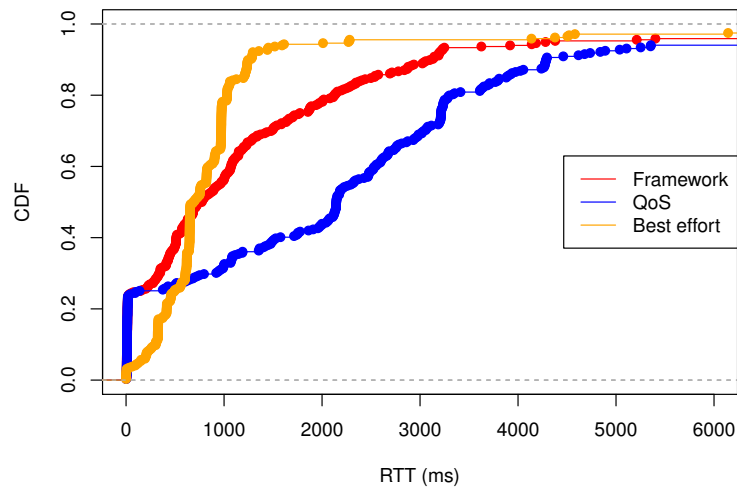


Figure 4.22: ECDF of Application E - RTT

Table 4.11: App E Round trip time

Approach	1st Qu.	Median	Mean	3rd Qu.
Framework	28.540	684.000	1005.000	1526.000
QoS	24.240	2128.000	1883.000	3065.000
Best effort	458.400	653.400	735.600	971.400

The ECDF of E application shows that more than 90% of the values are less than 1500 ms with the Best effort approach, while with our Framework 90% of the values are less than around 3000 ms and with the QoS only approach this value is around 5000 ms.

Is important to note that considering the RTT mean, the value with proposed solution is 1000 ms, and this doesn't impact significant in the application, since its KPI is related to data rate and 50% of the values was less than 684 ms, what is less than the mean of 735.60 ms in Best effort. The mean with proposed solution is approximately 1005 ms against 1883 ms with QoS only approach.

In summary, with Best effort approach are the best results, and the use of QoS causes an impact in the RTT of the network. Otherwise, the proposed solution meet the data rate KPI of E application using QoS and traffic redirect, reaching better values of RTT than QoS only approach.

The Figure 4.23 shows the ECDF results for RTT obtained with E2 application data with the approaches evaluated and in Table 4.12 the results for all emulation time are summarized.

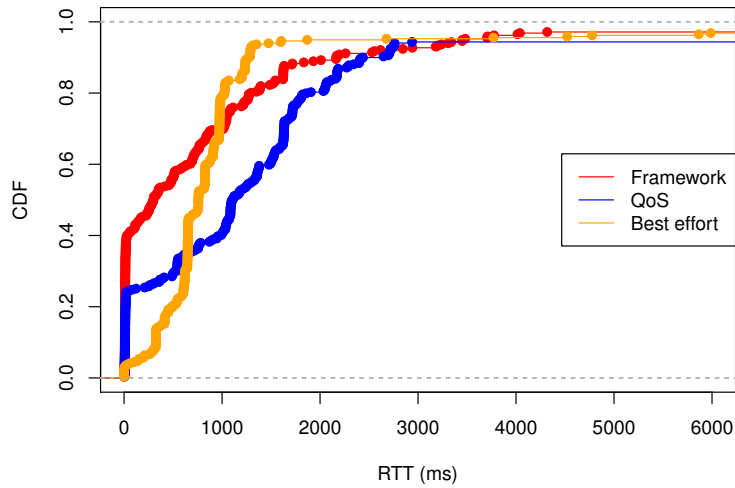


Figure 4.23: ECDF of Application E2 - RTT

Table 4.12: App E2 Round trip time

Approach	1st Qu.	Median	Mean	3rd Qu.
Framework	10.520	294.700	680.500	1045.000
QoS	22.0300	1086.000	1058.000	1635.000
Best effort	605.100	756.600	759.800	973.200

