Video Article

Automated Deployment of an Internet Protocol Telephony Service on Unmanned Aerial Vehicles Using Network Functions Virtualization

Borja Nogales*[1], Ivan Vidal*[1], Victor Sanchez-Aguero*[1,2], Francisco Valera*[1], Luis F. Gonzalez*[1], Arturo Azcorra*[1,2]

[1]Department of Telematic Engineering, Universidad Carlos III de Madrid

[2]IMDEA Networks Institute

*These authors contributed equally

Correspondence to: Ivan Vidal at ividal@it.uc3m.es

URL: https://www.jove.com/video/60425
DOI: doi:10.3791/60425

Keywords: Engineering, Issue 153, unmanned aerial vehicles (UAVs), network functions virtualization (NFV), management and orchestration (MANO), cloud computing platform, virtual network function (VNF), IP telephony service, open source, 5G

Date Published: 11/26/2019

Citation: Nogales, B., Vidal, I., Sanchez-Aguero, V., Valera, F., Gonzalez, L.F., Azcorra, A. Automated Deployment of an Internet Protocol Telephony Service on Unmanned Aerial Vehicles Using Network Functions Virtualization. *J. Vis. Exp.* (153), e60425, doi:10.3791/60425 (2019).

Abstract

The Network Function Virtualization (NFV) paradigm is one of the key enabling technologies in the development of the 5th generation of mobile networks. This technology aims to lessen the dependence on hardware in the provision of network functions and services by using virtualization techniques that allow the softwarization of those functionalities over an abstraction layer. In this context, there is increasing interest in exploring the potential of unmanned aerial vehicles (UAVs) to offer a flexible platform capable of enabling cost-effective NFV operations over delimited geographic areas.

To demonstrate the practical feasibility of utilizing NFV technologies in UAV platforms, a protocol is presented to set up a functional NFV environment based on open source technologies, in which a set of small UAVs supply the computational resources that support the deployment of moderately complex network services. Then, the protocol details the different steps needed to support the automated deployment of an internet protocol (IP) telephony service over a network of interconnected UAVs, leveraging the capacities of the configured NFV environment. Experimentation results demonstrate the proper operation of the service after its deployment. Although the protocol focuses on a specific type of network service (i.e., IP telephony), the described steps may serve as a general guide to deploy other type of network services. On the other hand, the protocol description considers concrete equipment and software to set up the NFV environment (e.g., specific single board computers and open source software). The utilization of other hardware and software platforms may be feasible, although the specific configuration aspect of the NFV environment and the service deployment may present variations with respect to those described in the protocol.

Video Link

The video component of this article can be found at https://www.jove.com/video/60425/

Introduction

One of the most coveted goals within the new era of the mobile communications (most commonly known as the 5th mobile generation or 5G) is to be able to provide robust information technology services in situations where the primary telecommunications infrastructure may not be available (e.g., due to an emergency). In this context, the UAVs are receiving increasing attention from the research community due to their inherent versatility. There are numerous works that use these devices as a cornerstone for the provision of a large variety of services. For instance, the literature has analyzed the capacity of these devices to build an aerial communication infrastructure to accommodate multimedia services[1,2,3]. Furthermore, prior research has shown how the cooperation among several UAVs can extend the functionality of different communication services such as surveillance[4], collaborative search and rescue[5,6,7,8], or agribusiness[9].

On the other hand, the NFV technology has acquired great significance within the telecom operators as one of the 5G key enablers. NFV represents a paradigmatic change regarding the telecommunications infrastructure by alleviating the current dependency of network appliances on specialized hardware through the softwarization of the network functionalities. This enables a flexible and agile deployment of new types of communication services. To this purpose, the European Telecommunications Standards Institute (ETSI) formed a specification group to define the NFV architectural framework[10]. Additionally, the ETSI currently hosts the Open Source Mano (OSM) group[11], which is in charge of developing an NFV Management and Orchestration (MANO) software stack aligned with the definition of the ETSI NFV architectural framework.

Given all the aforementioned considerations, the synergic convergence between UAVs and NFV technologies is currently being studied in the development of novel network applications and services. This is illustrated by several research works in the literature that point out the advantages of these types of systems[14,15,16], identify the challenges of this convergence and its missing aspects, highlight future research lines on this topic[17], and present pioneer solutions based on open source technologies.

In particular, the integration of NFV technologies into the UAV arena enables the rapid and flexible deployment of network services and applications over delimited geographic areas (e.g., an IP telephony service). Following this approach, a number of UAVs can be deployed over a specific location, transporting compute platforms as payload (e.g., small-size single board computers). These compute platforms would provide a programmable network infrastructure (i.e., an NFV infrastructure) over the deployment area, supporting the instantiation of network services and applications under the control of a MANO platform.

