Universidade do Minho
Escola de Engenharia

Luís Carlos Sousa Magalhães

SDN support on V2X devices

month, year

Universidade do Minho
Escola de Engenharia

Luís Carlos Sousa Magalhães

SDN support on V2X devices

Master Thesis
Master in Informatics Engineering

Work developed under the supervision of:
António Costa
Maria João Nicolau

month, year

Resumo

Suporte de SDN em dispositivos V2X

Relativamente ao seu conteúdo, os resumos não devem ultrapassar uma página e frequentemente tentam responder às seguintes questões (é imprescindível a adaptação às práticas habituais da sua área científica):

1. Qual é o problema?
2. Porque é que é um problema interessante/desafiante?
3. Qual é a proposta de abordagem/solução?
4. Quais são as consequências/resultados da solução proposta?

Palavras-chave: Redes vehiculares, Redes definidas por software

Abstract

SDN support on V2X devices

Concerning its contents, the abstracts should not exceed one page and may answer the following questions (it is essential to adapt to the usual practices of your scientific area):

1. What is the problem?
2. Why is this problem interesting/challenging?
3. What is the proposed approach/solution/contribution?
4. What results (implications/consequences) from the solution?

Keywords: Vehicular networks, Software defined networks

Contents

Annexes

List of Figures

List of Tables

Acronyms

3GPP	3rd Generation Partnership Project *(pp. 17, 18)*
5GAA	5G Automotive Association *(pp. 17, 18)*
AU	Autonomous Unit *(pp. 6–8, 11)*
BSS	Basic Service Set *(pp. x, 15)*
BTP	Basic Transport Protocol *(p. 19)*
C-V2X	Cellular V2X *(pp. 17, 18, 26)*
C2C-CC	Car-2-Car Communication Consortium *(pp. vii, 4, 6–10, 12)*
CAM	Cooperative Awareness Message *(pp. vii, 20, 21)*
CEN	European Committee for Standardization *(p. 20)*
DENM	Decentralized Environmental Notification Message *(pp. vii, 20, 21)*
ECC	Electronic Communications Committee *(pp. 15, 16)*
ETSI	European Telecommunications Standards Institute *(pp. 4, 6–8, 10, 11, 15, 16, 18–21, 24, 25)*
FCC	Federal Communications Commission *(pp. vii, 15, 16)*
GN6	GeoNetworking-IPv6 *(p. 19)*
IBSS	Independent Basic Service Set *(p. 15)*
IEEE	Institute of Electrical and Electronics Engineers *(pp. 14, 15, 17)*
IP	Internet Protocol *(pp. ix, 7, 26)*
IPv6	Internet Protocol version 6 *(pp. ix, 19, 26)*
ISO	International Organization for Standardization *(p. 20)*

ITS	Intelligent Transport System *(pp. vii, x, 4, 7–18, 20–22, 24, 25, 27, 49, 51–53)*
ITS-S	ITS station *(pp. 7, 8)*
ITSC	Architecture of Communications in ITS *(p. 11)*
MAC	Data Link *(pp. 13, 14)*
MANET	Mobile ad hoc Network *(pp. 3–5, 18)*
NOS	Network Operating System *(pp. 44, 46, 47)*
OBU	On Board Unit *(pp. vii, 6, 7, 9, 11)*
OCB	Out of Context BSS *(pp. 15, 17)*
ODL	Open Day Light *(p. 46)*
ONF	Open Networking Foundation *(pp. 28, 43)*
ONIE	Open Network Install Environment *(p. 42)*
ONOS	Open Network Operating System *(p. 46)*
P4	Programming Protocol-independent Packet Processors *(pp. 30, 37–39, 41, 43)*
PHY	Physical *(pp. 13, 14)*
PISA	Protocol Independent Switching Architecture *(p. 41)*
POX	Pythonic Network Operating System *(p. 46)*
PSA	Portable Switch Architecture *(p. 41)*
QoS	Quality of Service *(pp. 6, 53)*
RSU	Roadside Unit *(pp. vii, 6, 7, 10, 11)*
SD-VANET	Software Defined Vehicular ad hoc Network *(p. 48)*
SDN	Software defined networking *(pp. vii, 28–39, 42–53)*
SDVN	Software Defined Vehicular Network *(pp. vii, 48–52)*
SNMP	Simple Network Management Protocol *(p. 43)*
TNA	Tofino Native Architecture *(p. 41)*
UDP	User Datagram Protocol *(p. 19)*
V2I	Vehicle to Infrastructure communication *(pp. 12, 15)*
V2V	Vehicle to Vehicle communication *(pp. 12, 15)*
V2X	Vehicle to Everything communication *(pp. ix, 12, 14, 18)*

VANET	Vehicular ad hoc Network *(pp. vii, x, 3–7, 9, 10, 12–15, 17–19, 21–23, 25, 26, 48–51, 53)*
WAVE	Wireless Access in Vehicular Environments *(pp. 14, 15)*

1 Introduction

This chapter serves as an introduction to this thesis, providing an overview of its context, motivation, objectives, and methodology.

1.1 Contextualization

The Internet is universally acknowledged as one of the most significant technological advancements in human history, with its widespread adoption and the numerous benefits it has brought about having irrevocably transformed the manner in which societies in the 21st century live and communicate.

Despite the Internet's numerous strengths, it is evident that, in its current form, it is not fully capable of satisfying the demands of the modern digital landscape. One such problem relevant to this thesys stems from the fact that our society is becoming increasingly mobile. With the widespread adoption of automobiles, researchers and engineers soon realised that allowing these these vehicles to comunicate with each other could offer major benefits to road security and could even bring quality of life improvements to drivers.

A new networking paradigm, software defined networks, promises to offer more innovation, reduced cost, simplify management and improve performance. SDN is poised to address the limitations of the Internet and facilitate further growth.

As such, in recent years, researchers have turned to this new and exciting field, with SDN promising a solution to many of the challenges faced by the current internet landscape.Therefore, a lot of recent studies have applied SDNs to vehicular networks in order to create better solutions for the future. This thesis aims to introduce SDN principles to V2X communication.

1.2 Motivation

In recent years, there has been a growing interest in the application of SDN technology in vehicular networks, with a large number of projects in SDVNs being created. Even doe these projects aim to be used in the real world, the large majority of them is restricted to simulation-based testing which reduces the veracity of the results. Therefore, this highlights a pressing need for the availability of real hardware testing platforms, in order to achieve more accurate and reliable results in such a critically important area. The development of a real hardware testing platform for SDVN projects will allow for a more comprehensive evaluation of the technology, which is a step further for this technology to become accepted and used.

1.3 Objectives

Our main goal in this thesis is to design and assemble a hardware device similar to an OBU, utilizing open-source software where possible. This device must be SND compatible, so its routing tables must be configurable by a local or remote SDN controller. To meet these requirements, the device will run an open-source V2X protocol stack on a Linux-based OS.

1.4 Methodology

Initially, an investigation will be done to find the software required for the implementation of an OBU controlled by a free controller. Subsequently, a new OBU device will be designed and assembled, using modern hardware. Finally, performance and security vulnerabilities will both be tested.

1.5 Document structure

Chapter 2 is divided into three main sections. Section 2.1. aims to provide a broad context about vehicular networks, in order to understand the main challenges in this field. In section 2.2., a very detailed overview of SDN is made, with a large emphasis on the data and control plane, to show the power and potential of this technology. Section 2.3. serves to address what are the advantages and shortcomings of using SDN in a vehicular context. In Chapter 3, a state of play is made, where the plan to achieve the goals of this thesis is highlighted.

2 Vehicular ad hoc Networks

The automobile was one of the most revolutionary inventions in human history, for most cities around the world are dependent on cars for the transportation of goods and people and, therefore, the prevalence of vehicles in our daily lives is greater than ever before. Even though it is impossible to predict if future vehicle usage will continue to be as dominant as today, it is reasonable to assume that this technology will never disappear completely. Even today, in cities dominated by alternative modes of transportation, the use of motorized vehicles is still present and, in some cases, a necessity.

It has been a long-time goal for car manufacturers and researchers to bring internet connectivity to automobiles, as the benefits are enormous. The Internet is an extremely powerful technology, not only for the countless services it provides but also for the ability of two endpoints to communicate almost instantly. Moreover, recent improvements to wireless communication technologies not only make this goal more realistic but also even more attractive, as integrating network interfaces with GPS receivers and sensors only increases the potential of allowing cars to communicate with each other [1].

The efforts to bring connectivity to automobiles resulted in the emergence of the technology named VANET, which is an adaptation of the Mobile ad hoc Network (MANET) technology, specialized for automobiles [2] [3]. In contrast to wired networks, where all devices are connected to the infrastructure at all times to communicate with each other, this technology allows devices to transmit data directly to other devices inside its range, as illustrated in Figure 1. As a result, networks are generated spontaneously between nodes that are in range of each other, which makes topologies unstable and requires each node in the network to act as both a transmitter and a receiver.

As the Internet was not designed with mobility in mind, most protocols and standards were not developed to support the high-speed scenarios of VANET. A tremendous amount of work and research efforts has already been conducted in this area, amounting to decades of research, development, and standardization. However, the development of VANET technology began without any major centralized entity to standardize its implementation worldwide, resulting in various architectures emerging simultaneously all around the globe. It is common for different regions of the world to develop different standards for infrastructure-related technologies and these solutions tend to converge as they seek to solve the same problems in the most optimal way.

The leaders in global research on VANETs are considered to be the United States, Japan, and the

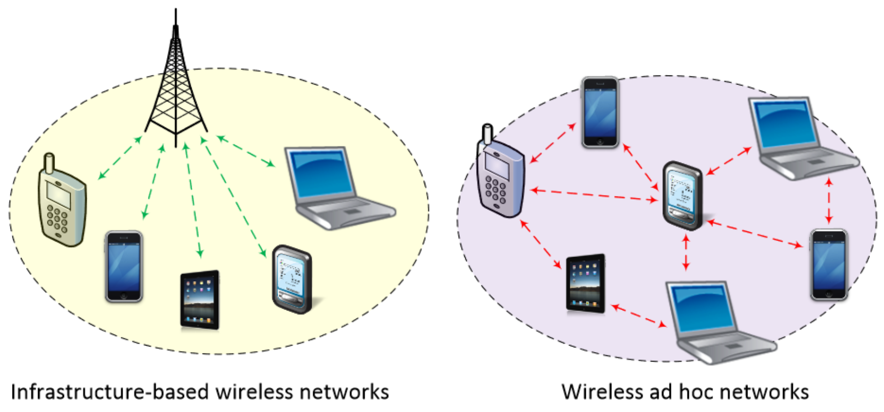

Figure 1: Comparison between infrastructure networks [4]

European Union. These regions are noteworthy as they have provided the majority of the technological foundations and advancements in the field. For the sake of clarity and relevance, this paper focuses on the European VANETs architecture through the discussion of the C2C-CC and the European Telecommunications Standards Institute (ETSI), two prominent and influential institutions in the field.

The C2C-CC architecture proposal is designed with a high level of abstraction and is therefore fairly easy to understand. In contrast, the ETSI standards are significantly more detailed and technical, since they provide the official standard for the European Union. ETSI uses the term ITS to refer to VANETs, as their efforts to connect vehicles together goes beyond cars and aims to interconnect planes, trains, boats, etc.

There are other conventions in this field, such as the C-ROADS platform, which are important for the development of this technology. In fact, several conventions have influenced the ETSI standards and would be worth discussing. However, in order to avoid overcomplicating the subject, only the ETSI and C2C-CC architectures will be covered.

2.1 VANET characteristics

The VANET technological domain was established as a specialized subset of the MANET research field based on some unique characteristics, and thus VANET shares most of its properties with MANET. Some key differences require the development of novel solutions, and in this section, we will go through all the characteristics of VANET while shedding light on what sets this technology apart from its origin.

2.1.1 High mobility

MANET topologies are highly dynamic, driven by the mobility of nodes. High mobility leads to uncertain connectivity, which makes ad-hoc topologies extremely volatile and unpredictable, with a high chance of partitions[5]. VANETs not only inherit this characteristic but also amplify its volatility since VANET nodes are vehicles capable of traveling at extremely high speeds.

For instance, two cars traveling in opposite directions on the same road will only be within range of each other for a few seconds. In this short time, the vehicles must attempt to establish a connection before attempting to exchange valuable information, which means some links may be disconnected before there is a chance to use them [3]. On the other hand, two cars traveling in the same direction can maintain a connection indefinitely as long as they keep on the same path at the same speed. These two scenarios are polar opposites, making the reliability of connections inconsistent.

2.1.2 Predicted mobility

Beyond being more dynamic, MANET topologies are also random, as nodes can move arbitrarily and at unpredictable times, with virtually no restraint. VANET topologies diverge from MANET in this matter, as it is possible to reliably predict the path of nodes in an ad-hoc network. Vehicles are primarily used on pre-built infrastructure [3], like roads and motorways, and have to follow predetermined paths to reach their destination, which makes it possible to predict their future location. Vehicles are also forced to obey road signs and traffic lights and have to respond to other vehicles' movements[2], making topologies even more predictable.

On top of all of that, if drivers disclose their desired destination it becomes possible to predict the complete route of a node with near 100% precision. It is important, however, to keep in mind that this is not always true, and in some instances, drivers may adjust their path in reaction to the information received by the network[3].

A negative consequence of road use can be seen in more linear topologies when compared with MANETs, which inevitably undermines path redundancy[5].

2.1.3 Power and Computational ability

Another relevant change from MANETs is the computational capabilities and power constraints of the nodes. MANET was designed with small devices in mind, which have limited battery capacity and computational power constraints. In contrast, vehicles have access to a reliable power source[3][1], so there are essentially no concerns in this regard. However, cars should not be thought of as an infinite power source, and normal precautions should be taken when optimizing routing algorithms, etc.

2.1.4 Congestion and Scalability issues

VANET suffers from an absurdly variable node density. A vehicle can be on a road in a remote location with the nearest internet connection of other vehicle kilometers away from their location, or in the middle of a traffic jam at an intersection surrounded by hundreds of vehicles all trying to communicate within its range.

Congestion scenarios create a bandwidth problem that is exacerbated by the presence of nearby obstacles like other cars and buildings, which common in an urban environment[5]. Wireless links are significantly more fragile than their wired counterparts because they are subject to fading, noise and interference[6]. This also forces researchers to come up with effective measures for sharing physical media, as another issue of concern is ensuring Quality of Service (QoS) measures for critical messages, like road hazard warnings, which presents a core issue[5].

Variable network density presents a necessity for the protocols developed to be capable of dealing with any situation thrown at them in the real world.

2.2 VANET components

The first step in the understanding VANET scenarios is the analysis of the basic elements involved. With this in mind, the atomic components of both the C2C-CC and ETSI are explained in this section.

2.2.1 C2C-CC approach

According to the C2C-CC architecture, three main components can be identified, these being the RSU, OBU and Autonomous Unit (AU).

1. Roadside Units are stationary devices placed alongside roads, usually in high-density zones such as junctions, intersections and traffic lights.

   These devices have a reliable connection to the Internet and communicate with all vehicles within their effective range, bridging the gap between vehicle ad hoc networks and the rest of the Internet. RSUs can also be equipped with multiple communication devices for optimal wireless connectivity. In addition, RSUs can also take advantage of the ad hoc characteristics of on-board units by using them as transmitters, thereby extending their effective range, as depicted on Figure 2.

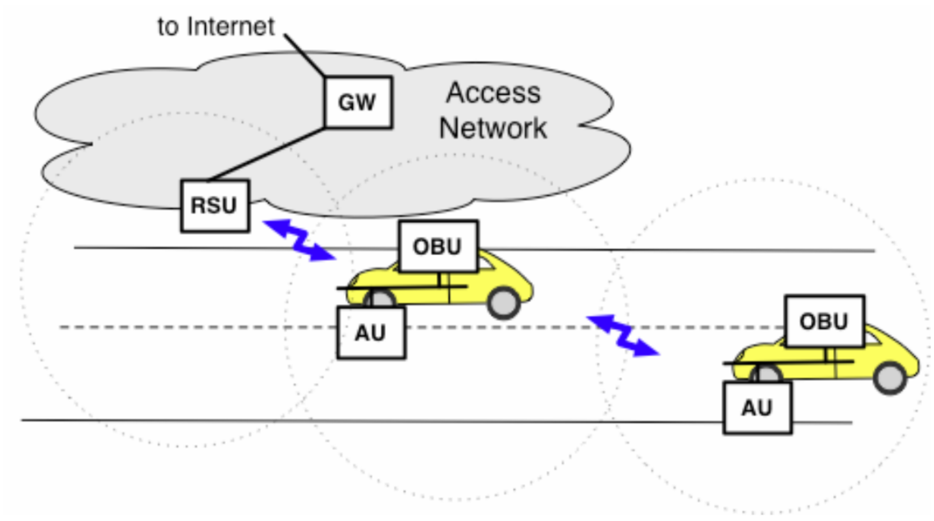

Figure 2: A RSU provides Internet connectivity through an OBU. [7]

RSUs can run safety applications like fixed road hazard warnings (work-zone warnings, bridge warnings, etc.) by broadcasting safety messages to nearby vehicles, making them more than mere connection points for vehicles.

2. On Board Unit is a small device integrated into vehicles that operates in the ad hoc space as they receive, send, and forward messages between each other and with RSUs in an ad hoc manner. OBUs provide it's wireless communication capabilities to any connected device used within the vehicle.

3. The term Autonomous Units refers to all the various devices that run a single or a set of applications that connect to the OBU and harness its wireless capabilities. The AUs can be physically integrated or permanently wired to the OBU being an integrated part of the vehicle. These types of AUs are typically safety-related applications. AUs can also be wireless devices like laptops or smartphones, wirelessly connecting to an OBU and utilizing its capabilities for leisure applications.

4. In addition to these three main components, this institution acknowledges the potential presence of other network entities, like public hotspots, instances of a Mobile Internet Protocol (IP) infrastructure, application servers, control centers, etc. However, these are not categorized and are therefore outside of the scope of this Consortium.

2.2.2 ETSI approach

The ETSI standard diverges from the simple and straightforward definition of the C2C-CC by specifying two categories to define the communication entities of VANETs: the ITS sub-systems and the functional components. The former, comparable to the C2C-CC components, represent similar system categories that exist in the VANET space. The latter are the building blocks that make up the ITS subsystems and are therefore responsible for specific functions or tasks within the overall systems.

2.2.2.1 Functional components

The functional components described in this architecture follow the layered conceptual structure of the OSI model [8], comprising five such components: the ITS station (ITS-S) host, the ITS-S gateway, the ITS-S router, the ITS-S border router and the ITS-S interceptor.

The most significant component of this architecture is the ITS-S host, since it provides the minimum level of functionality required to run ITS applications. It represents the basic reference architecture for any device wishing to communicate in the VANET space, and is present on all ITS sub-systems. This architecture is described in depth in section 2.5

In addition, all other components are a variant of the ITS-S host, adding communication capabilities to it. The ITS-S gateway converts messages from the Internet to the ITS domain at layers 5 to 7 of the OSI model. The ITS-S router converts various ITS protocols at layer 3, and the ITS-S border router performs a similar task, doing the same as the router, but with a more generic network, without management or security awareness. The ITS-S interceptor is a generic term equivalent to any of the previous three components, plus it can represent an implementation-specific connection of the ITS station-internal network to another network.

2.2.2.2 Sub-systems

As previously introduced, all functional components come together to form various sub-systems, of which there are four. These sub-systems represent a specific context in which the functional components combine to attempt to provide wireless communication capabilities, either in the form of direct contact with the ad hoc environment or indirectly.

All ITS sub-systems consist of the unique context they support and a specific ITS station to meet the needs of that unique situation. These stations can also be implemented in a single physical unit or in multiple physical units.

1. Personal ITS sub-system:

   The Personal ITS sub-system represents a mobile or other personal device that can connect to the vehicle's internal network and use its ad-hoc wireless capabilities. It includes the user's device and a Personal ITS Station. This station is the simplest of the four, consisting of a single ITS host. This can be observed in Figure 3.

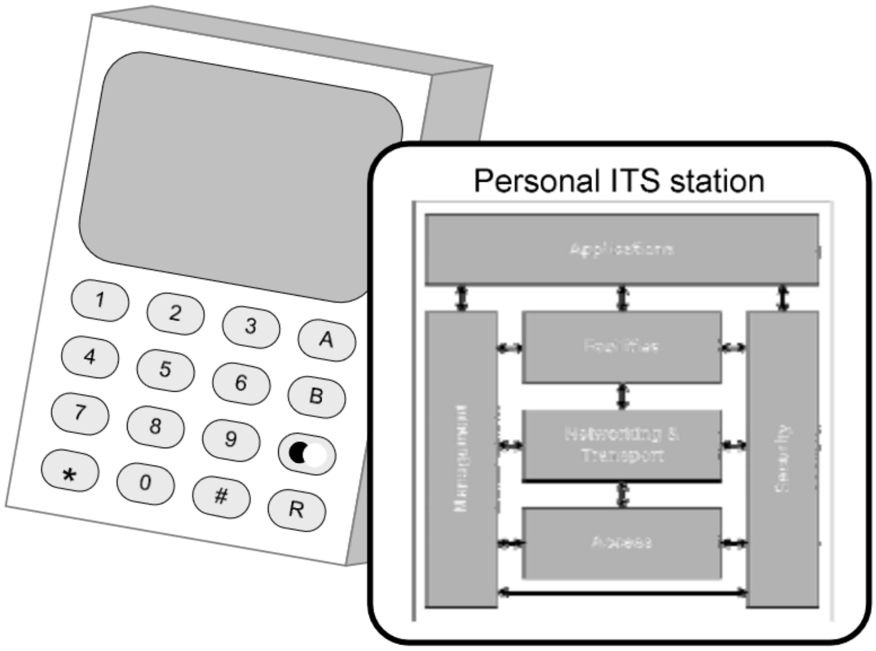

Figure 3: Personal ITS station in a Personal ITS sub-system [8]

This sub-system is comparable to an AU, with a narrower definition. The ETSI architecture defines the existence of an ECU, which also falls under the AU definition created by the C2C-CC. The ECU will be described along with the vehicle ITS sub-system.

