ICT-777137

5G-RANGE

5G-RANGE: Remote Area Access Network for the 5th Generation

Research and Innovation Action

H2020-EUB-2017 – EU-BRAZIL Joint Call

D5.3 Final report on Network-level mechanisms implementation

Due date of deliverable: 31st December 2019

Actual submission date: 31st January 2020

Start date of project: 1 November 2017 Duration: 30 months

Project website: http://5g-range.eu

Lead contractor for this deliverable: UC3M

Version 5 date 30th June 2020

Confidentiality status: Public

5G-Range receives funding from the European Union Horizon 2020 Programme (H2020/2017-2019) under grant agreement n° 777137 and from the Ministry of Science, Technology and Innovation of Brazil through *Rede Nacional de Ensino e Pesquisa* (RNP) under the 4th EU-BR Coordinated Call Information and Communication Technologies.

Abstract

This document provides a report of the different implementations developed during the 5G-RANGE project related to the Network Layer. Up to the date of this deliverable, an architectural deliverable has already been produced in the project in work package 5 (public deliverable 5.1). That D5.1 deliverable, together with its update, deliverable D5.2, are closely related to this document and there are cross references between them since they are complimentary reports.

In addition, some implementations reported in this document will also be used as part of the 5G-RANGE proof of concept (PoC) and will also receive further attention in upcoming deliverables (WP6).

Finally, different implementations (software) have been released as open source and are also reported in this document.

Target audience

This document is particularly focused on the 3GPP standard for 5G access networks and requires a background in related technologies to be able to fully understand the contribution.

Impressum

Full project title:	5G-RANGE: Remote Area Access Network for the 5th Generation
Document title:	D5.3 Initial Version of the Network-level Architecture and Procedures
Editor:	Francisco Valera, UC3M
Work Package No. and Title:	WP5, Project Management
Project Co-ordinator:	Marcelo Bagnulo, UC3M (EU), Priscila Solis, UnB (BR)
Technical Manager:	Luciano Mendes, Inatel (BR)

Executive Summary

This document provides a report of the different implementations developed during the 5G-RANGE project related to the Network Layer. Up to the date of this deliverable, an architectural deliverable has already been produced in the project in work package 5 (public deliverable 5.1). That D5.1 deliverable, together with its update, deliverable D5.2, are closely related to this document and there are cross references between them since they are complimentary reports.

This document firstly describes the network-level components that have been implemented to support the integration of 5G architecture access network protocol structure into the 5G-RANGE Proof of Concept. These components have been deployed as virtual machines to facilitate their integration but also to be able to be properly deployed and orchestrated using a Management and Orchestration platform (MANO platform). These implementations have been deployed as Open Source and are available in a public directory[1].

In addition, this document includes a description of the different components related to diverse use cases other than the PoC itself that have been published in different conferences or journals.

Finally, this deliverable includes the description of a virtualized environment that has been developed so as to be able to emulate network virtual environments (VENUE[2]). This tool has proven to be very important since it allows testing in laboratory conditions real components without the complexity inherent to field tests. This environment has been used in most of the Network Layer developments and has also been made publicly available under an open source license in a public directory.

[1] Public directory for the 5G-RANGE network layer: http://vm-images.netcom.it.uc3m.es/5GRANGE/

[2] VENUE has been published as open source code in: https://github.com/vsaguero/VENUE

List of Authors

Victor Sánchez (UC3M)

Iván Vidal (UC3M)

Francisco Valera (UC3M)

Marcelo Bagnulo (UC3M)

Marcos Caetano (UnB)

Priscila Solis (UnB)

Cristoffer Leite (UnB)

Geraldo Filho (UnB)

Rafael Amaral (UnB)

Table of contents

List of figures

Definitions and abbreviations

3GPP	3rd Generation Partnership Project
5GC	5G Core Network
5GPPP	5G Public-Private Partnership
VENUE	Virtualized ENvironment for UAV Emulation
PoC	Proof of Concept
UAV	Unmanned Aerial Vehicle
SIP	Session Initiation Protocol
MANO	Management and Orchestration Platform
GW	Gateway
UE	User Equipment
NFVI	NFV Infrastructure
VNF	Virtual Network Function
NFV	Network Function Virtualization
SUAV	Small UAV
5TONIC	5G Telefonica Open Network Innovation Centre
VIM	Virtualized Infrastructure Manager
OSM	Open Source MANO
SBC	Single Board Computer
N3IWF	Non-3GPP InterWorking Function
GRE	Generic Routing Encapsulation
AN	Access Node
RTT	Roundtrip Time
IMS	Internet Multimedia Subsystem
HSS	Home Subscriber Server
CSCF	Call Session Control Functions
DNS	Domain Name Server
IP	Internet Protocol
ITU	International Telecommunication Union
LPWAN	Low-Power Wireless Area Networks
LPWA	Low-Power Wireless Area
CSS	Chirp Spread Spectrum
UNB	Ultra Narrow Band
LoRa	Long Range
DR	Data Rate
SDN	Software-Defined Networking
NS	Network Service
FANET	Flying Ad-Hoc Network
GCS	Ground Control Station
DTN	Delay Tolerant Networking
VM	Virtual Machine
REST	Representational State Transfer
MQTT	Message Queuing Transport
CoAP	Constrained Application Protocol
HTTP	Hypertext Transfer Protocol
TCP	Transport Control Protocol
DSR	Dynamic Source Routing
AODV	Ad-hoc On-Demand Distance Vector
OLSR	Optimized Link-State Routing Protocol
TLS	Transport Layer Security
CPU	Central Processing Unit
LXC	Linux Containers

MPB	Mission Planned Based
LXD	Linux Daemons
RPi	Raspberry Pi
VXLAN	Virtual eXtensible Local Area Networks

1 Introduction

This document provides a summary of the different implementations that have been developed during the 5G-RANGE project related to the network layer (Work Package 5).

These implementations are closely related to the architectural document described both in deliverable D5.1 and deliverable D5.2, where most network layer components are introduced and also will be further detailed in work package 6 final documents with the report of the 5G-RANGE proof of concept.

There is a unique section in this document including all the implementations (section 2) but these implementations have been classified in two main groups (subsections):

1. Subsection 2.1, “Implementation of network-level components to support the 5G architecture into the PoC”, where the document describes the different virtual machines that have been developed in order to be able to integrate different identified 5G architectural components into the 5G-RANGE proof of concept. These implementations have been deployed as Open Source and are available in a public directory referenced in this document. The section includes different details about the configuration of the proposed solution, lessons learned or different implementation alternatives.
2. Subsection 2.2, “Implementation of other network-level components for the use cases”, where different components related to different use cases (not necessarily included finally in the PoC) are described. Most of these components have been used as base implementations for different research works that have been published in different conferences or journals (properly referred in the document).

In addition, this deliverable includes the description of a virtualized environment that has been developed within the framework of Work Package 5 so as to be able to emulate network virtual environments (in particular multi-UAV environments but it can be used in many other scenarios). This tool has proven to be of paramount importance when testing different implementations since it allows testing in laboratory conditions real components without risks and without the complexity inherent to field tests. This environment has been used in most of the developments described in section 2 and has also been made publicly available under an open source license in a public directory.

Further reports on some of these network layer components will be provided within work package 6 deliverables after the final integration and tests with the proof of concept that will be developed in the project.

2 Implementation of network-level components

2.1 Implementation of network-level components to support the 5G architecture into the PoC

In this subsection, we describe the implementation details of the different architectural components of the network layer of 5G-RANGE. These components have been prototyped and validated in the context of WP5. In addition, they will be used to support proof-of-concept activities in collaboration with WP6.

For the sake of completeness, the architectural design of the 5G-RANGE network layer is outlined in Figure 1. This layer includes the components of a **5G core network**, as defined by 3GPP specifications [1] [2], which provides network-layer connectivity to the end-user terminals via the 5G-RANGE access network. These terminals can directly be connected to the access network, as User Equipment (**UE**) or can get network access connectivity through a gateway (**GW**) that is in fact an access router. On the other hand, the architecture includes an IP Multimedia Subsystem core [3], based on the Session Initiation Protocol (SIP) [4], to support the provision of IP telephony services to the operator subscribers (i.e., the **SIP/IMS** core of Figure 1). In concordance with 3GPP specifications, the network layer architecture considers that the different components of the 5G core network and the SIP/IMS core can be virtualized and executed as Virtual Network Functions (**VNFs**). The purpose of the Management and Orchestration (**MANO**) platform, defined by ETSI in the context of Network Functions Virtualization (NFV) [5], is precisely to support the lifecycle management of the network-layer components as VNFs. These VNFs would run over an infrastructure of NFV-capable equipment, i.e., an NFV Infrastructure (**NFVI**), under the control of the MANO platform. In the ETSI NFV reference architecture, the MANO platform comprises the following elements: the Virtual Infrastructure Manager (**VIM**), which handles the allocation of NFVI resources to the VNFs; the VNF Manager (**VNFM**), being in charge of managing and monitoring the lifecycle of the VNFs, including their instantiation, configuration, scaling and deletion; and the NFV Orchestrator (**NFVO**), which coordinates and manages the composition of functional network services across the NFV environment, interacting with VNFM and VIM entities.

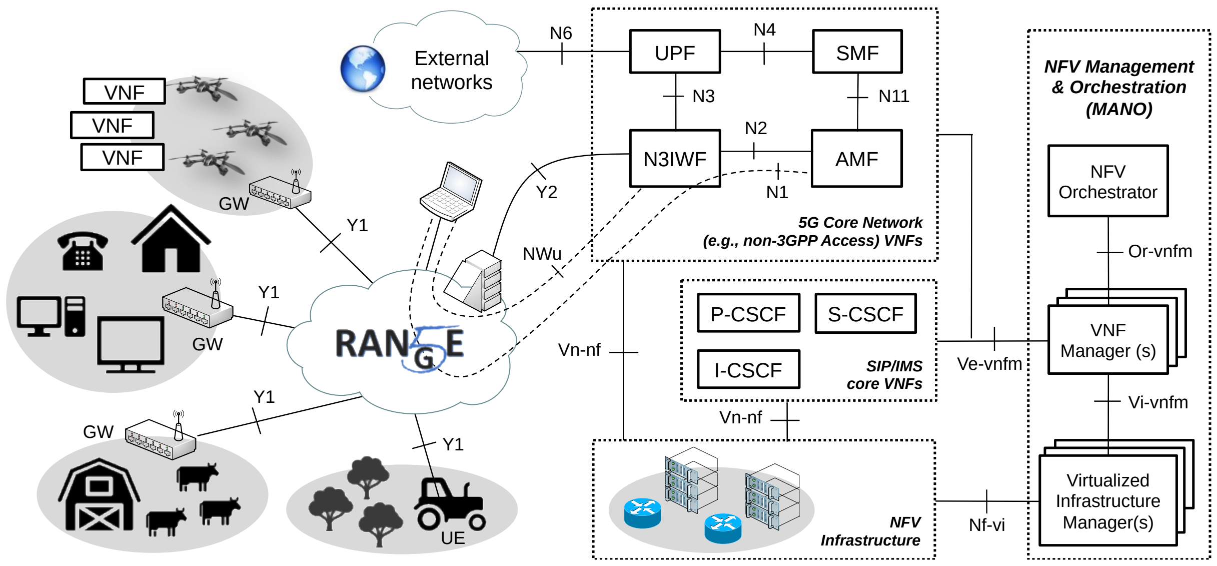


Figure 1. Architectural view of the 5G-RANGE network layer

On the other hand, the design of the 5G-RANGE network layer also defines extensions to complement the network infrastructure of the access network, and provide cost-effective telecommunication services to the end users (e.g., to support voice and data communications beyond the 5G-RANGE radio cells). To this end, the network layer considers the utilization of Small Unmanned Aerial Vehicles (**SUAVs**), which can fly over a geographic area, as well as other vehicles that may be available on the ground (e.g., harvesters, tractors, etc.) and other purpose-specific facilities (e.g., a mobile ground station powered by a generator set, which may have been temporarily utilized to support communications during a specific event). These devices conform an ad-hoc NFV infrastructure which can be leveraged by the 5G-RANGE MANO platform to deploy value-added network services and functions over a specific geographic area.

Deliverable D5.2 [31] provides a comprehensive description of the network layer. In the following subsections, we detail how its architectural components have been prototyped and validated.

2.1.1 Background and related work

2.1.1.1 Technologies for virtual infrastructure management and NFV orchestration

Virtual infrastructure management can be nowadays considered a mature and consolidated technology, with multiple well-known open source and proprietary solutions in the field of cloud computing services. Examples of successful, widely adopted solutions are OpenStack [6], OpenVIM [7], VMWare vCloud Director [8], and Amazon Web Services Elastic Compute Cloud [9]. These solutions are typically used by cloud-computing service providers to support the execution of virtual functions through hypervisor-based virtualization technologies, supported by datacenters with high-profile server computers interconnected by high-speed local networks.

As an alternative to traditional hypervisor-based approaches, container virtualization aims at providing a more lightweight solution, by sharing the operating system kernel of a compute node by all the containerized functions running on top of it. This approach enables to extend the application scope of the VIM towards more resource-constrained environments, where traditional hypervisor-based virtualization may impose inconvenient overheads in terms of computing and storage. Well-known examples of container-based cloud technologies are LXC/LXD [10], Docker [11], or Kubernetes [12]. However, despite their reduced footprint in terms of computing, storage and networking requirements, these solutions still present limited applicability in highly resource-constrained environments, especially in those that may be available beyond the access and the edge of telecommunication operator infrastructures. This is the reason why new alternatives are appearing to fit those requirements, such as K3S [13], a lightweight version of Kubernetes for Internet of Things (IoT) and edge environments; and Fog05 [14], a completely distributed end-to-end virtualization solution for provisioning and managing NFVI resources of IoT environments. The latter is a recent open source initiative, although preliminary tests in the scope of virtual and augmented reality [15] showcase its potential for supporting distributed resource-constrained NFV environments.

Regarding NFV orchestration and VNF management, there are several well-known and widely used open source implementations aligned with the ETSI NFV architectural framework. These include Open Source MANO (OSM) [16], ONAP [17], Cloudify [18], and Tacker [19]. Table 1 collects a summary of different software projects and initiatives that provide NFV orchestration and virtual infrastructure management, with an indication on the capacity of the latter to support resource-constrained devices.

2.1.1.2 Complementing access network infrastructure resources with UAV devices

The utilization of devices with limited computing resources, but with outstanding and inherent capacities in terms of mobility and flexibility, has been considered in the technical literature to enable ubiquitous network and service access, even in rural areas with low population densities. For instance, the authors in [20] propose an architecture for 5G and the upcoming generations of mobile communications that integrates Unmanned Aerial Vehicles (UAVs), aiming at assisting the cellular networks to accommodate occasional flash crowds in specific areas. Works in [21] and [22] address the extension of network coverage in future wireless networks, distributing UAVs around a macro base station to aid relying and sharing of user information, or even to let the UAVs to act as base stations to support end-to-end communications among multiple end-users. Following a similar research line, the work presented in [23] analyzes the next generation of wireless communications, and identifies UAVs as key devices to achieve the expected performance indicators in 5G networks, as enabling platforms to complement the cellular network resources and assist network communications in underserved areas.

Technology	Brief description	Maturity	Functions	Support for resource-constrained devices
OpenStack	Open source software for creating public and private cloud platforms	High	VIM	✗
OpenVIM	Reference VIM included within the ETSI-hosted Open Source MANO project	Medium	VIM	✗
VMware	Cloud solution offering secure, flexible and efficient computing resources	High	VIM	✗
Amazon Web Services	Comprehensive and broadly adopted cloud platform solution	High	VIM	✗
Kubernetes	Open source system that facilitates the orchestration of containerized applications	High	Orchestration of containers	✓
K3s	Lightweight Kubernetes distribution for resource-constrained appliances	At early stage	Orchestration of containers	✓
Eclipse fog05	Decentralized infrastructure for provisioning and managing compute, storage, communication and I/O resources	At early stage	VIM	✓
Open Source MANO (OSM)	ETSI-hosted project that implements an open source MANO software stack aligned with the ETSI NFV architectural framework	High	NFVO, VNFM	n/a
Open Network Automation Platform (ONAP)	Ecosystem development for network automation based on open standards	High	NFVO, VNFM	n/a
Cloudify	Open source cloud orchestration framework	High	NFVO, VNFM	n/a
Tacker	OpenStack project implementing an ETSI aligned VNFM and NFVO	Medium	NFVO, VNFM	n/a

Table 1. Summary of management and orchestration software technologies

Following an interesting and complementary direction, the research community is giving the first steps towards realizing the softwarization of network functions over mobile ad-hoc networks. Whereas these networks have been intensively studied in diverse application fields, their prospect of realizing the view of building a potentially unlimited pool of NFV infrastructure resources beyond the access network, opens new research directions in this field that need to be explored with careful detail. In [24], the authors use Software-Defined Networking (SDN) techniques to manage a heterogeneous vehicular network. The work in [25] proposes a model for a future infrastructure of Internet-of-vehicles using both SDN and NFV technologies. In [26], authors use SDN techniques to manage a flying ad-hoc network built by Unmanned Aerial Vehicles (UAVs). In [27], the authors present a video-surveillance system for large underserved areas using UAVs. NFV is used as the platform to transmit the video signal through a network of several VNFs running on the aircrafts. Considering the limitations on the resources that can be onboarded on the aircraft, the authors propose using para-virtualization, i.e., a solution where virtual machines directly share the hardware with each other, simplifying the interaction between the hypervisor and the operating system. This helps saving valuable execution time and allows a better management of the available resources. The work in [28] addresses the design and validation of an NFV system capable of deploying moderately complex network services over a wireless ad-hoc network of single board computers. The NFV system relies on container virtualization (LXC/LXD) to support the network functions, and on mobile ad-hoc routing protocols to facilitate multi-hop end-to-end communications across the single board computers. The management and orchestration platform was based on OSM and OpenStack. The system design targets SUAV networks, although it is generic to be considered for other resource-constrained environments. As it will be commented in the next section, this system has served as a reference platform to support the SUAV based extensions to the network-layer of 5G-RANGE.

2.1.2 The MANO platform and the NFVI

In the following, we detail the deployment aspects of the MANO platform and the NFVI for the 5G-RANGE network layer. These components have been set up following the methodology and the lessons learned from our work on NFV experimentation platforms [29] and on MANO operations on SUAV networks [28]. The realization of a use case of this platform has been accepted by ETSI as a proof of concept for Open Source MANO (OSM) [30].

The MANO platform and the NFV infrastructure of 5G-RANGE have been deployed in the 5G Telefonica Open Network Innovation Centre (5TONIC) [32], located in Madrid, where UC3M participates as a member. An overview of this deployment is outlined in Figure 2, which also highlights the architectural components of the network layer.

In particular, the MANO platform has been deployed as two independent virtual machines, both running in a server computer with 32 cores, 128 GB of RAM, and a hard drive of 6 TB. In particular, one of the virtual machines provides the functionalities of an NFV orchestrator and a VNF manager, using Open Source MANO (OSM) [34], i.e., an ETSI-hosted open-source project providing a MANO software stack aligned with ETSI NFV. A second virtual machine features a Virtualized Infrastructure Manager (VIM), based on the OpenStack cloud computing platform [35]. The MANO platform controls an NFVI conformed by three server computers, each equipped with 6 cores, 32 GB of RAM, 2 TB of hard drive and 4 GbE DPDK ports. These servers are interconnected by a GbE data-plane switch.