Notwithstanding the benefits, the realization of this view presents a set of fundamental challenges that needs to be carefully addressed, such as the appropriate integration of these compute platforms as an NFV infrastructure, using an existing NFV software stack, so that an NFV orchestration service can deploy virtual functions on the UAVs; the constraints in terms of the computational resources provided by the compute platforms, as the UAVs transporting them may typically present limitations in terms of size, weight, and computing capacity of payload equipment; the proper placement of the virtual functions onto UAVs (i.e., selecting the best UAV candidate to deploy a particular virtual function); the maintenance of the control communications with the UAVs in order to manage the lifecycle of the VNFs in spite of the potentially intermittent availability of network communications with them (e.g., caused by mobility and battery constraints); the limited operation time of the UAVs due to their battery consumption; and the migration of the virtual functions when a UAV needs to be replaced due to its battery exhaustion. These benefits and challenges are detailed in previous work[18,19] that includes the design of an NFV system capable of supporting the automated deployment of network functions and services on UAV platforms, as well as the validation of the practical feasibility of this design.

In this context, this paper focuses on describing a protocol to enable the automated deployment of moderately complex network services over a network of UAVs using the NFV standards and open source technologies. To illustrate the different steps of the protocol, a re-elaboration of an experiment presented in Nogales et al.[19] is presented, consisting of the deployment of an IP telephony service. To aid the reproducibility of this work, real flight is considered as optional in the presented procedure, and performance results are obtained with the UAV devices on the ground. Interested readers should be able to replicate and validate the execution of the protocol, even in a controlled laboratory environment.

Figure 1 illustrates the network service designed for this procedure. This network service is built as a composition of specific softwarization units (categorized within the NFV paradigm as Virtual Network Functions, or VNFs) and provides the functionality of an IP telephony service to users in the vicinity of the UAVs. The VNF composing the service are defined as follows:

- Access Point VNF (AP-VNF): This VNF provides a Wi-Fi access point to end-user equipment (i.e., IP phones in this experiment).
- IP telephony server VNF (IP-telephony-server-VNF): It is responsible for managing the call signaling messages that are exchanged between IP phones to establish and terminate a voice call.
- Domain Name System VNF (DNS-VNF): This VNF provides a name resolution service, which is typically needed in IP telephony services.
- Access router VNF (AR-VNF): provides network routing functionalities, supporting the exchange of traffic (i.e., call signaling in this experiment) between the IP phones and the telecommunication operator domain.
- Core router VNF (CR-VNF): provides network routing functionalities in the telecommunication operator domain, offering access to operator-specific services (i.e., the IP telephony server) and external data networks.

Moreover, **Figure 1** presents the physical devices used for the experiment, how they are interconnected, and the specific allocation of VNFs to devices.

Protocol

1. Prior requisites for the experiment

1. Install the Management and Orchestration (MANO) software stack provided by the Open Source MANO (OSM) project. Specifically, this experiment uses OSM Release FOUR[20], which can be executed in a single server computer or in a Virtual Machine (VM) fulfilling the requirements specified by the OSM community: Ubuntu 16.04 as the operating system (64-bit variant image), two central processing units (CPUs), 8 GB random access memory (RAM), a 40 GB storage disk, and a single network interface with Internet access. The procedure to install OSM Release FOUR along with its technical details are available in the online documentation provided by the OSM community[21].
2. Set up a cloud computing platform, providing the functions of a virtual infrastructure manager (VIM) compliant with OSM Release FOUR. For this experiment, OpenStack release Ocata[22] is used, running in a VM with Ubuntu 16.04 as the operating system, four CPUs, 16 GB RAM, and 200 GB storage disk. In the experiment, the VIM manages an NFV infrastructure (NFVI) integrated by two high-profile server computers, each with Ubuntu 16.04 as the operating system, eight CPUs, 128 GB RAM, and 4 TB storage disk). All the information on how to set up a cloud computing platform is included in the installation guide included in the OpenStack documentation[23]. This cloud platform is referred to as the core cloud platform.
3. Set up an additional cloud computing platform for the UAVs is referred to as the UAVs cloud platform.
    1. Ensure that this platform features a VIM based on OpenStack release Ocata. In this case, the resources used by the VIM installation are Ubuntu 16.04 as operating system, two CPUs, 6 GB RAM, 100 GB storage disk, and an external Wi-Fi USB adapter.
    2. The NFVI integrated in this cloud platform consists of a single fixed compute server (Ubuntu 16.04 as operating system, eight CPUs, 8 GB RAM, 128 GB storage disk, and an external Wi-Fi USB adapter) and three single board computers (SBCs). The latter provide a hardware platform that can easily be onboarded on a UAV. See Section 3 for the procedure to setup a UAV cloud platform with these devices as compute nodes.

4. Equip each SBC with a battery-power supply hardware attached on top (HAT) to ensure the operation of these units even when they are in motion, being carried by a UAV.
    NOTE**:** Step 1.5 is optional because the provision of the network service in the experiment does not depend on having UAVs. In addition, the SBCs are carried as the payload of the UAVs and no other additional connections (e.g., Ethernet or USB) are needed, because the network communications required for the proper operation of the IP telephony service are provided by the SBCs, through their Wi-Fi adapters, and the power supply is provided by the power-supply HAT mentioned in step 1.4.

