



Software-based Networking in Railway Systems

Mariana Macedo de Faria Rodrigues Cruz

Thesis to obtain the Master of Science Degree in

Telecommunications and Informatics Engineering

Supervisor: Prof. Rui António dos Santos Cruz

Examination Committee

Chairperson: Prof. Ricardo Jorge Fernandes Chaves

Supervisor: Prof. Rui António dos Santos Cruz

Member of the Committee: Prof. Fernando Henrique Côrte-Real Mira da Silva

January 2021

Acknowledgments

First of all, I would like to thank my mother for all the patience she had in accompanying me throughout the course, but mainly in this most challenging phase that was the realization of this project. Thank you for always encouraging me in all my adventures and for always supporting my ideas. Also would like to thank my father, grandparents, brother and aunt for all the support and encouragement not only in the last few years but since ever I can remember.

A huge part of my journey in Instituto Superior Técnico (IST) were the colleagues that have done this journey with me, that I'm lucky enough to assure that some of them became great friends, and that they are coming with me for life. I also want to thank my high school friends who never hesitate to help me and are always and always present in my adventures and life experiences, without them nothing would be possible. To all of them, a special thank you for all the good times that we lived until now that, for sure, helped this journey becoming easier and funnier.

I would also like to acknowledge my dissertation supervisor Prof. Rui Santos Cruz for his insight, support, sharing of knowledge and big patience that has made this Thesis possible.

Last but not least i would also like to thank Thales Portugal and specially Engineer Moacir Ferreira for all the help during this phase and for accepting my thesis proposal, allowing me to be involved in a real project in order to gain knowledge of a real implementation on a Portugal railway and roadside infrastructure.

To each and every one of you – Thank you.

Abstract

Roadside and Railway networking systems are critical infrastructures that must support fast, reliable and low delay communications. Technologies still used in those infrastructures—such as GSM-Railway (GSM-R), evolving into LTE-Railway (LTE-R) in the case of railways—are becoming more complex, yet more flexible, and evolving to “software defined” architectures. The technologies presented and explored in this work are expected to change the future of these types of critical systems.

The main goal of this work is to study the key benefits of using the Software Defined Networking (SDN) model to control the next generation railways/roadside networking infrastructures. The safety, security and correctness properties of SDN controllers in high availability railway/roadside networks will be analysed, with the focus on railway networks for compliance with the European Rail Traffic Management System (ERTMS)/European Train Control System (ETCS) specifications, and in the context of a real project being deployed in the Portuguese Railways by Thales Portugal. From the analysis of the project requirements and the selected network equipment, it was possible to evaluate and propose adequate methods to automatically derive network re-configuration strategies, namely in case of failure events—e.g., preventing packet loss affecting vital railway traffic signalling procedures—in order to reduce field intervention and complexity in those infrastructures.

Keywords

Software Defined Networking (SDN); Network Functions Virtualization (NFV); Railway Networks; Roadside Networks; Open Network Operating System (ONOS) Controller; LTE-Railway (LTE-R);

Resumo

Os sistemas de comunicação para redes rodoviárias e ferroviárias são infraestruturas críticas que devem suportar comunicações rápidas, confiáveis e apresentar atrasos muito baixos. As tecnologias ainda usadas nessas infraestruturas—tais como *GSM-Railway (GSM-R)*, evoluindo para *LTE-Railway (LTE-R)* no caso das ferrovias—estão-se tornando mais complexas, porém mais flexíveis, e evoluindo para arquiteturas “definidas por software”. Espera-se que as tecnologias apresentadas e exploradas neste trabalho mudem o futuro desses tipos de sistemas críticos.

O objetivo principal deste trabalho é o estudo dos benefícios chave na utilização do modelo *Software Defined Networking (SDN)* para o controle das infraestruturas de comunicação ferroviárias/rodoviárias de próxima geração. As propriedades de segurança, proteção e exatidão dos controladores SDN em configurações de alta disponibilidade para essas redes, serão analisadas, com foco nas ferrovias e conformidade com as especificações *European Rail Traffic Management System (ERTMS)/European Train Control System (ETCS)*, e no contexto de um projeto real em implementação na rede Ferroviária Portuguesa pela Thales Portugal. A partir da análise dos requisitos do projeto e do equipamento de rede selecionado, foi possível avaliar e propor métodos adequados para derivar automaticamente estratégias de reconfiguração de rede, nomeadamente em caso de eventos de falha—para, por exemplo, procedimentos que permitam prevenir a perda de pacotes que afetem a sinalização de tráfego ferroviário vital—afim de reduzir a complexidade e a intervenção de campo nessas infraestruturas.

Palavras Chave

Software Defined Networking (SDN); *Network Functions Virtualization (NFV)*; Redes de comunicação ferroviárias; Redes de comunicação rodoviárias; Controlador *Open Network Operating System (ONOS)*; *LTE-Railway (LTE-R)*;

Contents

1	Introduction	1
1.1	Motivation	3
1.2	Organization of the Document	4
2	Theoretical Background and Literature Review	5
2.1	Roadside Communications Networks	7
2.2	Railway Communications Networks	12
2.3	Software Defined Networks	17
2.3.1	Southbound Protocols	19
2.3.2	SDN Controllers	21
2.3.3	Network Functions Virtualization	22
2.3.4	Controller clustering	22
2.4	SDN applications	23
2.4.1	For Roadside Networks	23
2.4.2	Application of SDN in a Roadside Network - Example	25
3	Proposed Architecture	27
3.1	Railway Architecture Design	29
3.2	SDN implementation in a Railway Network	31
3.2.1	Core Network	33
3.2.2	Access Network	33
3.2.3	Access Functions	33
4	Demonstration	35
4.1	Implementation	37
4.1.1	Alcatel-Lucent Omniswitch (ALU-OS)	37
4.1.2	EdgeCore switch	38
4.2	Proof of concept scenario	38
4.2.1	SDN Controller	40
4.2.2	Simulation Framework	41

4.3	Use cases	44
4.3.1	MPLS-like features in SDN	45
4.3.2	Advantages of SD-WAN Over MPLS	46
4.3.3	Use Case Scenarios	47
5	Evaluation	49
5.1	Use Case: Video surveillance	51
5.2	Use Case: Recovery from link failure	53
5.3	Intent-based networking	55
5.4	Use Case: Controller failure	58
6	Conclusion	61
6.1	Conclusions	63
6.2	System Limitations and Future Work	64
A	Topology Code	71

List of Figures

2.1	Enhanced OSI reference model of IDB.	11
2.2	Roadside ITS Data Bus (RIDB) communication system.	12
2.3	Elements of the European Rail Traffic Management System (ERTMS).	13
2.4	GSM-R simplified system architecture.	13
2.5	Schematic overview of ETCS Level 2 architecture.	16
2.6	SDN Interfaces.	18
2.7	Software Defined Networking (SDN) architecture.	19
2.8	Types of treatment for incoming packets of OpenFlow protocol.	20
2.9	Proposed architecture.	26
3.1	The Ovar-Gaia section high-level network topology of the IP Railway Project.	30
3.2	High-level SDN architecture for the railway communications network.	32
4.1	Diagram of the Demonstrator Test Bed of the IP Railway project.	39
4.2	ONOS architecture.	40
4.3	ONOS GUI, showing the Topology of the Demonstration scenario.	41
4.4	Mininet-wifi topology initialization for the Demonstration scenario.	42
4.5	Mininet-wifi topology creation of Demonstration scenario in MiniEdit.	43
4.6	Mininet-wifi wireless coverage graphical representation in real-time.	43
4.7	iwconfig command in ap1	44
4.8	Output of iwconfig command in sta1	44
5.1	Video stream characteristics on the server side.	51
5.2	Video Streaming Test in the emulated network	52
5.3	Media statistics of the Streaming session	52
5.4	Flow entries in tableID=0 related to video streaming.	52
5.5	Simulating a Link Down in the path of h13 to sta1	53
5.6	Shortest path between h13 and sta1	54

5.7	Shortest path between h13 and sta1 after link failure.	54
5.8	Failure of links while h13 ping sta1	55
5.9	Switch Contumil (s9) ports.	56
5.10	Flows in Contumil (s9) before link failure.	56
5.11	GUI representation of the intent created between hosts.	57
5.12	GUI representation of the intent created after the link failure.	57
5.13	Flows in Contumil (s9) after link failure.	58
5.14	Relationship between Atomix nodes and ONOS nodes.	59
5.15	Creation of the ONOS cluster.	59

List of Tables

2.1	Quality Of Service (QoS) parameters for GSM-Railway (GSM-R) (European Train Control System (ETCS)	13
2.2	Comparison of the most popular open source SDN controllers	22
3.1	ALU-OS main characteristics.	31

Listings

A.1	PYTHON Topology Code	71
-----	--------------------------------	----

Acronyms

ACL	Access Control List
AI	Artificial Intelligence
ALU	Alcatel-Lucent
ALU-OS	Alcatel-Lucent Omniswitch
API	Application Programming Interface
AP	Access Point
ATP	Automatic Train Protection
BSC	Base Station Controller
BSS	Base Station Subsystem
BS	Base Station
BTM	Balise Transmission Module
BTS	Base Transceiver Stations
CAV	Automatic Vehicle Counting
CCTV	Closed-Circuit Television
CCO	Operational Command Center
CLI	Command Line Interface
c-PIPE	Circuit 'Pipe' Emulation Services
DDoS	Distributed-Denial-of-Service
DMI	Driver Machine Interface
DSRC	Dedicated Short Range Communications
E2E	End-To-End
EPC	Evolved Packet Entities
e-PIPE	Ethernet 'Pipe' Services

ERTMS	European Rail Traffic Management System
ETCS	European Train Control System
EVC	European Vital Computer
FTN	Fixed Telecommunications Network
GPRS	General Packet Radio Service
GSM-R	GSM-Railway
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
H.265/HEVC	High Efficiency Video Coding
HD	High Definition
HR	Hirschmann
HetVNETs	Heterogeneous Vehicular Network
IBN	Intent-based networking
IDB	ITS Data Bus
IP	Infraestruturas de Portugal
IST	Instituto Superior Técnico
IT	Information Technology
ITS	Intelligent Transportation System
IT	Information Technology
IVHS	Intelligent Vehicle–Highway System
IoT	Internet of Things
LAN	Local Area Network
LTE-R	LTE-Railway
LTE	Long Term Evolution
MA	Movement Authorities
MPLS	Multi Protocol Label Switching
MSC	Mobile Switching Center
ML	Machine Learning
MS	Mobile Stations
NBI	Northbound Interface

NETCONF	Network Configuration Protocol
NFV	Network Functions Virtualization
NMC	Network Management Center
NSS	Network Switching Sub-system
OBE	On Board Equipment
OBU	On Board Unit
OF	OpenFlow
OMC	Operation and Management Centre
ONIE	Open Network Install Environment
ONL	Open Network Linux
ONOS	Open Network Operating System
OSI	Open System Interconnection
OSS	Operational Support System
PCEP	Path Computation Element Communication Protocol
PDH	Plesiochronous Digital Hierarchy
PMR	Private Mobile Radio
PMV	Variable Message Panels
PPDR	Public Protection and Disaster Relief
QSFP	Quad Small Form-factor Pluggable
QoS	Quality Of Service
RAR	Roadside-Aided Routing
RBC	Radio Block Centre
REC	Railway Emergency Call
REST	REpresentational State Transfer
RIDB	Roadside ITS Data Bus
RSE	Roadside Equipment
RSUC	RSU controller
RSU	Roadside Unit
RS	Relay Stations

RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SDH	Synchronous Digital Hierarchy
SDN	Software Defined Networking
SDVN	SDN based Vehicular ad-hoc Network
SD-WAN	Software-Defined WAN
SFP	Small Form-Factor Pluggable
SNMP	Simple Network Management Protocol
SPOF	Single Point of Failure
SSL	Secure Sockets Layer
TCC	Traffic Control Centre
TCP	Transmission Control Protocol
TC	Traffic Control
TETRA	Terrestrial Trunked Radio
TOR	Top of Rack
UHF	Ultra High Frequency
UMTS	Universal Mobile Telecommunication System
V2R	Vehicle to Roadside
V2V	Vehicle to Vehicle
V2X	Vehicle-to-everything
VANET	Vehicular Ad Hoc Network
VBS	Voice Broadcast Services
VCN	Vehicular Communication Network
VGCS	Voice Group Call Services
VII	Vehicle Infrastructure Integration
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VoIP	Voice over IP
VPLS	Virtual Private LAN Service

WAN	Wide Area Network
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
eNB	eNodeB

1

Introduction

Contents

1.1 Motivation	3
1.2 Organization of the Document	4

Each day, the advancement of technology is a reality, with computer networks rapidly evolving in size, complexity and capabilities. This rapid development covers not only industrial networks, but also other special purpose types, such as for railway or roadside networks, which are the main focus of this work, that intends to study their improvement in the light of new architectural models such as Software Defined Networking (SDN). Both railway and roadside communication networks bring a lot of complexity accompanied by a variety of critical and important features that influence the whole system and are essential for the good functioning of large metropolitan areas and smart cities. However, some of the technologies on which, traditionally, those types of networks are based, although still quite reliable, are becoming more and more complex and out-dated, so the adoption of SDN and Network Functions Virtualization (NFV) models may streamline and simplify their design, deployment and implementation processes. More importantly, in the technical perspective, these types of networks deal every day with citizens, almost 24/7, ensuring that they provide their services safely, reliably and in the best way possible, the mentioned aspects must be the main concerns for their implementation. From the human perspective, the typical specialized work of a network operations engineer is increasingly being revolutionized (and replaced), with the introduction of SDN and NFV technologies, as other software development oriented technicians, with the minimal capabilities and minimum knowledge of those technologies, become also able to perform most basic network operations tasks. So, it is also important to study and understand how those technologies work and what benefits they can bring to real life projects and to daily Operations, in order to keep up with this technological “boom”.

1.1 Motivation

Technologies such as SDN or NFV are not yet widely adopted for railway or roadside networks infrastructures, mostly because the change would be a real challenge, specially due to the fact that those infrastructures are complex, and deal with millions of people and critical systems that have a huge impact (in case of malfunction) on the every day life.

With regard to road networks, the implementation and adoption of SDN, although technologically similar to railway networks, is of a different nature, as their focus is more on vehicle automation and the implementation of autonomous driving than in traffic management (in railways) and vehicle control (of trains), and there are actually SDN projects focused on the mentioned aspects for road networks.

With regard to railway networks, critical systems such as signaling, centralized control of traffic and vehicles, including speed control, telecommunications (voice and data), among others, would benefit with the adoption of cutting edge technologies such as SDN, much more than in the roadside case, mainly because railway networks are more challenging due to the complexity and importance involved in their types of critical systems. Railway networks have special needs and requirements, such as:

(a) End-To-End (E2E) delays lower than 0.5s, or Error rates less than 1%/h; (b) support adequate bitrates for large numbers of video streams (namely for surveillance and traffic control), and (c) fast and efficient (desirably automated) network recovery time, which must be met, in the best possible way, by the design of their architecture and the careful selection of technologies for their implementation and operation (including provisioning and configuration). Therefore, in the case of their possible evolution to SDN-based architectures, those requirements must be met.

As such, one of the objectives of the work described in this document is to study the technologies, standards and solutions related and/or applicable to roadside and railway networks, namely the recent/emerging technologies such as SDN and NFV.

Another objective is of practical nature, in order to understand and evaluate if and how the integration of these technologies is viable and if they really bring advantages to these very important and critical systems, namely in a real project for a railway network.

This work was possible due to the support of Thales Portugal, which is involved in the implementation of Portuguese roadside and railway communications networks.

Therefore, although the company is involved in the deployment of a real roadside project, it has been decided to focus the study in the railway network project.

1.2 Organization of the Document

This document is organized as follows:

Chapter 2 presents the theoretical background and literature review in the state of the art and related works. Chapter 3 describes a possible architecture for a SDN-based railway communications network.

Chapter 4 presents the high-level solution for the Portuguese railway network, followed by Chapter 5 that provides the results in terms of network performance when certain events occur in a emulated scenario based on the real network infrastructure.

Chapter 6 will draw conclusions on the performed study, as well as the system limitations, suggesting also subsequent future work.

2

Theoretical Background and Literature Review

Contents

2.1 Roadside Communications Networks	7
2.2 Railway Communications Networks	12
2.3 Software Defined Networks	17
2.4 SDN applications	23

This chapter addresses the relevant concepts about railway and roadside networks, technologies such as cellular communications, SDN, NFV, describing also some projects related with those technologies in the context of railway and roadside networks, in particular, applications of SDN for both types of infrastructures.

2.1 Roadside Communications Networks

There are several infrastructures that are still poorly served, with respect to communications networks, such as suburban roads, highways and other road services, since there are still many challenges and factors that influence the performance and existence of those infrastructures, such as the high speed movement of vehicles, or their longitudinal nature.

It has become a huge concern to have good and reliable communication systems, especially in modern highways.

Emerging roadside networks, or vehicular networks are expected to contribute to improving traffic efficiency and road environment, but due to the fact that vehicle network demands are challenging, it is sometimes hard to establish solutions to meet the expectations [1].

Roadside communication systems are becoming important parts of modern roads as they can provide or enhance various services, such as voice, data video transmission and signalling. In Intelligent Transportation Systems (ITSs), which were formally known as Intelligent Vehicle–Highway Systems (IVHSs), some wireless technologies have been proposed to accomplish the many communication requirements. Usually an ITS considers three important elements: the vehicle, the driver and the road. The use of ITS is normally proposed in order to build shareable standard interfaces for in-vehicle information systems.

With the accelerated evolution of communication technologies in vehicular networks and with the increasing urgency of more and better network infrastructures, many factors contribute to the development of Vehicular Communication Network (VCN) [2]:

1. Ample adoption of IEEE 802.11 technologies, dropping the cost of technology;
2. More use of wireless broadband Internet services, such as 4G or even 5G (in a near future);
3. Broader adoption of Information Technology (IT) advanced solutions by vehicle manufacturers to address security issues of vehicles;
4. Commitment to allocate wireless spectrum for vehicular wireless communication.

The VCN technology has two levels of communication network in its infrastructure [2]:

1. Vehicular Ad Hoc Network (VANET), that enables direct communication between On Board Unit (OBU), Vehicle to Vehicle (V2V), Roadside Unit (RSU) and Vehicle to Roadside (V2R).
2. Roadside network:
 - (a) Access network, that involves the RSU and allows V2R-communications, through appropriate connections to the backbone.
 - (b) Backbone network, which is basically the backbone of RSUs, through which they communicate with each other and with the Internet.

The way the development methods of applications and solutions has changed, made possible for different solutions to emerge, and it all started with the rapid development of wireless networks.

As time goes by, the development of these technologies is contributing to the improvement of public transportation. These infrastructures have to be integrated with each other and with the vehicles, therefore the reason for implementing Vehicle Infrastructure Integration (VII), as it aims to be a vehicular network system with inter-vehicle and roadside-to-vehicle communication capabilities. The foremost elements in the VII are the OBUs, braced on vehicles to provide wireless communication, and the RSUs that together form a backbone network and are capable of providing wireless interfaces to vehicles. These RSU elements are normally connected with each other through wired links. The challenges of hybrid networks like these are huge and ultimately causing that routing becomes a challenge. The interaction between network elements can be exchanged in three different modes: Roadside Equipments (RSEs), OBUs and RSEs, or on-road vehicles and OBUs [3].

The main characteristics of this exchange of information between those elements are essentially the following [3]:

1. fluctuating data rates.
2. huge demands of robustness;
3. large transmission range;
4. broadcast of high-priority packets in a short period of time.

These types of hybrid architectures are suitable to *ad-hoc* networks due to the performance degradation when the size of the network escalates. Due to this, the network is usually divided in small clusters that form the backbone network, where each host only talks to the nearest neighbor, and the communication between remote peers is leaded and conducted by clustered heads [4].

Another important aspect in these infrastructures, is that of a set of vehicles forming VANETs and are able to communicate directly with a RSU or reach a certain RSU through multiple hops. The RSUs that are present in the same area are all joint in a "sector", which is basically a road surface surrounded

by multiple RSU neighbors. Each “sector” is created on the basis of an affiliation method, to which a unique ID is provided and a vehicle is associated to it. The numerous RSU serve one “sector” but each RSU can also serve several “sectors”, thus each RSU has to cooperate with its neighbors, such that, whenever it enters or leaves a “sector” the information has to be spread across all RSUs present in the “sector” [4].

