jove

Integration of 5G Experimentation Infrastructures into a Multi-Site NFV Ecosystem

Borja Nogales[*,1], **Luis F. Gonzalez**[*,1], **Ivan Vidal**[*,1], **Francisco Valera**[*,1], **Jaime Garcia-Reinoso**[*,1], **Diego R. Lopez**[*,2], **Juan Rodríguez**[*,2], **Neftali Gonzalez**[*,3], **Ignacio Berberana**[*,3], **Arturo Azcorra**[*,1,3]

[1] Department of Telematic Engineering, Universidad Carlos III de Madrid [2] Telefónica I+D [3] IMDEA Networks Institute

[*] These authors contributed equally

Corresponding Author

Borja Nogales
bdorado@pa.uc3m.es

Citation

Nogales, B., Gonzalez, L.F., Vidal, I., Valera, F., Garcia-Reinoso, J., Lopez, D.R., Rodríguez, J., Gonzalez, N., Berberana, I., Azcorra, A. Integration of 5G Experimentation Infrastructures into a Multi-Site NFV Ecosystem. *J. Vis. Exp.* (168), e61946, doi:10.3791/61946 (2021).

Date Published

February 3, 2021

DOI

10.3791/61946

URL

jove.com/video/61946

Abstract

Network Function Virtualization (NFV) has been regarded as one of the key enablers for the 5th Generation of mobile networks, or 5G. This paradigm allows to reduce the dependence on specialized hardware to deploy telecommunications and vertical services. To this purpose, it relies on virtualization techniques to softwarize network functions, simplifying their development and reducing deployment time and costs. In this context, Universidad Carlos III de Madrid, Telefónica, and IMDEA Networks Institute have developed an NFV ecosystem inside 5TONIC, an open network innovation center focused on 5G technologies, enabling the creation of complex, close to reality experimentation scenarios across a distributed set of NFV infrastructures, which can be made available by stakeholders at different geographic locations. This article presents the protocol that has been defined to incorporate new remote NFV sites into the multi-site NFV ecosystem based on 5TONIC, describing the requirements for both the existing and the newly incorporated infrastructures, their connectivity through an overlay network architecture, and the steps necessary for the inclusion of new sites. The protocol is exemplified through the incorporation of an external site to the 5TONIC NFV ecosystem. Afterwards, the protocol details the verification steps required to validate a successful site integration. These include the deployment of a multi-site vertical service using a remote NFV infrastructure with Small Unmanned Aerial Vehicles (SUAVs). This serves to showcase the potential of the protocol to enable distributed experimentation scenarios.

Introduction

The introduction of the fifth generation of mobile networks (5G) has implied revolutionizing the telecommunications industry since the beginning of the decade, requiring telecommunication operators to address the much more demanding specifications of the new networking services and applications developed under the 5G umbrella[1,2]. These

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

new specifications include, but are not limited to, data rate increases, wireless transmission latency improvements, and operational costs reduction. Among the technologies that constitute the foundations of the improvements for this new generation, Network Functions Virtualization[3] (NFV) has become one of its key enablers. NFV provides the capacity to softwarize network functions, traditionally relaying on specialized hardware, by using generic-purpose physical equipment instead, such as server computers in a datacenter. With this new paradigm, telecommunication operators and vertical industries can deploy network functions and services as a set of software components, and save costs in both service deployment and maintenance, as well as facilitating a much higher network infrastructure elasticity. This approach alleviates or eliminates the necessity to use dedicated (and usually more complex and less reusable) devices for most network and vertical-specific functions, and supports a much higher and denser degree of operational automation, hence reducing deployment and maintenance costs.