It is possible to observe in E2 application ECDFs that the probability of RTT to be less than 500 ms is more significant with the proposed solution, that is around 60%, against less than 40% with others. For all approaches, 90% of the values were less than 3000 ms. Considering the mean, proposed solution approach present the best results, even with the high RTT observed during T4, as showed in last section, and the much more dispersed RTT of the proposed solution. The dispersed RTT can be calculated using the interquartile distance, that is the difference between 1st and 3rd quartiles ($1045.00 - 10.52 = 1034.48$). The greater the distance, the more scattered the data will be.

In summary, with the proposed solution the RTT mean was the best to E2 application, despite the dispersion of the values in function of the congestion periods. This way, it is possible to affirm that the actions used to prioritize the applications does not have an adverse effect in RTT of E2 application communication.

The Figure 4.24 shows the ECDFs of RTT results to G application considering all

approaches and in Table 4.13 the results are summarized considering all 300s emulation time.

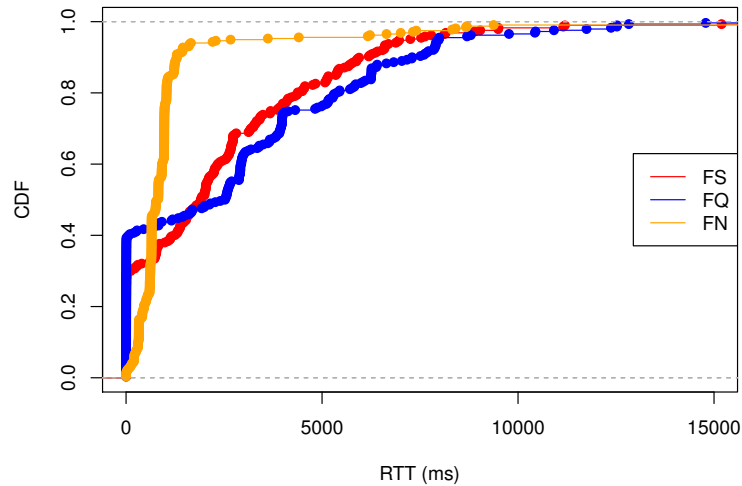


Figure 4.24: ECDF of Application G - RTT

Table 4.13: App G Round trip time

Approach	1st Qu.	Median	Mean	3rd Qu.
Framework	5.48	1909.00	2552.00	3771.00
QoS	2.831	2528.000	2792.000	4269.000
Best effort	596.200	751.900	760.600	974.300

It can be observed that RTT results with Best effort approach are significantly better, and this is the expected since G application was limited in QoS and Best effort approaches, resulting in high RTT values.

Considering the first quartile showed in summarized data, the Framework and QoS approaches present better results than Best effort, but analyzing all data, its evident that the Best effort presents better values of RTT, with a mean of 760.60ms against 2552.00 in Framework and 2792.00 in Best effort.

In summary, with the proposed solution the value of RTT is subtly better than with using only QoS and very worst than using Best effort. However, it is not a problem since this application was no priority in the network and it was necessary to manipulate it to prioritize the others.

The Figure 4.25 shows the ECDFs with RTT results obtained in S application and the Table 4.14 summarizes them. It is valid to note that once in this application was used a MEC strategy, the approaches used in the network did not act in the data of this application

and hence it is expected no significant difference in results between approaches and low RTT values in function of MEC strategy.