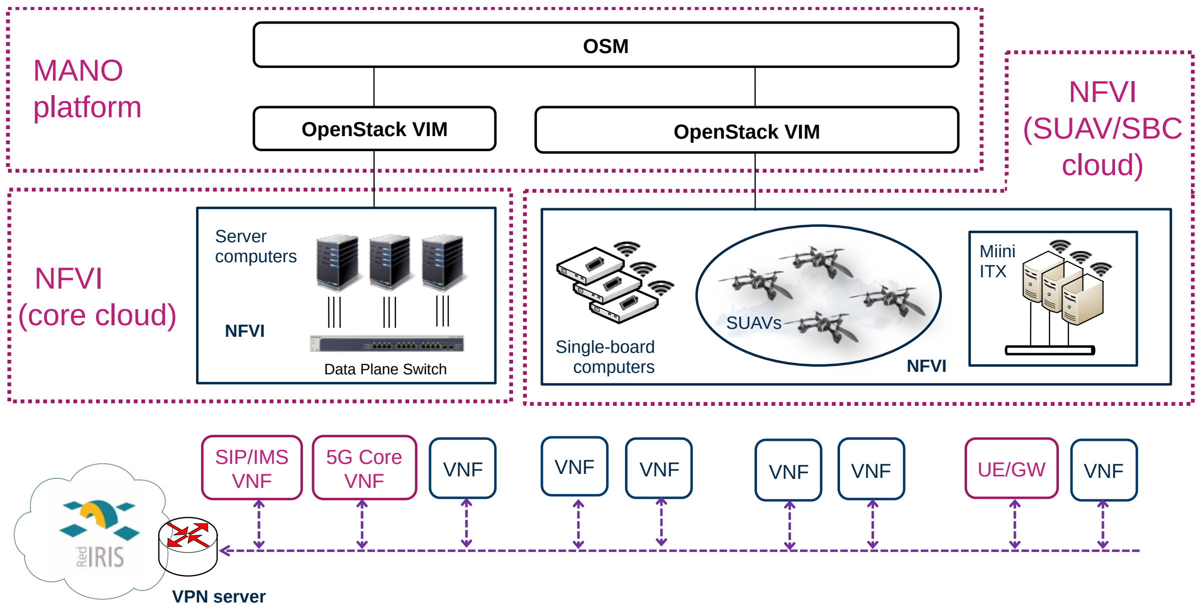


Figure 2. The 5G-RANGE MANO platform and the NFVI at 5TONIC

To support prototyping, validation, and experimentation activities with the ad-hoc infrastructures that are considered in the context of 5G-RANGE (e.g., SUAVs, ground vehicles, mobile ground stations, etc.), a second NFV infrastructure has been configured at 5TONIC. This infrastructure includes six Single Board Computers (SBCs), Raspberry Pi 3 Model B+, each with two Wi-Fi interfaces. Given their size, weight, and their capacity to operate for a reasonable period of time using an external battery, these single board computers conveniently represent the type of resource-constrained platforms that could be onboarded onto SUAV units, as well as onto other devices that can complement the access network infrastructure. To support flight tests, a set of four SUAVs, *Parrot Bebop 2*, each onboarding a

Raspberry Pi 3 Model B+, has also been made available in the laboratory, where a specific location has been enabled for indoor flight experiments. A set of six mini-ITX computers completes the NFV infrastructure, enabling the deployment of more resource-demanding VNFs, which could be executed on ground vehicles and/or facilities in realistic scenarios. Additionally, these computers may serve to execute other supporting functions, such as wireless access points and VPN clients. The SUAV/SBC NFVI can be flexible configured to build different aerial and ground ad-hoc network topologies, with the appropriate disposition of SUAVs, SBCs, and mini-ITX computers. This NFVI is under the control of a portable VIM, based on OpenStack, running in a laptop as a virtual machine. The VIM is also attached to the OSM stack that, this way, has control over the compute, network, and storage resources provided by the SUAV/SBC NFVI.

Finally, we want to mention that tests involving 5TONIC and other partners' facilities is feasible through a VPN service offered by the laboratory. This feature has served to facilitate the integration of the partners' developments (from Brazil and Europe) in the proof-of concept of the project.

2.1.3 5G Core and UE/GW components

In order to provide end-to-end IP network connectivity to 5G-RANGE user terminals in the proof-of-concept, we have produced an implementation of the following components:

- **5G Core**. This component will support the connectivity of any UE and gateways of the proof-of-concept through the 5G-RANGE access network. It has been developed to serve for validation purposes in the proof-of-concept of the project. Hence, the prototype does not provide all the functions and services specified in [1] for the 5G core network. In particular, it does not implement a control-plane protocol stack, as defined by 3GPP.

  The 5G Core component has been prototyped as a VNF. This way, it may automatically be deployed by the 5G-RANGE MANO platform. The VNF provides the implementation of the user-plane protocol stack of a Non-3GPP InterWorking Function (N3IWF), defined by 3GPP, to support the connectivity of UE to the 5G core network via an untrusted non-3GPP access. In addition, it provides routing functionalities towards external networks. This way, the 5G Core VNF can be understood as a component that provides a basic implementation of the user-plane protocol stack for a N3IWF and a User Plane Function (UPF). The development has been based on the Linux *ip-gre* module and the *ipsec-tools* and *racoon* Linux packages, which provide the required support for Generic Routing Encapsulation (GRE) [36] and IP security (IPsec) [37], respectively.

- **UE/GW.** The UE/GW component provides the functions of a 3GPP UE. In concordance with the implementation requirements for the 5G Core component, the prototyped UE/GW does not include all the functions and services specified by 3GPP for the 5G system. In particular it implements the user-plane protocol stack to support the access to the 5G core network through an untrusted non-3GPP access. In addition, the UE/GW supports IP routing functions. Hence, it may be used to enable the access of multiple end-user devices to the 5G-RANGE access network. The UE/GW component has also been implemented as a VNF.

The protocol stacks of both the 5G Core and the UE/GW are provided in Figure 3, along with the resource requirements to support their execution as VNFs. The picture also outlines the role of the 5G Core and the UE/GW in the provision of end-to-end IP network connectivity towards external networks.

To validate the implementation of both components, we built a scenario similar to the one illustrated Figure 3, replacing the Access Network (AN) nodes by a single IP router. This router, along with the UE/GW and the 5G Core components, were deployed as virtual machines over a server computer. The external equipment was connected to the 5G Core using: 1) a 1 Gb/s; and 2) a 100Mb/s data link. Using the *Iperf* tool[3] and the Linux *ping* command tool, we measured the maximum average throughput and the average Round-Trip Time (RTT) that could be achieved between the UE/GW and the external equipment. The results are shown in Table 2. For a better understanding of the impact of the GRE and IPsec overhead on the performance metrics, the table shows the obtained values considering three different cases: 1) the UE/GW and the 5G Core components implement the whole user-plane protocol stack for untrusted non-3GPP accesses, including the GRE and the IPsec layers; 2) the UE/GW and the 5G Core components do not implement GRE tunnelling; and 3) GRE and IPsec are not provided, i.e., the prototyped components only support IP layer functions (they provide a vanilla IP solution).

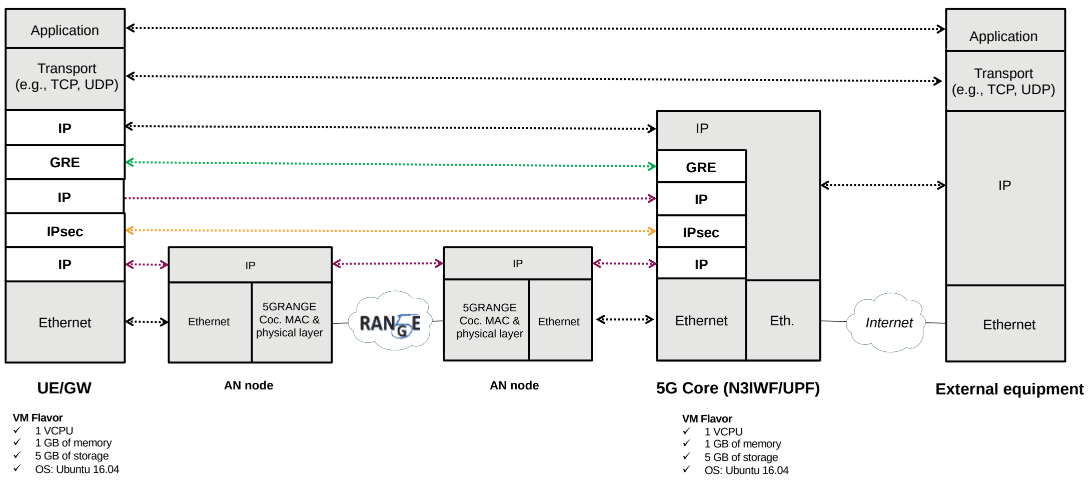


Figure 3. Implementation of VNFs: UE/GW and 5GC

The validation results show a maximum average throughput of approximately 600 Mb/s between the UE/GW and the external equipment, when they are connected through the 1 Gb/s data link. This throughput decreases with the IPsec and the GRE overhead (an overall decrease of approximately 31% when both tunnelling protocols are used), mainly due to the execution of the IPsec cryptographic operations. When using the 100 Mb/s data link, IPsec can process the incoming data packets at the transmission rate, and the throughput decrease with respect to the vanilla IP solution can be considered almost negligible (approximately 0.3%). Given that the throughput required to the 5G-RANGE access network is 100 Mb/s, these results suggest that the developed prototypes could properly operate in the proof-of-concept.

Comparing the RTT values against the vanilla IP solution, we observe a limited impact of the cryptographic and tunnelling operations on delay performance. However, these delay values have been obtained in a simple scenario with no background traffic. A more comprehensive evaluation of the delay metrics will be accomplished over the proof-of-concept in the scope of WP6.

2.1.3.1 Implementation details

To illustrate how the 5G Core and the UE/GW components have been implemented using the Linux *ip-gre* module and the *ipsec-tools* and *racoon* Linux packages, in the following we detail the set of steps that can be followed to provide a functional instance of the 5G Core function as described in Figure 3, starting from a baseline Linux machine. Similar steps could be followed to configure a functional UE/GW.

[3] iPerf3, a tool for active measurements of the maximum achievable bandwidth on IP networks (last access on Jan. 2020): https://iperf.fr

	IP	IPsec	GRE & IPsec
Throughput (1 Gb/s link)	873,30 Mb/s	614,26 Mb/s	600,40 Mb/s
Decrease (max. 1 Gb/s link)		-29,7%	-31,2%
Throughput (max. 100 Mb/s)	94,10 Mb/s	93,85 Mb/s	93,80 Mb/s
Decrease (100 Mb/s link)		-0,3%	-0,3%
RTT (1 Gb/s link)	2,29 ms	2,58 ms	2,69 ms
Increase (%)		0,29	0,40

Table 2. Performance values for the 5G Core and the UE/GW prototypes

In our implementation, we have used a Linux Ubuntu 16.04, although the configuration could be reproduced with other Linux versions considering the specificities of the Operating System (to verify this, we have also reproduced the GRE/IP/IPsec configuration on a Raspberry Pi 3 Model B+ running Raspbian 9 Stretch). We assume that the Ethernet interface of the Linux machine has properly been configured, and that the machine has Internet access. The different steps to configure the 5G Core instance are detailed below.

Step 1. Installation of the required Linux packages (*ipsec-tools* and *racoon*):

```
sudo apt-get install racoon ipsec-tools
```

Step 2. Enable forwarding, such that the 5G Core can behave as a network router. To this purpose, open the configuration file *"/etc/sysctl.conf*" and uncomment the "*net.ipv4.ipforward=1*" line.

Step 3. Create a virtual interface (*tun*) to support the functions of the IP layer that is immediately below GRE in Figure 3, using the following commands:

```
ip tuntap add name <tun_interface_name> mode tun
ip link set <tun_interface_name> up
ip addr add <tun_local_address> dev <tun_interface_name>
ip route add <tun_remote_address> via <gateway_address>
```

Where:

- `<tun_interface_name>` is the name assigned to the *tun* interface.
- `<tun_local_address>` is the IP address to be configured at the *tun* interface.
- `<tun_remote_address>` is the IP address of the correspondent *tun* interface of the UE/GW.

Step 4. Configure the GRE tunnel between the 5G Core and the UE/GW using their *tun* interfaces:

```
ip tunnel add <gre_interface_name> mode gre remote <tun_remote_address>
local <tun_local_address> ttl 255"
ip link set <gre_interface_name> up
ip addr add <GRE_local_address> dev <gre_interface_name>
```

Where:

- `<gre_interface_name>` is the name assigned to the GRE interface at the 5G Core.
- `<tun_local_address>` is the IP address of the 5G Core in the GRE tunnel.
- `<tun_remote_address>` is the IP address of the UE/GW in the GRE tunnel.
- `<GRE_local_address>` is the IP address to be configured at the GRE interface of the 5G Core.

Step 4. Configure the IPsec security associations between the 5G Core and the UE/GW. To do this, first edit the file "*/etc/racoon/racoon.conf*":

```
path pre_shared_key "/etc/racoon/psk.txt";
remote <remote_address> {
       exchange_mode main, aggressive;
       proposal{
              encryption_algorithm aes;
              hash_algorithm sha1;
              authentication_method pre_shared_key;
              dh_group 2;
       }
}
sainfo address <tun_local_address> any address <tun_remote_address> any{
       pfs_group 2;
       encryption_algorithm aes;
       authentication_algorithm hmac_sha1;
       compression_algorithm deflate;
}
```

Where:

- `<remote_address>` is the IP address of the UE/GW in the IPsec tunnel (i.e., the IP address configured at the Ethernet interface of the UE/GW).
- `<tun_local_address>` is the IP address of the *tun* interface at the 5G Core.
- `<tun_remote_address>` is the IP address of the *tun* interface at the UE/GW.

Now, generate a pre-shared key and configure it in the file "/etc/racoon/psk.txt", following the format:

```
<remote:address> <pre-shared-key>
```

Then, configure IPsec operation in *tunnel* mode. To this purpose, update the security policy database in the file "*/etc/ipsec-tools.conf*", using the IP addresses of the abovementioned *tun* interfaces and the following format:

```
spdadd <tun_local_address> <tun_remote_address> any -P out ipsec
       esp/tunnel/<local address>-<remote address>/require;

spdadd <tun_remote_address> <tun_local_address> any -P in ipsec
       esp/tunnel/<remote address>-<local address>/require;
```

Finally, restart the IPSec related services:

```
/etc/init.d/setkey restart
/etc/init.d/racoon restart
```

2.1.4 SIP/IMS core components

To support testing procedures for the voice and data connectivity use case considered in 5G-RANGE, we have prototyped the following components using open-source software tools:

- **IMS Core**. This component provides the main functional elements of the IP Multimedia Subsystem, as defined by 3GPP: the Call Session Control Functions (CSCFs), that is the Proxy-CSCF, the Interrogating-CSCF, and the Serving-CSCF; and the Home Subscriber Server (HSS). The IMS Core implementation was made available as a single virtual machine, which has been produced using an open source IMS implementation, Open IMS Core [38].

- **IP Telephony Server**. It implements the functionalities that are needed to provide an IP telephony service based on the Session Initiation Protocol (SIP) [4]. In particular, it supports the registration of users from SIP compliant IP phones, and provides proxying functions of SIP messages to support the establishment of user calls. The IP telephony server has been implemented as a VNF, using the open source SIP server *Kamailio* [39].

Both components can be used to provide an IP telephony service. In the case of the IMS core, the call signalling procedures are also based on the SIP protocol, following a specific SIP profile defined by 3GPP.

To validate the operation of the IMS Core and the IP Telephony Server prototypes, we have used the UE/GW and 5G Core components, previously described, to build the scenario shown in Figure 4. In this scenario, two users, *Alice* and *Bob*, register into an IP Telephony service and establish an IP telephony call. All the elements shown in the figure, including *Alice* and *Bob's* equipment, have been deployed as independent virtual machines. To create and process the SIP signalling messages at the users' equipment, we have used an open-source SIP test tool, SIPp [40], which enables the execution of scripts to send and receive SIP messages.

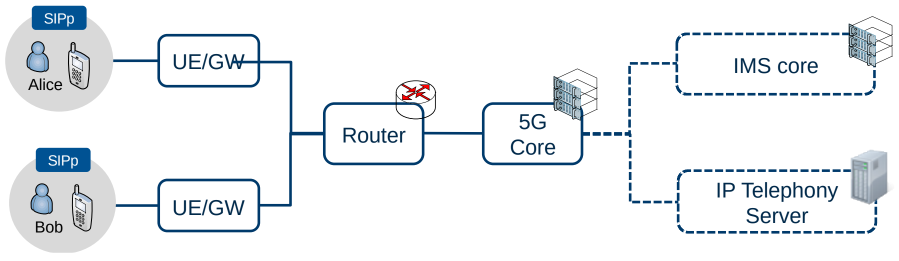
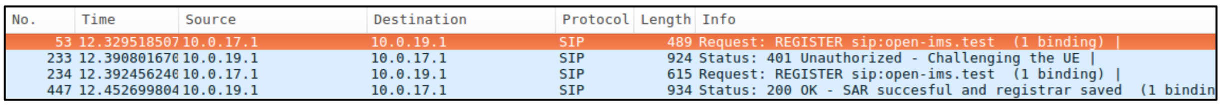


Figure 4. Validation scenario for the IMS Core and the IP Telephony Server

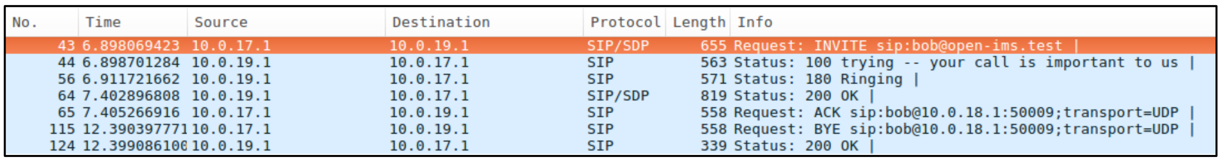
We executed the registration and the call setup procedures using both the IMS Core and the IP Telephony Server. In both cases, we verified the proper exchange of SIP messages by our prototype implementations. As an example, Figure 5 shows a traffic capture, made with the Wireshark network protocol analyser[4], with the SIP signalling messages exchanged between *Alice* equipment (with IP address 10.0.17.1) and the IMS Core (the P-CCSF has the IP address 10.0.19.1). On the other hand, Figure 6 illustrates the SIP messages originated and terminated by *Alice* equipment during the call setup with *Bob* (the capture also includes the message of the call release) through the same IMS Core.

[4] Wireshark network protocol analyser. Available online (last access on Jan. 2020): https://www.wireshark.org



```
No.   Time        Source       Destination   Protocol Length Info
   53 12.329518507 10.0.17.1   10.0.19.1     SIP         489 Request: REGISTER sip:open-ims.test  (1 binding) |
  233 12.390801676 10.0.19.1   10.0.17.1     SIP         924 Status: 401 Unauthorized - Challenging the UE |
  234 12.392456246 10.0.17.1   10.0.19.1     SIP         615 Request: REGISTER sip:open-ims.test  (1 binding) |
  447 12.452699804 10.0.19.1   10.0.17.1     SIP         934 Status: 200 OK - SAR succesful and registrar saved  (1 bindin
```

Figure 5. IMS registration



```
No.   Time        Source       Destination   Protocol Length Info
   43 6.898069423 10.0.17.1    10.0.19.1     SIP/SDP     655 Request: INVITE sip:bob@open-ims.test |
   44 6.898701284 10.0.19.1    10.0.17.1     SIP         563 Status: 100 trying -- your call is important to us |
   56 6.911721662 10.0.19.1    10.0.17.1     SIP         571 Status: 180 Ringing |
   64 7.402896808 10.0.19.1    10.0.17.1     SIP/SDP     819 Status: 200 OK |
   65 7.405266916 10.0.17.1    10.0.19.1     SIP         558 Request: ACK sip:bob@10.0.18.1:50009;transport=UDP |
  115 12.390397771 10.0.17.1   10.0.19.1     SIP         558 Request: BYE sip:bob@10.0.18.1:50009;transport=UDP |
  124 12.399086106 10.0.19.1   10.0.17.1     SIP         339 Status: 200 OK |
```

Figure 6. IMS session setup

Finally, we want to emphasize that, in addition to call signalling procedures, the IMS covers other aspects that are relevant to telecommunication operators. These aspects are related with QoS provision, charging functionalities, security, roaming, and interworking with Internet and circuit-switched network services. For simplicity, and taking into account that these aspects are not under consideration in the 5G-RANGE proof-of-concept, the IP Telephony Server VNF has been selected to provide call signalling functionalities in the validation and experimentation activities of WP6.