Taking into consideration all the advantages that an NFV environment is able to provide, it is natural that a great number of relevant stakeholders from the telecommunications sector have increasingly been involved in testing new service ideas on NFV environments. In this context, Telefónica and IMDEA Networks Institute have created 5TONIC[4], an open research and innovation laboratory focused on 5G technologies. Based in Madrid (Spain), this laboratory has a wide range of technologies available for researches and partners to boost the development and validation of 5G services. In particular, this laboratory has an experimental NFV platform where developers are able to deploy and test their new NFV-based applications and services over on an ETSI-compliant NFV ecosystem[5]. Thus, experimental conclusions about design choices and technology proposals can be derived in a realistic much more flexible environment than production networks. This platform has been designed to support experimentation activities across multiple external sites, which may be flexibly interconnected to 5TONIC using a well-defined protocol.

The technical solution adopted for the 5TONIC NFV ecosystem considers the utilization of a single NFV orchestrator, implemented using the ETSI-hosted Open Source MANO (OSM) software[6]. This is the element in charge of managing and coordinating the lifecycle of Network Services (NS). These services may be built as a composition of Virtualized Network/Vertical Functions (VNF), which can be deployed at any of the sites integrated on the NFV platform. The design of the 5TONIC NFV ecosystem has been done in the context of the H2020 5GINFIRE project[7,8], where the platform was used to support the execution of more than 25 experiments, selected through a competitive open-call process, across eight vertical-specific experimental infrastructures located in Europe and one in Brazil, the latter connected through a transoceanic link. In addition, the platform was leveraged to build a distributed NFV testbed at a national scale, in Spain, supporting experimentation activities within the Spanish 5GCity project[9,10]. More recently, an additional Brazilian site has been integrated into the platform, to support joint demonstration activities in the context of a research and innovation cooperation established between Brazil and Europe (i.e., the 5GRANGE project[11,12]). Last but not least, the infrastructure has been used to support third-party experiments in the scope of the 5G-VINNI project[13,14]. The geographic distribution of the NFV platform can be seen in **Figure 1**.

Interested organizations hosting their own NFV infrastructure can flexibly connect to the 5TONIC NFV ecosystem, subject to approval by the 5TONIC Steering Board, become testbed

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

providers within the distributed ecosystem, and be involved in joint experimentation and demonstration activities. To this end, they must feature a VIM (Virtual Infrastructure Manager) compliant with the OSM software stack. The 5TONIC NFV orchestrator is able to interact with the VIMs at the sites involved in a given service deployment, coordinating the allocation and setup of the computing, storage and network resources needed for the instantiation and interconnection of the VNFs that compose a network service, and controlling its lifecycle, from its on-boarding to its final decommissioning.

In order to manage the exchange of control and data traffic within all the interconnected sites, the 5TONIC NFV ecosystem makes use of an overlay network architecture based on Virtual Private Networks (VPN). This approach provides secure PKI-based access to the external sites that are integrated into the 5TONIC ecosystem, allowing the exchange of NFV control information between the OSM software stack and the different VIMs distributed across the testbeds, as well as the exchange of information that is required to manage and configure all the VNFs. Moreover, this overlay network supports the dissemination of data traffic among VNFs that are deployed at different sites.

In this context, this paper details the protocol designed to incorporate an external site to an NFV ecosystem. The protocol assumes that the ecosystem is governed by a single NFV orchestrator, installed at a central site, and external sites feature a VIM solution compliant with the orchestrator software stack. The proposed protocol allows to increment the portfolio of resources of the experimental ecosystem, with the flexible incorporation of NFV sites and vertical-specific infrastructures. This enables the creation of a distributed MANO platform capable of testing and validating novel network and vertical services across multiple sites, under the control of a single NFV orchestrator. In order to illustrate the inner operation of the protocol, the process will be exemplified by adding an external NFV site to the current 5TONIC NFV ecosystem, describing the needed components at the external site and 5TONIC, as well as all the steps to be taken during the integration process. **Figure 2** provides an overview of the objective of the integration, with the new NFV-based testbed attached to the 5TONIC platform from where network services can be deployed, by means of VPN connections between the central site and the rest of the external infrastructures.