The *ad-hoc* routing is used to provide data in a local area. There are two routing approaches: single-phase and two-phase. In both, the source tries to discover from its address the destination. If it receives a reply it knows that the receiver is based in the same area and the packets are usually directly forwarded. If not, it is expected that the destination is located on another (remote) cluster, and so the entities responsible for the incoming of packets start to act, making it the second phase of the two-phase routing [4].

The study in [4] discusses about a new and improved way of routing approach, named Roadside-Aided Routing (RAR). This technology was the first to handle the particular characteristics of roadside networks. In its essence it is a framework for efficient routing in vehicular hybrid networks instead of using a specific traditional routing protocol.

In ITS, wireless technologies have been implemented to respond to various communication requirements. The first entity responsible for building a standard interface to vehicle systems was the ITS Data Bus (IDB), but soon it was found that it did not meet the necessary capabilities to fulfill the required demand of information exchanges between on-road and on-board units. So the authors in [3] presented a new and improved Roadside ITS Data Bus (RIDB) system capable of providing high quality road characteristics. The backbone infrastructure for this type of networks must support fast and reliable vehicle communications.

The authors in [2] reinforce that in some cases, especially in V2V and V2R, it is supposed that the existent public communications backbone infrastructure is already enough, and that it meets the requirements of VCNs, which may not be the case, in particular when considering the challenges associated with vehicles’ mobility, and the still relatively high latency at IP Network and Transport Layers of current public wireless access technologies and the backbone infrastructure supporting them.

In order to make successful VCNs the development of a reliable road network backbone is a very important subject. The backbone infrastructure of roadside networks can be divided into three possible categories, more precisely can be based either on wired technologies, such as fiber optic, on wireless technologies, such as Wireless Local Area Network (WLAN), Long Term Evolution (LTE) or a hybrid of both:

1. Wired Backbone Infrastructure: The IEEE 802.11-based solutions for V2V and V2R exhibited a flat architecture, where the Access Point (AP)s and/or relays, typically implemented in RSUs, are directly connected to a non-specified backbone network through Local Area Network (LAN) con-

nections and to the Internet through gateways. In [2] the authors thought about the possibility to replace the road communication network systems with an “Integrated IP/optical network” allowing it to be also used as a backbone network platform for V2R communications. Since most highways generally do not integrate many communication network systems into their infrastructures, adopting the wired backbone solution may lead to a lot of wasted time and high capital costs [2].

2. **Wireless Backbone Infrastructure:** There are certain types of direct communications, like V2R and V2V that cannot handle voice communications and Internet services that other technologies such as cellular networks, i.e., Global System for Mobile Communications (GSM), General Packet Radio Service (GPRS), Universal Mobile Telecommunication System (UMTS), and LTE can provide. Due to the cellular networks restriction of the backbone architecture and the respective use of bandwidth the adoption of some services and the use of adequate technology is a challenge for the vehicular communications. So, the efficiency of known wireless solutions for backbone infrastructure is not enough to meet the requirements of vehicular communications, and the same applies to ITS services and wireless Internet services [2].
3. **Hybrid Backbone Infrastructure:** The authors in [2] could not find a solution for the cooperation of the previous types of communications, but state that their combination would be an added advantage, because they would provide a reliable communication for the roadside network.

The backbone infrastructure has to correspond to demands of several applications. These demands are mostly determined through Quality Of Service (QoS) parameters.

The ITS services need to be supported as a real-time packet transmission on the backbone network, due to the time critical event handling.

These types of architectures involve many applications. The safety applications require hard-real-time support, which means that a delay of up to only 100ms can be tolerated and that the backbone network must provide a bandwidth of up to 27Mbps, to allow adequate bandwidth for ITS services. Besides these hard-real-time applications, the mostly used are the soft-real-time applications, that usually are entertainment applications, which tolerate an E2E packet delay of a few seconds. However, real-time applications, like Voice over IP (VoIP), or video conferencing, tolerate a maximum delay of 150ms [2].

The authors in [3], presented and discussed an improved architecture model (Figure 2.1). That enhanced Open System Interconnection (OSI) model is composed by different layers, the first is the Physical combined with the Data Link Layer. The physical layer, or IDB-CAN is based on controller area network, the other layer, named IDB-1394, can provide support to the entertainment devices. However both layers can only keep up with short distance communications, which is not the case of roadside communications, so it is strictly necessary to re-ensure the existence of another physical layer for the RIDB. The authors in [3] discuss about the creation of a physical RIDB made possible through the com-

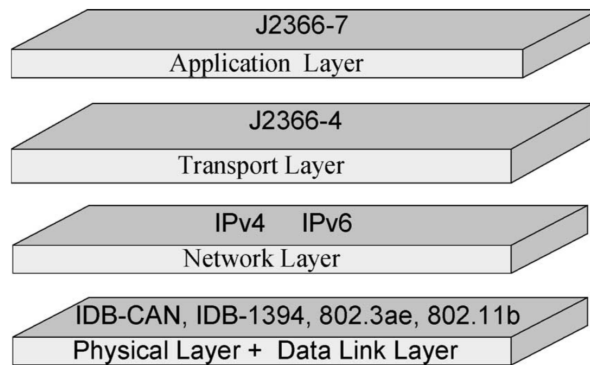


Figure 2.1: Enhanced OSI reference model of IDB. Source: [3].

combination of high-volume optical fiber and flexible wireless communication, eliminating the dependence of using only wired communications.

The IEEE 802.3 standards are potentially applicable to the physical layer of the RIDB, namely the IEEE 802.3ae which defines a version of Ethernet with a nominal data rate of 10 Gbps, for example. This new 10 Gbps Ethernet standards includes seven different media types for LAN and acWAN, that support respectively 10 km and 40 km links over a single-mode fiber. Additionally, these types of media are designed to operate on the OC-192/STM-64 synchronous optical network/synchronous digital hierarchy Synchronous Digital Hierarchy (SDH) equipment.

It is possible then to share some OSI layers, the in-vehicle IDB and the RIDB can do it, except the physical layer. In the network layer, it is possible to use both IPv4 and IPv6 and in the application layer the same service primitives and message sets can be adopted.

In roadside communications systems, both wired and wireless interfaces are needed to respond to different requests. Figure 2.2 shows the presence of the RIDB on a roadside communication system. Since it is necessary to have a broad bandwidth, the optical fibre acts as the backbone of the communication system. In this case the access points are placed at every 10 km or 20 km along the road. Wireless Base Station (BS) connect to the fibre through the AP present along the highway. Wireless Relay Stations (RS) are placed between the BS to complement lack of coverage between them.

Some equipment with high consumption of bandwidth, like video cameras for example, should always be connected to the AP and through coaxial cable or fiber because achieving high-quality video transmission over a multihop wireless network is usually expensive.

The authors in [3] present a demonstration system based on the IDB model, that is basically a SDH-based fiber communication system able to provide 10 Mb/s interfaces where both central station and base station are on the same LAN. In this system, the WLAN is composed by one BS and four RS, with their respective identification, so that is possible to perform an effective and improved routing performance.

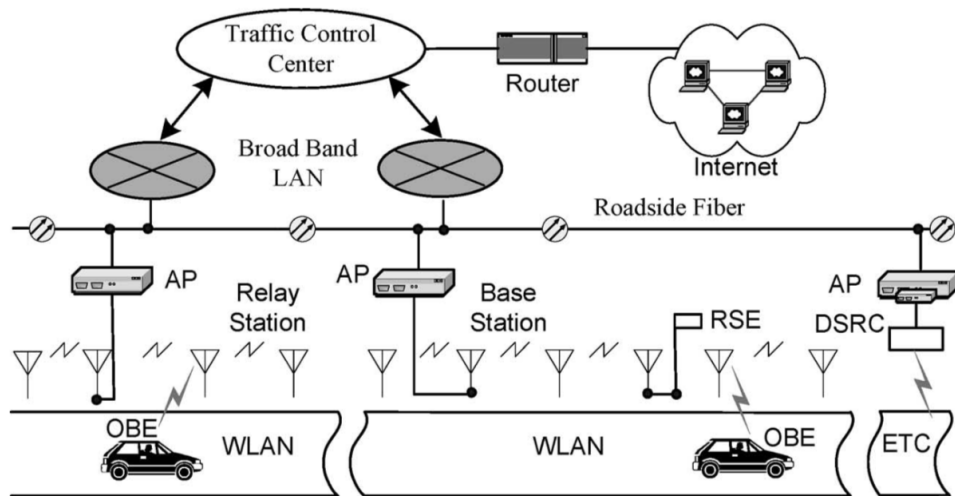


Figure 2.2: Roadside ITS Data Bus (RIDB) communication system. Source: [3]

Currently, long-range roadside wired communication systems are mainly based on optical fibers and coaxial cables. In some cases, long-distance point-to-point broadband radio links are also used [3].

2.2 Railway Communications Networks

Rail communications services have many needs and requirements that differ not only from those of commercial communications networks, but also from those of Public Protection and Disaster Relief (PPDR) systems (which normally include the Railway infrastructures as critical due to the significant impact they may have in case of disaster). Bandwidth requirements for operational rail communications (i.e., communications to ensure that rail functions safely and reliably) were traditionally quite low, with no expectation to grow significantly once implemented, and distinct from passenger entertainment services.

The early train-to-ground communication systems, such as Private Mobile Radio (PMR), were operated using Terrestrial Trunked Radio (TETRA) to carry mainly voice operations and narrow band data services. TETRA was operated in the range of 420-470 MHz band, at Ultra High Frequency (UHF). However, due to the limitation in bandwidth and the excessive use of UHF spectrum, PMR has migrated to higher bandwidth digital system GSM-Railway (GSM-R) [5].

The first train command-control and train-to-ground communication international standard was the European Rail Traffic Management System (ERTMS), that defines two very important elements, the European Train Control System (ETCS) and the GSM-R.

The ETCS is a digital railway control-command system, that includes the Automatic Train Protection (ATP) system, in-cab signaling, Driver Machine Interface (DMI), Braking curves and Train positioning, as illustrated in Figure 2.3.

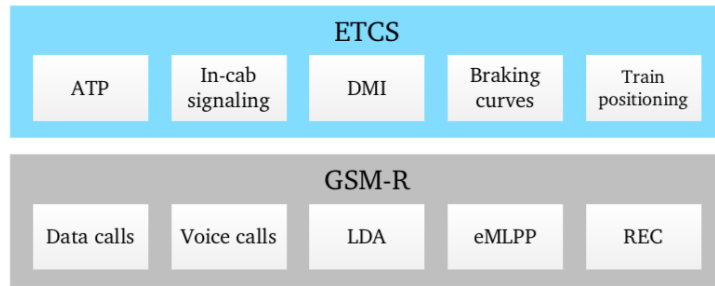


Figure 2.3: Elements of the European Rail Traffic Management System (ERTMS). Source: [6].

GSM-R is a special version of the GSM system, uses a specific frequency band around 800/900MHz just for railways, ensuring interoperability across country borders, and allowing real-time message exchange between computer-based Traffic Control Centre (TCC) and locomotive computerised OBU, and command-control systems that support and supervise train drivers, including features such as in-cab signalling and ATP.

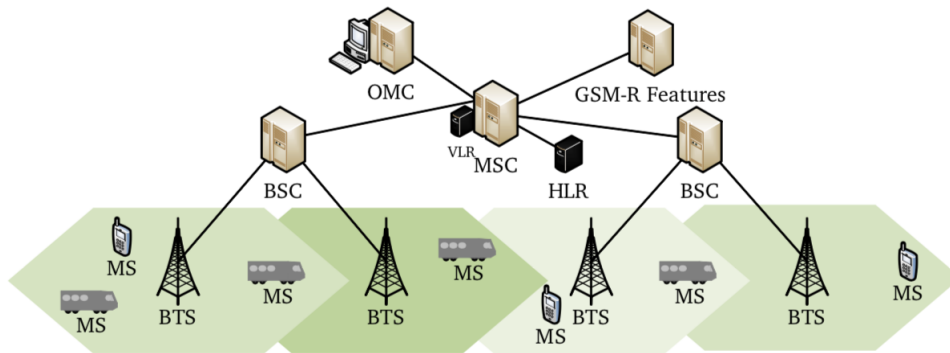


Figure 2.4: GSM-R simplified system architecture. Source: [6].

In addition, the frequency bands 873-876 MHz (uplink) and 918-921 MHz (downlink) are used as extenders for GSM-R at national level [7]. The GSM-R simplified system architecture is depicted for reference in Figure 2.4, and it is not significantly different from the standard commercial GSM architecture. Typically, GSM-R is implemented via dedicated BSs near the track, with distances between two adjacent BS of maximum 7-15 km to ensure higher availability and reliability. GSM-R must meet severe requirements for the availability and performance of radio services [8]. Table 2.1 lists the main QoS requirements for the existing GSM-R system.

Table 2.1: QoS parameters for GSM-R (ETCS). [9]

QoS Parameters	Demand Value
Call setup time	≤ 10s (100%)
Connection establish failure probability	≤ 1% (100%)
End-To-End delay	< 0.5s (99%)
Error rate	< 1%/h (100%)

ETCS and GSM-R form the ERTMS to carry both signaling information and voice communication, so GSM-R currently provides communication between the ETCS elements.

The system is divided into three subsystems: the Base Station Subsystem (BSS), the Network Switching Sub-system (NSS) and the Operational Support System (OSS). The Mobile Stations (MS) correspond to the terminal equipment, the radio cells, a.k.a. Base Transceiver Stations (BTS), provide wireless communication for the elements in the cell and are responsible for the communication between the MS and other elements of the system. Sets of BTSs are controlled by a Base Station Controller (BSC), and both combined form the BSS. The core network center in the NSS is the Mobile Switching Center (MSC), responsible for managing the switched communications in the system. The OSS is composed by the Operation and Management Centre (OMC) and the Network Management Center (NMC). Different from commercial GSM, the GSM-R includes functionalities and specific service elements, such as the Dispatcher System, that provides Voice Broadcast Services (VBS) and Voice Group Call Services (VGCS) to permit the distribution of a voice call from users or dispatchers to a user group located in defined geographical zones, and Rail Traffic Control System.

The specific features for GSM-R radio technology, allows the ETCS to offer data channels that connect trains and centralized control centers, and to make all rail voice communication a unified solution.

Several Rail communications services in Europe (and worldwide) using analogue radio systems, have evolved, since the year 1999, adopting digital communications, such as Ethernet for Fixed Telecommunications Network (FTN), and GSM-R for wireless mobile networks, as recommended in ERTMS, making GSM-R become a global standard and a very important element of the ERTMS, present all around the world.

However, even at that time the ERTMS selected GSM for its wireless mobile communications, it was clear that it could not fulfil all the requirements necessary for an efficient railway service, even with the specific extensions for railway (GSM-R) included in the system specification.

For example, the interference with public mobile networks in GSM-R is a reality, as both strive to get a good coverage along the rail tracks. This situation could be fixed if the public network operators would be able to avoid using frequency bands adjacent to those of GSM-R. Additionally, GSM-R cannot adapt to the demands of bandwidth variation for data-based applications, such as group voice communications, but since these calls are usually short and do not occur all at the same time, makes the problem of allocation of radio network resources much less dramatic, but still inefficient.

Another aspect, is related with limitations in the radio channel structure. In order to establish an ETCS connection it is necessary to occupy one of the few time-slots available on the GSM-R radio channel. Considering that each cell can only accommodate as many trains as the number of traffic channels it has available, and that some channels must be used for either voice communication or other procedures necessary for the good function of the radio system, the leftover channels, if any, can be used

for trains. This condition dramatically limits the number of trains that can operate in the same GSM-R cell, making the system ineffective, specially in areas with some concentration of train traffic, as each train must establish a continuous data connection with the Radio Block Centre (RBC), and each RBC connection needs to constantly occupy a time slot in the GSM-R radio channel. This telecommunication limitation, reducing the availability of the rail infrastructure, turned GSM-R not a reliable future railway technology [6, 10].

Nevertheless, the adoption of this standard has brought many advantages and features, like the improvement of interoperability, through the establishment of new common European standards, a new control system available for high speed trains, more efficiency and security, more track capacity. All this leads to a better performance of trains and less complexity of train driver work, due to the existence of the DMI interface, standardized for all European trains, the establishment of a single radio communication system, the appearance of a Railway Emergency Call (REC) to allow a reliable and fast communication, and cost reduction due to the presence of more suppliers in the market. These are some examples of what standards like ERTMS can offer and the fact that all of these features are available in one international system brings not just advantages but also makes it an unique and special standard.

Although GSM-R was developed specifically for railway communication, because it beared some advantages such as the support for fast handover with train speeds up to 500 km/h, it is now turning into an out-dated technology [11], especially due to its performance of data transmission mechanisms, and many more different types of problems were also identified such as inefficient use of network resources, insufficient capacity and limited support for broadband data communication. Alternative packet-oriented technologies would bring higher capacity and efficiency [8].

As such, the fast technological evolution of telecommunications standards, turned GSM-R an obsolete solution right by 2025, with the end of support of GSM-R already announced from 2030 onwards.

The successor of GSM-R is most likely the LTE-Railway (LTE-R), an adaptation of the 4G LTE communications network, dedicated for railway services, enabling high-speed wireless voice and data communications inside trains, from the train to the ground, and from train to train. This will be explored and explained in more detail in Section 3.2. LTE-R when compared to GSM-R offers several advantages, such as very low latency, much higher data capacity (broadband) and high security. LTE-R, as basically, an Internet Protocol based wireless infrastructure can also support passenger information applications, Closed-Circuit Television (CCTV), traffic management, Group Communications (with Dispatch features, and also train-to-train communications) ticketing and other services [10].

As presented in section 2.2, the ERTMS has two important elements in its structure, ETCS and GSM-R, as illustrated in fig. 2.3.

The GSM-R network serves as a data carrier for the ETCS, used for sending and delivering information to trains, in three levels of operation: in ETCS-1 the GSM-R is just used for voice communication,

in ETCS-2 and 3 the system is used mainly for data transmission and this is where it gains importance, since the train travels at a speed of up to 350 km/h.

The ETCS is a system based in communication, responsible for managing the movement of trains, replacing the way the driver receives system signals. It can react in case of an event of potential danger by helping the train driver, so it is basically a “backup” help to the driver. It also has a classic signaling system, that consists in a few steps to ensure that the journey can proceed safely and this is done through the communication between the Traffic Management System and the interlocking engine [6].

With this system it is possible to ensure that two routes do not exist at the same time, but the system is very dependent of human interaction and therefore much more conducive to the existence of possible errors.

Being such a complex system the ETCS has two main features:

1. In-cab signalling, usually mandatory for trains running faster than 160km/h. It assists the driver in terms of interpretation and action of the signals received, thereby reducing any error the driver may cause from erroneous deduction of a received signal, which ultimately improves efficiency and safety of train operation. This feature is provided by the DMI.
2. ATP, responsible for checking if the train driver work is being done as it should be. This is usually done through the on-board computer that knows all the mandatory characteristics of the train, like velocity and safe stops locations.

The ETCS is divided in two parts as depicted in Figure 2.5. The “on board” part that just consists of the OBU, and the “trackside” part constituted by the RBC and Eurobalises [6, 10].

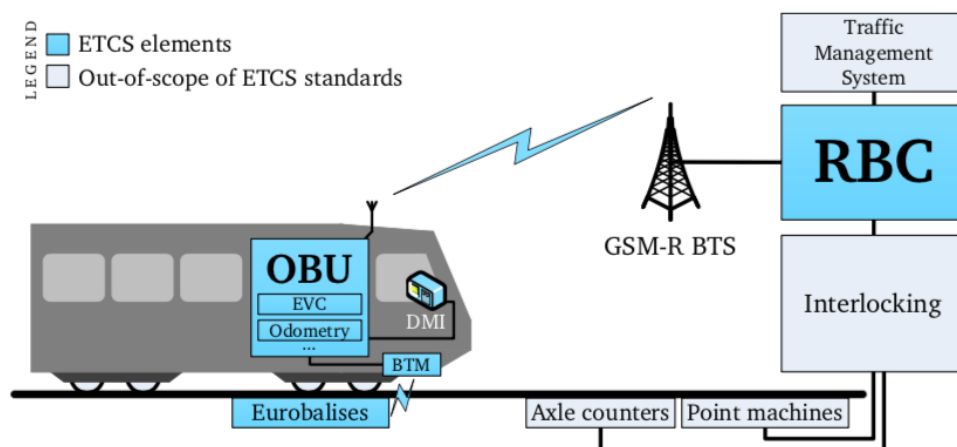


Figure 2.5: Schematic overview of ETCS Level 2 architecture. Source: [6].