5. Attach each SBC as the payload of a UAV through a fixing accessory. In this experiment, three commercial UAVs were chosen to transport the compute units offered by the SBCs.
6. Select two wireless voice-over-IP (VoIP) phones that support the IEEE 802.11b wireless communications standard; this model provides wireless communications via Wi-Fi. As an alternative, the voice call could be executed using softphone applications such as Linphone[24] or Jitsi[25].
7. As an experimental requirement, make sure of the availability of: a) layer-3 communications between the OSM software stack and each of the VIMs to enable the orchestrated deployment of the network service developed for this experiment, b) layer-3 communications between the OSM and the VNFs at each cloud platform to support VNF configuration procedures, and c) the layer-3 communications among the VNFs running at every VIM to enable the proper functioning of the network service.
8. All the content needed to carry out the experiment is provided in the public experiment repository http://vm-images.netcom.it.uc3m.es/JoVE/.

2. Validating the functionality of the softwarization units via emulation

NOTE: To prove the appropriate operation of the network service of the experiment (see **Figure 1**) under realistic deployment conditions, a purpose-specific emulation platform based on Linux containers[26] and ns-3[27] was used. This platform allows emulating multi-hop aerial links and defining the characteristics of those links (e.g., length of the wireless communication links, pattern of data packet losses, the radio technology used in the wireless communications, etc.). Thus, this section of the protocol describes the steps to be followed to verify the appropriate operation of the IP telephony service under realistic wireless communication link conditions through the emulation platform.

1. Download the emulation platform from the experiment repository. The platform is available as a virtual machine, named "uav-nfv-jove-experiment.qcow", compliant with the KVM virtualization technology[28]. This machine contains a precreated template that emulates the network service and the multi-UAV scenario presented in **Figure 1** and a user with administrator privileges capable of executing that template.
   NOTE: By default, the following steps are automatically executed when the emulation platform virtual machine is started: a) the virtual environment is configured to enable the network emulation (i.e., network interfaces, Linux bridges[29]); b) the Linux Containers representing the different physical components of the testbed (i.e., the SBCs and the fixed compute server for the UAV cloud platform, and the compute server for the core cloud platform) are created; and c) the functions provided by the different VNFs of the IP telephony service (i.e., access points, routers, DNS serve, and IP telephony server) are deployed as Linux Containers over their corresponding emulated SBCs and compute servers.
2. Before the validation process, set up an emulated multi-hop aerial network using the ns-3 simulator, in order to enable the connectivity between the different network participants. This procedure will emulate the realistic wireless communications that take place in the scenario depicted in **Figure 1** (i.e., the Wi-Fi ad-hoc network, which enables the data exchange among the nodes of the UAV cloud platform and the wireless networks offered by the two Wi-Fi access points provided in the service).
   1. Create the multi-hop aerial network. For this purpose, execute the **multi-hop-aerial-net.sh** script (available within the emulation platform machine) using the following command: **sudo sh /home/jovevm/scripts/multi-hop-aerial-net.sh > multi-hop-aerial-net-trace.log 2>&1 &**. This command portrays the simulation trace in the specified log-file to enable debugging in the case of errors.
   2. Check if the network has been successfully created. To this end, verify if the Linux Containers "IP-phone-a" and "IP-phone-b" (illustrated in **Figure 1** as the end-user equipment that connects to an AP-VNF) have obtained an IP address through the DHCP service, which is only accessible through the multi-hop aerial network. The status of the Linux container executed within the emulation machine, as well as their IP addresses, can be checked using the command **lxc list**.
3. Verify the capacity of the emulated network service to process the signaling messages needed to set up the IP telephony call. For this purpose, both the "IP-phone-a" and "IP-phone-b" Linux containers have installed the "SIPp" tool[30]. "SIPp" provides the functionality to emulate an IP phone creating the mentioned signaling messages, send them to an IP telephony server, and process the response to verify the correct operation of the latter.
   1. Execute the script **test-signaling.sh** in both containers, which runs the "SIPp" tool to generate and send signaling messages to the IP-telephony-server-VNF.
   2. Check the scenario screen provided by execution of the previous step. The reception of "200" response shows the appropriate functioning of the IP-telephony-server-VNF.
4. Validate that the network service can process the data traffic that is generated during an IP telephony call. To do so, the flow scheduling "Trafic" tool[31] is installed in the "IP-phone-a" and "IP-phone-b" Linux containers.
   1. Execute the following command to start the server agent of Trafic: **lxc exec IP-phone-b sh called-party.sh**.
   2. Then, execute the following command to start the client agent of Trafic and get the network statistics: **lxc exec IP-phone-a sh caller.sh**. The data traffic emulating a voice call is terminated after 60 s. The script displays a confirmation message and the most significant performance metrics concerning the voice traffic.
   3. Check the obtained metrics and verify that the IP telephony service can effectively support an interactive voice conversation. To do so, see the information included in the section on representative results.