In addition, to showcase the effectiveness of the protocol, the deployment of a simple vertical service will be shown, using the 5TONIC ecosystem and an external site with NFV-capable small unmanned aerial vehicles (SUAVs). The design of the vertical service has been inspired by an experiment presented in Vidal et al.[9], which has been simplified for the illustration purposes of this paper. **Figure 3** outlines the service, which aims at aiding smart farming activities on a remote area. The service considers a smart farming service provider who uses SUAVs to collect and disseminate the data produced by meteorological sensors scattered over a crop field. For simplicity, the experiment presented in the paper considers a single SUAV and a sensor, capable of providing temperature, humidity, and pressure measurements. In the experiment, the external NFV site hosts a Wi-Fi access point that is deployed as VNF over the SUAV. This VNF offers network access connectivity to the sensor, forwarding the sensed data towards a gateway function. The latter is deployed as a VNF on a ground equipment (a mini-ITX computer). The dissemination of data from the sensor to the gateway function follows a Publish/Subscribe approach based on the Message Queuing Telemetry Transport (MQTT) protocol[15].

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

The gateway function processes and then disseminates the data towards an Internet-of-things (IoT) server, which is made available as a VNF at the central site of the NFV ecosystem, based on the Mainflux[16] open-source platform. Finally, the scenario assumes a remote area where Internet connectivity is provided by a cellular non-3GPP access network. Hence, the service includes two additional VNFs: 1) an access router VNF, which implements the user-plane protocol stack of a 3GPP user equipment connected to a non-3GPP access network[17]; and 2) a baseline implementation of a 5G core network, supporting the forwarding of information between the access router and the IoT server VNFs. To this purpose, the 5G core VNF provides a simplified implementation of the user-plane of a non-3GPP interworking function and a user plane function, as defined by 3GPP[17].

Finally, **Figure 4** represents the most relevant processes involved during the development of the protocol, highlighting their logical interconnections and the entities in charge of their execution.

Protocol

1. Provision of the central site of the NFV ecosystem (prior requisites of the experiment)

1. Allocate an IP address space to be used by the central site. For the purposes of this protocol, the private address space 10.4.0.0/16 will be used.
2. Install the Management and Orchestration (MANO) software stack in the central site. In particular, the experiment carried out throughout this protocol uses the Open Source MANO (OSM) Release SEVEN[18], which requires the following resources: Ubuntu 18.04 as operating system, 2 Central Processing Units (CPUs), 8 GB of Random-Access Memory (RAM), 40 GB hard-drive-disk, and at least one network interface with Internet access. For the installation, follow the instructions available at the OSM Release SEVEN documentation[18].
3. Set up a Virtual Infrastructure Manager (VIM) compatible with OSM in the central site. Specifically, the experiment uses OpenStack release Ocata[20], running on a Virtual Machine (VM) with Ubuntu 16.04 , 4 CPUs, 16 GB of RAM and 200 GB of hard drive. The NFV Infrastructure (NFVI) handled by this VIM comprises three server computers, each with Ubuntu 16.04, 8 CPUs, 32 GB of RAM and 2 TB of storage. For the installation, follow the Ocata release documentation[21].
    1. Deploy a virtual network within the OpenStack cloud platform, using an IP address range from the address space allocated in step 1.1. This network, henceforth referred to as management network, will be used to support the exchange of NFV orchestration information between the OSM and the virtual network functions (VNFs) instantiated at the central site.
    2. Configure a virtual network (henceforth denominated as data network) to support inter-site data communications, between the VNFs of the central site and other VNFs executed at external sites. To this end, use an IP address range from the address space of step 1.1.

       NOTE: The implementation of the networks mentioned in steps 1.3.1 and 1.3.2 has been done using provider networks of OpenStack. Provider networks must be connected to the physical network infrastructure of the central site to guarantee an appropriate operation.

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

4. Connect both virtual private networks (i.e., the management and the data networks), as well as the VIM and the OSM machines, to an equipment providing edge routing functionalities. This router will serve as the entry point to the central site of the NFV ecosystem.