The RBC is basically a centralized computer responsible for all trains inside a certain area, it has interfaces to the Traffic Management System, the Interlocking and the OBU, but only the OBU interface

is standardized, so it can be used to connect devices from different vendors. Using these three interfaces the RBC can get a much wider view of the area in charge.

The RBC is responsible for the movement of the trains, it has a data vector that defines the speed limits of the track section. This vector is essentially a digital message named Movement Authorities (MA).

The other element of the trackside part, named Eurobalises, is a electronic element placed between the rails that sends low-bit rate signals to trains that pass through, allowing also the ETCS to know some crucial information, like the location of the Eurobalise or which is the RBC in charge of the area.

The OBU is a set of train systems present in trains and checks if the train is moving according to what was imposed by the MA. It has other parts on its structure, as illustrated in Figure 2.5, such as the DMI, the Balise Transmission Module (BTM), the Odometry system, a GSM-R radio module and the European Vital Computer (EVC)—that is responsible for the logic part of the system, while the others just provide interfaces and support functions to the system.

Since the RBC issues the MA messages, it is up to the receiving EVC to interpret them and calculate safe braking curves. This message processing is done via GSM-R [6].

1. Train-to-ground data communication for ETCS level 2 and 3.
2. Railways need a different kind of voice communication, so the GSM-R offers specific features.

To transmit all data from the BTSs to the respective BSC, and from each BSC to the NSS core, a transport infrastructure typically composed by SDH transmission systems, connected by Fiber Optic Cables and/or Microwave radio links is required in order to implement and support the GSM-R network and all of its required features.

2.3 Software Defined Networks

Software Defined Networking (SDN) brought the idea of decoupling the Control Plane from the Data Plane and allowing the control of the network to be made through a logically centralized controller.

SDN offers three fundamental attributes which make great contributions and not just on security level: the logically centralized intelligence, programmability and abstraction of the whole network [12].

SDN is all about making an “abstraction” of the network, allowing to see it as a “whole system”, allowing to make optimal decisions (on routes, on flows, on policies, etc.). The essential concept in SDN is moving the control software off the network device into a compute resource located centrally, enabling the efficient network traffic control while the data plane is used for data forwarding, so the SDN significantly simplifies network management, and offers a programmable and flexible network architecture [13].

SDN has three areas of focus: abstractions of distributed state, forwarding, and configuration.

- *Distributed state abstraction*: allows the network to have a certain level of abstraction—i.e., shields the programmer from the reality of a network full of machines, each one with its own state, working in a collaborative environment and allowing a global network view.
- *Forwarding abstraction*: allows the programmer to specify forwarding behaviours without the knowledge of vendor-specific hardware.
- *Configuration abstraction*: allows the goals of the overall network to be achieved without losing the details of how to implement them in the physical network.

Besides all of this, in reality, it is believed that the SDN model was born with the appearance of a communication protocol, named OpenFlow, which is a protocol specification that describes the communication between OpenFlow-capable switches and a SDN controller, in secure and encrypted sessions via Secure Sockets Layer (SSL) [14].

The OpenFlow specification establishes the line bounded by the protocol that must be used between the controller and the switch, the so called Southbound Application Programming Interface (API).

Apart from data and control planes, SDN also has a third plane called application plane, which consists of third party network services and applications that gain access to the Control Plane via the so called Northbound API.

In large scale networks, it is common to implement distributed controllers, each with its own domain of forwarding devices. The Eastbound/Westbound API of SDN allows controllers to share information of their own domains, for a consistent global view of the entire network (Figure 2.6) [15].

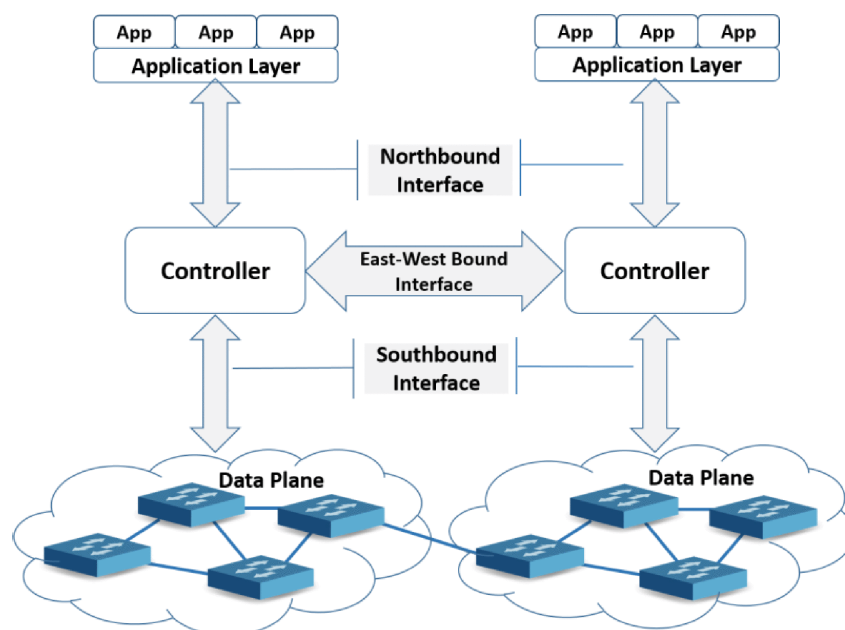


Figure 2.6: SDN Interfaces. Source: [15]

One of the big advantages of a SDN-based network is the fact that it can take advantage of two important features: the variety of applications capable of providing a good network programming capability, and the possibility of decoupling the control of the infrastructure through the previously mentioned controllers [16].

Figure 2.7, provides an overview of the SDN architecture, with the three distinct layers: application layer (end-user applications), control layer (logically centralised control functionality) and infrastructure layer (network elements) [17].

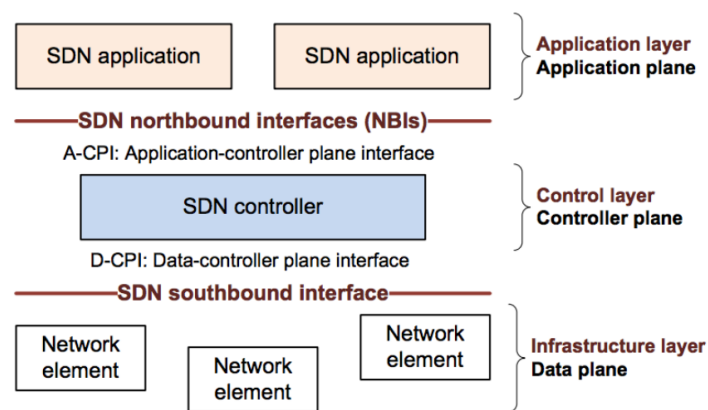


Figure 2.7: Software Defined Networking (SDN) architecture. Source: [17]

2.3.1 Southbound Protocols

To make possible the communication between the controller and the network devices several protocols are being developed.

One is the OpenFlow protocol that defines the communication between the controller and the OpenFlow-based switch and the specific messages and formats exchanged between controller (control plane) and device (data plane), in order to obtain and manage their resources. They do this by exchanging OpenFlow specific messages, named controller-to-switch messages [16].

In a typical SDN network, OpenFlow switches connect to each other and with end-user devices, usually sources or destinations of the flows present in the network. Every OpenFlow switch has its own implemented Flow tables with entries (Matching functions and Actions) for packet processing.

The SDN controller modifies the entries in the Flow tables of the switches, so that all incoming packets in a port matching a condition are associated with an action, e.g., to be dropped, or to be forwarded by the switch to an out port. If the switch has no matching rule for a certain packet, then sends it to the controller allowing it to program a new rule in the switch Flow tables. The most basic Flow programming defines, modifies and deletes flows. When the controller defines a Flow, it is providing the

switch the information (Matching rules and Actions) to treat incoming packets for that Flow.

There are three fundamental options, as illustrated in Figure 2.8 for the disposition of the arriving packet: **A-forward the packet out a local port**, **B-drop the packet** or **C-pass the packet to the controller for exception handling**, and so, every time the controller receives an asynchronous message it knows that the packet does not match any entry in the Flow tables [14, 18].

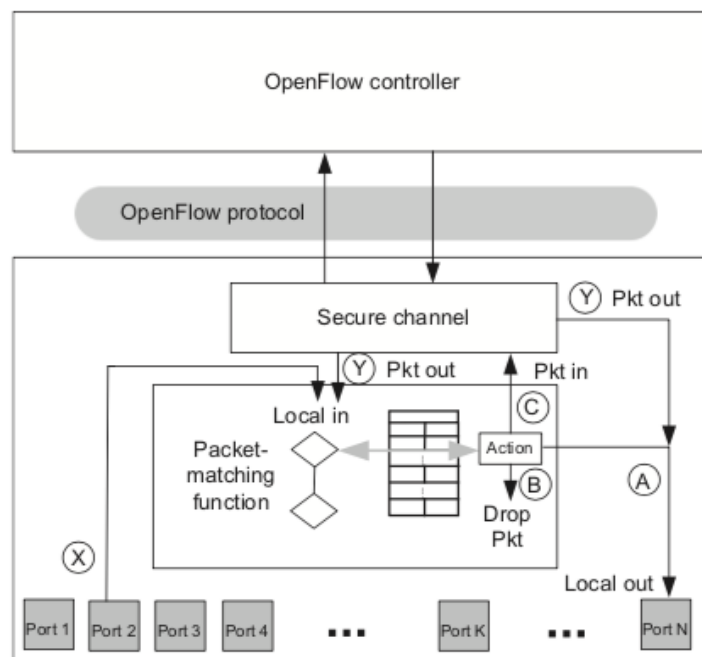


Figure 2.8: Types of treatment for incoming packets of OpenFlow protocol. Source: [19].

The possibilities for flow treatment have grown more complex as the OpenFlow protocol evolved (from initial version 1.0 to current stable version 1.3, and with version 1.5 already under development).

Another protocol is the Network Configuration Protocol (NETCONF), emerged as a substitute of Simple Network Management Protocol (SNMP), since SNMP is used to monitor networked devices and not to configure them.

Compared to OpenFlow, NETCONF has more components and is able to separate configuration, set on the device to perform a specific action, and operational data, set by the device as a result of an event. If the device has features like Access Control Lists (ACLs), data models to configure QoS or static routes the NETCONF is capable of implementing them. One advantage of the use of NETCONF is the possibility to invoke certain procedures/functions on a device remotely, something that SNMP is not capable to do [19].

2.3.2 SDN Controllers

As described earlier, the controller—by having a view of the network as a whole—can track the network topology, learning it through the existence of switches (SDN devices) and end-user devices. It can also implement policy decisions and control all SDN devices present in the infrastructure through the Northbound API, the interface responsible for the communication between the controller and the applications.

The core functions of the controller are device and topology discovery and tracking, flow management, device management and statistics tracking [19].

The number of possibilities to choose a SDN controller is immense, the market availability is huge and suppliers are increasingly focused on presenting SDN solutions in their portfolios. There are both open source, usually developed in C-language or Java, or commercial solutions of SDN controllers [20]. The most reputable open source controllers are listed in Table 2.2. The currently most prominent are Open Network Operating System (ONOS) and OpenDaylight, with Java in their core and have acquired high versatility.

ONOS was created as a “Carrier-grade” controller, to work in high speed and large scale networks of providers, and is a project of the Open Networking Foundation (ONF).

OpenDaylight was embraced as a project of the Linux Foundation, has a modular architecture, and has a clear focus on network programmability, being now able to support different types of applications, besides being more suitable to Datacenter environments, and one of the first to have an Internet of Things (IoT) domain.

The other three controllers worth mentioning are Floodlight, POX and Ryu.

FloodLight is an enterprise-class, Apache-licensed open-source controller, that consists of a set of modules, where each provides a service to the other modules and to the control logic application.

The POX controller has its code based in Python, is used to explore SDN debugging, network virtualization, controller design, and programming models.

Lastly, Ryu (Japanese for “Flow”) is a component-based SDN controller and has a set of pre-defined components that can be modified, extended, and composed for creating a customized controller application.

These latest three controllers are best suited for research, development and campus applications, which is not the case of our project [21].

Considering that the types of networks we will address in this project, i.e., large scale networks, having in their basis the high velocity of trains or vehicles, with critical requirements, the ONOS or the OpenDaylight controllers seem to be better suited, not just for their possible domains of application but also due to the support of the documentation, that is high in both cases.

Table 2.2: Comparison of the most popular open source SDN controllers [20].

	Programming Language	Documentation	Application Domain
ONOS	Java	High	Datacenter, WAN and Transport
OpenDaylight	Java	High	Datacenter
Floodlight	Java	Good	Campus
POX	Python	Poor	Campus
Ryu	Python	Medium	Campus

2.3.3 Network Functions Virtualization

Two important technologies that can perfectly work together and complement each other are SDN and NFV, allowing networks to reach greater flexibility and scalability. SDN and NFV are independent, meaning that with their partnership they can complement each other. Unlike SDN, NFV uses IT virtualisation technology to decouple network functions from dedicated, often proprietary, hardware devices [17]. With the use of NFV is now possible to reduce implementation and support costs, due to the fact that with the use of virtualized elements the need to acquire physical equipment or specific services and its insertion in the network was obliterated. SDN moved from supporting packet routing to implementing more complex functions that are better suited to roadside/railway networks, and through this partnership between SDN and NFV it is now possible to reach the desired efficiency, flexibility and scalability, keeping the simplicity already involved in this type of services [22].

2.3.4 Controller clustering

A Single Point of Failure (SPOF) consists of a system failure in terms of design, configuration or implementation, which ends up constituting a huge risk as it can lead to a situation in which the entire system may stop working due to a malfunction or a fault.

In SDN the controller is considered the potential SPOF of the network. Therefore, it is very important to ensure that if the controller fails, the network remains functional and that, consequently, no equipment changes its behavior due to its failure. And how is this done? Through the use of a second controller or even separating the network in different domains where each one has its own controller.

This is so-called a distributed ONOS implementation, with the use of a ONOS cluster, meaning that more than one controller will be used to avoid the system failure as a whole. It is then possible to configure several characteristics of the network, like using a load balancing strategy to make a distributed decision, reducing the decision delay caused by network transmission [23], or even to prevent against Distributed-Denial-of-Service (DDoS) attacks [24].

In this work the implementation of ONOS will be made using a single controller, a unique centralized component will provide the global view of the network. Of course, this will bring several challenges and a probable failure of the system, so in section 5.4 an approach of using more than one controller and

configure a distributed controller topology will be presented, and the importance/advantage of its use will be presented.

2.4 SDN applications

All the alternative technologies discussed in previous sections, like LTE, LTE-R or GSM-R, have a rigid and complicated architecture that makes hard to keep up with the various demands in the vehicular environment. For that reason SDN was thought to be an adequate technology to be applied in vehicular networks in order to provide a better service quality, due to its capability of a flexible management, better service capabilities, support of bigger data rates and lower delays [1].

2.4.1 For Roadside Networks

The programmable and flexible management and the centralized knowledge across the network of VANET is possible due to the appearance of this type of networks based in SDN. The authors in [16] proposed a dynamic scheme on the Northbound Interface (NBI) for the VANET based in SDN, named BENBI, where the network resources are controlled dynamically through the application of broadcast encryption, instead of using the typical method and altering the source codes of the controller.

SDN has gained more and more recognition due to all of its features, and that makes it one of the best future technologies applicable to this type of roadside networks. With its use it is possible to adapt certain kinds of services to topology changes, leading to a better route planning due to a previous network-wide knowledge. Since SDN enables the vision of the network as a whole it is possible to make wise, coordinated and great decisions regarding vehicles in that type of environment.

The principle of SDN, as said before, is to have a controller that is responsible for the rest of the network among other features, so in the VANET case the objective is to have a dynamic cooperation between SDN applications and VANET devices that in this case, behave like mini distributed controllers, in order to get real-time resources, like road conditions, for example. Through the unbundling of the control and data planes, network intelligence and state can be “centralized” and the network infrastructure will no more interfere with the applications.

The proposed access control scheme, BENBI [16], becomes important not just because it enables the flexible network management but also because it can dynamically control the access to network resources and only the chosen and qualified applications can access those resources, available in controllers. Since there are a few messages exchanged between applications and controllers it is important to keep the integrity of messages and that is why BENBI reinforces the security via the *NBI* interface.

There are five entities that make this use of SDN possible in VANET, i.e., the SDN applications that provide great network services, the SDN Controller with full knowledge of the network and with the ability

to control all the network and program network policies or configurations on the networking devices, the RSU with OpenFlow features and the RSU controller (RSUC) responsible for grouping the RSUs and act like a controller inside the main the controller, so it just owns local network knowledge, a BS with a function similar to RSU, and elements (in this case vehicles) acting like SDN infrastructures [14, 16].

The authors in [14] present a SDN based Vehicular ad-hoc Network (SDVN) whose objective is to simplify network management in roadside networks. Its main goal is to unburden the overhead of the management of the controller and the channel dedicated to it. In this approach the controller, by proving the abstraction between the VANET applications and the network infrastructure, can support flexible and fast network configurations, ending up in saving a lot of time and resources, and leading to the improvement of user experience. Although this approach seems perfect there are some challenges that this type of networks needs to face. Since the high velocity of vehicles is a reality, the constant topology change causes instability on the communication in the channels used and a likely lack of network information to keep up with the global view that consequently leads to system delays. Attacks to the SDVNs become also a huge concern, because there a lot of diversity in the types of networks and elements used, and so, this types of networks cannot be prone to a possible accident or failure. Additionally, the existence of heterogeneous Vehicle-to-everything (V2X) networks leads to a necessity of efficient mechanisms to enable an efficient communication between them [14].

The authors in [1], also presented an SDVN solution. They called it SINET-V, an instance of smart collaborative networking, smartly schedule ubiquitous network resources that fully meet diversified vehicular demands, capable of building function slices through crowd collaborations, and very similar to what simple SDN can do.

Soft-Defined Heterogeneous Vehicular Networks are also mentioned in literature, as is the case in [25] where the authors discuss about an SDN-based heterogeneous vehicular network solution, focusing on a approach that integrates not only vehicle-to-vehicle communication, but also vehicle-to-infrastructure and vehicle-to-cloud communication. In the vehicular scenario, the data plane contains all vehicles, RSU, and base stations, abstracted as SDN switches. The control plane retains the status (vehicle location, velocity, and network connectivity) of all switches and is responsible for making packet forwarding decisions. In this approach the main challenge is the prediction of vehicles trajectory, having in mind that they are in constant movement and not always connected to the control plane. With this predicted trajectory the controller can construct a network topology, installing a Flow table that can support not only routing, but also adaptive protocol deployment.

In its turn, the authors in [26] present a cloud-RAN architecture to a Heterogeneous Vehicular Network (HetVNETs) solution, by integrating different access networks including Dedicated Short Range Communications (DSRC) and LTE to meet the requirements of ITS services.

Another approach is made in [27], where the authors analyse SDN in vehicular networks through a

construction similar to a game, which consists on the interaction between the controller and the vehicles, having in concern that the bandwidth needed for each case is different, reaching an equilibrium analysis.

2.4.2 Application of SDN in a Roadside Network - Example

The idea for the work described in this section, came up during a conversation with Thales Portugal with the objective of determining ***“How can technologies, like SDN, improve our projects in terms of cost, time and resources effectively?”***

Since the main goal of Thales Portugal is to efficiently design, develop, and deploy systems and services in the areas of defence, security and transportation, it became the perfect combination.

As two real projects with Infraestruturas de Portugal (IP) and Thales Portugal were taking place, then the idea turned up: use them as an experiment on the application of SDN technologies and study all the benefits and the limitations.