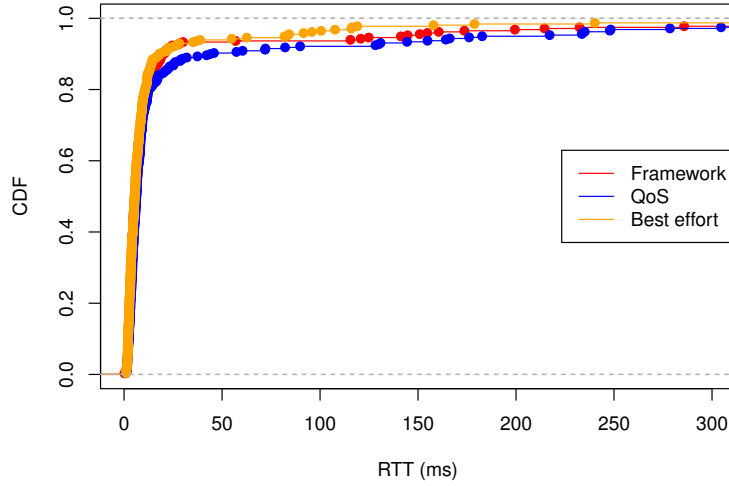


Figure 4.25: ECDF of Application S - RTT

Table 4.14: App S Round trip time

Approach	1st Qu.	Median	Mean	3rd Qu.
Framework	3.1720	5.1970	12.3600	8.8090
QoS	3.6690	6.0550	23.4900	9.2750
Best effort	3.126	5.233	28.710	8.569

As expected, ECDFs for S application shows close values and very lower than with other applications. The mean was kept lower than 30ms for all approaches. As mentioned earlier the approach chose doesn't significantly interfere in this values and the differences are little an mainly in function of the random simulation components.

In summary, with the proposed solution the results for S application presents a mean better than with the other approaches, but this is not in function of the approach chosen since the data of this application does not face congestion in the backbone and hence is not prioritized. Independent of any factor, it is possible to affirm that using the proposed solution to prioritize the other application does not cause an adverse effect in this application using MEC cloud.

4.5.5 Discussion

The results showed that with the use of the proposed Framework it is possible to orchestrate the network resources to respond to the dynamics of the vehicular environment

to meet data rate KPI and priority of the applications. It was possible to obtain better results with the proposed solution than with alternative evaluated approaches, concerning the data rate reception in application servers and the PDR, in different traffic congestion levels over time.

It was observed that the use of QoS mechanisms could increase the RTT, but using the necessary QoS to meet the requirements of the applications, the proposed Framework presented better results than with the QoS only approach. Concerning for example application E, that is the more priority that faces congestion in the backbone, the result with the proposed framework was 53,37% of the RTT obtained with QoS only approach.

The proposed solution presented for application E a result of PDR 15% above the QoS only approach and 29% above the Best Effort approach. It is worth to note that these advantages could have been greater once were impacted mainly in function of the limitation of the local controllers' implementation, where it was needed to wait at least 20 seconds to act, which is a significant value considering each congestion period with 75 seconds of the overall 300 seconds emulation time.

There is no possible to assert that the proposed solution will present better results in all possible scenarios. Scenarios with more vehicles moving following other mobility patterns in a broader geographic region may be evaluated to analyze the impact in results. Another point is that the traffic generated by applications in the vehicles was always in the direction from vehicles to application servers and changing it to bidirectional may represent a need to adjust control algorithms implemented and hence is necessary to evaluate the results in this situation. At last, employing mechanisms to prioritize resources in radio access networks to operate in conjunction with the functionalities of proposed frameworks may be evaluated to improve the results even more.

5. Conclusion and future works

This work addressed the problem of how to deploy a mobile infrastructure with a vehicular network that can dynamically meet the communication requirements of different vehicular applications in the complex environment of these networks. Based on the analysis of related works, it is possible to assert that this is the first work that presents a detailed approach, considering in conjunction the requirements of different vehicular applications, the specific characteristics of vehicular networks, like the vehicles mobility between different RSUs coverage, and validation using realistic SDN components.

The proposed solution consists of a framework using software-defined vehicular networks with algorithms that use the information about applications requirements, the mobility of vehicles, and the calculated balance of resources in the network to act in order to meet applications requirements.

