Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática
2019

Miguel Ângelo Pereira da Silva

Funções Virtuais em Redes Veiculares Multihomed

Virtual Functions in Multihomed Vehicular Networks

Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

Miguel Ângelo Pereira da Silva

Funções Virtuais em Redes Veiculares Multihomed

Virtual Functions in Multihomed Vehicular Networks

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Nuno Miguel Abreu Luís, Investigador Auxiliar do Instituto de Telecomunicações de Aveiro, e da Doutora Susana Isabel Barreto de Miranda Sargento, Professora Catedrática do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president	Professsor Doutor António José Ribeiro Neves Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro
vogais / examiners committee	Professora Doutora Ana Cristina Costa Aguiar Professora Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto
	Doutor Nuno Miguel Abreu Luís Investigador Auxiliar do Instituto de Telecomunicações de Aveiro (orientador)

agradecimentos /
acknowledgements

Em primeiro lugar gostaria de agradecer aos meus pais pela oportunidade que me deram de continuar os meus estudos. Gostaria também de agradecer à professora Susana Sargento por me ter dado a oportunidade de fazer parte do NAP. Gostaria também de agradecer ao Instituto de Telecomunicações pelo acolhimento no decorrer deste trabalho. Agradeço também a todos os colegas do NAP que sempre estiveram dispostos a prestar a sua ajuda e a partilhar os seus conhecimentos. Quero agradecer também ao Doutor Miguel Luís e à professora Susana Sargento pela orientação e toda a ajuda prestada durante o desenvolvimento desta dissertação.

Palavras Chave

Redes Veiculares Multihomed, Virtualização de Funções de Rede, Gestão e Orquestração, Mobilidade, N-PMIPv6

Resumo

Nas redes veiculares atuais, o número de serviços e aplicações que estão a ser usados pelos veículos e seus ocupantes está a aumentar. Atualmente, a maioria dos serviços e aplicações estão localizados fora da Rede Veicular o que pode implicar um atraso adicional em serviços que são sensíveis ao atraso (e.g. segurança nas estradas). Para além disso, estas aplicações e serviços tornam-se inacessíveis sempre que a Rede Veicular perde contacto com a infraestrutura.

Esta dissertação apresenta uma solução prática que visa minimizar o impacto destes problemas. A solução concentra-se no uso de tecnologias de Network Function Virtualization (NFV) para suportar o lançamento de serviços na extremidade de uma Rede Veicular com mobilidade e suporte para multihoming, permitindo assim que certos serviços estejam acessíveis em situações de conectividade intermitente, assim como garantir menores atrasos para serviços críticos. Estes serviços são compostos por Funções Virtuais leves que são lançadas na extremidade da Rede Veicular, o mais próximo possível dos utilizadores.

Para avaliar o desempenho da solução proposta foram desenvolvidos vários cenários de teste assim como casos de uso. Os resultados obtidos mostram que a solução é capaz de lançar serviços na extremidade de uma Rede Veicular com baixos atrasos e com recuperação em situações de *handover* e mobilidade. Os casos de uso desenvolvidos mostram que, por exemplo, para um serviço que abrange apenas um veículo, caso o veículo perca ligação com a infraestrutura, quer o funcionamento quer a utilização do serviço não são afetados. Os casos de uso mostram também que, serviços lançados usando a solução, apresentam menores valores de atraso quando comparados com os mesmos serviços quando estes estão disponíveis na cloud.

Keywords

Multihomed Vehicular Networks, Network Functions Virtualization, Management and Orchestration, Mobility, N-PMIPv6

Abstract

In the current Vehicular Ad-hoc Networks (VANETs), the number of services and applications being used by the vehicles and its occupants is increasing. Nowadays, the majority of the services and applications are located outside the vehicular network which may imply an additional delay in services that are delay sensitive (e.g. road safety). In addition to that, these applications and services become inaccessible whenever the vehicular network loses contact with the infrastructure.

This dissertation presents a practical solution that aims to minimize the impact of these problems. The solution focuses on using Network Function Virtualization (NFV) technologies to support the deployment of services at the edge of a mobility-enabled multihomed VANET, thus allowing certain services to be accessible in intermittent connectivity situations, as well as ensuring lower delays for critical services. These services are made up of lightweight Virtual Functions (VxFs) that are deployed at the edge of the VANET, as close as possible to the users.

To evaluate the performance of the proposed solution several test scenarios, as well as use cases were developed. The results obtained show that the solution is capable of deploying services at the edge of the VANET with low delay and with recovery when in handover and mobility situations. The uses cases developed show that, for example, for a service which encompasses a single vehicle, if the vehicle loses connection with the infrastructure, both the operation and usage of the service are not affected. The use cases also show that, services deployed using the solution have lower delay values when compared with the same services when they are available in the cloud.

Contents

List of Figures

List of Tables

Acronyms

VANET	Vehicular ad-hoc Network
MANET	Mobile ad-hoc Network
AU	Application Unit
C-V2X	Cellular vehicle-to-everything
NFV	Network Function Virtualization
NS	Network Service
VNF	Virtual Network Function
VxF	Virtual Function
OBU	On-Board Unit
RSU	Road Side Unit
LMA	Local Mobiliy Anchor
IaaS	Infrastructure-as-a-Service
NFVI	NFV Infrastructure
WAVE	Wireless Access in Vehicular Environments - IEEE 802.11p
WiFi	IEEE 802.11 a/g/n
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
GPS	Global Positioning System
NAP	Network Architectures and Protocols
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
MTU	Maximum Transmission Unit
QoS	Quality-of-Service
MIPv6	Mobile Internet Protocol version 6
MIP	Mobile Internet Protocol
NEMO	Network Mobility
MN	Mobile Node
HA	Home Agent
FA	Foreign Agent
PMIPv6	Proxy Mobile Internet Protocol version 6
MAG	Mobile Access Gateway
RS	Router Solicitation
RA	Router Advertisement
PBU	Proxy Binding Update
PBA	Proxy Binding Acknowledgement
BCE	Binding Cache Entry
MR	Mobile Router
MNP	Mobile Network Prefix
MNN	Mobile Network Node
AR	Access Router
CoA	Care-of-Address
BU	Binding Update
BA	Binding Acknowledgement
CN	Correspondent Node
FN	Foreign Network
N-PMIPv6	Network PMIPv6
mMAG	mobile MAG
SCTP	Stream Control Transmission Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SHIM6	Site Multihoming by IPv6 Intermediation
UCE	User Cache Entry
TM	Terminal Manager
MAC	Media Access Control
FM	Flow Manager
IM	Information Manager
FCE	Flow Cache Entry
NIS	Network Information Server
UIS	User Information Server
RSSI	Received Signal Strength Indicator
PoA	Point of Attachment
NSP	Network Service Provider
VIM	Virtualized Infrastructure Manager
OPEX	Operational Expenditure
CAPEX	Capital Expenditure
VM	Virtual Machine
ETSI	European Telecommunications Standards Institute
ISG	Industry Specification Group
OSS/BSS	Operations and Business Support System
OSS	Operation Support System

BSS	Business Support System
MANO	NFV Management and Orchestration
VNFM	Virtual Network Function (VNF) Manager
NFVO	NFV Orchestrator
EM	Element Management
FCAPS	Fault, Configuration, Accounting, Performance and Security management
VNFD	VNF Descriptor
NSD	Network Service (NS) Descriptor
OSM	Open Source MANO
OS	Openstack
VCA	VNF Configuration & Abstraction
RO	Resource Orchestrator
NSO	Network Service Orchestrator
VC	Vehicular Cloud
VuC	Vehicles using Clouds
HVC	Hybrid Vehicular Clouds
SenaaS	Sensor-as-a-Service
IOS	Intelligent Onboard System
SUAV	Small Unmanned Aerial Vehicle
VoIP	Voice over Internet Protocol
RPi	Raspberry Pi
LXC	Linux Containers
VLAN	Virtual LAN
VXLAN	Virtual Extensible LAN
VNI	VXLAN Network Identifier
VTEP	VXLAN Tunnel Endpoint
IANA	Internet Assigned Numbers Authority
PMTUD	Path MTU Discovery
NTP	Network Time Protocol
AP	Access Point
SBC	Single-board Computer
AMQP	Advanced Message Queuing Protocol
DHCP	Dynamic Host Configuration Protocol

CHAPTER 1

Introduction

Nowadays, providing a good Internet connection and access to various types of services everywhere is becoming a requirement to fulfill: this is true even when inside our own vehicles. VANETs are seen as one of the key enablers for the *always connected* paradigm, providing useful communications among vehicles, and between vehicles and the infrastructure. Besides providing access to the Internet, vehicular communications can be used for information sharing, which may include vehicle's location, rest areas, fuel stations, etc., or even more important to share important information to be used for the detection of road congestion, dangerous road conditions or even car accidents [1].

With the 5^{th} generation of mobile networks gaining ground, which will include beyond the 5G New Radio and Cellular vehicle-to-everything (C-V2X) technology for vehicular communications, resource sharing and the use of softwarized networks, replacing hardware network functions through software functions, are becoming increasingly popular. This is where the concept of Network Function Virtualization (NFV) comes into play, a technology capable of decoupling software from hardware, enabling flexibility, programmability and extensibility to the network [2], [3]. Through NFV, the network functions are available in the cloud, and pushed into the edge of the network through the connection to the cloud. However, due to the intermittent connectivity of VANETs, the provisioning of softwarized network functions in the network nodes, such as the On-Board Units (OBUs) in vehicles, is not straightforward.

In a typical vehicular network scenario, most of the communications that originate from the vehicles are performed with services or applications located outside the vehicular network (i.e. on the Internet). This happens mostly due to the types of applications and services that are used by the vehicles' occupants, such as entertainment applications, but also due to hardware limitations and network configurations when it comes to more useful services in a vehicular scenario, such as safety applications. The solution that is used nowadays is to host virtualized versions of all the network functions, that make up the services which are used by the network and its end-users, in the cloud. This way, the VANET's users can use the services whenever they are needed. Nevertheless, this solution is not the best when it comes to certain types of services, such as services with delay sensitive requirements and capabilities. Something that is also problematic in a solution like this one is, given the fact that VANETs have a very dynamic network topology, handovers and loss of connection are to be expected, meaning that the access to all types of services would be dependent on the connection between the vehicles and the VANET.

Having all that in mind, the main objective of this dissertation is to develop a practical solution which uses NFV technologies to support the deployment of lightweight Virtual Functions (VxFs) at the edge of a vehicular network. These VxFs are typically deployed as Network Services (NSs), which are groups of VxFs working together to provide a complex service. This work focuses on allowing the deployment of VxFs as close as possible to the end-users. Several solutions have explored various types of integration of cloud computing into the scope of a VANET. Most of them are focused on the concept of Vehicular Cloud (VC), where the VANET infrastructure itself is part of the cloud [4], or on the concept of Infrastructure-as-a-Service (IaaS) in a VANET, where the vehicles' resources can be used to provide different types of services [5].

The solution developed in this dissertation brings some of the cloud characteristics to the edge of a multihomed VANET with mobility support, while at the same time making use of the IaaS concept. By extending the NFV Infrastructure (NFVI) up to the edge of a vehicular network with mobility and multihoming support, it is possible to explore the use of additional computing, networking and storage resources closer to the end user.

1.1 Objectives

The main goal of this dissertation is to create a solution that aims to allow users to access specific services even when the vehicles are not connected to the VANET, as well as reduce the delay between the services and the user by bringing lightweight VxFs closer to the user, with the help of general purpose hardware platforms deployed at the edge of the network. The following list denotes the main objectives of this dissertation:

- Study the evolution of mobility and multihoming protocols' architectures and implementations, with a specific focus on the currently used protocols;
- Study and understand how NFV technologies work;
- Study how to implement and deploy NFV technologies using open-source software;
- Design and implement a NFV solution that enables the deployment of lightweight VxFs at the edge of a VANET;
- Evaluate the behaviour of the implemented NFV solution presented in this dissertation given the unpredictable network conditions of a VANET;
- Design and evaluate Use Cases that demonstrate the capabilities of the solution.

1.2 Contributions

The main contributions of the work accomplished in this dissertation are:

- Creation of a solution capable of allowing the deployment of lightweight VxFs at the edge of a VANET, by using open hardware platforms with virtualization capabilities which, alongside OBUs, are present inside the vehicles;
- The ability to allow the operation and usage of specific services even in intermittent connectivity situations with the infrastructure, by having such services deployed at the edge of the VANET, on the vehicles themselves;
- The capability of allowing lower delays for critical services, such as in the case of road safety services;
- The ability to use different hardware platforms as the solution's on-boarded hardware, as long as it is compatibility with the solution's required software.

1.3 Document structure

This dissertation is organized into the following chapters:

- **Chapter 2 - State of the Art**: this chapter describes the current state of the art of Vehicular Networks, the mobility protocols used in such networks, the concepts of multihoming as well as Network Function Virtualization (NFV) and its usage;
- **Chapter 3 - Proposed Solution**: this chapter presents the proposed solution, explains its design, how it was implemented, and also all the technical challenges that arose while integrating the NFV concept on a Vehicular Network;
- **Chapter 4 - Evaluation**: this chapter presents the evaluation scenarios and uses cases that were developed in order to evaluate the proposed solution, as well as the obtained results and their discussion;
- **Chapter 5 - Conclusions and Future Work**: this chapter presents the conclusions of the work done and proposes possible goals and improvements for future work.

CHAPTER 2

State of the Art

This chapter covers the fundamental concepts required to understand the work done in this dissertation. Given the fact that this dissertation is based upon both Vehicular ad-hoc Network (VANET) and Network Function Virtualization (NFV) concepts its important to understand all the research already done, specially in the field of VANETs, mobility and NFV. The topics mentioned will be covered and organized as follows:

- **Section 2.1 - Vehicular Networks**: this section presents the main concepts of a VANET, its features, challenges, possible applications and typical architecture.
- **Section 2.2 - Mobility Protocols**: this section presents the main mobility protocols that exist, focusing on explaining how they operate and where they meet or fail to overcome the requirements of a VANET.
- **Section 2.3 - Multihoming**: this section presents some of the multihoming protocols that exist alongside their features and shortcomings. This section will focus more on the multihoming solution that was implemented in the VANET used in this dissertation.
- **Section 2.4 - Network Functions Virtualization**: this section presents an introduction to the concept of NFV, a general overview of the NFV architecture and its main components, as well as some of the practical NFV solutions that exist.
- **Section 2.5 - Related Work**: this section describes the related work on the field of cloud computing and NFV solutions focusing mainly on VANET environments.
- **Section 2.6 - Summary**: this section presents a short summary of the concepts tackled in the present chapter.

2.1 Vehicular Networks

VANETs are a subclass of the Mobile ad-hoc Networks (MANETs), which are a promising approach for future intelligent transportation systems. These types of networks are formed by vehicles and fixed stations placed in specific locations. In order for vehicles to be a part of the network, they must carry an OBU, an entity responsible for connecting the vehicle's occupants to the Internet via an Intra-Vehicle network, as well as, connecting to other vehicles on the network. The fixed stations are called Road Side Units (RSUs), and as the name implies, they are situated alongside roads and enable the communication between other RSUs and OBUs. Given these entities, the VANET supports two main types of communications, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I), the former being supported by the OBUs where they communicate between each other, while the latter being supported by the RSUs that enable communications between OBUs and RSUs (the infrastructure).

2.1.1 Features

VANETs bring a set of features, some of them being quite unique, such as [1]:

- **Predicted mobility** Vehicles are constrained by the road topology and layout, as well as, the requirement to obey traffic laws. If we combine this with the fact that the vehicles can be equipped with Global Positioning System (GPS) technology, it makes it so that it is possible to predict how the vehicle will move and to which locations;
- **No power constraints** Given the fact that an OBU sits inside a vehicle, it can be powered by the vehicle itself, which means that as long as the vehicle has power, the OBU should have as well. This makes it possible to support OBUs with more processing capabilities and power requirements;
- **Dynamic network topology** Since vehicles can move at varying levels of speed, as in a very slow when in a traffic jam or very fast when on a free highway, makes it so that the network topology reacts differently. When moving slow, fewer changes to the network topology are to be expected; the opposite is also true when the vehicle is moving fast;
- **Large Scale Network** If we think of a VANET in an urban setting, it is easy to see that the network can become quite large, given the amount of vehicles that drive in the cities.

2.1.2 Applications

VANETs can provide various types of applications as seen in [1], [6], [7]. These applications can be grouped in safety-related, traffic management and comfort (commercial) applications:

- **Safety Applications** Given the types of communications that VANETs provide, it is easy to send messages to other vehicles or to the infrastructure itself allowing for the creation and use of safety applications that work to improve road safety and avoid accidents;
- **Optimization of Traffic Management** Given the information that a VANET can provide, such as a vehicles' current location, it is possible to better manage how the vehicles navigate in the cities. Some examples of this could be to suggest alternative routes to vehicles in cases of traffic jams or to provide itineraries to points of interest;
- **End-user applications** These types of applications provide users with useful information such as weather, state of the traffic and, for example, the nearest points of interest in the city; the VANET also allows end-users to access the Internet. These types of applications aim to improve drivers and passengers comfort levels. This is all possible due to the connectivity that VANETs provide.

2.1.3 Challenges

Even though VANETs present numerous features and possible applications, several challenges still arise given their nature. Some of these challenges are [1]:

- **Privacy and Security** Given the technologies used in VANETs, keeping the network's traffic private and secure becomes a hard task. This is mainly due to the Wireless Access in Vehicular Environments - IEEE 802.11p (WAVE) technology being used, which has a broadcast nature and does not use authentication, thus allowing for all of the network's traffic to be captured by nodes not in the network;
- **Signal fading and degradation** As before, given the nature of the technologies used in VANETs and given the fact that VANETs are mainly present in urban environments, which often have obstacles between communicating nodes, leads to signal degradation which in turn makes it so that the quality of the wireless communications decreases;
- **Network Fragmentation** Given the high mobility of the nodes that make up the VANET rapid changes of the network's topology can happen, this can lead to network fragmentation.

2.1.4 Architecture

A typical VANET architecture is comprised of several main components. These are [1]:

- *Road Side Units (RSUs)* are the fixed entities that sit at strategic locations allowing the establishment of communications between RSUs and OBUs, therefore enabling the connection between the OBUs and the remaining infrastructure. These entities communicate with the OBUs using different types of technologies such as WAVE, IEEE 802.11 a/g/n (WiFi) or cellular. In the VANET used in this dissertation, the RSUs mainly use WAVE and WiFi technologies;
- *On-Board Units (OBUs)* are the entities that sit inside the vehicles. These entities have communication and processing capabilities, and are responsible for providing the vehicles' occupants Internet access via an intra-vehicle network. This entity communicates with the remaining VANET' infrastructure by connecting itself to RSUs using different types of technologies such as WAVE, WiFi or Cellular. In the VANET used in this dissertation, the OBUs mainly use WAVE and WiFi technologies to connect themselves to the VANET;
- *Application Units (AUs)* are the equipment present inside a vehicle; they can be a dedicated equipment by providing a specific application or service, or be a normal user equipment such as a laptop. The AUs connect themselves to the network exclusively via the OBUs either by a wired or wireless connection.