2.1.5 Summary of implemented components

Table 3 summarizes the main implementation details about the network-layer components that have been prototyped and made available to WP6 to support proof-of-concept activities. All these components have been implemented with the utilization of existing open-source software technologies. Moreover, the NFVO & VNFM, the VIM, the 5G Core, the UE/GW and the IP Telephony server components have been contributed to the OSM upstream project, as part of an ETSI OSM proof of concept [30].

Component	Footprint	Open source techonlogies
NFVO & VNFM	4 vCPUs, 12 GB of RAM, and 100 GB of storage	Open Source MANO (OSM) Release SEVEN [16]
VIM (core cloud in Figure 2)	4 vCPUs, 16 GB de RAM, 200 GB	Linux Ubuntu 16.04 LTS; OpenStack Ocata [6]

5G Core	1 vCPU, 1GB of memory, and 5GB of storage	Linux Ubuntu 16.04 LTS;
		Linux *ip-gre* module;
		ipsec-tools and *racoon* Linux packages

UE/GW	1 vCPU, 1GB of memory, and 5GB of storage	Linux Ubuntu 16.04 LTS;
		Linux *ip-gre* module;
		ipsec-tools and *racoon* Linux packages

IMS Core	1 vCPU, 2GB of memory, 9.6 GB of storage	Linux Ubuntu 16.04 LTS; Open IMS Core [38]
IP Telephony server	1 vCPU, 1GB of memory, and 5GB of storage	Linux Ubuntu 16.04 LTS; *Kamailio* [39]

Table 3: summary of network-layer components

2.1.6 Lessons learned from the development of the MANO platform and the NVI

As part of the development of the 5G-RANGE MANO platform and the NFVI, we have studied the amalgamation of NFV and SUAV technologies to complement the access network infrastructure resources and support vertical services. A concrete use case of 5G-RANGE making use of these technologies has been published in [41] and is presented in Section 2.2.1. This use case has been accepted by ETSI as a proof of concept for Open Source MANO (OSM), being awarded as the best proof of concept with OSM during the Release EIGHT cycle. In addition, a smart farming use case has also been realized and validated using the 5G-RANGE MANO platform [42].

In the remainder of this section, we identify the main lessons learned from the development of the 5G-RANGE MANO platform and the NFVI, as well as future directions of interest from the work done in WP5. These lessons learned have been published in [42].

2.1.6.1 Software and hardware platforms

A challenging aspect during the development of the MANO platform was to identify the software and hardware platforms that could support effective NFV operations on Single Board Computers (SBCs), while at the same time providing appropriate performance to NFV services (e.g., voice and data connectivity, or smart farming). This was a complex task that required a careful analysis of diverse SBC models, operating systems, and OpenStack releases. More concretely, the fundamental problem was to identify SBC models and operating systems that supported wireless ad-hoc network setups, while at the same time could be integrated as compute nodes into a functional NFV infrastructure. This was the main driver for the selection of Raspberry Pi 3 Model B+ and Ubuntu 18.04 LTS (Bionic) to build the SBC/SUAV NFVIs at 5TONIC, and OpenStack Queens as the virtual infrastructure manager. In any case, these compatibility aspects need careful consideration by stakeholders aiming at reproducing our work.

Regarding the NFV orchestration platform, our experimental results suggest that OSM provides a mature and stable software base to support the MANO functionalities, with the commitment of a strong community to incorporate new and appealing features in subsequent software releases. Still, some OSM features are yet to be explored for SUAV-based vertical services, such as the capacity to scale VNF resources to satisfy elastic service demands, according to different service and resource-utilization metrics.

The MANO platform will be kept running at 5TONIC as a key research asset after 5G-RANGE, and its development will follow a continuous integration approach, allowing a seamless evolution of the operational environment at 5TONIC while the hardware and software base also evolves (e.g., with new OSM and OpenStack versions). As part of the platform development, we will study the feasibility of utilizing Kubernetes clusters to support NFV services over SUAVs, taking advantage of their recent support from OSM Release SEVEN.

2.1.6.2 VNF configuration processing and delay

The virtual machine where we have installed the OSM software stack was allocated 4 vCPUs, 12 GB of RAM, and 100 GB of storage. This comfortably covers the recommended footprint for the proper operation of the OSM software (2 CPUs, 8 GB of RAM, and 40GB of hard drive, according to the OSM documentation). Still, our preliminary experiments indicate that the execution of VNF configuration primitives may impose a significant CPU workload to the OSM machine. This could be a symptom of problematic situations in production NFV environments, hindering the capacity of the MANO platform to perform concurrent operations, such as managing the lifecycle of multiple moderately complex vertical services.

A work line of interest in the NFV context is to consider this aspect, and address the design and implementation of lightweight mechanisms for VNF configuration, aiming at: 1) distributing the configuration burden among sites where VNFs are actually deployed; and 2) reducing the execution delays of configuration primitives. The latter will enable to better support scenarios where the deployment of services must be satisfied with very stringent time constraints (e.g., in emergency use cases).

2.1.6.3 Research on VIM and NFVI solutions

One aspect that was raised when building the infrastructure was the complexity of the mechanisms that were needed to integrate new compute nodes, such as server computers and single board computers, into a functional NFV infrastructure. Whereas the deployment of services over the NFV infrastructure can be automatized by the OSM stack, the composition of the NFVI, as a prior step to accomplishing service deployments, is still a rather manual process requiring complex and time-consuming configurations. This may be appropriate for a static experimentation testbed, but we expect that real scenarios will require a much higher degree of automation and interoperation, when building an NFV infrastructure of SUAVs and other resource-constrained devices that may be available over a geographic area.

On the other hand, our preliminary experimentation stressed the challenges related with the limited operational time of battery powered SUAVs. In fact, this is the reason why SBCs in our testbed are powered by external batteries that provide extended autonomy. While we may expect longer operational times for larger SUAVs in real scenarios, there is a clear motivation to consider battery status of SUAV units to be as relevant as the status of the compute, storage and network resources. Monitoring battery status by a VIM solution may enable enhanced operations, particularly: supporting energy-aware policies for VNF placement; applying optimized replacement mechanisms for SUAVs with the goal of providing resilient services over delimited geographic areas with a limited number of SUAVs; and aiding the coordination of the overall NFV environment, by exposing information regarding the remaining lifetime of devices to the NFV orchestrator (e.g., to take adequate decisions regarding VFN migration, possibly across multiple VIMs).

From a general perspective, all these aspects call for enhanced VIM solutions, capable of deploying virtual functions over novel NFV infrastructures that can be dynamically conformed by resource-constrained aerial devices. As a first step in this work line, WP5 has conducted a theoretical analysis of these and other aspects that is included later on in this document.

2.1.6.4 Emulation of communications channels

Last but not least, we understand that laboratory experiments may not capture all the specificities of real situations, particularly regarding the conditions of the wireless medium and the geographic positioning of SUAVs. After the conclusion of 5G-RANGE, we consider the incorporation of an emulation system into the NFV experimental infrastructure. This will allow to increase the portfolio of experimentation enablers at 5TONIC, supporting experimentation activities with NFV services and resource-constrained platforms, while using realistic communication channels for the SUAVs. As a first step in this direction, WP5 has produced an open-source emulation platform for multi-UAV scenarios. This platform is described in Section 2.3.

2.2 Implementation of other network-level components for the use cases

2.2.1 Implementation of network-layer extensions

As a first step to verify the practical feasibility of utilizing virtualization technologies, small-sized drones, and other ground vehicles and facilities in the 5G-RANGE use cases, we have designed and performed a specific experiment that is illustrated in Figure 7. For the realization of this experiment, we have utilized the VNFs described in the previous section. In addition, we have produced a set of lightweight VNFs that provide diverse network functionalities over the SUAV units. The detailed steps needed to reproduce this experiment can be found in [41]. All the content needed to carry out the experiment, including NFV descriptors and images, is provided in a public repository[5].

[5] The experiment repository is available online at (last access on Jan. 2020): http://vm-images.netcom.it.uc3m.es/5GRANGE/

The experiment reproduces a scenario where an IP telephony service is to be provided in a geographic area beyond the radio coverage of the 5G-RANGE access network. The service is built by the following components, each supported as a VNF:

- **Router/Access Point**: it provides a Wi-Fi access point to end-user IP phones, supporting their automatic configuration through the DHCP protocol; additionally, it provides routing functions for the call signaling messages and the voice traffic of the IP telephony service that are originated and terminated at the IP phones.
- **IP Telephony Server**: this is the operator element responsible for managing the call signaling messages that are exchanged between IP phones to establish and terminate a voice call. For this experiment, we used the IP Telephony Server VNF described in Section 0, which provides a subset of the functionalities of an IMS core.
- **Domain Name System**: it provides a name resolution service, as it is needed to support user identification in a functional IP telephony service. In our setup, this component is implemented along with a routing function in a single VNF (i.e., Router/DNS VNF).
- **UE/GW**: this component implements the protocol stack defined by 3GPP to support the connectivity of the end-user terminals to a 5G core network through an untrusted non-3GPP access. It forwards traffic received from the end-user IP phones towards the 5G core network, and vice versa. This component is provided by the UE/GW VNF implementation described in Section 2.1.3.
- **5G Core**: it provides network routing functionalities within the telecommunication operator domain. To support this component, we used the 5G Core VNF described in Section 2.1.3, which implements the basic user-plane functionality of a N3IWF and a UPF.

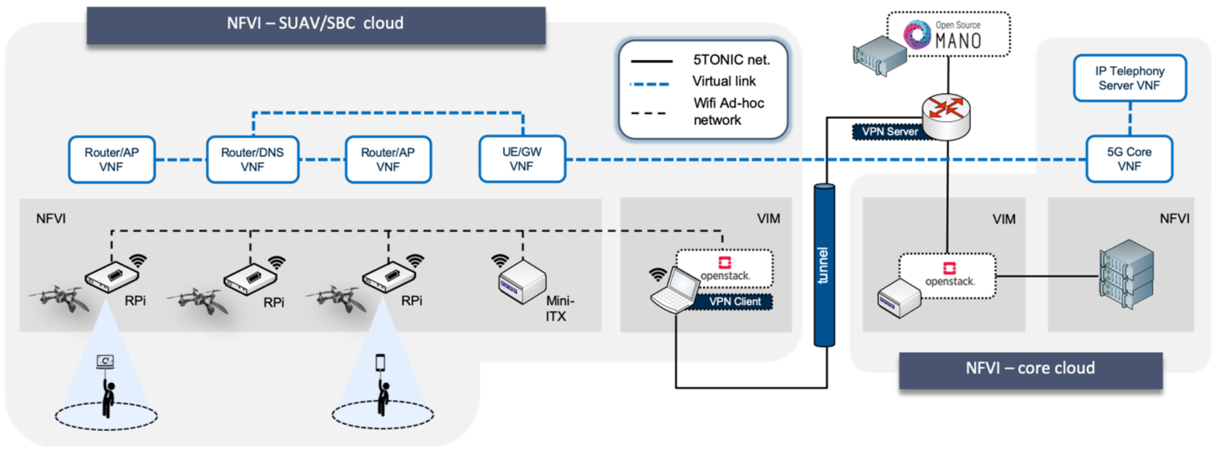


Figure 7. Experiment to validate network-layer extensions

For the experiment, we used the SUAV/SBC NFVI of 5TONIC to create a wireless ad-hoc network of three SUAVs and one mini-ITX computer. We emulated the 5G-RANGE access network using the VPN service over the fixed high-speed network of 5TONIC. Then, we used the 5GRANGE MANO platform deployed at 5TONIC (see Section 2.1) to trigger the deployment of the IP telephony service. The allocation of VNFs to NFVI components is shown in Figure 7.

Once the MANO platform completed the automatic deployment of each VNF, one user attached a softphone[6] to the Router/Access Point VNF offered by one SUAV, and established a call with another user with a phone attached to a different access point. The call establishment took place through the IP Telephony Server VNF deployed at 5TONIC. Right after the call was established, both users could

[6] In the experiment, we used *Linphone*, an open source VoIP SIP softphone for voice/video calls and instant messaging, running it on a laptop. Available online (last access on Jan. 2019): https://www.linphone.org.

maintain an interactive real-time communication, where voice was directly exchanged over the access points and router functions deployed over the SUAVs.

The top part of Figure 8 illustrates the exchange of DNS and SIP signaling messages during the execution of the experiment. These messages correspond to the registration of one of the users in the IP telephony server and to the establishment of the voice call. The bottom part of the figure shows the throughput of the voice traffic observed at one of the Router/AP VNFs. This traffic is delivered using the Real-Time Transport Protocol (RTP) [43].

Figure 9 illustrates the jitter in the forward direction during the call, with an average value lower than 1 ms. Figure 10 depicts the cumulative distribution function of the end-to-end delay measured between two devices connected to different Router/AP VNFs in our experiment. The picture has been elaborated from the Round-Trip Times obtained using the “*ping*” command line tool. More than 80% of the end-to-end delay measurements were below 60 ms, and none of them were higher than 150 ms, which guarantees appropriate delay metrics for the execution of a voice call.

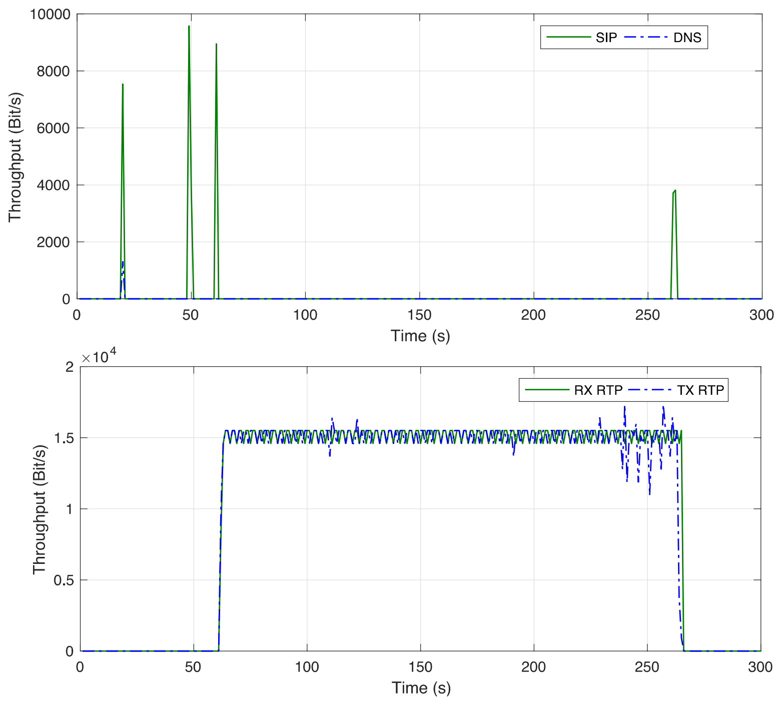


Figure 8. Signalling and voice data exchange (RX = receive, RX = transmit, RTP = real-time transport protocol)

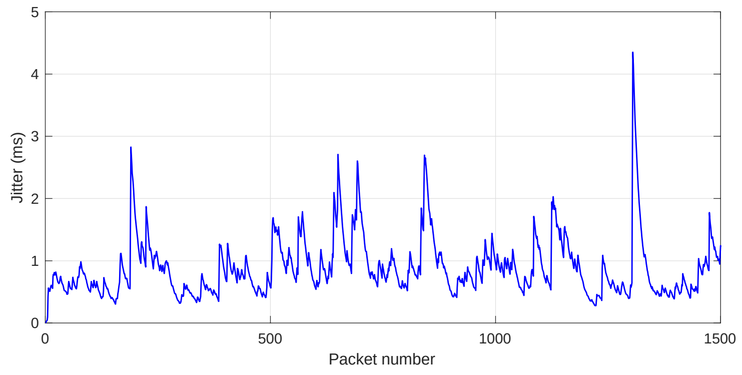


Figure 9. Evolution of the network jitter during the call.

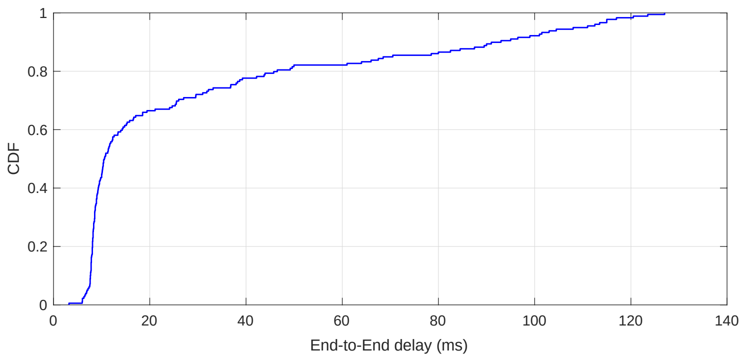


Figure 10. Cumulative distribution function of the end-to-end delay

The results that were obtained in the experiment for the delay figures (jitter and end-to-end delay) satisfy the recommendations specified by the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) [44]. The call progressed without errors and with appropriate sound quality, while one of the involved drones was flying (the drone providing the Router/DNS VNF), demonstrating the feasibility of our approach in laboratory conditions.

2.2.2 Implementation and Evaluation of Low Power Wide Area components

Low Power Wide Area (LPWA) technologies are interesting for IoT applications considering their low power consumption, the expected coverage of tens of kilometers and a promise of ten-year battery life [45] However, LPWA achieve data rates in the order of tens Kbps, which restricts the range of suitable IoT applications for these technologies. Most of Low-Power Wireless Area Networks (LPWAN) operate in the ISM unlicensed frequency band at 169 MHz, 433 MHz, 868/915 MHz and 2.4 GHz. The most common non-cellular technologies being are Chirp Spread Spectrum based (CSS) LoRa, Symphony Link, Ultra Narrow Band based (UNB) SigFox and Weightless. Long Range (LoRa) is an LPWA physical layer standard and, together with LoRaWAN MAC layer, they compose the open networking layer standard maintained by the non-profit association LoRa Alliance [46]. Bidirectional data transfer is possible by using the CSS derivative scheme as modulation, partitioning narrowband signal data over a broad bandwidth and implementing orthogonal division among data streams. The chirp streams shift their frequency at a slow pace while ensuring phase consistency to enable the synchronization of attenuated signals at distant targets. Then, LoRA/LoraWAN is intended for low power applications within large areas and provides data rates between 0.3 kbps and 50 kbps, depending on channel bandwidth, while allowing ranges of 2-5 km in crowded areas.

In the scope of the 5G-RANGE project, as discussed in D5.1, the use of LPWAN technologies was proposed to be investigated for the extension of the network-level architecture (Section 3.4). The LoRa and LoRaWAN technologies were defined as an out-of-band communication channel to transport the control messages generated by the Virtualized Infrastructure Manager (VIM) to the VNFs (Figure 11). Using ETSI NFV standard, with Open Source MANO (OSM) acting as both NFV orchestrator and VNF Manager, while OpenStack controls the VIM. OSM communicates with any VNF to deliver lifecycle management using the Ve-Vnfm interface, while OpenStack communicates with the compute nodes using the Nf-Vi interface to deliver infrastructure orchestration.