3. UAVs cloud platform construction

1. Select the model of SBC that can provide the virtualization substrate to execute lightweight VNFs. The technical specifications of the SBC devices utilized during the experiment are: four CPUs, 1 GB RAM, and a 32 GB storage disk. Additionally, each SBC has three network interfaces: an Ethernet interface, an integrated Wi-Fi interface, and an external Wi-Fi USB adapter.
2. Prepare the SBCs to be subsequently integrated into the UAVs cloud platform.
   1. Install Ubuntu Mate[32] 16.04.6 as the operating system, given that the OpenStack installation packages are included in this Linux distribution.

2. Install and configure the required packages as indicated in the OpenStack documentation[33] to allow the SBCs to act as the compute nodes of the UAV cloud platform. Following the previous guide, enable the utilization of Linux containers in the configuration of the OpenStack packages. Container virtualization is used due to the resource constraints of the devices that can typically be onboarded on small-sized UAVs.
3. In the SBC, download and execute the script **rpi-networking-configuration.sh**, available within the experiment repository. This script enables the wireless communications of the SBCs, as well as the required configuration to allow the creation of virtual networks attached to the wireless interfaces.
4. Download and execute the script **VIM-networking-configuration.sh**, available within the experiment repository, in the host running the UAV cloud platform VIM. This script oversees setting up the wireless communications of the VIM to enable the information exchange with the SBCs.
   NOTE: Once the networking is well configured and the VIM has connectivity with the SBCs, the VIM automatically integrates them into the UAV cloud platform as computational units capable of executing VNFs

3. Create an OpenStack availability zone for each of the SBCs. This will allow deploying each of the lightweight VNFs of the experiment in an appropriate UAV unit. To do so, log in to the web graphical user interface provided by the VIM with the administrator credentials, create the availability zones in the **Administrator** > **System** > **Host Aggregates** tab, and edit each availability zone to add the appropriate host (i.e., each SBC integrated into the UAV cloud platform).
4. Verify the correct setup of the UAV cloud platform. To do so, access the **Administrator** > **System** > **System Information** tab with the same login as in the previous step, and click in the **Computing Service and Network Agents** section to check that the status of the displayed items is "Alive" and "UP".

4. Configuring the experiment

1. Download the VNF images that implement the different components of the IP telephony service: the AP-VNF, the DNS-VNF, IP-telephony-server-VNF, the AR-VNF, and the CR-VNF. These images can be downloaded from the experiment repository.
2. Upload the VNF images to their correspondent VIM (i.e., the AP-VNF and the DNS-VNF to the UAV cloud platform VIM) and the VoIP-VNF to the core cloud platform VIM. To do so, log in into the web graphical user interface provided by each VIM with the administrator credentials, click on the **Create Image** button of the **Administrator** > **System** > **Images** tab, and create an image using the displayed form and selecting the appropriate image. This process is done at the corresponding VIM for each image that has been downloaded in the prior step.
3. Download the VNF descriptors (VNFDs) of the experiment from the experiment repository. These descriptors provide the templates that describe the operational requirements of a VNF, as well as the placement policies that indicate the availability zone in charge of hosting the VNF itself. More information on NFV descriptors can be found in the information model of OSM[34].
4. Upload the VNFDs. Use a web browser to access the OSM graphical user interface, and sign in with the administrator credentials. Then, drag and drop the VNFDs into the **VNF Packages** tab.
5. Download the network services descriptor (NSD) from the experiment repository. This descriptor is a template that specifies the VNFs comprising the service, as well as how those VNFs are interconnected.
6. Upload the NSD. Drag and drop the NSD into the **NS Packages** tab of the OSM graphical user interface.
7. Using the graphical user interface of OSM, add a VIM account for the UAV cloud platform VIM and for the core cloud platform VIM. To do this, access the **VIM accounts** tab with the administrator credentials, click on the button **+ New VIM** and complete the displayed form with the requested information. Repeat this action for both VIMs.