2.2 Mobility Protocols

To support the deployment of a VANET, given the previously presented features and challenges, a mobility protocol is required. The mobility protocol must be able to support the constant network topology changes, and thus be able to support the movement of connected users and their connections and sessions on the network by keeping track of them. A more extensive list of the mobility protocol's requirements is shown next:

- **Seamless mobility** As noted before, for the end users, the mobility protocol should be transparent, that is, if the vehicle changes its point of access to the network, their connections should be kept with the same quality;
- **Multi-hop Support** Given that RSUs are stationary entities, their range is limited, meaning that OBUs that are not in range cannot connect to the network. A solution to this would be for OBUs that are connected to RSUs to provide connectivity to other OBUs that are not in range of an RSU;
- **Internet Protocol version 6 (IPv6) support** The mobility protocol should be based upon IPv6 has it can support a larger number of nodes, something that is common when talking about a VANET, given that its address space is larger when in comparison with that of Internet Protocol version 4 (IPv4). Moreover IPv6 presents more features namely better security and Quality-of-Service (QoS);
- **Multihoming support** The mobility protocol must support multihoming in order to use all available connections at the time to improve network connectivity and performance;
- **Ease of use** A end-user sitting on a vehicle should not have to worry about how to connect to the network or keep himself connected;
- **Efficient handovers** Given the fact that in a VANET the vehicles can be constantly moving, a high number of handovers is something to be expected, so the mobility protocol should be able to perform them as fast as possible, in an efficient manner.

Another thing to note is that, even though mobility protocols can be seen as centralized, distributed or even hybrid, this section focuses on the centralized protocols, since the protocol used in this dissertation is of this type. In the next subsections, a brief history on the main mobility protocols, their features and shortcomings will be presented. The last subsection will be dedicated to the protocol that was used as the basis of the protocol implemented in the VANET used in this dissertation.

2.2.1 Mobile Internet Protocol version 6 (MIPv6)

The MIPv6 [8] comes as an improvement over its IPv4 version, the Mobile Internet Protocol (MIP) [9]. MIPv6 is based on IPv6 and, for that reason, it takes advantage of its features.

To ensure the mobility for which the protocol was created, the following three functional entities must be present:

- **Mobile Node (MN)** This entity is the node that moves throughout the network while keeping its communications;
- **Home Agent (HA)** This entity is where the MN registers its CoA when changing networks. While the MN is not in his home network, the HA intercepts traffic destined for the MN's home address and tunnels them to the MN registered CoA;
- **Foreign Agent (FA)** This entity is responsible for keeping the MN's HA updated in regards to the MN's current CoA.

The following terminology is also important to understand how the protocol works:

- **Care-of-Address (CoA)** is the address given to the MN while it is visiting a Foreign Network (FN);
- **Binding Update (BU)** is the message sent to the HA by the MNs in order to inform the HA of their new CoAs.

Having had a look at the protocol's entities and important terminology, it is possible to give an overview of how the protocol operates:

1. When a MN moves to a FN, it sends a BU to its HA with the new CoA it got from joining the FN. Upon receiving the BU, the HA stores the MN's new CoA and creates its end of a bi-directional IPv6 tunnel between itself and the MN;
2. On the other hand, the MN creates its end of the IPv6 tunnel between itself and his HA. With the tunnel set up, traffic that arrives at the HA which is destined to the MN is routed via the tunnel.

Given the presented overview of the protocol, it is possible to conclude that this protocol is not suitable for a VANET. This protocol provides terminal mobility but does not handle network mobility, which is a crucial VANET requirement. Another set of problems present in this protocol are the high latency values when performing handovers, the signaling overhead as well as packet loss [10]. These aspects are not acceptable for a VANETs and its applications.

2.2.2 Proxy Mobile Internet Protocol version 6 (PMIPv6)

The PMIPv6 [11] came as an improvement to the MIPv6 protocol, the main improvement being that now the MNs do not need to be involved in the exchange of signaling messages between themselves and their HA. In order to present an overview of the protocol, it is important to mention its new entities and most important terminology. Figure 2.1 shows an overview of the PMIPv6 network architecture.

This protocol improves on the previous iteration by adding the following two entities:

- **Local Mobiliy Anchor (LMA)** This entity provides the same functional capabilities of the HA present in MIPv6 while, at the same time, supporting new capabilities such as the management of the MNs' binding states as well as the routing process;
- **Mobile Access Gateway (MAG)** This entity provides connectivity to the MNs (it acts as an RSU as explained in 2.1.4), and is responsible for tracking the MNs' movements and for informing the LMA about the MNs' mobility related aspects.

The following terminology is also important to understand how the protocol works:

- **Binding Cache Entry (BCE)** is a cache entry that keeps information about the MNs connected to the network;
- **Proxy Binding Update (PBU)** is a message that is sent by a MAG to the LMA with the intention of informing the LMA about a MN binding intention;
- **Proxy Binding Acknowledgement (PBA)** is a message that is sent by the LMA to the MAG in response to a PBU message.

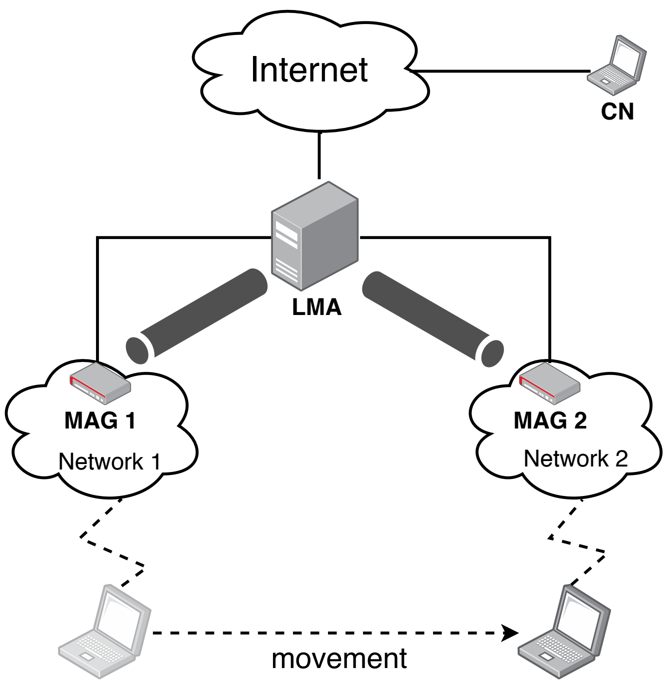

Figure 2.1: PMIPv6 protocol overview.

Having had a look at the protocol's new entities and important terminology, it is possible to give an overview of how the protocol operates:

1. When a MN moves into the range of a new MAG, it sends a Router Solicitation (RS) message to the MAG in order to initiate the connection process; upon receiving a RS from the MN, the MAG starts its binding process.
2. The MAG sends a PBU to the LMA: if the LMA accepts the PBU, it creates a new BCE for the MN, it creates a bi-directional IPv6 tunnel between itself and the MAG, and finally it sends a PBA to the MAG.
3. When the MAG receives the PBA, it creates its end of the bi-directional IPv6 tunnel between itself and the LMA and performs all the necessary configurations in order to connect the MN to the network; the MAG also sends a Router Advertisement (RA) message to the MN in order for it to configure its network interface.
4. When a MN disconnects from a MAG, the MAG will detect the disconnection and will start the process of removing the binding state for that MN, and it will inform the LMA of the MN's disconnection. After a timeout, the LMA will remove the MN's BCE. If the IPv6 tunnel that is created between the MAG and LMA is not being used by any other MNs, it is also removed.

This protocol still does not meet all the requirements of a VANET, specifically the requirement that the network moves alongside the MN. In this protocol the MNs are the only entities that have mobility: they can move throughout the networks provided by the MAGs which are static.

2.2.3 Network Mobility (NEMO)

The NEMO [12] protocol came as an extension to MIPv6 in order to enable network mobility. The NEMO protocol enables network mobility by allowing a network and its users to keep their sessions, even when the network changes its point of access to the Internet. In order to present an overview of the protocol, it is important to mention its new entities and most important terminology.

This protocol introduces the two following entities:

- **Mobile Router (MR)** This entity is a router capable of changing its Point of Attachment (PoA) to the Internet; it provides a network (the mobile network) allowing devices to connect to it and it also serves as the device's gateway to the Internet;
- **Access Router (AR)** This entity is responsible for serving the MR, e.g. give it Internet access.

The following terminology is also important to understand how the protocol works:

- **Mobile Network Node (MNN)** is the name given to the devices connected to the mobile network provided by the MR;
- **Mobile Network Prefix (MNP)** is the IPv6 prefix assigned to the MR's mobile network; all nodes present in the network have this prefix.

Having had a look at the protocol's terminology and entities, it is possible to give an overview of how the protocol operates based on Figure 2.2:

1. When a MR connects to a new AR, the AR replies back with a RA message that contains the MR's new Care-of-Address (CoA) and Mobile Network Prefix (MNP) to be used in its mobile network. The MR then sends a Binding Update (BU) to its HA in order for it to store its new CoA.
2. Upon receiving the MR's CoA, the HA creates a new Binding Cache Entry (BCE) with the MR's CoA and MNP that makes it possible to redirect the traffic to the mobile network via the MR's CoA. The HA then creates a bi-directional IPv6 tunnel between itself and the MR, just before sending a Binding Acknowledgement (BA) to the MR.
3. Upon receiving the BA, the MR creates its end of the bi-directional IPv6 tunnel between itself and the HA. Once the tunnel is established, any MNN that wants to communicate with a Correspondent Node (CN) will do that so using the tunnel; the same thing happens when a CN wants to communicate with a MNN.

This protocol provides network mobility, a critical VANET requirement that was missing in the previous protocols, but it still has its problems. One of those problems, shown in [13], is the performance limitation of the protocol when it comes to scenarios which are highly dynamic, as in the case of VANETs. Another problem, shown in [14], is the high latency values of the handovers when the communication links experience instability.

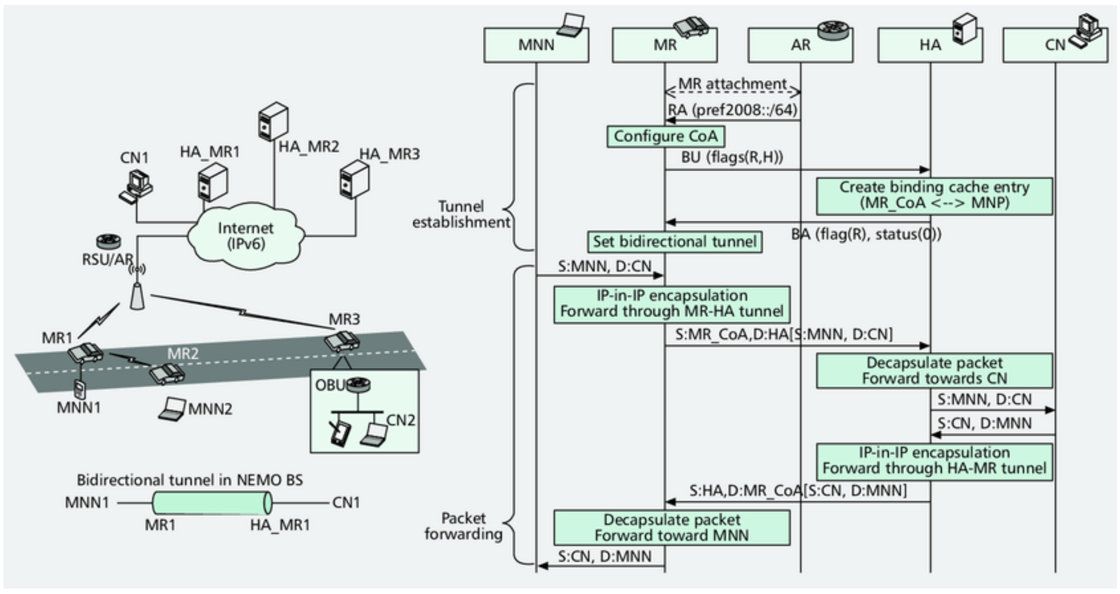
MNN
MR
AR
HA
CN
MR attachment
RA (pref2008::/64)
Configure CoA
BU (flags(R,H))
Create binding cache entry (MR_CoA <--> MNP)
BA (flag(R), status(0))
Set bidirectional tunnel
Tunnel establishment
S:MNN, D:CN
IP-in-IP encapsulation Forward through MR-HA tunnel
S:MR_CoA,D:HA[S:MNN, D:CN]
Decapsulate packet Forward towards CN
S:MNN, D:CN
S:CN, D:MNN
IP-in-IP encapsulation Forward through HA-MR tunnel
S:HA,D:MR_CoA[S:CN, D:MNN]
Decapsulate packet Forward toward MNN
S:CN, D:MNN
Packet forwarding
CN1
HA_MR1
HA_MR2
HA_MR3
Internet (IPv6)
RSU/AR
MR1
MR2
MR3
MNN1
MNN2
OBU
CN2
Bidirectional tunnel in NEMO BS
MNN1
CN1
MR1
HA_MR1

Figure 2.2: NEMO protocol operation overview (from [13]).

2.2.4 Network PMIPv6 (N-PMIPv6)

The N-PMIPv6 is a protocol that builds upon the original PMIPv6 by extending it to support network mobility, that is accomplished by following the approach presented by the NEMO protocol. Given the fact that this protocol builds upon PMIPv6, the major entities are the same, the only difference is the addition of a new entity:

- **mobile MAG (mMAG)** This entity merges the MAG present in PMIPv6 with the MR present in NEMO. This entity is capable of providing a mobile network to MNs while being able to change its PoA to the network.

Having introduced the new entity in this protocol, it is possible to give an overview of how the protocol operates:

1. The mMAG operates the same way a MN does in PMIPv6. This means that, when a mMAG moves into the range of a new MAG (or mMAG), it sends a RS message to that MAG (or mMAG). Once the RS is received, it starts the binding process for the connecting mMAG.
2. The MAG (or mMAG) sends a PBU to the LMA with information on the connecting mMAG. Upon receiving the PBU, the LMA checks the PBU message to see if the message was sent from a MAG or a mMAG. It then creates a new BCE and assigns a new prefix to the connecting mMAG. This prefix is then sent back to the mMAG via the PBA message. The LMA then creates its end of a bi-directional IPv6 tunnel between itself and the MAG (or mMAG).
3. Upon receiving the PBA, the MAG (or mMAG) creates its end of the bi-directional IPv6 tunnel and sends a RA to the connecting mMAG informing it of its new network prefix in order for it to configure its network interface.
4. When a user wants to connect itself to the network, it does that via the mMAG. The connection process for the user is the same process shown above.

This protocol is the most complete of the ones presented. It provides network mobility alongside other important requirements of a VANET. This protocol allows end-users to connect themselves directly to the network served by the mMAG without having to perform any specific configurations. The fact that the mMAG aggregates various users under its network means that, when it performs a handover, only the mMAG itself (as opposed to each of the users) needs to communicate with the LMA regarding the network's movement. This means that the latency of the handover process is greatly reduced.

2.3 Multihoming

In a VANET environment, the nodes have at their disposal various types of wireless technologies to choose from when connecting to the network, but they only connect themselves via one of those technologies, making it so that the other options are not being taken advantage of. An obvious solution to this problem would be to find a way to use all of the available connections.

By incorporating multihoming in a VANET environment, it is possible to make use of all available network access technologies simultaneously in order to provide better network performance, reliability and load sharing [15], [16].

In the next subsections, a brief overview of some of the multihoming protocols alongside their features and shortcomings will be presented. The last subsection will be dedicated to the multihoming protocol that was developed by the Network Architectures and Protocols (NAP) [1] research group and is present in the VANET used in this dissertation.

2.3.1 Stream Control Transmission Protocol (SCTP)

The Stream Control Transmission Protocol (SCTP), much like the Transmission Control Protocol (TCP), is a connection-oriented protocol that provides message-oriented data transfers like the User Datagram Protocol (UDP), that operates in the transport layer. One of the more important features of SCTP is the multihoming support. At the beginning of a SCTP connection both the endpoints exchange all the possible paths through which they can communicate with each other, but this does not mean that all the paths will be used in the communication process. SCTP uses one of those paths as the primary path, while the others stay as backups. This means that the multihoming is used as a backup mechanism in this protocol. If we think of the multihoming needs of a VANET, this protocol does not provide a proper solution.

2.3.2 Site Multihoming by IPv6 Intermediation (SHIM6)

SHIM6 [17], [18] is a protocol that enables IPv6 multihoming, allowing multihomed hosts to make use of all the available paths to reach the network. The protocol was designed as a new sub-layer inside the IPv6 network layer. Since the protocol gives the end-hosts a way to manage their paths, they can choose which one is the best for them at any given time. This protocol does indeed provide multihoming, but it does not meet all the multihoming requirements of a VANET,

[1] `https://www.it.pt/Groups/Index/36`

as can be seen by the inability to decide what type of traffic flows through which available paths, which in turn means that it is not possible to provide differentiated load balancing.

2.3.3 Proxy multihoming as PMIPv6 extension

Proxy multihoming as a PMIPv6 extension was the name given to the multihoming extension developed by the NAP research group [19], [20] for PMIPv6. The solution works by using PMIPv6 as the protocol responsible for the mobility of the network, while the developed multihoming solution is responsible for managing the multihoming process itself. This is possible due to the addition of a set of new entities to the ones already present in PMIPv6, namely to the LMA and the MAGs.

An overview of this multihoming solution's framework is depicted in Figure 2.3. In order to better understand it, a brief overview of its components will be given.

On the LMA, the following entities are present:

- **Terminal Manager (TM)** is the entity that was developed to overcome the PMIPv6's inability to correctly identify a single terminal which is connected via different interfaces. Since PMIPv6 uses the Media Access Control (MAC) address of the connecting interface as the user's identifier, when a terminal connects itself using different interfaces, it assumes that each connecting interface is a different user. The TM is then responsible for managing a User Cache Entry (UCE) that will contain the terminals' identifiers, their number of connected interfaces, as well as information about each one of their connected interfaces. The cache is updated whenever a terminal's interface is connected or disconnected from the network.
- **Flow Manager (FM)** is the entity that was developed to overcome the PMIPv6 inability to choose through which router the terminal's traffic should be sent. The FM was then developed as a way to intelligently and dynamically route the terminal's traffic. This entity is responsible for managing a Flow Cache Entry (FCE) that will contain information on all the flows associated with a terminal, it is also responsible for the distribution of the traffic and the calculation and application of the multihoming rules based on the information gathered by the multihoming solution's entities.
- **Information Manager (IM)** is the entity responsible for keeping updated information about the environment. The FM requests information from this entity when it needs to determine multihoming rules.