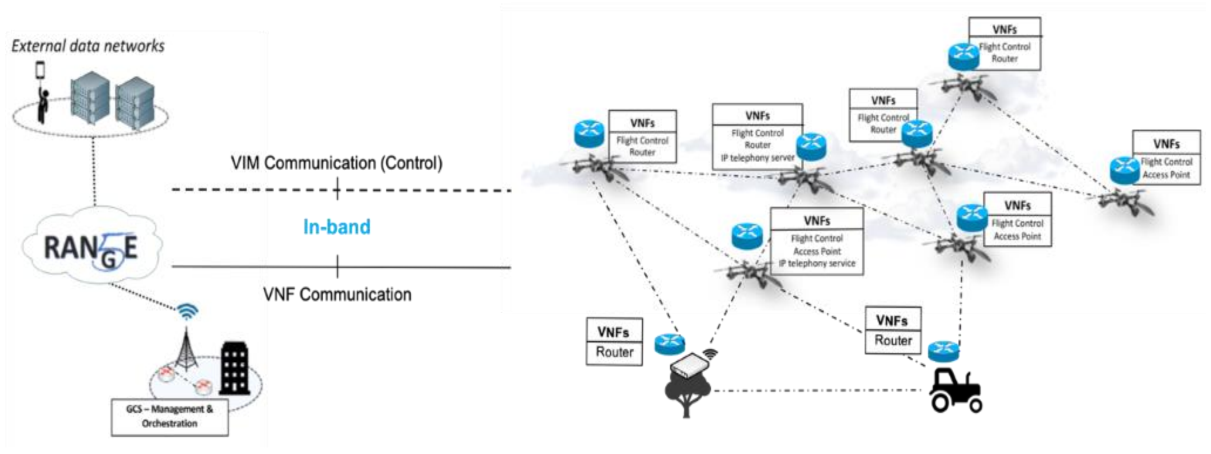

External data networks
VIM Communication (Control)
In-band
VNF Communication
RAN5GE
GCS – Management & Orchestration
VNFs
Flight Control Router
Flight Control Router IP telephony server
Flight Control Access Point
Flight Control Access Point IP telephony service
Router

Figure 11. Scenario for LPWA/LoRa/LoRaWAN Evaluation.

As an example, Figure 12 shows a VoIP based test over the NFV implementation and the results of how a physical infrastructure reacts to virtualized functions running common tasks. The graphic shows the DNS and SIP traffic, in which, the DNS server only produced a single packet of 1,758 bits at 5 seconds, which served SIP Core request for name resolution.

In D5.2 we presented a traffic characterization study regarding NFVI control data traffic to better understand the potential use of other communication technologies to complement the infrastructure described in Figure 11. The implementation was used to evaluate and understand the additional network traffic generated by the virtualized infrastructure, which was based on devices with limited processing resources (in this case, Raspberry Pi - RPi). With that, we made tests to evaluate the data and control traffic in a VoIP connection. The results and traffic characterization, described in D5.2, allowed to better understand the ratios of data and control data traffic that is expected in the NFV infrastructure. With these results, then, in the context of the 5G-Range project, the chosen alternative is to evaluate the potential use of an out-of-band communication channel to transmit the VNFs control data to manage the VNF topology using LoRA, which will allow a cost-effective alternative to expand the coverage of some applications scenarios in rural areas.

The results of traffic characterization in the VNF infrastructure, as described in D5.2, resulted in a total throughput of 44Kbps for a VoIP application, in which 30 Kbps are data (RTP) and 14Kbps are for control data (31%), with a minimum of 6Kbps (14%).

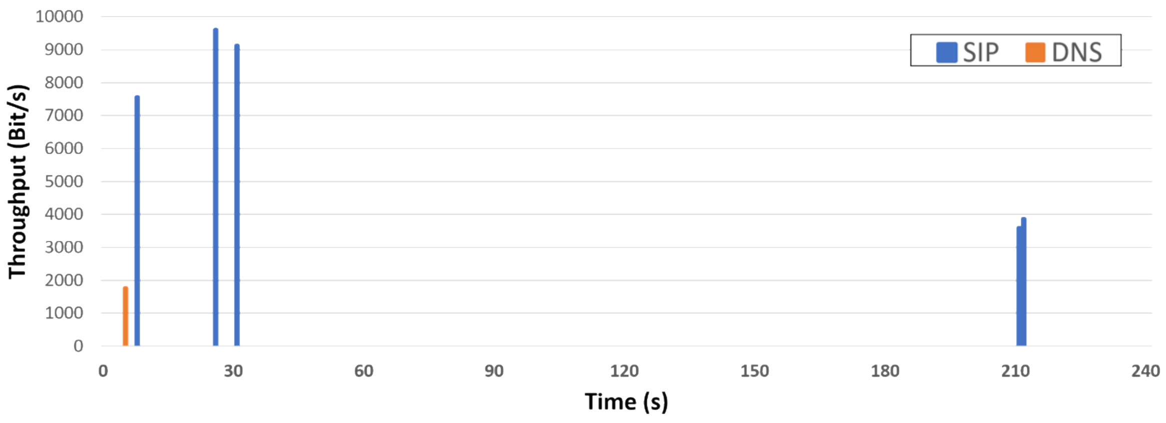


Figure 12. User registration and call signalling messages.

2.2.2.1 LoRaWAN and LoRa Basic Specifications

According to the LoRaWAN Specification [46], LoRaWAN network protocol is optimized for battery-powered end-devices, organized in star topology in which gateways relay messages between end-devices and the central network server. Gateways are connected to the network server through IP-based infrastructure and end-devices use single-hop LoRa modulation to communicate with one or many gateways. LoRaWAN implements a bidirectional communication between end-devices and the gateway, spread in different frequency channels and data rates. The data rate selection is a trade-off between communication range and message duration. Transmissions using different data rates are orthogonal and do not interfere among them. The medium is accessed using a pure Aloha fashion. The MAC layer also defines three classes for scheduling receive windows slots for downlink communication, which are Class A, B and C. The Class A is mandatory, but Classes B and C are optional. Figure 13 describes the LoRaWAN communication stack protocol.

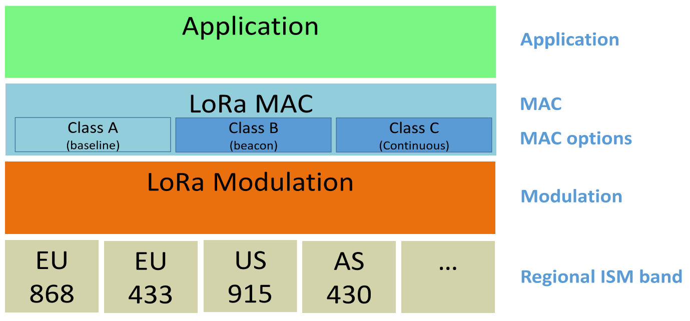


Figure 13. LoRaWAN Classes from [46]

The LoRAWAN Mac classes operation mode are defined as:

- **Bi-directional end-devices (Class A):** allows bi-directional communications whereby each end-device uplink transmission is followed by two short downlink receive windows, as shown in Figure 14. The receive window start times are defined using the end of the transmission as a reference. This class operation mode is the lowest power end-device system for applications, that only requires downlink communication from the server shortly after the end-device has sent an uplink transmission. A downlink communication from the server at any other time will have to wait until the next scheduled uplink.

- **Bi-directional end-devices with scheduled receive slots (Class B):** allows more receive slots than Class A operation mode. In addition to the Class A random receive windows, Class B devices open extra receive windows at scheduled times. In this operation mode, the gateway transmits a time-synchronized Beacon to the end-device to open its receive window at the scheduled time. This allows the server to know when the end-device is listening.
- **Bi-directional end-devices with maximum receive slots (Class C):** the end-devices have nearly continuously open receive windows, only closing when they are transmitting, as shown in Figure 15. In this operation mode, end-devices will expend more power to operate than Class A and B. However, it can offer the lowest latency for the server to end-device communication.

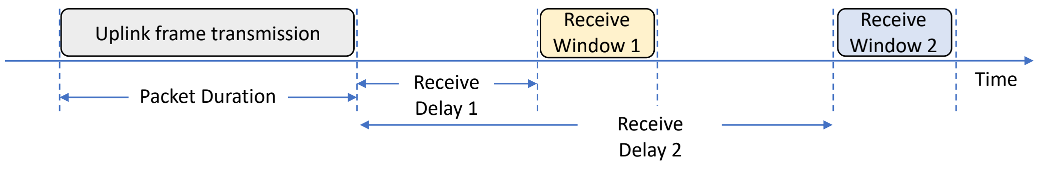


Figure 14. End-device receive slot timing on a class A LoRaWAN device [46]

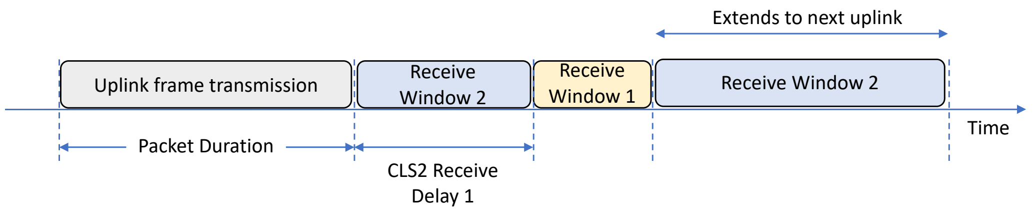


Figure 15. End-device receive slot timing on a class C LoRaWAN device [46]

	Data Rate (DR)	Spreading Factor (SF)	Bandwidth (kHz)	Physical bitrate (bit / sec)
Upstream	0	10	125	980
	1	9	125	1760
	2	8	125	3125
	3	7	125	5470
	4	8	500	12500
	5:7	RFU		
Downstream	8	12	500	980
	9	11	500	1760
	10	10	500	3900
	11	9	500	7000
	12	8	500	12500
	13	7	500	21900
	14:15	RFU		

Table 4. Transmission data rate table for US 902-928 MHz ISM Band [47]

2.2.2.2 LoRaWAN and LoRA Evaluation

To evaluate the use of LoRA/LoRAWAN, the following definitions were implemented:

- use of LoRaWAN Class C operation mode;
- considering the transmission data rate table for US 902-928 MHz ISM Band, as described in Table 4, in which is possible to verify all Data Rates (DR) available to upstream and downstream communication channels. Related with DR values is the Spreading Factor (SF) used by LoRa modulation phy in a specific Bandwidth (kHz), which translates into a Physical bitrate (b/s). We used Data Rate (DR) 4 for upstream and DR13 for downstream, as upper bound limit information to evaluate the data rate achieved in the experiments. Those are the parameters to achieve the highest physical bitrate and they were chosen to evaluate the viability to apply LoRa technology as an out-of-band communication channel to transport the control messages from VIM. This means, according that SF equals to 8 and 7.500 kHz bandwidth and physical bitrate of 12.500 bps and 21.900 bps, for DR4 and DR13.
- 02 (two) LoRa certified Pycom Fipy boards [48] with expansion board features – working on 915MHz at 20dBm maximum, with power consumption of 10mA Rx and 28mA Tx, implementing Class A and C, with theoretical maximum communication range of 40km [49].
- 02 (two) External antenna RF Cable Assemblies SMA(F) JK-IPEX MHF U.FL 1.13 100MM SMA Tilt Swivel 1/2 Wave Whip antenna.
- 01 (one) Raspberry Pi (RPi) 3B+ and 01 (one) RPi 4B to allow a low-cost and power-constrained test scenario. All RPis operate with Raspbian Release from 06-2019.
- 01 (one) LG01-N Single Channel LoRa IoT Gateway, working on 915MHz, equipped with SX1276/SX1278 LoRa module, with maximum communication range between 5 and 10 km [50]. According to its manufacturer it has a limitation of LoRaWAN support [50].
- the LoRa devices were with at least two meters distance among them using direct line of sight.

Figure 16 presents a view of the hardware setup described above.

Two scenarios were defined to perform the evaluation:

- **Scenario 1 –** Fipy Pycom board and LG01-N Dragino gateway. The evaluation is performed between an end-device and a LoRa gateway.
- **Scenario 2 –** Two Fipy Pycom boards. This scenario evaluates the communication performance between two end-devices.

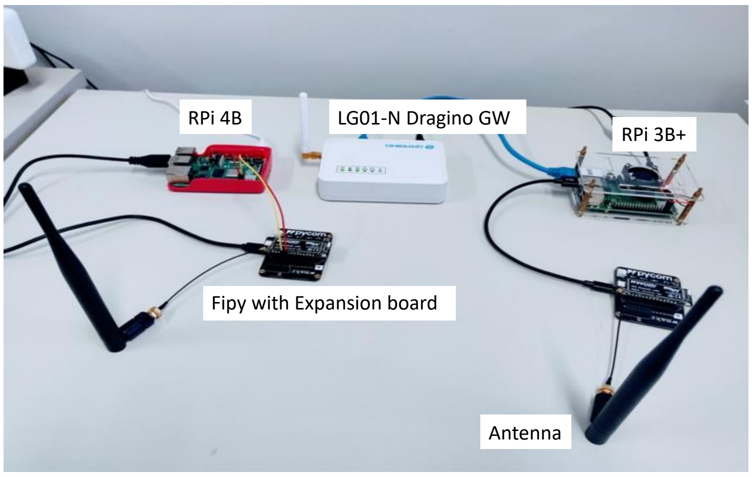


Figure 16. Testbed used in LoRaWAN and LoRa performance evaluation

In both scenarios, the goal is to verify if, considering almost ideal conditions and using a short distance and low interference, the current devices implementing LoRa/LoRAWAN have the potential of providing the necessary throughput for a VNF infrastructure, as the one described in Figure 11.

In both scenarios, the downstream and upstream communication were tested using LoRa, with (Ack-based) and without (non-Ack) confirmation scheme. The confirmation scheme is a naive Ack-based protocol sent by the receiver after it received data information from the sender. This confirmation scheme was implemented and uploaded to the Fipy Pycom board to make the experiment possible. In scenario one, the LoRaWAN was also evaluated in the upstream communication.

For each scenario, the experiment had an average of 10 repetitions with a confidence interval of 95%, based on the t-student distribution. For each repetition, 33 messages were transmitted with the maximum payload size allowed by the LoRa's chipset of the evaluated devices.

Table 5 shows the measured results in the first scenario, with the **FiPy Pycom Board and Gateway Dragino**. As expected, the simple Ack-based protocol had a good impact on the number of delivered messages. It delivered 100% of the messages using the downstream and upstream communication channel. However, it had a direct negative impact on the transmission time, and consequently, the measured throughput. Comparing it with the Physical bitrate from Table 4, the communication between end-device and the gateway, using Downstream and Upstream communication, without the confirmation mechanism (Non-Ack), achieved only 49% and 55%, respectively. Using LoRaWAN class C, the results deteriorate even faster, achieving only 10% of the bitrate defined in the standard.

The second scenario uses **Two FiPy Pycom Board**s, which are LoRa certified devices [48], measuring the communication between two end-devices. The results from this scenario are presented in Table 6. Regarding scenario 1, we observed an increase in the measured throughput. In the best configuration, without confirmation mechanism (Non-Ack), it was possible to achieve 17,377 bps and 10,105 bps, for the Downstream and Upstream communication, respectively. Those throughputs are near to 80% of the Physical bitrate using Data Rate DR4 and DR13. The transmission time also had a huge decrease, near to 10 times in the best case.

	LoRa				**LoRaWAN**
	Downstream		**Upstream**		**Upstream**
	Ack-based	**Non-Ack**	**Ack-based**	**Non-Ack**	
Transmission time (s)	39.67 ±1.10	5.44 ±0.31	25.80 ±1.47	4.86 ±0.14	28 ±0.85
Delivered messages (%)	100 ±0.00	91.8 ±2.50	100 ±0.00	99.4 ±0.30	100 ±0.00
Throughput (b/s)	1,602 ±0	10,660 ±0	1,316 ±92	6,917 ±89	1,180 ±80

Table 5. Results obtained from Scenario 1 using one Fipy Pycom board and one Gateway Dragino

	LoRa			
	Downstream		**Upstream**	
	Ack-based	**Non-Ack**	**Ack-based**	**Non-Ack**
Transmission time (s)	3.37 ±0.01	1.97 ±0.01	4.60 ±0.02	3.48 ± 0.01
Delivered messages (%)	100 ±0.00	100 ±0.00	100 ±0.00	99.4 ± 0.37
Throughput (b/s)	10,588 ±54	17,377 ±59	7,734 ±53	10,105 ±156

Table 6. Results obtained from Scenario 2 using two Fipy Pycom boards

The results show a considerable increase in the measurement outcomes when using certified LoRa devices for communication (Second Scenario). However, even improving the throughput by almost 63% (Downstream/Non-Ack) and decreasing the transmission time near 10 times (Downstream/Ack-based), the evaluated testbed still has space for improvements. The results of both scenarios can be evaluated considering the following factors:

- The first scenario used the LG01-N Dragino, which isn't a certified LoRa Alliance device. In fact, according to its manufacturer, it has a limitation of LoRaWAN support [50]. This could explain the poor measurement results obtained during the experiment, especially using the LoRaWAN Class C operation mode.
- The LoRaWAN Regional Parameters standard [48] defines 250 bytes for the payload using Data Rate of DR4 and DR13. Nevertheless, even using certified LoRa devices this value wasn't possible to be set on the device. During the experiments, the maximum payload allowed kept changing (150bytes, 200 bytes, 230 bytes, etc.), forcing a dynamically choice of the maximum payload value allowed, without generating a firmware transmission error warning. The firmware upgrade didn't solve the problem and the lack of technical documentation about the devices didn't allow us to reach even better results. One possibility is that the lack of 20% throughput, in the Second Scenario could be improved fixing this problem.
- Petajajarvi et al. [51] achieved 45,660 bps using GFSK modulation with a distance between 15-30 km. This result shows the possibility to evaluate other modulation schemes for LoRaWAN, which could provide enough bandwidth to apply this technology in a different context, allowing to explore even more its wide-range and low-power communication characteristics.

Considering the evaluation scenario of Figure 11, the LoRaWAN should support a throughput between 6Kbps to 14Kbps for control data, in order to be considered suitable for the NFVI control messages context. The results in the first experiment (Table 5) show that LoRA can not achieve the desired throughput using both (Ack and non-Ack) confirmation scheme. The results in the second experiment (Table 6) show that using certified low cost devices and device to device communication, LoRa has the potential to be used since fits the throughput requirements. Then, this result shows the potential use of LoRa for providing the possibility to extend the application management area and complementing the applications of 5G-Range. However, further studies should be done to evaluate the use of LoRa with other type of certified devices, integrated into the NVF management infrastructure in real field applications, considering also longer distances, different modulation schemes and modified MAC protocols.

2.2.3 Implementation of endpoint mechanism to resource management - rLEDBAT

We produced an implementation of the designed rLEDBAT mechanism for Linux. In this section, we first briefly describe the implementation and then we describe a number of experimental results obtained using the implementation to verify how the proposed rLEDBAT mechanism achieves the expected goals of its design.

2.2.3.1 Implementation.

The rLEBAT implementation has 4 components: the Sender and Receiver modules and two code hooks in iptables in the Linux kernel (Linux 3.13.0-24 version), one for incoming packets and one for outgoing packets.

The iptables hook for outgoing packets is triggered every time a packet is sent, and the iptables hook for incoming packets is triggered every time a packet is received. We filtered the packets to process by means of destination ports, although other criteria can be used to identify rLEDBAT flows.

Upon the reception of a packet through an rLEDBAT enabled TCP connection, the hook for incoming packets calls for the Receiver module. The Receiver module essentially runs the congestion control algorithm. In order to do that, it first computes the RTT value matching the TSecr of the received packet with the TSval of a previously sent one, and it computes the queuing delay. In order to smooth the value of the queuing delay, we use the minimum of the value measured for the last 4 packets received. With this information, it computes the window to be used by the next packet to be send, with the following parameter values (T=100 ms, a = 0.1, b=0.5).

When a packet is sent through an rLEDBAT enabled TCP connection, the hook for outgoing packets calls the Sender Module. The Sender module then computes the minimum of the Receive window computed by TCP's flow and the one computed by rLEDBAT and includes this value in the Receive Window field of the TCP header of the outgoing packet.

The communication between the Receiver Module (that computes the value of the congestion window) and the Sender Module (that includes the value of the calculated congestion window in the Receive Window field of outgoing packets) is done through a number of global variables using the EXPORT_SYMBOL function of the Linux Kernel.

Regarding the complexity of the implementation, we next describe the number of lines of code of the different elements of the implementation:

Module	Lines of code
Receiver module (code in C)	867
Sender module (code in C)	211
iptables hooks	63
Installation and configuration script (bash)	50

Table 7: Lines of code for the rLEDBAT implementation

The source code for the rLEDBAT implementation is available in rledbat.netcom.it.uc3m.es.

2.2.3.2 Experimental results

We performed a number of experiments using the rLEDBAT implementation described in the previous section to test how it performs in different situations. In particular, we aimed to verify that rLEDBAT complies with our initial design objectives and that it can accommodate the 5G-RANGE use cases and requirements.