5. Make available a public experiment repository to provide all the content needed to carry out the experiment. In particular, this protocol uses the public repository at[22].

2. Configuration of the virtual private network service

1. Allocate an IP address space to support the appropriate operation of the multi-site ecosystem, so that network communications can effectively be established among multiple sites.

   NOTE: Enabling effective network communications among multiple sites requires a careful design of the IP address space to be used by the NFV ecosystem, as well as by external sites that need to connect to it. In particular, the address space allocated for inter-site communications should not collide with the address space already in use at every other site for other purposes.

   1. Allocate an IP address space to be used by external sites. Addresses in this block will be assigned to NFV entities (e.g., VIMs) and VNFs of the external site. To exemplify this protocol, the private address space 10.154.0.0/16 will be used.

   2. Allocate an IP address space to the virtual links between the external sites and the NFV ecosystem. These virtual links will be supported by a VPN service. To exemplify this protocol, the address range 10.154.254.0/24 will be utilized for these virtual links.

2. Set up an equipment to provide the Virtual Private Network (VPN) service (i.e., a VPN server). In particular, the experiment uses a server computer with Ubuntu 16.04 (64-bit variant image), six independent CPUs, 16 GB RAM, 1 TB storage disk, and two network interfaces.

   1. Configure one of the network interfaces of the VPN server to allow the reception of connection requests from external sites through the Internet. To that end, it is necessary to use an interface of the server configured with a public IP address.

   2. Configure the link between the VPN server and the edge router of central site. In the experiment this link was allocated the address range 10.4.255.0/24. Configure appropriate network routes at the VPN server, so that the NFV ecosystem becomes accessible from external sites connected to the VPN service.

3. Install the VPN open-source software provided by the *OpenVPN*[23] project into the VPN server. Specifically, this experiment uses the *OpenVPN* version 2.3.10, and its deployment was done with the bash script "openvpn-install.sh", available at http://github.com/Nyr/openvpn-install (other installation options are described in the *OpenVPN* documentation[24]). The bash script presents the alternative parameters that will result in the configuration of the VPN service.

   1. Select the IP address to listen to VPN connection requests (i.e., the public IP address).

   2. Decide which protocol (UDP or TCP) should be used to drive the communications over the VPN. In this

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

case, the experiment leverages on UDP that is the recommended protocol.

3. Specify the port that will comprise the duple (together with the public IP address) that will be used to receive the service connection requests. By default, the assigned value is 1194.

4. Choose one of the DNS servers of the list prompted by the assistant that will handle the name resolution requests performed by the clients of the VPN service.

5. Press any key to enable the automatic initiation of the VPN service installation process.

4. Edit the configuration file "server.conf" that is located under the "/etc/openvpn/server/" directory and include the "client-to-client" directive aiming to extend the basic setup provided by step 2.3. Thus, different clients connected to the VPN service will be able to reach each other.

5. Enable the individual client configuration within the VPN setup to be able to independently manage the routing assignments for each client.

    1. Add the "client-config-dir ccd" directive, editing the same configuration file as in step 2.4.

    2. Create the directory "ccd" using the command "mkdir /etc/openvpn/ccd/". This directory will serve during the next section of the protocol to place the files comprising the routing directives associated to the clients intended to be integrated within the platform.

6. Set up the firewall rules that are needed to allow the connections with the service, while protecting the VPN server against malicious attack. To that end, this experiment leverages on *iptables*[25], which is a command line utility developed to configure the Linux kernel firewall.

    1. First, block incoming traffic to the VPN server with the command "iptables -P INPUT DROP".

    2. Allow the reception of VPN connection requests with the commands "iptables -A INPUT -i <public-intf> -m state --state NEW -p udp --dport 1194 -j ACCEPT" (<public-intf> is the name of the VPN server interface with the public IP address) and "iptables -A INPUT -i tun+ -j ACCEPT".