On the MAG, the following entity is present:

- **Network Information Server (NIS)** is the entity responsible for providing and storing information regarding the terminals' points of attachment to the access network as well as their wireless characteristics. This entity is also responsible for providing this information to the IM.

The final entity is present on the multihoming users:

- **User Information Server (UIS)** is the entity responsible for keeping a database of updated information regarding the terminals' interfaces such as: interface name, RSSI, packet loss, noise level, throughput and the PoA load. This entity has the ability to communicate with the NIS, and thus it provides the NIS with the information it has stored.

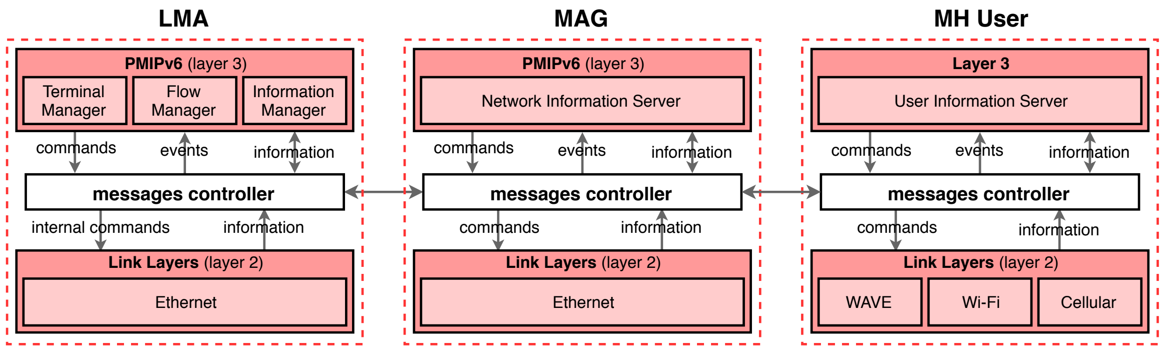

Figure 2.3: Multihoming framework overview (based on [19]).

The developed multihoming solution takes into account various real-time parameters of the traffic flows, the characteristics of the access technologies as well as the characteristics of the end-user's equipment. This multihoming solution was later integrated with the N-PMIPv6 protocol instead of the PMIPv6, culminating in a solution [21] that makes use of all the features and improvements present in both N-PMIPv6 and the presented multihoming solution.

2.4 Network Function Virtualization (NFV)

In a typical network environment, the network functions are provided by a network equipment (like switches, routers, etc.). These equipment are made out of a combination of proprietary software and hardware whose only job is to perform a specific task. The communication networks and services provided by Network Service Providers (NSPs) are made up of proprietary equipment, each one of them performing their task, resulting in network functions and services which are not very flexible.

If a new service needs to be implemented or a service is reaching its maximum capacity, the NSPs need to add new network equipment or replace old equipment with newer, more powerful and capable equipment, in order to keep up with the demand. This is something that is neither efficient nor cheap for the NSPs, resulting in having more equipment, which takes more space, consumes more power, meaning more costs, both in terms of Operational Expenditure (OPEX) and Capital Expenditure (CAPEX), which cannot be covered by increasing the service's fees without the risk of losing clients.

To come up with a solution to this problem, NSPs came together to think of a way to replace dedicated networking equipment with more flexible alternatives still capable of performing networking tasks. If we think about it, today's networking equipment, which perform very distinct tasks, have very similar technology when it comes to their hardware. The real difference presents itself in the form of the software that runs on top of such hardware, which implements the networking functions and services. So the NSPs got to the conclusion that, instead of having dedicated networking equipment, it is possible to run the software that performs the equipment's tasks on powerful general purpose hardware (i.e. servers): this is what in essence NFV is.

NFV achieves virtualized functions by virtualizing general purpose hardware, allowing for the use of Virtual Machines (VMs). Then, the virtualized versions of the software that runs on the dedicated network equipment, the Virtual Network Functions (VNFs), can be instantiated on the general purpose hardware with the help of the VMs. This means that a single computer can run a number of these VNFs, thus replacing several dedicated network equipment. Given the fact that the functions are all virtualized, they can quickly and easily be taken down or swapped from one VM to another when they require more resources such as processing power.

When talking about the development of NFV technologies, European Telecommunications Standards Institute (ETSI) is the most prominent name, considering it has created an Industry Specification Group (ISG) for the field of NFV in order to create an architecture capable of supporting VNFs in a consistent manner. The ISG is comprised of several of the most important telecommunication carriers, such as Deutsche Telekom, Telefónica, AT&T, Orange and more. This group is leading the standardization in the field of NFV, in particular with their definition of an NFV reference architecture framework [2] for NFV solutions (shown in Figure 2.4).

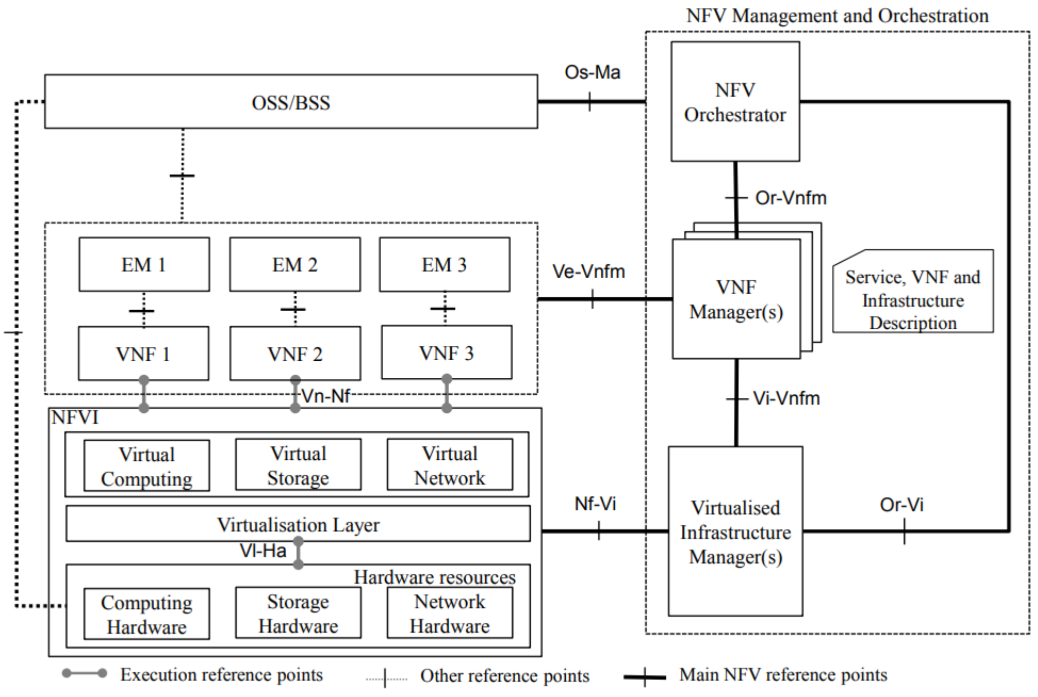

Figure 2.4: NFV architectural framework (from [2]).

2.4.1 NFV Management and Orchestration (MANO)

One of the architectural frameworks defined by ETSI is the MANO architectural framework [22] (depicted in Figure 2.5) which is the main component of the NFV architecture, and thus, responsible for, as the name indicates, the management and orchestration of the NFV solution. This section will give a brief overview of the MANO architectural framework's main components.

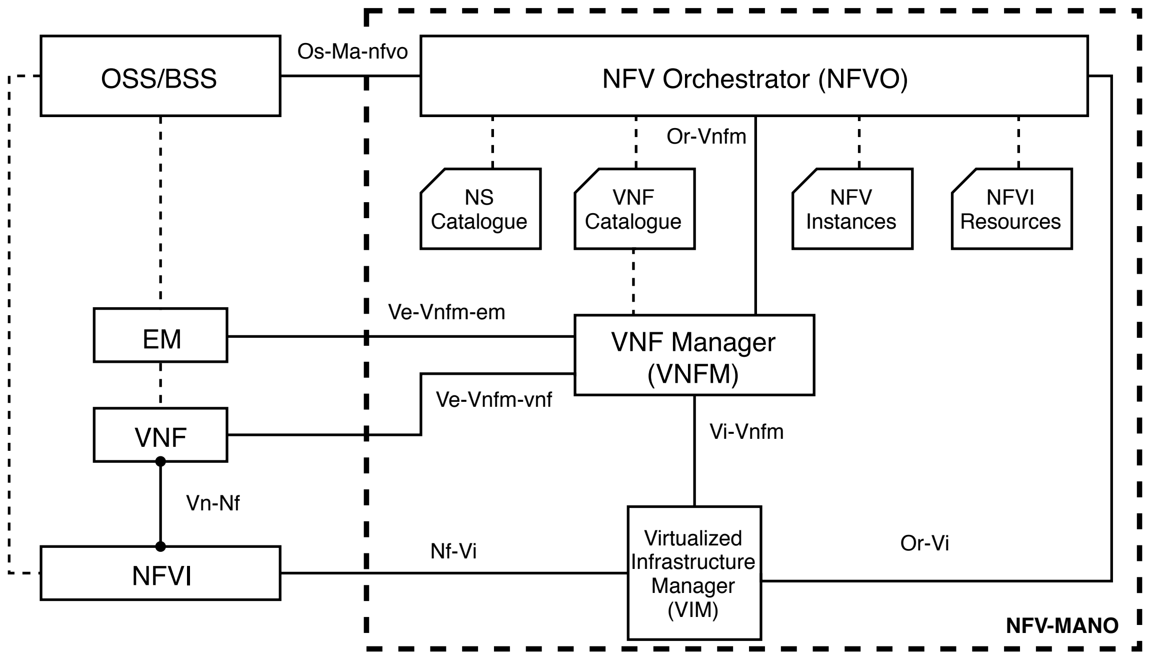

Figure 2.5: NFV-MANO architectural framework (based on [22]).

As can be seen in Figure 2.5, MANO is made up of four main blocks:

- **Virtualized Infrastructure Manager (VIM)** is responsible for the management of the NFVI compute, storage and network resources in a network's domain, such as being responsible for the allocation of NFVI resources to VMs, keeping a repository with information about which NFVI resources are being used by what VMs, etc. It is worth noting that multiple VIMs can be present, each one being responsible for a different network domain.
- **VNF Manager (VNFM)** is responsible for the management of VNF instances, such as being responsible for their life-cycles (e.g. instantiation, termination) and for their Fault, Configuration, Accounting, Performance and Security management (FCAPS). A single VNFM can manage multiple VNFs or various VNFMs can manage multiple VNFs.
- **NFV Orchestrator (NFVO)** has two main responsibilities, the first one being the orchestration of NFVI resources across multiple VIMs given that, as mentioned above, there can be more than one VIM managing different NFVIs; and the second one being the life-cycle management of Network Services (NSs). Given the fact that VNFs can be managed by multiple VNFMs, the NFVO has to manage the creation of the NSs that involve VNFs from different VNFMs.
- **Repositories** are the last main block; it is made up of four types of repositories:

  VNF Catalog is the repository of all the on-boarded VNF packages. It supports the management of the VNF packages (which contains the VNF Descriptor (VNFD), software images, etc). The information present in the VNFD can be queried by both the NFVO and VNFM to support different operations.

  NS Catalog is the repository of all the on-boarded NSs. It supports the creation and management of the NS deployment templates, the NS Descriptors (NSDs). These templates describe the NS in terms of their VNFs and their virtual links.

NFV Instances is the repository that holds information on all NSs instances and their VNF Instances.

NFVI Resources is the repository that holds information about the NFVI resources that are currently either available, reserved or allocated.

In addition to these four main blocks that make up the MANO, there are four more entities outside of the MANO's scope, which are:

- **Operations and Business Support System (OSS/BSS)** refers to the OSS/BSS of a network operator. Operation Support System (OSS) is responsible for dealing with the network, fault, configuration and service management, while the Business Support System (BSS) is responsible for dealing with customer, product and order management. The OSS/BSS and the NFV are supposed to cooperate.
- **Element Management (EM)** is the entity responsible for the FCAPS of the functional side of the VNFs. It is worth noting that, as mentioned above, the VNFM also does the FCAPS for the VNFs but for their virtual side.
- **Virtual Network Functions (VNFs)** can be seen as the basic block of the NFV architecture. They are the virtualized versions of the software that runs on top of the NFVI.
- **NFV Infrastructure (NFVI)** provides the environment in which the VNFs run. This environment is made up of the compute, storage and networking resources, as well as a virtualization layer. This layer is what is responsible for abstracting the actual physical resources into virtual resources; this layer is commonly called the *hypervisor*. The VNFs are then instantiated in the form of VMs on the *hypervisor*.

2.4.2 NFV solutions

Having given a brief overview of the MANO architectural framework's components, it is possible to present the main relevant NFV projects when it comes to the development of solutions which are based on the MANO architectural frameworks.

Main VIM solutions

- **OpenVIM** [2] is a light implementation of a VIM, it communicates with the NFV Infrastructure (NFVI) and an OpenFlow controller in order to provide computing and networking capabilities, as well as the ability to deploy the VMs. The project is currently part of the Open Source MANO (OSM) project, a project that implements an open source MANO solution aligned with the ETSI's architectural frameworks. It is worth noting that, even though this solution is open source, it is limited in terms of features, and usually is only used when in conjunction with the OSM project to deploy a quick all-in-one solution used for development.
- **VMware** is a company that provides cloud computing and virtualization products. One of those products is the VMware vCloud NFV [3]. This product is a commercial implementation

[2]`https://github.com/nfvlabs/openvim`

[3]`https://www.vmware.com/products/vcloud-director.html`

of a MANO solution aligned with the ETSI's architectural frameworks. The actual VIM solution provided by the product is the VMware vCloud Director, which is a multi-tenant solution that provides the management of the computing, networking and storage resources. Being a commercial product, this solution provides dedicated end-user support and is very stable.

- **Openstack (OS)** [4] is an open source software solution for cloud computing. It consists of several interrelated services, which are deployed as components and are responsible for managing the computing, networking and storage resources. It also contains other components providing more services, such as a dashboard which enable the management of the OS services. OS is the most prominent open-source VIM solution: it is a very stable and mature project, given the fact that it is used by many renowned companies to support their services.

Main MANO solutions

- **OpenBaton** [5] is an open source platform that provides an implementation of the ETSI NFV MANO architecture framework. Its architecture can be seen in Figure 2.6. This platform is composed of several components which provide a number of different services, some of the most important ones are:
  - An NFVO which was completely designed and implemented following the ETSI MANO specifications;
  - A driver mechanism that allows different types of VIMs without the need to change anything in the orchestration logic;
  - A generic VNFM able to manage the life cycle of VNFs based on their descriptors;
  - A Juju VNFM Adapter that allows the deployment of Juju Charms [6];
  - A monitoring plugin which integrates Zabbix as monitoring system;
  - A fault management system which can be used for automatic runtime management of faults;
  - A Docker VNFM and VIM driver for instantiating containers on top of the Docker Engine.

  Given the fact that OpenBaton has a driver mechanism that allows for different types of VIMs, it is possible to integrate it with the VIM solutions presented above, such as OS. It is a project developed by Fraunhofer FOKUS and TU Berlin, and it represents one of the main components of the 5G Berlin initiative.
- **Open Source MANO (OSM)** [7] is an ETSI hosted production-quality open source MANO stack aligned with ETSI NFV architectures. It is an operator-led ETSI community, which includes among its members Bell, Telefónica, RIFT.io, Canonical, etc. Its architecture and components can be seen in Figure 2.7, a brief overview of each of them is given next:

[4] `https://www.openstack.org/`

[5] `https://openbaton.github.io/`

[6] Juju is a service orchestration tool from Canonical that aids in the configuration and deployment of services through the use of a set of scripts called Charms. Further explanation present in `https://jaas.ai/how-it-works/`

[7] `https://osm.etsi.org/`

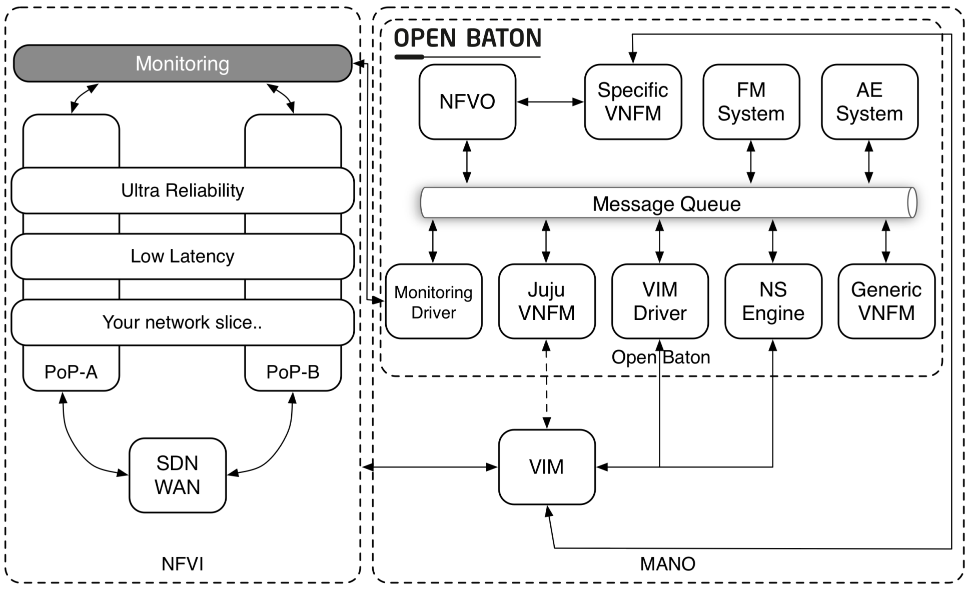

Figure 2.6: OpenBaton architectural framework.