In the first project for a Roadside Network, the objective of Infraestruturas de Portugal was to replace existing (in obsolescence) solutions for the Roadside infrastructure they explore, with modern solutions. The existing infrastructure was composed of a primary SDH transport network, that consists of optical rings supported by 1 or 2 fiber pairs along the tracks, and two transport subnetworks, one for video surveillance—which supports video transmission from control points to primary network rings using optical fiber Ethernet technology over optical ring connections—and one for communication regarding Variable Message Panels (PMV), Automatic Vehicle Counting (CAV) and Mobile Stations (MS), formed by Plesiochronous Digital Hierarchy (PDH) equipment, supported by 1 optical fiber along the track, connecting to the Control Center through the primary transport network, thus constituting a secondary ring topology.

In a previous phase, the aged network equipment was replaced by new devices from Huawei, and using Huawei protocols responsible for doing the forwarding of packets. The equipment was deployed on the network in ring topology, with each ring having a restricted number of routers (due to the limitation caused by the used protocol).

All services were also separated through the implementation of Virtual Local Area Networks (VLANs) and all equipment was configured in the same way, meaning that manual configuration was an observable reality when developing the embedded solution, carried out as efficiently as possible, that is, as autonomously and quickly as desirable.

Of course, the human factor was still present, and so the system became more prone to errors. In the current architecture of the project it is possible to see several rings connected to a central switch in a dual stack topology, solving the redundancy problem.

So the goal of the current project phase would be to implement equipment supporting new solutions, such as SDN and NFV in order to provide zero-touch provisioning, centralized network control and

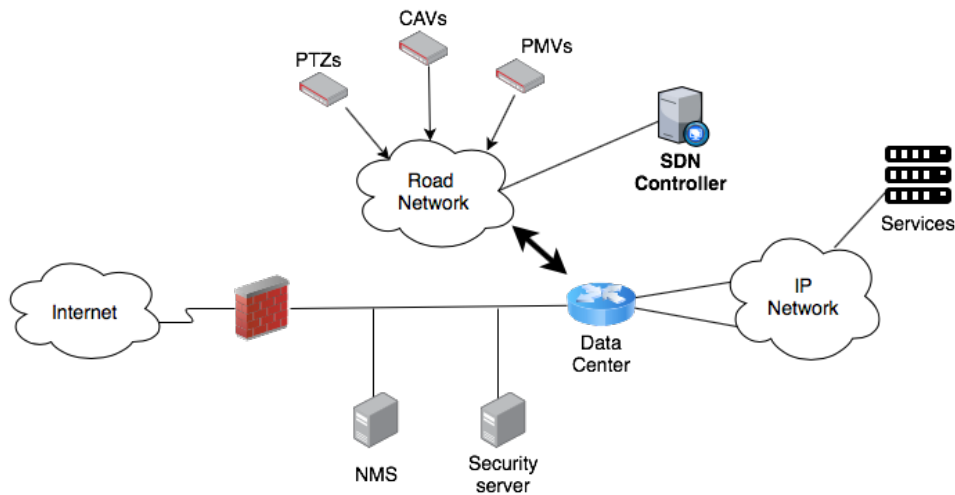


Figure 2.9: Proposed architecture.

abstraction, making it easier to manage and faster to deploy. The generic idea for the architecture with SDN is illustrated in Figure 2.9.

This envisioned solution has a lot of potential in similar projects, since it is a roadside system, and the requirements and implementation methods are already well known and well established, such as ensuring distinct QoS profiles for different services, redundancy levels equal to or higher than those that already existed, a latency of less than 25ms and a jitter of less than 5ms, for example.

Considering that both roadside and railway systems have a fixed network (normally TCP/IP based) in their backbone and combine mobile network and/or wireless infrastructure, the idea was to check where (in part or in the whole architecture) and how SDN could be applied.

Having in mind that the railway system has specific elements and is a much critical system, the Railway Project was considered to be a lot more interesting to study than the Roadside Project, that is why the focus of this work ended up more on the implementation of SDN technologies for Railway types of infrastructures.

3

Proposed Architecture

Contents

3.1 Railway Architecture Design	29
3.2 SDN implementation in a Railway Network	31

This chapter describes a possible architecture for the railway communications network of a specific project of Infraestruturas de Portugal (IP) being deployed by Thales Portugal. The proposed solution is only presented in high level of detail, due to confidentiality requirements, and focused on the key functionalities it is supposed to achieve.

3.1 Railway Architecture Design

In order to provide flexibility and adaptability to the railway communications network, the application level should be independent of the access technologies in use (3G, 4G, 5G, etc.) to allow different types of development, and so, it is crucial to use resilient architectures that apply spatial and temporal redundancy techniques. However, redundancy without path diversity does not provide the needed resiliency for facing correlated communication errors. This motivates the necessity of novel traffic engineering techniques to reserve network resources and make data travel faster with low data rates [28]. The SDN-based architecture proposed for the Infraestruturas de Portugal (IP) railway communications network and solution, must therefore respect, and should improve the key requirements, in order to present all the benefits that a SDN implementation in a network can bring. The details of the Infraestruturas de Portugal (IP) railway communications network are confidential, and so, some of those details cannot be disclosed in this work (due to a request of the end customer). The high-level detail for the network is nevertheless quite complex and is divided into several sections so that the project can be split up into phases for its implementation in the best possible way.

Thus, for this work, one section of the project was chosen, the “Ovar-Gaia” section, as it presents more details of its implementation and also because it has a greater number of equipment, thus making possible to demonstrate the network dimension with greater precision. The high-level detail of that network section is illustrated in Figure 3.1.

The main scope of the project corresponds to the following:

- *Signaling*: command, control and supervision systems for railway traffic including automation of level crossings.
- *Telecommunications*: telecommunications systems to support railway operations.
- *Speed Control*: systems that transmit information about the status of the signaling and the speed allowed by the infrastructure to the trains.
- *Centralized Command at Operational Command Centers (CCOs)*: command and remote control of signaling installations, from the CCOs.

Explaining the current network architecture a little better, there are two network infrastructures for communications. One consists of a ring of industrial switches from Hirschmann (HR), Model RS30.

The other consisting of Alcatel-Lucent Omniswitch (ALU-OS) and Nokia switches and routers, all hybrid (supporting SDN mode). The current architecture considers the creation of a Multi Protocol Label Switching (MPLS) network that will have to be present at all railway stations on this network section, providing Ethernet interfaces for critical services, which, for reasons of confidentiality, the connections to the network cannot be shown. The Nokia IP/MPLS technology, where Virtual Private LAN Service (VPLS), Ethernet 'Pipe' Services (e-PIPE) (an implementation of Ethernet Virtual Leased Lines)) and Circuit 'Pipe' Emulation Services (c-PIPE) services are available, among others, is also responsible for Routing (Layer 3) on the network. In turn, the IP technology network of the manufacturer ALU-OS implements the functionality of Routing (Layer 3) and Switching (Layer 2), while the manufacturer Hirschmann only implements the functionality Switching (Layer 2).

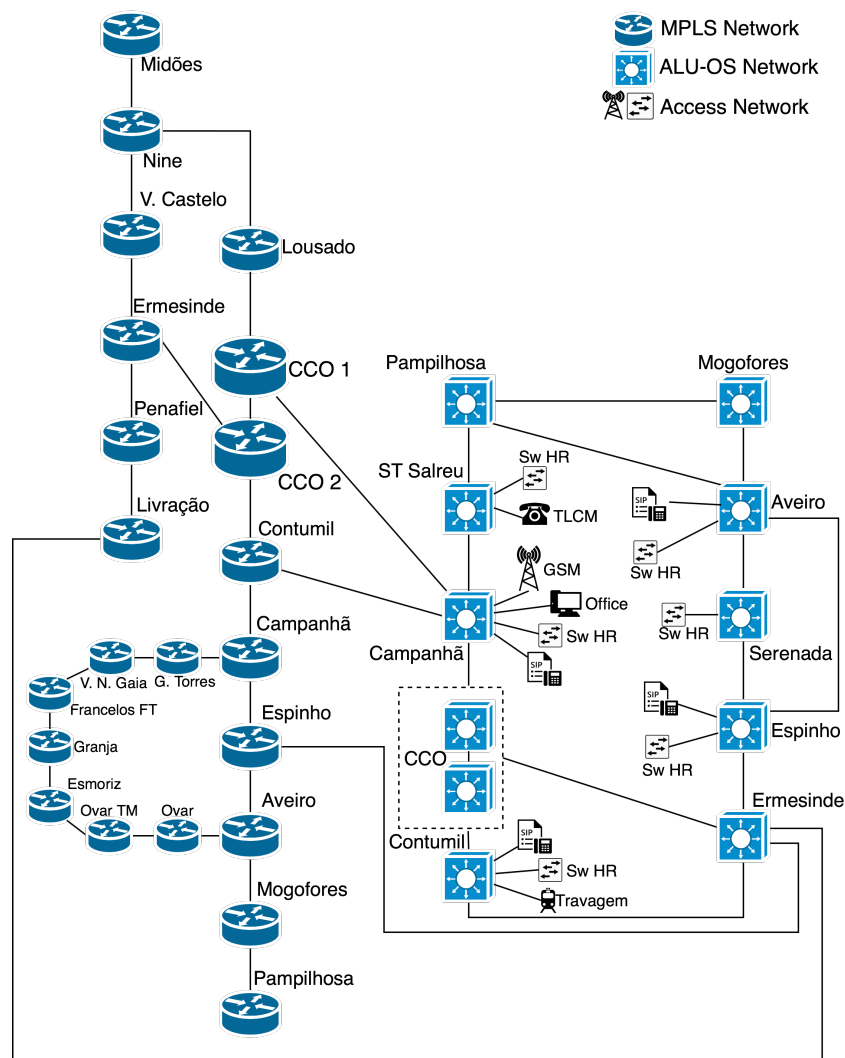


Figure 3.1: The Ovar-Gaia section high-level network topology of the IP Railway Project.

In Figure 3.1 the Hirschmann industrial switches are represented as **Sw HR** and only the connec-

tions of the Alcatel-Lucent (ALU) switches to the HR are shown, the HR ring in the infrastructure is not represented. Some services are shown as elements connected to the ALU equipments, like links to signalling, or of the breaking system or to the radio network, in this case GSM, there are also links to the office, or to the communication system present in trains.

The type of connection between each node uses optical fiber. For the closest stations, SFP multi-mode optical fiber links are used. In the case of longer distances, LX, LH40 or LH70 fiber links are used, more precisely monomode optical fiber for distance less than 40km, up to 40km or up to 70km.

As previously mentioned, three types of equipment are used in this project, Nokia, Alcatel-Lucent Omniswitch and Hirschmann. Hirschmann switches are 'access switches' having no SDN compatibility.

In the case of Nokia, the currently deployed model 7705 SAR-8 routers, although compatible with SDN, in the current implementation of the project they are used with Path Computation Element Communication Protocol (PCEP) protocol. Therefore, they will be replaced by models responsible for Layer 2/Layer 3 switching, more precisely the Alcatel-Lucent Omniswitch (ALU-OS).

In the case of Alcatel-Lucent Omniswitch (ALU-OS) switches, two models are used, the OS6860E-E24 and OS6860E-U28. These two models will be the ones selected (configuration and characteristics) to be 'emulated' on the simulated network of our work. Table 3.1 describes the main characteristics of those equipments.

Table 3.1: ALU-OS main characteristics.

Model	Gigabit copper and fiber ports	Uplinks	Throughput
OS6860E-24	24 RJ45	4 x 1/10G SFP+	160.9 Mpps
OS6860E-U28	28 100/1000 BaseX, SFP	4 x 1/10G SFP+	172.6 Mpps

Both equipments support 1 Gbps distribution links and interconnects of 10 Gbps. This means that the model OS6860E-24 will be used for the access network, and for the stations where it is required higher data rates and stronger link capacities the model OS6860E-U28 will be used. The 10 Gbps ports will be used to connect the data center to the Campanhã CCO. These parameters will be used in the simulation in order to replicate the project characteristics.

3.2 SDN implementation in a Railway Network

Currently, in the IP infrastructure for the railway communications network, to configure a set of devices, a technician has to manually and locally configure each unit/module board. This is a very time consuming procedure and synchronization between devices has to be configured. In a scenario like the one presented in this work, the technician can operate remotely on the devices that he has to configure, but for devices that he is not allowed to intervene, the access is blocked.

This section describes the proposed train to ground network architecture complemented with SDN technology. SDN and/or NFV provide new ways to conceive networks and services and, in the railway domain, the progressive introduction of softwarization of services can make next generation railway systems highly flexible and re-configurable [29].

On the one hand, end-hosts are connected to SDN switches which have multiple network interfaces in order to achieve E2E spatial and temporal redundancy. On the other hand, SDN is used in the network, where a centralized SDN controller runs a path-computing application to make forwarding decisions and determine the flow entries for each switch. Other approaches can be considered, orchestration at the infrastructure level can be used in the case of a single provider when specific service requests arrive, like managing and supervising certain activities present in ERTMS, like the Radio Block Center and Command and Control System.

The idea is to deploy at each place general purpose IT resources that can host a number of Virtual Machines (VMs) dynamically created by an orchestrator, placed at a Central Point, to support the required services, e.g., Virtual Network Functions (VNFs). In turn, the SDN network equipment (e.g., a Layer 2 switch) can be programmed by the SDN controller to direct to each VM only the specific traffic flows. This approach can put together two technologies that are crucial for network softwarization, SDN and NFV [29].

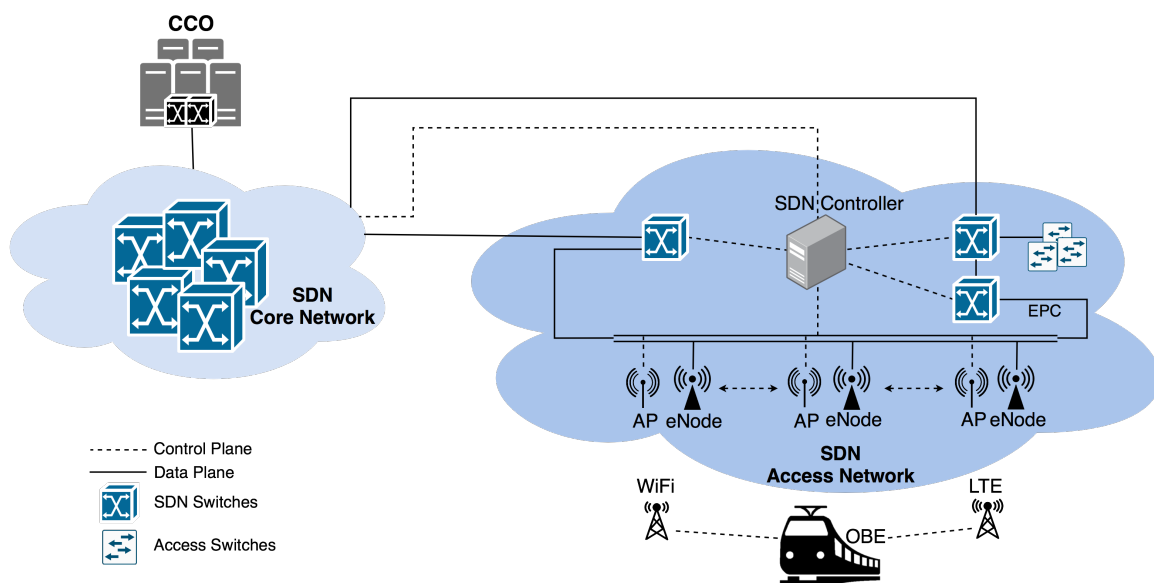


Figure 3.2: High-level SDN architecture for the railway communications network.

The architecture can be logically divided into three architectural components: core network, access network, and access functions. In the next chapter (4) this solution is going to be studied and with the use of network simulator, a small part of the Infraestruturas de Portugal (IP) project (Figure 3.1) will be placed as a use case, to prove the advantages of SDN usage in this type of infrastructures.

3.2.1 Core Network

The core network is a dedicated backbone network managed by the railway operator. The SDN controller is a logical centralised entity (typically implemented in a resilient cluster configuration), which connects in-band or out-of-band with the network elements of the infrastructure, i.e., each SDN-based device (OpenFlow (OF) switches) as represented in Figure 3.2. Additionally, OF is set as the southbound interface.

3.2.2 Access Network

The railway operator is also responsible for managing the access network. The access network is composed by fixed and wireless networks (WiFi and LTE-R), as represented in Figure 3.2. The fixed access network correspond to (OF switches) connecting the LANs of railway stations and other locations of the organization, connection to fixed trackside elements (signaling, Eurobalises, Interlocking, etc.), and also to non SDN supported wireless elements—the LTE-R Evolved Packet Entities (EPC) and eNodeBs (eNBs). The wireless network elements (OF-APs and eNB) provide connectivity to the On Board Equipment (OBE), which is in charge of offering connectivity between the train and the application services that must be present on board.

- *WiFi*: This network is fully SDN-based, having several OF-APs that are connected to an OF switch that concentrates all the traffic coming from the OF-APs. The use of additional aggregation OF switches depends on the number of deployed OF-APs.
- *LTE-R*: This is the chosen technology due to the advantage of high uplink/downlink data rate for mobile broadband services [15]. It consists of EPC entities and some eNBs that are connected to each other through OF switches. The EPC entities and the eNBs have no OF support, which means that they are treated as end-hosts by the SDN controller.

3.2.3 Access Functions

The access functionality relies on a locally developed SDN applications. The applications run on top of the controller, and are focused on reducing the delay of the communication and providing path diversity. Also OpenFlow specifications provide basic reliability functionalities (i.e., fast fail over groups, flow entries priority), that can be effectively exploited to support effective recovery procedures.

4

Demonstration

Contents

4.1 Implementation	37
4.2 Proof of concept scenario	38
4.3 Use cases	44

A Demonstration can be considered the first step for evaluating the proposed solution. For that goal, during the viability phase we proposed to perform a proof of concept of the architecture in order to either confirm and prove that the objectives are achievable, and the benefits it may bring to real projects of Thales Portugal, or just the opposite, i.e., that, however effective the studied solution might be, it would not eventually bring benefits when compared the solutions already implemented.

For the Demonstration phase a possible scenario was designed in order to show what are the advantages of using SDN in railway networks, with the objective of evaluating the performance of the proposed architecture when certain critical network events occur, typical of railway networks, like the crash of a link in the core network or even a link overload.

4.1 Implementation

The main goals of using SDN in a network is the fact that it brings **programmability**, meaning that it provides links between the application control and network control layers to optimize application performance, increase visibility, and **application awareness**, allowing also the network to automatically react to the dynamic requirements of workloads, and providing a **global control view**. Additionally, the network control systems have a global view of current network conditions in order to improve local actions of individual network nodes on how to treat traffic streams of a particular application. The initial idea for the Demonstration was to use real equipment in a test bed in order to study the possibility and advantage of applying SDN in these types of architectures with the equipment used in the project. For that purpose, Thales Portugal provided one EdgeCore switch, model 5712-54x, and two ALU-OS, model OS6250-24E.

4.1.1 ALU-OS

The two ALU-OS models OS6250 made available [30], were not identical to the ones being deployed in the project, despite being of the same brand, they did not, unfortunately, have the same capabilities. From our research, and comparing the features and capabilities we believe that the models being deployed have a lot more compatibility with more recent versions of OpenFlow and SDN than the ones provided, as the following

- *Outdated:* the provided switch models [30] are no longer in the market, and were replaced by other recent models with improved features.
- *Lack of support:* since those models were discontinued Alcatel does not even provide corrections or firmware upgrades to those switches anymore.

- *Compatibility*: an old versions of the operating system (firmware) was implemented on switches, but not possible to be upgraded, and so the OpenFlow version of that firmware was not anymore able to correctly connect with the SDN controller.
- *No out-of-band management interface*: contrary to the other models being deployed, the switch OS6250, do not have an Ethernet Management Port in order to provide out-of-band management. Usually, the SDN controller makes use of that interface to communicate with the OpenFlow switch. For example, the OFTP_HELLO handshake message used by both the switch and the controller to identify and negotiate the OpenFlow version supported by both the devices, failed with error.