The central and local SDN controllers, with a shared database, in a scenario with vehicles, RSUs, SDN switches, and servers related to four applications with different data rate requirements and priorities were implemented, with the use of a realistic emulation strategy, to evaluate the efficiency of the proposed solution. The mobility of vehicles followed an urban congestion mobility pattern. One of the applications used MEC servers, allowing to analyze the impact on their communication and to compare it with the others, even that it did not face congestion in the network links with the backbone.

The data rate received by application servers over time, the RTT, and the PDR of communication flows were used as performance metrics. The results were compared between the proposed solution and two alternative approaches, named QoS only and Best Effort. In the QoS only approach, the network was programmed only with QoS to meet the requirements of the applications, while in the Best Effort the data of the applications were treated independently of each other, i.e., without prioritization. The results showed that in the evaluated scenario the proposed solution presented better results than the

alternative approaches.

As future work there are opportunities like the evaluation of the solution in scenarios with more vehicles, moving following other mobility patterns and with the applications generating bidirectional traffic. Also, another opportunity is one opportunity is to evaluate how the proposed solution could meet other applications requirements besides data rate, like for example latency and reliability (related to packet loss).

The MANO component in the architecture of the proposed framework was not implemented in the conducted experiments. Thus its implementation with the functionality to orchestrate dynamically computational resources in the MEC and Remote clouds, integrating its actions with the network configurations made by the controllers in the vehicular network is an opportunity of future work too. Also, another opportunity of research is the study of the best heuristics and technologies, like for example Machine Learning, to manipulate the information related to the vehicular environment and to provide inputs to the control algorithms, improving the efficiency of the overall solution.

Also, another possible study that will add value to this work is the study of the integration of the proposed framework with radio access technologies, like for example LTE-V, in order that the controllers could manipulate radio resources to improve the efficiency of the solution in meeting the requirements of prioritized applications.

Bibliography

- [1] YAQOOB, I. et al. "Overcoming the Key Challenges to Establishing Vehicular Communication: Is SDN the Answer?", *IEEE Communications Magazine*, v. 55, n. 7, p. 128–134, Jul. 2017.
- [2] CHEN, M. et al. "Cognitive Internet of Vehicles. *Computer Communications*", v. 120, p. 58–70, Feb. 2018.
- [3] SHAH, S. A. A. et al. "5G for Vehicular Communications", *IEEE Communications Magazine*, v. 56, n. 1, p. 111–117, Jan. 2018.
- [4] MARQUEZ, C. et al. "How Should I Slice My Network? A Multi-Service Empirical Evaluation of Resource Sharing Efficiency", *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking - MobiCom '18*, pp. 191-206. New Delhi, India, Oct. 2018.
- [5] DOS REIS FONTES, R. et al. "From Theory to Experimental Evaluation: Resource Management in Software-Defined Vehicular Networks". *IEEE Access*, v. 5, p. 3069–3076, Feb. 2017.
- [6] LI, X. et al. "Network Slicing for 5G: Challenges and Opportunities. *IEEE Internet Computing*", v. 21, n. 5, p. 20–27, Sep. 2017.
- [7] BOZKAYA, E.; CANBERK, B. "QoE-Based Flow Management in Software Defined Vehicular Networks". *IEEE Globecom Workshops*, pp. 1-6. San Diego, CA, Dec. 2015
- [8] LIU, J. et al. "A Scalable and Quick-Response Software Defined Vehicular Network Assisted by Mobile Edge Computing". *IEEE Communications Magazine*, v. 55, n. 7, p. 94–100, Jul. 2017.
- [9] AVELAR, E. et al. "Interoperability issues on heterogeneous wireless communication for smart cities". *Computer Communications*, v. 58, p. 4–15, Mar. 2015.