    3. Allow traffic forwarding between the VPN server interfaces (i.e., the public interface and the virtual interface created by the VPN service called *tun0*), to enable the VPN server to process the service connection request. For this purpose, execute the command "iptables -A FORWARD -i tun+ -o <public-interface> -m state --state RELATED,ESTABLISHED -j ACCEPT && iptables -A FORWARD -i <public-interface> -o tun+ -m state --state RELATED,ESTABLISHED -j ACCEPT".

    4. Enable the VPN server to provide the network address translation (NAT) capability with the aim of supplying Internet access to the central site, executing: "iptables -t nat -A POSTROUTING -s 10.4.0.0/16 -o <public-interface> -j MASQUERADE && iptables -A OUTPUT -o tun+ -j ACCEPT".

3. Integration of an external NFV site

1. Obtain an appropriate IP address range to integrate the site into the NFV ecosystem. This address range will be provided by the network operations center of the NFV ecosystem. According to step 2.1.1 of this protocol, the

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

experiment will use a range of IP addresses for the external site within 10.154.0.0/16.

2. Create and provide the security credentials to connect to the NFV ecosystem.

    1. Generate a VPN credential that will allow the new infrastructure to establish a secure connection with the VPN server. To this purpose, execute the command "bash openvpn-install.sh" in the VPN server, select the option "1) Add a new client" of the prompted list, and provide the name to be associated with that credential, e.g., uc3m_infrastructure. This step will generate a file with the VPN credentials (named "uc3m_infrastructure.ovpn" in the example).

    2. Create a text file in the "/etc/openvpn/ccd/" directory of the VPN server, including the routing directives (as specified in the *OpenVPN* documentation[24]) that must be pushed by the VPN server each time a connection to the VPN service is established using the VPN credentials.

        NOTE: The name of the text file must match the name specified during the creation of the VPN credential (e.g., uc3m_infrastructure) to provide a customized configuration for every VPN client.

    3. Provide the VPN credential file to the technical staff of the external site. This must be done through a secure and reliable channel. In this experiment, a manual encryption process is used. To encrypt the VPN credential, execute the command "7za a -tzip '-p<password>' -mem=AES256 <encrypted-file> <credential-name>", setting <password> as the desired encryption key, <encrypted-file> as the chosen name for the encrypted file, and <credential-file> as the file name of the VPN credential file (e.g., uc3m_infrastructure.ovpn).

    4. Provide the encrypted credential to the technical staff of the new site, along with the key that allows the decryption procedures, through a secure communications channel.

        NOTE: In this experiment, the encrypted credentials were provided by electronic email, whereas the decryption key was sent through a separate channel, using the short message service (SMS), with an offline agreement of the telephone number.

3. Set up the environment at the new site, so as to establish the connection with the NFV ecosystem, and to allow the remote NFVI be attached to the OSM stack of the central site.

    1. Install the VPN software provided by *OpenVPN*[24] in a computer, to enable a virtual link between the external site and the central site of the NFV ecosystem. The computer with the *OpenVPN* software will serve as a VPN client or VPN endpoint at the external site. The virtual link will be realized by means of a protected VPN tunnel between the VPN endpoint and the VPN server. In the experiment, the VPN endpoint runs in a server computer with Ubuntu 18.04, 8 CPUs, 8 GB RAM, 128 GB storage disk, and 3 GbE interfaces (one for connecting with the VPN service over Internet).

    2. Activate IP forwarding in the VPN endpoint to support network routing capabilities. To that end, include the line "net.ipv4.ip_forward=1" in the system configuration file located in the "/etc/sysctl.conf" path, and load the updated configuration with the command "sudo sysctl -p".

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

3. Decrypt the VPN credential file with the information received in step 3.2.4, using the command "7za e <encrypted-file>", where the <encrypted-file> is the file name of the encrypted VPN credential. Specify the decryption key when prompted by the command.