4.1.2 EdgeCore switch

The EdgeCore AS5712-54X switch provided, had for characteristics, 48-Port 10 Gbps Small Form-Factor Pluggable (SFP)+ with 6x40 Gbps Quad Small Form-factor Pluggable (QSFP)+ uplinks, that builds an Open Network Install Environment (ONIE)-compatible installer and a switch Open Network Linux (ONL) image which contains a complete Debian distribution with added drivers and configuration for running on bare metal switches (<https://github.com/opencomputeproject/OpenNetworkLinux>). The product has completed the end of life process effective on April 1st, 2020 and the replacement model is AS5812-54X. So the version of the switch kernel was no longer compatible with the OF version available for ONL switches, and so, no upgrades were available in these switches.

Due to the fact that the company did not had any other equipments available to be used in the demonstration, we decided to replaced the test bed environment with a virtualized approach, using an SDN network emulation running virtual OpenFlow switches configured with the essential similar capabilities of the real switches being deployed in the project.

4.2 Proof of concept scenario

Figure 4.1 represents the demonstration test bed used for the tests. In that scenario the wireless networks were recreated using a SDN-based WiFi network emulator (<https://mininet-wifi.github.io>) that can emulate and perform functions of SDN and OF-AP- WiFi. The controller is ONOS, that provides scalability, high performance and high availability, specially for the core network. In the data plane, the core network consists of four OF switches (CCO1, Contumil connected to Campanhã and Espinho, Livração connected to Ermesinde), previously represented in the diagram of the project (Figure 3.1) as MPLS routers. Using SDN, the MPLS can be easily replaced by flows, as will be explained later.

The access network or the equipment present in physical stations, is represented by ten OF switches. The Campanhã site, is considered to be a central interconnect point of the network, providing also

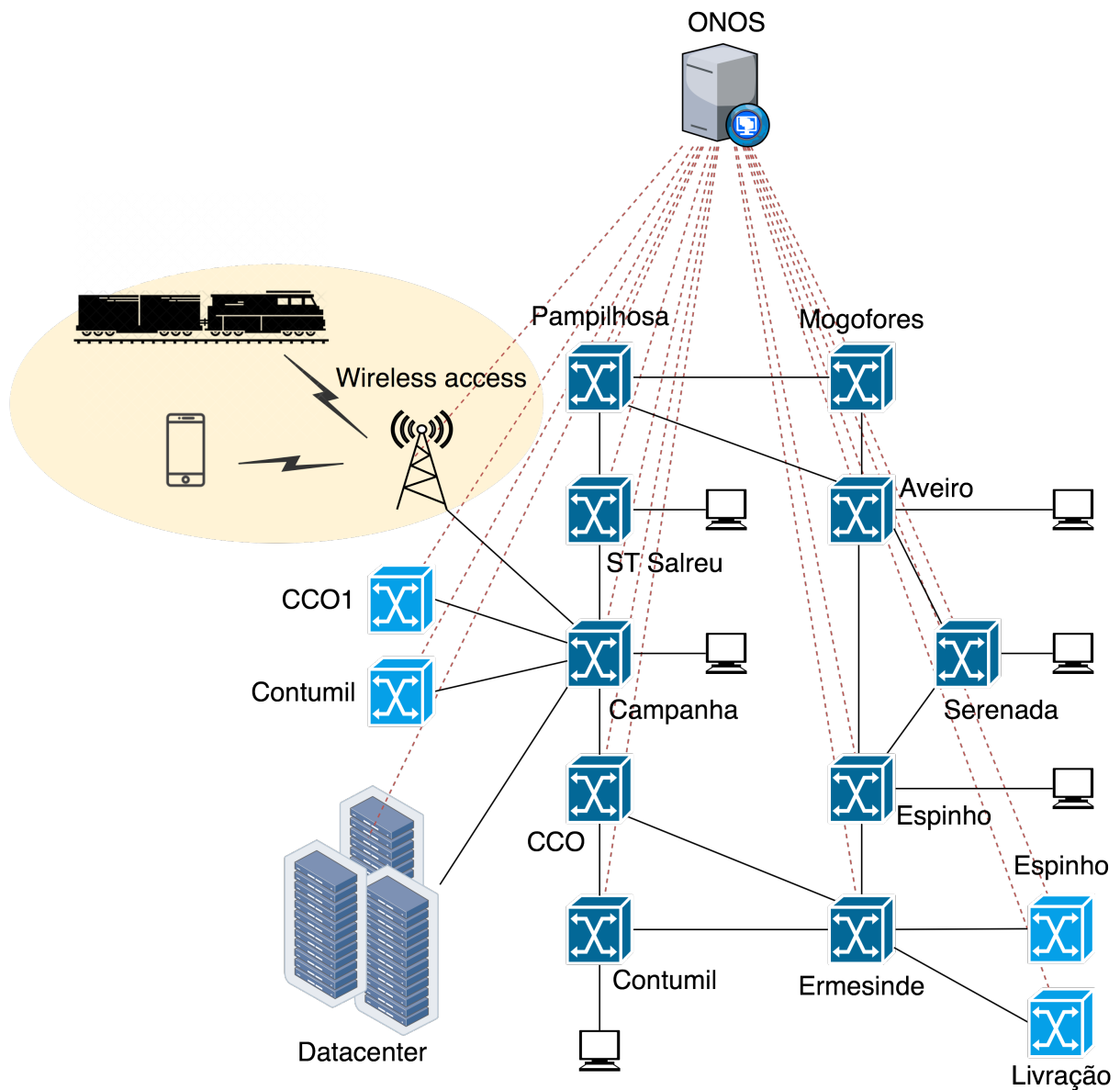


Figure 4.1: Diagram of the Demonstrator Test Bed of the IP Railway project.

connectivity to a Datacenter, designed with a spine-leaf topology, with Top of Rack (TOR) switches on the leafs and multiple switches on the spine. The Campanhã site also interconnects the wireless APs and eNBs, serving mobile users (inside trains and railway stations), as well as trains OBUS. Some switches have hosts connected so it is possible to simulate railway components connected to each railway station's LAN. Both OF switches and OF-APs-eNBs will be controlled by the ONOS controller and provide an OpenFlow compatible implementation. The wireless (WiFi/LTE) "stations" (mobile, trains) and fixed hosts in LANs have no OF support, so the controller will only see them as end hosts.

4.2.1 SDN Controller

The SDN controller is ONOS version 2.3.0 (<https://wiki.onosproject.org/display/ONOS/Downloads>), created in a build VM in VirtualBox named *onos* running Ubuntu 18.04, 64-bit Desktop. ONOS is a leading open source SDN controller for building next-generation SDN/NFV solutions, allowing to create new network applications without the need to alter the data plane systems [31]. ONOS provides some analogous types of functionality, including APIs and abstractions, resource allocation, and permissions, as well as user-facing software such as a Command Line Interface (CLI), a Graphical User Interface (GUI), and system applications. It manages the entire network rather than a single device, which can dramatically simplify management, configuration, and deployment of new software, hardware and services. The ONOS kernel and core services, as well as ONOS applications, are written in Java as bundles that are loaded into the Karaf OSGi container, a component system for Java that allows modules to be installed and run dynamically in a single java virtual machine [32]. The architecture of the ONOS is presented in Figure 4.2.

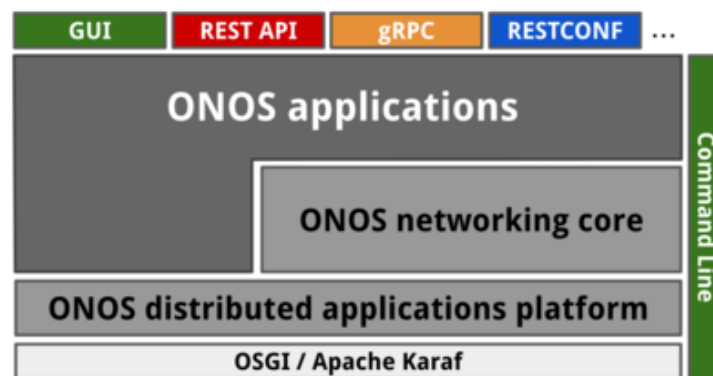


Figure 4.2: ONOS architecture.

The controller provides applications with a number of high-level abstractions, through which the applications can learn about the state of the network and through which they can control the flow of traffic in the network. The flow objective is a device-centric abstraction that allows applications to direct flow of traffic through a specific device without the need to be aware of the device table pipeline. Similarly, the concept of “intent”—for Intent-based networking (IBN)—is a network-centric abstraction that incorporates Artificial Intelligence (AI), network orchestration and Machine Learning (ML) to give application programmers the ability to control the network by specifying what they desire to accomplish rather than specifying how they want to accomplish it. Applications (core extensions) can be loaded and unloaded dynamically, via REpresentational State Transfer (REST) API or GUI, and without the need to restart the cluster or its individual nodes.

Some applications are already installed in ONOS, being one of them the **Reactive Forwarding**, which is crucial to ensure communication by installing flows in response to every packet arriving at

the controller (not having a *match* at a switch ingress). The **Reactive Forwarding** feature, although pre-installed, is not activated by default.

The topology for the Demonstration scenarios, as already explained in Section 4.2, and described in Figure 4.1 was created in Mininet, a simulation framework described in Section 4.2.2, using ONOS as a remote controller. Figure 4.3 corresponds to that topology, but now “discovered” and presented in the ONOS GUI.

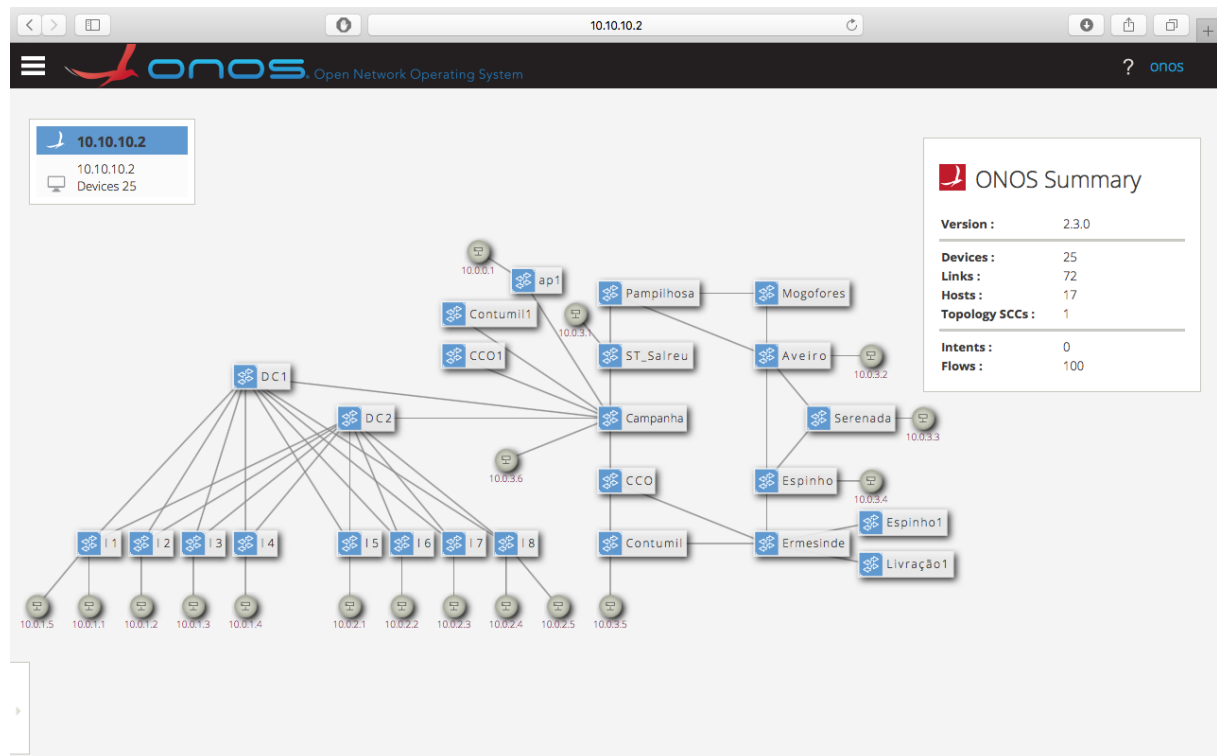


Figure 4.3: ONOS GUI, showing the Topology of the Demonstration scenario.

4.2.2 Simulation Framework

Mininet is a network emulator that can emulate and perform the functions of virtual hosts, switches, controllers, and links on a single host (physical or virtual) [33]. This is accomplished by the implementation of Linux namespaces. Network namespaces are containers for network state. Each network namespace has its own interfaces, ports, and routing tables and provides exclusive processes [34]. During the execution of an emulated network packets are transferred through “virtual Ethernet interfaces” that can be configured. The Switches in Mininet support the OpenFlow protocol (from version 1.0 up to 1.3) for high flexibility in routing customization in SDN architectures. A host in Mininet is simply a shell process (e.g. bash) moved into its own network namespace with the unshare system call. With Mininet it is possible to create topologies using simple Python scripts, allowing fast and easy development of networks.

The version of Mininet used in this work (Mininet-WiFi [35]), is a version that has been extended to support Wireless stations, emulating the attributes of mobile/wireless hosts and radio BSs (i.e., position, coverage and movement relative to the APs/BSs). A wireless station (host) is implemented by configuring a wireless interface in order to allow the host to connect to an OF switch with AP capabilities. The virtualized APs are created through *hostapd3* (Host Access Point Daemon). The current implementation supports the configuration of many AP features, such as the **SSID**, channel, mode, password, cryptography, etc. Finally, The wireless channel is emulated by using Linux Traffic Control tools.

The whole Demonstration scenario topology was created in Python, added in Appendix A, and initialized as illustrated in Figure 4.4.

```
wifi@wifi-virtualbox:~/Desktop$ sudo python IP.py
*** Adding controller
*** Add switches/APs
*** Add hosts/stations
*** Configuring Propagation Model
*** Configuring wifi nodes
*** sta1-wlan0: signal range of 20m requires tx power equal to 5dBm.
*** ap1-wlan1: signal range of 30m requires tx power equal to 13dBm.
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Add links
*** Starting network
*** Starting controllers
*** Starting switches/APs
*** Post configure nodes
*** Starting CLI:
mininet-wifi> █
```

Figure 4.4: Mininet-wifi topology initialization for the Demonstration scenario.

The Mininet network emulator includes MiniEdit, a simple GUI editor for Mininet. MiniEdit is an experimental tool created to demonstrate how Mininet can be extended. This feature allows the creation of topologies and the generation of python scripts in a graphical canvas. Figure 4.5 illustrates the Demonstration scenario built in MiniEdit.

Since locations of nodes in space is an important aspect of wireless networks, Mininet-WiFi provides a graphical display (Figure 4.6) showing locations of wireless nodes and “radio coverage” of APs in a graph. The graph can be created by calling its method in the Mininet-WiFi Python API.

OpenFlow Switches: The ingress of a packet to an OpenFlow switch starts the matching of the packet header fields from the first Flow table entry, progressing to the next table in pipeline processing until the match is found. If a **match** is found then the specific **actions** are executed, as defined by ONOS controller, and counters get updated. In an OpenFlow-based centrally controlled network, when anytime first event occurs (a packet arrives) to a switch the controller resolves the address and immediately installs Flow entries in the Flow tables of the switch with some timeout time called idle timeout [36].

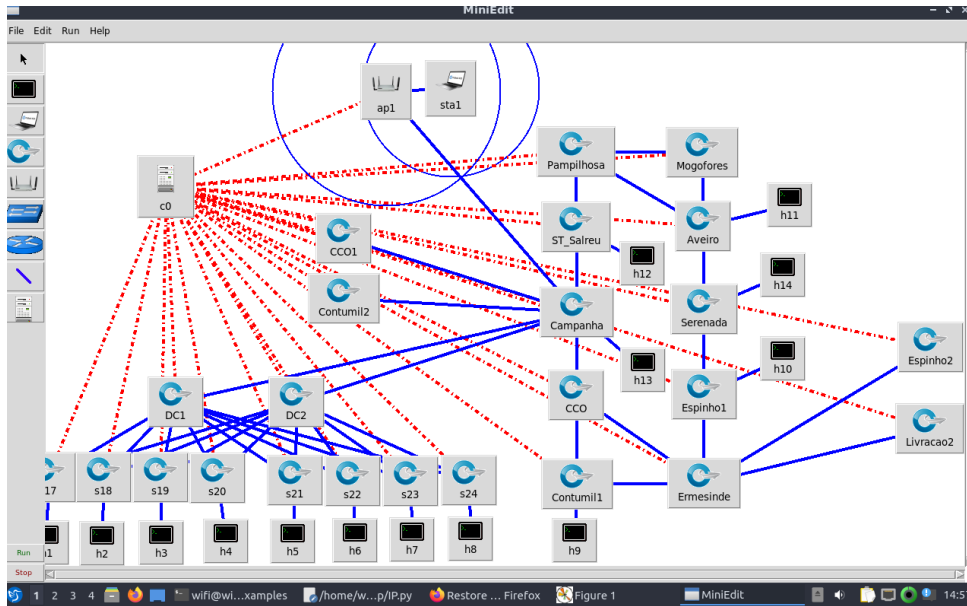


Figure 4.5: Mininet-wifi topology creation of Demonstration scenario in MiniEdit.

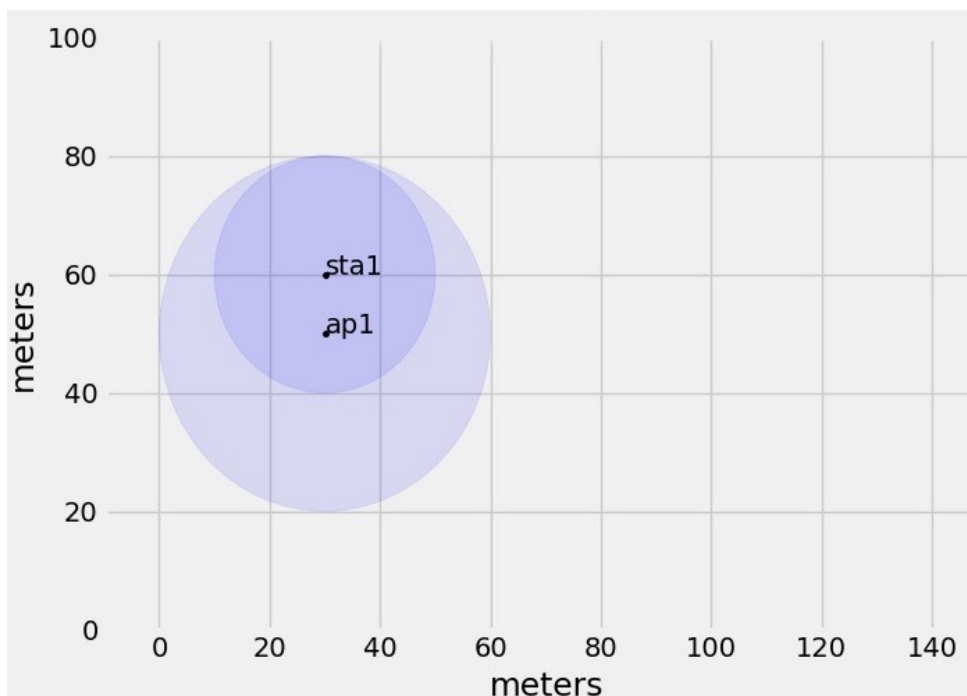


Figure 4.6: Mininet-wifi wireless coverage graphical representation in real-time.

OpenFlow Access Points: Each AP has only one interface towards the network infrastructure. By default, wireless stations associated with an AP connect in infrastructure mode, and so, the wireless traffic between stations must pass through the AP. The AP works similarly to a switch in standard Mininet emulation, and so we expect to see OpenFlow messages exchanged between the AP and

the controller whenever the access point sees traffic for which it does not already have Flow entries installed. The AP in Mininet-WiFi supports OpenvSwitch as the kernel implementation. Each AP can be managed in Mininet-WiFi, as any standard WiFi AP, as illustrated in Figure 4.7.

```
ap1-wlan1 IEEE 802.11 Mode:Master Tx-Power=9 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Power Management:on
```

Figure 4.7: iwconfig command in **ap1**.

Hosts: Mininet can create kernel or user-space OpenFlow switches, controllers to control the switches, and hosts to communicate over the emulated network. Mininet connects switches and hosts using virtual Ethernet pairs.