- **Network Service Orchestrator (NSO)** The NSO started with the RIFT.ware solution as its seed, having expanded on it. This component is able to interact with the Resource Orchestrator (RO) and VNF Configuration and Abstraction components of the OSM solution, in order to support the management of the Network Services (NSs), the management of the MANO repositories (presented in subsection 2.4.1), as well as the process of on-boarding and configuring NSs and their VNFs.
- **Resource Orchestrator (RO)** The RO started from the OpenMANO, another MANO solution developed by Telefónica, which was later contributed into OSM. This component is responsible for coordinating the compute, storage and network resources that are required for the instantiation of the NS's VNFs. It does that by interacting with the VIM, which is the component that actually manages the resources. The RO can also interact with multiple VIMs. It provides most of the functions of the NFVO defined by the ETSI MANO framework.
- **VNF Configuration & Abstraction (VCA)** This component is responsible for providing the functions of the VNFM defined by the ETSI MANO framework. It is responsible for the configuration of VNFs given the instructions present in the VNFDs. It is based on Canonical's Juju Charms [8].
- **GUI & Design-Time Tools** This component is, as the name suggests, the user interface that presents users with mechanisms to interact with the NSO. It gives users the ability to on-board VNFs and NSs (via their descriptors), as well as the ability to launch NSs.

• **RIFT.ware** [9] is a NFV MANO solution which is a commercial distribution of the OSM

[8]`https://jaas.ai/how-it-works/`
[9]`https://www.riftio.com/riftware/`

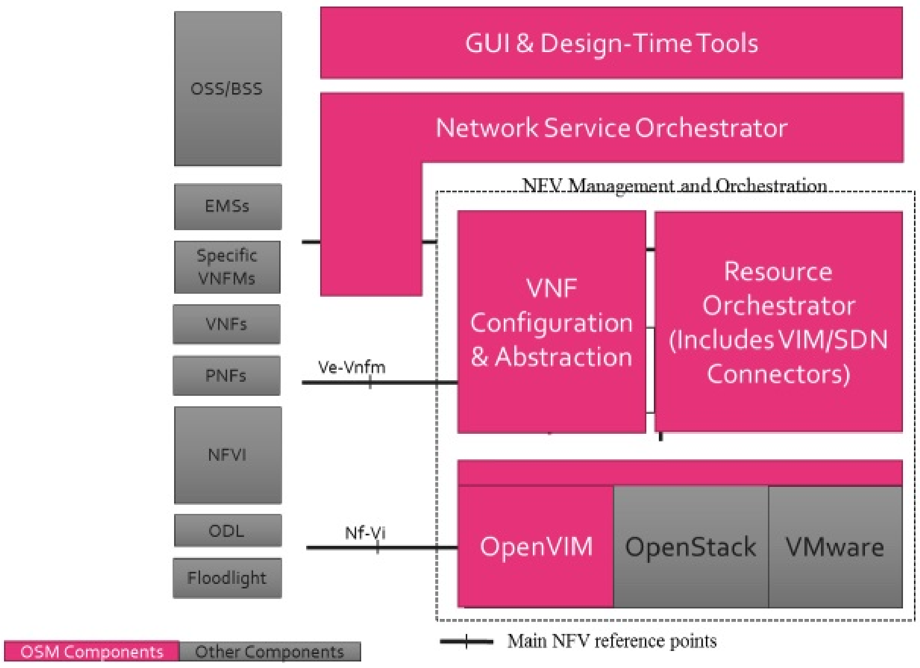
GUI & Design-Time Tools
Network Service Orchestrator
NFV Management and Orchestration
OSS/BSS
EMSs
Specific VNFMs
VNFs
PNFs
NFVI
ODL
Floodlight
VNF Configuration & Abstraction
Resource Orchestrator (Includes VIM/SDN Connectors)
Ve-Vnfm
Nf-Vi
OpenVIM
OpenStack
VMware
OSM Components
Other Components
Main NFV reference points

Figure 2.7: OSM Mapping to ETSI NFV MANO (from [23]).

project. It aims to simplify the deployment of multi-vendor NSs, and VNFs in carrier networks and enterprise clouds. It offers everything needed in order to have automated end-to-end service delivery and life cycle management. RIFT.ware is a product offered by Rift.io, a founding member of the OSM project, which has greatly contributed to its progress, and as seen above, its most important contribution is the OSM's NSO. All in all, it is a well established commercial product in the field of MANO.

2.5 Related Work

During the last few years many authors have investigated ways to bring the cloud concepts to the scope of VANETs.

Olariu *et al.* [5] envisioned the idea of cloud computing in the scope of VANETs. They show that vehicles are under-utilizing their on-board computation, communication, and storage resources, and that these can be shared among drivers or rented over the Internet to other customers, mimicking the concept of cloud resources. The authors then provide their vision on the concept of Infrastructure-as-a-Service (IaaS) in a VANET, where they discuss several use cases on how to better utilize the vehicle's resources, but they do not discuss a potential structural framework for Vehicular Clouds (VCs).

Hussain *et al.* [4] follow up on the previous work by actually proposing potential framework architectures for different types of cloud scenarios in VANETs. The authors divide VANET clouds into three major architectures namely Vehicular Cloud (VC), Vehicles using Clouds (VuC), and Hybrid Vehicular Clouds (HVC). The VCs are sub-divided into two types, static and dynamic: the former refers to the stationary vehicles providing cloud services (parking lots, etc) while the latter refers to clouds that are formed on demand in an ad hoc manner. When talking about VuC, they connect VANETs to traditional clouds where the VANET users can make use of the services hosted on the cloud while on the move. As for the HVC, they are a combination of both the VC and VuC where they serve as consumers, since they use services from the cloud, and as producers where the vehicles themselves provide their resources.

A pure VC solution was proposed by Zingirian *et al.* [24] where a cloud made up of only vehicles was developed. It works under a new experimental service modality for vehicular platforms, called Sensor-as-a-Service (SenaaS), that makes third-party entities have access to the vehicles' devices and sensors. This then means that third-parties can then take advantage of those resources to create various types of applications, such as vehicle monitoring applications.

One example of a VuC solution is the cloud-assisted system for autonomous driving developed by Kumar *et al.* [25] called Carcel. This solution works on the basis of information sharing to allow for better decisions: the autonomous cars effectively share their sensor's information with a cloud in order to receive information which helps them define safer and more efficient paths. The solution's cloud is based on two modules: the request module which is responsible for gathering the vehicle's sensors information, and the planner module which aggregates the received data, analyses it in order to detect obstacles that may be in the path of the vehicles, and then sends that information alongside alternate paths to the vehicles.

One interesting HVC solution was presented by Bitam *et al.* [26], the authors call it VANET-Cloud. Their proposed VANET-Cloud model is made up of three layers, there is the client layer which is formed by the end-users of the cloud, be it a general user (no in the VANET) or a VANET entity; the actual cloud layer that refers to the stationary data centers which provide the VANET-Cloud services; and finally the communication layer which ensures communication between the previous layers via different ways, such as Internet gateways, VANETs, 4G networks, satellite, etc. The solution can support a variety of services, some of them being road safety services that allow users to deal with scenarios such as accidents or collisions, or more end-user oriented services such as web services. The authors also briefly touch on the subject of security and privacy which will be a big concern in the future.

In [27], Zhu *et al.* present a solution which makes use of both cloud and NFV concepts. Currently many automotive companies release vehicles with an Intelligent Onboard System (IOS), which allows vehicles to have access to various types of services, such as updates on the current traffic situation, services based on the vehicle's current location such as nearby gas station, etc. The problem with the IOSs is that they have a closed architecture, meaning that, they cannot easily be upgraded to support new services. The authors propose a solution to this problem based on NFV technologies, that can transform these IOSs into more open platforms capable of easily being upgraded.

Another NFV solution that was developed for a different type of MANET is the solution presented by Nogales *et al.* [28], where the authors designed an NFV system capable of deploying Network Services (NSs) over a cloud platform composed of an infrastructure of Small Unmanned Aerial Vehicles (SUAVs). The authors used the ETSI's OSM with Openstack (OS) as the VIM, where the SUAVs made up the NFVI. The results presented by the authors suggest that their solution is capable of deploying NSs over the limited resources that the SUAVs platforms have, with good performance as was shown with the case of a Voice over Internet Protocol (VoIP) NS.

The solution developed in this dissertation focuses more on the concept of extending the cloud to the edge of the VANET: it can be seen as a combination of the concepts described in the last two articles presented. Much like in [27], this dissertation aims to develop a solution which will enhance the VANETs, making it possible to easily deploy flexibly NSs in a vehicular scenario. The main difference is that in [27] the authors' solution focuses on the hardware already present on the vehicles, the IOS, in order to transform them into more open platforms. On the other hand, in this dissertation the focus is on developing a more open and flexible solution which does not revolve around a specific hardware platform: our solution is capable of using various types of hardware platforms. As for the the solution presented in the last article [28], the NFV concepts used in the work developed in this dissertation are similar to the ones presented in [28]. The solution in [28] focuses on the deployment of its solution in a MANET infrastructure made up of SUAVs, where in this dissertation the focus is on deploying a similar solution over a different kind of MANET, a VANET.

2.6 Summary

This chapter presented an overview of the topics that encompass this dissertation. The concept of VANETs was introduced alongside its features and challenges. Then, the mobility protocols which are used in VANETs were presented culminating with the N-PMIPv6 protocol, which is the mobility protocol used by the VANET considered in this work. Several multihoming approaches were also presented, with emphasis on the solution currently in use by the aforementioned VANET, the Proxy multihoming as PMIPv6 extension. An overview of NFV technologies and its architectures was presented, as well as the main projects and solutions which are aligned with such architectures. Lastly, the related work to the one performed in this dissertation has been presented and discussed.

CHAPTER 3

Proposed Solution

This chapter will cover the main aspects of the solution developed in this dissertation. The two main sections will discuss the solution's design and implementation. Section 3.1 will discuss the steps and choices taken when coming up with the solution's design, while Section 3.2 will focus on the technical aspects and challenges of the solution's implementation, given the design presented.

3.1 Solution Design

This dissertation studies the applicability of NFV technologies in a VANET scenario, as shown in Chapter 1, whose main goal and motivation is to develop a solution capable of supporting the automated deployment and configuration of moderately complex services that enhance the VANET's end-users experience. These services are implemented by VxFs, which can be seen as the abstraction of VNFs. Unlike VNFs that specifically implement network functions, VxFs can implement any type of virtual function, thus they can provide more diverse services that can range from security to entertainment. With this objective in mind, the use of NFV technologies, along side general purpose hardware on-boarded in the vehicles, will allow the solution to use the hardware's resources and virtualization to provide varied services with the help of the VxFs.

By using NFV technologies in a VANET environment, the solution offers some advantages, such as: (1) the ability to promptly deploy services on the vehicles, by quickly instantiating and configuring the VxFs; (2) the capability to adapt and change the services deployed on the vehicles according to different situations (i.e. deploy specific security service in case of emergency or deploy entertainment service when on a long trip); (3) unlike other solutions, it provides an open platform where developers can experiment with the development of different types of services that can be useful in a vehicular environment; (4) by using an open platform which permits quick changes to the services that are deployed, it is possible to keep up with different vehicular scenario's needs by incorporating new services or scaling existing ones; (5) the use of general purpose hardware means that it is possible to swap or expand it when required without many costs.

But like most solutions, even though it provides several benefits, it also presents some downfalls and challenges which are discussed throughout the next subsections.

An overview of the proposed solution is depicted in Figure 3.1, where its design relies mainly on the following two components:

1. The MANO system, located outside the VANET's scope, is the component that supports both the management and orchestration of the NFVI and the NSs;
2. The hardware on-boarded on the vehicles alongside the OBUs which supports the execution of the NSs and VxFs.

Another important component of the proposed solution comprises the virtual networks that will sit on top of the pre-existing VANET's mobility network. These virtual networks will allow the communication between every element of the proposed NFV solution. The next subsections will give a more detailed overview of each of the components that make up the solution, as well as justify the choices that were made.

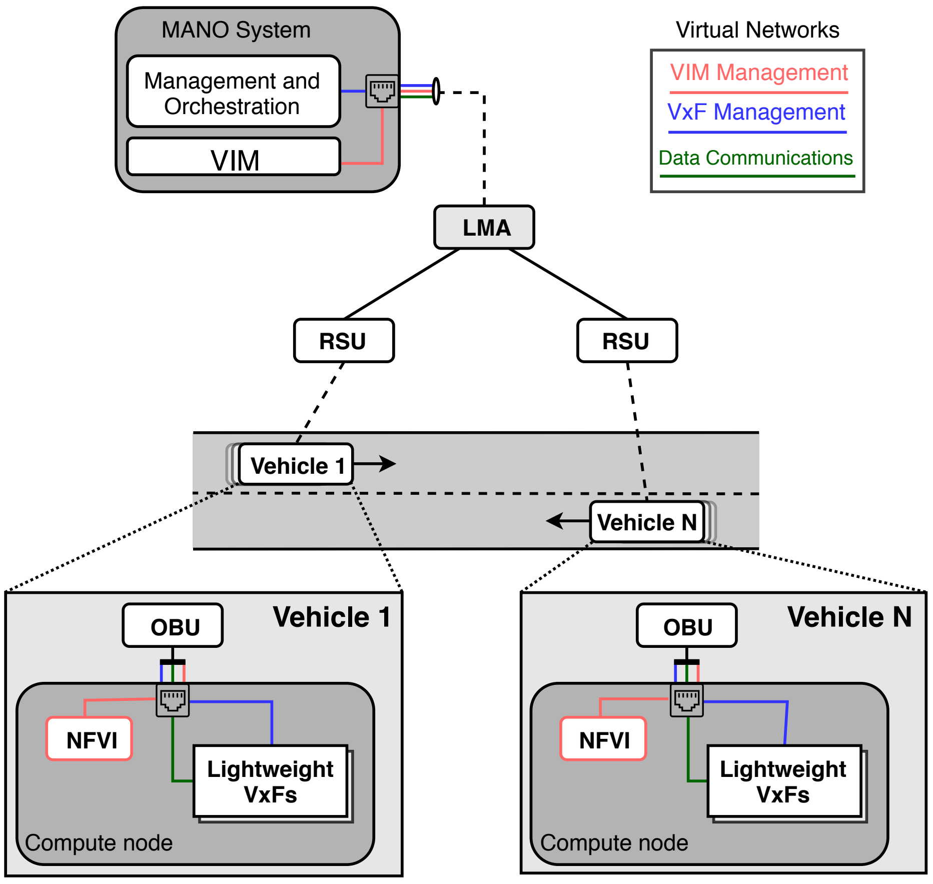

Figure 3.1: Overview of the proposed solution's architecture.

3.1.1 Base VANET Architecture

The first thing to take into consideration when thinking about how to design the solution is over what kind of VANET will the solution be deployed, what protocols will take care of its mobility, what are the constraints imposed by the VANET's architecture, what are the constraints imposed by its mobility protocol, etc. Such questions will be discussed throughout this chapter. Taking that into consideration, a brief presentation of the VANET's architecture used in this dissertation is given next.

The VANET over which the solution is to be deployed is entirely based upon the mobility solution developed in our research group, a solution based on the Network PMIPv6 (N-PMIPv6) architecture (presented in Section 2.2.4), with additional mechanisms developed to support transparent handovers and simultaneous multihoming [19], [21]. As previously mentioned in subsections 2.1.4 and 2.2.2, the main entities of such a network are the LMA, the RSUs and the OBUs. These are the actual physical components which make up the VANET and where the mobility software runs. A simple representation of the VANET is depicted in Figure 3.2.

Another main aspect of this VANET, given the mobility protocol it uses, is the IPv6 support. In this protocol, IPv6 tunnels are created between the network nodes by the mobility protocol, which are used for protocol and data communications. As can be seen in Figure 3.2, the tunnels between the LMA and the RSUs are IPv6/IPv6 tunnels, while the tunnels from the LMA to the OBUs are IPv4/IPv6 tunnels. This is worth mentioning because traffic going from the LMA to the OBUs, and the other way around, will have an additional double tunnel header, as it has to go through both tunnels. This presented itself as a constraint when preparing the NFVI at the edge of the network.

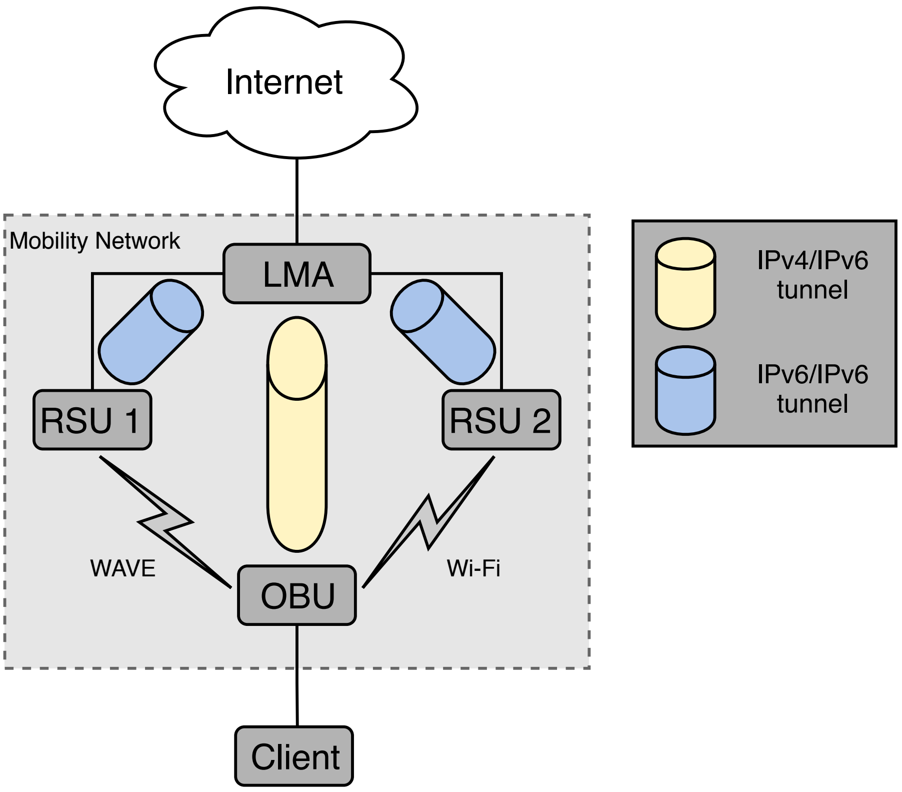

Figure 3.2: Simple representation of the N-PMIPv6 based VANET.

3.1.2 MANO System

The MANO system is the component in charge of the NFV MANO of the solution's NFVI, as well as the deployment of the Network Services (NSs). As discussed in subsection 2.4.1, it provides an orchestration service, through the NFV Orchestrator (NFVO) and a VNF Manager (VNFM), as well as a Virtualized Infrastructure Manager (VIM). This component can be seen as the base of operations; it stays outside the scope of the VANET network, in the cloud effectively.