As we described in section 4.3.2. of Deliverable 5.2 where the rLEDBAT mechanism design is presented, there are two fundamental observations that guided the design of rLEDBAT. The first observation is that the 5G-RANGE wireless link exhibits large changes in available bottleneck capacity, notably due to the opportunistic use of TVWS. The second observation is that several 5G-RANGE use cases present a mix of traffic with different QoS requirements, including real time traffic with stringent QoS requirements and background traffic with less demanding requirements. Indeed, as presented in Deliverable D2.1, the Voice and Data connectivity over long distances for remote areas use case requires the for a traffic mix that includes both interactive traffic (with a maximum latency between 100ms and 500 ms) and background traffic application with a maximum latency of up to several seconds. Similarly, the Agribusiness and Smart Farming for Remote Areas use case, also calls for a traffic mix that includes some traffic with stringent latency requirements and other traffic with less constraints. Surges in the amount of traffic of one class should not negatively affect the performance of the other traffic class.

The proposed rLEDBAT mechanism to manage available capacity in 5G-RANGE:

- It should accommodate the changes in the available capacity by increasing and reducing the amount of bulk traffic carried, while preserving the performance of real-time/interactive traffic, isolating it from the capacity variations
- It should also be able to handle surges in the amount of background traffic without negatively affecting the performance of real-time traffic.
- It should be able to make full utilization of resources.

We design a number of experiments to show that the proposed rLEDBAT mechanism as currently implemented supports the described use cases.

We start by presenting the testbed used and then the different experiments performed.

Experimental setup

In Figure 17 we present the virtualized environment we use for testing rLEDBAT behavior, featuring a dumbbell topology. C1 and C2 are Ubuntu 14.04 LTS running our rLEDBAT implementation and will act as pure receivers. R1, R2, S1 and S2 are also Linux systems. The link connecting R2 with R1 simulates the 5G-RANGE link with variable capacity, which we set using the tc traffic control tool. The links between S1 (S2) and R2 are configured with larger capacity so they are not the bottleneck. A drop-tail buffer is configured in the R2 to R1 link, with a size we may vary on different experiments, to represent different network conditions. We configure a fixed delay of 15 ms in each direction to represent the propagation delay of the path.

In the different experiments, nodes S1 and S2 transfer data to C1 and C2 respectively. Flow control never limits the communication rate. To compute the rates for each flow, we start a tcpdump capture in R1. The rLEDBAT module also provides traces about the RTT values measured, and the queuing delay derived from them.

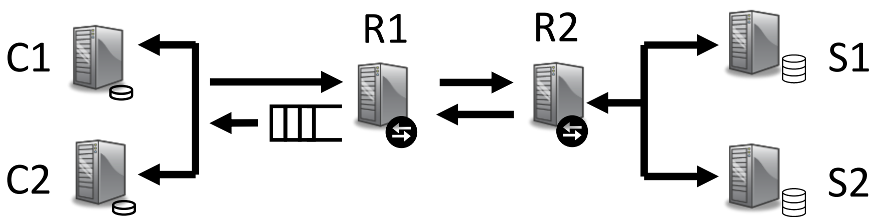


Figure 17. rLEDBAT Experimental setup

Experiment results

In this experiment, we represent a scenario with a mix of real-time and background traffic in a situation with large variations of the available capacity. The scenario is as follows. The client C1 is receiving a VoIP communication from S1. This is implemented as train of 160 bytes-long packets, generated every 20 ms. This results in a flow of 64kbps. This flow is not using rLEDBAT. In addition, client C2 is retrieving a bulk download from S2. This is a greedy flow that uses as much available capacity as possible. C2 is using rLEDBAT to modulate this flow.

In addition, the bottleneck capacity changes during the lifetime of the communication between 2Mbps and 10 Mbps. The actual activation sequence used in the experiment is described in the table below.

Time	t=0 s	t=60 s	t=120 s	t=180 s	t=240 s	t=300 s	t=330 s
VoIP flow	Starts	Cont.	Cont.	Cont.	Cont.	Cont.	Fin
Bulk flow	Starts	Cont.	Cont.	Cont.	Cont.	Fin.	
Capacity	10Mbps	5Mbps	2 Mbps	5 Mbps	10 Mpbs	10 Mbps	10 Mbps

In Figure 18 we plot the RTT experienced by the real-time traffic packets (VoIP stream). We observe that throughout the whole experiment, the RTT remains below the maximum allowed delay of 100 ms, irrespectively of the changes in available capacity and the presence of a greedy flow of background traffic that it using as much capacity as possible.

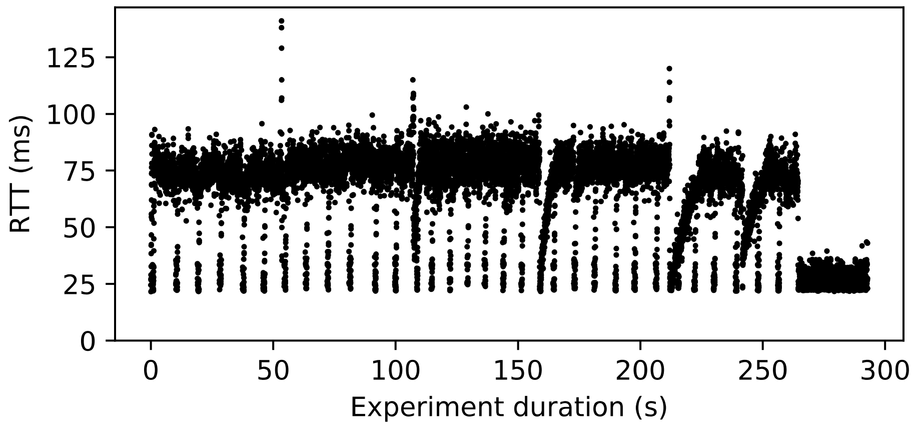


Figure 18. RTT of the real-time traffic with background traffic and capacity variations

In Figure 19 below we plot the throughput used by the background traffic flow. We observe that it adapts to capacity variations to use as much capacity available without negatively affecting the performance of the VoIP traffic.

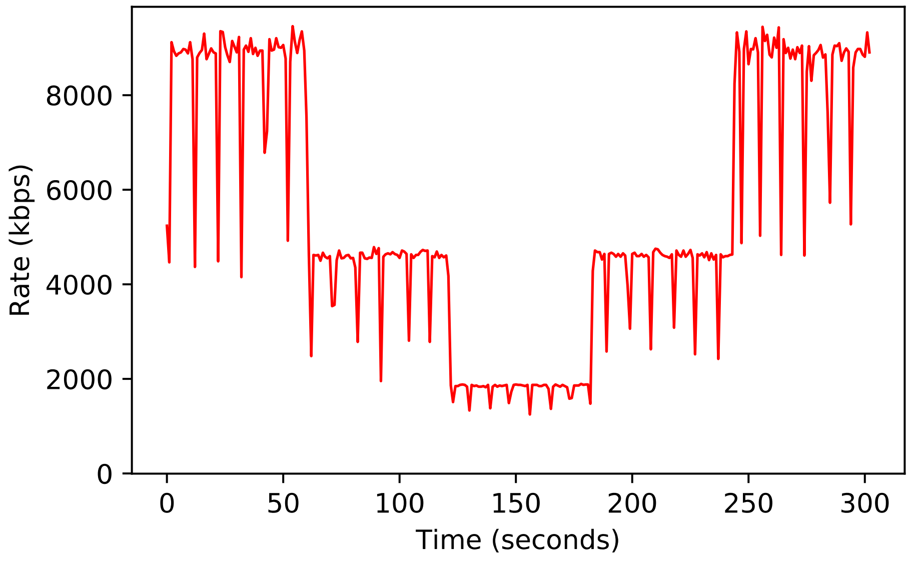


Figure 19: Throughput of background traffic with variations of capacity.

We now analyse the interaction of rLEDBAT with standard-TCP. In this case, we are simulating a scenario where the high priority traffic required as much of the available capacity as possible. In the simulate scenario C1 is retrieving a high priority traffic stream from S1using a standard TCP connection and 30 seconds later, C2 starts retrieving a low priority traffic stream from S2 using a rLEDBAT enabled TCP connection (to reflect the low priority nature of this traffic) Figure 19 shows the rate measured for this case.

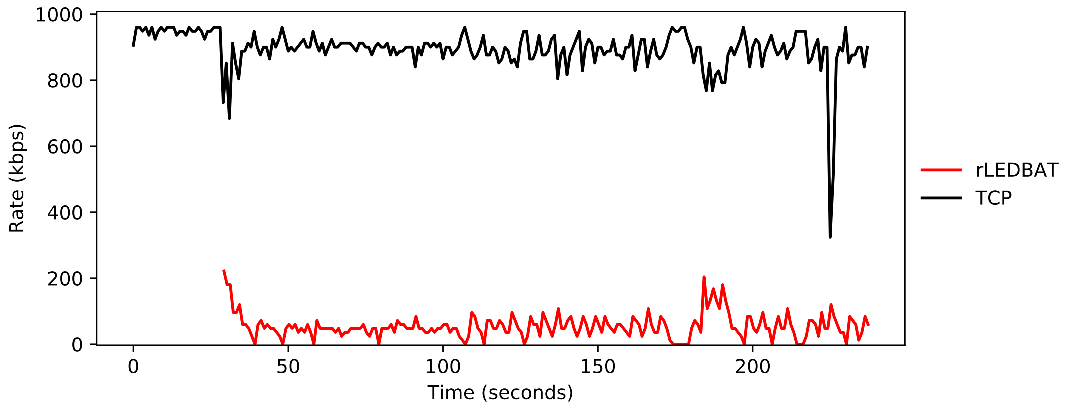


Figure 19. rLEDBAT vs TCP, delay based

We observe that the introduction of the low priority traffic connection does not affect significantly the throughput of the high priority traffic. This guarantees that bulk downloads of low priority traffic will not affect negatively the high priority traffic, even if the high priority traffic requires the use of all available capacity.

2.2.4 Analysis of challenges to NFV orchestration in resource-constrained aerial networks

2.2.4.1 NFV, SUAVs and their synergies

Despite the existing research effort that we have presented, there are still important challenges and hurdles that need to be addressed to effectively support NFV operations over resource-constrained platforms, such as the ones that can be onboarded on SUAVs. In different works [58] and [74] we have analysed and identified these hurdles and this section is presenting part of these works.

Improving data rates and lowering communication latencies are some of the main challenges for developing new infrastructures and technologies for the 5th generation of mobile networks in the near future. However, accomplishing these objectives requires evolving and adapting all current technologies and concepts to produce significant improvements in those areas, by introducing new paradigms with the potential to reshape the next decade of mobile infrastructures. In this fashion, NFV [52] has risen as one of the key enablers of 5G communications, aiming at softwarizing currently deployed physical network functions, i.e., eliminating the usage of specialized equipment to build network functions in favour of using generic hardware.

On the other hand, SUAVs have been an attractive technology for the research community thanks to their increased mobility support with respect to other technologies, such as terrestrial vehicles. There has been quite an effort to develop new applications using these devices in many engineering fields including the communications one. The work in [53] presents a solution that uses UAVs for supporting backbone communications to mobile ground stations. In [54], the authors design a solution to enable a broadband network for emergency communications using geolocalization. The work in [55] presents several architectures for creating a computing platform based on several aircraft, discussing the possibility of using UAVs for data collection in wireless sensor networks as well. In [56], a UAV network architecture is presented to relay data traffic for ground vehicles, which also supports data collection functionalities.

However, according to our previous research on this area, there is still a lack of consensus from the research community and the industry on a standard design for SUAVs that using existing protocols and technologies, enables the flexible execution of general-purpose telecommunication services and functionalities [57]. Despite the market opportunities in this area, commercial SUAV products are typically proprietary solutions, i.e., usually manufactured to accomplish specific missions.

In our future view, swarms combining UAVs of different sizes and characteristics might be positioned over delimited geographic areas, and provide a baseline resource infrastructure, capable of flexibly executing different network functions and services, which could then be offered on-demand by a service provider over that areas. As an example, a telecommunication operator could deploy several SUAVs to temporarily provide Internet access over a specific geographic location, for instance because the access network infrastructure of the telecommunication operator in that area is unavailable (e.g., ina a remote areas or as a consequence of a natural disaster) or insufficient (e.g., in a crowded event). In these cases, SUAV platforms could enable a cost-effective deployment of network functionalities (e.g., wireless access points, routing functions, IP telephony servers, etc.) over a target geographic area. This way, SUAVs could be used to complement the available network infrastructure over specific areas, which could favour the provision of the network services expected for 5G.

To realize this view, NFV emerges as an enabling technology to decouple network functions and services from the hardware they are deployed on: by using virtualization, network services can be executed independently of the hardware and software platforms used underneath. This would allow their execution on multiple and heterogeneous aircraft and devices, as long as these can support virtualization. This way, NFV has the potential to take fleets of SUAVs to a higher degree of flexibility in the provision of network functions and services. A degree of flexibility that can barely be achieved with other specialized SUAV equipment that has been designed and/or configured offline to provide specific services. In addition to its inherent flexibility to adapt the SUAV functionalities to serve to different purposes, the use of NFV brings other relevant advantages in the field of the SUAV arena, particularly: simplifying the development cycle of functions and services, which may prototyped and tested in production-like virtualized environments; easing the evolution of these functions and their migration to production state, as they are implemented in software; and the facility to adapt the resources allocated to specific functions and services to better accommodate varying demands (e.g., an increasing number of users).

2.2.4.2 Orchestration in Intermittently Available Platforms

A prior work of the authors [58] points out the existence of several challenges that should be considered when dealing with NFV in SUAV scenarios, to enable the automatic deployment, configuration, and operation of the aforementioned NSes. In the following, we provide a more comprehensive description of these hurdles, using the reference scenario shown in Figure 20 as an example.

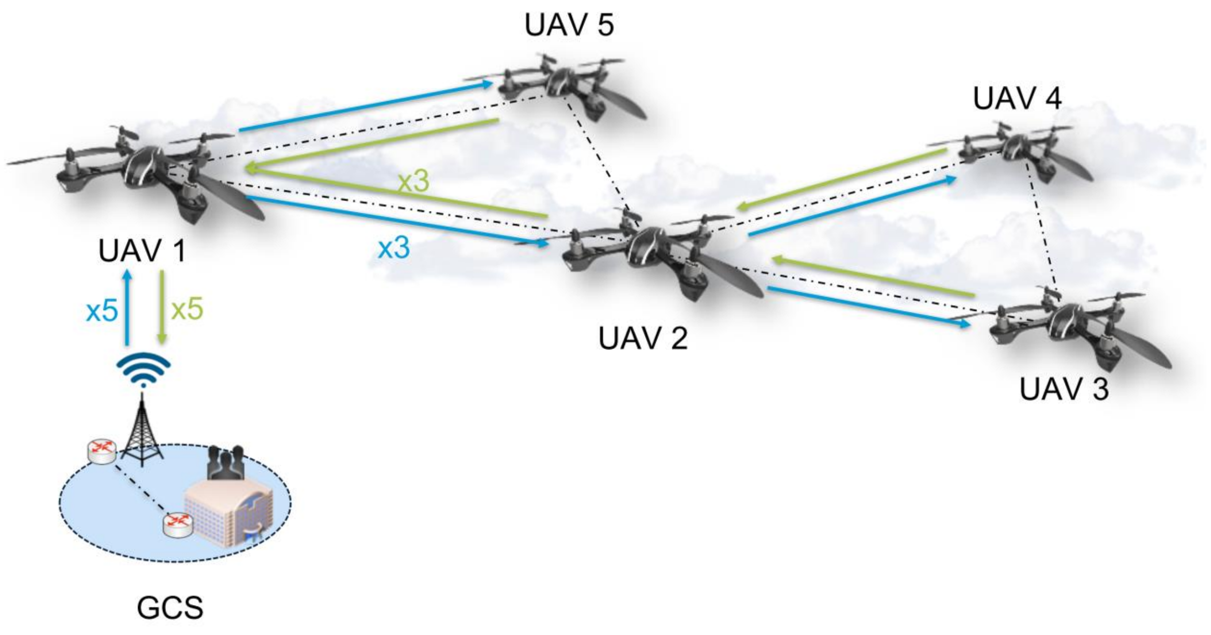


Figure 20. Example of an NFV-SUAV network with their corresponding HTTP orchestration traffic flows

As it can be seen in Figure 20, the scenario is composed of several SUAVs with enough computational, storage and networking resources to build an NFV infrastructure, i.e., each one of the aircraft is an NFVI node or serves to onboard an NFVI node as payload. These NFVI nodes are interconnected through wireless communication technologies (e.g., WiFi of line-of-sight radio links), building a Flying Ad-Hoc Network (FANET) that enables multi-hop data communications. The infrastructure is controlled from a Ground Control Station (GCS), where a VIM is located. The NFVI will communicate with the VIM to coordinate the automatic deployment of NSes over the ad-hoc network formed by the NFVI nodes, as well as retrieving information from the status of the compute, storage and network resources, as well as from the VNFs, from each one of the aircraft. Every aircraft can be placed at any location over a delimited geographical area, either remaining its position static the whole time or moving through a scenario, both autonomously or instructed by an operator from the GCS, depending on the situation and the type of aircraft and/or service.

Despite all its parallelisms with traditional cloud-based NFV infrastructures, there are some differences that must be pointed out. First, SUAVs have a communication range between aircraft due to the limitations of wireless communications. In consequence, they need an adequate distribution to provide enough coverage for the whole target area, implying that not all SUAVs will be able to reach the GCS directly in most cases and, in consequence, VIM-SUAV communication will be mostly performed through intermediate nodes, forwarding both orchestration traffic (i.e., the traffic needed to manage and monitor the available NFVI resources and/or services) and data traffic (information transmitted and/or received by VNFs as part of their normal operation) through the network. Finally, when mobility is enabled, SUAVs may be constantly breaking links that might disrupt SUAVs communication, as new ones must be dynamically created again with other nodes constantly, which can harm certain types of communications. Moreover, these SUAVs need the aid of a battery to stay afloat in the air during an operation/service, requiring a subsequent replacement when this battery is depleted, disrupting even more this communication. Despite all those differences, in a prior work [28], we presented a functional prototype of an NFV system capable of deploying virtual functions and services over resource-constrained SUAV networks. This work served to demonstrate that the introduction of NFV components and processes does not prevent nor negatively impacts the proper operation of the system. Regarding mission planning functionalities, they can be delegated to external modules in charge of specifying the NFV descriptors of the network services to be deployed, in addition to other parameters such as those related to the configuration of the VNFs.

We have classified into different categories the hurdles that could be present when dealing with NFV orchestration in SUAV platforms as well as proposing some alternatives to solve, or at least mitigate, these difficulties in future implementations.

Limited Lifetime of NFVI Nodes

SUAVs are autonomous devices that require some power supply source to be operative. As these units spend most of their time flying, they usually incorporate an autonomous rechargeable battery to act as its power source. Naturally, performing different actions (e.g., routing, transmitting/receiving wireless traffic, executing software functions, etc.) will eventually deplete their battery lifetime: even when they are not executing any specific task, the mere act of staying in the air will reduce their charge until their eventual depletion. Hence, all VNFs running inside its payload must be migrated into another operative unit to avoid, or mitigate, operational cuts on the services being provided. Unfortunately, this action is not as trivial as it may seem at first: not only exists several limitations regarding the computing, storage and networking resources involved in the migration, but also its transition should be anticipated with enough time to ensure a seamless migration procedure, only possible when a VIM is able to monitor the battery lifetime of the aircraft. This VIM could use algorithms to indicate when the aircraft should leave/enter a recharging station such as the one seen in work [59] or [60], allowing it to also predict when the services would need migration before replacing the unit. However, at the time this analisys is being written, most VIM implementations do not consider battery lifetime as a limited resource, making VNF migration in these environments suboptimal. Other measures to preserve battery lifetime could be perching some SUAVs on land, for instance in specific-purpose ground structures, or using different batteries for flight management and computational resources (making sure that the battery on the payload exceeds the flight time of the other one).