5. Executing the experiment

1. Deploy the network service. From the **NS packages** tab of the OSM graphical user interface, click on the **Instantiate NS** button of the NSD uploaded in step 4.6. Then, fill the displayed form, indicating the VIM that will be used to deploy each VNF composing the NS. In addition, the OSM is responsible for processing the placement policies indicated in the VNFDs to specify the VIM which availability zone (i.e., a compute unit in our testbed) is in charge of hosting each VNF. For this experiment, the VNFs are placed in the compute units as illustrated in **Figure 1**.
   NOTE: As an alternative method, the OSM provides a command line interface that enables direct user interaction. A user reproducing this experiment can use this command line interface, instead of the graphical interface, to execute the different steps defined in this protocol, particularly those steps related with onboarding a VNF or an NS descriptor, as well as deploying a network service.
2. Wait until the OSM graphical user interface indicates the success on the network service deployment.
   NOTE: The operation of the network service is totally independent from the flight of the UAVs: The IP telephony service can be provided when the UAVs are flying or saving battery consumption perched on a surface. Thus, step 5.3 is optional.
3. Take off the UAVs. Log in to the mobile application and control the flight of each UAV to stably maintain it in an intermediate height and avoid the turbulence caused by the rotation of the motors close to a surface.
4. Prepare each of the IP phones to carry out the call.
   1. Connect a wireless VoIP phone to each of the access points offered by the network service. For this purpose, specify the SSID (*Service Set Identifier*) in the **Menu** > **Wireless** > **SSID** tab and choose the **Infrastructure** mode in the **Menu** > **Wireless** > **Network Mode** section. Finally, select the networking configuration through the *Dynamic Host Configuration Protocol* (DHCP) in the **Menu** > **Net Settings** > **Network Mode** tab.
   2. Configure the *Session Initiation Protocol* (SIP) parameters to enable the appropriate exchange of signaling messages with the IP telephony server. In this context, access to the **Menu** > **SIP Settings** tab and specify the host name of the IP telephony server VNF ("dronesVoIP.net") in the **Registrar** > **Registrar IP** and **Proxy Server** > **Proxy IP** tabs. Furthermore, create a user account introducing the name of the user (e.g., caller-A) in the **User Account** > **Phone Number** and **User Account** > **Username** sections.
   3. Create an entry in the phonebook of one of the IP phones providing the information of the user that is to be called. To do so, select the **Menu** > **Phonebook** > **Add Entry** tab, and fill in the requested parameters that appear in the display as follows: Display name = caller-B; User Info = caller-B; Host IP = dronesVoIP.net; Port = 5060. Finally, select the "Proxy" option versus the P2P (peer-to-peer).

5. Start the call to the other party. To do so, select the called party using the **Menu** > **Phonebook** > **Search** option of the IP phone. Then, press the call button. Once the other IP phone starts ringing, accept the incoming call with the call button.

6. Procedure to gather experimental results

1. Connect a commodity laptop to one of the wireless APs and run the **ping** command line tool to the IP address of the phone connected to the other AP during 180 s. The IP address can be checked in the **Menu** > **Information** > **IP address** option of the IP phone once the connection is established with the AP. Save the Round-Trip Time (RTT) measurements, redirecting the output provided by the **ping** tool into a file.
2. Execute the **tcpdump** command line tool in one of the running AP VNFs to capture the traffic exchanged during the IP call. Save this traffic into a file enabling the writing flag of the command line tool at the execution time and specifying the name of the file.
3. Perform a new IP telephony call. Maintain the call for the desired time period (e.g., 1 min). Then, terminate the call, pressing the hang up button of one of the IP phones.
4. Keep the files generated by the **tcpdump** and **ping** tools for further processing. See Representative Results.

Representative Results

Based on the data obtained during the execution of the experiment, in which a real VoIP call is executed and following the steps indicated by the protocol to gather this information, **Figure 2** depicts the cumulative distribution function of the end-to-end delay measured between two end-user equipment items (i.e., a commodity laptop and an IP telephone). This user equipment represents two devices that are interconnected through the AP VNFs of the deployed network service. 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 3 illustrates the exchange of DNS and SIP signaling messages. These messages correspond to the registration of one of the users in the IP telephony server (i.e., the user whose IP phone is connected to the AP VNF where the "**tcpdump**" tool is running) and to the establishment of the voice call.

Finally, **Figure 4** and **Figure 5** show the data traffic captured during the call. In particular, the first one represents the constant stream of voice packets transmitted and received by one of the wireless phones during the call, whereas the latter illustrates the jitter in the forward direction with an average value lower than 1 ms.

The results that were obtained in the experiment for the delay figures (end-to-end delay and jitter) satisfy the recommendations specified by the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T)[35]. Accordingly, the voice call progressed with no glitches and good sound quality. This experiment validated the practical feasibility of using NFV technologies and UAVs to deploy a functional IP telephony service.

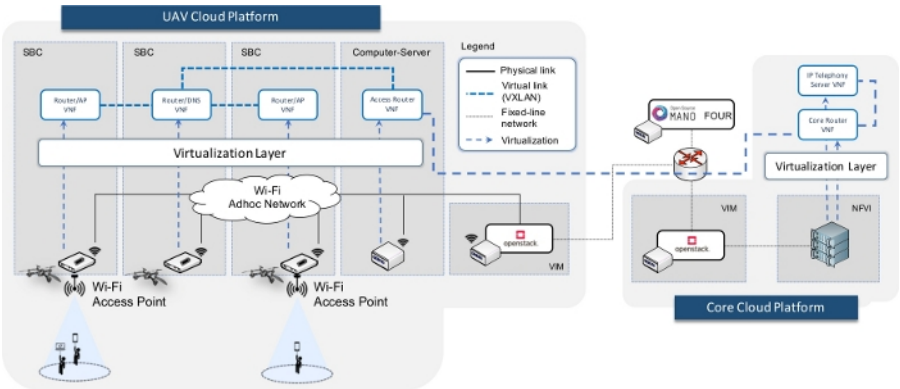

Figure 1: Overview of the network service, depicting the VNFs, the entities in which they are executed, and the virtual networks needed for the provision of the IP telephony service. Please click here to view a larger version of this figure.