2. Central ITS sub-system; The purpose of the Central ITS sub-system is to provide centralized ITS applications, like traffic operations or content delivery, to VANET users. It is composed of a central ITS station that can be located anywhere. Examples of a Central ITS sub-system are traffic management centers and road operator centers.

   The Central ITS station contains an ITS host and two optional ITS interceptors, being an ITS gateway and border router. The Central ITS sub-system is represented in Figure 4.

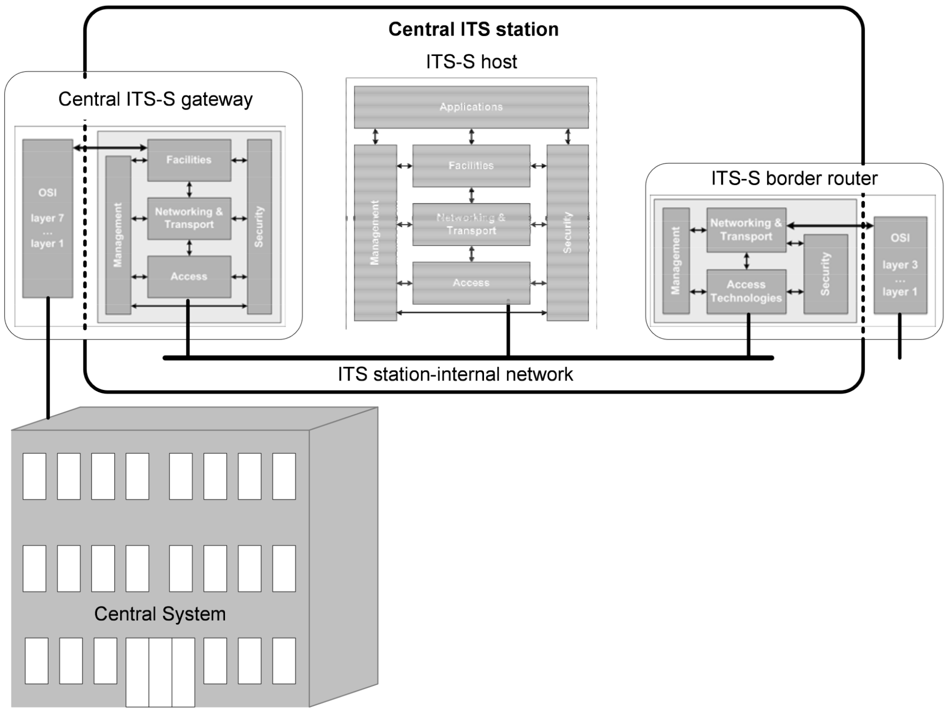

Figure 4: Central ITS station in a Central ITS sub-system [8]

   The border router communicates with any other ITS system via any external network, while the gateway provides a more standardized connection.

3. Vehicle ITS sub-system:

   The Vehicle ITS sub-system represents the cars, buses, trains, trucks, airplanes, and any other vehicle or means of transportation that can communicate wirelessly in an ad hoc manner and belongs to the VANET space. This definition is very similar to the OBU definition made by the C2C-CC, but there are some important minor differences that make it more rigorous.

   This subsystem is made up of the Vehicle ITS station and the internal vehicle network, which is connected to a variety of Electronic Control Units (ECUs). ECUs represent any kind of system or application that requires a connection to the Internet or to any other vehicle. Examples of such applications are GPS, traffic control, road hazard warnings, etc.

   Like all stations, the Vehicle ITS station is made up of an ITS host with two optional ITS interceptors, nominally an ITS gateway and an ITS router. This sub-system is depicted in Figure 5.

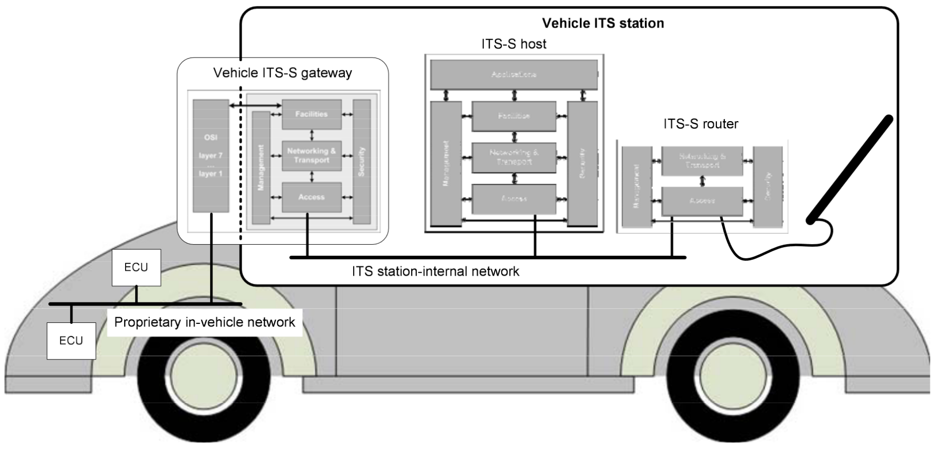

Figure 5: Vehicle ITS station in a Vehicle ITS sub-system [8]

The ITS gateway connects the internal vehicle network, which is proprietary in most cases, to the ITS station. Meanwhile, the ITS router connects to other Vehicle ITS stations and, most importantly, to the Roadside ITS station.

4. Roadside ITS sub-system:

   The final element is the Roadside ITS sub-system. It is the connection point between the vehicle ad hoc network and the rest of the Internet. Like the RSU, it is placed in locations of high vehicle volume next to roads and intersections.

   This sub-system comprises a roadside ITS station and any proprietary roadside network that may exist. It connects to any vehicle ITS station and can provide ITS applications to the vehicle users. This station is also connected to a central ITS station, being the bridge that allows the central ITS station to provide ITS applications to vehicle users.

   The roadside station is made up of an ITS host with optional ITS interceptors. These interceptors can be any of the other types of functional components beyond the ITS host. The Roadside ITS sub-system is better depicted in Figure 6.

   The ITS router connects the station to the ITS vehicle station, the border router connects to an ITS central station and the gateway connects to any existing proprietary existing networks.

2.3 Communication Domains

With the main building blocks of both architectures laid out, we can divide communications into various domains. Both the C2C-CC and ETSI have different ways to tackle this question.

2.3.1 C2C-CC approach

The C2C-CC focuses on the VANET space and divides it into three areas: the in-vehicle domain, the ad hoc domain, and the infrastructure domain. The first represents the network inside a vehicle, composed of an

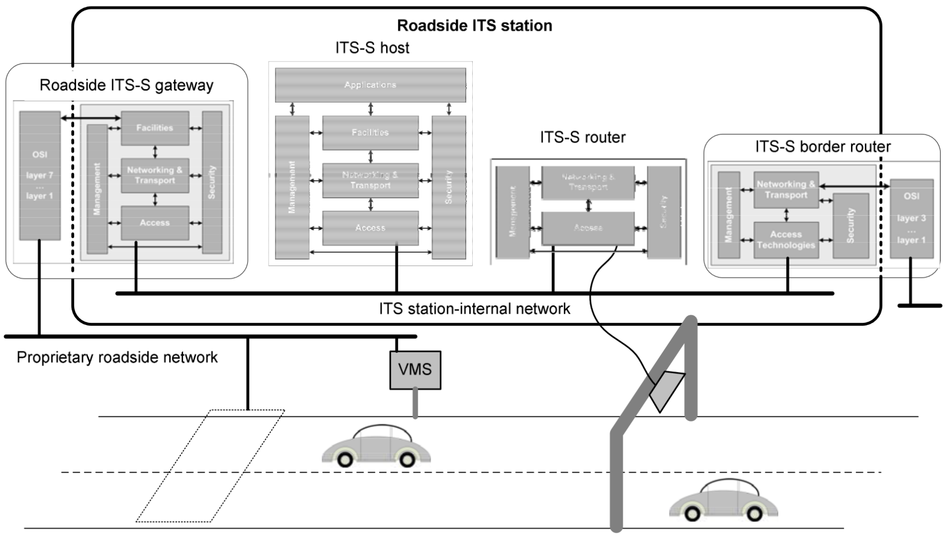

Figure 6: Roadside ITS station in a Roadside ITS sub-system [8]

OBU and the multitude of AUs connected to it. It encompasses all communications between the different AUs and the OBU. The ad hoc domain involves all ad hoc communications between different OBUs and between vehicles' OBUs with RSUs. The remaining components of the Internet that don't directly affect it, but help provide connectivity to vehicles, are included in the Infrastructure domain.

All of these domains, as well as the functional components of the C2C-CC architecture can be observed on Figure 7.

2.3.2 ETSI approach

The ETSI architecture proposes a broader categorization by establishing two domains. The first and most important is the ITS domain, since it contains all the elements and communication engaged by the ITS/Architecture of Communications in ITS (ITSC) standards. Everything else is encompassed by the generic domain, including everything in the rest of the Internet that might interact and communicate with the ITS domain.

2.4 Communication categories

In addition to communication domains, most authors use a classification based on communication types. This communication-oriented view of architecture aims to classify the multitude of types of communication that can occur between all the different entities that exist.

Different categories have emerged and evolved throughout the years as they have been used by most of the relevant researchers in the field. It is therefore difficult to find an origin for these concepts, and the following are most commonly used terms.

1. Intra-Vehicle communication This term encompasses all communications that occur inside the vehicle between different vehicle components.

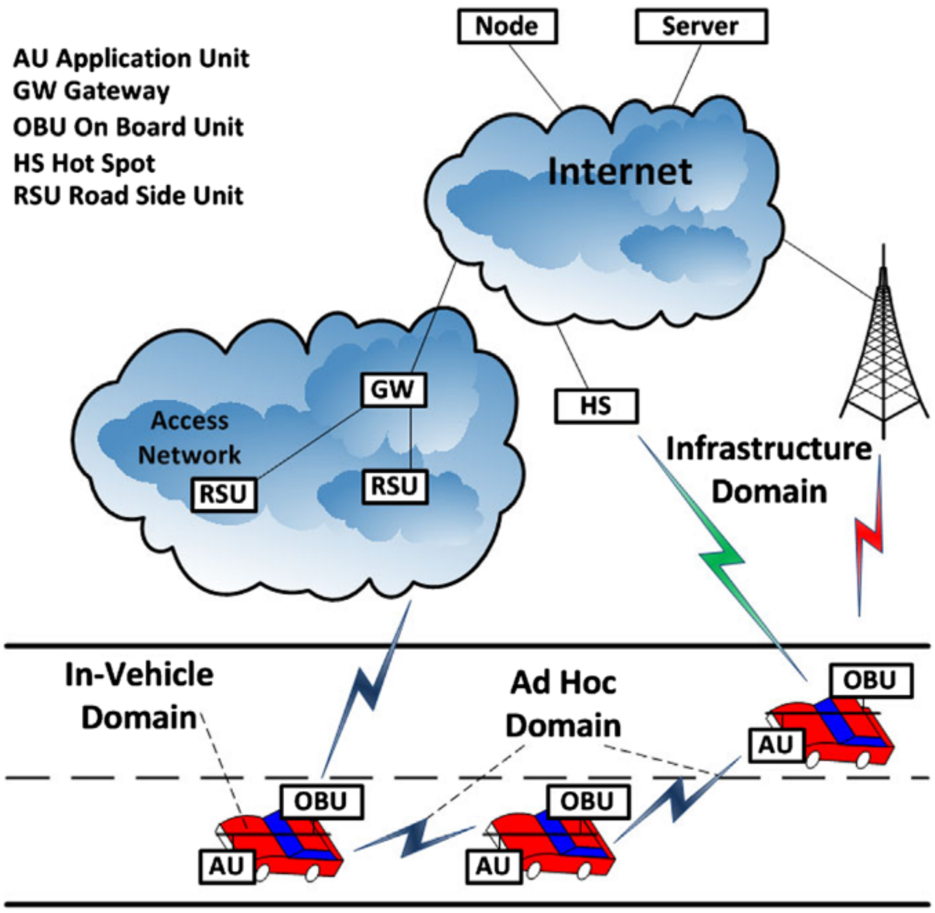

Figure 7: Communication domains in VANET as categorized by the C2C-CC [2]

2. Vehicle to Vehicle communication (V2V) communication V2V communication refers to the ad hoc communication between vehicles.

3. Vehicle to Infrastructure communication (V2I) communication V2I communication refers to ad hoc communication between vehicles and roadside infrastructure. These communications dont encompass all messages that get exchanged between vehicles and the roadside infrastructure, and only include messages whose source and destination are a vehicle and a roadside station.

4. Vehicle to Everything communication (V2X) The last type of communication is the most encompassing. V2X communication includes all communication that can occur in the VANET space between a vehicle and anything else. It is generally used to refer to the previous two types of communication.

2.5 VANET architecture

The architecture of an ITS host can be visualized on Figure 8. This section will dissect this architecture layer by layer in order to explain the functionalities and capabilities expected from it. The last layer, called Applications, will not be described as it represent ITS applications that use the services provided by the rest of the stack to connect one or more applications together and provide services to users[8].

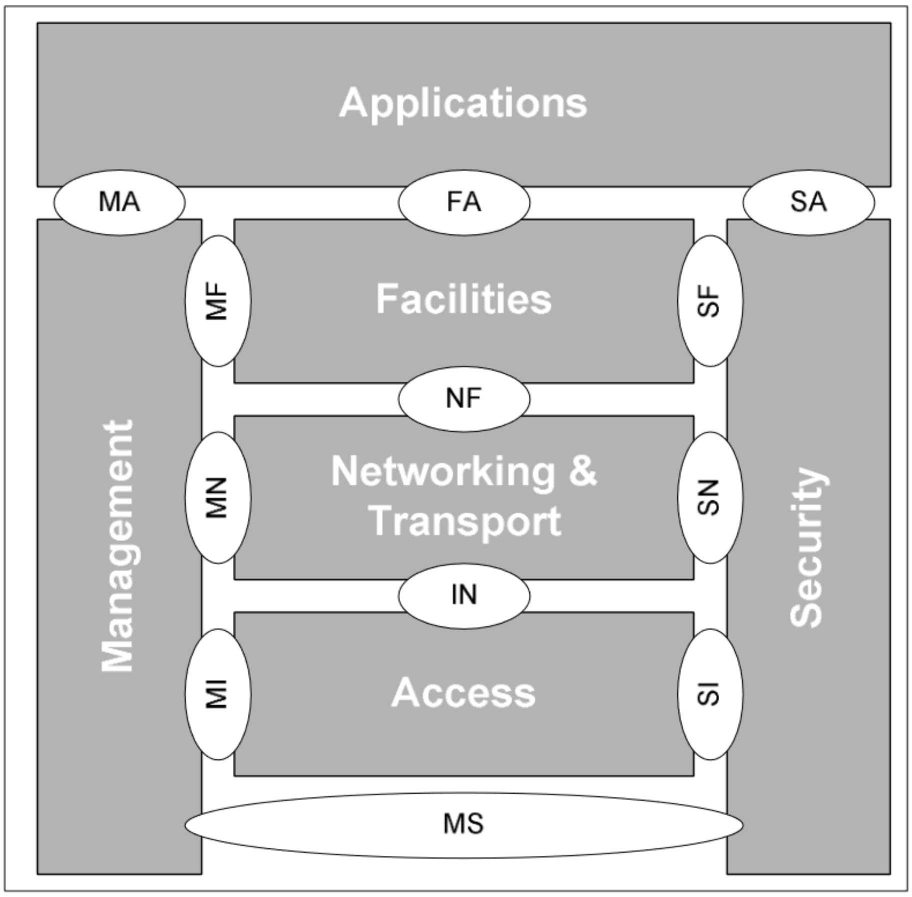

Figure 8: ITS station reference architecture [8]

The functionalities described in every layer are not necessarily function to be implemented in every instance of the architecture, as it depends on the specific implementation of the ITS station[8].

2.5.1 Access Layer

The Access Layer, which is illustrated in the architecture of an ITS host in Figure 8, corresponds to the first two layers of the OSI model, the Physical (PHY) Layer and the Data Link (MAC) Layer[9].

Among all the existing Access Layer technologies, the best suited to meet the necessary requirements for use in VANETs were the existing wireless access technologies[2]. These technologies ranged from long range signaling technologies such as Microwave and WiMAX to short range technologies such as Infrared or Bluetooth[10].

Over the years, most of these technologies have been studied and tested as possible solutions for use in VANET communications, and in most cases they were deemed unsuitable for the VANET environment. For instance, short-range communication protocols such as the examples above were discarded due to lack of range, throughput, and reliability and as a result, medium and long range communications took center stage.

While unable to fully address the demanding scenarios of VANETs, a few of these technologies showed promise as possible solutions. Over the years since VANET research began, new standards and protocols have been developed based on these existing technologies, focusing on solving the problems posed by the characteristics of VANETs. In addition, these technologies themselves have evolved to meet the high data rates required by today's Internet.

Today, the IEEE 802.11 WiFi and Cellular technologies are considered as the two primary solutions for VANETs worldwide.

2.5.1.1 IEEE-based technologies

The Institute of Electrical and Electronics Engineers (IEEE) is an American organization dedicated to promoting innovation and technological excellence for the benefit of humanity. Founded in 1963, it is the world's largest professional society of engineers, scientists, and other technical professionals. IEEEs primary areas of contribution are the electrical, electronic, and computing fields[11].

The IEEE 802.11 family of standards, an Access layer technology, was considered a promising solution for VANETs from the start. Not only would the existing 802.11 standards be cheaper to adopt due to being already established technologies, but WiFi also already met many ITS requirements due to its compatibility with mobile environments[12].

Despite these significant advantages, the existing IEEE 802.11 standards were not able to meet all the requirements imposed by the unique characteristics of VANETs, necessitating the creation of a new variant that would be better suited for such scenarios. Thus, 802.11p was created as an adaptation of the existing 802.11 standards. Given its American origin and contribution to the development of V2X communication technologies, it is important to mention that IEEE 802.11p was conceived in the context of Wireless Access in Vehicular Environments (WAVE)[13]. The EU, while initially open to many different access technologies, eventually converged on a IEEE 802.11p based solution dubbed ITS-G5.

IEEE's main motivation behind the development of IEEE 802.11p was to enable effective communication capable of handling the rapidly changing environments of VANETs[13], while making the minimum necessary changes to the IEEE 802.11 PHY layer. Modifying the MAC layer is akin to a software update, while a PHY level amendment would require the design of an entirely new wireless airlink technology[13], which would be expensive and time consuming.

With this in mind, the IEEE began work on a variant of the 802.11 standard that would be feasible for V2X, capable of operating at speeds up to 200 km/h and ranges up to 1 km[1]. Experimental work began with the full deck of 802.11 technologies[5], but soon narrowed down to IEEE 802.11a because it operates at 5 GHz, which is closest to 5.9 GHz, the desired frequency for ITS operations in the US. This made it easier to configure devices to operate at the desired frequency[13].

IEEE 802.11p uses orthogonal frequency division multiplexing. The 802.11 family of standards offers three channel bandwidths of 5, 10, and 20 MHz, and while 802.11a uses the full 20 MHz bandwidth, the 10 MHz bandwidth was chosen for ITS scenarios. This is achieved either by halving the clock/sampling rate or by simply doubling all the standard orthogonal frequency division multiplexing timing parameters[12].

Reducing the channel bandwidth was done in an effort to increase the root mean square delay spread[5]. Root mean square delay spread measures the time difference in seconds between the first and last signal components. These components represent different versions of the same transmitted signal, with the first to arrive coming from the shortest path which is the light of sight and the remaining coming from reflections and other interactions with the environment.

The standard 20MHz guard used by 11a proved sufficient to stop intersymbol interference between a radio and itself, so measures had to be taken to reduce the interference caused by multipath delays and by the Doppler effect. Halving channels width was the most simple and sensible solution to this problem[13]. Beyond reducing the signal bandwidth, the data throughput ranges were also reduced to 3 to 27 Mb/s instead of the original 6 to 54 Mb/s[5].

With the goal of enabling effective communications that can cope with rapidly changing environments, 802.11p communications needed to get faster in order to better utilize the sometimes narrow windows available for communication. Such a feat demanded a reduction in the overhead required before actual data transmission, which was achieved by simplifying the Basic Service Set (BSS) operations present in 802.11a[13].

The BSS represents a group of stations that are wirelessly connected to the same access point. It is created by an access point, and any device requesting to join it can exchange information with it after proper authorization. A BSS variant for ad hoc environments exists, called Independent Basic Service Set (IBSS), which works without the need for infrastructure. However, it also proved to be insufficient to deal with the peculiarities of VANETs.