Wireless stations: Mininet-WiFi provides several utilities to manage the wireless elements, such as **iw**, **iwconfig** (illustrated in Figure 4.8 for **sta1**) and **wpa** supplicant. The first two are used for interface configuration and for getting information from wireless interfaces and the last one is used with **Hostapd**, in order to support Wi-Fi Protected Access (WPA), among other things. Another fundamental utility is Traffic Control (TC), which is a user-space utility program used to configure the Linux kernel packet scheduler, responsible for controlling the rate, delay, latency and loss, applying these attributes in virtual interfaces of stations, representing with higher fidelity the behavior of a real world network.

Stations are devices that connect to the APs through authentication and association. In our implementation, each station has one wireless card (sta1-wlan0), and so Mininet hosts can connect to the AP attached wireless stations.

```
mininet-wifi> sta1 iwconfig
sta1-wlan0 IEEE 802.11 ESSID:"ap1-ssid"
Mode:Managed Frequency:2.412 GHz Access Point: 02:00:00:00:04:00
Bit Rate:54 Mb/s Tx-Power=1 dBm
Retry short limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:on
Link Quality=68/70 Signal level=-42 dBm
Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
Tx excessive retries:0 Invalid misc:12 Missed beacon:0
```

Figure 4.8: Output of **iwconfig** command in **sta1**.

4.3 Use cases

Railway communication network infrastructures have characteristic applications or network aspects that need to meet very specific requirements. For the Demonstration, the tests will focus on the links capac-

ities, by trying to replicate the specifications of the equipments being deployed in the real project, and in terms of link and connection types.

Three typical Use Case scenarios were designed in order to evaluate the behaviour and performance of the proposed architecture: Video Surveillance, Link Failures in the Core network, and Fault Tolerance of the network.

4.3.1 MPLS-like features in SDN

The IP project, as previously described, relies on MPLS in the core routers. This component is present in all railway stations and is responsible for providing “Ethernet pipes” and “virtual circuits” for critical services across the network. MPLS is considered an architectural framework responsible for decoupling transport from services, this is done through the encapsulation of instructions in packets headers. This instructions take a crucial part in the packet forwarding since they are represented as labels and MPLS can stack them and transform a set of instructions into a sequence, so it operates similarly to switches and routers, sitting between layers 2 and 3, and so the forwarding decisions are then made through the uses of packet-forwarding technology via those labels [37].

The main features of using MPLS in a network are:

- Disassociation control plane from forwarding plane.
- Decoupling service from transport.
- Differentiation of overlay from underlay.
- Layered architecture approach, with a feature-rich edge [38] and a fast transport core.
- Support multi-tenancy and multi-service with the building of overlay networks at the edge.
- Decreasing the forwarding state on the core.
- Advanced packet steering by either signaling forwarding paths and/or by stacking instructions on packet headers.

As previously said SDN is represented as the key concept of separating the control and forwarding planes and the fact of centralization of control, either logical or physical, which is another fundamental architectural ingredient [37]. This types of railway infrastructures can be considered as Wide Area Network (WAN), a telecommunications network that extends over a large geographic area for the primary purpose of computer networking. So the need to introduce SDN in WAN, besides being the main focus of this work is increasingly recognized and adopted in large infrastructures. An Software-Defined WAN (SD-WAN) virtualizes the network functions that run on the network infrastructure so they can run as software on hardware. In this respect, it becomes obvious that SD-WAN functionalities can perfectly

replace the MPLS role in most WAN environments. This replacement brings many advantages, but in spite of that, in some specific cases, it can also have some disadvantages, which will be presented below.

4.3.2 Advantages of SD-WAN Over MPLS

There are many advantages of using MPLS in a network. One is how it can very reliably deliver packets to destinations with a minimal processing overhead. In railway networks MPLS can generally offer a high quality of service when it comes to avoiding packet loss and keeping traffic flowing. This reliability is especially essential to maintain the quality of real-time protocols, which is possible due to the fact of using labels for packet forwarding, and also on the ability to assign different traffic priorities to improve network performance, as packets travel only along the paths to which they were directed. All this together end up bringing the possibility to perform a better traffic predictability within the network.

One downside of MPLS is bandwidth cost. Another one is that a MPLS network does not offer built-in data protection, and if incorrectly implemented, it can open the network to vulnerabilities [39].

SD-WAN offers considerable benefits when compared with traditional MPLS networks. With SD-WAN, geographic boundaries become less relevant, and the key benefits, such as visibility, scalability, performance, and control are strengthened.

Unlike MPLS, SD-WAN comes with no bandwidth penalties, and usually has a better performance. Companies can upgrade easily by adding new links, with no changes necessary to the infrastructure or network. One of the main advantages of SD-WAN is the ability to cost-effectively configure network links as desirable, according to content type or priority. It represents much less cost in terms of both internet broadband and cellular than MPLS, so customers can choose those links instead of the expensive MPLS network for certain types of lower-priority traffic [40].

Security, policy and orchestration is what today's organizations value the most in their network architectures. SD-WAN security covers those bases by unifying secure connectivity approaches, with all devices and endpoints fully authenticated, thanks to a scalable key-exchange functionality and software-defined security. In the SD-WAN architecture, an organization benefits from end-to-end encryption across the entire network.

Compared to MPLS, SD-WAN can be less expensive, more secure, and provide higher performance. MPLS can have increased bandwidth costs, while SD-WAN protects the network from vulnerabilities that MPLS cannot. Finally, we can conclude that SD-WAN offers better visibility, availability, enhanced performance, and more freedom of action than an implementation with MPLS [41].

4.3.3 Use Case Scenarios

The following Use Case scenarios were designed in order to evaluate the behaviour and performance of the proposed architecture.

Video Surveillance: IP-based CCTV systems are the defining factor in the success of modern train surveillance systems. In addition to ensuring passenger safety, these systems have expanded their scope and are now providing mission-critical information that will increase operational efficiency. Normally the type of security cameras used need the image quality to be quite good, in order to ensure reliability of the captured scene imaging. So the camera must allow a video stream with a High Definition (HD) resolution of 1920×1080 pixels and coded in High Efficiency Video Coding (H.265/HEVC) [42]. For the test scenarios we will use the VLC media player [43], a free and open source framework that plays most multimedia files, and various streaming protocols, in order to measure the instant data rate of the communication, allowing to set up multimedia streams and taking advantage of the SDN configuration. The protocol used for delivering video is Real Time Streaming Protocol (RTSP)/Real-time Transport Protocol (RTP), a stateful presentation-layer protocol for real-time audio/video transport in IP networks with very low latency (typically less than 2 seconds) that lets end users command media servers via pause and play capabilities. RTSP uses by default Transmission Control Protocol (TCP) to maintain an end-to-end connection [44]. TCP provides reliable, ordered, and error-checked delivery of a stream of octets between applications running on hosts communicating via an IP network, and it was chosen due to the fact that video streaming in video surveillance should not allow loss of packets or high delays in the transmitted streams, and must be as reliable as possible [45]. Video surveillance services are usually present in railway stations, that is why a streaming video server was placed in our scenario running in **sta1** and a video client in **h11**. The VLC video server will run in **sta1**, and the RTSP/RTP stream can then be captured by the VLC player and saved for further analysis.

Link Failure: As described in Table 2.1, the maximum E2E delay of a user data block in GSM-R is 500 ms. So, in case of a link failure in the network the tolerated path recovery delay is bounded by that value. As such, and for our SDN test scenario, it is expected that the flow recovery delays would be even much lower than 500 ms. The Ping tool will be used to test if a particular host is reachable across the network. The Ping tool, using *ICMP*, can measure the time it takes for packets to be sent from the local host to a destination computer and back. The Ping tool measures and records the round-trip time of the packet and any losses along the way. This will be used to gather link and flows statistics. When a link failure occurs in the network the controller is responsible for the network re-convergence (i.e., restore the Flows via a different path). The failure recovery process in SDN starts with a detection of the failed links. This detection is crucial to the overall process and takes

usually a very small amount of time. By having a global view of the network, the controller can use several methods for the failure identification process, and that is why modern-day networks leverage the SDN centralized control and flexibility of managing the data plane in the link failure recovery. [46] The OpenFlow implementations use messages for detection of specific network events. A heartbeat message is exchanged between the nodes at regular time intervals, which determines the status of the network. The links and network communication is checked by the rate of exchange of HELLO packets. If a node does not receive a HELLO packet within the regular time interval of 16 ± 8 ms, the controller is notified about the failed link. If a node does not receive a response within the time of 50–150 ms, the link is considered disconnected [47]. It would be therefore quite fast the link failure recovery procedure, by installing the Flow rules in alternative switches in the new computed path, so that the Flow is quickly recovered.

Fault tolerance network: It is very important for railway communications networks to be designed with fault tolerance in perspective and redundant mechanisms in order to prevent crashes of critical services or longer unavailability. In networks with certain specifications that provide critical services like the ones studied in this work, a failure or a high delay may constitute a massive problem, being therefore necessary to ensure a redundant architecture, which additional or alternate instances of network devices, equipment and communication links. With the use of SDN, the control of the networking devices is performed by the controller, and so, it is also very important to ensure that the controller is not the Single Point of Failure of the infrastructure. Controllers such as ONOS are designed as a distributed SDN operating system, and configured typically as high-availability clusters, providing fault-tolerance and resilience when individual controller instances fail [46].

5

Evaluation

Contents

5.1 Use Case: Video surveillance	51
5.2 Use Case: Recovery from link failure	53
5.3 Intent-based networking	55
5.4 Use Case: Controller failure	58

This chapter provides the results of the evaluation performed with the proposed architecture, using the emulation scenario previously described, for the three Use Cases of Video surveillance, recovery from Link Failure and Fault Tolerance (in terms of Controller failure).

5.1 Use Case: Video surveillance

A video with HD resolution of 1920x1080 pixels encoded in H.265 (Figure 5.1) was streamed from **sta1**, by means of a VLC video server.



Figure 5.1: Video stream characteristics on the server side.

The VLC client in **h11** requests the network stream from the VLC server at **sta1**. It can be observed in Figure 5.2, that we have the hosts **xterm** terminals in the upper part of the figure and the VLC player of each host in the bottom part. The VLC GUI window in the bottom left of the figure corresponds to the streaming server (without image as it is not rendering the video but streaming it in the network). The VLC GUI window in the bottom right of the figure shows the rendered video stream, played in real-time [48].

The input bitrate refers to the data that is being passed into VLC, and the content bitrate refers to the amount of data being displayed on the screen at any given point. After analysing the traffic generated during the streaming, we measure an uplink data rate of around 634 kbps and no frames were lost, as illustrated in Figure 5.3.

The network streaming between hosts is only possible because the controller allows this type of communication. For that a bidirectional Flow was created between **sta1** and **h11**.

Figure 5.4 shows the flows installed by the ONOS controller, having **sta1** as source (vlc server) and **h11** as destination (vlc client).

The video despite being HD 1080p was encoded with low bitrate (less than 1 Mbps), which is quite normal in surveillance cameras, as the stream has only temporal variations (people entering) and not very spatial (the background is almost always the same). The most important aspect was the fact that the transmission was made without loss and without apparent delay.

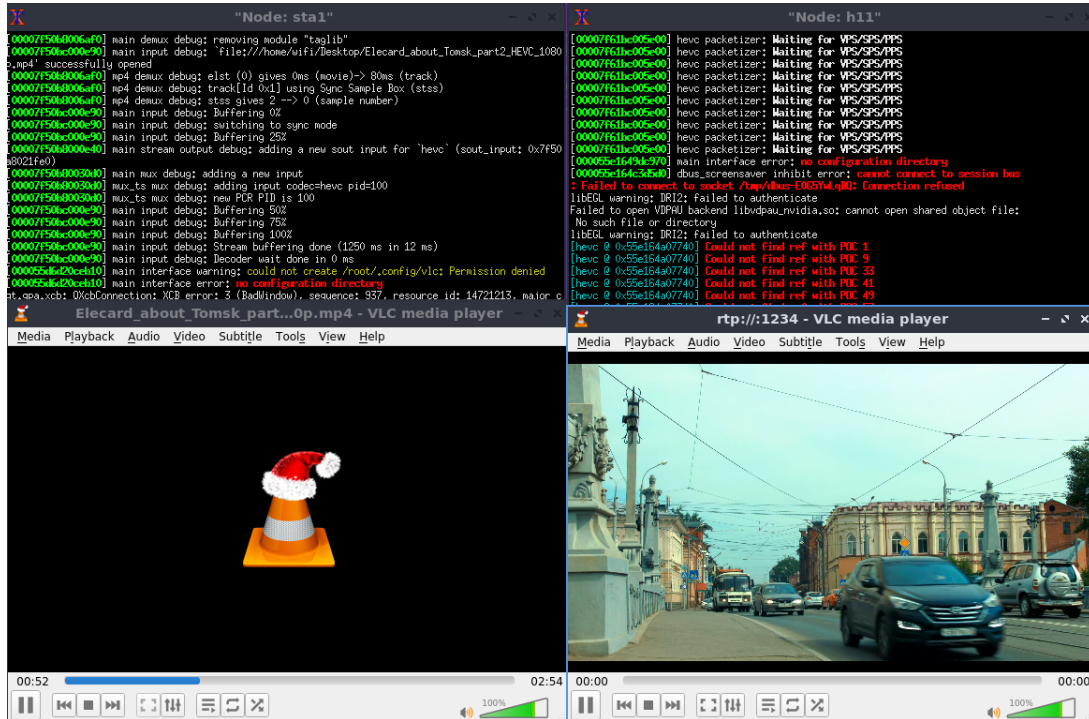


Figure 5.2: Video Streaming Test in the emulated network.

Current media / stream statistics

▼ Audio	
Decoded	2577 blocks
Played	1288 buffers
Lost	0 buffers
▼ Video	
Decoded	841 blocks
Displayed	812 frames
Lost	0 frames
▼ Input/Read	
Media data size	0 KiB
Input bitrate	0 kb/s
Demuxed data size	2695 KiB
Content bitrate	634 kb/s
Discarded (corrupted)	0
Dropped (discontinued)	0

Figure 5.3: Media statistics of the Streaming session.

```

deviceId-of:000000000000000a, flowRuleCount=5
  id=100003233c5c0, state=ADDED, bytes=1364447, packets=9788, duration=6147, liveType=UNKNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:bddp], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[]}, transition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
  id=100007aa3d8d8, state=ADDED, bytes=1364447, packets=9788, duration=6147, liveType=UNKNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:lldp], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[]}, transition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
  id=10000d78dcf79, state=ADDED, bytes=0, packets=0, duration=6072, liveType=UNKNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:arp], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[]}, transition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
  id=600006bf00b09, state=ADDED, bytes=30120900, packets=22000, duration=145, liveType=UNKNOWN, priority=10, tableId=0, appId=org.onosproject.fwd, selector=[IN_PORT:2, ETH_DST:66:D6:9E:00:CE:01, ETH_SRC:02:00:00:00:00:00], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:3], deferred=[]}, transition=None, meter=[], cleared=false, StatTrigger=null, metadata=null}
  id=10000a2aa5bb5, state=ADDED, bytes=96496, packets=128, duration=6147, liveType=UNKNOWN, priority=5, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:ipv4], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[]}, transition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}

```

Figure 5.4: Flow entries in tableID=0 related to video streaming.

5.2 Use Case: Recovery from link failure

In railway networks the term resilience is commonly spoken and is defined as the ability of the railway system to provide effective services in normal conditions and to have the best performance in the recovery from disruptions or disasters. Resilience is a comprehensive system measure and covers the following building characteristics which represent distinct system states: vulnerability, survivability, response and recovery [49].

Therefore, as more changes to the network occur the more fragile urban mobility becomes. Such events may range from disturbances (daily variations in operations), disruptions (due to failures of infrastructure, vehicles, engineering works and detrimental weather conditions such as rain, snow storm, wind), to disasters (earthquakes, floods and hurricanes). An example happened in the Netherlands, in 2018, when on average 14 disruptions occurred in a day, lasting about 2 hours each. From these numbers, vehicle and infrastructure failures took about 70%. This ended up causing many canceled and delayed trains and consequently compromised the customers' need for mobility, as passengers ended up being late at their destinations.

Another example was back in 2012, a hurricane named Sandy, flooded several subway stations and tunnels in New York City causing severe damage to the system, and took them two weeks to put all systems back to normal and several months for stations seriously affected to be fully functional again.

Critical infrastructure networks, such as transport and power networks, are essential for the functioning of a society. The rising transport demand increases the congestion in railway networks and thus they become more interdependent and more complex to operate. Also, an increasing number of disruptions due to system failures as well as climate changes can be expected in the future. As a consequence, many trains are cancelled and excessively delayed, and thus, many passengers are not reaching their destinations which compromises customers need for mobility. Currently, there is a rising need to quantify impacts of disruptions and the evolution of system performance. SDN is expected to decrease the recovery time when a failure in these types of systems occur, of course that in some cases, like natural disasters, it is impossible to ensure that the network continues to function but is possible to assure that when possible the network will reach the steady-state as fast as possible.

To test the performance of the Controller when a link fails, the link between the **Contumil** switch (s9) and the **CCO** switch (s7) was changed to down, Figure 5.5.

```
mininet-wifi> link s9 s7 down
mininet-wifi> link s9 s7 up
mininet-wifi> link s9 s10 down
```

Figure 5.5: Simulating a Link Down in the path of **h13** to **sta1**.

The test started with a ping from host **h13** (IP address=10.0.3.5) to the wireless station **sta1** (IP

address=10.0.0.1). The Controller automatically installs the flow entries for the shortest path (*h13–Contumil–CCO–Campanhã–ap1–sta1*), computed using the Dijkstra algorithm, Figure 5.6.

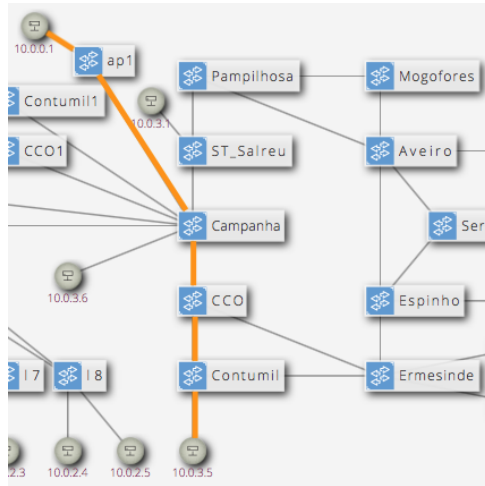


Figure 5.6: Shortest path between **h13** and **sta1**.

When the link between **Contumil** (s9) and **CCO** (s7) fails, the path is re-computed in the Controller and new flow entries are installed in the switches so that packets have to go to **Ermesinde** (s10) and then to **CCO** (s7). The new shortest path is now (*h13–Contumil–Ermesinde–CCO–Campanhã–ap1–sta1*), Figure 5.7.