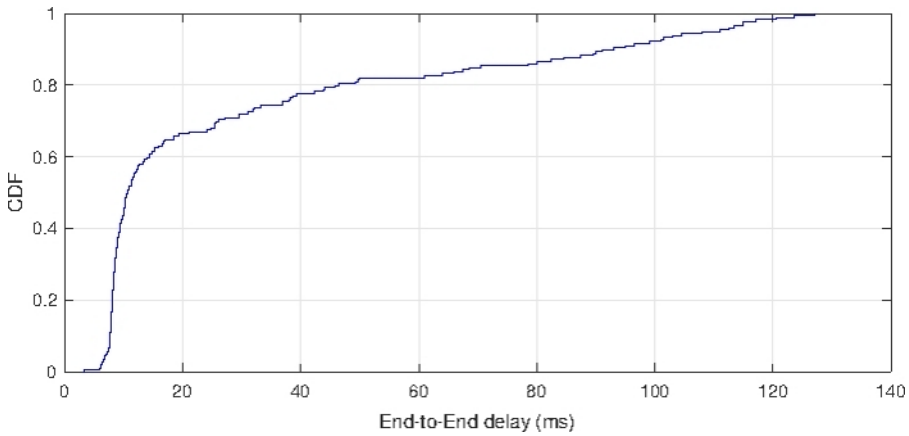

Figure 2: End-to-end delay. Representation of the end-to-end delay offered to the end-user equipment connected to the AP VNFs. For this purpose, the cumulative distribution function of the end-to-end delay has been computed from the measured RTT samples obtained with the "**ping**" command line tool. Please click here to view a larger version of this figure.

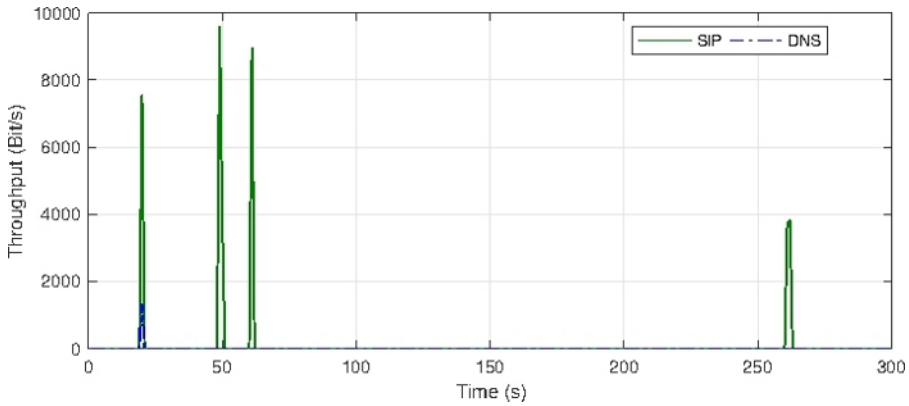

Figure 3: User registration and call signaling messages. Illustration of the signaling traffic (DNS and SIP) exchanged to register a user in the IP telephony server and to create and terminate the multimedia session that supports the execution of the voice call. Please click here to view a larger version of this figure.

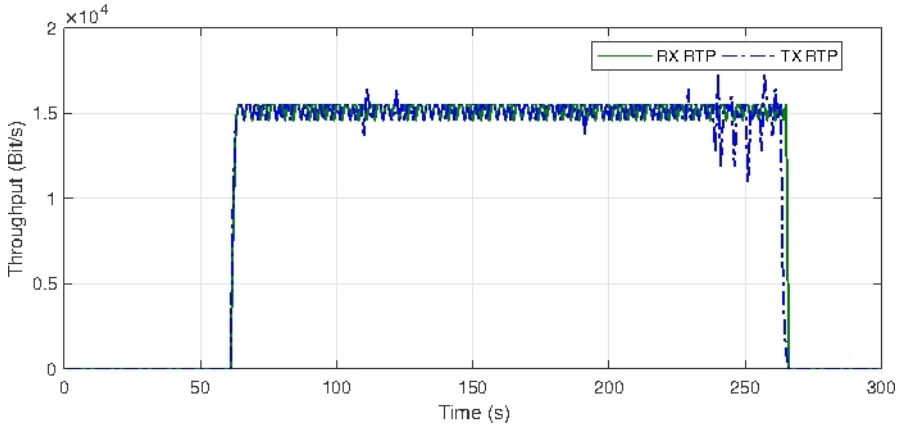

Figure 4: Stream of voice packets. Representation of the voice traffic exchanged during the call, measured at one of the AP VNFs. (Abbreviations: RX = receive, RX = transmit, RTP = real-time transport protocol). Please click here to view a larger version of this figure.