As a solution, IEEE 802.11p uses an ad hoc mode called Out of Context BSS (OCB). OCB simplifies setup operations compared to its counterparts in other 802.11 standards by eliminating management procedures such as channel scanning, authentication, and association. This means that 802.11p has no setup time allowing stations to communicate with each other directly and immediately[14].

Removing all of these processes makes the network less secure. However, these security vulnerabilities are addressed by other standards in the ITS domain.

In Europe, ETSI created a solution for the access layer called ITS-G5. It not only defines the technologies and protocols to be used in communications, but also includes the frequency allocated exclusively for V2V and V2I communications for the whole of the EU.[15] ITS-G5 uses the existing 802.11p standards created for the American WAVE, adapting it for the European region.

In the US, the FCC was responsible for the allocation of the frequency from 5.850 to 5.925 in 1999. In 2003, the FCC divided the American band into seven non-overlapping 10 MHz channels with a 5 MHz unused band at the lower end. These channels were numbered from 172 to 184. This division also allows for operations in the 20 MHz channels 175 and 181, which are the overlap of two 10 MHz channels[16].

In the EU, the Electronic Communications Committee (ECC) is responsible for spectrum regulation and the European Commission is responsible for enforcing the regulated spectrum in all member states. The ECC is made up of radio and telecommunications regulatory authorities from all member countries[16][15].

The spectrum allocation for ITS-G5 began in 2005, with a recommendation from ETSI to the ECC. In its TR 102 492-1[17], ETSI recommended the allocation of 30 MHz in the 5,875 GHz to 5,905 GHz

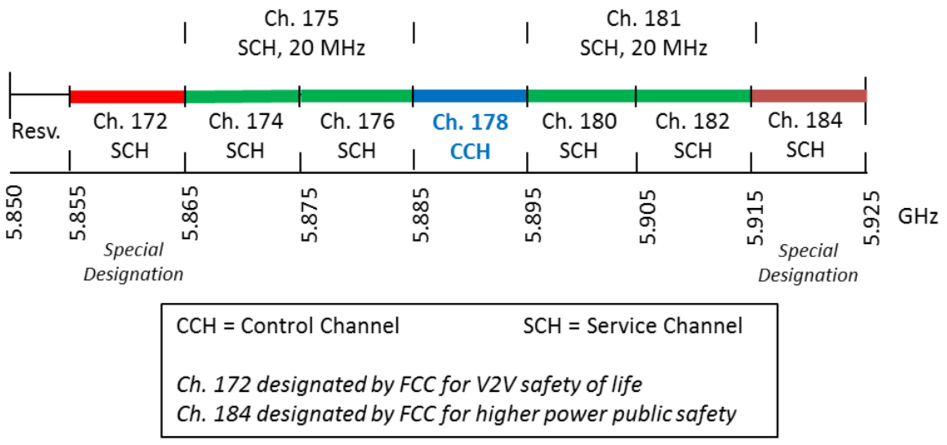

Figure 9: Channels allocated in the US by FCC for ITS [16]

band for safety applications, which would align with the US control channel frequency of 5,885 GHz to 5,895 GHz. This recommendation also expected the allocation of a future 20 MHz band from 5,905 GHz to 5,925 GHz for future ITS extensions[16][15].

ETSI based this recommendation on the allocation in the US frequency band, and also took into account the 5.8GHz toll collection band used in EU countries[16][15].

In 2008, the ECC accepted this recommendation and allocated the requested frequency range plus another 20 MHz frequency band from from 9,855GHz to 9.875GHz intended to be used by non-safety applications[16][15]. All of the spectrum involved for present and future ITS operations can be easily observer in Figure 10.

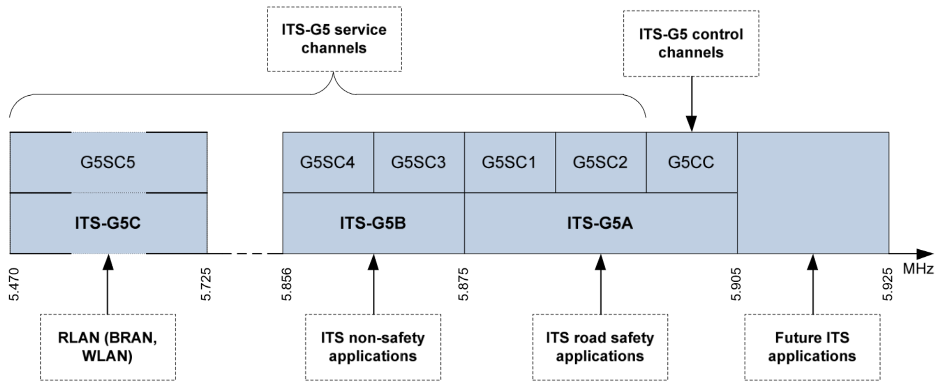

Figure 10: European ITS channel allocation [18]

Even though the American spectrum was taken into consideration, in the end the control channels of both the EU and US solutions were not in the same frequency, with the European channel being 10 MHz higher. This is not a total loss, as the same hardware can still be used in both countries requiring just a software change to work[16][15].

The different allocated frequencies have different standardized transmission power limits based on use cases and interferences on adjacent bands and on the more important control bands[14]. The specific restrictions for each 10 MHz channel can be observed in figure 11.

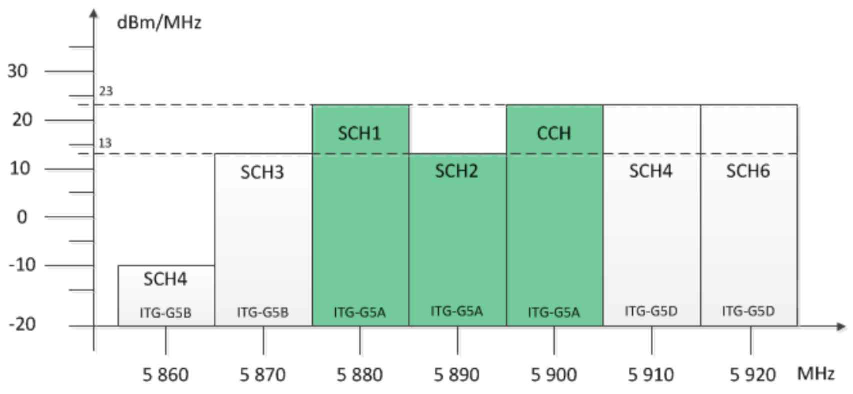

Figure 11: Power spectral limits on each ITS channel [16]

In recent years, the IEEE began work on IEEE 802.11bd, a successor to the 802.11p standard, which was approved in October 2023[19]. As a new standard, it is expected to take several years before it completely replaces 11p, and most automakers will continue to release cars with 802.11p for the foreseeable future.

This new version promises twice the throughput and longer range, while ensuring interoperability, coexistence, backward compatibility and fairness with existing OCB terminals. In addition, it operates beyond the 5.9 GHz band by utilizing the 60 GHz band.

2.5.1.2 Cellular-based technologies

Cellular systems have been used since the 1970s to transmit data over long distances via radio waves[10]. The 3rd Generation Partnership Project (3GPP) is an international consortium established in 1980 with the goal of developing technical specifications and technical reports for 3G mobile systems and is now the leading global reference for mobile standards and is responsible for the development of LTE and 5G[20].

Although most work on VANETs has been based on 802.11p, Cellular V2X (C-V2X) has gained renewed support from academia and industry as this technology has matured[21]. Rival performance to the dominant IEEE 802.11p has led to Cellular technologies being considered by some organizations as a replacement for existing IEEE standards, so its status as the second most important VANET technology is not surprising.

In September 2016, the 5G Automotive Association (5GAA) was established as a global, cross-industry organization of companies from the automotive, technology, and telecommunications industries with its main objective to promote the use of cellular 3GPP standards in ITS scenarios[22].

Most researchers dismissed this new technology because its shortcomings made it seem inadequate compared to the strengths of 802.11. C-V2X's main drawback is its centralized nature. Cellular technology relies on a centralized infrastructure, which introduces latency by forcing all data to pass through a base station.

In addition, the large coverage area of cellular antennas can also pose an issue, as the coverage area of an antenna is typically larger than the zone of relevance of safety messages. This makes broadcast an inadequate solution, as many vehicles would receive warnings not intended for them. Multicast can be introduced in an attempt to reduce this problem, but it introduces new delays from the overhead required to create these multicast groups[21].

On the other hand, Cellular technologies have several technical and economic advantages for their application in V2X communications. On the technical side, cellular technologies provide wide coverage, support high vehicle speeds, support a large number of connections to a single cell tower and can deliver high data rates. Economically, the wide use of these technologies allows the reuse of already deployed hardware for use in V2X[21].

These benefits, along with the renewed support from the industry created from 5GAA, motivated the 3GPP to begin studying the feasibility of LTE technology for supporting V2X communications[21]. In 2019, 3GPP began work on C-V2X efforts with Release 14, which made great strides in reducing existing drawbacks.

In order to work as a wireless access technology capable of serving all VANET test cases, this technology needed to ditch the necessity of using infrastructure and become ad hoc. Therefore, Release 14 established a new mode of communication, called direct communication.[23] This resulted in LTE and 5G supporting two relevant interfaces for communication in VANETs, the Uu interface and the PC5 interface.

The Uu interface is used for long range communication with cellular infrastructure in the commercial cellular spectrum. It uses existing LTE's large coverage to provide vehicles with all kinds of services[23].

The direct communication mode, also referred to as "Mode 4", utilizes the short range PC5 interface in the ITS 5.9GHz band and allows for direct exchange of information between vehicles without passing through a cell tower.

ETSI initially considered Cellular technologies as possible solutions, and at that time included 2G and 3G as possible access technologies in the initial ITS station reference architecture. As stated earlier, ETSI centralized the Access Layer of its architecture on the 802.11p-based ITS-G5. Recently, however, reflecting on the efforts of 3GPP, 5GAA, and ITS implementation in other countries like China, ETSI made amendments to existing standards in order to achieve compatibility with C-V2X technologies[23]. This indicates that in the C-V2X architecture, the layers above the Access Layer consists of the pre-established standards of different countries, demonstrating significant technological compatibility.

2.5.2 Networking & Transport Layers

The Networking and Transport Layer, as implied by its name, corresponds to the third and fourth layers of the OSI model.

Layer 3 algorithms commonly deployed in traditional networks are unsuitable for use in VANETs[5], and even existing MANETs algorithms have proven to be inadequate for use in VANETs[3]. The characteristics of VANETs, as described in section 2.1, present several unique challenges not only to existing Layer 3 algorithms, but also to Layer 4 protocols, the most important of which are high node mobility and variable node density.

The high mobility of nodes leads to frequent topology partitions, which results in highly unstable routes. Therefore, pre-determining routes in these conditions is often irrelevant and extremely complicated. Additionally, the algorithm must operate under conditions of both extreme congestion and isolation due to variable node density, further increasing its complexity.

Scholars have recognized the adoption of broadcast as the primary communication mode as a great solution to mitigate interference in high congestion scenarios. Nevertheless, this approach alone falls short of the desired outcome. For instance, in case of a crash, it is essential to rapidly disseminate security messages as the number of affected vehicles can rapidly rise. However, if every vehicle broadcasts simultaneously, it will lead to channel congestion, making it more difficult for the event to spread[5]. Therefore, novel approaches are required to curb message duplicates.

Although the challenges presented by these aspects of VANETs affect algorithm reliability, researchers have leveraged other characteristics of VANETs to develop novel solutions to the aforementioned problems. Besides the lack of power constraints, the predictable mobility of nodes allows algorithms to perform more effective link selection, which facilitates network management and opens opportunities to optimize the flow of information, ultimately improving routing protocols.

In Europe, researchers at ETSI developed new algorithms and defined new protocols in an attempt to find an adequate solution to the aforementioned challenges while taking advantage of VANETs newfound opportunities. As part of its architecture, ETSI introduced a new Layer 3 protocol called GeoNetworking[24][25][26][27][28][29], a new Layer 4 protocol called Basic Transport Protocol (BTP)[28] and a sublayer 3 extension called GeoNetworking-IPv6 (GN6)[29].

GeoNetworking is an ad hoc routing protocol that was developed solely for the transmission of safety related messages and therefore it provides the ability to forward packets without the need to exchange any signaling messages beforehand.

The GeoNetworking protocol uses the geographical coordinates of nodes as their address and utilizes location-based data to help make packet forwarding decisions.

By using a device's geographic location as its address, this protocol enables packets to be sent to all nodes within a specific geographical area. Nodes can effortlessly broadcast messages to an entire geometrically shaped area without congesting the entire network. This approach proves to be more efficient than broadcasting because it curbs congestion by minimizing transmissions to unintended destinations. Moreover, it makes the packet relevance area independent of the sender's range.

This protocol was built and optimized for multi-hop communication with geo-addressing, providing more technical features in application support than the alternative in the US. These capabilities come to a cost to protocol complexity and message overhead[30].

The GeoNetworking protocol was designed to be integrated with the BTP protocol. BTP is a transport layer protocol that is similar in function to User Datagram Protocol (UDP), functioning as a connectionless protocol[14].

ETSI has established the GeoNetworking extension GN6 as an alternative to routing packets through BTP. By using the GN6 sub-layer, Internet Protocol version 6 (IPv6) packets can be transmitted over the GeoNetworking protocol without the need to modify the IPv6 protocol[14].

2.5.3 Facilities Layer

The Facilities Layer encompasses layers 5, 6, and 7 of the OSI model, and its standardization defines application-specific functionalities[30]. The development of Facility Layer messages is the responsibility of European institutions such as ETSI, European Committee for Standardization (CEN), and International Organization for Standardization (ISO).

All the messages defined by European standard makers serve safety purposes, and therefore run over the GeoNetworking protocol[30]. Multiple different messages have been defined over the years[14], which makes it difficult to list them all. However, CAMs and DENMs are universally acknowledged as the most significant in the context of ITS.

A CAM[31] is a periodic safety message that contains vital status information about the originating vehicle, with the main goal of enabling other vehicles to take appropriate preemptive measures to avoid potentially dangerous situations[2].

This objective is accomplished by exchanging data, including speed, location, direction, and additional non-safety application data with nearby ITS stations, allowing vehicles to monitor each other's movements.[30].

CAMs begin transmitting as soon as the vehicle enters a safety-relevant context, which is considered to be anytime the vehicle is in operation.[14] The transmission rate of CAMs is subject to specific rules, including both maximum and minimum transmission times between CAMs, relevant changes in position, direction, or velocity, and congestion in the wireless channel[14].

CAM fields are shown in figure 12. It includes an obligatory ITS PDU header, Basic container, and High frequency container, along with optional Low frequency and Special vehicle containers.

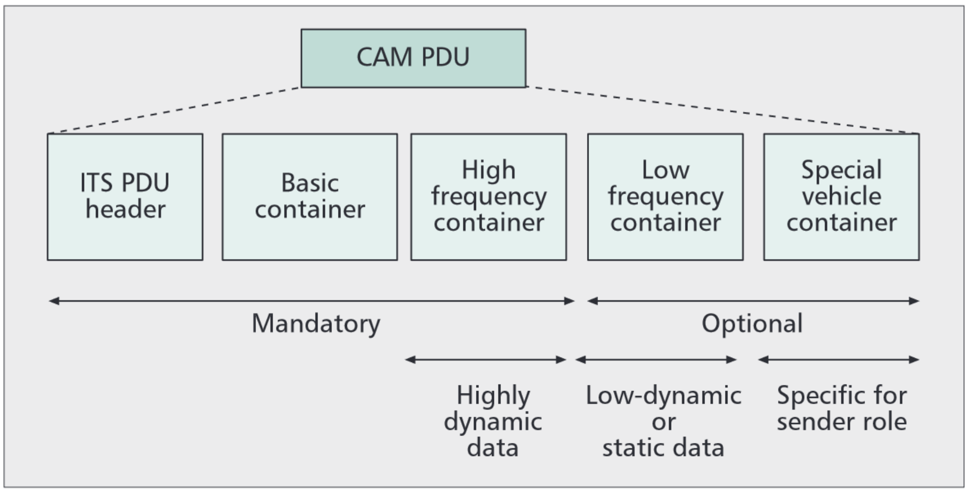

Figure 12: CAM structure [14]

This structure allows for a high degree of flexibility in message format, allowing messages to adapt more effectively to their environment, thereby minimizing congestion on wireless channels[14].

A DENM[32] is an event-driven safety message that can be triggered by an ITS station in the event that hazardous conditions are detected. ITS stations are capable of detecting a wide range of events and this message type is employed to describe such events to other ITS applications.

DENMs serve the purpose of warning ITS applications to a detected event within a designated geographical area[14]. This type of messages are only relevant in a specific location and therefore are only disseminated in that specific geographical area. These geographical areas are the ones defined by the GeoNetworking protocol.

As an event-triggered message, DENMs are considered high-priority messages, as ensuring a quick delivery is crucial to diminishing the consequences of the events that triggered their generation[2].

Throughout the lifespan of the DENM, a number of techniques are employed to ensure the dissemination of the message in its relevant area. DENMs are repeated, generally at a lower frequency than CAMs, with the intent to allow vehicles entering the relevant area to receive said information. Similarly, if the originator ceases to transmit the DENM message for any reason, another ITS station can replace the originator and continue to transmit the message. DENM messages can also be canceled by the ITS station that created them[14].

DENM fields can be observed on figure 13. The ITS PDU header and the management container are mandatory, while the situation, location and a la carte container are all optional. These message fields mainly contain information about the relevance area of the message, the type of event and the time in which the message remains relevant.[2]

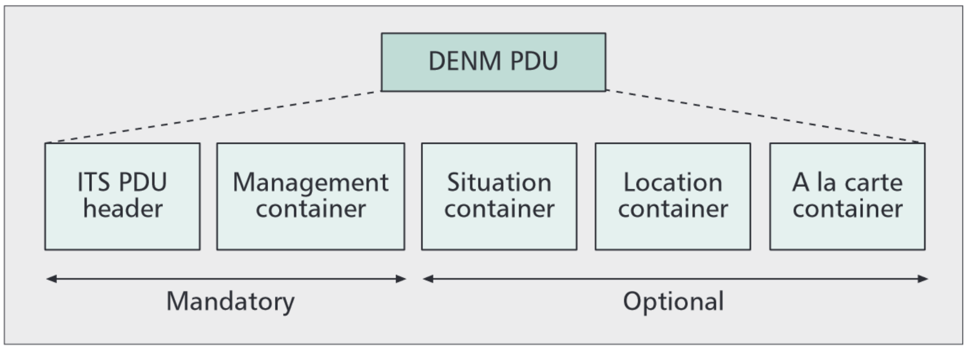

Figure 13: DENM structure [14]

2.5.4 Management Entity

Network management is a critical component of network maintenance as it ensures that a network operates as intended. The unique challenges presented by VANETs contribute to a significantly more unstable network environment than usual, making management even more essential. With this motivation in mind, ETSI incorporated a management entity into its architecture 14.

The ITS management entity is responsible for both configuring and operating its ITS station, while also overseeing cross-layer information exchange between multiple layers. It contains interfaces to every other component in the ITS station and a management information base. [26]

Decentralized congestion control is a cross-layer ITS functionality that is coordinated by the management entity and is one of its most important responsibilities. Its main purpose is to control congestion

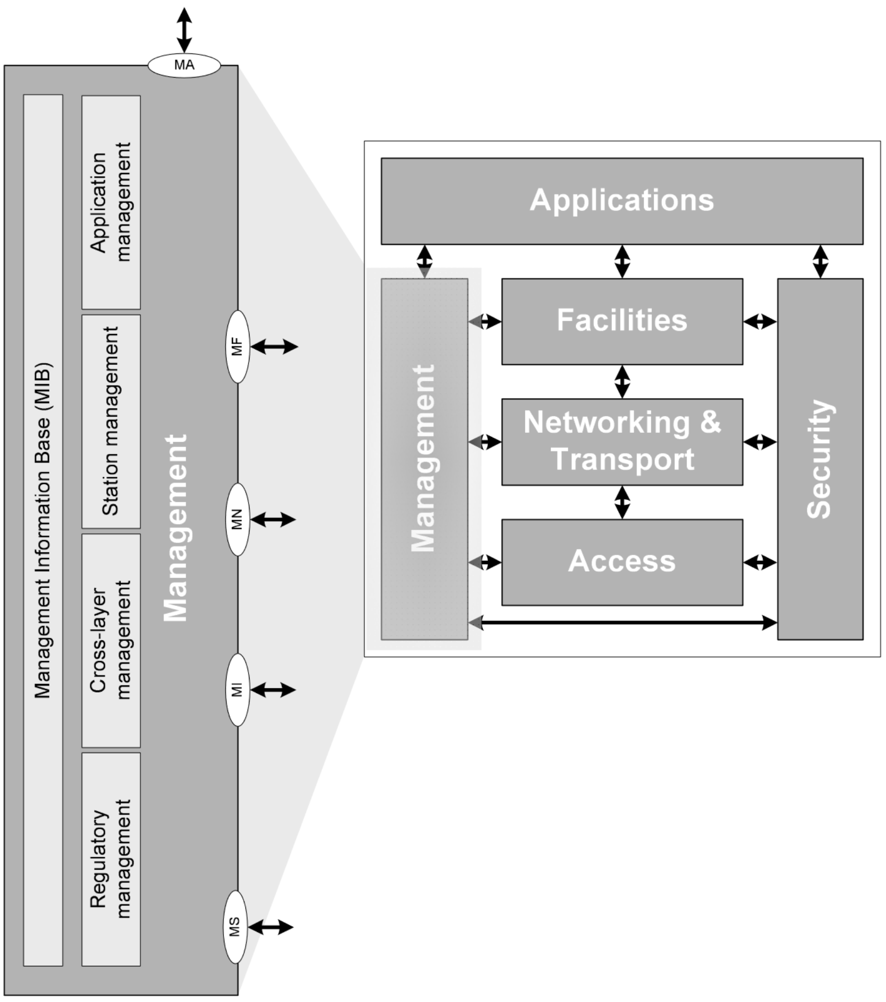

Figure 14: ITSC management entity as part of the ITS station reference architecture [8]

in the channel by managing the amount of messages exchanged, the transmission power and any other useful parameter. The changes to these parameters, being orchestrated by the management entity, are made based on various information extracted from different layers. [14]

2.5.5 Security Entity

Security in VANETs has been one of the biggest concerns and one of the biggest obstacles to its deployment. It is crucial to protect the ITS domain from malicious attacks and network abuse, and as such it has been a high priority for researchers and developers to address security threats prior to deployment. Vehicle messages can contain a large amount of sensitive information, such as vehicle trajectory and location data. This information can be used to deduce the driver's identity, activities, habits, and so on. This type of information must not only be kept private under EU law, but could also be used by bad actors for extortion.[3] [33] Attackers could also exploit safety messages for their own benefit or to the detriment of others. For instance, greedy drivers could broadcast false traffic alerts to reduce traffic on their own routes. In a more extreme case, robbers or terrorists could abuse traffic alerts to clog roads and thereby delay emergency vehicles. [33] With this motivation in mind, researchers have laid out several security requirements VANETs must meet. The following list contains the most important ones, based on the works of [34] [33]:

1. Authentication: this security requirement ensures that a recipient can identify the sender of all messages received, thereby ensuring that each message is generated by an authenticated user.