Intermittent Availability of Control Communications

As we have previously mentioned, SUAVs battery depletion turns these units into volatile nodes that must be replaced by other units. These disconnections will break links between the SUAVs because they are organized in a multi-hop wireless ad-hoc network, causing communication interruptions between nodes and the GCS. This disruption will be mostly transient however, as either the routing protocols will likely find a new path, and/or the units are replaced after some time have passed to converge into a stable state. Unfortunately, during this period, the units affected may be momentarily unavailable to the MANO system, which will not be able to either deploy or configure VNFs in them, even though those resources will be working properly with their VNFs running normally. Hence, it is important for a VIM to consider that this transient state is the normal behaviour on the aircraft, and not interpret these failures as permanent. One possible solution could be supporting a reasonably increased delay when performing orchestration and/or expecting information from the NFVI nodes inside the network. Another possible alternative to support the increased delay could be using new paradigms such as Delay Tolerant Networking (DTN), whose main characteristics consists of storing its data until there is an available hop to reach the next node to the destination (i.e., it is not necessary to have an available path, but rather finding available nodes and forward the traffic until it reaches its destination). Nevertheless, the VIM should adapt to the increased delay on the traffic caused by the network as well, providing a higher window to receive/send updates of the state of the NFVI resources on the devices. Moreover, battery depletion is not the only phenomena that can produce these communication failures. SUAVs rely on wireless media for their information exchange, which is a far less reliable medium because it can be affected by a lot of parameters involved during the communication, e.g., weather conditions, distance, frequency used, etc., introducing more link breakdowns and eventual reconnections. This problem can be amplified when mobility is added to the equation, as topology changes and aircraft temporarily being out of range will induce more of those disconnections, causing the temporary unavailability of NFVI nodes in consequence.

Limited Capacity of NFVI Nodes

It is important to emphasize that SUAVs are smaller than regular aircraft. In consequence, their payloads must be reduced in weight and size to be onboarded without harming their flight capabilities.

This impacts the nature and the capacity of the available resources in every NFVI unit provided by the SUAVs. This issue has been explained in more detail in [28], where authors encourage the use of lightweight VNFs with container virtualization instead of the traditional hypervisor-based one to overcome this resource limitation. An alternative to these lightweight containers could be the one seen in [27], where authors propose using paravirtualization instead. This technique allows a VM to directly communicate with the OS directly, as opposed to hypervisor virtualization where the VM communicates with its "virtualized kernel". This direct communication speeds the information exchange between the host and the VM (or in this case VNF), increasing orchestration efficiency in the process.

SUAVs limitations are not exclusive to the equipment being used on their payloads: how VIMs exchange information with NFVI units also plays an important role in their communication performance. Most commercial VIMs, which are highly based on OpenStack [6], use HTTP as their default application-layer protocol for exchanging information related with their NFVI resources, getting information about the status of the resources and/or sending actions to be applied on both resources and/or VNFs. With this context in mind, it is understandable the selection of this protocol for this purpose: not only is one of the most used application-layer protocol on the Internet, but also the HTTP embedded Representational State Transfer (REST) model perfectly fits with its information exchange that has to be performed between the aircraft and the GCS. However, this protocol was never intended to operate as a lightweight protocol, as it can be considered a process-intensive protocol compared to alternatives such as the Constrained Application Protocol (CoAP) [61] or Message Queuing Transport (MQTT) [62]. This could be a problem in SUAV networks, since the resources that can be onboarded cannot be very powerful due to both its limited size to fit equipment and the necessity of saving precious battery lifetime. Therefore, some of the aforementioned protocols could be used as alternatives to reduce message overhead for a more cost-effective solution to save battery lifetime and computational resources needed to process the messages.

Transport-Layer Protocols for Control Communications

Continuing with the VIM-SUAVs communication, we must focus on certain aspects related with the behaviour of HTTP that harms the overall performance of NFV orchestration in SUAV scenarios. HTTP uses TCP as its default transport-layer protocol. Although this protocol is one of the main pillars of telecommunications in current networks, certain works have shown that TCP performance over (mobile) ad-hoc networks makes it not the most appropriate choice for them, as it can be severely lowered compared to completely wired networks [63], [64], [65], [66] and [67]. On its conception, TCP was developed as a protocol for reliable data transfer over wired networks. In most cases, link-related error would rarely occur, and most packet losses would be related to congestion. However, wireless technologies completely changed how communications could be done using more than cables to interconnect devices, allowing free node movement as well as flexibility to the distribution of a network, but also introduced new challenges that increased link-related failures due to several factors such as the instability of the medium and mobility. In consequence, TCP was never intended to deal with temporary and/or common link failures, which could potentially affect its use in wireless environments, creating new drawbacks that had to be dealt with in order to ensure efficient communication through this medium. Most of them can be found in [63], [64], but the most relevant for NFV orchestration in SUAV networks are the following ones:

- Physical layer: Wireless links induce more errors in the packets. Problems such as fade-away and interferences can modify bits during the transmission, while poor reception/channels will create packet losses, severely impacting its performance [63]. The reason for this decrement is a constant retransmission of packets due to these losses/modifications, keeping its congestion window low and reducing the overall throughput in consequence [64].
- Network layer: Routing in ad-hoc networks has some differences over traditional networks, using protocols such as Dynamic Source Routing (DSR) [68], Ad-hoc On-Demand Distance Vector (AODV) [69] or Optimized Link-State Routing Protocol (OLSR) [70]. These protocols assume that nodes form dynamic topologies that may change over time. Therefore, they need

some time to recompute their routes when one or more nodes become unavailable before reaching a stable state, which can be dependent on parameters such as the chosen mobility model [65]. However, the disparity between the time to recompute a route and TCP retransmission timers may cause the sender to use outdated routes, increasing the number of retransmissions performed and, again, reducing the congestion window along with its throughput. This problem is further exacerbated when the number of intermediate nodes is increased, as they must also recompute routes as well [66].

Most of the problems described above can be summarized in one sentence: TCP is not able to recognize the source of the failure in wireless media. Therefore, it invokes the only mechanism to, in theory, increase overall transmission performance: the congestion control mechanism. However, here it is disadvantageous because, in almost all cases, there is no congestion in the network, but the protocol interprets SUAV disconnections as a congestion problem even though it is not the case. Hence, this reaction will decrease throughput, harming overall performance instead of increasing it since there is no congestion in the network, as the control information is usually not high in comparison with other types of data exchange between nodes.

Although TCP might not be the most suitable option to transmit information in SUAV environments, we still want some of its characteristics when transmitting control information in a NFV context, as it is vital to ensure reliable data transfer to every node inside the network. However, we also need to avoid harming its throughput and performance as much as possible for control communications, which is the main problem that TCP can introduce in these environments. One possible alternative to solve it could be using protocols that rely on UDP as its main transport-layer protocol, moving its reliability mechanisms to the application layer since this information should safely arrive to the aircraft. In this fashion we can find CoAP, which tries to apply the HTTP REST model to constrained environments but reducing its overhead size as well as using UDP as its transport-layer protocol. Similarly, another attractive protocol that uses UDP is QUIC [71], which tries to mimic an implementation of HTTP (version 2 [72]) and Transport-Layer Security (TLS) [73]. QUIC would allow keeping the REST model intact used for the exchange of information between the VIM and the aerial nodes. In any case, we want to highlight that QUIC is still a connection-oriented transport-layer protocol, which implements a congestion control mechanism based on that of TCP. Therefore, it may be sensitive to the same limitations that can be observed for TCP.

In [74] we perform a detailed analysis of TCP performance and we also compare it with other alternatives.

Enhanced Policies for VNF Placement

Recalling how VIMs work in NFV orchestration, they usually consider certain parameters for allocating VNFs on the NFVI nodes (e.g., CPU, memory usage, etc.). However, battery life is not included among those parameters, so VIMs do not take it into account when deploying/managing VNFs on NFVI nodes. Nonetheless, if a VIM were able to use this parameter in SUAV NFV environments, SUAVs could potentially improve its energy efficiency by assigning critical VNFs to healthier nodes, i.e., those aircraft with extended battery lifetime. Moreover, orchestration could also be improved by using this parameter, as knowing the remaining capacity is essential to trigger VNF migration (discussed in more detail in section 3.1) and support re-allocation policies. Another aspect that needs to be addressed is the placement of SUAVs over an area. In traditional scenarios, NFVI node placement might not be as critical, as these nodes are placed in fixed locations interconnected through a cabled network. In this case, however, SUAVs can move around a delimited field, but its target position for their trajectories should be provided to the flight control engine running at the SUAV. The authors in [91] propose the implementation of the flight control engine as a VNF on every unit, using a VNFM for providing its flight trajectories as configuration parameters. Applying this method, these trajectories could be specified in the deployment to the NFVO, or even appear in the NS descriptor to be configured on the NS deployment.

2.2.4.3 Challenge Summary

In this subsection, we aim to summarize all the challenges and hurdles of resource-constrained aerial networks in Table 8, introducing the main problems of each category as well as some alternatives that could provide a solution to them.

Table 8. Challenges of NFV orchestration in Resource-Constrained Aerial Networks

Object of Analysis	Challenges/Hurdles	Potential Approaches
Limited lifetime of NFVI nodes	• SUAVs are autonomous devices that require some power supply source to be operative, and performing different actions such as traffic relaying or executing software functions will eventually deplete their battery lifetime. • All VNFs running inside its payload must be migrated into another operative unit to avoid, or mitigate, operational cuts on the services being provided. • There are no known VIM implementations that consider battery lifetime as a limited resource, making VNF migration in these environments suboptimal.	• Develop a VIM able to consider battery lifetime for scheduling • VNF migration in advance to increase NFV management and orchestration performance using planning algorithms. • Perch some SUAVs on land in specific-purpose ground structures. • Use different batteries for flight management and computational resources
Intermittent availability of control communications	• Battery depletion turns these units into volatile nodes that must be replaced by other units. These disconnections will break links between the all SUAVs and the GCS, potentially leaving some nodes unavailable to the MANO system. • This disruption will be mostly transient however, as either the routing protocols will likely • find a new path, and/or the units are replaced after some time have passed to converge into a stable state. • It is important for a VIM to take into account that this transient state is the normal behavior on the aircraft, and not interpret these failures as permanent. • The inclusion of mobility and other wireless phenomena (fade-away, weather, etc.) could increase this problem since links breakdowns will be more frequent.	• Develop a VIM able to take into account the intermittent availability of control communications by supporting a reasonable delay for management and orchestration actions/information retrieval for the NFVI nodes.
Limited capacity of NFVI nodes	• The reduced size of SUAVs constrain the available resources for their payload, including its weigh, size and processing power. • Most commercial VIMs use HTTP as its default application-layer protocol, which is not the most appropriate choice for resource-constrained environments, as it could be considered a process-intensive protocol for this units due to the amount of processing power required for its usage.	• Reduce the necessary resources to support NFV by using techniques such as lightweight VNFs or paravirtualization. • Use lightweight protocols/IoT such as CoAP or MQTT to reduce message overhead for a more cost-effective solution to save battery lifetime and reduce the computational resources needed.
Transport-layer protocols for control communications	• On its conception, TCP was developed as a protocol for reliable data transfer over wired networks. In most cases, link-related error would rarely occur, and most packet losses would be related to congestion. However, wireless technologies completely changed how communications could be done, but introduced new challenges that increased link-related failures due to several factors such as the instability of the medium and mobility. • TCP was not intended to deal with temporary or common link failures, which may affect its use in wireless environments, creating new drawbacks that had to be dealt with in several layers such as the physical and network layers. • In summary, TCP is not able to recognize the source of the failure in wireless media, having to use the congestion control mechanism when there is no congestion in the network, decreasing the overall throughput that can harm its management and orchestration performance because its traffic is not high in comparison with the available bandwidth. • Even though TCP might not be the most suitable protocol for this exchange, there are some characteristics we want to preserve such as reliable data transference or retransmission policies.	• Use protocols that rely on UDP as its main transport-layer protocol, while moving its reliability mechanisms to the application layer. Examples of this solutions could be: 1) CoAP, which applies the HTTP REST model to be available in resource-constrained environments. 2) QUIC, which is a new transport-layer protocol to imitate an implementation of HTTPv2 + TSL using UDP.
Enhanced policies for VNF placement	• VIMs do not take into account parameters such as battery lifetime or geographical positions when allocating VNFs on NFVI nodes.	• A specialized VIM could take into account battery lifetime to assign critical VNFs in healthier nodes. Moreover, orchestration could also be improved by using this parameter, as knowing the remaining capacity is essential to trigger VNF migration as well as re-allocation policies. • To manage the placement of SUAVs in a geographical area, implement the flight control engine as a VNF on every unit, providing its flight trajectories using a VNFM as configuration parameters, which could be used to specify their movement at the deployment of the NS.

2.3 Implementation of a virtualized environment for network emulation

2.3.1 Introduction

Multi-UAV systems like the ones presented in this document and in D5.2, do present some challenges that must be solved for suitable performance. Some of these essential design challenges are associated with wireless communications as it has been previously commented, because of the particular characteristics of multi-UAV networks [75].

As it has been seen in the use cases presented in D5.1, D5.2 and in this document, the environment for service provisioning that multi-UAV systems are showing is a particularly appropriate context for the 5G softwarization technologies. In this scenario, one of the most prominent challenges that still must be faced is the enormous existing gap between all these new solutions and their deployment in real scenarios, since field tests are difficult (complex, but also quite restrictive due to regulations) and expensive to perform, and simulation alternatives are not appropriate because they have not been specifically designed for these use cases. Consequently, to address the inherent challenges of multi-UAV systems and solutions, an intermediate step between the design and the real validation process is required.

Taking into account the aforementioned considerations, this section presents VENUE (Virtualized Environment for Network Emulation) an open-source validation platform for multi-UAV and FANET scenarios where different services based on 5G programmable UAVs can be deployed and tested. This solution is built on top of Linux Containers (LXC, [10]) and the ns-3 network simulator (based on [76]), and provides an emulation framework that allows the integration of different Virtual Network Functions (VNFs) (or virtual entities in general), which can be later used into real SUAV hardware, together with a network simulator (used to emulate the specific characteristics of wireless channels).

Besides, the framework enables real hardware (that can be on-boarded into SUAVs as payload) to be directly integrated with the simulation environment, in order to not only test the performance of applications and developments but also to test the correct operation in the hardware that will host the developments in the real world.

VENUE allows testing a wide variety of scenarios including protocols, services, and technologies embedded in the on-board computer of the UAVs. This includes FANETs communication technologies based on standard protocols such as OLSR, AODV or BATMAN, or based on new trends such as SDN. This contribution also includes the possibility of testing from traditional services and applications such as voice over IP or video streaming, to innovative services based on virtualization or 5G technologies such as NFV.

Another of VENUE significant contribution is the possibility to interact with real hardware that is frequently used as a UAV payload. The real payload allows not only to test the development functionality but also to test if the hardware has adequate resources. Modifications had to be made to the ns-3 source code to allow the interaction on different entities (virtualized or real hardware), as this functionality is not entirely supported.

The platform also enables to test scenarios that require network nodes mobility. In this respect, it supports Mission Planned Based (MPB) mobility pattern [77], i.e., predetermined trajectory information, which is usually planned in advance. This way, UAV mobility patterns can be specified by the platform user following a predetermined format described in [78]. The platform supports the installation of a predefined mobility pattern for each of the UAVs in the emulation.

Therefore, it opens the possibility to use any of the mobility models defined in [79]. Each UAV follows MPB information with realistic flight traces.

Finally, the platform incorporates a number of pre-created modules that enable scenarios with the utilization of different routing protocols (e.g., OLSR, SDN based routing solutions), as well as a set of supporting tools to test user applications and developments. These modules can be flexibly incorporated into user-defined test scenarios, as they have been implemented using virtualization containers.

For more details about VENUE please refer to [80]. This emulator has been published as an open source contribution in [78].

2.3.2 VENUE Design

VENUE has specifically been designed to satisfy different aspects that are not fully supported in current network validation solutions (simulators/emulators), and that is important to be able to validate current FANET related technologies.

As it is detailed in this section, the most relevant requirements that have guided VENUE design are (i) The necessity to provide mobility models to accommodate real UAV applications and that are undoubtedly important since they define the FANET physical topology evolution during the service time (VENUE can introduce two-dimensional (x,y axes) mobility patterns in a predefined altitude (z-axis)).

Second, (ii) the necessity not only to simulate services but also to be able to emulate them, testing the real applications that can later be migrated into real hardware. VENUE can serve as an emulator of the physical channel linking real applications and processing real packets (the platform can model and select the communication channels and technologies between the different participants of the FANET). This feature also enables VENUE to serve as a validation platform until almost the final integration with the hardware devices where the services will be installed (in fact, this hardware can also be directly attached to the platform to validate this phase too as it will be seen). Finally, (iii) the necessity to support 5G softwarization technologies, such as NFV and SDN [81], [82] and [83]. These technologies are being introduced into the UAVs field as innovative alternatives to be able to support flexible (and agile) service provisioning (with NFV) but also as a possibility to evolve current ad hoc networks routing protocols adapting them to the particularities of FANETs (with SDN). VENUE improves the support provided in ns-3 to be able to integrate multiple external nodes into ad hoc networks emulating the communication channel. These external nodes can incorporate the corresponding network functions into the system and test them all together.

One of the main strengths is that the framework operates with real applications, e.g., Linux Containers, Virtual Machines, real hardware, that can be directly used in real infrastructure afterward (consequently, VENUE is suitable for both IPv4 and IPv6). This functionality allows reducing prototyping and validation cycles and reducing the time-to-market or time-to-operation period. The framework is also suitable to emulate both wired and wireless (infrastructure-based and ad hoc) networks. Since this implementation is designed to trial FANET scenarios, all the examples provided in the framework are based on the 802.11 Wi-Fi technology.

Still, with small effort, this platform may also assist in wired network scenarios.

The platform also enables more than one real host (virtual or physical) to interact with the network emulation because of the source code modifications (available in the patch file [78]). This functionality is crucial since there are usually numerous participants in a FANET. To facilitate the prototyping of all this potential participants, the framework incorporates pre-created Linux Containers that include the installation and configuration of different FANET routing protocols, such as the OLSR, SDN technologies (like RYU [84] controller and OVS [85]), and network analysis tools (iPerf [86] and Trafic [97]). In combination, the ns-3 simulator is installed and configured inside a Linux Container. Thus, the emulator can be used as a standard virtual function, with all the advantages it brings (the platform can be instantiated by any Virtual Infrastructure Manager (VIM) such as OpenStack and is portable to any Linux machine without any specific configuration). The platform also incorporates some scripts that configure the emulation environment for a smooth development process. This functionality allows the user to create complex multi-UAV networks in a simple way enabling advanced experiments to be carried out. To analyse and measure the service performances, ns-3 generates standard network traffic traces [87] that assist the process of code debugging and traffic analysis.

The results can be studied using regular tools like Wireshark [88], which is utilized for network troubleshooting, analysis and also allows us to analyse the traffic that passes through a network and thus can solve or even prevent possible problems that may arise.

Finally, by default, the framework incorporates the Mission Planned Based (MPB) mobility model [77]. The platform user can introduce real flight traces (VENUE works with two dimensional traces, but the analysis is considered to be correct since landing and take-off are made far from the network service area and in the network service area in most situations it is reasonable to consider the UAVs flying in the same plane) following a predefined format to enable realistic multi-UAV mission.