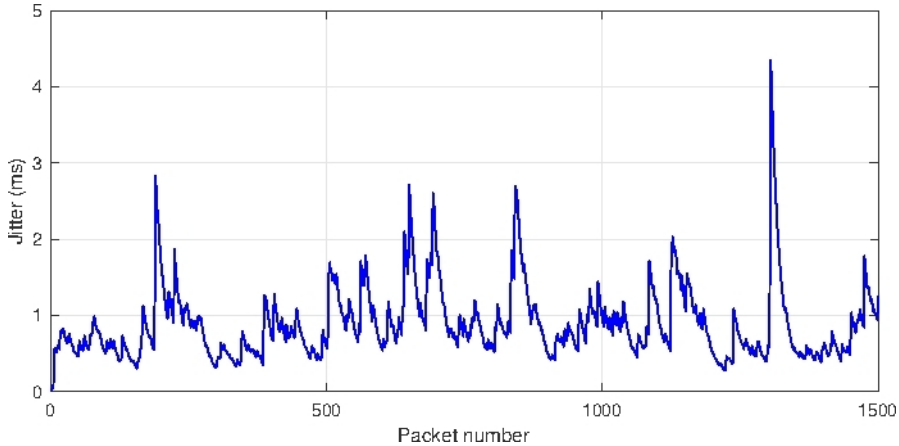

Figure 5: Evolution of the network jitter during the call. Representation of the jitter experienced by the transmitted voice packets in the forward direction from one phone to the other. Please click here to view a larger version of this figure.

Discussion

One of the most important aspects of this experiment is the use of virtualization technologies and NFV standards with UAV platforms. NFV presents a new paradigm aiming to decouple the hardware dependency of the network functionalities, thus enabling the provision of these functionalities through softwarization. Accordingly, the experiment does not depend on the use of the hardware equipment specified in the protocol. Alternatively, different models of single board computers can be selected, as long as they are in line with the dimensions and the transport capacity of the UAVs and they support Linux containers.

Notwithstanding this flexibility in terms of hardware selection, all the content provided for the reproducibility of the experiment is oriented towards the use of open source technologies. In this context, the configuration aspects and the software tools are conditioned to the use of Linux as the operating system.

On the other hand, the experiment considers the interoperation of two different computational platforms (i.e., the UAV cloud platform and the core cloud platform) to provide a moderately complex network service. However, this is not strictly needed, and the protocol could be followed to support scenarios in which only the UAV cloud platform is involved.

In addition, the solution presented could potentially be used in other environments, where resource-constrained hardware platforms might be available with the needed capacity to execute virtualization containers (e.g., the Internet of Things, or IoT, environments). In any case, the applicability of this solution to different environments and its potential adaptations will require a careful study in a case-by-case basis.

Finally, it should be noted that the results presented have been obtained in a laboratory environment and with the UAV devices grounded or following a limited and well-defined flight plan. Other scenarios involving outdoor deployments may introduce conditions affecting the stability of the flight of the UAVs, and hence the performance of the IP telephony service.

Disclosures

The authors have nothing to disclose.

Acknowledgments

This work was partially supported by the European H2020 5GRANGE project (grant agreement 777137), and by the 5GCIty project (TEC2016-76795-C6-3-R) funded by the Spanish Ministry of Economy and Competitiveness. The work of Luis F. Gonzalez was partially supported by the European H2020 5GinFIRE project (grant agreement 732497).

References

1. Sanchez-Aguero, V., Nogales, B., Valera, F., Vidal, I. Investigating the deployability of VoIP services over wireless interconnected Micro Aerial Vehicles. *Internet Technology Letters.* **1** (5), e40 (2018).
2. Maxim, V., Zidek, K. Design of high-performance multimedia control system for UAV/UGV based on SoC/FPGA Core. *Procedia Engineering.* **48**, 402-408 (2012).
3. Vidal, I. et. Al. Enabling Multi-Mission Interoperable UAS Using Data-Centric Communications. *Sensors.* **18** (10), 3421 (2018).
4. Vidal, I., Valera, F., Díaz, M. A., Bagnulo, M. Design and practical deployment of a network-centric remotely piloted aircraft system. *IEEE Communications Magazine.* **52** (10), 22-29 (2014).
5. Jin, Y., Minai, A. A., Polycarpou, M. M. Cooperative real-time search and task allocation in UAV teams. *In 42nd IEEE International Conference on Decision and Control.* (IEEE Cat. No. 03CH37475) (Vol. 1, pp. 7-12). IEEE (2003).