- [10] VAHDAT-NEJAD, H. et al. "A survey on context-aware vehicular network applications". *Vehicular Communications*, v. 3, p. 43–57, Jan. 2016.
- [11] HAGENAUER, F. et al. "Interconnecting smart cities by vehicles: How feasible is it?". *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 788–793. San Francisco, CA, Abr. 2016
- [12] SAINI, M.; ALELAIWI, A.; SADDIK, A. EL. "How Close are We to Realizing a Pragmatic VANET Solution? A Meta-Survey". *ACM Computing Surveys*, v. 48, n. 2, p. 1–40, 3. Nov. 2015.
- [13] BONOLA, M. et al. "Opportunistic communication in smart city: Experimental insight with small-scale taxi fleets as data carriers". *Ad Hoc Networks*, v. 43, p. 43–55, Jun. 2016.
- [14] SJOBERG, K. et al. *Cooperative Intelligent Transport Systems in Europe: Current Deployment Status and Outlook*. *IEEE Vehicular Technology Magazine*, v. 12, n. 2, p. 89–97, Jun. 2017.
- [15] SANGUESA, J. A. et al. "A Survey and Comparative Study of Broadcast Warning Message Dissemination Schemes for VANETs". *Mobile Information Systems*, v. 2016, p. 1–18, Mar 2016.
- [16] CARPENTER, S. E. "Obstacle Shadowing Influences in VANET Safety". *IEEE 22nd International Conference on Network Protocols*, pp. 480–482. Raleigh, NC, Oct. 2014
- [17] HUANG, C.; LU, R.; CHOO, K.-K. R. "Vehicular Fog Computing: Architecture, Use Case, and Security and Forensic Challenges". *IEEE Communications Magazine*, v. 55, n. 11, p. 105–111, Nov. 2017.
- [18] MOLINA-MASEGOSA, R.; GOZALVEZ, J. "LTE-V for Sidelink 5G V2X Vehicular Communications: A New 5G Technology for Short-Range Vehicle-to-Everything Communications". *IEEE Vehicular Technology Magazine*, v. 12, n. 4, p. 30–39, Dez. 2017.
- [19] CUNHA, F. et al. "Data communication in VANETs: Protocols, applications and challenges". *Ad Hoc Networks*, v. 44, p. 90–103, Jul. 2016.
- [20] IEEE-Standards Association, Standards Development Working Group, "1609 - Dedicated Short Range Communication Working Group". 2017

- [21] RENDA, M. E. et al. "IEEE 802.11p VANets: Experimental evaluation of packet inter-reception time". *Computer Communications*, v. 75, p. 26–38, Feb. 2016.
- [22] ZHENG, K. et al. "Soft-defined heterogeneous vehicular network: architecture and challenges". *IEEE Network*, v. 30, n. 4, p. 72–80, Jul. 2016.
- [23] FOUKAS, X. et al. "Network Slicing in 5G: Survey and Challenges". *IEEE Communications Magazine*, v. 55, n. 5, p. 94–100, May. 2017.
- [24] GOMES, N. J. et al. "Boosting 5G Through Ethernet: How Evolved Fronthaul Can Take Next-Generation Mobile to the Next Level". *IEEE Vehicular Technology Magazine*, v. 13, n. 1, p. 74–84, Mar. 2018.
- [25] TALEB, T. et al. "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration". *IEEE Communications Surveys & Tutorials*, v. 19, n. 3, p. 1657–1681, May 2017.
- [26] ISMAIL, B. I. et al. "Evaluation of Docker as Edge computing platform". *IEEE Conference on Open Systems (ICOS)*, pp. 130-135, Ago. 2015
- [27] KU, I. et al. "Towards software-defined VANET: Architecture and services". 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET). pp. 103-110. Piran, Jun. 2014
- [28] HE, Z.; CAO, J.; LIU, X. SDVN: enabling rapid network innovation for heterogeneous vehicular communication. *IEEE Network*, v. 30, n. 4, p. 10–15, Jul. 2016
- [29] KAUL, A. et al. "Dynamically Distributed Network Control for Message Dissemination in ITS". *IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pp. 1-9, Rome, Oct. 2017
- [30] TEGUEU, F. S. et al. "Towards application driven networking," *IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pp. 1-6, Rome, Jun. 2016
- [31] KHAN, A. A.; ABOLHASAN, M.; NI, W. "5G next generation VANETs using SDN and fog computing framework", 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 1-6, Las Vegas, NV, Jan. 2018.
- [32] CORREIA, S.; BOUKERCHE, A.; MENEGUETTE, R. I. An Architecture for Hierarchical Software-Defined Vehicular Networks. *IEEE Communications Magazine*, v. 55, n. 7, p. 80–86, Jul. 2017.