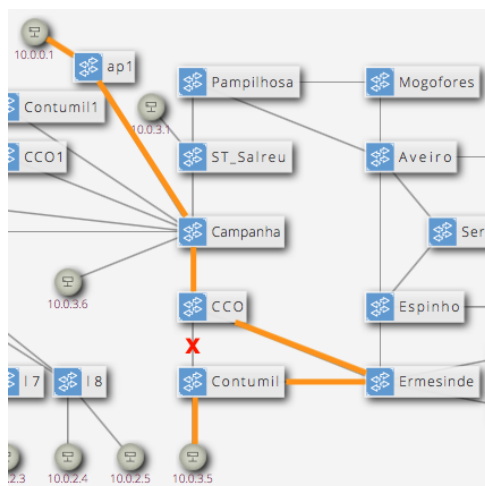
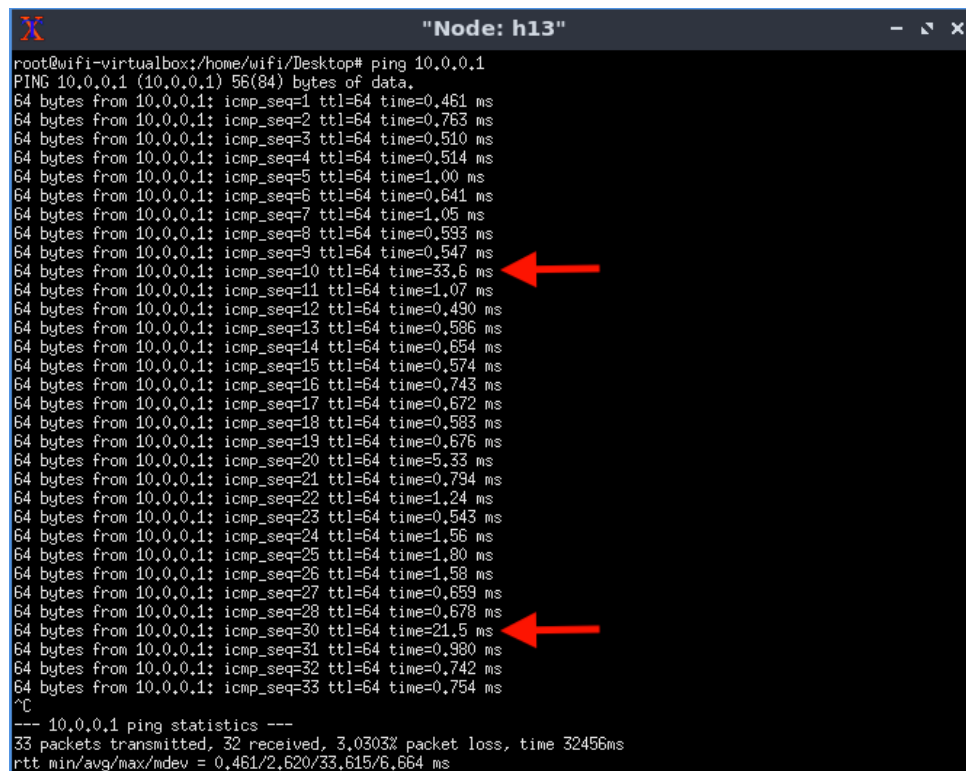


Figure 5.7: Shortest path between **h13** and **sta1** after link failure.

Although the Controller takes some time to understand that the topology changed due to occurrence of a network event and compute new path, it can be observed in Figure 5.8 that when the link fails the ping echo replies increases from around 1ms to 33.6ms (marked by the first red arrow in that figure), and immediately back to around 1ms. A second test is made when the link between **Contumil** (s9) and **Ermesinde** (s10) is down and we can see in the second red arrow that the path to **sta1** is once again

changed because the Ping increases to 21.5ms.



```
root@wifi-virtualbox:/home/wifi/Desktop# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.461 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.763 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.510 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.514 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=1.00 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.641 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=1.05 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.593 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.547 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=33.6 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=1.07 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=0.490 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=0.586 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=0.654 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=0.574 ms
64 bytes from 10.0.0.1: icmp_seq=16 ttl=64 time=0.743 ms
64 bytes from 10.0.0.1: icmp_seq=17 ttl=64 time=0.672 ms
64 bytes from 10.0.0.1: icmp_seq=18 ttl=64 time=0.583 ms
64 bytes from 10.0.0.1: icmp_seq=19 ttl=64 time=0.676 ms
64 bytes from 10.0.0.1: icmp_seq=20 ttl=64 time=5.33 ms
64 bytes from 10.0.0.1: icmp_seq=21 ttl=64 time=0.794 ms
64 bytes from 10.0.0.1: icmp_seq=22 ttl=64 time=1.24 ms
64 bytes from 10.0.0.1: icmp_seq=23 ttl=64 time=0.543 ms
64 bytes from 10.0.0.1: icmp_seq=24 ttl=64 time=1.56 ms
64 bytes from 10.0.0.1: icmp_seq=25 ttl=64 time=1.80 ms
64 bytes from 10.0.0.1: icmp_seq=26 ttl=64 time=1.58 ms
64 bytes from 10.0.0.1: icmp_seq=27 ttl=64 time=0.659 ms
64 bytes from 10.0.0.1: icmp_seq=28 ttl=64 time=0.678 ms
64 bytes from 10.0.0.1: icmp_seq=30 ttl=64 time=21.5 ms
64 bytes from 10.0.0.1: icmp_seq=31 ttl=64 time=0.980 ms
64 bytes from 10.0.0.1: icmp_seq=32 ttl=64 time=0.742 ms
64 bytes from 10.0.0.1: icmp_seq=33 ttl=64 time=0.754 ms
^C
--- 10.0.0.1 ping statistics ---
33 packets transmitted, 32 received, 3.0303% packet loss, time 32456ms
rtt min/avg/max/mdev = 0.461/2.620/33.615/6.664 ms
```

Figure 5.8: Failure of links while h13 ping sta1.

In both situations the Controller was very efficient to compute the alternative path and the E2E delay did not get higher than 50ms. Having in mind that railway networks are considered critical infrastructures and provide critical services, the delay in the network should be as low as possible, as already said in Section 2.2, in GSM-R the E2E should be less or equal to 500ms, being that with the use of SDN the E2E was not higher than 50ms it complies with these types of architectures requirements. So we can verify that the use of SDN/NFV would be quite beneficial.

5.3 Intent-based networking

The Intent Framework is a subsystem for IBN that allows applications to specify their network control desires in form of policy declarations rather than mechanisms or procedures. We refer to these policy-based directives as intents. The ONOS core accepts the intent specifications and translates them, via intent compilation, into installable intents, which are essentially actionable operations on the network [50]. These actions are carried out by intent installation process, which results in some changes to the environment, such as tunnel links being provisioned, flow rules being installed on a switch, etc. [51].

Intents may be described in terms of:

- *Network Resource*: A set of object models, such as links, that tie back to the parts of the network affected by an intent.
- *Constraints*: Weights applied to a set of network resources, such as bandwidth, optical frequency, and link type.
- *Criteria*: Packet header fields or patterns that describe a slice of traffic.
- *Instructions*: Actions to apply to a slice of traffic, such as header field modifications, or outputting through specific ports.

Figure 5.9 shows all active ports in **Contumil** (s9) and the device that each port is connected to. Port 1 is connected to switch with id *of:0000000000000007*, **CCO** (s7), port 2 is connected to *of:000000000000000a*, **Ermesinde** (s10) and finally port 3 is connected to host with mac address *1E:FC:03:71:B3:AA*, **h13**.

Ports

Enabled	ID	Speed	Type	Egress Links	Name
false	Local	0	Copper		s9
true	1	10000	Copper	of:0000000000000007/2	s9-eth1
true	2	10000	Copper	of:000000000000000a/1	s9-eth2
true	3	10000	Copper	1E:FC:03:71:B3:AA/None	s9-eth3

Figure 5.9: Switch **Contumil** (s9) ports.

Using the same example in Section 5.2 an intent was created between **h13** and **sta1**. And the shortest path between both them was created (*h13–Contumil–CCO–Campanhã–ap1–sta1*). Immediately the controller sets the smallest path and the connection is established. Right away a specific flow is created in ONOS with the specifications of the intent created, Figure 5.10. We can see in the figure that a bidirectional flow is established between the two hosts, the traffic received in port 1 with source **h13** is forward to the destination **sta1** with id *02:00:00:00:00:00* through port 3, connected to **CCO** (s7).

```
id=be000004f836df, state=ADDED, bytes=0, packets=0, duration=40, liveType=UNKNOWN, priority=100, tableId=0, appId=org.onosproject.net.intent, selector=[IN_PORT:1, ETH_DST:1E:FC:03:71:B3:AA, ETH_SRC:02:00:00:00:00:00], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:3], deferred=[], transition=None, meter=[], cleared=false, StatTrigger=null, metadata=null}
id=be000075655bd3, state=ADDED, bytes=0, packets=0, duration=40, liveType=UNKNOWN, priority=100, tableId=0, appId=org.onosproject.net.intent, selector=[IN_PORT:3, ETH_DST:02:00:00:00:00:00, ETH_SRC:1E:FC:03:71:B3:AA], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:1], deferred=[], transition=None, meter=[], cleared=false, StatTrigger=null, metadata=null}
```

Figure 5.10: Flows in **Contumil** (s9) before link failure.

In Figure 5.11 the ONOS GUI is now possible to see the created intent represented by a yellow dot line indicating that the traffic between the two hosts is following the path represented by that line.

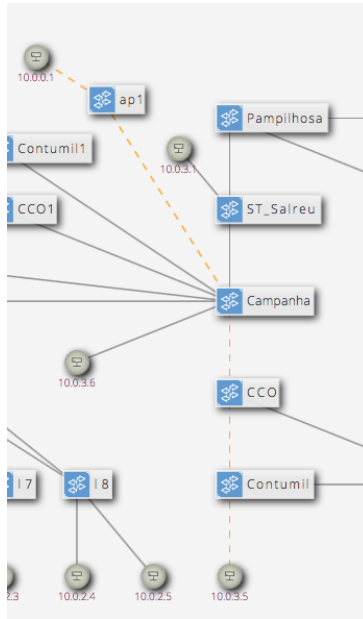


Figure 5.11: GUI representation of the intent created between hosts.

After the success in the establishment of the connection a link between **Contumil** (s9) and **CCO** (s7) is placed as down.

Now the controller has to discover a different path for the packets to be forwarded to their destination. This is done very quickly and when the controller finds an alternative path, the yellow line changes and shows the new shortest path (*h13–Contumil–Ermesinde–CCO–Campanhã–ap1–sta1*), Figure 5.12.

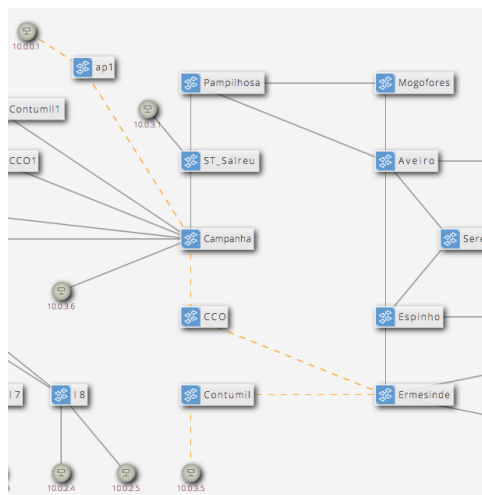


Figure 5.12: GUI representation of the intent created after the link failure.

In ONOS a new flow is created to establish the new path through the end hosts, Figure 5.13. We can see in the figure that a bidirectional flow is established between the two hosts, the traffic received in port 3 with source **h13** is forward to the destination **sta1** with id *02:00:00:00:00:00* through port 2 that is

connected to **Ermesinde** (s10).

```
id=be000075655bd3, state=ADDED, bytes=0, packets=0, duration=305, liveType=UNKNOWN, priority=100, tableId=0, appId=org.onosproject.net.intent, selector=[IN_PORT:3, ETH_DST:02:00:00:00:00:00, ETH_SRC:1E:FC:03:71:B3:AA], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:2], deferred=[], transition=None, meter=[], cleared=false, StatTrigger=null, metadata=null}
id=be00009d3474bf, state=ADDED, bytes=0, packets=0, duration=61, liveType=UNKNOWN, priority=100, tableId=0, appId=org.onosproject.net.intent, selector=[IN_PORT:2, ETH_DST:1E:FC:03:71:B3:AA, ETH_SRC:02:00:00:00:00:00], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:3], deferred=[], transition=None, meter=[], cleared=false, StatTrigger=null, metadata=null}
```

Figure 5.13: Flows in **Contumil** (s9) after link failure.

It is also possible to compare the creation of flows duration time when the flow changes, in the first excepted route the controller previously established the path between hosts, meaning that the time for creating that flow is relatively low, about 40ms. When the link is down and the topology changes the controller has to determine an alternative route, so it is normal to take a little longer to do that, so the duration of the new flow is about 305ms, it is really fast compared to some network protocols convergence times.

5.4 Use Case: Controller failure

Physically-distributed SDN controllers are mainly used in large-scale SDN networks for scalability, performance and reliability reasons.

In distributed SDN architectures, the SDN control plane supports the interaction between multiple controllers (normally controlling different zones of a network) through their “east-west” interfaces. Inter-controller communications are indeed needed to synchronize the controllers’ shared data structures to maintain a consistent global network view, and therefore ensure the correct behavior of the network applications running on top of the distributed controllers.

Atomix is a reactive Java framework for building scalable, fault-tolerant distributed systems, is now used as a new architecture which physically decouples cluster management, service discovery, and persistent data storage from the ONOS nodes themselves. Atomix nodes must be created first and then a cluster of those nodes will be created for data storage and coordination. ONOS nodes are then configured with a list of Atomix nodes to which to connect Figure 5.14.

In this section, a explored cluster hypothesis to demonstrate the importance of implementing a distributed controller is demonstrated. For this experiment Docker was used to run images corresponding to both Atomix and ONOS nodes. Two Atomix containers and two ONOS containers were created in order to assure network reliability and fault tolerance. As we can see in Figure 5.15, two ONOS nodes were formed and put together as a cluster, with IP addresses 172.20.0.4 and 172.20.0.5.

The idea is to separate network applications and services. The same Mininet topology was once again created, but this time with no radio networks to simplify the deployment of the infrastructure. After that, the previously created ONOS instances were used to separate network parts. This is done through the use of two controllers, one with IP=172.20.0.4, that controls the railway core and access

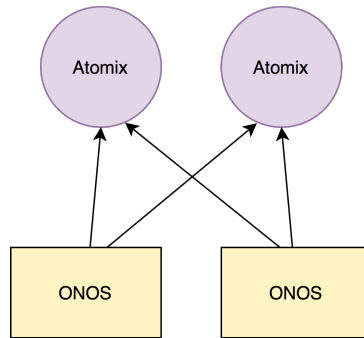


Figure 5.14: Relationship between Atomix nodes and ONOS nodes.

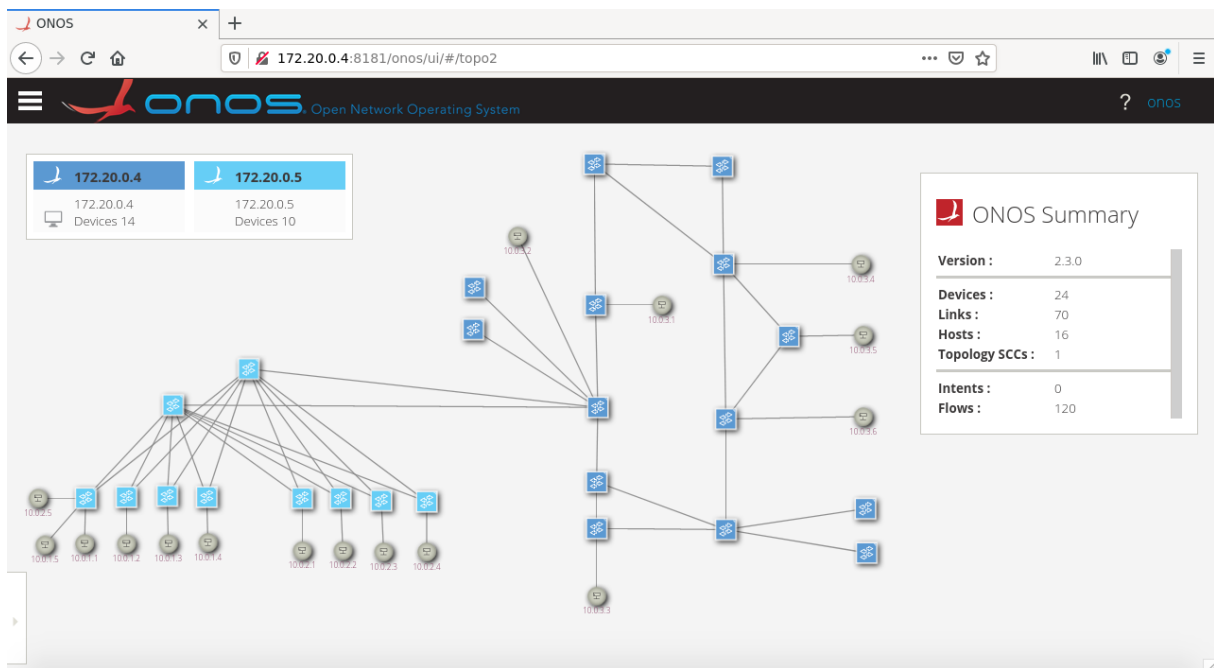


Figure 5.15: Creation of the ONOS cluster.

network, and the other with IP=172.20.0.5 that is responsible for controlling the datacenter network. This way the amount of data/flows and packets handled by each controller is decreased, and so the overall performance will surely be better.

6

Conclusion

Contents

6.1 Conclusions	63
6.2 System Limitations and Future Work	64

This final chapter finalizes the work, drawing conclusions about the achieved goals, as well as addressing the limitations of the proposed solution, and the future work that can be pursued in that regard.

6.1 Conclusions

After analyzing and understanding how technologies like roadside and railway networks work it is possible to verify that they are getting “old”, very quickly, and have a lot of constraints concerning connection, routing performance, interference of other networks, high velocity and others. So, the need to evolve and use technologies like SDN and NFV is huge, because they can easily streamline and optimize this type of infrastructures, and that is what we proposed to explore with this project. There are already a lot of works and researches in this area but in all the solutions the scope is very relative and not very specific. That is due to the fact that we are dealing with critical systems, involved in the day to day of our society, and also that there are lives at stake, and so, assuring the safety of the population is mandatory. So this kind of revolutionary technology should only be put into operation when it is possible to ensure that they are stable enough and reliable. Our belief is that they will revolutionize not only the area of transportation but also the technological world in general due to all of the advantages they bring. This document presented the research aimed at understanding the mentioned technologies, and a study on the utilization of SDN technologies, NFV and associated frameworks with a focus in Railway communications networks, in the context of a project being deployed in Portuguese Railways that arose from the partnership between Thales Portugal and Infraestruturas de Portugal.

A possible railway network architecture was designed based on the project carried out by Thales Portugal, which allowed a demonstrator implementation to be created as close to real life as possible in order to ensure that the conditions and requirements expected in the project were in fact fulfilled. It was possible to perceive through the results obtained that when having a controller, in the case of a network event, like a link failure, the End-To-End delay is not greater than 50ms, and the expected E2E delay in GSM-R must be less than or equal to 500ms, so is a very important result to take into account when adopting SDN in this type of networks. Regarding the transmission of streams or the data transfer rate on the network is quite significant, through the transmission and simulation of a HD video stream it was possible to observe that it remained at an average of around 634kbps of data rate uplink, which again is a point to favor of using SDN. Regarding technologies for network recovery or even increasing the network in terms of equipments or links, the process is very streamlined due to the fact that the SDN is prepared for these situations and is able to do it in the best and fastest way possible.

It is then possible to conclude that SDN and NFV are able to effectively meet all expectations and that their implementation only brings benefits to networks like railway, both in terms of performance as well as safety and reliability.

6.2 System Limitations and Future Work

In the simulated implementation used in this work, through the Mininet emulator, there is a known limitation with regard to the characteristics of the links, which are already implemented from scratch and even if their characteristics are changed in the topology code, the Mininet emulator does not reflect those changes correctly, and so, the speeds of the links are limited to the CPU capacity during the emulation. So the initial objective of configuring each network link to precisely 1Gbps, in order to have the same characteristics of the real equipment used, was overlooked due to the system limitation for this case, so the default links have a capacity of approximately 10Gbps.

With Mininet it is possible to make the interconnection between virtualized OpenvSwitches with real life switches outside the virtualized environment. In order to obtain an even more close to reality implementation, it would have been quite advantageous the access to identical equipment deployed in the project, to be able to study in more depth the results obtained with real equipment instead of a totally virtualized solution. But knowing that the equipment currently operating in the project supports and is compatible with SDN and OpenFlow, no obstacle to the implementation of SDN is foreseen. Additionally, the radio network part that railway infrastructures are evolving to (namely LTE-R are one of the main components of the architecture, it would have been quite beneficial for the mobile network backhaul to be made and implemented using LTE in the wireless part, as it would bring even more advantages in the study of the solution, due to the fact that LTE contributes to a more technology-independent and resilient communication service, instead of a simple radio network implemented with a wifi station and an access point, but it was enough to prove the benefits of the SDN adoption.