2.3.3 VENUE architecture

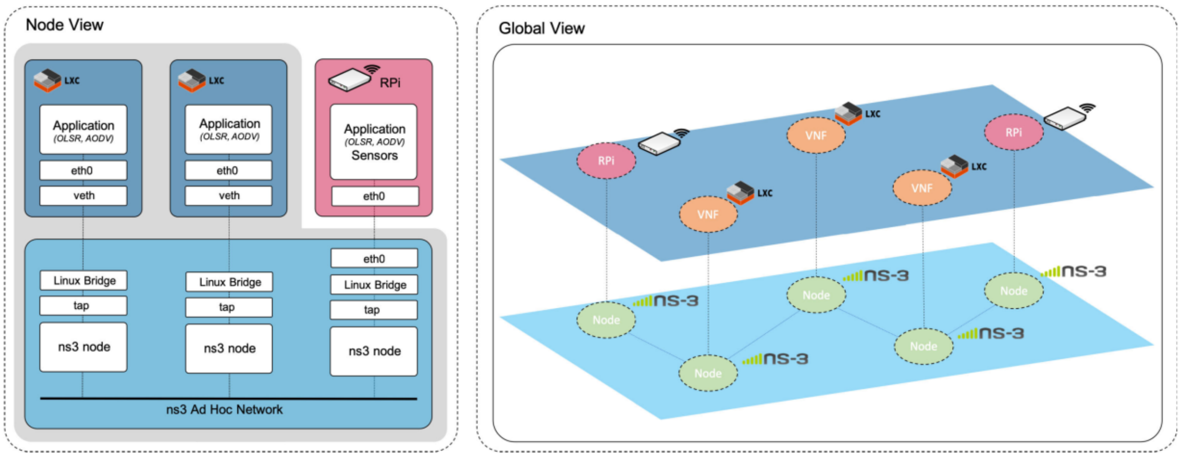


Figure 21. Framework architecture, node and global view

Figure 21 summarizes the architecture of VENUE whose components are described in the following subsection. The global view of the system (complete architecture) is shown in the right part of the picture. The emulation layer is in charge of creating and modelling the FANET, and it also emulates UAV mobility. The top layer represents the real nodes (virtual entities and general-purpose hardware). Those nodes contain the developments and applications that enable the multi-UAV scenario (routing protocols, network services, data collection/transmission, etc.). The integration of the two layers not only allows linking real nodes through an emulated FANET but also provides each UAV with mobility. In the left part of the figure (node view), the connection between the ns-3 nodes and the real nodes is highlighted. Moreover, it includes all the required components to make the association possible such as Linux Bridges (software used to join two or more networks that behave like a virtual network switch) or TAP interfaces (network interface entirely supported in software) [76].

To implement the emulation layer, the ns-3 network simulator has been selected. ns-3 is a discrete event-based network simulator frequently used in the investigation of ad hoc mobile networks. It implements a wide variety of routing protocols from both wired and wireless networks. ns-3 also allows the configuration of several network parameters; meanwhile, it provides extensive data collection modules for exhaustive analysis. However, the main reason to choose ns-3 is that in addition to its simulation capabilities, it implements a module that allows network emulation, i.e., this module allows external entities (real or virtual) to interact with the ns-3 environment. The emulation module provides a fully real-time controllable and reproducible environment where the routing protocols and application developments can later be used in UAV equipment without modifications.

This functionality introduces a notorious added value as compared to pure ns-3 simulations since the ns-3 developments cannot be used in real equipment.

In FANETs, the number of participant nodes may be high, and in order to be able to validate these scenarios beyond the integration of real devices (to test real hardware inside a FANET with numerous participants), VENUE must be able to handle inputs from multiple virtual devices allowing that way to provide a scalable solution. Thus, the experimentation is not only limited by the availability of physical devices (which in these cases are usually expensive) but also by the selected virtualization technology that must be as lightweight as possible aiming to run several singular applications (each of the SUAVs inside the multi-UAV system) on a single server. For this purpose, the Linux Containers have been selected. However, LXC is the tool that VENUE provides to ease prototyping; the developed application could then be used on any virtualization platform or directly on the physical UAV device. In Figure 21 (Node view), it can be seen that all the required applications to enable communications between the UAVs (e.g., OLSR, SDN) are in the LXC (or in real hardware). This feature allows the tested applications to be portable to commodity equipment.

A Linux Container is a set of processes that are detached from the rest of the Operating System. Unlike standard Virtual Machines (VM), Linux Containers share the kernel with the operating system and separate the application from the rest of the system. LXC is portable and modular, including in the production stage (for instance, the same Linux Container can run in different hosts using an NFV platform).

These characteristics make the prototyping and development process faster in comparison with traditional test environments. Nevertheless, the containers must be compatible with the underlying operating system (because LXC share the kernel). The selected hypervisor to manage the Linux Containers is the Linux Daemon (LXD) [92]**¡Error! No se encuentra el origen de la referencia.** because of its simplicity. LXD adds new possibilities and functionalities compared to the conventional system container management of LXC, e.g., container migration or physical devices passthrough. These Linux Containers have proven to be usable in UAV regular payload equipment (see our previous work in [28] [91] where the design of the solution is based on NFV and lightweight VNFs).

Although one of the most significant strengths of the platform is the possibility of virtualization, this framework also allows the interaction with real hardware, allowing VENUE users to get an insight about the behaviour of real hardware during a mission. Typically, UAV embedded equipment is reduced in size and limited in both computing capacity and battery. Therefore, although the developed application may have the correct functionality, it cannot be guaranteed that the hardware in charge of its execution will have enough resources. Thanks to this integration, the limitations of the multi-UAVs payload hardware can be estimated. To integrate real hosts (including either real hardware, Raspberry Pi (RPi) in Figure 21, or virtual hosts) into the ns-3 emulation, it is necessary to use the TapBridge Model [93]. This module allows the replacement of specific nodes (previously determined by the user) from the ns-3 network by real hosts. The TapBridge Model overwrites the ns-3-device MAC address by the overlying real-host (virtual entity or UAV payload) MAC address. After this association, the real-host considers the ns-3 net device as a local device and the TapBridge Model sends all the ns-3 node incoming network traffic through a virtual TAP interface (which is connected to the LXC container or the UAV equipment through a Linux Bridge as it is shown in Figure 21). Similarly, the Tap Bridge Model sends all the outgoing traffic (virtual entity or UAV payload) through the emulated ad hoc network.

Thus, real devices can communicate with each other using the underlying network created by the ns-3, as can be seen in Figure 21. TapBridge uses an existing TAP interface previously created and configured by the user. Nonetheless, in VENUE, the process of creating and configuring the system environment has been automated, and the user only needs to provide some input parameters, according to the experimentation scenario. More details can be found in [76] and [78].

However, the default TapBridge device presents problems to perform more than one ‘‘<ns-3 - real host>’’ MAC association in wireless networks (to connect more than one real device to the emulated network). For VENUE, it has been required to apply some modifications to the source code of the ns-3 to make the connection between several real hosts and several ns-3 nodes. These bugs have been reported [83] and are under revision. However, in the meantime, a patch file with the corrections is provided to apply those changes in [78].

2.3.4 VENUE example

In this section we show how VENUE can serve to validate a complex scenario such as the one explained in Figure 22: an NFV platform that uses a FANET to allow a smooth deployment of a VoIP service (including, for instance, the automatic deployment of virtualized SIP servers, DNS servers, or OLSR routers using OpenStack). Also, the VNFs have been instantiated into real hardware that is ready to be included as SUAV payload (Raspberry Pi). It must be clarified that a MANO system was not used to configure the VNFs in order to simplify the experiment explanation (detailed information about this configuration can be found in our previous work [28] [91]).

In this experiment, VENUE will be used to emulate a scenario closer to the real flight conditions. In this scenario, it is not only expected that the VNFs can interact with each other or that the whole service or the FANET itself can be adequately established. All these things are assumed to have been tested in advance with regular trials. The main goal is to verify the scenario under the changing conditions that there will be when SUAVs take off. However, it will be possible to appreciate in the FANET the effects of the SUAVs movement, the loss of network connectivity implications in the routing protocols. It is particularly relevant (because these previous things can already be done with some existing simulators) to see the effect of this intermittent connectivity in the real deployed VNFs in the real services under execution and in the NFV orchestrator that is managing the whole service.

This scenario includes a three SUAVs fleet and a ground control station intended to be used, for instance, to enable communications in emergencies as it can be seen in Figure 22.

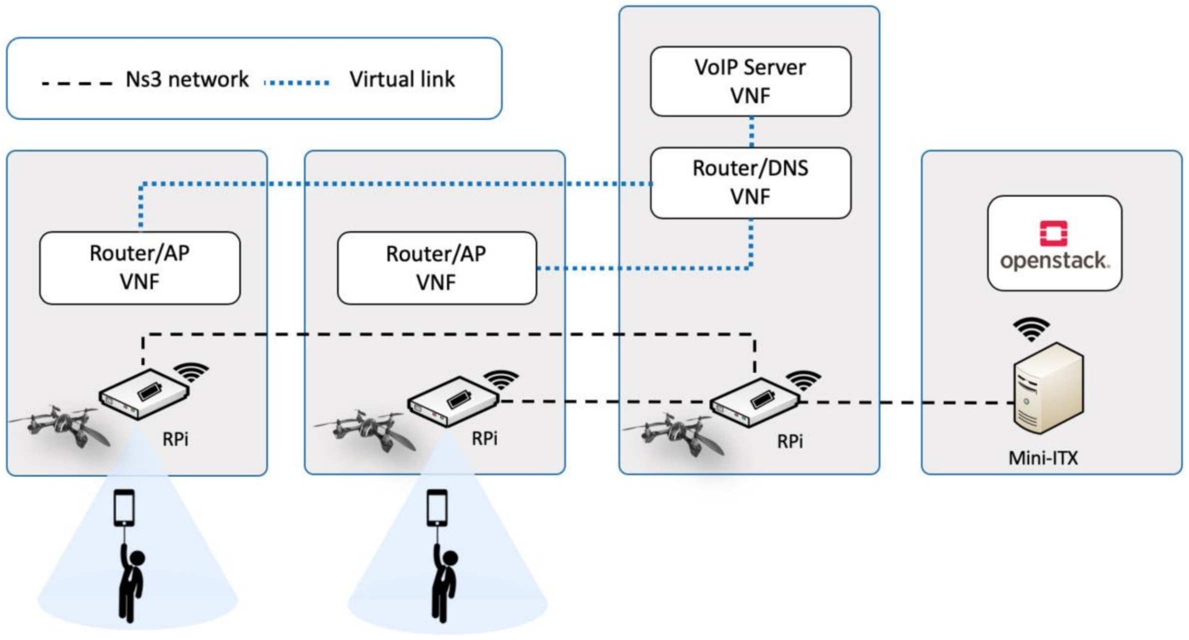


Figure 22. Multi-UAV network to extend 5G connectivity

The service has been instantiated using a set of virtual functions and virtual networks that operate on top of the FANET to provide a flexible and dynamic connectivity backbone and service deployment.

Each SUAV will carry as payload a Raspberry Pi 3B (RPi), [94] single-board computer (SBC). All the RPis include an external battery-power supply (3.7 V and 3,800 mAh) so that the network service operation does not affect the SUAV battery itself (which is intensively required by the SUAV engines). The selected hardware is not a random choice. The small size of the RPi (85.60 mm×56 mm×21 mm), in combination with its reduced weight, allows almost any commercial UAV, e.g., DJI Phantom 4 or Parrot AR Drone, to fly loading these devices without any problem. In Figure 23, it can be seen how the payload has been incorporated into the aircraft. The ground control station is a mini-ITX computer (Intel Core i7 2.3 GHz, 16GB RAM, 128GB SSD, 4 GbE ports) that also acts as a cloud operating system

(OpenStack [6]). This equipment can be appreciated in Figure 23. As it is shown in the picture, the physical network topology is emulated using VENUE (VENUE emulates the realistic conditions of the wireless ecosystem), and on top of the network, a virtual network service has been deployed. Two SUAVs (1 and 2) also provide real Wi-Fi access points enabling end-users to utilize the network. The created (emulated by VENUE) Wi-Fi network follows the 802.11a standard. The created (real) access points follow the 802.11n standard. VENUE provides mobility to the commodity equipment, as can be appreciated in the following section figures. Otherwise, making these scheduled replacements would not be possible.

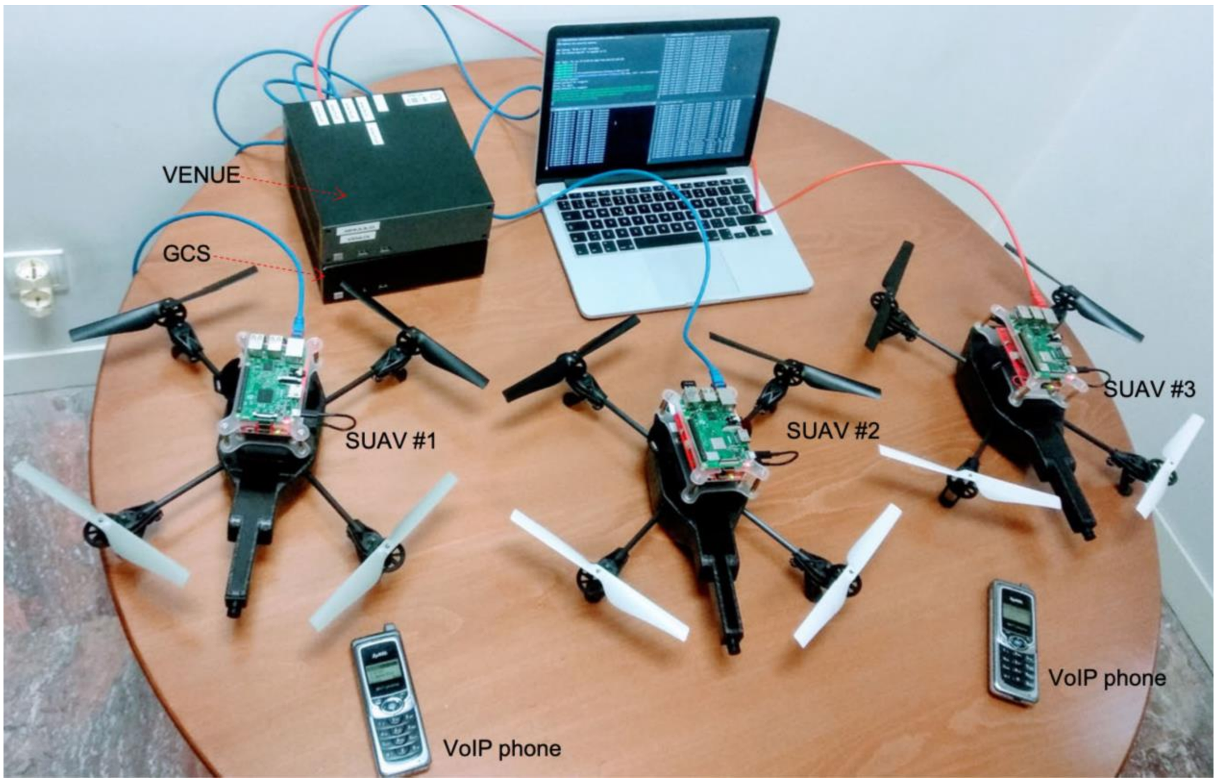


Figure 23. SUAVs, Single-board Computers, GCS, VoIP terminals and VENUE emulation platform

In order to create a network service, different VNFs have been used: (i) Two VNFs implement the router functionality, (ii) another VNF implements a router and also includes a DNS service and finally, (iii) another one implements a Voice-over-IP server based on Session Initiation Protocol server (Kamailio [39]), to allow the ground users to ‘‘register’’ wireless terminals in the VoIP server and maintain telephone conversations with other users. On the other hand, two APs have been configured (including a DHCP server) that allow users to connect to the network deployed by the SUAVs.

All these VNFs are instantiated and configured using OpenStack. The ground control station and the devices hosting the deployed VNFs use OLSR to enable the communications. Remarkably, the virtual networks are deployed using the Virtual eXtensible Local Area Networks (VXLAN) [95]. A complete VoIP call (including the signalling process to start the call) has been performed to test the whole network service. The ZyXEL Prestige 2000W terminals have been utilized to make the call. Besides, a video stream is also sent through the network using the VLC [96] tool. Finally, during the mission, SUAV 2 is replaced to force an additional topology change.

Network traffic is captured using the Wireshark tool to analyse this scenario. Captures are performed in SUAV 3 both in the ad hoc network interface (emulated network by the VENUE platform) and in the AP interface (real), in order to analyse all the traffic generated by the service.

Figure 24 shows the control traffic between the OpenStack controller and SUAV 3 compute node necessary to manage the computing, storage, and networking resources of the NFV platform. As a result of the replacement of SUAV 2, the figure reveals a drop in the received traffic while the computer node keeps on sending traffic to the controller steadily.

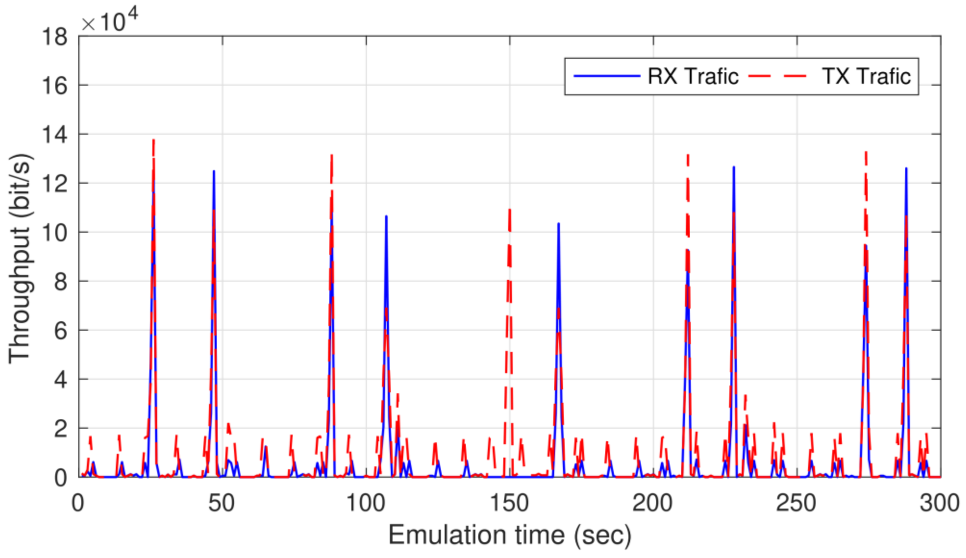


Figure 24. OpenStack control traffic in a compute node

Similarly, OLSR signalling traffic can also be appreciated in Figure 25. Likewise, during SUAV 2 replacement, there is a stop in the received traffic. OLSR average throughput is around 3 Kbit/s (which is negligible as compared to multimedia services).

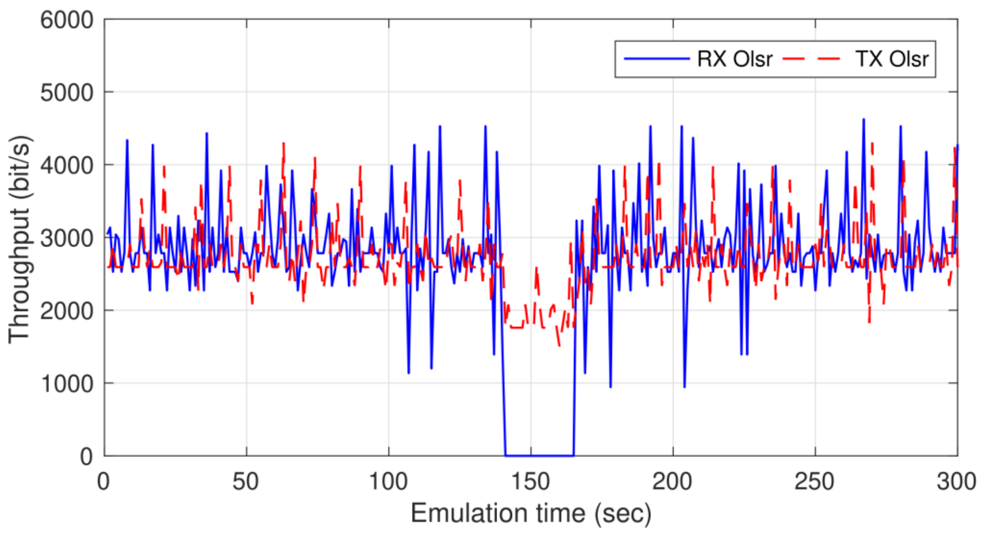


Figure 25. OLSR signalling traffic at the compute node

3 Conclusions

This document provided a detailed description of the different implementations developed during the 5G-RANGE project related to the Network Layer. Prior to this deliverable, in WP5, deliverable D5.1, defined the network-level architecture of 5G-RANGE, including the different protocol stacks for the different entities and then, deliverable D5.2 uptaded the architecture definitions with the most recent 3GPP Technical Specifications and provided the final version of the network-level architecture. Then, the present deliverable together with the previous ones, are closely related and complimentary, showing the evolution of the proposed architecture and a further analysis of the use cases proposed along with the project development.