4. Boot the *OpenVPN* software with the decrypted credential file using the command "sudo openvpn --config <credential-file>" (<credential-file> is the file name of the VPN credential). With this, the VPN endpoint will authenticate to the VPN server, and will automatically receive appropriate VPN configuration parameters and network routes. This way, the VPN endpoint will behave as an edge router with a virtual link to the central site of the NFV ecosystem.

5. Verify the proper operation of the VPN endpoint, using the *ping* command to verify the availability of connectivity to one of the nodes of the central site (e.g., the OSM stack equipment).

6. In the new site, select an OSM compliant VIM to allow operations with the MANO platform. For this experiment, OpenStack release Ocata is used.
   NOTE: OSM Release SEVEN supports the following virtual infrastructure managers: OpenStack, OpenVIM[26], VMware's vCloud Director[27], Amazon Web Service[28], Microsoft Azure[29], and Eclipse fog05[30] (see OSM documentation[18] for specific configuration details).

7. Install OpenStack release Ocata[20] (see the detailed procedures in the release documentation[21]).

8. Deploy the NFV infrastructure in the external site and attach it to the VIM. In particular, this experiment uses an NFV infrastructure comprising three single board computers (SBCs), each with a compute capacity of 1 GB RAM, 4 CPUs, and 32 GB storage disk; and a single mini-ITX computer with 8 CPUs, 8 GB RAM and 128 GB for storage.
   NOTE: The external site exemplified in this protocol is based on an NFV infrastructure of NFV-capable small unmanned aerial vehicles (SUAVs) . The details followed to enable such infrastructure are provided in Nogales et al[31]. Steps 3.3.6 to 3.3.8 are optional, as an NFV infrastructure may already exist at the external site.

9. Create an OpenStack project to specify the set of computational resources of the external site that will be integrated into the NFV ecosystem. To do so, access to the graphical user interface (GUI) provided by OpenStack, log in to the system with the administrator credentials, click on the + **Create Project** button of the **Identity -> Projects** tab, and create a project completing the displayed form with the requested information.

10. Create a valid user that will manage the project created in the previous step. To this purpose, access the **Identity -> Users** tab with the same login as in the previous step, click on **+ Create User,** and fill in the required fields of the displayed form (username and password), selecting the new created project as the primary project, and choosing the **admin** role.

11. Modify the security rules to allow VNF communication permissions in the new site (in particular, enable SSH and ICMP traffic). To that end, access the OpenStack GUI with the credentials of the user created in the previous step, follow the sequence: **Project -> Network -> Security Groups -> + Add Rule**, and select the **SSH** option of the **Rule**

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

drop-down. Repeat the process but selecting the **All ICMP** option included in the drop-down menu.

12. Download the images of a trial service offered by the OSM community, the *Ping Pong* network service ("Fedora-x86_64-20-20131211.1-sda-ping" and "Fedora-x86_64-20-20131211.1-sda-pong") from the public experiment repository, and upload them to the VIM of the external site. To this purpose, follow the sequence **Project -> Compute -> Images -> + Create Image**, and create the images using the displayed form and selecting each of the image.

13. Assign two IP address ranges within the address space of the external site (allocated in step 3.1). These ranges will be used to support the management of the VNFs of the external site and to enable inter-site data communication among VNFs, respectively.

14. Create a provider network (*control-provider*) using the VIM. This network will support NFV communications between the OSM stack at the central site and the VNFs deployed at the new site for management purposes. This type of communications will also enable the OSM stack to configure VNFs after their deployment. To create a provider network in OpenStack, follow the sequence **Admin -> System -> Networks -> + Create Network** and fill in the details of the new network, using the selected IP address range in the previous step.

15. Create a second provider network (*data-provider*) using the VIM. This network will support data communications among the VNFs of the site and other VNFs of the NFV ecosystem. To create this provider network in OpenStack, follow the sequence **Admin -> System -> Networks -> + Create Network**, and fill in the details of the new network using the assigned address range.