Given the main practical implementations of the NFV MANO architectural framework presented in subsection 2.4.2, the project that was chosen to provide the MANO functionalities in the proposed solution for this dissertation is the Open Source MANO (OSM). This choice was based on a few specific reasons and those are: (1) this is a project hosted by ETSI, which is the entity that defined the NFV and MANO reference architectural frameworks [2], that provides a production-quality, extensive and working implementation of the MANO stack; (2) it is a project backed by a large group of network operators and other NFV related parties; (3) it has built-in support for different types of VIMs, which allows the deployment of more complex multi-site NSs; (4) it supports fully automatic instantiation of NSs alongside their VxFs, as well as, their automatic configurations via the use of the Juju Charms present in the VNF Configuration & Abstraction (VCA) component; (5) it is a fully open-source solution which provides extensive documentation and a straightforward installation process.

As for the VIM the choice came down to Openstack (OS), since the VIM options supported by OSM exactly the ones presented in Section 2.4.2 with the additional choice of Amazon Web Services. OpenVIM is a solution that is part of the OSM project; it is limited in terms of features and is typically used in conjunction with the OSM project as a all-in-one solution. Both VMware and AWS, since they are commercial solutions, they were not considered, so that leaves OS. Given the modular design of the OS software, when performing a minimal installation, only a handful of services are required. The solution proposed in this dissertation works on the minimal install of OS, given the fact that it provides all the requirements of a VIM; the only extra service that was installed for convenience was the Horizon service, which is responsible for providing a web based user interface to interact with the other OS services.

3.1.3 On-Boarded Hardware

The second main component that makes up the solution is the on-boarded hardware. Unlike other solutions (see section 2.5) which make use of already existing hardware present on vehicles to provide a platform capable of making up the NFVI, that is, capable of hosting the VxFs, this dissertation focuses on providing a solution which is not restricted to a specific type of hardware that will make up the NFVI, as long as it is possible to install the required software (in the case of this dissertation, Openstack (OS) and its services).

In the proposed solution, considering the nature of the VANET used in this dissertation, the node that sits closer to the end-user is the OBU. Therefore it would make sense to use the OBU as the platform on which the VxFs could be deployed. However, considering that OBUs were conceived with the intent to solely run the mobility-related software, they are limited in terms of hardware resources, thus it was better to separate the VANET's communication modules from the solution's computational modules. This means that it was decided to add an extra hardware element alongside the OBUs in the vehicle, one capable of supporting the execution of the VxFs.

Considering that the hardware is going to be on-boarded on a vehicle, we need to take into account the constraints that it presents. In terms of size, the solution cannot be based around having a typical computer sitting inside a vehicle. This implies that the hardware has to be relatively small in size, which in turn means that it is going to be more limited in terms of both processing and storage capabilities. As a result, the NSs and VxFs that can be instantiated will obviously be dependent on the hardware's resources that are available to them; the more resources the hardware provides, the better and more complex services are possible. Having said that, the developed VxFs should provide their functions, but at same time, it should be taken into account the limitations of the hardware that will host these VxFs.

Taking all that into account, for the proposed solution, we decided to use the Raspberry Pi (RPi) as the hardware platform to be on-boarded alongside the OBUs in the vehicles. Given the choice of using a RPi, there are some constraints; one of the most obvious is the limitation in terms of resources; another important constraint is imposed by the fact that the RPi is restricted in terms of its capabilities to support virtualization. The RPi does not support the typical hypervisor alternatives, but it does support container virtualization via the use of Linux Containers (LXC), as was presented in [29], where the authors showed that RPis were successfully able to host LXCs. This works very well for the solution, since Openstack (OS) offers support for LXC too, making it possible to use the RPis as the NFVI of the MANO system. The RPis will then be the compute nodes of the OS VIM.

3.1.4 Overlay Networks

Another important component that makes up the proposed solution concerns the virtual overlay networks that sit on top of the pre-existing VANET's mobility network. These are the networks that interconnect the components of the solution, the MANO system and the on-boarded hardware. One question that might arise is why the use of virtual networks: since the VANET already provides a network, why not use that as the way to connect the solution's components. When thinking about the solution, the use of the mobility network was considered but ultimately that choice was dropped in favor of the use of virtual networks to provide the overlay networks. This choice was based on a few specific reasons. First the software that makes up the MANO system (namely OSM and OS) has specific networking requirements. In the case of OSM, it requires that itself and all the compute nodes have a network interface present on the same IP sub-network. This network is required by OSM in order for it to configure the VxFs once they are running, as for OS it also requires that itself and its compute nodes have an interface on the same IP sub-network, for management purposes. Second, given the way the mobility network works on the VANET used in this dissertation, changes to it would need to be made to accommodate for those requirements, so by using virtual networks, no major changes need to be made in order to fulfill the networking requirements of the MANO system. Finally, the isolation of the network's traffic, by using overlay networks, the traffic of the MANO system's networks is isolated from the rest of the mobility network's traffic.

The two main virtual overlay networks are the VIM management network and the VxF management network (depicted in Figure 3.1). The former is the network responsible for allowing all the communications between the VIM and the compute nodes (the on-boarded hardware), i.e., it will enable the control of the computing, storage and networking resources of the compute nodes; the latter is the network responsible for allowing all the communications between the OSM's RO and the VxFs; it will enable the management and configuration of the VxFs once they are deployed. There is still another type of virtual networks that needs to be mentioned: these are the virtual internal networks required for data communication between the VxFs that make up a NS. In order for the solution to work, both the VIM and the VxF management networks have to be pre-created on the respective nodes of the solution, particularly on the MANO system and on the on-boarded hardware. As for the virtual internal networks which are responsible for the data communications, these are not pre-created; they are instead created by the MANO system whenever a NS requires such a network.

An example of this would be the deployment of a simple NS made up of two VxFs. In this case, the MANO system would task its VIM to instantiate the VxFs on their respective compute nodes. It would then task the VIM with setting up the VxF management network on each VxF to later allow the MANO system's OSM to configure the VxFs using the set of Juju Charms that were created for each VxF. The VIM would also be tasked with creating the virtual networks required for the data communications between the two VxFs. A more detailed explanation of actual use cases will be given in the next chapter.

3.2 Solution Implementation

As presented in subsection 3.1.2, the software that makes up the components of the MANO system are all based on open-source solutions. For OSM, the Release Four [30] was used; as for Openstack (OS), the choice came down to the use of the OS Ocata release. Given the constraints presented in subsection 3.1.3, the choice of using RPis as the on-boarded hardware, alongside the use of LXC as the way to deploy the VxFs was made. This was a good choice given the fact that OS also supports LXC, but this is where a slight problem appeared. At the time of the development of the solution the OS's virtualization library responsible for allowing the use of LXC on the RPis was not supported for all of the OS versions; the latest version which had support was Openstack (OS) Ocata, and so this was the version used. The next subsections will give more detailed explanations on how the proposed solution and its components were implemented and deployed over the top of the VANET, as well as showcase the problems faced and how they were solved.

3.2.1 Connectivity between MANO System and On-boarded Hardware

The first and most critical step of the solution's implementation is to establish full connectivity between the host machine that runs the MANO system and the on-boarded hardware (i.e. RPis).

Given the fact that the RPis stay inside the scope of the VANET, inside the vehicles alongside the OBUs, the communication between the MANO system and RPis will have to go through the VANET's mobility network. The configuration of the connectivity between the MANO system and the RPis was done in a series of steps that all come together to allow the full connectivity in the end. A representation of this process can be seen in Figure 3.3. Its steps will be presented next.

The first step was to make the the RPis apart of the VANET's mobility network. The choice for connecting the RPis to the VANET came down to using either Ethernet or WiFi. Since the proposed solution makes use of the RPi's wireless interface in order to be able to provide special VxFs (explained in more detail in the last subsection), the choices were reduced to either using the RPi's Ethernet port or using a wireless USB adapter. Since the OBUs used in the VANET have unused Ethernet ports, what made most sense was to use both the RPis' and the OBUs' Ethernet ports to connect them, thus bringing the RPi inside the scope of the mobility network. This was accomplished by creating an IPv4 sub-network between the RPis and the OBUs, where all the traffic that originates from the RPis is routed through the OBUs via the addition of a default Internet Protocol (IP) route to the OBU (which acts as the sub-network's gateway) and into the mobility network's scope.

The next step revolves around the way the mobility protocol used in the VANET works. As said before, the protocol communication works on the basis of tunnels. The protocol creates bi-directional tunnels from both the LMA to the RSUs as well as from the LMA to the OBUs. In order for the RPis to be reachable from the rest of the network (e.g. from the LMA), routes to the previously created IP sub-networks between the OBUs and the RPis needed to be added on the LMA. These were configured on the LMA by adding an IP route to each created sub-network via the respective IPv4/IPv6 tunnel device which connects the LMA and each OBU. With this configuration, the RPis were now able to reach the outside of the mobility network, which means that they could reach the Internet and most importantly for the proposed solution, they could reach the host running the MANO system, all while using the mobility network.

The final step to establish full connectivity between the components has to do with the configuration of the host running the MANO system. In the case of the host, not many changes were necessary; in fact, only the addition of IP routes was necessary. Given the fact that the LMA already had an interface outside the mobility network, for communications outside the scope of the mobility network (e.g. communication with the Internet), the connection between the host running the MANO system and the on-boarded RPis was enabled by simply adding IP routes to the IP sub-networks previously created via the LMA's outside interface.

As an example, like shown in Figure 3.3, Network 1 was setup when connecting RPi 1 to OBU A, where OBU A acts as Network 1's gateway. An IP route to Network 1 was then added on the LMA, this route is responsible for routing traffic to Network 1 via the tunnel device which connects the LMA and OBU A. Lastly an IP route was added on the MANO system to enable communication with RPi 1.

With all these configurations complete, the full connectivity between the host running the MANO system and the RPis was established, and the deployment of the overlay networks was now possible.

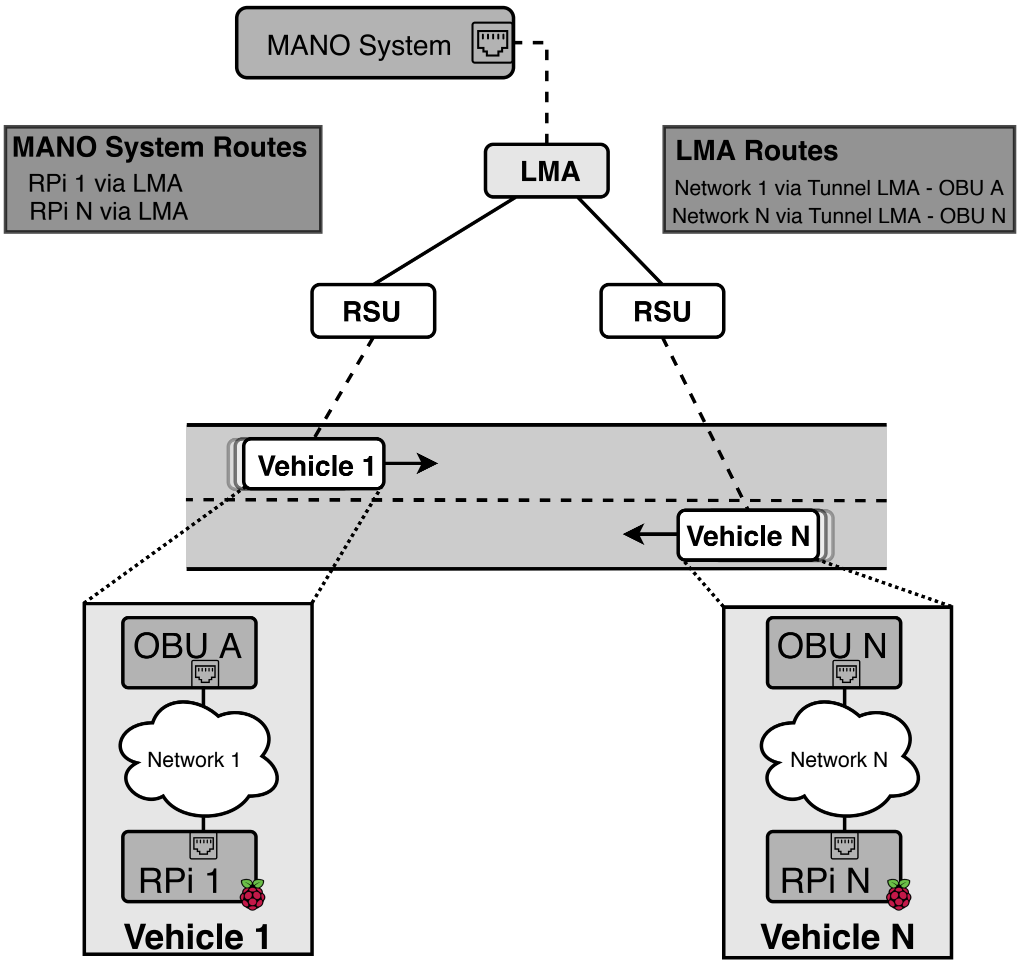

Figure 3.3: Representation of the configuration process to enable connectivity between the MANO System and the On-boarded Hardware.

3.2.2 Deployment of the Overlay Networks

Through the connectivity between the host and the RPis, it was possible to start deploying the support overlay virtual networks depicted in Figure 3.1. Taking into account the reasons presented in subsection 3.1.4 on why the use of virtual networks was the best choice for the solution, we decided to use Virtual Extensible LAN (VXLAN) as the technology responsible for supporting these virtual networks. VXLANs are a good choice as they provide a way to create a logical network for machines across different networks, allowing the creation of a layer 2 network on top of an already existing layer 3 network through the help of encapsulation, thus making it possible to create virtual overlay networks on top of the mobility network, and that way, surpass the constraints mentioned earlier. As mentioned in subsection 3.1.4, both the VIM and VxF management virtual networks need to be pre-configured on the host running the MANO system, as well as on the RPis, for the solution to work. The creation of these virtual overlay networks using VXLAN devices was done on each component of the proposed solution.

To explain how the process was done, first a brief explanation of how VXLANs work is necessary. VXLAN [31] is an extension technology of the known Virtual LAN (VLAN): it is a technology that works based on encapsulation. It encapsulates normal Layer 2 Ethernet frames within IP using the UDP protocol via a specific port assigned by the Internet Assigned Numbers Authority (IANA), port 4789. Much like VLAN, VXLAN has a VXLAN Network Identifier (VNI), which defines the VXLAN broadcast domain. The main devices in the VXLAN technology are called the VXLAN Tunnel Endpoint (VTEP): these refer to the devices that create or receive VXLAN traffic. These are divided into two main types: hardware devices (e.g. switches) which handle the VXLAN traffic in hardware, or software devices which handle it in software. In the case of the software devices, which are the ones used in this dissertation, when they receive traffic for a host of the overlay network, they encapsulate it and send it via the device which connects that host with the underlay network, to the VTEP where the overlay network recipient is located.

Given a brief overview of the VXLAN technology, the creation process of the virtual overlay networks using VXLANs can be explained. The process is the same for both the host and the RPis with only a few obvious changes. This process is explained below:

1. **Creation of the VXLAN devices** A new VXLAN device is created, which acts as a VXLAN tunnel endpoint: it is given a name, a VNI (to distinguish between different virtual networks), the interface name which is responsible for accessing the underlay network, and the UDP destination port set by IANA.
2. **Creation of the Linux Bridges** A new Linux Bridge is created: these are used to bridge the VXLANs that are created in order to enable multiple VxF to share a single VXLAN device (multiple VxFs can be running on the same RPi).
3. **Setting up the Forwarding Database** A new forwarding database entry is added for each one of the remote host's IP addresses of the underlay network. This makes it so the VXLAN device can know how to reach the remote VTEPs.
4. **Bridging the VXLAN device** The VXLAN device is bridged with the help of a newly created bridge, and an IP address for the overlay network can be assigned to this bridge.

A sample example of the configuration used in the proposed solution can be seen in the following example:

```
ip link add vxlanA type vxlan id 1 dev eth0 dstport 4789
brctl addbr brA
bridge fdb append to 00:00:00:00:00:00 dst <ip-remote-underlay> dev vxlanA
brctl addif brA vxlanA
ifconfig brA <ip-addr-overlay>
```

In the example, we observe that a VXLAN device named *vxlanA* with id *1* is created, with interface *eth0* as the interface that connects to the underlay network. Then a Linux bridge called *brA* is also created; next a forwarding database entry to the remote host's IP address on the underlay network is added. Lastly, the VXLAN device is bridged with the help of the newly created bridge. With all this done, the overlay network is created and finally an IP address can be configured on the bridge. This IP address is now an address of the overlay network and can be reached by other nodes on the overlay network.

The configuration done in both the host running MANO system and the RPis follows the process outlined above for the configuration of both the VIM and VxF management virtual overlay networks. In the case of the host that runs the MANO system, two VXLAN devices were created with different names and VNIs (for each management virtual network), where the interface name was the name of the interface which can access the RPis through the underlay network. In the case of the RPis, the same was done: two VXLAN devices were created, with the same names and VNIs as the ones used in the host running the MANO system, where the interface name, was the name of the interface which the RPis used to communicate with the mobility network, and thus, the host running the MANO system. Figure 3.4 shows a better look at how the network configuration setup at the RPis (the compute nodes) looks.

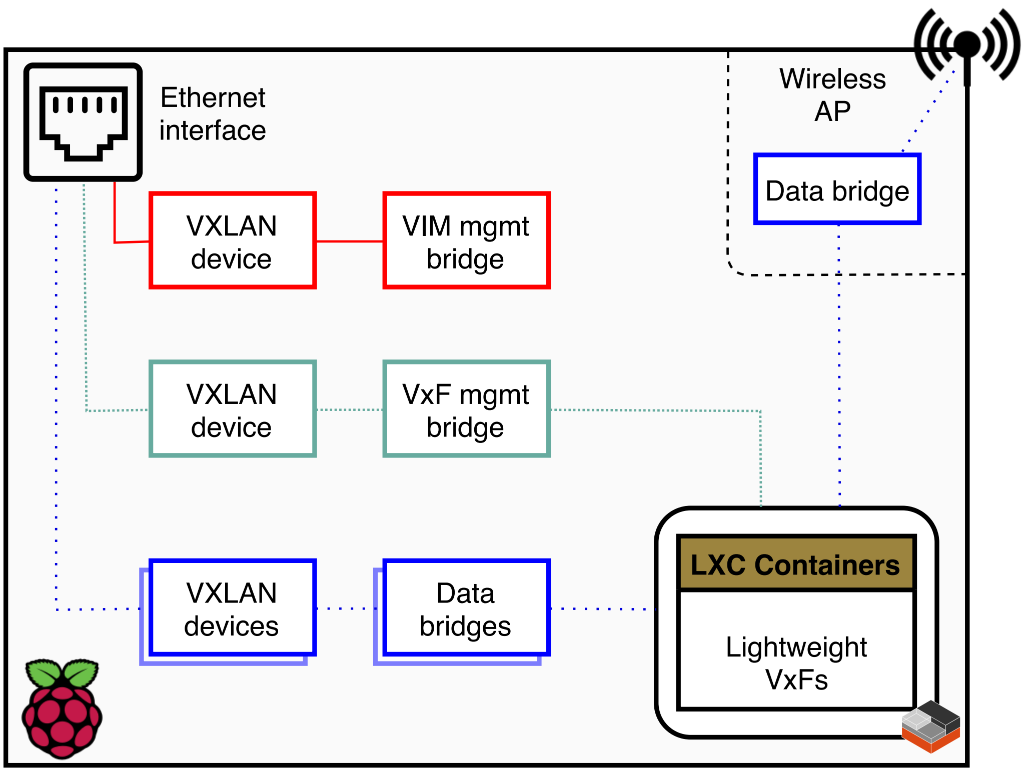

Figure 3.4: Virtual Networks setup on the RPis (compute nodes).