6. Maza, I., Ollero, A. Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems.* 6 (pp. 221-230). Springer, Tokyo (2007).
7. Quaritsch, M. et al. Collaborative microdrones: applications and research challenges. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems.* (p. 38). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2008).
8. Waharte, S., Trigoni, N., Julier, S. Coordinated search with a swarm of UAVs. In 2009 *6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops.* (pp. 1-3). IEEE (2009).
9. De Freitas, E. P. et al. UAV relay network to support WSN connectivity. In *International Congress on Ultra-Modern Telecommunications and Control Systems.* (pp. 309-314). IEEE (2010).
10. European Telecommunications Standards Institute. Network Functions Virtualisation (NFV); Architectural Framework; Research Report ETSI GS NFV 002 V1.2.1; *European Telecommunications Standards Institute.* (ETSI) (2014).
11. ETSI OSM. *An Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV.* Available online: https://osm.etsi.org/ last access on 8 August (2019).
12. Nogales, B. et al. Design and Deployment of an Open Management and Orchestration Platform for Multi-Site NFV Experimentation. *IEEE Communications Magazine.* **57** (1), 20-27 (2019).
13. Omnes, N., Bouillon, M., Fromentoux, G., Le Grand, O. A programmable and virtualized network & IT infrastructure for the internet of things: How can NFV & SDN help for facing the upcoming challenges. In the *18th International Conference on Intelligence in Next Generation Networks.* (pp. 64-69). IEEE (2015).
14. Rametta, C., Schembra, G. Designing a softwarized network deployed on a fleet of drones for rural zone monitoring. *Future Internet.* **9** (1), 8 (2017).
15. Garg, S., Singh, A., Batra, S., Kumar, N., Yang, L. T. UAV-empowered edge computing environment for cyber-threat detection in smart vehicles. *IEEE Network.* **32** (3), 42-51 (2018).
16. Mahmoud, S., Jawhar, I., Mohamed, N., Wu, J. UAV and WSN softwarization and collaboration using cloud computing. In the *3rd Smart Cloud Networks & Systems (SCNS).* (pp. 1-8). IEEE (2016).
17. González Blázquez, L. F. et al. NFV orchestration on intermittently available SUAV platforms: challenges and hurdles. In *1th Mission-Oriented Wireless Sensor, UAV and Robot Networking (MISARN).* IEEE (2019).
18. Nogales, B., Sanchez-Aguero, V., Vidal, I., Valera, F., Garcia-Reinoso, J. 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.* (pp. 39-44). ACM June (2018).
19. Nogales, B., Sanchez-Aguero, V., Vidal, I., Valera, F. Adaptable and automated small UAV deployments via virtualization. *Sensors.* **18** (12), 4116 (2018).
20. Hoban, A. et al. An ETSI OSM Community White Paper, OSM Release FOUR: A Technical Overview; Whitepaper; *European Telecommunications Standards Institute.* (ETSI) (2018).
21. Open Source MANO Release FOUR. *Quick start installation and use guide*. Available online: https://osm.etsi.org/wikipub/index.php/OSM_Release_FOUR last access on 8 August (2019).
22. OpenStack. *Open Source Software for Creating Private and Public Clouds.* Available online: https://docs.openstack.org/ocata last access on 8 August (2019).
23. *OpenStack Installation Tutorial for Ubuntu.* Available online: https://docs.openstack.org/ocata/install-guide-ubuntu/ last access on 8 August (2019).
24. Linphone. *An Open Source VoIP SIP Softphone for voice/video calls and instant messaging.* Available online: https://www.linphone.org last access on 8 August (2019).
25. *Jitsi.* An Open Source Project to easily build and deploy secure video-conferencing solutions. Available online: https://jitsi.org last access on 8 August (2019).
26. *Linux Containers (LXC).* Infrastructure for container projects. Available online: https://linuxcontainers.org last access on 8 August (2019).
27. *Ns-3.* A Discrete-Event Network Simulator for Internet Systems. Available online: https://www.nsnam.org/ last access on 8 August (2019).
28. *Kernel-based Virtual Machine (KVM).* A virtualization solution for Linux. Available online: https://www.linux-kvm.org last access on 8 August (2019).
29. *Bridging & firewalling.* Available online: https://wiki.linuxfoundation.org/networking/bridge last access on 8 August (2019).
30. *SIPp.* An Open Source test tool and/or traffic generator for the SIP protocol. Available online: http://sipp.sourceforge.net/ last access on 8 August (2019).
31. *Trafic. An open source flow scheduler.* Available online: https://github.com/5GinFIRE/trafic last access on 8 August (2019).
32. *Ubuntu Mate for the Raspberry Pi.* Available online: https://ubuntu-mate.org/raspberry-pi/ last access on 8 August (2019).
33. *OpenStack.* Enabling LXC (Linux Containers) as virtualization technology. Available online: https://docs.openstack.org/ocata/config-reference/compute/hypervisor-lxc.html last access on 8 August (2019).
34. *Open Source MANO Information Model.* Available online: https://osm.etsi.org/wikipub/index.php/OSM_Information_Model last access on 8 August (2019).
35. 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).