2. Non-repudiation: sometimes called auditability, it is a tricky to enforce but essential security requirement. Non-repudiation ensures that once a message is sent, the sender can't deny ownership of the message. In VANETs, this requirement is essential for identifying compromised users.

3. Integrity: this security requirement ensures that the message remains unchanged during transmission.

4. Confidentiality: a security requirement for any type of network, and VANET is no different. Some messages contain important information and therefore should only be accessible to the sender and receiver to prevent eavesdropping.

5. Availability: one of the most important concerns in VANETs, this requirement seeks to ensure the availability of the wireless channel, as a message that is delayed by seconds becomes useless.

6. Access control: this property creates different levels of access for different entities. Access control bars users from accessing any information or sending message types they are not allowed to. Distinguishing multiple levels of access allows for a higher degree of network control and is essential to stop known bad actors from exploiting network dynamics.

7. Privacy: Privacy is the ability of users to hide their personal information from the rest of network users. Implementing mechanisms to protect the privacy of all drivers is a top priority to ensure driver privacy, with the main goal to provide location privacy. Anonymity is the process of hiding one's identity, which is extremely important to achieve privacy.

8. Data verification: this property is essential to avoid false messaging as it allows all messages to be tested by their time relevance. This is necessary to avoid replay attacks in the network.

9. Physical Security: as vehicles are a widely spread technology that will be distributed indiscriminately, it is important to implement hardware security in order to avoid tampering and compromising of the vehicle.

Authentication and privacy are desirable properties of VANETs but ensuring anonymity while enforcing network liability in VANETs is a contradictory property in securing vehicular networks. This comes from the necessity to protect drivers' privacy while being able to track down attackers. [13] [33] Another important consideration to take is that security measures can create performance problems, as they add overhead time in communications and can significantly degrade message quality. [5] [35]

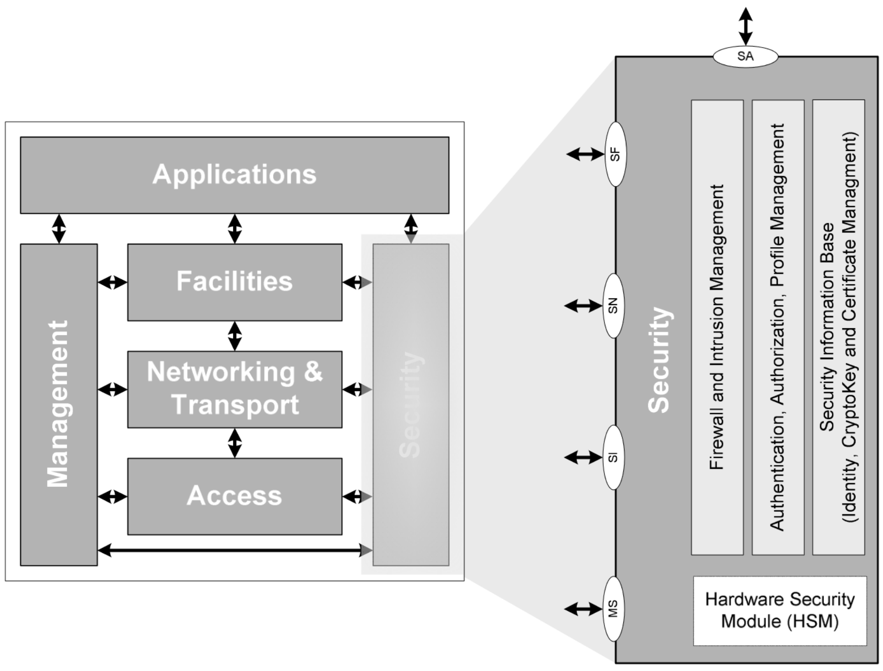
Applications
Management
Facilities
Networking & Transport
Access
Security
SA
SF
SN
SI
MS
Security
Firewall and Intrusion Management
Authentication, Authorization, Profile Management
Security Information Base (Identity, CryptoKey and Certificate Managment)
Hardware Security Module (HSM)

Figure 15: ITSC security entity as part of the ITS station reference architecture [8]

The Security Entity provides a plethora of services to ensure security and privacy. These include a multitude of secure messages at different layers of the communication stack, management of identities and security credentials, and other aspects relevant to secure platforms such as firewalls, security gateways, and tamper-proof hardware. [26]

ETSI developed an ITS communication system that relies on indirect trust relationships, which are built upon trusted third party certificates. It is a solution to the privacy problem and results in the implementation of a so-called authority hierarchy. This hierarchy is composed of manufacturers, enrolment authorities, authorizations authorities and the ITS station.

When a car wishes to begin communicating, it must use its canonical credentials given by the manufacturer to request valid enrolment credentials to an enrolment authority. Then, using the enrolment credentials it must request an authorization authority for authorization credentials. These last ones are the ones that will be used for communications, and as such in order to protect a driver's privacy are only valid in a time frame. After these expire, the ITS host must request new authorization credentials to the authorization authority. From this example, we can observe that the enrollment authority validates that the vehicle has a valid ITS station and that the authorization authority uses the authorizations given by the enrollment authority to allow the ITS station access to the ITS domain and to utilize a specific type of applications, service or privilege. This process can be observed on Figure 16. [37]

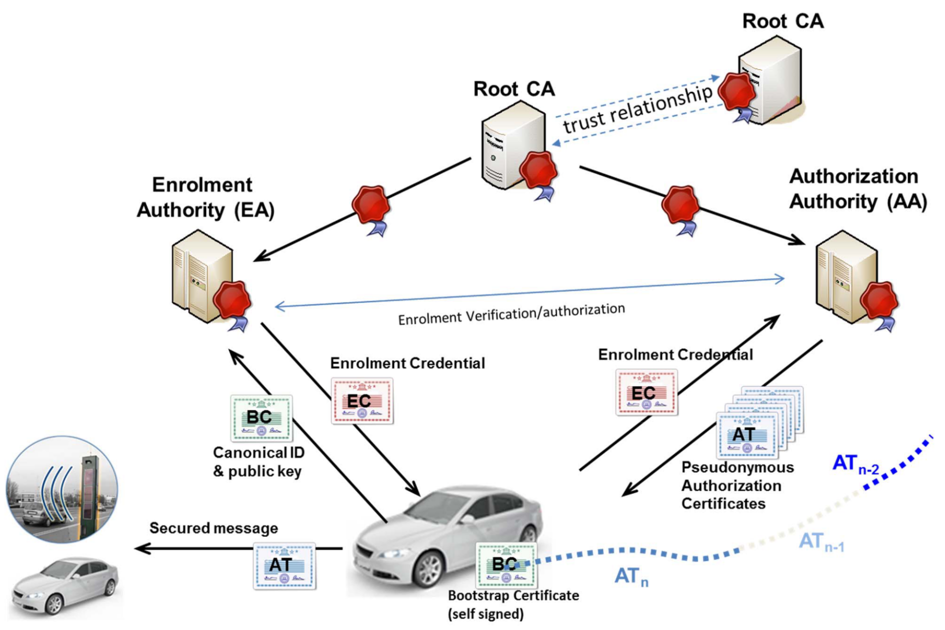
Root CA
Root CA
trust relationship
Enrolment Authority (EA)
Authorization Authority (AA)
Enrolment Verification/authorization
Enrolment Credential
EC
Enrolment Credential
EC
BC
Canonical ID & public key
AT
Pseudonymous Authorization Certificates
AT_{n-2}
AT_{n-1}
AT_n
Secured message
AT
BC
Bootstrap Certificate (self signed)

Figure 16: ITS Security Certificate Management System [36]

2.6 Applications of VANET

The motivation behind VANET research and development is to provide wireless services to both drivers and vehicles. The goal of these service applications is to improve driving safety, passenger comfort, and traffic efficiency. Hence, VANET applications can be classified into safety and non-safety applications according to their intended goal. Safety applications are considered the main driving force behind the development of VANETs[3] as the main goal of these types of applications is to decrease both road fatalities and road pollution[2]. Safety applications can be divided into two categories: Cooperative road safety and Cooperative traffic efficiency. Certain applications have the potential to improve both of them. Cooperative road safety applications harness the wireless communication capabilities of vehicles to provide useful information to the vehicle and driver, with the ultimate goal of mitigating the likelihood and severity of accidents. Therefore, any application intended to enhance the safety of individuals is classified as an Cooperative road safety application. Examples of such applications are notifications for upcoming hazardous locations and approaching emergency vehicles. Cooperative traffic efficiency applications aim to make road operations more efficient. These efforts can both greatly reduce unnecessary carbon emissions and save the time of drivers. Examples of such applications are off street parking information and green light optimal speed advisory. VANETs could also become a crucial step in the journey to fully autonomous vehicles, by exploiting expected advanced cooperation systems to exchange sensor information and status information among vehicles. These future applications also fall under the cooperative traffic efficiency umbrella. Some of these envisioned safety VANET applications require a certain percentage of road-wide deployment to become useful[1]. Therefore, ETSI has defined a basic set of applications to be deployed as ITS systems mature, with the aim of deploying them simultaneously at that time. These applications are commonly referred to as Day 1 applications and aim to provide societal and economic benefits to both the private and public road transportation sectors. Remaining safety applications have been grouped into Day 1.5 deployments, with the goal of improving and extending Day 1 applications. These applications can be further divided into bundles, which can be viewed onI. The remaining applications fall under the category of non-safety applications. Such applications, also known as co-operative local services and global internet services, are those designed to enhance the comfort of drivers and passengers by enabling access to internet services from their vehicles[2]. Based on this, any application that provides value-added services is considered non-safety applications[5]. All types of entertainment and information-based applications, such as music, movies, podcasts, online games, or instant messaging platforms, are examples of the applications covered under this category. Essentially, any application that is hosted on the World Wide Web and is accessible via the IP stack falls into this category. Non-safety applications should not interfere with safety applications, and thus they use different physical media and protocols[1]. In detail, non-security applications are typically delivered using IPv6 over C-V2X, while safety applications exclusively use GeoNetworking.

2.7 Future trends and challenges in VANET research

Predicting a date for when the full potential of the VANET technology will be unleashed is incredibly difficult but it is a case of when, not if, as the benefits of VANETs remain attractive and a lot of work has already been done.

Current and future research seeks to update current technologies in use, such as the aforementioned development of 802.11bd 2.5.1. Additionally, efforts are being made to augment this technology with other technologies such as SDN, Edge Computing, and AI to take it to the next level[38].

The deployment of VANET infrastructure in Europe is currently underway, with projects such as TEN-T. In line with the European Green Deal and the renewed focus on the climate crisis, VANETs are expected to help reduce traffic emissions by increasing traffic efficiency. Besides, the goal to reduce traffic fatalities to 0 by 2050 remains a top priority[39]. Figure 17 provides a comprehensive representation of the plans for VANETs in the EU.

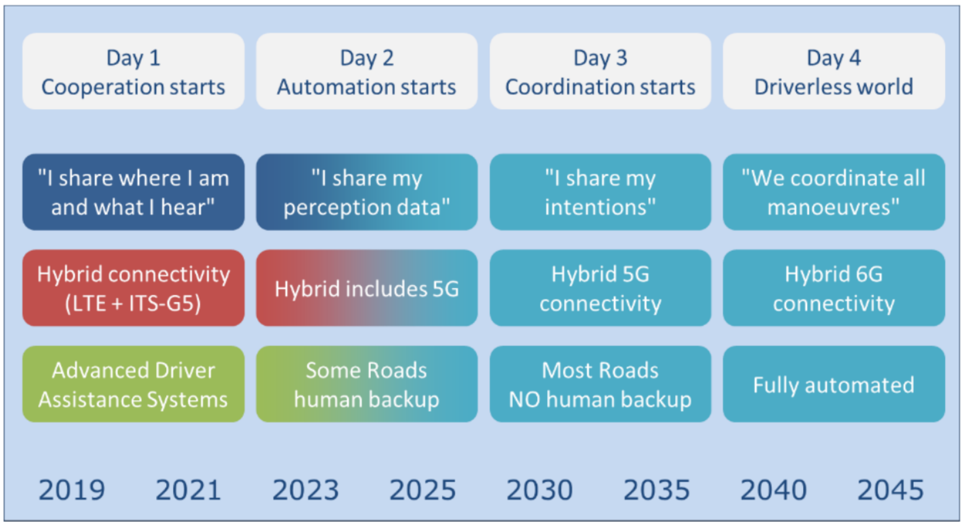

Figure 17: ITS strategy for the European Union [39]

3

Software defined networking

SDN is a networking paradigm that simplifies network management and promotes innovation through software-based approaches, rather than traditional hardware-based configurations. SDN has the potential to shape the future of the Internet and is therefore often referred to as a "radical new idea in networking" [40].

SDN dates back to the 2000s, and its concepts can be traced as far back as the 1990s [41], so this technology is not exactly new. The core ideas and motivations behind SDN have been worked on and evolved over the past thirty years, with several previous efforts already being undertaken to promote network programmability[41]. Even though it is not the first, SDN is considered the most promising as its uniqueness stands from providing programmability while offering to greatly simplify network devices[42].

In 2011, the Open Networking Foundation (ONF) [43] consortium was founded with support from major players in the technology industry, including Google, Facebook, and Microsoft. ONF develops open standards for SDN technologies to promote the SDN vision. It is currently the leading consortium in this field.

The information in this paper on SDN is largely supported by the book "Software-Defined Networks: A Systems Approach"[44], authored by leading researchers in the field and cited by the ONF itself. Before delving deeper into this topic, it is worth emphasizing that SDN is an approach to networking, rather than a point solution, so while some of the most important and widely adopted technologies will be mentioned in this chapter, they should be viewed as an attempt to achieve the SDN vision.

3.1 SDN definition

Traditional networks have a highly decentralized structure, which was essential for the early Internet to achieve high network resiliency. As a consequence, Internet devices are vertically integrated, meaning the network control logic is directly coupled to the underlying forwarding hardware. This leads to routing decisions being made by complex programs running on each device through the definition of low-level policies[45].

For a long time, researchers and developers have acknowledged that changes in the way the Internet works, nominally in its structure, were necessary to simplify and automate network management. These changes came to take place as the technology known as SDN, which breaks down the vertical integration of devices[46].

3.1.1 Main principles of SDN

SDN technology nowadays is rather an umbrella term, as many different researchers have their own nuanced definition arising from different levels of implementation of the same SDN vision. Such discrepancies in the SDN definition is acknowledge by [44] and can be visualized in [41], [46], [40], and [45]. The early SDN proposal stood on three main principles, which are still relevant to this day and are as such:

- First and foremost this technology is characterized by the decoupling of the control plane from the data plane.

  The control plane can be considered as the "brain" of the network. It is responsible for commanding forwarding operations in the whole network, making decisions on how to handle network traffic.

  The data plane is the remaining forwarding hardware that follows the commands set by the control plane. By removing the decision making components from network devices, it makes them simple forwarding devices.

  Network intelligence is thus removed from individual devices, transforming all networking equipment into simple packet-forwarding switches.

- The second principle of SDN requires a logically centralized control plane that manages the data plane through well-defined APIs. This means that a single entity, commonly known as the controller, oversees all of the network.

- Lastly, the controller must have the power to control network behavior through the programming of network functions. SDN must allow software developers to harness and utilize network resources similarly to computing resources and storage.

To sum up, and as can be seen in Figure 18, SDN requires the decision making elements of a network are decoupled from forwarding devices and logically centralized in a single entity, which is a programmable controller.

As a consequence from these principles, an SDN switch can function as any traditional Internet device, such as a router, switch, firewall, network address translator, or even a combination of these.

3.1.2 Recent advancements

These three principles remain relevant today and form the de facto definition of SDN. However, in recent years, new efforts have been made beyond this initial definition to further fulfill SDN's original goal. These efforts aim to increase programmability to the data plane, allowing network operators to fully customize all traffic in the network.

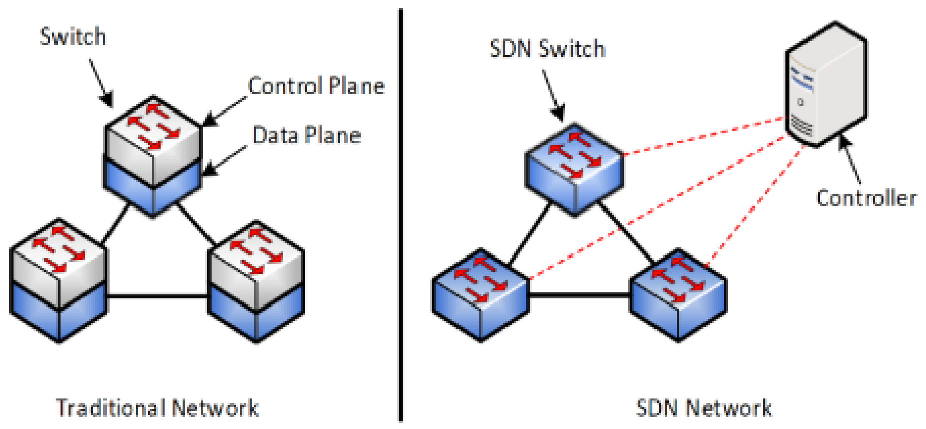

Figure 18: Traditional network view Vs SDN Network view[47]

Some articles refer to these recent efforts as next generation SDN [48][49], while predicting further and further coupling of SDN with network virtualization.

Being one of the first use cases found for SDN, Network virtualization traces its roots to the 90's, around the same time the idea of programmable networks was surging[50] [41].

Network virtualization enables the creation of network topologies decoupled from the physical topology. This allows multiple virtual networks to share the underlying physical infrastructure, functioning seamlessly over the same physical network[46]. The advantage of network virtualization is that it allows each individual virtual network to be simpler than the real network topology, granting increased flexibility.

Network virtualization and SDN are two distinct technologies that do not require each other to exist. However, SDN technology enables network virtualization, effectively creating a mutually beneficial relationship between the two that has grown much closer over the years[41].

Another way to interpret these recent SDN developments is to divide efforts into two phases. In the first phase the control plane is centralized and opened for operators, for more network control, and in the second phase the data plane is opened, allowing network operators to better control packet processing in the network.

In reality, SDN implementations may mean more or less than the original definition depending on the involved stakeholder that is deploying the technology, as each network operator has the freedom to fulfill this view however they wish[44].

From this, and from the fact that SDN is a disruptive technology, hybrid solutions that aim to harness some of the SDN benefits while maintaining the status quo in the industry have appeared[44]. These solutions mainly appear as attempts from manufacturers to maintain their established business models.

Lastly, it should also be noted phase 1 SDN was largely impelled forward by Openflow, while phase 2 is marked by the development of Programming Protocol-independent Packet Processors (P4). Both these technologies will be further discussed in Section 3.3.

3.2 Motivation behind SDN

The Internet currently faces numerous challenges, ranging from expensive, complex network structures that are difficult to manage and optimize, to significant barriers to the development and deployment of new network design ideas. The networking research community and industry have recognized that these challenges require a novel approach to network structure. SDN promises to solve these problems by providing programmability, flexibility, improved performance, and support for implementing and testing new network ideas.

This section explores the benefits of SDN to demonstrate its usefulness and explain its motivations. As this thesis will not delve into the SDN disadvantages, it is important to keep in mind that SDN is not a universal solution for all networking issues. For example, in high performance networks, where each millisecond in packet processing is precious, the benefits of SDN might not be worth its small drawbacks, as a vertically integrated device is theoretically faster[40].

3.2.1 Network innovation

New ideas in networking have had increasingly less impact in the real world. This phenomenon is commonly referred to as "Internet ossification" and it means that the Internet has become static, like a fossil[40]. Most researchers consider this to be the greatest challenge to networking in the last three decades.