- [33] TAO, M. et al. "Vehicular Data Cloud Platform with 5G Support: Architecture, Services, and Challenges," IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), pp. 32-37, Guangzhou, Jul. 2017.
- [34] WHAIDUZZAMAN, M. et al. "A survey on vehicular cloud computing". Journal of Network and Computer Applications, v. 40, n. 1, p. 325–344, Abr. 2014
- [35] FONTES, R. R. et al. "Mininet-WiFi: Emulating software-defined wireless networks," 11th International Conference on Network and Service Management (CNSM), pp. 384-389. Barcelona, Nov. 2015
- [36] LANTZ, B.; HELLER, B.; MCKEOWN, N. "A network in a laptop". In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Network. p. 1–6. New York, USA. Oct. 2010
- [37] PFAFF, B. et al. "The Design and Implementation of Open vSwitch". NSDI. pp. 117-130. May. 2015
- [38] CZIVA, R. et al. SDN-Based Virtual Machine Management for Cloud Data Centers. IEEE Transactions on Network and Service Management, v. 13, n. 2, p. 212–225, Jun. 2016.
- [39] LI, H.; DONG, M.; OTA, K. "Control Plane Optimization in Software-Defined Vehicular Ad Hoc Networks". IEEE Transactions on Vehicular Technology, v. 65, n. 10, p. 7895–7904, Oct. 2016.
- [40] SARAIVA, T. V.; CAMPOS, C. A. V. "Comunicações veiculares em cenários realísticos usando diferentes traces de mobilidade". XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais - SBrT2017. São Pedro, SP. pp. 954–958, Sep. 2017.
- [41] SARAIVA, T. V.; CAMPOS, C. A. V. "O impacto na comunicação das redes veiculares em função de parâmetros escolhidos em ambiente de simulação realística". XLIX Simpósio Brasileiro de Pesquisa Operacional. Blumenau, SC. Ago. 2017
- [42] YU, Y. et al. "End-to-end throughput evaluation of consensus TPC algorithm in multihop wireless networks," International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 941-946, Dubrovnik, Aug. 2015
- [43] AL-HOURANI, A.; GOMEZ, K. Modeling Cellular-to-UAV Path-Loss for Suburban Environments. IEEE Wireless Communications Letters, v. 7, n. 1, p. 82–85, Feb. 2018.

- [44] SAID, A. M. et al. Modeling interactive real-time applications in VANETs with performance evaluation. *Computer Networks*, v. 104, p. 66–78, Jul. 2016.
- [45] SAHOO, P. K.; YUNHASNAWA, Y. "Ferrying vehicular data in cloud through software defined networking," *IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1-8. New York, NY, Oct. 2016
- [46] HASHEM EIZA, M. et al. "Situation-Aware QoS Routing Algorithm for Vehicular Ad Hoc Networks". *IEEE Transactions on Vehicular Technology*, v. 64, n. 12, p. 5520–5535, Dez. 2015
- [47] DUAN, X.; LIU, Y.; WANG, X. SDN Enabled 5G-VANET: Adaptive Vehicle Clustering and Beamformed Transmission for Aggregated Traffic. *IEEE Communications Magazine*, v. 55, n. 7, p. 120–127, Jul. 2017