As for the internal data communication networks, the configuration performed on the MANO system and the RPis enables the dynamic creation of such networks between VxFs (which can be located either at the same or at different compute nodes), as requested by the OS VIM and according to instructions provided by the MANO system. More details on these configurations will be provided in the next subsection.

Problems with Maximum Transmission Unit (MTU)

It is well known that, by using an overlay encapsulation protocol such as VXLAN, overhead is going to be introduced. Figure 3.5 shows the format of a packet when using VXLAN. It is possible to observe the overhead the protocol introduces, which comes in the form of the underlay

IP header, underlay UDP header, the VXLAN header which contains the protocol's metadata and the overlay Ethernet header. All of these create overhead that consumes a portion of the underlay IP packet, thus reducing the MTU size available to the machines on the overlay network (in the case of the proposed solution, the VxFs instances). This means that the network interfaces of the machines running in the overlay network must use the underlying physical network's MTU minus the overhead introduced by the VXLAN protocol. As presented earlier, in the configuration that was done on the solution's components, the VXLAN devices used Ethernet (IEEE 802.3 Ethernet) interfaces to access the underlay network. These have a default MTU value of 1500 Bytes, which was the value present in the interfaces. Therefore, when created, the VXLAN devices took that value and subtracted the 50 Bytes of overhead introduced by the protocol, ending with an MTU value of 1450 Bytes: this was the value that was then available to the machines on the overlay network.

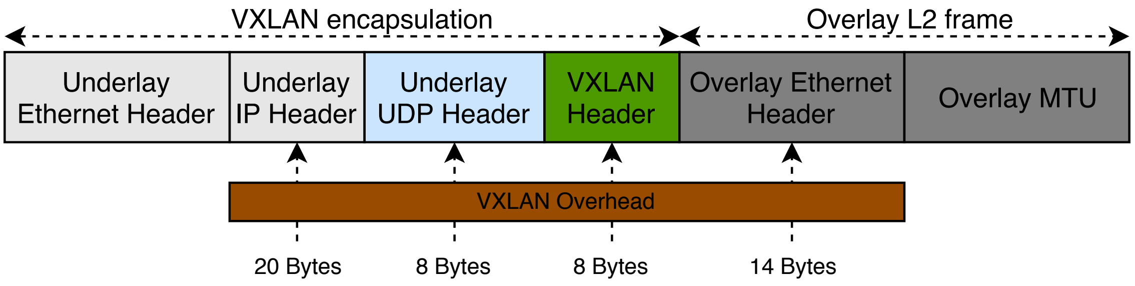

Figure 3.5: VXLAN protocol overhead in the context of overlay networks.

With the virtual overlay networks set up and the interfaces properly configured, some tests were conducted, but while performing those tests some issues started happening regarding the packet exchange between MANO system and the compute nodes, specifically, packets closer to the MTU limit in terms of size (given the MTU of 1450 Bytes) were being dropped at certain elements of the underlying network. After some investigation, the problem started to become apparent, given how the VANET's mobility protocol works. As mentioned in subsection 3.1.1, it uses two IPv6 tunnels to establish communication between the LMA and the OBUs, which means that the traffic to and from the RPis gets encapsulated twice, once when entering the IPv4/IPv6 tunnel between the OBU and the LMA, and again when entering the IPv6/IPv6 tunnel between the RSU and the LMA. This means that much like with the VXLAN protocol, the double IPv6 tunnel headers consume a portion of the outer IP packet, reducing the size available for the payload.

In the case of the mobility network, traffic that originated at either the LMA or the OBUs did not have any problems in terms of dropping packets, as the tunnels present at each of those components were correctly setting up their MTU values, which meant that they were correctly taking into account both the IPv6 tunnel headers when automatically setting their MTU values. But in the case of the overlay networks there were problems: traffic originated at either the MANO system or the compute nodes (in the overlay network) was taking into account an MTU value (the 1450 Bytes presented above) which was greater than the one supported by the mobility network, so when a packet was of a size greater than the MTU supported by the mobility network, it would obviously get dropped. A possible solution to this problem would be to make use of a technique like Path MTU Discovery (PMTUD) as a way to determine the maximum MTU value supported by a specific network path between two end-points. The use of this technique would then make it possible to determine the maximum MTU value supported on the network path between the MANO system and RPis. With a maximum supported MTU value, it would then be possible to correctly configure their interfaces' MTU with that value, that would also mean that, when the VXLANs devices were created, they would not take into account the default 1500 Bytes MTU value of the Ethernet interfaces, but instead the new maximum MTU value that was configured, which would make the machines in the overlay network have the correct MTU value and thus the problem would be solved.

Unfortunately, the use of PMTUD did not work how it was intended. On some of the hardware that makes up the VANET used in this dissertation, due to possibly hardware or software related problems on the equipment used by the vehicular network that were not discovered, the packets used by the PMTUD technique were sometimes being discarded, which resulted in the process not working and consequently the maximum MTU value supported not being determined automatically. A makeshift solution to this problem was to discover the maximum MTU value supported by the mobility network manually without using the PMTUD technique and use that value to configure the MTU of the bridge interfaces that bridge the VXLANs devices on both the host running the MANO system and on the RPis by using that value minus the VXLAN headers size. With this solution, the issue was resolved and the overlay networks were fully operational, so the installation and configuration of the MANO system's components was now possible.

3.2.3 MANO System's installation and configuration

After having the overlay networks in place and fully operational, it was possible to start installing and configuring the components of the MANO system, that is, the Open Source MANO (OSM) and Openstack (OS). The first component that was installed was OS. As mentioned in subsection 3.1.2 and at the beginning of this section, OS was installed based on the minimal installation guide of the Ocata version, with the addition of the Horizon service for providing the web interface to interact with all the other OS's services. The installation process was straight forward given the guide provided by OS, but even then the main steps and configuration differences for the proposed solution are discussed next.

In the process of installing OS, the first step was to configure IP addresses on the bridge interfaces that bridge the VXLAN devices which are responsible for the VIM management overlay network on both the MANO system and on the RPis, to make it possible for the OS controller to effectively manage the compute nodes (the RPis). Once the connectivity between all the nodes using the new addresses on the overlay network was operational, several software packages required by OS, such as databases, synchronization software to use the Network Time Protocol (NTP) to synchronize the OS services among different nodes, OS packages which contain its services and more were installed. Then, it was possible to start installing the actual OS services. The services installed were the Identity service, Image service, Compute service, Networking service, and finally, the Horizon service. These are responsible for managing authentication, virtual machine images, compute nodes, networking and the dashboard respectively. For most of these services, the installation was straight forward given the OS install guide only requiring slight changes. It is worth noting that, in the case of the RPis, given that they are the compute nodes of the OS controller, only the Compute and Networking services are required and thus only those were installed and configured.

The networking service configuration was the most interesting, since OS provides two different options regarding networking configurations. Option 1 only supports attaching instances to existing networks, in the case of this solution one of the pre-created overlay networks. Option 2 builds upon option 1 by also supporting the creation and attachment of instances to private networks. As mentioned in subsection 3.1.4, the proposed solution relies on virtual internal networks to provide data communications between the VxFs that make up a NS. This means that option 2 was the adequate choice, given the fact that it provides a way to create these networks which are required by the proposed solution. In OS the private networks are typically created using overlay networks and these can be configured using different protocols supported by OS, one of those is the VXLAN protocol which is also used in the proposed solution. To keep everything the same throughout the solution, this was the protocol used to configure the overlay networks which make up the OS's private networks. Considering that the configuration was only required on the MANO system's OS controller, the networking configuration files were correctly edited with the options required to support the creation of the private networks using overlay protocols, in this case the VXLAN protocol.

Before proceeding to the installation of the MANO system's OSM, some of its requirements needed to be addressed first. When using OS as the OSM's VIM, OSM imposes some requirements and these are: (1) guarantee that the OS's API endpoints are reachable from OSM, this is how OSM interfaces with the OS VIM; (2) have a management network reachable from OSM, which is required by the OSM's VCA for configuring the VxFs once they are running using the Juju Charms; (3) have a valid user with full permissions that OSM can use to interface with the OS API.

The first requirement is taken care of, given the fact that the OS VIM and OSM are hosted on the same machine, the MANO system, so communication between them is guaranteed. The third requirement was also simple to take care of, since a new OS account with admin permissions was created and ready to be used by OSM. As for the second requirement, some additional OS configurations were required.

This is where the previously created VxF management network comes into play, since it is the network that OSM will use to configure the VxFs. In order for the VxF instances (launched by OS) to have an interface attached to this network, some additional OS configurations need to be made. The first thing that was done was the creation of a OS virtual network in the OS controller, which is a network used to represent the VxF management network. Then, that virtual network was configured to provide IP addresses of a specific sub-network to any VxF instance that would be attached to this network. After the creation and configuration of the virtual network in the OS controller, it was possible to configure the controller and the RPis to use the network. The OS configuration files of both the controller and the RPi were edited to create a mapping between the newly created OS virtual network and their respective network interfaces which are responsible for the VxF management network. The next step was to configure an IP address on the bridge interface that bridges the VXLAN device, which is responsible for the VxF management overlay network on the MANO system. This IP address was an address of the same sub-network configured in the newly created OS virtual network, thus allowing communication between OSM and any VxFs that were attached to that network. With all these configurations complete, the OSM requirements were met and its installation could be started.

The installation of OSM was straight forward; the OSM website provided a simple installation script that installed all of the OSM's components. After the installation, the only thing left to do was to add the OS controller as a VIM and, considering that the OS controller was previously correctly configured to run as an OSM VIM, the process was simple.

As mentioned in subsection 3.2.1, the RPis's wireless interfaces are used to provide a special type of VxF, the Access Point (AP) VNF, which is used to allow the end-users to communicate with the remaining VxFs of a NS when it is required. In order to make this possible, some more changes on the OS controller and the RPis had to be made. The first thing was to setup the RPis' wireless interface as an AP using the *hostapd* software. The next step was similar to the process presented above regarding the configuration of the VxF management network in OS. So, on the RPis, the OS networking configuration file was edited to add an additional mapping between a new OS virtual network (which represents the network between the RPis's AP and the users) and the wireless network interface which provides the actual AP. The remaining configuration was done on the OS controller on the MANO system where the new virtual network was created, but unlike in the creation of the OS virtual network used to represent the VxF management network, which was configured to give out IP addresses, in the OS virtual network used in this case no configuration to give out IP addresses was done, as these are to be provided by the actual AP VNF.

3.3 Summary

This chapter presented the proposed solution, starting with its design and finishing with how that design was implemented. The individual components of the proposed solution were presented alongside the problems that were faced and how they were solved. The implementation section showed how the different components were configured in a way to bring together the whole solution. This chapter showed how the proposed solution was designed to be able to be deployed over an existing VANET without the need for major changes to the VANET itself or its mobility protocol, by making use of different components like the overlay networks. It also presented how the solution can be used with any type of on-boarded hardware, as long as it is compliant with the solution's requirements in terms of the software that it has to be able to support. It also demonstrated how the solution is capable of instantiating NSs made up of various VxFs, which can be placed at different points of the edge of the VANET and still have full connectivity between them.

CHAPTER 4

Evaluation

This chapter presents the evaluation of the proposed solution. It focuses on examining how the proposed solution behaves and on exploring its limitations considering the fact that it is deployed over a VANET. It also presents two use cases in the form of two services that were developed with the intent of showcasing what the solution brings to a VANET scenario. These use cases were then evaluated in order to showcase the improvements brought by the proposed solution.

4.1 Testbed

As mentioned before, the proposed solution was developed to be deployed over a VANET. In this dissertation the system components used to perform the evaluation and tests are the components showcased in Figure 3.1. In the case of the VANET's components, the LMA was running on a dedicated computer, whereas the RSUs and OBUs are implemented in NetRiders (depicted in Figure 4.1). These are Single-board Computers (SBCs) with IEEE 802.11p, 802.11b/g/n and cellular interfaces. As for the proposed solution's components, the MANO System was running on a dedicated computer; the RPis used were RPi 3 Model B. The overall main specifications and characteristics of the equipment used in the experiments are presented in Table 4.1.

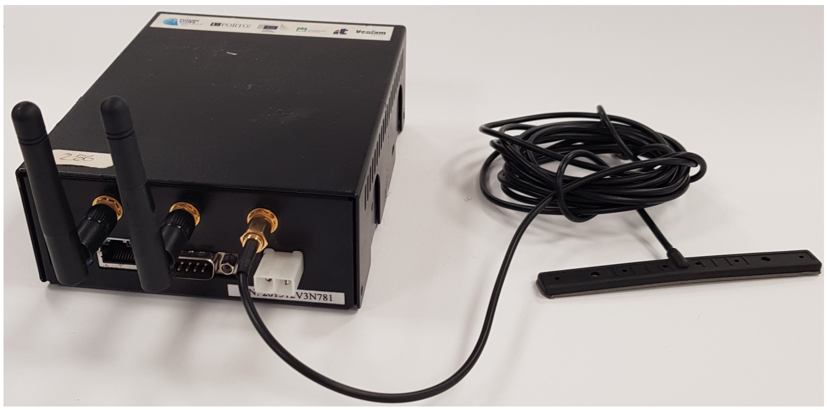

Figure 4.1: NetRider v3.

Equipment	CPU [MHz]	Memory [MB]	Linux Kernel	Operating System
LMA	3600 (2 cores)	4096	4.14.3-mobility-networks	Ubuntu 16.04.3 LTS
NetRider V3	680	64	3.7.4	VeniamOS 19.2
RPi	1200 (4 cores)	1024	4.4.38-v7+	16.04.6 LTS
MANO System	3600 (4 cores)	8192	4.4.0-154-generic	16.04.6 LTS

Table 4.1: Main specifications and characteristics of the equipment.

4.2 NFV solution behaviour

The solution is deployed over a VANET, which as seen before, is a type of network whose topology is constantly changing, with several disconnections and handovers. Therefore, to evaluate the proposed solution, a number of test scenarios were created to evaluate how several critical parts of the solution behaved when handover/disconnect situations of the OBUs (where the RPis are connected) from the rest of the VANET, occur. These test scenarios took a more qualitative approach, given that these are tests which will not measure network metrics (e.g. throughput). Instead they will check if specific components of the solution, which are required for it to operate correctly, still work when typical events of a VANET, such as handovers or loss of connection, occur.

4.2.1 Test scenario 1

The first test scenario consisted of observing the impact of OBUs's handovers/loss of connection to the VANET infrastructure in the communication between the Openstack (OS) controller and the RPis (the computes nodes of the OS controller). This is done by performing handovers/disconnections of different time intervals. It is shown that, for any time interval where a disconnection happens, the communication between the OS controller and the RPis is not possible until the connection is re-established. This is something that makes sense as the traffic between the RPis and the OS controller uses the mobility network, even though they communicate via the VIM management network (a overlay network). It is worth noting that OS uses the Advanced Message Queuing Protocol (AMQP) (in the form o RabbitMQ) to allow the OS controller and the compute nodes to communicate in a loose manner. What happens is that, once the compute nodes connect themselves to the OS controller, they will exchange heartbeats periodically according to the configuration. This means that they will not get disconnected immediately from the queue when a handover or disconnection from the infrastructure happens; that will only happen if the duration of the disconnection is longer than the timeout value configured for the heartbeats (in the case of the proposed solution, one minute, as can be seen in Table 4.2). Overall, this means that, if the disconnection duration is lower than the configuration value, OS controller still considers the compute node up and able to host new instances. This was something important to know for the test scenario presented in subsection 4.2.3.

OBU loss of connection	5 (s)	15 (s)	30 (s)	60 (s)	300 (s)
Compute node disconnection from queue	no	no	no	yes	yes

Table 4.2: Compute node queue connection status for different values of OBU loss of connection to the infrastructure.

4.2.2 Test scenario 2

The second test scenario focused on observing what would happen to the VxFs once they were instantiated, in terms of communication, that is, what kind of communication was possible to and from the VxFs when the OBUs, and thus the compute nodes had no connection to the infrastructure. Something that was also tested was: if a VxF was exchanging data with a user (e.g. providing a real-time service), would the communication be resumed without any problems after a handover or loss of connection to the infrastructure. The results indicate that, for any time interval where an OBU, and thus a compute node gets disconnected from the infrastructure, the VxFs which are hosted on that compute node cannot establish communication with anything outside that compute node. There is one exception to this, which comes from a specific type of VxF, the Access Point (AP) VNF; in that case, communication between the VNF and the end-users which are connected to it is possible, even when a handover or loss of connection happens, given the fact that the communication between the VNF and the end-users does not use the mobility network. An overview of these findings can be seen in Table 4.3.

The next point that was tested was to find out what kind of communication is possible between the multiple VxFs which make up a NS, in the presence of handover/loss of connection to the infrastructure. The results were different depending on where the VxFs were instantiated, that is, if they were instantiated on the same compute node or on different compute nodes. The results that consider all the VxFs present on the same compute node indicate that, for any time interval where a handover or loss of disconnection happens, the VxFs can still communicate with each other without any problems; as for when the VxFs were instantiated on different compute nodes, the results indicate that for any time interval where one of those compute node gets disconnected, communication between the VxFs, which are hosted on one of those compute nodes and the remaining VxFs, is not possible. An overview of these findings can be seen in Table 4.4.

As for the final point, the recovery of the communications or sessions after a handover or loss connection was tested, and the results indicate that as long as the communication or sessions do not timeout (based on each specific application) until the connection to the infrastructure is re-established, they are resumed without any problems. This result is applied to all the cases mentioned above, meaning that the place where the VxFs are instantiated does not matter.