Today's networks are dominated by proprietary equipment. The distributed nature of the Internet already introduces a great deal of complexity, but these components must also support an extensive set of features while maintaining network flexibility, dynamicity, performance, and efficiency[41]. This results in extremely complex network systems that require a tremendous amount of effort from network engineers while being extremely dispendious.

This status quo causes new networking features to require a long and arduous process with immense implementation, experimentation, and deployment problems. Current networking devices and their maintenance also doesn't come cheap, with long monetary returns for manufacturers in investment cycles. These facts hamper innovation by inducing network developers to delay the introduction of new network features until they are widely requested[45][50].

Another important factor is the reluctance to test new ideas for fear of disrupting existing traffic. The internet has a massive deployment base and has long been considered as critical infrastructure[40], so network ideas are not tested for fear of disrupting current services. Consequently, experimentation scenarios are rarely possible and conducted in simplistic scenarios, which diminishes confidence in their results[42]. Realistic testing scenarios are necessary to gain the confidence needed for widespread acceptance[51].

The perfect example of internet ossification can be seen in the transition from IPv4 to IPv6. This transition started more than two decades ago and still remains incomplete, while being a simple protocol update[50].

Network architecture should encourage network innovation, rather than stand in its way. Use cases for the internet in the form of internet applications are continuing to transform and evolve and it is impossible to predict future application requirements and perfectly design future networks to meet those requirements. Instead, networks should evolve and adapt in the face of surging problems. [42]

The primary motivation behind programmable networks has been to enable innovation and minimize the barrier of adoption to new network services[41], making this an integral goal of SDN. As a methodology, SDN provides a great platform for development with new ideas, techniques, and designs, encouraging experimentation.

The separation of the control plane from the data plane abstracts the definition of network policies from the underlying implementation in the switch hardware. This abstraction separates network problems into smaller traceable pieces, which enables innovation of individual components[50]. As an example, technological developments in the control plane can be made independently of the data plane, enabling faster technological development in both planes[46]. SDN is also based on open standards for communication between planes, which further stands to enable greater innovation through interoperability.

Through network programmability, SDN enables the development of new ideas using these hardware abstractions, which allows the programmer to avoid dealing with low-level hardware details. This enhances convenience and flexibility, which in turn makes network application programming more accessible to a wider audience.

Lastly, the programmable controller also benefits from a network-wide view and control, meaning it can instantly and seamlessly deploy new network applications, such as protocols or policies, across the entire network with minimal network downtime. This ability to easily update devices enables devices to be incrementally tested and refined as feedback is received. In traditional networks, these processes required the installation of expensive hardware and manual updates to individual devices. [42]

3.2.2 Network cost

Routing devices are extremely costly to purchase and maintain, especially given a handful of companies have tight control over this market. The complexity of the devices prevents new competitors from entering, which is the main reason for this dominance. SDN promotes a more affordable alternative to these traditional devices through its open source approach, which allows any hardware device to be used as a networking device, commoditizing networking hardware. [40]

In an attempt to address the lack of functionality in current network devices, traditional networks are filled with a myriad of specialized components and middleboxes performing a variety of different functions[41][50], further driving up network costs. Examples of these middleboxes include firewalls, load balancers, network access controllers, and network address translators [40]. In SDN, these devices are replaced by software programs which are implemented by the network controller as network-wide policies, thus reducing network costs.

SDN can also reduce costs by optimizing energy consumption. In large-scale network scenarios, such as data centers, the cost of energy consumption from networking equipment may seem small, but it still constitutes a significant expense. SDN allows networks to dynamically power on and off devices based on expected network traffic, reducing energy consumption and costs[40].

3.2.3 Network control

Today's computer networks are often too large, complex, and heterogeneous for conventional methods of configuration, optimization, and troubleshooting to be effective[42][50]. Network management becomes very difficult due to this complexity, which in turn hinders efforts to optimize network performance.

Network operators are required to configure their networks for a wide range of applications through manual translation of desired high-level policies into low-level commands, while coping with changing network conditions. This task is further complicated by vendor-specific configuration interfaces, which are often complex, limited, and can vary from model to model, even within the same vendor[41].

Beyond being extremely tedious, manual network configuration is extremely error-prone. Configuration changes can therefore cause network instability, leading to outages, security flaws and performance disruptions[40]. Additionally, configuration errors require a significant amount of effort to troubleshoot.

Efforts to improve network performance are also severely limited, with current optimization efforts typically confined to small regions of the network or specific network services, efforts that are further constrained by the lack of global network information[42].

Another major motivation for SDN is to shift network control from vendors to operators, and ultimately from operators to users[44]. In today's Internet, the aforementioned complexity of networks impedes this goal so SDN is designed to make networks more flexible, open, and simple to configure.

Networks are made simpler first by breaking down networking into smaller pieces through the abstraction gained from the decoupled control and data planes. This allows network managers to implement the desired network outcome in the control plane without having to deal with complex lower-level instructions.

Secondly, the complete and centralized view of the entire network helps simplify network management[45], given that current network policies are extremely complex due to the need to cope with the distributed nature of the Internet. SDN enables a more consistent application of network policies throughout the entire network, rather than on individual devices, making policies more uniform and easier to implement. [40]

Finally, SDN networks are also designed to be highly programmable, granting unprecedented control that can leverage the aforementioned benefits to enable simple, flexible and agile management of network resources. This in turn simplifies the design, deployment, and management of complex network environments and allows for more effective use of network monitoring through the ability for the entire network to homogeneously adapt to any necessary reconfiguration.

These evolutions in network structure enable new powerful solutions to classical networking problems such as end-to-end congestion control, energy efficient operations, load balanced packet routing, quality of service support, and data traffic scheduling [42]. In contrast to traditional networks, where these would be implemented by deploying specialized middleboxes at precise locations in the network, SDN allows them to be developed in software as network applications. This capability is what makes physical middleboxes obsolete, as noted above, and contributes to simplifying the deployment of new network services.

Network applications can be tailored for their purpose, ensuring that hardware resources are not wasted on irrelevant functions[45]. They can be easily shared and deployed as third-party "apps" on any network[40][50], making more efficient implementations readily available and thus improving network performance.

Implementing automated network management is another major SDN use case[42][52]. Intelligent network control can reduce reliance on network administrators by enabling network applications to quickly and easily reconfigure the network to meet changing requirements and make real-time adjustments to automatically optimize network performance. SDN enables network performance and traffic flow monitoring, enabling automatic and dynamic network configuration with centralized validation that automatically optimizes network performance based on desired high-level network policies.

SDN enables networks to be more application-aware by allowing applications to communicate with the controller and request network services[46]. An example of this is transport with quality of service insurances.

3.3 SDN architecture

The SDN architecture divides networks into abstracted layers, inspired by computer systems, each with its own unique purpose[50]. It consists of three main layers: the data plane, the control plane, and the application plane. These layers are linked by the Southbound API, and the Northbound API. Variations of this SDN view are numerous and will be explained when relevant.

A fundamental design principle of SDN, which is a cornerstone of the architecture, is to emphasize standard, open interfaces between the different planes. The purpose of this structure is to promote compatibility and interoperability between different data, control and application plane projects so that all components can work together interchangeably[50]. A result of this open approach to development is that SDN relies on the aforementioned open APIs and a collection of software components that support those APIs, rather than any specific protocol stack[44].

When surveying the SDN architecture, it is best to approach it and each of its components from the bottom up. This section will therefore cover all possible layers and components in SDN, starting from the data plane and ending on the application plane. Specific technologies will also be mentioned as they represent the current state of the art. However, it is important to keep in mind that there exist several alternatives, and these technologies are constantly evolving, with different implementation approaches that may become more prevalent in the future.

3.3.1 Data plane

The data plane is the first and the lowest layer of the SDN architecture and is made up of networking infrastructure which in traditional networks are switches, routers and any other middlebox appliance.

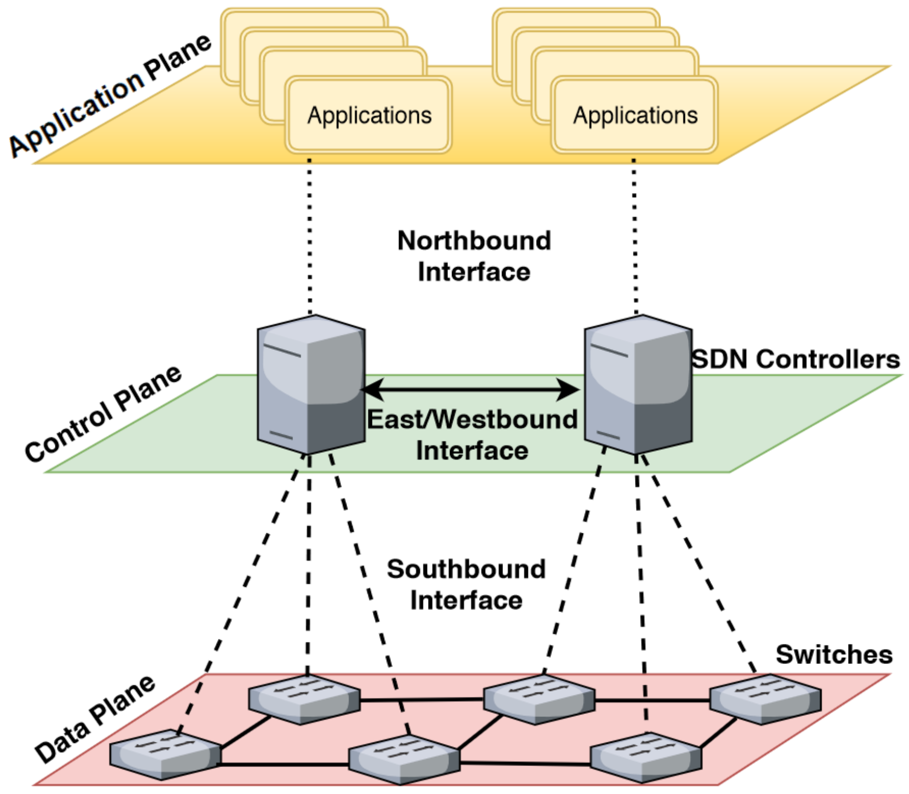

Figure 19: Layered view of SDN Architecture[53]

These distributed forwarding devices are interconnected with each other through traditional means like wireless radio channels or wired cables. Unlike in traditional networks, however, these devices don't have the autonomy to make networking decisions and instead follow the instructions given by the controller[50].

This last property means that in SDN, all network equipment is referred to as "switches" or "bare metal switches"[44] because without embedded intelligence in them, forwarding hardware becomes interchangeable and different names for equipment become redundant. On top of the hardware, SDN switches also have software running on every individual device. This is distinct from the control plane software and its purpose is to abstract the switch hardware, communicate with the control plane and implement forwarding decisions.

The data plane is responsible for receiving, processing and forwarding packets based on rules set by the controller, implementing the packet processing policies defined by the control plane[51] [42]. These devices may also gather and store network status information, including network topology, traffic statistics, and network utilization, to later transmit to upper layers[42]. This data can then be used to verify network behavior, monitor network performance and handle failures, all of which improve overall network health[45].

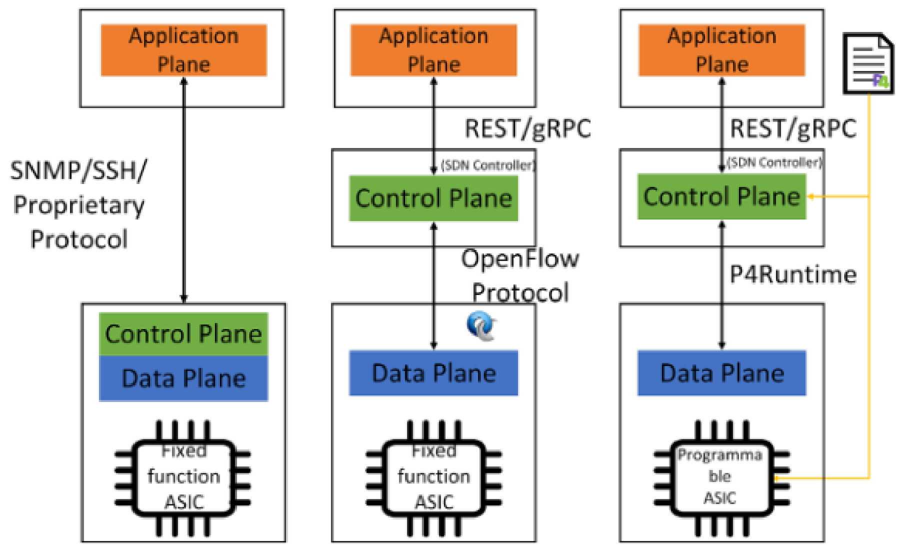

Figure 20: Evolution of SDN[48]

3.3.1.1 Data plane programmability

SDN switches can be divided into two categories based on whether they are programmable or static[44]. This classification is driven by changes not only in behavior, but also in hardware. Static devices use fixed-function chips that process headers based on fixed protocols that cannot be changed. Programmable devices, on the other hand, use programmable chips for dynamic processing.

The programming of data plane devices is a relatively recent development, which has led to the two phases of SDN mentioned in section 1. Figure 20 depicts these two phases, accompanied by a comparison to traditional networks, for a better understanding of this transition.

These efforts ultimately allow for the behavior of the data plane to be modified. Data plane programmability refers to the ability of a network operator to systematically and extensively reconfigure the processing logic of a switch as desired[45][54]. Simply put, data plane programming enables users to implement their own data plane algorithms on any switch.

Most importantly, data plane programming can unlock complete control over network packet processing. This allows customization of network protocols, meaning they can be added, modified, or removed by the programmer. In traditional networks, this capability can only be achieved by a manufacturer[54].

The main benefit of data plane programmability is therefore its increased flexibility[45]. This feature allows for the optimization of packet processing functions to meet specific use cases or requirements, tailoring the data plane to the network's specific requirements. Additionally, it facilitates the deployment of brand new network protocols.

The concept of data plane programming emerged in response to an issue found with then standard SDN solutions. SDN's ambitious goals naturally lead to initial approaches falling short. The first phase of SDN is defined by the Southbound API OpenFlow. The soundbound API is an interoperable interface that is used by the control plane to communicate routing decisions to devices, dictating how the control plane

Version	Date	Header Fields
OF 1.0	Dec 2009	12 fields (Ethernet, TCP/IPv4)
OF 1.1	Feb 2011	15 fields (MPLS, inter-table metadata)
OF 1.2	Dec 2011	36 fields (ARP, ICMP, IPv6, etc.)
OF 1.3	Jun 2012	40 fields
OF 1.4	Oct 2013	41 fields
OF 1.5	Dec 2014	44 fields
OF 1.5	Sep 2016	Restricted to ONFs members

Table 1: Fields recognized by the OpenFlow standard[55]

communicates with switches[52]. There are several other Southbound APIs available, with OpenFlow being the most popular in this first phase. OpenFlow is a protocol-dependent API that defines a specific set of headers for controllers to use in defining rules in the data plane.

As OpenFlow evolved, researchers realized that its static nature was extremely limiting[55][54][52]. Protocol-dependent Southbound APIs assume a specific data plane functionality that cannot be altered. When working with OpenFlow, network policies must adhere to the existing matching fields and actions of OpenFlow, which undermines the controller's power over the protocols used on the network.

To work around its static nature, and as a necessity to support many different protocols, the number of actions and matching fields increased with each version of OpenFlow, which has a price in complexity. The first version of OpenFlow standardized 12 matching fields and 10 actions. As new versions were released, this number skyrocketed to 44 matching fields and 19 actions in version 1.5 [52][55], as can be seen on Table 1. This remedy can be thought of as a band-aid solution, as it increases complexity without solving the underlying problem of inflexibility.

Another important issue to consider is the reactive nature of definition of these headers. OpenFlow solely defines new headers when they become widely requested. Therefore, any new useful header fields must wait for standardization in a new version of OpenFlow before they can be integrated into networks. A proactive and open approach would be more appropriate, as this approach contradicts the SDN principle of enabling innovation[52].

OpenFlow attempted to fix this problem in version 1.2 through an OpenFlow Extensible Match, which allows for adding custom matching fields[50]. However, the fundamental problem remained and a brand new approach was preferable. Many different technologies aimed to further enhance data plane programmability emerged. This thesis will focus on the most popular, P4. As final observation, the protocol-dependent nature of OpenFlow also limits the protocols that can be used to standardized ones, meaning the defined matching fields of flow tables are limited to traditional network protocols.

3.3.1.2 P4

Programming Protocol-independent Packet Processors is a high-level programming language designed to specify switch packet processing pipelines. Similarly to other data plane programming efforts, it aims to generalize the OpenFlow match-action framework and it has emerged as the most popular technology in this race, with strong support from both industry and academia[54].

The P4 project began with three main objectives[55]. The first was to enable device reconfigurability, so that the way a switch processes packets could be easily modified even after deployment. The second objective was to promote protocol independence by allowing switches to run any arbitrary protocol. Lastly, to achieve the last two goals while abstracting the underlying hardware, so that P4 code can run anywhere.

P4 achieves these goals by a new revolutionary process that, rather than keeping track of the vast number of standardized protocols used, defines an open, high-level and flexible approach for defining custom packet parsing and header matching mechanisms. This vision argues that instead of having to repeatedly extend the OpenFlow specification, hardware manufacturers should support flexible mechanisms for parsing packets and matching header fields so that controllers can leverage these capabilities through a common, open interface. [55]

P4 can actually be used with both programmable and fixed-function devices[44] and it works in two distinct phases [55][45].

During the initial phase, operators utilize the P4 language to define the parser and deparser, determine the order of match+action stages, and specify the header fields processed by each stage. For programmable devices, P4 prescribes the desired behavior in the pipeline, while in fixed-function devices, P4 merely describes the chip.

The second step occurs at runtime and consists of populating the configured pipeline with rules. This process is device-independent, with the control plane adding or removing entries in the match+action tables of the switch pipeline using the Southbound API. This phase determines the usage of defined protocols and the application of policies at a particular moment, such as adding entries to flow tables and associating actions with flows.

3.3.1.3 Hardware

When constructing an SDN network, virtually any hardware can be used for the switches, each with its own set of advantages and disadvantages. Following the approach of [42], hardware can be classified into three categories: general purpose computers, vendor specific hardware, and open networking hardware.

1. Open network hardware: Open networking hardware refers to switch specific hardware that is based on open standards and can be used with a variety of software. This provides a programmable and vendor-independent solution for building networks. The most appropriate analogy is to compare a switch to a PC built from commodity, off-the-shelf components and just like a PC, it allows for the assembly of a high-performance switch[44].

2. Vendor specific hardware: Continuing the analogy with the PC world, in that ecosystem pre-built computers are available for purchase from several vendors, such as Dell or HP. Similarly, bare-metal switch vendors like EdgeCore and Delta offer pre-built switches[44]. This hardware is specifically designed and manufactured by a particular vendor for networking purposes in SDN.

In fact, there exist networking hardware vendors that offer a variety of SDN strategies and solutions. For example, the Open Compute Project takes advantage of existing commodity switching components and offers full architectural specification for switches online [56]. Other projects like Indigo[57] aim to enable SDN features in existing traditional hardware. The use of this solution runs counter to SDN visions, representing a step sideways rather than forward.

3. General purpose computers: These are regular computers, not specifically designed for networking, that are used as a platform to implement the data plane by running a software switch that performs packet forwarding functions. Hardware virtualization is used to run virtual switches on standard operating systems running on regular hardware.

   The widespread availability of general-purpose hardware makes it inexpensive, and easy to reuse for different tasks. However, this hardware is not only not optimized for networking tasks, but also relies on computer network interface cards, which typically have a limited number of ports and slower speeds.

Both open network and vendor-specific hardware exhibit similar high performance, while using general hardware can result in decreased performance as it is not specialized for networking. General purpose hardware is usually the least expensive, followed by open network hardware, with vendor-specific devices being the most expensive. This price difference can be attributed to the scalability characteristics, ease of use, and performance of each type of solution. All and all, each approach has its benefits and drawbacks, so the best choice should be based on the specific needs and constraints of the target network.

Figure 21 presents a high-level schematic of a bare-metal switch. This abstraction better represents the open networking approach because Vendor-specific solutions are proprietary and therefore harder to generalize and the general hardware only has a CPU, virtualizing to operate in a similar manner to the depicted.

The diagram depicts three main components. The Network Processing Unit (NPU), represented as a combination of SRAM-based memory and an ASIC-based forwarding pipeline, is a chip optimized for parsing packet headers and making forwarding decisions. This representation ensures packets are buffered in memory while being processed in the pipeline. A general-purpose processor is connected to and controls the NPU. This chip runs the software of the switch and deals with the off-switch communications with the control plane. The last component is the device ports, which interconnects devices.