Bibliography

- [1] W. Quan, Y. Liu, H. Zhang, and S. Yu, "Enhancing Crowd Collaborations for Software Defined Vehicular Networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 80–86, 2017.
- [2] M. Krohn, R. Daher, M. Arndt, and D. Tavangarian, "Aspects of roadside backbone networks," *Proceedings of the 2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology, Wireless VITAE 2009*, pp. 788–792, 2009.
- [3] H. Cai and Y. Lin, "A roadside ITS data bus prototype for intelligent highways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 2, pp. 344–348, 2008.
- [4] Y. Peng, Z. Abichar, and J. M. Chang, "Roadside-aided routing (RAR) in vehicular networks," *IEEE International Conference on Communications*, vol. 8, no. c, pp. 3602–3607, 2006.
- [5] European Commission, "COMMISSION REGULATION (EU) 2016/919 of 27 May 2016 on the technical specification for interoperability relating to the 'control-command and signalling' subsystems of the rail system in the European Union," vol. 33, p. 919, 2016.
- [6] A. Sniady, "Communication Technologies Support to Railway Infrastructure and Operations," Ph.D. dissertation, 2015.
- [7] W. Jiang, S. Lin, and Z. Zhong, "QoS requirements of GSM-R for locomotive synchronization operation and control service," *7th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2011*, 2011.
- [8] A. Wolf, "Communication Systems for Next-generation Railways: D4.2 - Report on Technical and Quality of Service Viability," (Public Access), MISTRAL Consortium, Tech. Rep. D4.2, 2019. [Online]. Available: <https://projects.shift2rail.org/download.aspx?id=13de043a-dde6-4216-a41e-4e2a7e6c16ea>
- [9] F. Mazzenga, R. Giuliano, A. Neri, F. Rispoli, A. Ruggeri, M. Salvitti, E. Del Signore, and V. Fontana, "The adoption of public telecom services for the evolution of the ERTMS-ETCS train control sys-

- tems: Challenges and opportunities,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9066, no. May, pp. 177–188, 2015.
- [10] R. He, B. Ai, G. Wang, K. Guan, Z. Zhong, A. F. Molisch, C. Briso-Rodriguez, and C. P. Oestges, “High-Speed Railway Communications: From GSM-R to LTE-R,” *IEEE Vehicular Technology Magazine*, vol. 11, no. 3, pp. 49–58, 2016.
- [11] A. Gopalasingham, Q. Pham Van, L. Roullet, C. S. Chen, E. Renault, L. Natarianni, S. De Marchi, and E. Hamman, “Software-Defined mobile backhaul for future Train to ground Communication services,” *2016 9th IFIP Wireless and Mobile Networking Conference, WMNC 2016*, pp. 161–167, 2016.
- [12] X. Liang and X. Qiu, “A software defined security architecture for SDN-based 5G network,” *Proceedings of 2016 5th International Conference on Network Infrastructure and Digital Content, IEEE IC-NIDC 2016*, vol. 16, no. 6, pp. 17–21, 2016.
- [13] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “A Survey on Software-Defined Networking,” *Asian Pacific Journal of Reproduction*, vol. 7, no. 2, pp. 27–51, 2015.
- [14] W. B. Jaballah, M. Conti, and C. Lal, “A Survey on Software-Defined VANETs: Benefits, Challenges, and Future Directions,” 2019. [Online]. Available: <http://arxiv.org/abs/1904.04577>
- [15] J. H. Cox, J. Chung, S. Donovan, J. Ivey, R. J. Clark, G. Riley, and H. L. Owen, “Advancing Software-Defined Networks: A Survey,” *IEEE Access*, vol. 5, pp. 25 487–25 526, 2017.
- [16] J. S. Weng, J. Weng, Y. Zhang, W. Luo, and W. Lan, “BENBI: Scalable and dynamic access control on the northbound interface of SDN-Based VANET,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 822–831, 2019.
- [17] J. S. Marcus and G. Molnar, “Network Sharing and 5G in Europe: The Potential Benefits of Using SDN or NFV,” *SSRN Electronic Journal*, 2018.
- [18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69, 2008.
- [19] P. Goransson, C. Black, and T. Culver, *Software defined networks: a comprehensive approach*. Morgan Kaufmann, 2016.
- [20] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, “SDN controllers: A comparative study,” *Proceedings of the 18th Mediterranean Electrotechnical Conference: Intelligent and Efficient Technologies and Services for the Citizen, MELECON 2016*, no. 978, pp. 18–20, 2016.

- [21] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers," *2014 World Congress on Computer Applications and Information Systems, WCCAIS 2014*, 2014.
- [22] M. P. Anastasopoulos, A. Tzanakaki, and D. Simeonidou, "ICT platforms in support of future railway systems," *Proceedings of 7th Transport Research Arena TRA 2018*, no. 1, 2018.
- [23] Y. Zhou, M. Zhu, L. Xiao, L. Ruan, W. Duan, D. Li, R. Liu, and M. Zhu, "A load balancing strategy of SDN controller based on distributed decision," *Proceedings - 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2014*, no. 978, pp. 851–856, 2015.
- [24] R. Macedo, R. De Castro, A. Santos, Y. Ghamri-Doudane, and M. Nogueira, "Self-organized SDN controller cluster conformations against DDoS attacks effects," *2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings*, pp. 0–5, 2016.
- [25] Z. He, J. Cao, and X. Liu, "SDVN: Enabling rapid network innovation for heterogeneous vehicular communication," *IEEE Network*, vol. 30, no. 4, pp. 10–15, 2016.
- [26] K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, and L. Lei, "Soft-defined heterogeneous vehicular network: Architecture and challenges," *IEEE Network*, vol. 30, no. 4, pp. 72–80, 2016.
- [27] H. Li, M. Dong, and K. Ota, "Control Plane Optimization in Software-Defined Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 7895–7904, 2016.
- [28] D. Franco, M. Aguado, and N. Toledo, "An adaptable train-to-ground communication architecture based on the 5G technological enabler SDN," *Electronics (Switzerland)*, vol. 8, no. 6, 2019.
- [29] A. L. Ruscelli, S. Fichera, F. Paolucci, A. Giorgetti, P. Castoldi, and F. Cugini, "Introducing network softwarization in next-generation railway control systems," *MT-ITS 2019 - 6th International Conference on Models and Technologies for Intelligent Transportation Systems*, vol. 9, no. 4, 2019.
- [30] "Alcatel-Lucent OmniSwitch 6250." [Online]. Available: https://www.konighartman.nl/UserFiles/Product/Datasheet/Datasheet_Alcatel_lucent_OS6250.pdf
- [31] P. Berde, W. Snow, G. Parulkar, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, and P. Radoslavov, "ONOS: Towards an Open, Distributed SDN OS," pp. 1–6, 2014.
- [32] A. Nguyen-Ngoc, S. Lange, T. Zinner, M. Seufert, P. Tran-Gia, N. Aerts, and D. Hock, "Performance evaluation of selective flow monitoring in the ONOS controller," *2017 13th International Conference on Network and Service Management, CNSM 2017*, vol. 2018-Janua, pp. 1–6, 2017.

- [33] M. P. Anastasopoulos, A. Tzanakaki, G. S. Zervas, B. R. Rofoee, R. Nejabati, D. Simeonidou, G. Landi, N. Ciulli, J. F. Riera, and J. A. García-Espín, "Convergence of heterogeneous network and IT infrastructures in support of fixed and mobile Cloud services," *2013 Future Network and Mobile Summit, FutureNetworkSummit 2013*, no. August, pp. 155–161, 2013.
- [34] G. Brebner, "Programmable hardware for high performance SDN," *Conference on Optical Fiber Communication, Technical Digest Series*, vol. 2015-June, pp. 5–7, 2015.
- [35] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-WiFi: Emulating Software-Defined Wireless Networks," 2015.
- [36] A. Bianco, R. Birke, L. Giraud, and M. Palacin, "OpenFlow switching: Data plane performance," *IEEE International Conference on Communications*, 2010.
- [37] A. S. Monge and K. G. Szarkowicz, *MPLS in the SDN Era: Interoperable Scenarios to Make Networks Scale to New Services*. O'Reilly Media, 2015. [Online]. Available: <https://books.google.pt/books?id=OBsoCwAAQBAJ>
- [38] R. Interdonato, M. Atzmueller, S. Gaito, R. Kanawati, C. Largeron, and A. Sala, "Feature-rich networks: Going beyond complex network topologies," *Applied Network Science*, vol. 4, no. 1, pp. 1–13, 2019.
- [39] C. Luciani, "From MPLS to SD-WAN: Opportunities, Limitations and Best Practices," Ph.D. dissertation, KTH, School of Electrical Engineering and Computer Science (EECS), 2019.
- [40] I. Šeremet and S. Čaušević, "Extending IP / MPLS network in order to meet 5G requirements," *Journal of Mechatronic, Automation and Identification Technology Vol 4, No. 1*, pp. 13-18, 2019, vol. 4, no. March, pp. 20–22, 2019.
- [41] E. K. Ali, M. Manel, and Y. Habib, "An efficient MPLS-based source routing scheme in software-defined wide area networks (SD-WAN)," *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, vol. 2017-Octob, pp. 1205–1211, 2018.
- [42] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors (Switzerland)*, vol. 19, no. 9, 2019.
- [43] T. Porter and X. H. Peng, "An objective approach to measuring video playback quality in lossy networks using TCP," *IEEE Communications Letters*, vol. 15, no. 1, pp. 76–78, 2011.
- [44] A. Aloman, A. I. Ispas, P. Ciotirnae, R. Sanchez-Iborra, and M. D. Cano, "Performance Evaluation of Video Streaming Using MPEG DASH, RTSP, and RTMP in Mobile Networks," *Proceedings - 2015 8th IFIP Wireless and Mobile Networking Conference, WMNC 2015*, pp. 144–151, 2016.

- [45] R. Anup, L. Rob, and S. Henning, "Real Time Streaming Protocol (RTSP)," *RFC Editor*, vol. 1, no. 071116072, p. 92, 1998. [Online]. Available: <https://rfc-editor.org/rfc/rfc2326.txt>
- [46] S. Waleed, M. Faizan, M. Iqbal, and M. I. Anis, "Demonstration of single link failure recovery using Bellman Ford and Dijkstra algorithm in SDN," *ICIEECT 2017 - International Conference on Innovations in Electrical Engineering and Computational Technologies 2017, Proceedings*, pp. 0–3, 2017.
- [47] D. Sanvito, D. Moro, M. Gulli, I. Filippini, A. Capone, and A. Campanella, "ONOS Intent Monitor and Reroute service: Enabling plugplay routing logic," *2018 4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018*, no. NetSoft, pp. 456–461, 2018.
- [48] P. Panwaree, J. Kim, C. Aswakul, J. Kim, and C. Aswakul, "Packet Delay and Loss Performance of Streaming Video over Emulated and Real OpenFlow Networks," *The 29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC), Phuket, Thailand, July 2014*, no. February 2015, pp. 777–779, 2014.
- [49] N. Bešinović, "Resilience in railway transport systems: a literature review and research agenda," *Transport Reviews*, vol. 40, no. 4, pp. 457–478, 2020.
- [50] M. Kiran, E. Pouyoul, A. Mercian, B. Tierney, C. Guok, and I. Monga, "Enabling intent to configure scientific networks for high performance demands," *Future Generation Computer Systems*, vol. 79, pp. 205–214, 2018.
- [51] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, H. Flinck, and M. Namane, "Benchmarking the ONOS Intent Interfaces to Ease 5G Service Management," *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, 2018.



Topology Code

The Python code of the mininet wifi topology used to reproduce the IP project implementation is presented in Listing A.1.

Listing A.1: PYTHON Topology Code

```
1 #!/usr/bin/python
2
3 from mininet.node import RemoteController, OVSKernelSwitch, Host
4 from mininet.log import setLogLevel, info
5 from mn_wifi.net import Mininet_wifi
6 from mn_wifi.node import Station, OVSKernelAP
7 from mn_wifi.cli import CLI
8 from mn_wifi.link import wmediumd
9 from mn_wifi.wmediumdConnector import interference
10 from subprocess import call
11
```

```

12
13 def myNetwork():
14
15     net = Mininet_wifi(topo=None,
16                        build=False,
17                        link=wmediumd,
18                        wmediumd_mode=interference)
19
20     info( '*** Adding controller\n' )
21     c0 = net.addController(name='c0',
22                            controller=RemoteController,
23                            protocol='tcp',
24                            port=6633,
25                            ip='10.10.10.2')
26
27     info( '*** Add switches/APs\n' )
28     s1 = net.addSwitch('s1', cls=OVSKernelSwitch, protocols='OpenFlow13')
29     s2 = net.addSwitch('s2', cls=OVSKernelSwitch, protocols='OpenFlow13')
30     s3 = net.addSwitch('s3', cls=OVSKernelSwitch, protocols='OpenFlow13')
31     s4 = net.addSwitch('s4', cls=OVSKernelSwitch, protocols='OpenFlow13')
32     s5 = net.addSwitch('s5', cls=OVSKernelSwitch, protocols='OpenFlow13')
33     s6 = net.addSwitch('s6', cls=OVSKernelSwitch, protocols='OpenFlow13')
34     s7 = net.addSwitch('s7', cls=OVSKernelSwitch, protocols='OpenFlow13')
35     s8 = net.addSwitch('s8', cls=OVSKernelSwitch, protocols='OpenFlow13')
36     s9 = net.addSwitch('s9', cls=OVSKernelSwitch, protocols='OpenFlow13')
37     s10 = net.addSwitch('s10', cls=OVSKernelSwitch, protocols='OpenFlow13')
38     s11 = net.addSwitch('s11', cls=OVSKernelSwitch, protocols='OpenFlow13')
39     s12 = net.addSwitch('s12', cls=OVSKernelSwitch, protocols='OpenFlow13')
40     s13 = net.addSwitch('s13', cls=OVSKernelSwitch, protocols='OpenFlow13')
41     s14 = net.addSwitch('s14', cls=OVSKernelSwitch, protocols='OpenFlow13')
42     s15 = net.addSwitch('s15', cls=OVSKernelSwitch, protocols='OpenFlow13')
43     s16 = net.addSwitch('s16', cls=OVSKernelSwitch, protocols='OpenFlow13')
44     s17 = net.addSwitch('s17', cls=OVSKernelSwitch, protocols='OpenFlow13')
45     s18 = net.addSwitch('s18', cls=OVSKernelSwitch, protocols='OpenFlow13')
46     s19 = net.addSwitch('s19', cls=OVSKernelSwitch, protocols='OpenFlow13')
47     s20 = net.addSwitch('s20', cls=OVSKernelSwitch, protocols='OpenFlow13')
48     s21 = net.addSwitch('s21', cls=OVSKernelSwitch, protocols='OpenFlow13')
49     s22 = net.addSwitch('s22', cls=OVSKernelSwitch, protocols='OpenFlow13')

```

```

50 s23 = net.addSwitch('s23', cls=OVSKernelSwitch, protocols='OpenFlow13')
51 s24 = net.addSwitch('s24', cls=OVSKernelSwitch, protocols='OpenFlow13')
52 ap1 = net.addAccessPoint('ap1', cls=OVSKernelAP, ssid='ap1-ssid',
53                          channel='6', mode='g', position='30,50,0', range
54                          =30, protocols='OpenFlow13')
55
56 info( '*** Add hosts/stations\n')
57 sta1 = net.addStation('sta1', ip='10.0.0.1',
58                       position='30,60,0', range=20)
59 h1 = net.addHost('h1', cls=Host, ip='10.0.1.1', defaultRoute=None)
60 h2 = net.addHost('h2', cls=Host, ip='10.0.1.2', defaultRoute=None)
61 h3 = net.addHost('h3', cls=Host, ip='10.0.1.3', defaultRoute=None)
62 h4 = net.addHost('h4', cls=Host, ip='10.0.1.4', defaultRoute=None)
63 h5 = net.addHost('h5', cls=Host, ip='10.0.2.1', defaultRoute=None)
64 h6 = net.addHost('h6', cls=Host, ip='10.0.2.2', defaultRoute=None)
65 h7 = net.addHost('h7', cls=Host, ip='10.0.2.3', defaultRoute=None)
66 h8 = net.addHost('h8', cls=Host, ip='10.0.2.4', defaultRoute=None)
67 h9 = net.addHost('h9', cls=Host, ip='10.0.3.1', defaultRoute=None)
68 h10 = net.addHost('h10', cls=Host, ip='10.0.3.2', defaultRoute=None)
69 h11 = net.addHost('h11', cls=Host, ip='10.0.3.3', defaultRoute=None)
70 h12 = net.addHost('h12', cls=Host, ip='10.0.3.4', defaultRoute=None)
71 h13 = net.addHost('h13', cls=Host, ip='10.0.3.5', defaultRoute=None)
72 h14 = net.addHost('h14', cls=Host, ip='10.0.3.6', defaultRoute=None)
73
74 info("*** Configuring Propagation Model\n")
75 net.setPropagationModel(model="logDistance", exp=5)
76
77 info("*** Configuring wifi nodes\n")
78 net.configureWifiNodes()
79
80 info( '*** Add links\n')
81 net.addLink(s1, s2)
82 net.addLink(s2, s4)
83 net.addLink(s4, s1)
84 net.addLink(s1, s3)
85 net.addLink(s3, s5)
86 net.addLink(s6, s4)
87 net.addLink(s8, s6)

```

```
87 net.addLink(s5, s7)
88 net.addLink(s7, s9)
89 net.addLink(s9, s10)
90 net.addLink(s7, s10)
91 net.addLink(s10, s8)
92 net.addLink(s10, s11)
93 net.addLink(s10, s12)
94 net.addLink(s5, s13)
95 net.addLink(s5, s14)
96 net.addLink(sta1, ap1)
97 net.addLink(ap1, s5)
98 net.addLink(s5, s15)
99 net.addLink(s16, s5)
100 net.addLink(s17, s15)
101 net.addLink(s15, s18)
102 net.addLink(s15, s19)
103 net.addLink(s15, s20)
104 net.addLink(s15, s21)
105 net.addLink(s15, s22)
106 net.addLink(s15, s23)
107 net.addLink(s15, s24)
108 net.addLink(s16, s17)
109 net.addLink(s16, s18)
110 net.addLink(s16, s19)
111 net.addLink(s16, s20)
112 net.addLink(s16, s21)
113 net.addLink(s16, s22)
114 net.addLink(s16, s23)
115 net.addLink(s16, s24)
116 net.addLink(s17, h1)
117 net.addLink(s18, h2)
118 net.addLink(s19, h3)
119 net.addLink(s20, h4)
120 net.addLink(s21, h5)
121 net.addLink(s23, s22)
122 net.addLink(s22, h6)
123 net.addLink(s23, h7)
124 net.addLink(s24, h8)
```

```

125     net.addLink(s4, s8)
126     net.addLink(h14, s5)
127     net.addLink(s3, h9)
128     net.addLink(s9, h13)
129     net.addLink(s8, h12)
130     net.addLink(s6, h11)
131     net.addLink(s4, h10)
132
133     net.plotGraph(max_x=160, max_y=100)
134
135     info( '*** Starting network\n')
136     net.build()
137     info( '*** Starting controllers\n')
138     for controller in net.controllers:
139         controller.start()
140
141     info( '*** Starting switches/APs\n')
142     net.get('s1').start([c0])
143     net.get('s2').start([c0])
144     net.get('s3').start([c0])
145     net.get('s4').start([c0])
146     net.get('s5').start([c0])
147     net.get('s6').start([c0])
148     net.get('s7').start([c0])
149     net.get('s8').start([c0])
150     net.get('s9').start([c0])
151     net.get('s10').start([c0])
152     net.get('s11').start([c0])
153     net.get('s12').start([c0])
154     net.get('s13').start([c0])
155     net.get('s14').start([c0])
156     net.get('s15').start([c0])
157     net.get('s16').start([c0])
158     net.get('s17').start([c0])
159     net.get('s18').start([c0])
160     net.get('s19').start([c0])
161     net.get('s20').start([c0])
162     net.get('s21').start([c0])

```

```
163     net.get('s22').start([c0])
164     net.get('s23').start([c0])
165     net.get('s24').start([c0])
166     net.get('ap1').start([c0])
167
168     info( '*** Post configure nodes\n')
169
170     CLI(net)
171     net.stop()
172
173
174 if __name__ == '__main__':
175     setLogLevel( 'info' )
176     myNetwork()
```