	Compute node loss of connection to infrastructure	
	Connection dependent	Connection independent
VxF communication	No communication	Seamless communication

Table 4.3: Possible communication by a VxF to the outside of the compute node in the event of loss of connection to the infrastructure.

Compute node loss of connection to infrastructure		
Multiple VxFs	**Same compute node**	**Different compute nodes**
hosted on	Seamless communication between VxFs	No communication to remote VxFs

Table 4.4: Possible communications between VxFs that make up a NS in the event of compute node's loss of connection to the infrastructure.

4.2.3 Test scenario 3

The third test scenario focused on testing how the proposed solution would react when a handover/loss of connection to the infrastructure happened while a NS and its VxFs were being instantiated or terminated. The test scenario was designed to figure out if the loss of connection during the instantiation or termination of a NS would cause inconsistencies in any component of the proposed solution, and thus require manual intervention to recover. This means that it is important to know how both processes works.

This instantiation process is initiated by OSM when a NS is launched and is divided into two main steps: first the VxFs that make up the NS that was launched are instantiated at their respective compute nodes by OS; then, after the instantiation process is complete and the VxFs are up and running, the OSM's VNF Configuration & Abstraction (VCA) module launches Juju charms which are responsible for automatically configuring the VxFs. This is accomplished by executing predefined commands on the VxFs remotely using the SSH protocol. This means that there are two possible points of failure in the instantiation process. The termination process is also initiated by OSM: when the termination process of a NS is initiated, OSM orders the OS controller to remove the VxFs by informing the compute nodes which host them that they are no longer needed. Then, OSM deletes the Juju charms used to configure the VxFs: once that is complete, the service is fully terminated.

The results gathered for the instantiation process show that, during the first step of the process if a handover/loss of connection happens, depending on its duration, the OS controller will either recover and instantiate the VxFs or it will timeout and report an error. In the case of a scenario like the one presented in subsection 4.2.1 where the disconnection duration is long enough for the OS controller to consider a compute node as down, when the OS controller tries to instantiate a VxF on a compute node which is down, the instantiation will not be possible and an error is reported. When any error occurs in the first step, OSM halts the whole instantiation process of the NS and presents an error message. It is worth noting that, in the event that some of the VxFs that make up the NS were instantiated but any other failed, when the NS is terminated by OSM, the OS controller terminates any instances that might have been correctly instantiated. This means that, when an error occurs in this step, the solution can recover without any inconsistencies by simply terminating the NS that was launched in OSM.

As for the second step of the instantiation process where the OSM configures the VxFs, the results show that, if a handover/loss of connection happens during the time the Juju charms try to connect to the VxFs in order to execute the predefined configuration commands, an error occurs and the process stops. The same applies if the disconnection happens while the configuration process is in course. Like in the previous step, when the configuration reports an error, the solution can recover without any inconsistencies, by simply terminating the NS in OSM. Possibly due to a bug in OSM, sometimes the termination of the NS does not properly cleanup the Juju charms, which means that manual intervention is required, but other than that the solution can recover without any problems.

The results gathered for the termination process show that, if a handover/loss of connection happens during the process, the OS controller and consequently OSM have to wait until the connection is resumed between the controller and the compute nodes to complete the termination. This is because the communication is required for the controller to be able to inform the compute nodes that the VxFs are no longer needed and have to be removed. In the case of a scenario like the one presented in subsection 4.2.1 where the disconnection duration is long enough for the OS controller to consider a compute node as down, the controller reports an error and OSM skips to removing the Juju charms. When the connection is eventually resumed, the compute nodes and the OS controller synchronize themselves and the VxFs which are not supposed to be running are then terminated. A general overview of these findings can be seen in Table 4.5.

Loss of connection to the infrastructure		
VxFs Instantiation	**Lower than queue timeout**	**Greater than queue timeout**
	Seamless recovery	Error reported
VxFs Configuration	**At the start of configuration**	**During configuration**
	Error reported	Error reported
VxFs Termination	**Lower than queue timeout**	**Greater than queue timeout**
	Seamless recovery	Skipped until recovery

Table 4.5: Overview of the possible outcomes when loss of connection happens during several critical steps of the solution.

4.2.4 Discussion

The fact that the network topology of the VANETs is dynamic means that it can change constantly. This means that the proposed solution could have experienced problems which could have jeopardized the whole solution, given that the technologies present in the components of the MANO system are generally used in situations where the networks which interconnect the components are very stable and normally do not experience changes in terms of their topology. Only recently there has been more research and experimentation with these types of technologies on more constrained and dynamic environments, such as in the case of the solution developed in this dissertation, where these technologies were used in a VANETs environment.

The results obtained when testing how the proposed NFV solution behaved, given that it was deployed over a VANET, demonstrate that it is capable of providing one of its main goals: the deployment of services at the edge of the VANET, without many inconveniences. The test scenarios presented in the previous sections focused on determining the limitations that the VANET imposes on the proposed solution. The results show that the communication to and from the VxFs is only limited by the connection to the VANET, that is, whenever the OBUs perform a handover or isolate themselves from the VANET, any VxFs which are hosted on the compute nodes which are connected to one of these OBUs, loses the ability to communicate with other VxFs deployed at other compute nodes until the OBUs regain connection to the VANET.

When it came to the instantiation and termination processes of the NSs, this was the most critical point of the solution. The results that were obtained when testing both processes indicate that the proposed solution is capable of handling the handovers/loss of connection that is common in a VANET environment while performing the processes, without creating any inconsistencies which could cause problems. Given the results, it is possible to conclude that, during any of the processes, the compute nodes which are used to deploy the VxFs need to be connected to the VANET without any interruptions until the process is complete. This is something that would require the NSs to be deployed or terminated when the target vehicles were in a position where, for the duration of the instantiation or termination process, they would not get disconnected from the infrastructure. In a real life scenario, these processes could be performed at strategic times and locations, such as when the vehicles are parked.

4.3 Use Cases

With the aim of presenting the possibilities given by using the proposed solution, two use cases were developed. These uses cases focus on showcasing the range of different services which can be developed for use in a vehicular scenario. These use cases aim to showcase the advantages of deploying the services at the edge of the VANET, as opposed to having to access the services on the cloud.

The services showcased by both of the use cases were created using two different Network Services (NSs). The VxFs that were used in these NSs were developed using open-source software solutions taking into account the limitations of the on-boarded hardware and the fact that they run on Linux Containerss (LXCs). As for the automatic configuration of the NS's VxFs once they are deployed, its was done by creating the Juju charms which are used by OSM. For each VxF, a custom Juju charm was created based on the template present in the OSM wiki [32]. Using this template, the charms use Ansible playbooks to execute the configurations on the VxFs once they are deployed. Once the VxFs and the respective charms were created, the NS Descriptors (NSDs) and the VNF Descriptors (VNFDs) that are required to on-board the NSs in OSM were also created.

4.3.1 Use Case 1: Safety service in a single compute node

The first use case consists of a NS that aims to show how a safety service could be created and provided using the proposed solution. For this use case, the NS shown in Figure 4.2 was developed. This NS provides a simple safety service that can be deployed on a single compute node, where a specific VxF processes a video feed in order to detect distinct objects. The results obtained can then can be accessed or viewed by the end-users. This NS is made up of two different VxFs, a *Object Detector* VxF and a Access Point (AP) VNF. The AP VNF provides a way to support the data communications between the *Object Detector* VxF and the video feed, and between the VxF and the users. The AP VNF contains a Dynamic Host Configuration Protocol (DHCP) server which is responsible for providing the necessary network configuration (such as IP addresses) to the equipment that connects to the network it provides. As for the *Object Detector* VxF, it provides object detection using software based on the open-source library TensorFlow [1], the results obtained by the software are then available through a simple HTTP server which can be accessed by the end-user equipment.

As for the automatic configuration of this NS, two charms were created for each of the two VxFs. In the case of the AP VNF, its charm is responsible for configuring static IP addresses, starting the DHCP server, configuring IP network routes and enabling IP forwarding. In the case of the *Object Detector* VxF, the charm is also responsible for configuring static IP addresses, IP network routes and for starting the object detection software. After the creation of the VxFs and their respective charms, the VNFDs and the NSD were created in order to allow the on-boarding of both the VxFs and the NS in OSM. Once this process was complete, the NS could now be instantiated via the MANO system's OSM interface.

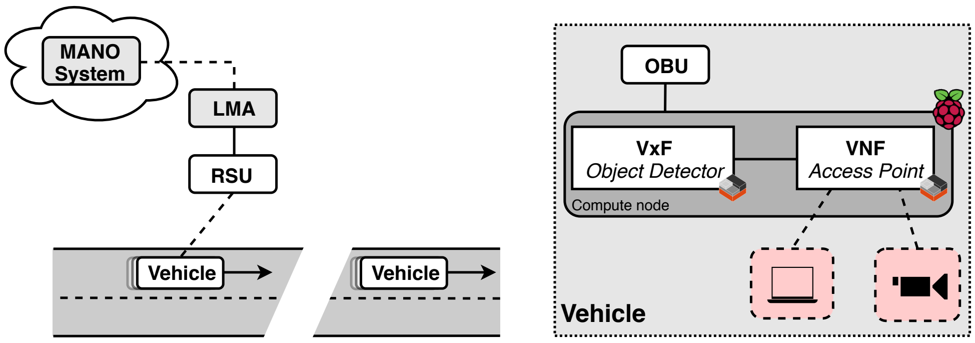

Figure 4.2: Overview of the safety service used in Use Case 1.

Evaluation of the use case

In order to showcase and evaluate the advantages that the proposed solution brings, the NS developed for this use case was deployed over a VANET like the one depicted in Figure 4.2, where both the OBU and the RPi were grouped as one as if they were inside a vehicle (like it is depicted in the figure). Once the NS was deployed, it was possible to test and showcase the two main advantages that the solution brings. The first one is the ability to deploy services, like the one presented in this use case, where even when the vehicle loses connection to the VANET's infrastructure, the service still works without any issues. The next advantage that was showcased and tested was the lower latency that the solution can provide by allowing the deployment of the services as close as possible to the end-users, as opposed to requiring the users to access the services on the cloud. To showcase and test the first advantage presented, the OBU was disconnected from the infrastructure after the deployment and configuration of the NS, given that these steps require

[1]`https://www.tensorflow.org/`

stable communication with the MANO system's components, as shown in subsection 4.2.3. As for the second advantage, the NS was compared in terms of communication delay with a different version of the same NS where the *Object Detector* VxF was deployed outside the scope of the VANET, as a way to mimic its deployment on the cloud.

When talking about the first advantage that the solution brings, the results show that, for this use case, the vehicle's connection to the infrastructure (via a RSU as depicted in Figure 4.2) is only needed to deploy or terminate the service considering that communication between the MANO system and the compute node (the RPi) is required for those processes. When the vehicle lost connection to the RSU, and thus to the infrastructure, it was clear that the communication between the MANO system and the compute node was not possible, as shown previously in subsection 4.2.1. In terms of the service that was deployed, there were no problems in terms of communication: this made sense given the fact that, once the service is fully deployed and configured, the communication between the VxFs that make up the service and the end-users is all local to the vehicle, meaning that the vehicle does not need to be connected to the infrastructure in order for the service to be operational and usable.

As for the second advantage, the results, which can be seen in Figure 4.3, show and support the advantage presented earlier, as it is possible to see lower latency values for the communication between both the VxF and the video feed, as well as between the user and the VxF for the case where the NS was deployed at the edge. The differences in terms of the delay are not huge, but nevertheless they are significant. It is also worth noting that, in the tests that were performed, even though the VxF was outside the scope of the VANET, it was still close to the LMA. In a more typical scenario, the VxFs could be hosted on a machine behind several networks and, in that case, the network delay would be more noticeable.

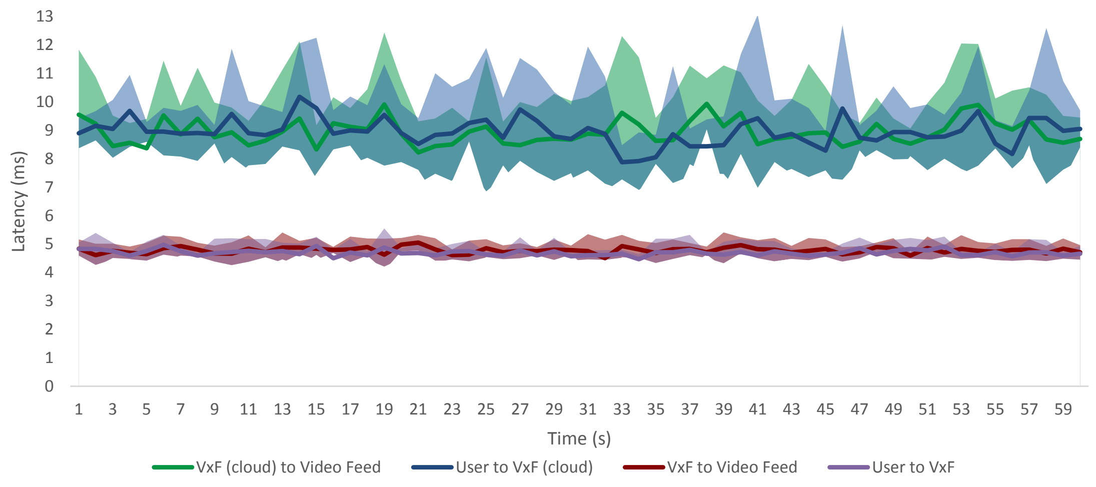

Figure 4.3: Use case 1: Latency results for local vs cloud VxF.

4.3.2 Use Case 2: Safety service in multiple compute nodes

The second use case consists of a NS, like the first one, that aims to show a safety service, but this time a service that involves more than one compute node. For that, the NS shown in

Figure 4.4 was developed. This NS provides a safety service where a video feed from one vehicle is transcoded and the output can be viewed by others vehicles. This can be useful in situations where smaller vehicles are trying to overtake a bigger vehicle like a truck: the video feed from the truck is transcoded and viewed by the overtaking vehicles to check if the road is clear and the overtake maneuver can be done safely. This NS is made up of three VxFs: two AP VNFs and a *Transcoding* VxF. The AP VNF, like in the previous use case, provides a way to support the data communications between the end-user equipment and the *Transcoding* VxF. The AP VNF was developed the same way as in the previous use case; as for the *Transcoding* VxF, it provides video transcoding using software based on the open-source project FFmpeg and also hosts the transcoded video output feed, so it can be accessed and viewed by other end-users.

As for the automatic configuration of this NS, much like for the previous NS, charms were created for each of the VxFs that make up the NS. For the AP VNFs, the charms are responsible for configuring static IP addresses, starting the DHCP server, configuring IP network routes and enabling IP forwarding. The *Transcoding* VxF charm is also responsible for configuring static IP addresses, IP network routes and most importantly, starting the transcoding and video output feed hosting software. As in the previous use case, once the VxFs and their respective charms were created, the VNFDs and the NSD were created to enable the on-boarding of the NS and its VxFs into OSM. It is worth noting that, in this case, where the NS has its VxFs deployed over more than one vehicle, specific configurations are required to instruct OSM where to deploy each VxF. This was done by defining availability zones (i.e. sets of resources) in the OS controller and by specifying the placement zone in which each VxFs is to be deployed on their VNFDs. After the on-boarding process, the NS was now ready to be instantiated via the OSM interface.

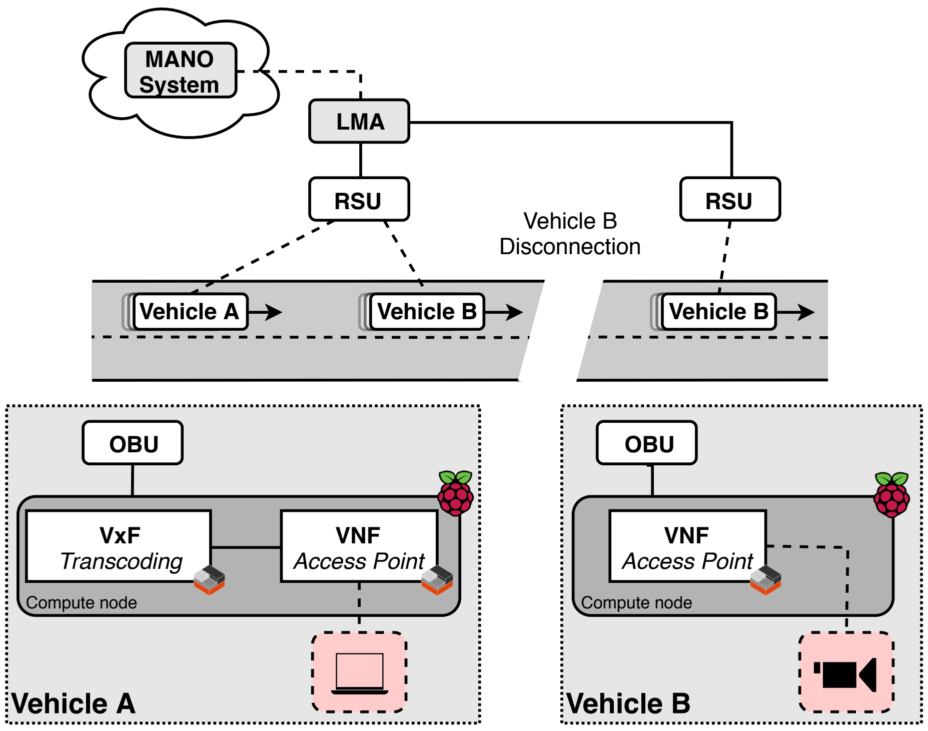

Figure 4.4: Overview of the safety service used in Use Case 2 (multiple compute nodes).

Evaluation of the use case

Like for the previous use case, as a way to showcase and evaluate the possible advantages that the proposed solution brings, the NS developed for this use case was deployed over a VANET like the one shown in Figure 4.4 where two OBUs and two RPis were grouped as if they were inside two different vehicles (vehicle A and B as depicted in the figure). Once the NS was deployed, it was possible to test and showcase how a multi-vehicle NS works in the proposed solution. Like in the previous use case, the first test consisted of performing disconnections of vehicle B's OBU from the infrastructure (as depicted in Figure 4.4) for a specific period of time. This use case's NS was also tested in terms of communication delay by comparing it with a different version of the same NS where the *Transcoding* VxF was deployed outside the scope of the VANET, as a way to mimic its deployment on the cloud.