To achieve its goal of interoperability between different hardware, P4 programs must be portable across different devices without requiring modification. Achieving this is a problem, however, because different vendors implement different physical pipelines. To address this issue, P4 introduces the concept of an intermediate layer between the core P4 language and the targets[54]. This layer abstracts the details of the physical pipeline into a common logical pipeline that provides a programming model that easily describes how packets are processed. As a result, P4 programs can be developed for a logical architecture that can be deployed on multiple targets. Currently, there is no consensus on a standard

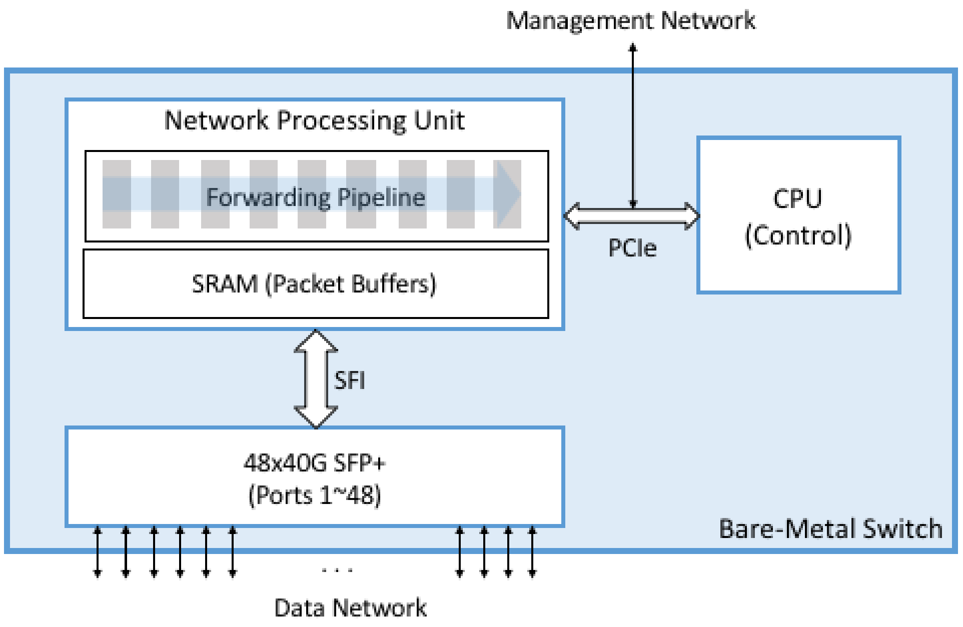

Figure 21: High-Level schematic of a bare-metal switch[44]

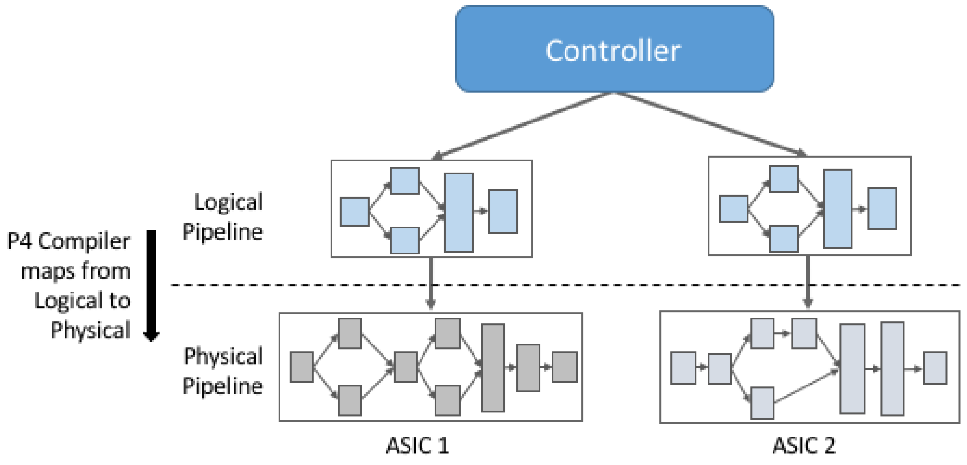

Figure 22: Definition of a logical pipeline to provide a pipeline-agnostic view the control plane[44]

logical pipeline for all devices. Nevertheless, this approach enables pipeline-agnostic controllers as can be seen on Figure 22.

The logical pipelines can be represented using different abstractions. In figure X, the pipeline is described using the data match-action pipeline. This abstraction describes the packet processing functions of a network device as a pipeline of lookup tables. Each table is responsible for matching specific fields in the packet header and performing an action based on the result[45][50]. Thus, the data match-action abstraction describes the NPU as a sequence of lookup tables.

Other abstractions exist for describing the switch pipeline, but this is the most popular by far. OpenFlow is largely credited with popularizing the match-action abstraction[45]. However, the original proposal for OpenFlow took advantage of the fact that most routers and switches already contained flow tables used to implement additional network functions such as firewalls, NAT, QoS, and statistics collection, to define a common set of functions and provided an open protocol for programming these flow tables[51].

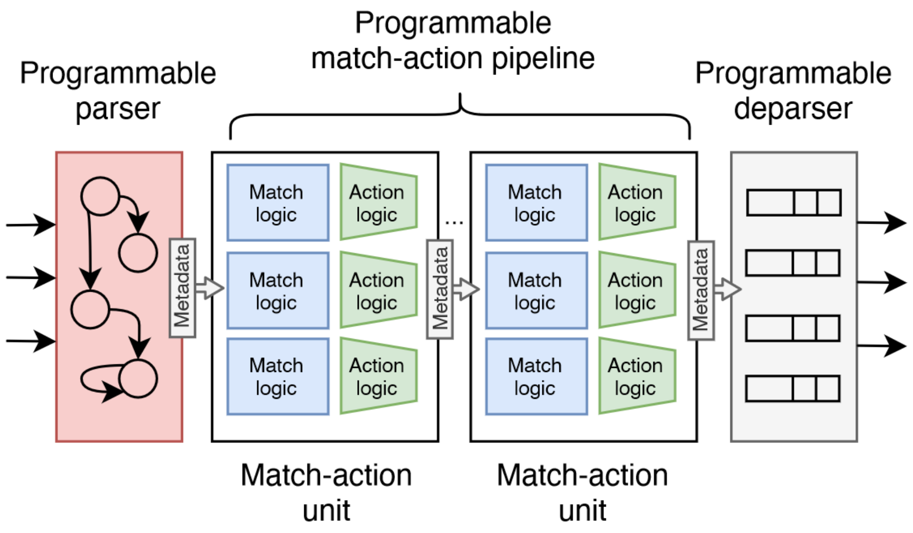

Figure 23: Protocol-Independent Switch Architecture[54]

This implies that the architecture of the NPU and the Southbound API strongly influenced each other.

This abstraction depicts multiple tables because high speed switches require a multi-stage pipeline for packet processing. It is the optimal approach for packet processing because this process involves checking multiple header fields, and with multiple stages, different stages can perform different tasks simultaneously[44].

There are various P4 architectures, including Protocol Independent Switching Architecture (PISA), Portable Switch Architecture (PSA), V1Model, SimpleSumeArchitecture, and Tofino Native Architecture (TNA)[44][54]. These architectures abstract the target hardware into a forwarding model representation.

PISA presents a simple representation of the device, so all of these examples can be considered variations of it. The other architectural models follow the same principles with some arbitrary enhancements. The PISA model will be discussed as it is the most general. However, it should be noted that the V1Model is currently the most popularly used.

Protocol Independent Switching Architecture is a data plane programming model of P4. It was introduced in P4_14 and is currently deprecated. It is applied in a variety of ways to the creation of concrete architectures[54]. As an example, Figure 23 gives a high-level overview of this pipeline.

This architecture depicts three main components. The parser is the first component and handles header fields by specifying their size, order, and other relevant details. This ensures accurate extraction and utilization of the fields in the subsequent stage. The next stage is the match action pipeline, which consists of the multiple sequential tables that define this type of data plane pipeline abstraction. It maps a desired action to a given match by utilizing previously defined headers. The third is the deparser and it does the opposite of the first. It deparses the packet metadata into the packet before it is transmitted on the output link. Beyond the packet, relevant metadata about it is also traversing the pipeline. Examples of this may be packet relevant, like input port and arrival timestamp, or device relevant switch counters, queue depth.

In practice, when a packet arrives at a switch, after being parsed, the packet headers are compared to the entries in the flow table. If a match is found, the corresponding action is taken, and the packet is processed accordingly. If no match is found, the packet is typically forwarded to the controller for further processing or dropped.

3.3.1.4 Software

Each switch must be equipped with a BIOS, which is similar to the firmware found in microprocessors as it provisions and boots a bare-metal switch. The Open Network Install Environment (ONIE) has emerged as the standard for switch BIOSes[44]. On top of it, every switch must run a local Switch OS, which is responsible for handling calls from the controller and taking appropriate actions on the switch's internal resources. The most widely used Switch OS is Open Network Linux[58], an open-source project based largely on the Debian distribution of Linux and extended to support hardware unique to switches[44]. Other open source Switch OSes include Stratum[59] and SONiC[60]. Vendor-provided software development kits for the on-board switching chips are also important. These kits are analogous to device drivers used in traditional operating systems.

3.3.2 Southbound API

The Southbound API is the second layer in SDN. As introduced in the previous section, the Southbound API is a logical interface that connects the controller with all network elements in the data plane[46]. It allows the control plane to push configurations to forwarding elements and the switches to push information to SDN applications, enabling the controller to have direct control and constant information on the state of the data plane elements. This protocol describes the capabilities of the switches, which defines the communication protocol and methods used to exchange data between the data plane and the controller. The interaction between the control plane and the data plane is therefore defined by this protocol[50].

A well-defined programming interface is crucial for the success of SDN, and its standardization offers several advantages for designing and deploying efficient SDN solutions[53] [50]. Specifically, it provides flexibility by allowing the control plane to communicate with the data plane through a variety of standardized channels. This enables network operators to choose from a range of controllers and switches, reducing vendor lock-in and expanding their options.

The responsibilities of this interface can be split between Control and Configuration[44] and usually, protocols only fulfill one of these categories. For a long time, OpenFlow was considered the state of the art for Southbound APIs and as testament to that is still the most widely used today[53]. However, the current state of the art for Southbound interfaces is a combination of P4Runtime, gNMI, and gNOI. P4Runtime and OpenFlow are control interfaces, while the gNXI combination of protocols is responsible for device configuration[44].

These different types of Southbound APIs also have the huge benefit of sheltering the controller and by extension the application running on top of it from the diversity of network devices that may exist in the network. [44]

3.3.2.1 OpenFlow

The previous section already touched on the origins of OpenFlow as a means to justify the industry's distancing from it. McKeown et al. proposed OpenFlow[51] as a means to enable experimentation on campus networks. OpenFlow leveraged the flow tables present in most Ethernet devices, which are used for additional configuration such as firewall setup and subnetting, to allow customization of campus network devices. [42]

OpenFlow has been the de-facto standard for the Southbound interface for a long time[40] and remains the most popular Southbound API. This is evidenced by the fact that most commercial switches support the OpenFlow API out-of-the-box[50].

3.3.2.2 P4Runtime

Nowadays, Openflow's development has been dropped by ONF in favor of P4, which means it has been replaced with the more open alternative of the P4Runtine. P4Runtime is a surging data plane API[54] that is closely coupled with the P4 programming language. P4 enables the data plane to be programmed according to network requirements. A static interface would eliminate the advantages of P4 since the controller would not be able to benefit from the custom programmable policies defined in the data plane.

To fully reap the benefits of P4, a dynamic communication interface between the controller and the switch is necessary[44]. To establish this interface, not only must the switch pipeline be known to the controller, but both the switch and the controller must be informed of the P4Runtime contract. P4Runtime is automatically generated along with the client and server-side stubs for the controller and devices, depending on the defined switch architecture[44].

3.3.2.3 gNXI

The term gNXI refers to the combination of gNMI and gNOI. The role of these two is somewhat blurred, but in general, gNMI handles the persistent state, whilst gNOI is responsible for clearing or setting the ephemeral state. In other words, gNMI is used for switch configuration, defining actions for retrieving or manipulating the state of a device via telemetry or configuration data, while gNOI is used for executing commands on a device, such as rebooting, pinging, and accessing other operational variables on the switch rather than directly changing its configuration[44][61].

In traditional networks, the configuration part of the Southbound api has been called the operations, administration, and maintenance interface which is usually accessed in command-line[44]. The equivalent of gNXI in traditional networks is Simple Network Management Protocol (SNMP). SDN has brought about the possibility for a different approach. SNMP was originally designed for closed devices and primarily focuses on reading rather than writing onto devices, making it more similar in capabilities to the configuration part of gNXI[44].

The gNMI has three methods. The well-known get and set, similar to their counterparts in SNMP, are used for reading and writing. The third is the subscribe method, which is used to request a stream of telemetry data[44].

OpenConfig[61] is responsible for the development of gNXI. OpenConfig is an industry-wide standardization effort to drive the industry toward a common set of configuration models. It is a collaborative effort within the networking industry to move toward a more dynamic and programmable approach to multi-vendor network configuration and management. gNXI is expected to become the standard management protocol due to strong industry support[44].

The underlying mechanism used by gNOI and by gNMI is exactly the same[44]. Actually, P4Runtime and gNXI all rely on the gRPC framework with protobuf[54]. The intricacies of protobufs and gRPC will not be detailed here, but these technologies are used for the benefit of not having to deal with the daunting list of formatting, reliability, backward compatibility, and security issues that other protocols, including OpenFlow, must deal with[44].

3.3.3 Control plane

Having outlined the structure of the data plane and its major technological developments, it is now time to move from the individual switch level to the controller network view. The control plane is widely considered the most important component of the SDN architecture[42] because it represents the centralized decision-making authority of the network, acting as the "network brain"[50].

The control plane consists of one or more controllers that oversee, manage, and manipulate the data plane through the Southbound API. All forwarding decisions are centralized in the control plane, which is dependent on a global view of the network to make accurate decisions. In this process, device-specific details are abstracted from the control plane[53].

Traditional operating systems provide abstractions of the underlying hardware that simplify the use and management of the underlying resources. As a result, it is possible today to build more complex applications faster, more securely, and more efficiently[50].

In various ways, the SDN layers bear deep resemblance to a regular computer system[46]. The application layer is comparable to the software applications that perform tasks using computing resources, the infrastructure layer is equal to the computer hardware and, most relevant to this case, the computer OS corresponds to the control plane.

For this reason, the controller is commonly referred to as the Network Operating System (NOS). This analogy introduces the same abstractions found in computer operating systems to networks[50], and is often used to better illustrate that the control plane is responsible for representing the network as a single system for network applications[40]. The terms NOS and controller can and will be used interchangeably from now on.

3.3.3.1 Controller responsibilities

A list of common applications implemented in the NOS can be collected from different SDN deployments and asserted as the base network service functions[50]. While such a list cannot be considered de facto obligatory due to the SDN root of modularity, it is still relevant to have a comprehensive list for familiarity's sake.

The controller is mainly responsible for topology management, enforcing the desired packet processing policies and performing other necessary functions to manage and regulate the network[44][53][45][50][62].

Topology management requires maintaining an up-to-date network view, which involves link discovery and host tracking. The main duty of the controller is to implement the desired packet processing policies. This task involves determining where packets should be forwarded by establishing traffic paths and defining packet flow rules. Furthermore, the controller selects the headers that require modification and installs all of these control commands on each forwarding device.

Lastly, the controller is responsible for performing other necessary functions to manage and regulate the network, such as monitoring device health by collecting status information and other important data from switches and performing maintenance procedures on those same switches.

3.3.3.2 Control centralization

Due to its prominent role, the performance of the controller largely influences overall network performance. The biggest criticism of SDN stems from the belief that centralizing the control plane will lead to scalability and performance issues. However, most practical results suggest that a single controller is capable of handling a substantial number of new flow requests, and therefore should be capable of managing most small to medium sized networks[40][62].

In medium to high network scenarios, however, the controller can quickly become overwhelmed. A single physical controller represents a performance bottleneck in the network with added security issues[62].

The architecture's logical centralized controller does not imply the existence of a single physical controller. For optimal performance, scalability, and availability, the network controller must be physically distributed[50]. A physically distributed but logically centralized control plane reduces lookup overhead and increases reliability while maintaining a centralized view of the network for applications to write in[40].

The physical distribution of controllers can be classified into flat or hierarchical[62]. In the flat architecture, all controllers maintain a global view of the network, whilst in the hierarchical architecture the network is divided into domains, where a single controller has jurisdiction over a single domain. To achieve a global view, a root controller is created to coordinate the actions of the other controllers and maintain the global view.

Another important advantage of distributed architecture is that a network controller can function similarly to other services by scaling based on workload to ensure availability and saved resources[44].

3.3.3.3 East/Westbound API

Some researchers propose the concept of an API specifically used for the coordination of these distributed scenarios[50][42][46][53][62]. This API is called the East/Westbound API and it provides the exchange of information between different instances of the controller in order help maintain a consistent network view. This API has no relevant standards. It is not typically considered a part of the overall SDN architecture, although in practice, something similar must always exist in distributed controller scenarios, as can be seen in Figure 24.

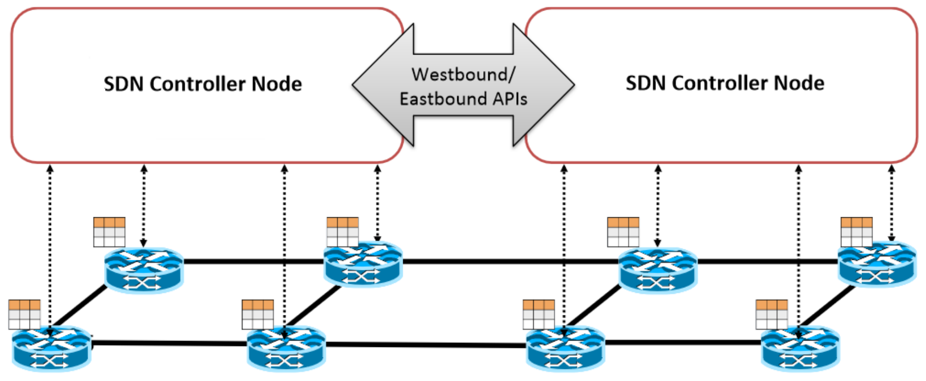

Figure 24: East/Westbound interface between distributed controllers[50]

A finer distinction can be made between the Eastbound and Westbound APIs[50]. The Eastbound API mostly to communication between SDN controllers and legacy routers, while the Westbound API refers to communication between different SDN controllers, whether they are different frameworks or the same in a distributed architecture.

3.3.3.4 Notable Controller Platforms

Most controllers are licensed as open source, with few having a proprietary license. They are programmed in a large variety of programming languages, with Java and Python being the most used, and some even contain more than one language for better performance. However, most don't receive frequent updates and maintenance, and lack proper documentation[62]. A lot of different controller platforms exist, and the most notable are Pythonic Network Operating System (POX), Floodlight, Open Day Light (ODL), Open Network Operating System (ONOS), and Ryu.

All controllers can run on any commodity hardware[50]. They are designed to be highly modular, with the controller being customized and configured to the given deployment, only having the subset of required modules[44]. This means that the behavior of most controllers is generally the same, with the only relevant differences being the deployment architecture. Another important point where controller architecture varies is the support for different noth, east, west and Southbound APIs, as not all controllers are compatible with all interfaces.

3.3.4 Northbound API

The Northbound API is a logical interface that connects the control and application plane. It is offered to the application developers by the NOS[50] and provides programming abstractions for networks, acting as a bridge between the control and application plane[53].

There is no fixed or widely accepted standard for Northbound APIs[50][53]. The reason for the lack of standardization of this interface is the huge variation of applications and user requirements, that makes the existence of a single interface for all of them very difficult.

There is therefore a large amount of different APIs that can be used depending on the chosen NOS. The purpose of this large number of different APIs is to sufficiently provide control over all underlying functionalities to the application plane[44].

As a Northbound interface implementation, programmers and most controllers typically use the REST API[53]. The gNXI interfaces are also popular, with them corresponding exactly to their counterparts in the Southbound API[44].

3.3.5 Management plane

The application layer is the upper-most layer, sitting above the controller. It consists of network applications designed to meet specific user requirements and, through the Northbound API, this layer has easy access to the capabilities of all the lower layers such as the global network view and constant status information, enabling software applications to instruct the network controller on how to implement network control and operation logic[42][48].

SDN applications can accomplish a variety of goals, from adaptive routing and boundless roaming to enhancing network security and many more[42]. In essence, this layer can request custom control policies for routing, firewalls, load balancers, monitoring, and so on. The controller interprets these policies and determines the actions to be taken by the network devices. The switches then translate the network commands into concrete actions in the forwarding tables[50].

The existence of this layer is sometimes omitted, depending on the interpretation. It can be considered an integral part of the control plane because the two are closely related. The application can be viewed as running on top of the controller as well as on top of the controller[44].

4 SDVN

The SDN paradigm has been identified by some researchers and other experts as a powerful and promising tool for addressing some of the challenges posed by VANETs[63]. As previously discussed, SDN is a methodology rather than a specific technology, which allows it to be applied to VANETs. This line of thought created a new networking paradigm that is primarily known as SDVN, although the term Software Defined Vehicular ad hoc Network (SD-VANET) is also occasionally used.

The concept of SDVN was first introduced by Ku et al. in [64], and since then there have been many different proposals of how to implement the SDN principles in VANETs. SDVN has recently received significant support from researchers, which has resulted in significant technical and architectural advances in SDN-enabled vehicular networks[65].

Similarly to traditional wired networks, non-SDN systems function reasonably well. Therefore, the implementation of SDN should not be seen as a solution to make VANETs work, but rather as a means to enhance them.

4.1 Benefits and Challenges

The implementation of the SDN paradigm in VANETs is driven by the desire to bring some of the improvements of SDN to VANETs to tackle the challenging requirements of VANETs. This paradigm has the potential to introduce flexibility, programmability, and centralized control to vehicular networks[65]. This section reflects upon the benefits of SDN and the characteristics of VANETs to extrapolate the benefits and possible challenges that SDVN can offer today.