This document concentrates in the network-level components that have been implemented to support the integration of the 5G architecture access network protocol structure into the 5G-RANGE Proof of Concept. These components used virtual machines orchestrated using a Management and Orchestration platform (MANO platform) and were deployed as Open Source. Also, this document describes other components related to diverse use cases other than the PoC, providing an analysis of viability and suggesting future developments.

This deliverable also describes the VENUE, a virtualized environment that has been developed to emulate network virtual environments, which has also been made publicly available under an open source license in a public directory, referenced in Section 2.3 of the deliverable.

References

[1] 3GPP, “System Architecture for the 5G System; Stage 2,” 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects, 3GPP Technical Specification 23.501, version 16.2.0, September 20189.

[2] 3GPP, “Procedures for the 5G System; Stage 2,” 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects, 3GPP Technical Specification 23.502, version 16.2.0., September 2019.

[3] 3GPP, “IP Multimedia Subsystem (IMS); Stage 2,” 3rd Generation Partnership Project; Aspects, Technical Specification Group Services and System, 3GPP Technical Specification 23.228, version 15.3.0, September 2018.

[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, SIP: Session Initiation Protocol, document RFC 3261, 2002.

[5] ETSI, “Network Functions Virtualisation (NFV); Architectural Framework,” ETSI GS NFV 002 V1.2.1, December 2014.

[6] OpenStack, “Open source software for creating private and public clouds,” [Online]. Available: https://www.openstack.org, Accessed on: Apr. 30, 2020.

[7] OpenVIM, “A light implementation of an NFV VIM contributed to the OSM project,” [Online]. Available: https://osm.etsi.org, Accessed on: Apr. 30, 2020.

[8] VMware, “VMware vCloud Director,” [Online]. Available: https://www.vmware.com/products/vcloud-director.html, Accessed on: Apr. 30, 2020.

[9] Amazon web services, “Amazon Elastic Compute Cloud (Amazon EC2),” [Online]. Available: https://aws.amazon.com/ec2, Accessed on: Apr. 30, 2020.

[10] Canonical, “Infrastructure for container projects,” [Online]. Available: https://linuxcontainers.org, Accessed on: Apr. 30, 2020.

[11] Docker, “Debug your app, not your environment,” [Online]. Available: https://www.docker.com, Accessed on: Apr. 30, 2020.

[12] Linux Foundation. “Production-Grade Container Orchestration,” [Online]. Available: https://kubernetes.io, Accessed on: Apr. 30, 2020.

[13] Rancher, “K3S: Lightweight Kubernetes,” [Online]. Available: https://k3s.io, Accessed on: Apr. 30, 2020.

[14] Eclipse Foundation. "Eclipse fog05: The End-to-End Compute, Storage and Networking Virtualization solution," [Online]. Available at: https://fog05.io, Accessed on: Apr. 30, 2020.

[15] G. Rigazzi, J. Kainulainen, C. Turyagyenda, A. Mourad and J. Ahn, "An Edge and Fog Computing Platform for Effective Deployment of 360 Video Applications," IEEE Wireless Communications and Networking Conference Workshop (WCNCW), Marrakech, Morocco, 2019, pp. 1-6.

[16] ETSI, “Open Source MANO (OSM),” [Online]. Available: https://osm.etsi.org, Accessed on: Apr. 30, 2020.

[17] The Linux foundation projects, “Open Network Automation Platform (ONAP),” [Online]. Available: https://www.onap.org, Accessed on: Apr. 30, 2020.

[18] Cloudify, “Open Source Network Automation and Network Orchestration framework,” [Online]. Available: https://cloudify.co, Accessed on: Apr. 30, 2020.

[19] OpenStack, “Tacker: OpenStack NFV Orchestration service,” [Online]. Available: http://docs.openstack.org/tacker, Accessed on: Apr. 30, 2020.

[20] Li, Bin, Zesong Fei, and Yan Zhang. "UAV communications for 5G and beyond: Recent advances and future trends," IEEE Internet of Things Journal 6.2 (2018): 2241-2263.

[21] M. Mohammad, et al. "Beyond 5G with UAVs: Foundations of a 3D wireless cellular network," IEEE Transactions on Wireless Communications 18.1 (2018): 357-372.

[22] Sharma, Vishal, et al. "Intelligent deployment of UAVs in 5G heterogeneous communication environment for improved coverage," Journal of Network and Computer Applications 85 (2017): 94-105.

[23] Huo, Yiming, et al. "Distributed and multilayer UAV networks for next-generation wireless communication and power transfer: A feasibility study," IEEE Internet of Things Journal 6.4 (2019): 7103-7115.

[24] A. Mahmood, B. Butler and B. Jennings, "Towards Efficient Network Resource Management in SDN-Based Heterogeneous Vehicular Networks," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, 2018, pp. 813-814.

[25] W. Zhuang, Q. Ye, F. Lyu, N. Cheng and J. Ren, "SDN/NFV-Empowered Future IoV With Enhanced Communication, Computing, and Caching," in Proceedings of the IEEE, vol. 108, no. 2, pp. 274-291, Feb. 2020.

[26] Z. Zhao, P. Cumino, A. Souza, D. Rosário, T. Braun, E. Cerqueira, M. Gerla, “Software-defined unmanned aerial vehicles networking for video dissemination services,” *Ad Hoc Networks*, 2019, 83, pp. 68–7.

[27] C. Rametta and G. Schembra, “Designing a Softwarized Network Deployed on a Fleet of Drones for Rural Zone Monitoring,” Future Internet, vol. 9, no. 1, p. 8, Mar. 2017.

[28] Nogales, B.; Sanchez-Aguero, V. ; Vidal, I.; Valera, F. Adaptable and Automated Small UAV Deployments via Virtualization. Sensors **2018**, 18, pp. 4116–4132.

[29] B. Nogales, I. Vidal, D. R. Lopez, J. Rodriguez, J. Garcia-Reinoso and A. Azcorra, "Design and Deployment of an Open Management and Orchestration Platform for Multi-Site NFV Experimentation," IEEE Communications Magazine, vol. 57, no. 1, pp. 20-27, January 2019.

[30] OSM. “OSM PoC 10, Automated Deployment of an IP Telephony Service on UAVs using OSM,” [Online]. Available: https://osm.etsi.org/wikipub/index.php/OSM_PoC_10_Automated_Deployment_of_an_IP_Telephony_Service_on_UAVs_using_OSM, Accessed on: Apr. 30, 2020.

[31] D5.2 Final version of the Network-level Architecture and Procedures. 5G-RANGE public deliverable.

[32] “5TONIC: an open research and innovation laboratory focusing on 5G technologies,” [Online]. Available: https://www.5tonic.org.

[33] B. Nogales et al., “Design and Deployment of an Open Management and Orchestration Platform for Multi-Site NFV Experimentation,” IEEE Communications Magazine, vol. 57, nº 1, pp. 20-27, January 2019.

[34] “Open Source MANO. ETSI-hosted project.,” [Online]. Available: https://osm.etsi.org.

[35] “OpenStack: Open source software for creating private and public clouds.,” [Online]. Available: https://www.openstack.org.

[36] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, “Generic routing encapsulation (gre),” RFC 2784, RFC Editor, March 2000. http://www. rfc- editor.org/rfc/rfc2784.txt.

[37] S. Kent and K. Seo, “Security architecture for the internet protocol,” RFC 4301, RFC Editor, December 2005. http://www.rfc-editor.org/rfc/ rfc4301.txt.

[38] Open IMS Core, Fraunhofer FOKUS research institute. Available online (last access on Jan. 2019): http://openimscore.sourceforge.net

[39] Kamailio, the Open Source SIP Server: https://www.kamailio.org/w/

[40] SIPp, a free Open Source test tool / traffic generator for the SIP protocol. Available online (last access on Jan. 2020): http://sipp.sourceforge.net

[41] B. Nogales et al., "Automated Deployment of an Internet Protocol Telephony Service on Unmanned Aerial Vehicles Using Network Functions Virtualization," Journal of Visualized Experiments (153), e60425, doi:10.3791/60425 (2019).

[42] I. Vidal *et al.*, "A Multi-site NFV Testbed for Experimentation with SUAV-based 5G Vertical Services," in IEEE Access, doi: 10.1109/ACCESS.2020.3001985.

[43] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[44] ITU-T. Recommendation G.114. General Recommendations on the transmission quality for an entire international telephone connection; One-way transmission time. International Telecommunication Union - Telecommunication Standardization Sector. (2003).

[45] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hannienen, and M. Pettissallo, "On the coverage of LPWANS: range evaluation and channel attenuation model for lora technology", ITS Telecommunications (ITST), 14th International Conference on, pp. 55-5, December, 2015.

[46] LoRa Alliance (2018), LoRaWAN 1.0.3 specification, Technical Report 1 [Online]. Available at: [https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf], last access: January 2020.

[47] LoRA Alliance (2017), LoRaWAN Regional Parameters v1.0.2rB, [https://lora-alliance.org/resource-hub/lorawanr-regional-parameters-v102rb], last access: January 2020.

[48] LoRa Alliance, The LoRaWAN Certified Device – Pycom FiPy: The world's only 5-network, MicroPython programmable development board, [https://lora-alliance.org/showcase/fipy], last access: January 2020.

[49] Pycom Team, Fipy Multipack, [https://pycom.io/wp-content/uploads/2018/08/fipySpecsheetAugust2017n2-1.pdf], last access: January 2020.

[50] Dragino, LG01-N Single Channel LoRa IoT Gateway Datasheet, [http://www.dragino.com/downloads/downloads/LoRa_Gateway/LG01N/Datasheet_LG01N_OLG01N.pdf], last access: January 2020.

[51] J. Petajajarvi, J. Janhunen, K. Mikhaylov, J. Linatti, "Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage", International Journal of Distributed Sensor Networks, Vol 13(3), 1-15, March, 2017.

[52] European Telecommunications Standards Institute. *Network Functions Virtualization (NFV)*; Architectural Framework; Research Report ETSI GS NFV 002 V1.2.1; European Telecommunications Standards Institute (ETSI): Sophia Antipolis, France, 2014.

[53] Sivakumar, A.; Tan, C.K.Y. UAV swarm coordination using cooperative control for establishing a wireless communications backbone. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 3-Volume 3, Toronto, ON, Canada, 10–14 May 2010; International Foundation for Autonomous Agents and Multiagent Systems; Waterloo, Canada; 2010; pp. 1157–1164.

[54] Grodi, R.; Rawat D. B. UAV-assisted broadband network for emergency and public safety communications. In Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP), Orlando, FL, USA, 14–16 December 2015; pp. 10–14.

[55] Jawhar, I.; Mohamed, N.; Al-Jaroodi, J.; Agrawal, D. P.; Zhang, S. Communication and networking of UAV-based systems: Classification and associated architectures. J. Netw. Comput. Appl. 2017, 84, 93–108.

[56] Zhou, Y.; Cheng, N.; Lu N.; Shen, X. S. Multi-UAV-Aided Networks: Aerial-Ground Cooperative Vehicular Networking Architecture. IEEE Veh. Technol. Mag. 2015, 10, 36–44.

[57] Vidal, I.; Bellavista, P.; Sanchez-Aguero, V.; Garcia-Reinoso, J.; Valera, F.; Nogales, B.; Azcorra, A. Enabling Multi-Mission Interoperable UAS using Data-Centric Communications. Sensors 2018, 18, 3421.

[58] Gonzalez, L. F.; Vidal, I.; Valera, F.; Sanchez-Agüero, V.; Nogales, B.; Lopez, D.R. NFV orchestration on intermittently available SUAV platforms: challenges and hurdles. In Proceedings of the Workshop in Mission-Oriented Wireless Sensor, UAV and Robot Networking (MiSARN), Paris, France, 29 April 2019.

[59] Tipantuña, C.; Hesselbach, X.; Sanchez-Aguero, V.; Valera, F.; Vidal, I.; Nogales, B. An NFV-Based Energy Scheduling Algorithm for a 5G Enabled Fleet of Programmable Unmanned Aerial Vehicles. Wireless Communications and Mobile Computing 2019.

[60] Erdelj, M.; Saif, O.; Natalizio, E.; Fantoni, I. UAVs that fly forever: Uninterrupted structural inspection through automatic UAV replacement. Ad Hoc Networks. 2019, 94.

[61] Shelby, Z.; Hartke, K.; Bormann, C. The Constrained Application Protocol (CoAP); RFC 7252; Internet Engineering Task Force (IETF); Delaware, USA; June 2014.

[62] OASIS. Information technology—Message Queuing Telemetry Transport (MQTT) v3.1.1; ISO/IEC 20922:2016; International Organization for Standardization (ISO); Vermont, USA; June 2016.

[63] Liu, J.; Singh, S. ATCP: TCP for mobile ad hoc networks. IEEE J. Sel. Areas Commun. 2001, 19, pp. 1300–1315.

[64] Larsen, E. TCP in MANETs—Challenges and Solutions; Norwegian Defence Research Establishment (FFI); Kjeller, Norway; 2012.

[65] Holland, G.; Vaidya, N. Analysis of TCP Performance over Mobile Ad Hoc Networks. Wirel. Netw. 2002, 8, 275–288.

[66] Devaraj, S.A.; Anita, R.H.V.; Christa, J.J. Comparative analysis of random based mobility models using TCP variant in MANETs. In proceedings of the 2014 International Conference on Communication and Network Technologies, Sivakasi, India, 18–19 December 2014; pp. 324–329.

[67] Fu, Z.; Meng, X.; Lu, S. How bad TCP can perform in mobile ad hoc networks. In Proceedings of the IEEE International Symposium on Computers and Communications (ISCC'02), Taormia-Giardini Naxos, Italy, 1–4 July 2002; pp. 298–303

[68] Johnson, D.; Hu, Y.; Maltz, D. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4; RFC 4728; Internet Engineering Task Force (IETF); Delaware, USA; February 2007.

[69] Perkins, C.; Belding-Royer E.; Das, S. Ad hoc On-Demand Distance Vector (AODV) Routing; RFC 3561; Internet Engineering Task Force (IETF); Delaware, USA; July 2003.

[70] Clausen, T.; Jacquet, P. Optimized Link State Routing Protocol (OLSR); RFC 3626; October 2003.

[71] Iyengar, J.; Thomson, M. QUIC: A UDP-Based Multiplexed and Secure Transport; draft-ietf-quic-transport-24; Internet Engineering Task Force; Delaware, USA; November 2019.

[72] Belshe, M.; Peon, R.; Thomson, R. Hypertext Transfer Protocol Version 2 (HTTP/2); RFC 7540; Internet Engineering Task Force (IETF); Delaware, USA; May 2015.

[73] Dierks, T.; Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2; RFC 5246;Internet Engineering Task Force (IETF); Delaware, USA; August 2008.

[74] Luis F. Gonzalez, Ivan Vidal, Francisco Valera, Borja Nogales, Victor Sanchez-Aguero, Diego R. Lopez. Transport Layer Limitations for NFV Orchestration in Resource Constrained Aerial Networks. MDPI Sensors. December 2019.

[75] A. Mukherjee, V. Keshary, K. Pandya, N. Dey, and S. C. Satapathy, ‘‘Flying ad hoc networks: A comprehensive survey,’’ in Information and Decision Sciences. Singapore: Springer, 2018, pp. 569–580.

[76] HOWTO Use Linux Containers to Set Up Virtual Networks. Accessed Jan. 31, 2020. [Online]. Available: https://www.nsnam.org/wiki/HOWTO_Use_Linux_Containers_to_set_up_virtual_networks

[77] H. Nawaz, H. M. Ali, and S. U. R. Massan, ‘‘A study of mobility models for UAV communication networks,’’ in Proc. 3C Tecnologia. Rochester, MN, USA: Mayo, May 2019, pp. 276–297.

[78] V. Sanchez-Aguero. VENUE, Accessed: Jan. 31, 2020. [Online]. Available: https://github.com/vsaguero/VENUE

[79] O. S. Oubbati, M. Atiquzzaman, P. Lorenz, M. H. Tareque, and M. S. Hossain, ‘‘Routing in flying ad hoc networks: Survey, constraints, and future challenge perspectives,’’ IEEE Access, vol. 7, pp. 81057–81105, 2019.

[80] V. Sanchez-Aguero, F. Valera, B. Nogales, L. F. Gonzalez and I. Vidal. VENUE: Virtualized Environment for multi-UAV network emulation. IEEE Access. October 2019.

[81] M. O. Kalinin, V. Krundyshev, and P. Semianov, ‘‘Architectures for building secure vehicular networks based on SDN technology,’’ Autom. Control Comput. Sci., vol. 51, no. 8, pp. 907–914, Dec. 2017.

[82] R. Kumar, M. A. Sayeed, V. Sharma, and I. You, ‘‘An SDN-based secure mobility model for UAV-ground communications,’’ in Proc. Int. Symp. Mobile Internet Secur. Springer, 2017, pp. 169–179.

[83] Z. Kaleem and M. H. Rehmani, ‘‘Amateur drone monitoring: State-of-the-art architectures, key enabling technologies, and future research directions,’’ IEEE Wireless Commun., vol. 25, no. 2, pp. 150–159, Apr. 2018.

[84] Ryu SDN Framework. Accessed: Jan. 31, 2020. [Online]. Available: https://osrg.github.io/ryu/

[85] Open vSwitch. Accessed: Jan. 31, 2020. [Online]. Available: https://www.openvswitch.org/

[86] GUEANT, IPERF—The TCP, UDP and SCTP Network Bandwidth 1018 Measurement Tool. Accessed: Jan. 31, 2020. [Online]. Available: https://osrg.github.io/ryu/

[87] TCPDUMP/LIBPCAP Public Repository. Accessed: Sep. 19, 2019. [Online]. Available: https://www.tcpdump.org/

[88] U. Lamping and E. Warnicke, ‘‘Wireshark user’s guide,’’ in Interface, vol. 4, no. 6. 2004.

[89] D2.1 Application and requirements report. 5G-RANGE public deliverable. Available at: [http://5g-range.eu/wp-content/uploads/2018/04/5G-Range_D2.1_Application_Requirement_Report_v1.pdf] last access December 2018.

[90] D2.2 Architecture, system and interface definitions of a 5G for Remote Area network. 5G-RANGE public deliverable. Available at: [http://5g-range.eu/wp-content/uploads/2018/04/5G-Range_D2.2-Architecture_5G_RemoteArea.pdf] last access December 2018.

[91] B. Nogales, V. Sanchez-Aguero, I. Vidal, F. Valera, and J. Garcia-Reinoso. 2018. A NFV system to support configurable and automated multi-UAV service deployments. In Proceedings of the 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet'18). ACM, New York, NY, USA, 39-44.

[92] Linux Containers—LXD—Introduction. Accessed: Jan. 31, 2020. [Online]. Available: https://linuxcontainers.org/lxd/

[93] Tap Net Device âAŤ NS-3 NS-3.14 Documentation. Accessed Jan. 31, 2020. [Online]. Available: https://www.nsnam.org/docs/release/3. 14/models/html/tap.html

[94] RPI. Accessed Jan. 31, 2020. [Online]. Available: https://www. raspberrypi.org/products/raspberry-pi-3-model-b/

[95] M. Mahalingam, D. G. 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, document RFC 7348, 2014, pp. 1–22.

[96] Official Download of VLC Media Player, the Best Open Source Player 1059-VideoLAN. Accessed Jan. 31, 2020. [Online]. Available: https://www.videolan.org/vlc/index.es.html

[97] Trafic. "A traffic mix generator based on iperf3," [Online] https://github.com/mami-project/trafic, Accessed on: Apr. 30, 2020.

[98] H. Niu, N. Gonzalez-Prelcic, and R. W. Heath, Jr., ''A UAV-based traffic monitoring system—Invited paper,'' in Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring), Jun. 2018, pp. 1–5.