    NOTE: Instructions on how to create virtual networks will vary depending on the VIM software. Check their respective software documentation for details.

16. Share the VIM-related information (in particular, the username/password, and the project created in the steps 3.3.9 and 3.3.10) with the technical staff of the central site, to enable the attachment of the VIM to the OSM software stack.

4. Attach the external NFV infrastructure to the OSM software stack of the central site, using the information obtained from the step 3.3.16.

    1. Verify the connectivity between the OSM stack of the central site and the VIM of the new site, using the *ping* tool.

    2. If the previous connectivity test is successful, attach the external VIM to the OSM stack of the central site. To do so, use the following command in the OSM machine: "osm vim-create --name <external-VIM-name> --user <VIM-username> --password <VIM-user-password> --auth_url <authentication URL> --tenant <project-name> --account_type <VIM-type>". In this command: <external-VIM-name> is the name selected to identify the VIM within the OSM stack, <VIM-username> is the name of the user authorized to handle the resources of the external site (see step 3.3.10), <VIM-user-password> is the password of the indicated user, <authentication URL> is the link to the API made available by the VIM to enable

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

requests from the OSM stack , <project-name> is the project name defined in step 3.3.9, and <VIM-type> is the VIM software used (in this experiment, OpenStack).

5. Verify the appropriate attachment of the new VIM to the OSM stack of the NFV ecosystem.
    1. Execute the command "ro_id=$(docker ps | grep osm_ro | cut -d ' ' -f 1)" to identify the id of the container implementing the Resource Orchestrator (RO) module within the OSM system. This module is the responsible for interacting with the VIMs in order to coordinate and allocate the needed resources in the deployment of subsequent network services.
    2. Access the RO container using the command "docker exec -it $ro_id bash". This command utilizes the identifier obtained in the execution of the previous step.
    3. Check that the new VIM is included in the list of available datacenters, using the command "openmano datacenter-list". The new site should appear in the list with the same name as the previously introduced one in the step 3.4.2 with the <external-VIM-name> parameter.
    4. List the images that have been uploaded to the VIM of the external site, using the command "openmano vim-image-list --datacenter <external-VIM-name>". The <external-VIM-name> parameter indicates the name selected to identify the VIM within the OSM stack. If the execution of this command is successful, the connectivity with the external VIM has successfully been stablished. Check that the *Ping Pong* images are included in the list.
    5. List the networks available at the new site with the command "openmano vim-net-list --datacenter <external-VIM-name>". Check that *control-provider* and *data-provider* are present.
6. Perform a preliminary validation of the proper integration of the new site, using a trial service offered by the OSM community (all the content in this regard is included within the experiment repository). For this purpose, the commands included in the following steps will be executed in the equipment hosting the OSM stack.
    1. Onboard the VNF descriptors (VNFDs) to the OSM stack running the command "osm vnfd-create <vnfd-package>" for each of the VNFs composing the trial service (<vnfd-package> corresponds to the file name of the VNFD package).
    2. Onboard the NS descriptor (NSD) of the trial service with the command "osm nsd-create <nsd-package-descriptor>", where <nsd-package> indicates the file name of the NSD package (in this experiment, ping_pong_ns.tar.gz)."
    3. Start the instantiation of the *Ping Pong* Network Service (NS) on the external and the central sites, using the command "osm ns-create --ns_name <instantiation-name> --nsd_name ping_pong_ns --vim_account <external-VIM-name> --config '{vnf: [{member-vnf-index: '2', vim_account: <central-VIM-name>}]}'". The <external-VIM-site> parameter identifies the VIM of the external site within the OSM stack. The "--config" option indicates that all the VNFs composing the service must be deployed on the external site handled by that VIM, except the VNF identified by the index 2 in the NS, which will be deployed in the central site (the VIM of the

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

central site is specified in the <central-VIM-name> parameter).

4. Check that the NS has been deployed and its status using the command "osm ns-list". If the instantiation is successful, the status will change to "READY".