4.1.1 Mandated Interoperability

As previously stated in Section 2.6, VANETs require high road acceptance for their useful applications to be effective. This implies that all implemented devices must be able to communicate with each other, and to this end the EU has tried to standardize devices to promote interoperability between different vehicle manufacturers.

In SDN networks, operators have complete control over a static network, making it easy to conceal the inner workings of devices from the rest of the Internet and implement them as desired. However, in SDVN the same is not possible because a vehicle ITS subsystem must be able to communicate with every single roadside ITS station in order to access infrastructure services deployed in the central ITS subsystem. SDVN devices must be compliant with existing regulation and communicate with standardized protocols in order to be interoperable with other devices.

Such an emphasis on interoperability undermines the powerful control that SDN brings. Network operators and developers can't take full advantage of the freedom this technology provides to implement new algorithms and test them in real-world scenarios. Even the effectiveness of the additional control provided by P4 is reduced by the inability to change network protocols.

This is not to say that the increased control that SDN brings is completely useless, as it can still provide some advantages in dealing with certain scenarios, but this VANET characteristic largely limits what can be accomplished with SDN.

It is also relevant to note one of the major goals of SDN, which is to move network control from manufacturers to operators, with the ultimate goal of giving it to network users. In vehicular networks, the identity of the network operator becomes an implementation detail, but it will most likely be the same government of private entities deploying the infrastructure. SDN continues to provide these operators with greater network control, but it cannot be given to users for fear of being misused for nefarious purposes.

4.1.2 Network innovation

Another important conclusion to be drawn from the above is that the technologies that will form future VANETs can be expected to be even more static than in traditional wired networks. On top of that, current hardware defined devices exacerbate this problem, as any fundamental change to VANETs would require the hardware of thousands of roadside systems and millions of vehicle systems to be manually replaced. This presents a logistical nightmare, making it highly unlikely that this scenario would happen except in the rarest of cases, and then only if mandated by the EU. This reality is evidenced by the slow and arduous standardization process that VANETs have undergone over the last 20 years, which has sought to find a single solution for VANETs that would act as a definitive and permanent answer to all of its problems.

SDVN allows new functionalities to be implemented in the same hardware through software updates, facilitating changes to be made throughout the network. This could considerably accelerate the adoption of this technology and facilitate the implementation of any future fix or optimization.

4.1.3 Network management and performance improvements

Even more than in static networks, SDN can provide VANETs with more flexible and intelligent packet flows, channel allocation, and connectivity. Traditional VANET routing algorithms are limited to the individual perception of each node as they are implemented in each individual device. SDN is built on a global view to make better decisions based on the combined information from multiple sources, which results in better network performance since informed decisions lead to more optimal outcomes. The automated network capabilities brought about by SDN also have the potential to address VANET's adversities. Listed below are some of the characteristics of a VANET and how they can be enhanced through the implementation of SDN.

High mobility In an environment where topology changes occur frequently, it is critical to be capable of adapting to sudden and unexpected events. SDN provides flexibility, which allows for more dynamic network configuration, enhancing the network's responsiveness to emergencies and changing requirements, while also enabling it to better adapt to changing conditions and needs[64].

Predictable mobility SDVN can leverage the centralized view of the network, to better exploit the predictable mobility when compared to traditional approaches. The controller can use a vehicle's information to predict future topology changes and set traffic rules accordingly, improving connectivity and overall network performance.

Congestion and scalability Network congestion and the variable vehicle density are issues that can be reduced by leveraging the global view of the network to better optimize the available mediums. Centralizing network control increases cooperation, allowing the optimal path to be calculated, which reduces delays and overhead[63]. Also, situations of high vehicle density can be more easily predicted and the transmission power levels on vehicles can be jointly adjusted based on future vehicle network density, greatly reducing interference[63]. Overall, automated tools promise to make runtime changes to networks based on the unique characteristics of VANETs, helping to achieve a better performing network.

4.1.4 Issues with maintaining a global network view

The advantages of centralized control are plentiful, but the establishment of this centralized view in VANET is not guaranteed. VANETs are based on wireless connectivity, which causes the southbound API to suffer from instability and even complete unavailability[66]. Combined with volatile topologies, this results in increased communication delays in the southbound interface, which can lead to inaccuracies in the controller's view of the network and in the rules established in the switches, casting doubt on the controller's ability to maintain an accurate and up-to-date view of the global topology[67].

Another major concern introduced with SDVN is the management overhead introduced by the southbound API required for the controller to manipulate the devices, as switches must be frequently updated to accurately reflect the network's state. This could result in additional traffic and increased network congestion compared to the distributed methods used in traditional networks.

At the same time, it should be noted that it may not be necessary for all instances of the controller to maintain a global view of the entire network. Instead, individual instances of the controller may only need to keep track of a small region of the network, as most messages in VANETs are only relevant in their immediate surroundings[68].

In summary, the dynamic state of the network combined with poor controller connectivity presents a major concern in SDVN. The process of maintaining an updated view of the global network topology becomes costly and time-consuming due to potential inaccuracies in the updated information[67]. To address this issue, it is important to ensure robust protection against loss of connectivity and to guarantee the availability of the control plane, which sometimes requires the integration of other technologies such as fog computing with SDN[67].

4.1.5 Cheaper networks

When considering the deployment of VANET technology, the cost of infrastructure is a major concern, with up to 86% of the total expected cost being attributed to hardware[15]. Just as in traditional networks, SDN can reduce costs by reducing complexity and increasing modularity while promoting interoperability.

4.1.6 Network security

The introduction of the SDN paradigm does not introduce any new significant security or privacy challenge to VANETs. Some researchers[67] mention the centralized controller as a new vulnerability that can be exploited, but in reality current VANET deployment already relies on services provided by the central ITS subsystem.

4.2 SDVN architecture

The SDVN architecture can be represented based on either of the architectures of the two technologies that define it. Of the two, the most suitable is the architecture of SDN. The VANET architecture is best represented by the ITS host architecture, which is built on the OSI model. The SDN paradigm aims to move the internet away from closed and complex standardized protocols and into open and simple network components. As such, the SDVN architecture inherits the format as the general SDN architecture, retaining the three layers present in it as can be seen in Figura 25.

4.2.1 Data plane

In SDVN, as in SDN, the data plane consists of all forwarding hardware but these components can be divided into two different categories based on whether they are wired or wireless. Mobile hardware refers to the vehicle ITS station, while stationary hardware refers to the routers and switches found in the remaining ITS subsystem. Network intelligence is extracted from all ITS stations, but due to the special characteristics of the mobile nodes all SDN enabled devices cannot be treated the same.

4.2.2 Control plane

The control plane remains almost unchanged as the centralized logical intelligence of the network, managing device behavior to achieve desired policies. It remains responsible for communicating the high-level applications defined in the application plane to the devices in the data plane, and for transmitting device

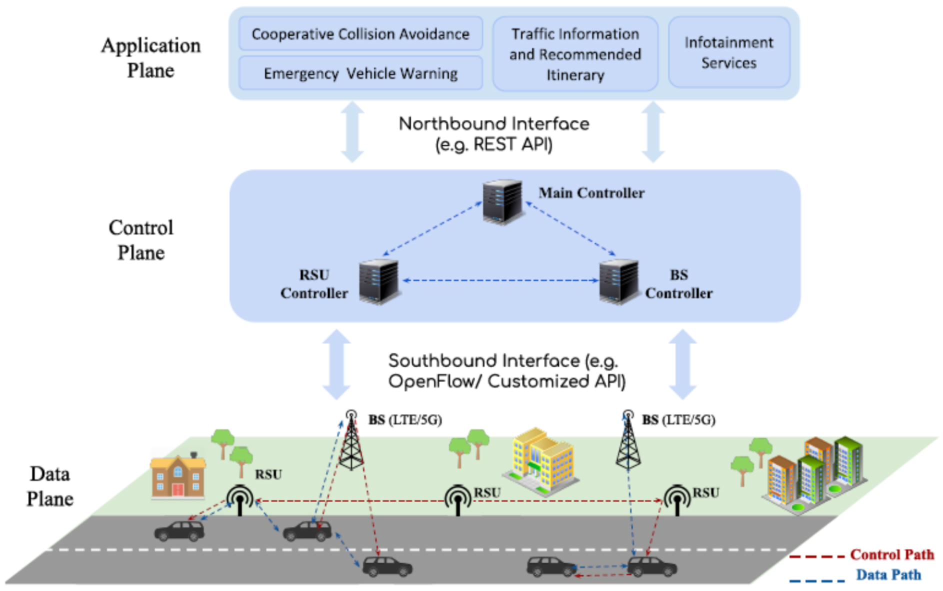
Application Plane
Cooperative Collision Avoidance
Emergency Vehicle Warning
Traffic Information and Recommended Itinerary
Infotainment Services
Northbound Interface (e.g. REST API)
Control Plane
Main Controller
RSU Controller
BS Controller
Southbound Interface (e.g. OpenFlow/ Customized API)
Data Plane
BS (LTE/5G)
BS (LTE/5G)
RSU
RSU
RSU
Control Path
Data Path

Figure 25: Conceptual view of SDVN architecture[69]

information from the data plane to the application plane. The main difference introduced by SDVN is a consequence of the need to manage devices in the ad hoc space. Due to the two distinct types of components in the data plane, unique and customized networking applications are required to address the unique nature of each scenario. Managing the wired portion of the network can be considered a simple task because the process is the same as SDN, whilst the wireless domain presents unique challenges and requires distinct solutions to manage wireless resources as effectively as wired resources[66].

In SDVN controller distribution becomes a central concern. Vehicular networks have immense coverage given that they include all roads, and due to this vast distribution, a single controller cannot handle the traffic of the entire network. Therefore, it is out of the question for scalability and delay reasons[70]. Instead, the control plane must consist of multiple physical controllers efficiently distributed across the network, coordinating their decision-making efforts to handle the high volume of traffic[67]. Another important motivation for distributing the control plane is to bring the controller closer to the vehicles to reduce communication delays between the data and control planes[71].

It should be noted that a truly distributed controller, where all instances have global information of the network, seems impossible and even counterproductive. To perform effectively, controllers usually only need information about a specific part of the network, which is in part because some messages are only relevant in the context of that particular section of the network[66] and the ones that aren't should be directed to the nearest roadside ITS station. The size and the transition between regions is dependent on implementation.

That being said, there exist many different controller placement distributions proposed in the literature. In practice, the controller can be located in the central, roadside, or vehicle subsystems. Different architectures place the controllers in a combination of these stations, giving them distinct responsibilities and utilizing different methods to distribute intelligence between them.

Most approaches to controller distribution in the literature[65][66][70] converge on a two-tier hierarchical controller. The top-level orchestrator controller, also known as the global or primary controller, is located in a server integrated into the central ITS subsystem. The lower-level controllers, referred to as local or secondary controller instances, are placed closer to vehicles in the roadside ITS subsystem in an effort to improve connectivity and reduce delays.

4.2.3 Application plane

The application plane remains unchanged. It consists of network applications designed to fulfill certain networking tasks using the capabilities given by the control plane. Some of these include monitoring, QoS, analytics, recovery, security, routing, load balancing, and management[65]. As stated in Section 2.6, VANET applications can be divided into two main categories, each with different QoS requirements. The most significant of these, the safety applications, are mostly delay sensitive, while the non-safety applications are more bandwidth intensive[63].

4.2.4 Communication interfaces

All communication APIs have no relevant changes. The only noteworthy detail is that by having access to both ITS-G5 and C-V2X for communication and control[70], a controller can has two main communication mediums through which, using the southbound API, it can communicate with SDN devices.

5 Related papers

Emulation is a convenient and cost-effective method for testing VANET solutions. However, due to the complex and unpredictable nature of real-life environments, relying solely on these tools is insufficient[**Cardona et al.**, **2020**]. In order to gain a realistic perspective on VANET, it is necessary to implement it in real-world scenarios so the validity of the expected benefits and drawbacks can be tested. The same is true for SDVN. SDVN is notable for the lack of experimental efforts conducted on actual hardware. In order to enable testing on real scenarios, open-source SDVN tools and frameworks that are compatible with a wide range of hardware are indispensable[**Cardona et al.**, **2020**]. This section reviews several related papers that aim to implement VANETs and SDVN in physical hardware using open sourced components, outlining the key details of their approach and architecture. [**Raviglione et al.**, **2019**] Raviglione et al. present a demo paper that assembles an open-source platform based on PC Engines' boards and Unex's WNICs. The purpose of this paper is to create a testbed for testing applications that communicate in the vehicular environment. Even though this paper does not attempt to implement SDN principles, it is relevant because it provides all the necessary hardware and software components to implement a vehicular testbed. The authors assembled two boards consisting of the embedded PC Engines APU1D board with an AMD G-series dual-core T40E x86 CPU with 64-bit support and 2 GB DRAM. Communication via 802.11p was achieved using the Unex DHXA-222 mPCIe card as the WNIC. This chip is based on the Atheros AR9462 chipset which is supported by the ath9k Linux driver. To enhance storage and memory performance, a SATA III Transcend MSA370 MCL NAND Flash SSD was installed. The OS used was OpenWrt release 18.06.1 with Linux kernel 4.14.63. The authors made modifications to the ath9k Linux driver in order to utilize the channels of the 5.8/5.9 GHz frequency band in accordance with ITS-G5 standards. These modifications were then integrated into the OpenC2X project, which was subsequently ported to OpenWrt. The paper also reviews some modifications and implementations made in higher layers, but these are not relevant to the problem addressed in this thesis.

[**Sedar et al.**, **2021**] Sedar et al. developed and validated an experimental, standards-compliant OBU. Their experimental platform is based on an open-source software implementation of the ETSI C-ITS protocol stack. Its purpose is to facilitate interoperability in communication between various devices and cloud-based services. The OBU was built using general purpose hardware in the form of a generic laptop, running Ubuntu 18.04. To grant cellular connectivity, it was connected to an LTE AirPrime EM7565 modem from Sierra Wireless which was in part connected to the 4G cellular network of Vodafone-Spain. The experimental OBU is also connected to external hardware devices, these being a 4G/5G cellular modem, a GPS/GNSS receiver and a connector to receive information from in-vehicle sensors. The article presents an overview of the current state of open-source software implementations of the ETSI C-ITS protocol stack. It mentions OpenC2X and Vanetza as the two primary implementations. Although real vehicles were not used in testing, the authors state that their experimental platform can be easily integrated into any vehicle.

[**Secinti et al.**, **2017**] Secinti et al. proposed an architectural model that implements SDN and virtualization principles in order to enable VANET with Wi-Fi access capability. In this architecture, both the OBU and the RSU are implemented using the same type of hardware and software. Both have been implemented using a Raspberry Pi, with the wireless connectivity being provided by the Realtek 5370 Wi-Fi SoC. These switches are implemented using OpenvSwitch v2.3.90 running on OpenWRT. OpenvSwitch is a software switch implementation that is open source and natively supported by the Linux kernel[**https://www.openvswitch.org/**]. Finally, the controller is implemented using OpenDaylight and the southbound API used is Openflow.

[**Rito et al.**, **2023**] Rito et al. present the deployment and experimentation architecture of the Aveiro Tech City Living Lab in Portugal. The implementation involves a diverse range of devices connected through fiber, radio ITSG-5, and cellular links, utilizing various technologies such as SDN, named data networking, and fog computing. Vehicle and roadside stations are implemented using the PC Engines APU2 board equipped with an SSD, an IEEE 802.11a/b/g/n mini-PCIe wireless card and an LTE CAT-1 mini-PCIe or 5G m.2 module. Additionally, an external USB dual-band wireless adapter was also installed. The operating system used is not specified, but it is a linux distribution because in order to enable the european version of 802.11p in it they used the Linux ath9k driver. This paper implements a myriad of different technologies in vehicular infrastructure, one of them being SDN. It is notable that it does not implement SDN in vehicle ITS stations, but only in the backbone of the network. The purpose of this SDN implementation was to use the increased control in the backbone of the vehicular infrastructure and the vehicle information to predict and execute handovers in advance. The authors developed a custom protocol dubbed OBUInfo to provide CAM information to the controller. This provides the controller with location, heading, speed, and vehicle type information, which is useful in predicting future handovers ahead of time.

[**Sadio et al.**, **2020**] Sadio et al. propose a complete SDVN prototype design. The hardware used for the OBU was a Raspberry Pi 3 with Cortex-A53 × 64 1.2 GHz and SRAM 1 GB. This board has access to WiFi 2.4 GHz 802.11 b/g/n and via a Huawei E8372 LTE USB modem to LTE. The operating system used was Raspbian Stretch Lite. The authors utilized the Python Twink library to transform the devices into OpenFlow switches. This implementation fails to use the standard for communication in the VANET environment, 802.11p.

6 Conceptual Solution

The comprehensive examination of the two fundamental pillars that constitute SDVN and SDVN itself has provided insights into the potential usefulness of this technology in real-world scenarios, thereby enabling the identification of an optimal practical deployment for this technology. As such, in this section, this ideal view created from the accumulated knowledge presented in this document will be described. Before proceeding to the specifics, it is important to note that this perspective describes some hypothetical optimal objectives for an implementation. Consequently, any practical implementations developed in this thesis will be constrained by the availability and costs of the requisite hardware and the availability and compatibility of the related software, in addition to timing and implementation constraints. In light of the requirement for interoperability, as delineated in section X, an SDN-compatible OBU deployment is only viable in real-world scenarios if it is capable of communicating with ETSI devices. This limitation for implementation is only applicable in real-world deployments, as the devices must comply with, or at the very least be compatible and able to collaborate with, the surrounding protocols in order to be effective. No existing software stack for SDN is compatible with the ITS-G5 protocol stack. Consequently, the most straightforward approach to create a compatible environment is to utilize P4's capacity to reprogram the packet logic of a network. The vast majority of research papers on SDVN only implement the first generation of SDN, with the few that delve into P4 and its advantages only doing so on a theoretical level, never providing an implementation platform. To the best of our knowledge, and in accordance with the findings of Sarpong et al. [**Sarpong et al.**, **2023**], no work has been published using P4 as the data plane technology in a SDVN implementation. It is important to acknowledge that the aforementioned limitation does diminish the freedom introduced by P4. However, it does not entirely negate its advantages, because the ability to modify network protocols enables protocol experimentation, thereby allowing existing protocols to be tested, validated, and optimized with greater ease. From all this, we can conclude the ideal SDVN scenario would have a P4 device mimicking protocol fields of the ETSI defined protocol stack, just with changes to the controller. The controller brings us to the second issue of controller distribution. Section X displays the three different possible locations where the controller can be implemented, but most SDN developments only implement controller instances in the wired infrastructure. Due to the issues touched upon in subsection Y(Mandated Interoperability) and X(Issues with maintaining a global network view) devices must be able to function in tandem with traditional devices and without connection to infrastructure. Ku et al.[**Ku et al.**, **2014**], although the first, had already raised these issues deserved attention by incorporating recovery mechanisms in their experimental scenario. They used a backup traditional algorithm when the device lost connection with the controller, demonstrating that SDVN must have failure recovery mechanisms to ensure the network works in all scenarios. Ultimately, our research does not aim to find the optimal controller distribution scheme in software-defined vehicular environments, but for the aforementioned reasons, a practical SDVN deployment must contain a local controller able to operate each individual device and take over with something similar to traditional software if the connection to the infrastructure is lost.

Bibliography

[1] J. Jakubiak and Y. Koucheryavy. “State of the Art and Research Challenges for VANETs”. In: *2008 5th IEEE Consumer Communications and Networking Conference*. 2008 5th IEEE Consumer Communications and Networking Conference. ISSN: 2331-9860. 2008-01, pp. 912–916. doi: 10.1109/ccnc08.2007.212 (cit. on pp. 3, 5, 14, 25, 26).

[2] S. Al-Sultan et al. “A comprehensive survey on vehicular Ad Hoc network”. In: *Journal of Network and Computer Applications* 37 (2014-01-01), pp. 380–392. issn: 1084-8045. doi: 10.1016/j.jnca.2013.02.036. url: https://www.sciencedirect.com/science/article/pii/S108480451300074X (visited on 2022-11-02) (cit. on pp. 3, 5, 12, 13, 20, 21, 25, 26).

[3] W. Liang et al. “Vehicular Ad Hoc Networks: Architectures, Research Issues, Methodologies, Challenges, and Trends”. In: *International Journal of Distributed Sensor Networks* 11.8 (2015-08-01), p. 745303. issn: 1550-1477. doi: 10.1155/2015/745303. url: http://journals.sagepub.com/doi/10.1155/2015/745303 (visited on 2023-02-22) (cit. on pp. 3, 5, 18, 22, 25).

[4] H. Dinh Thai et al. “Applications of Repeated Games in Wireless Networks: A Survey”. In: *IEEE Communications Surveys & Tutorials* 17 (2015-01-11). doi: 10.1109/COMST.2015.2445789 (cit. on p. 4).

[5] Y. Toor et al. “Vehicle Ad Hoc networks: applications and related technical issues”. In: *IEEE Communications Surveys & Tutorials* 10.3 (2008). Conference Name: IEEE Communications Surveys & Tutorials, pp. 74–88. issn: 1553-877X. doi: 10.1109/COMST.2008.4625806 (cit. on pp. 4–6, 14, 15, 18, 19, 23, 26).