Unlike in the previous use case, the results show that, for this NS to function properly, connection with the VANET's infrastructure is required at all times. For the process of instantiation and termination, both vehicles need to be connected to the infrastructure, considering that communication between the MANO system and the compute nodes (the RPis) is required for these processes to complete. Furthermore, once deployed, the service requires connection to the infrastructure for both of the vehicles, given that the service's communication is done using the underlay network (i.e. the mobility network). When one or both of the vehicles lost connection to the RSU, and thus to the infrastructure, it was clear that the communications between the vehicles was not possible, as was shown previously in subsection 4.2.2, meaning that the service was unusable until the connection to the infrastructure was re-established for both of the vehicles. This can be seen in Figure 4.5, where the vehicle that hosted the video feed got disconnected from the RSU for approximately 20 seconds, and while it was disconnected, the *Transcoding* VxF could not access the video feed, thus making the service unusable for that duration.

When talking about the advantage in terms of delay, the results, which are presented in Figure 4.6, show that, for a NS like the one presented in this use case, which involves more than one vehicle, the difference when it comes to the communication delay is not straightforward. It is clear to see that, for the NS presented in this use case, the delay value was significantly lower between the user and the VxF. This has a simple explanation given the fact that the user and the VxF were located on the same compute node, which meant that the communication between them was local. As for the latency values between the VxF and the video feed, they were significantly greater when comparing with values for the communication between the user and the VxF. This makes sense given the fact that, between the VxF and the video feed, the traffic has to cross the whole mobility network from vehicle A to vehicle B and back, as opposed to between the user and the VxF where the traffic communication is local. If the video communication is made direct, delays will be much lower.

When talking about the latency values for the case where the VxF was deployed outside the VANET's scope, these were very similar for both the communication between the user and the VxF and between the VxF and the video feed. When comparing these values against the use case's NS where the VxFs were deployed on the vehicles, the values were obviously greater for the case where the user communicated locally with the VxF, but still somewhat lower when compared with the values for the communication between the VxF and the video feed. Much like in the previous use case, the differences in terms of delay are not huge, yet they are significant, but it is worth mentioning that for these tests even though the VxF was outside the scope of the VANET, the VxF was still very close to the LMA, meaning that in a more real life scenario the VxF would be hosted behind several more networks which would increase the delay.

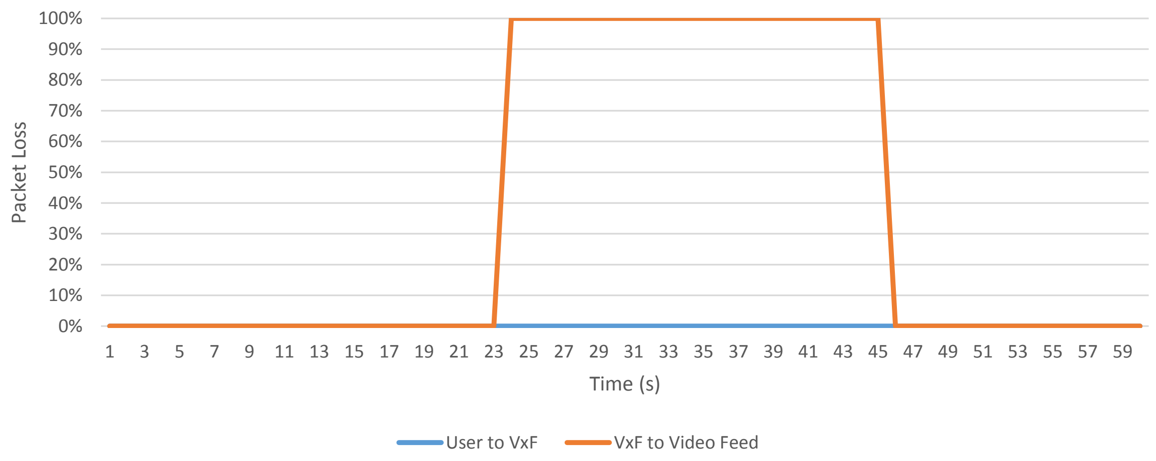

Figure 4.5: Use case 2: showcase of the service's communication given the disconnection of vehicle B from the infrastructure.

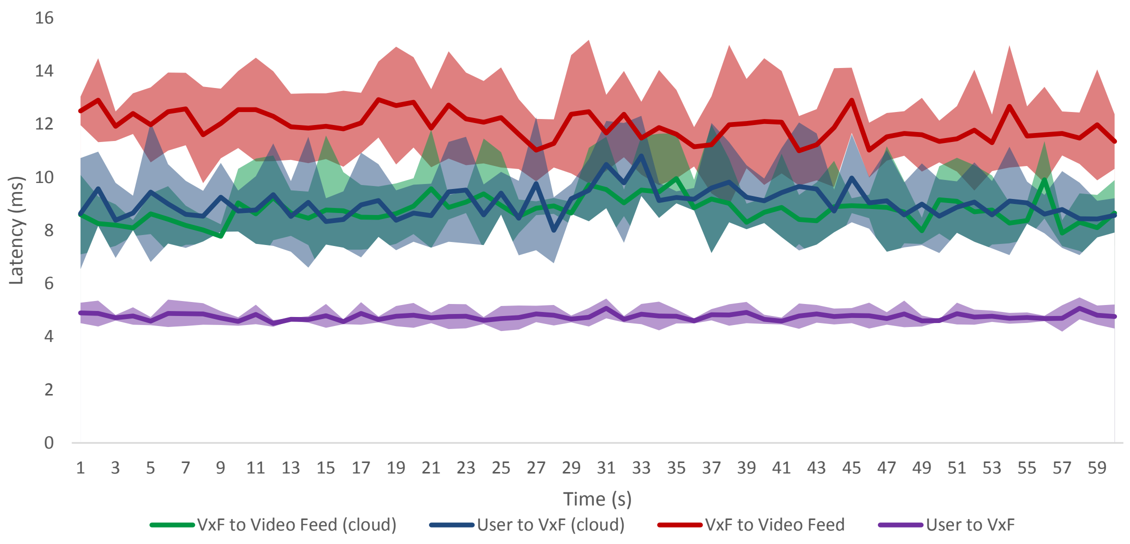

Figure 4.6: Use case 2: Latency results for local vs cloud VxF.

4.4 Discussion

Having developed and tested two different use cases, it is clear to see the type of advantages that the proposed solution can bring to a VANET scenario. By using the proposed solution in a VANET scenario, different types of services can be provided in a flexible and dynamic fashion.

The solution allows the deployment of services which can circumvent the loss of connection, as well as, have lower latency values, as opposed to the typical solution where the different services are deployed on the cloud, meaning that loss of connection to the VANET makes accessing any of those services impossible.

The NSs that are designed to provide local services like the one presented in the first use case, can be quickly instantiated and terminated while the vehicle has connection to the VANET; and after that if the vehicle loses connection to the VANET, the service is unaffected and works like normal. The proposed solution also allows NSs which can encompass multiple vehicles, like the one presented in the second use case, but in cases like that, in order for the services to work correctly, there has to be communication between the vehicles, meaning that the vehicles need to be connected to the VANET.

As the results showed, the proposed solution also offers lower latency values which can be crucial for time sensitive services, but this does not come without drawbacks. As mentioned before, considering that the hardware that supports the VxFs is on-boarded on the vehicles, it means that it is limited in terms of performance. When the VxFs are deployed on the cloud, they have access to better performing and more powerful hardware which means that the services provided can be more complex and faster in terms of processing. However, services like these require that the vehicles have access to the cloud and, by using the services on the cloud, the latency values of the communication increase. In essence, the proposed solution trades better hardware performance for the ability to allow the flexible deployment of different types of services with lower latency values.

CHAPTER 5

Conclusions and Future Work

This chapter presents the final considerations and conclusions of this dissertation. It also presents the aspects that can be improved in future work.

5.1 Conclusions

The main goal of this dissertation was to design a solution to be used in a VANET that could enable the deployment of virtual functions at the edge of the VANET, using the network architecture concept known as NFV, with the aim of being able to deploy flexible and dynamic Network Service (NS) as close as possible to the end-users. With that in mind, the main achievements and conclusions are presented next.

The design of a NFV solution that can be deployed over a N-PMIPv6 VANET was successfully done. The proposed solution was able to be deployed over a real VANET without requiring any changes to its mobility protocol and only a few changes to the VANET itself. The changes to the VANET came in the form of networking configurations to accommodate for the addition of the extra hardware that is on-boarded in the vehicles alongside the OBUs, in order to allow the full communication between the solution's components using the VANET's mobility network. The proposed solution focused on using the VANET's mobility network simply as an underlay network between the solution's components, by creating a set of virtual networks over the mobility network using overlay tunneling protocols, in this case VXLAN, to be used for the solution's communication.

The proposed solution was implemented using open-source projects and platforms which provided the NFV architectural framework components. The choices for which ones to use were based on solutions that provided production-quality, extensive and working implementations of the components. In regards to the on-boarded hardware, the solution was designed in a way that allows any kind of open hardware platform to be used in the solution as long as the required software can be installed on it.

Using a number of test scenarios, the proposed solution's behaviour was evaluated considering the fact that it was deployed over a dynamic network topology that a VANET provides. The results showed that the proposed solution was capable of deploying the virtual functions at the edge of the VANET without any major problems. Finally, as a way to showcase and present what kind of NSs are possible using the proposed solution, two use cases were developed and tested to demonstrate the advantages of deploying such services closer to the user as opposed to having the services on the cloud.

5.2 Future Work

Even thought the solution developed in this dissertation was capable of accomplishing its main goals, such as the ability to deploy services at the edge of a VANET, there are still ways to improve it in order to increase reliability and up-time of the deployed services.

For Network Services (NSs) which are made up of more than one vehicle, as shown in the use case presented in subsection 4.3.2, when one of the vehicles loses connection to the VANET's infrastructure, the service becomes inaccessible given that the communication between the vehicles is no longer possible. Something that could minimize this problem would be to make use of the VANET's multi-hop capabilities. This VANET feature allows vehicles to communicate with each other even when they are not connected to the VANET, meaning that the service that was deployed could work without connection to the vehicles connection to the VANET.

Something that would also make sense to explore and evaluate would be how the services would behave as well as which new services could be possible when the on-boarded hardware has more resources and a higher performance. The RPis are good platforms but they lack in terms of resources and performance capabilities, which means that not only are the number of VxFs that run on each RPi limited, but the NSs themselves have to be developed to take into account those limitations, meaning that they would also be limited in terms of their complexity. By using on-boarded hardware with more resources as the compute nodes, not only could the services become more complex, but the number of VxFs running at each compute node would increase, meaning that the number of services deployed at once would increase.

The addition of different elements to the VANET could also be considered. By incorporating other elements such as SUAVs into the vehicular network's scope, the types of NSs that could be provided by the solution could be even more complex and dynamic given that the VxFs that make a NS could be deployed on a mix of vehicles and SUAVs. Furthermore, by incorporating the SUAVs, the network's range could be extended, meaning that the vehicles could still be a part of the network even when losing connection with the infrastructure.

Bibliography

[1] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, “A comprehensive survey on vehicular ad hoc network”, *J. Netw. Comput. Appl.*, vol. 37, pp. 380–392, Jan. 2014, ISSN: 1084-8045. DOI: `10.1016/j.jnca.2013.02.036`. [Online]. Available: `http://dx.doi.org/10.1016/j.jnca.2013.02.036`.

[2] “Network Functions Virtualisation (NFV) Architectural Framework”, European Telecommunications Standards Institute, Sophia Antipolis, France, Tech. Rep. GS NFV 002, Jan. 2014.

[3] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, “NFV and SDN—Key Technology Enablers for 5G Networks”, *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2468–2478, Nov. 2017. DOI: `10.1109/JSAC.2017.2760418`.

[4] R. Hussain, J. Son, H. Eun, S. Kim, and H. Oh, “Rethinking Vehicular Communications: Merging VANET with cloud computing”, in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, Dec. 2012, pp. 606–609. DOI: `10.1109/CloudCom.2012.6427481`.

[5] M. Abuelela and S. Olariu, “Taking VANET to the Clouds”, in *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, ser. MoMM ’10, Paris, France, 2010, pp. 6–13, ISBN: 978-1-4503-0440-5. DOI: `10.1145/1971519.1971522`.

[6] J. Jakubiak and Y. Koucheryavy, “State of the art and research challenges for vanets”, in *2008 5th IEEE Consumer Communications and Networking Conference*, Jan. 2008, pp. 912–916. DOI: `10.1109/ccnc08.2007.212`.

[7] P. Papadimitratos, A. D. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, “Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation”, *IEEE Communications Magazine*, vol. 47, no. 11, pp. 84–95, Nov. 2009. DOI: `10.1109/MCOM.2009.5307471`.

[8] D. Johnson, C. Perkins, and J. Arkko, *Mobility Support in IPv6*, RFC 3775 (Proposed Standard), Obsoleted by RFC 6275, Internet Engineering Task Force, Jun. 2004. [Online]. Available: `http://www.ietf.org/rfc/rfc3775.txt`.

[9] C. Perkins, *IP Mobility Support for IPv4*, RFC 3344 (Proposed Standard), Obsoleted by RFC 5944, updated by RFCs 4636, 4721, Internet Engineering Task Force, Aug. 2002. [Online]. Available: `http://www.ietf.org/rfc/rfc3344.txt`.

[10] K.-S. Kong, W. Lee, Y.-H. Han, M.-K. Shin, and H. You, “Mobility management for all-ip mobile networks: Mobile ipv6 vs. proxy mobile ipv6”, *Wireless Communications, IEEE*, vol. 15, pp. 36–45, May 2008. DOI: `10.1109/MWC.2008.4492976`.

[11] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, *Proxy Mobile IPv6*, RFC 5213 (Proposed Standard), Updated by RFC 6543, Internet Engineering Task Force, Aug. 2008. [Online]. Available: `http://www.ietf.org/rfc/rfc5213.txt`.

[12] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, *Network Mobility (NEMO) Basic Support Protocol*, RFC 3963 (Proposed Standard), Internet Engineering Task Force, Jan. 2005. [Online]. Available: `http://www.ietf.org/rfc/rfc3963.txt`.

[13] S. Cespedes, X. Shen, and C. Lazo, “Ip mobility management for vehicular communication networks: Challenges and solutions”, *IEEE Communications Magazine*, vol. 49, no. 5, pp. 187–194, May 2011. DOI: `10.1109/MCOM.2011.5762817`.

[14] F. Teraoka and T. Arita, "Pnemo: A network-based localized mobility management protocol for mobile networks", English, in *ICUFN 2011 - 3rd International Conference on Ubiquitous and Future Networks*, 2011, pp. 168–173, ISBN: 9781457711763. DOI: `10.1109/ICUFN.2011.5949156`.

[15] P. Savola and T. Chown, "A survey of ipv6 site multihoming proposals", Feb. 2005, pp. 41–48, ISBN: 953-184-081-4. DOI: `10.1109/CONTEL.2005.185815`.

[16] R. Kuntz, J. Montavont, and T. Noel, "Multihoming in ipv6 mobile networks: Progress, challenges, and solutions", *IEEE Communications Magazine*, vol. 51, no. 1, pp. 128–135, Jan. 2013. DOI: `10.1109/MCOM.2013.6400449`.

[17] E. Nordmark and M. Bagnulo, *Shim6: Level 3 Multihoming Shim Protocol for IPv6*, RFC 5533 (Proposed Standard), Internet Engineering Task Force, Jun. 2009. [Online]. Available: `http://www.ietf.org/rfc/rfc5533.txt`.

[18] A. García-Martínez, M. Bagnulo, and I. V. Beijnum, "The shim6 architecture for ipv6 multihoming", *IEEE Communications Magazine*, vol. 48, no. 9, pp. 152–157, Sep. 2010. DOI: `10.1109/MCOM.2010.5560599`.

[19] N. Capela and S. Sargento, "An intelligent and optimized multihoming approach in real and heterogeneous environments", *Wireless Networks*, pp. 1935–1955, 2015. DOI: `10.1007/s11276-015-0896-1`.

[20] N. Capela and S. Sargento, "Machine learning for resources prediction in multihoming scenarios", in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–7. DOI: `10.1109/GLOCOMW.2015.7414202`.

[21] M. R. T. Oliveira, "Mobility in vehicular networks with dynamic connectivity and load balancing", Master's thesis, University of Aveiro, Jan. 2016.

[22] "Network Functions Virtualisation (NFV) Management and Orchestration", European Telecommunications Standards Institute, Sophia Antipolis, France, Tech. Rep. GS NFV-MAN 001, Dec. 2014.

[23] E. O. Community. (2016). Osm release one technical overview, [Online]. Available: `https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseONE-FINAL.pdf`.

[24] N. Zingirian and C. Valenti, "Sensor clouds for intelligent truck monitoring", in *2012 IEEE Intelligent Vehicles Symposium*, Jun. 2012, pp. 999–1004. DOI: `10.1109/IVS.2012.6232192`.

[25] S. Kumar, S. Gollakota, and D. Katabi, "A cloud-assisted design for autonomous driving", in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12, Helsinki, Finland: ACM, 2012, pp. 41–46, ISBN: 978-1-4503-1519-7. DOI: `10.1145/2342509.2342519`. [Online]. Available: `http://doi.acm.org/10.1145/2342509.2342519`.

[26] S. Bitam, A. Mellouk, and S. Zeadally, "Vanet-cloud: A generic cloud computing model for vehicular ad hoc networks", *IEEE Wireless Communications*, vol. 22, no. 1, pp. 96–102, Feb. 2015. DOI: `10.1109/MWC.2015.7054724`.

[27] M. Zhu, J. Cao, Z. Cai, Z. He, and M. Xu, "Providing flexible services for heterogeneous vehicles: An nfv-based approach", *IEEE Network*, vol. 30, no. 3, pp. 64–71, May 2016. DOI: `10.1109/MNET.2016.7474346`.

[28] B. Nogales, V. Sanchez-Aguero, I. Vidal, F. Valera, and J. Garcia-Reinoso, "A nfv system to support configurable and automated multi-uav service deployments", Jun. 2018, pp. 39–44. DOI: `10.1145/3213526.3213534`.

[29] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee, "A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters", in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Aug. 2016, pp. 117–124. DOI: `10.1109/W-FiCloud.2016.36`.

[30] A. Hoban, A. Israel, A. Tierno, C. Boyer, F. Salguero, G. G. de Blas, G. Lavado, M. Shuttleworth, M. Harper, and M. Marchetti, "An ETSI OSM Community White Paper, OSM Release FOUR: A Technical Overview", European Telecommunications Standards Institute, Sophia Antipolis, France, Tech. Rep., May 2018.

[31] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*, RFC 7348 (Informational), Internet Engineering Task Force, Aug. 2014. [Online]. Available: `http://www.ietf.org/rfc/rfc7348.txt`.

[32] *Example VNF Charms*, `https://osm.etsi.org/wikipub/index.php/Example_VNF_Charms`, Accessed: 2019.