[6] S. M. Corson and J. P. Macker. *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*. Request for Comments RFC 2501. Num Pages: 12. Internet Engineering Task Force, 1999-01. doi: 10.17487/RFC2501. url: https://datatracker.ietf.org/doc/rfc2501 (visited on 2023-09-28) (cit. on p. 6).

[7] C2C-CC. *CAR 2 CAR Communication Consortium Manifesto - Overview of the C2C-CC System*. 2007-08-28. url: https://www.car-2-car.org/fileadmin/documents/General_Documents/C2C-CC_Manifesto_Aug_2007.pdf (visited on 2023-04-23) (cit. on p. 6).

[8] ETSI. *Intelligent Transport Systems (ITS); Communications Architecture*. 2010-09. url: https://www.etsi.org/deliver/etsi_en/302600_302699/302665/01.01.01_60/en_302665v010101p.pdf (visited on 2023-04-27) (cit. on pp. 7–13, 22, 24).

[9] ETSI. *Intelligent Transport Systems (ITS); ITS-G5 Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*. 2020-01. url: https://www.etsi.org/deliver/etsi_en/302600_302699/302663/01.03.01_60/en_302663v010301p.pdf (visited on 2023-11-08) (cit. on p. 13).

[10] M. S. Anwer and C. Guy. "A Survey of VANET Technologies". In: *Journal of Emerging Trends in Computing and Information Sciences* (2014) (cit. on pp. 13, 17).

[11] *History of IEEE*. url: https://www.ieee.org/about/ieee-history.html (visited on 2023-11-14) (cit. on p. 14).

[12] R.
bibinitperiod Schwarz. "Intelligent Transportation Systems Using IEEE 802.11p". In: (2019-02-14) (cit. on p. 14).

[13] D. Jiang and L. Delgrossi. "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments". In: IEEE Vehicular Technology Conference. 2008-06-14, pp. 2036–2040. doi: 10.1109/VETECS.2008.458 (cit. on pp. 14, 15, 23).

[14] A. Festag. "Cooperative intelligent transport systems standards in europe". In: *IEEE Communications Magazine* 52.12 (2014-12). Conference Name: IEEE Communications Magazine, pp. 166–172. issn: 1558-1896. doi: 10.1109/MCOM.2014.6979970 (cit. on pp. 15, 16, 19–22).

[15] N. Asselin-Miller et al. "Study on the Deployment of C-ITS in Europe: Final Report". In: 1 (2016-05-02) (cit. on pp. 15, 16, 51, 67).

[16] J. Härri and J. Kenney. "Multi-Channel Operations, Coexistence and Spectrum Sharing for Vehicular Communications". In: ed. by C. Campolo, A. Molinaro, and R. Scopigno. Book Title: Vehicular ad hoc Networks. Cham: Springer International Publishing, 2015, pp. 193–218. isbn: 978-3-319-15496-1 978-3-319-15497-8. doi: 10.1007/978-3-319-15497-8_7. url: https://link.springer.com/10.1007/978-3-319-15497-8_7 (visited on 2023-10-05) (cit. on pp. 15–17).

[17] ETSI. *Electromagnetic compatibility and Radio spectrum Matters (ERM); Intelligent Transport Systems (ITS); Part 1: Technical characteristics for pan-European harmonized communications equipment operating in the 5 GHz frequency range and intended for critical road-safety applications; System Reference Document*. 2005-06. url: https://www.etsi.org/deliver/etsi_tr/102400_102499/10249201/01.01.01_60/tr_10249201v010101p.pdf (visited on 2023-11-17) (cit. on p. 15).

[18] Ş. ŞORIGA. “ITS-G5 AND MOBILE WIMAX PERFORMANCE IN VEHICLE-TO-INFRASTRUCTURE COMMUNICATIONS”. In: *ISSN 1454-234x* UPB Scientific Bulletin, Series C: Electrical Engineering. (2012). url: https://www.scientificbulletin.upb.ro/rev_docs_arhiva/full4d5_755707.pdf (visited on 2023-10-16) (cit. on p. 16).

[19] “IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Next Generation V2X”. In: *IEEE Std 802.11bd-2022 (Amendment to IEEE Std 802.11-2020 as amended by IEEE Std 802.11ax-2021, IEEE Std 802.11ay-2021, IEEE Std 802.11ba-2021, IEEE Std 802.11-2020/Cor 1-2022, and IEEE Std 802.11az-2022)* (2023-03). Conference Name: IEEE Std 802.11bd-2022 (Amendment to IEEE Std 802.11-2020 as amended by IEEE Std 802.11ax-2021, IEEE Std 802.11ay-2021, IEEE Std 802.11ba-2021, IEEE Std 802.11-2020/Cor 1-2022, and IEEE Std 802.11az-2022), pp. 1–144. doi: 10.1109/IEEESTD.2023.10063942. url: https://ieeexplore.ieee.org/document/10063942 (visited on 2023-10-25) (cit. on p. 17).

[20] *3GPP – The Mobile Broadband Standard*. 3GPP. url: https://www.3gpp.org/ (visited on 2023-11-16) (cit. on p. 17).

[21] S. Gyawali et al. “Challenges and Solutions for Cellular Based V2X Communications”. In: *IEEE Communications Surveys & Tutorials* 23.1 (2021), pp. 222–255. issn: 1553-877X, 2373-745X. doi: 10.1109/COMST.2020.3029723. url: https://ieeexplore.ieee.org/document/9217500/ (visited on 2023-10-16) (cit. on pp. 17, 18).

[22] *5GAA*. 5GAA. url: https://5gaa.org/ (visited on 2023-11-16) (cit. on p. 17).

[23] R. Weber, J. Misener, and V. Park. “C-V2X - A Communication Technology for Cooperative, Connected and Automated Mobility”. In: *Mobile Communication - Technologies and Applications; 24. ITG-Symposium*. Mobile Communication - Technologies and Applications; 24. ITG-Symposium. 2019-05, pp. 1–6. url: https://ieeexplore.ieee.org/abstract/document/8731783 (visited on 2023-10-23) (cit. on p. 18).

[24] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1: Requirements*. 2014-04. url: https://www.etsi.org/deliver/etsi_en/302600_302699/30263601/01.02.01_60/en_30263601v010201p.pdf (visited on 2023-11-27) (cit. on p. 19).

[25] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 2: Scenarios*. 2013-11. url: https://www.etsi.org/deliver/etsi_en/302600_302699/30263602/01.02.01_60/en_30263602v010201p.pdf (visited on 2023-11-27) (cit. on p. 19).

[26] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 3: Network Architecture*. 2014-12. url: https://www.etsi.org/deliver/etsi_en/302600_302699/30263603/01.02.01_60/en_30263603v010201p.pdf (visited on 2023-10-06) (cit. on pp. 19, 21, 24).

[27] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality*. 2020-01. url: https://www.etsi.org/deliver/etsi_en/302600_302699/3026360401/01.04.01_60/en_3026360401v010401p.pdf (visited on 2023-11-27) (cit. on p. 19).

[28] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol*. 2019-05. url: https://www.etsi.org/deliver/etsi_en/302600_302699/3026360501/02.02.01_60/en_3026360501v020201p.pdf (visited on 2023-11-27) (cit. on p. 19).

[29] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 6: Internet Integration; Sub-part 1: Transmission of IPv6 Packets over GeoNetworking Protocols*. 2014-05. url: https://www.etsi.org/deliver/etsi_en/302600_302699/3026360601/01.02.01_60/en_3026360601v010201p.pdf (visited on 2023-11-27) (cit. on p. 19).

[30] A. Festag. "Standards for vehicular communication—from IEEE 802.11p to 5G". In: *e & i Elektrotechnik und Informationstechnik* 132.7 (2015-11-01), pp. 409–416. issn: 1613-7620. doi: 10.1007/s00502-015-0343-0. url: https://doi.org/10.1007/s00502-015-0343-0 (visited on 2023-10-16) (cit. on pp. 19, 20).

[31] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. 2019-04. url: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.04.01_60/en_30263702v010401p.pdf (visited on 2023-11-27) (cit. on p. 20).

[32] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*. 2019-04. url: https://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.03.01_60/en_30263703v010301p.pdf (visited on 2023-11-27) (cit. on p. 20).

[33] A. K. Malhi, S. Batra, and H. S. Pannu. "Security of vehicular ad-hoc networks: A comprehensive survey". In: *Computers & Security* 89 (2020-02), p. 101664. issn: 01674048. doi: 10.1016/j.cose.2019.101664. url: https://linkinghub.elsevier.com/retrieve/pii/S0167404818312872 (visited on 2023-11-28) (cit. on pp. 22, 23).

[34] H. Hasrouny et al. "VANet security challenges and solutions: A survey". In: *Vehicular Communications* 7 (2017-01-01), pp. 7–20. issn: 2214-2096. doi: 10.1016/j.vehcom.2017.01.002. url: https://www.sciencedirect.com/science/article/pii/S2214209616301231 (visited on 2023-12-15) (cit. on p. 23).

[35] ETSI. *Intelligent Transport Systems (ITS); Security; Security Services and Architecture*. 2010-09. url: https://www.etsi.org/deliver/etsi_ts/102700_102799/102731/01.01.01_60/ts_102731v010101p.pdf (visited on 2023-11-28) (cit. on p. 23).

[36] ETSI. *Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management*. 2018-04. url: https://www.etsi.org/deliver/etsi_ts/102900_102999/102940/01.03.01_60/ts_102940v010301p.pdf (visited on 2023-12-05) (cit. on p. 25).

[37] ETSI. *Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management; Release 2*. 2021-07. url: https://www.etsi.org/deliver/etsi_ts/102900_102999/102940/02.01.01_60/ts_102940v020101p.pdf (visited on 2023-11-28) (cit. on p. 24).

[38] M. J. N. Mahi et al. "A Review on VANET Research: Perspective of Recent Emerging Technologies". In: *IEEE Access* 10 (2022). Conference Name: IEEE Access, pp. 65760–65783. issn: 2169-3536. doi: 10.1109/ACCESS.2022.3183605. url: https://ieeexplore.ieee.org/abstract/document/9797696 (visited on 2023-12-19) (cit. on p. 26).

[39] M. Lu et al. "Pan-European deployment of C-ITS: the way forward". In: (2019) (cit. on pp. 26, 27).

[40] B. A. A. Nunes et al. "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks". In: *IEEE Communications Surveys & Tutorials* 16.3 (2014). Conference Name: IEEE Communications Surveys & Tutorials, pp. 1617–1634. issn: 1553-877X. doi: 10.1109/SURV.2014.012214.00180 (cit. on pp. 28, 29, 31–34, 43–45).

[41] N. Feamster, J. Rexford, and E. Zegura. "The road to sdn". In: *Queue* 11 (2013-12). doi: 10.1145/2559899.2560327 (cit. on pp. 28–33).

[42] W. Xia et al. "A Survey on Software-Defined Networking". In: *IEEE Communications Surveys & Tutorials* 17.1 (2015). Conference Name: IEEE Communications Surveys & Tutorials, pp. 27–51. issn: 1553-877X. doi: 10.1109/COMST.2014.2330903 (cit. on pp. 28, 31–35, 38, 43–45, 47).

[43] *Open Networking Foundation*. en-US. url: https://opennetworking.org/ (visited on 2024-01-12) (cit. on p. 28).

[44] L. L. Peterson et al. *Software-Defined Networks: A Systems Approach*. English. Wroclaw: Systems Approach LLC, 2021-01. isbn: 978-1-73647-210-1 (cit. on pp. 28–30, 33–36, 38, 40–47).

[45] R. Bifulco and G. Retvari. "A Survey on the Programmable Data Plane: Abstractions, Architectures, and Open Problems". en. In: *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*. Bucharest, Romania: IEEE, 2018-06, pp. 1–7. isbn: 978-1-5386-7801-5. doi: 10.1109/HPSR.2018.8850761. url: https://ieeexplore.ieee.org/document/8850761/ (visited on 2023-12-22) (cit. on pp. 28, 29, 31, 33–36, 38, 40, 45).

[46] A. S. Thyagaturu et al. "Software Defined Optical Networks (SDONs): A Comprehensive Survey". In: *IEEE Communications Surveys & Tutorials* 18.4 (2016). Conference Name: IEEE Communications Surveys & Tutorials, pp. 2738–2786. issn: 1553-877X. doi: 10.1109/COMST.2016.2586999 (cit. on pp. 29, 30, 32, 34, 42, 44, 45).

[47] A. Alowa. “Scalable Reliable Controller Placement in Software Defined Networking”. In: (2020-09). url: https://core.ac.uk/reader/355870830 (visited on 2024-03-19) (cit. on p. 30).

[48] A. Liatifis et al. “Advancing SDN from OpenFlow to P4: A Survey”. In: *ACM Computing Surveys* 55.9 (2023-01), 186:1–186:37. issn: 0360-0300. doi: 10.1145/3556973. url: https://dl.acm.org/doi/10.1145/3556973 (visited on 2024-01-11) (cit. on pp. 30, 36, 47).

[49] R. C. Sofia and J. Soldatos. “Shaping the Future of IoT with Edge Intelligence; How Edge Computing Enables the Next Generation of IoT Applications”. en. In: (2024) (cit. on p. 30).

[50] D. Kreutz et al. “Software-Defined Networking: A Comprehensive Survey”. In: *Proceedings of the IEEE* 103.1 (2015-01). Conference Name: Proceedings of the IEEE, pp. 14–76. issn: 1558-2256. doi: 10.1109/JPROC.2014.2371999 (cit. on pp. 30–35, 37, 40, 42–47).

[51] N. McKeown et al. “OpenFlow: enabling innovation in campus networks”. In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008-03), pp. 69–74. issn: 0146-4833. doi: 10.1145/1355734.1355746. url: https://doi.org/10.1145/1355734.1355746 (visited on 2023-02-10) (cit. on pp. 31, 35, 40, 43).

[52] S. Li et al. “Protocol Oblivious Forwarding (POF): Software-Defined Networking with Enhanced Programmability”. In: *IEEE Network* 31.2 (2017-03). Conference Name: IEEE Network, pp. 58–66. issn: 1558-156X. doi: 10.1109/MNET.2017.1600030NM (cit. on pp. 34, 37).

[53] Z. Latif et al. “A comprehensive survey of interface protocols for software defined networks”. en. In: *Journal of Network and Computer Applications* 156 (2020-04), p. 102563. issn: 1084-8045. doi: 10.1016/j.jnca.2020.102563. url: https://www.sciencedirect.com/science/article/pii/S1084804520300370 (visited on 2022-11-07) (cit. on pp. 35, 42, 44–47).

[54] F. Hauser et al. “A survey on data plane programming with P4: Fundamentals, advances, and applied research”. In: *Journal of Network and Computer Applications* 212 (2021-08), p. 103561. issn: 1084-8045. doi: 10.1016/j.jnca.2022.103561. url: https://www.sciencedirect.com/science/article/pii/S1084804522002028 (visited on 2024-01-26) (cit. on pp. 36–39, 41, 43, 44).

[55] P. Bosshart et al. “P4: programming protocol-independent packet processors”. In: *ACM SIGCOMM Computer Communication Review* 44.3 (2014-07), pp. 87–95. issn: 0146-4833. doi: 10.1145/2656877.2656890. url: https://doi.org/10.1145/2656877.2656890 (visited on 2022-11-30) (cit. on pp. 37, 38).

[56] *Open Compute Project*. en. url: https://www.opencompute.org (visited on 2024-03-15) (cit. on p. 39).

[57] *Indigo - Confluence*. url: https://floodlight.atlassian.net/wiki/spaces/Indigo/overview?homepageId=2392077 (visited on 2024-03-15) (cit. on p. 39).

[58] *Open Network Linux*. url: http://opennetlinux.org/ (visited on 2024-03-18) (cit. on p. 42).

[59] *Stratum*. en-US. url: https://opennetworking.org/stratum/ (visited on 2024-03-18) (cit. on p. 42).

[60] *Sonic Foundation – Linux Foundation Project*. en-US. url: https://sonicfoundation.dev/ (visited on 2024-03-18) (cit. on p. 42).

[61] *OpenConfig*. url: https://openconfig.net/ (visited on 2024-03-15) (cit. on pp. 43, 44).

[62] L. Zhu et al. "SDN Controllers: A Comprehensive Analysis and Performance Evaluation Study". In: *ACM Computing Surveys* 53.6 (2020), 133:1–133:40. issn: 0360-0300. doi: 10.1145/3421764. url: https://doi.org/10.1145/3421764 (visited on 2023-02-04) (cit. on pp. 45, 46).

[63] K. Smida et al. "Efficient SDN Controller for Safety Applications in SDN-Based Vehicular Networks: POX, Floodlight, ONOS or OpenDaylight?" In: *2020 IEEE Eighth International Conference on Communications and Networking (ComNet)*. ISSN: 2473-7585. 2020-10, pp. 1–6. doi: 10.1109/ComNet47917.2020.9306095. url: https://ieeexplore.ieee.org/abstract/document/9306095 (visited on 2024-02-19) (cit. on pp. 48, 50, 53).

[64] I. Ku et al. "Towards software-defined VANET: Architecture and services". In: *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*. 2014-06, pp. 103–110. doi: 10.1109/MedHocNet.2014.6849111 (cit. on pp. 48, 50).

[65] J. Bhatia et al. "Software defined vehicular networks: A comprehensive review". en. In: *International Journal of Communication Systems* 32.12 (2019), e4005. issn: 1099-1131. doi: 10.1002/dac.4005. url: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4005 (visited on 2022-10-11) (cit. on pp. 48, 53).

[66] N. Cardona et al. "Software-Defined Vehicular Networking: Opportunities and Challenges". In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 219971–219995. issn: 2169-3536. doi: 10.1109/ACCESS.2020.3042717 (cit. on pp. 50, 52, 53).

[67] W. Ben Jaballah, M. Conti, and C. Lal. "Security and design requirements for software-defined VANETs". In: *Computer Networks* 169 (2020-03), p. 107099. issn: 1389-1286. doi: 10.1016/j.comnet.2020.107099. url: https://www.sciencedirect.com/science/article/pii/S1389128619306553 (visited on 2023-12-15) (cit. on pp. 50–52).

[68] R. Sarpong, B. Sousa, and F. Araujo. "The potential of SDNs and multihoming in VANETs: A Comprehensive Survey". en. In: (2023-11) (cit. on p. 50).

[69] S. Toufga et al. "Towards Dynamic Controller Placement in Software Defined Vehicular Networks". In: *Sensors* 20 (2020-03), p. 1701. doi: 10.3390/s20061701 (cit. on p. 52).

[70] S. Toufga et al. "OpenFlow based Topology Discovery Service in Software Defined Vehicular Networks: limitations and future approaches". In: *2018 IEEE Vehicular Networking Conference (VNC)*. ISSN: 2157-9865. 2018-12, pp. 1–4. doi: 10.1109/VNC.2018.8628349 (cit. on pp. 52, 53).

[71] L. Nkenyereye et al. “Software-Defined Network-Based Vehicular Networks: A Position Paper on Their Modeling and Implementation”. en. In: *Sensors* 19.17 (2019-01). Number: 17 Publisher: Multidisciplinary Digital Publishing Institute, p. 3788. issn: 1424-8220. doi: 10.3390/s19173788. url: https://www.mdpi.com/1424-8220/19/17/3788 (visited on 2022-10-10) (cit. on p. 52).

I

Bundles of services

Service bundle	C-ITS Services	Rationale
Bundle 1 Day 1, V2V, ITS-G5	• Emergency brake light • Emergency vehicle approaching • Slow or stationary vehicle(s) • Traffic jam ahead warning • Hazardous location notification	• Day 1 safety-based V2V services based on ITS-G5 communication, likely to be deployed to vehicles supported by US legislation
Bundle 2 Day 1, V2I, mainly applicable to motorways	• In-vehicle signage • In-vehicle speed limits • Probe vehicle data • Shockwave damping • Road works warning • Weather conditions	• Day 1 V2I, services that deliver most benefit to motorways. Some services listed here may also be applicable to other road types
Bundle 3 Day 1, V2I, mainly applicable to urban areas	• Green Light Optimal Speed Advisory (GLOSA) / Time To Green (TTG) • Signal violation/Intersection safety • Traffic signal priority request by designated vehicles	• Day 1 V2I, services expected to only be applicable in urban areas. Therefore, these services are in a separate bundle to those in Bundle 2
Bundle 4 Day 1.5, V2I, Parking Information	• Off street parking information • On street parking management and information • Park & Ride information • Information on AFV fuelling & charging stations	• C-ITS services intended to provide information regarding parking (and refuelling) to drivers
Bundle 5 Day 1.5, V2I, Traffic and other information	• Traffic information and smart routing	• C-ITS services intended to provide traffic information to drivers
Bundle 6 Day 1.5, Freight specific services	• Loading zone management • Zone access control management	• Zone management services
Bundle 7 Day 1.5, V2X (mainly applicable to urban areas), likely to be ITS-G5	• Vulnerable road user protection (pedestrians and cyclists)	• V2X service expected to be post day 1. Communication method is likely to be ITS-G5. Main benefits are likely to be seen in urban areas.
Bundle 8 Day 1.5, V2V, likely to be ITS-G5	• Cooperative collision risk warning • Motorcycle approaching indication	• Post day 1 V2V services that are likely to be based on ITSG5. As for Day 1 services, V2V and V2I services are in separate service bundles.
Bundle 9 Day 1.5, V2I	• Wrong way driving	• Post day 1 V2I service. As for Day 1 services, V2V and V2I services are in separate service bundles.

Table 2: C-ITS service bundles for scenario building[15]