5. Check the IP address of each of the two VNFs with "osm vnf-list" (necessary to log in to the machines afterwards).

6. Connect to each VNF via SSH, using the command "ssh fedora@<VNF-IP>" (<VNF-IP> represents the IP address of the VNF to connect to, obtained in the previous step). Introduce the password "fedora" when prompted by SSH. Once logged into both machines, check their interfaces using the command "ip address show", and obtain the IP addresses on their interfaces attached to the *data-provider* network (interface eth1 in both VNFs). From one of the VNFs, perform a ping to the other VNF, using the remote IP address in the *data-provider* network. If there is connectivity, the preliminary validation test will be considered successful.

4. Validation of the NFV multi-site platform with a realistic vertical service

1. Download the VNF images from the public repository and upload them into the VIM of their corresponding site (see **Figure 3**), following the procedure detailed in the step 3.3.12. In particular, the external site will host the *Access Point VNF, Router VNF, MQTT Gateway VNF, and Access Router VNF*. The central site will host the *5G Core VNF* and the *IoT Server VNF*.

2. Onboard the VNFDs and the NSD of the smart farming service to the OSM stack (all the descriptors can be downloaded from the experiment repository).

    1. Onboard the VNFDs to the OSM stack executing the command "osm vnfd-create <vnfd-package>", for each of the VNFs of the network service. In this case, the <vnfd-package> parameter corresponds to the file name of the VNFD package.

    2. Onboard the NSD to the OSM stack with the command "osm nsd-create <nsd-package>", where <nsd-package> indicates the file name of the NSD package (in this experiment, *jove_uavs_scenario_nsd.tar.gz*).

3. Deploy the smart farming network service. To this purpose, run the following command from the OSM command line interface: osm ns-create --ns_name <instantiation-name> --nsd_name jove_uavs_scenario_nsd --vim_account <external-VIM-name> --config '{vnf: [{member-vnf-index: "5", vim_account: <central-VIM-name>}, {member-vnf-index: "6", vim_account: <central-VIM-name>}], wim_account: False }'.

    NOTE: As indicated in the step 3.6.3., the <external-VIM-name> and <central-VIM-name> parameters indicate the sites where the VNFs are to be deployed. Particularly, all the VNFs composing the smart farming service will be placed into the new external site, except for those with index 5 and 6 (the *5G Core* and the *IoT server VNFs*) that will be allocated to the central site.

4. Check that the NS has been deployed, following the same procedure as in step 3.6.4.

5. Access to the *IoT server VNF* with the command "ssh mosquittosubscriber@<VNF-IP>" and check its interface configured to communicate with *MQTT Gateway VNF* through the command "ip address show dev eth1". The

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

IP address of the VNF (<VNF-IP>) can be obtained executing the "osm vnf-list" in the OSM command line.

6. Following an analogous procedure, access to the *MQTT Gateway VNF*, and run the command "sudo python3 publisher_MQTT_GW.py -ma <IoT-IP> -ba <MQTT-GW-IP>" where the <IoT-IP> is obtained in the previous step, and the <MQTT-GW-IP> executing the "ip address show dev eth1" command in the *MQTT Gateway VNF*. This step initializes the *MQTT Gateway VNF*, which will receive data generated by the sensor using the MQTT standard[15], transmitting these data to the *IoT server VNF* using the same standard.

7. Prepare a Single Board Computer (SBC) attaching a meteorological sensor, and with transceiver capacity to transmit sensor readings towards the MQTT Gateway VNF.

   NOTE: To exemplify this protocol, an SBC model in particular has been used. Hence, the following steps may need to be adapted in case of utilizing a different SBC platform.

   1. Connect (e.g., using tin-soldered copper wires) the board pins of the sensor to the general-purpose input/output (GPIO) pins of the SBC, following the configuration scheme of **Figure 5**.

   2. Enable the I2C kernel module in the SBC to be able to verify if the sensor is detected. For this purpose, run the command "sudo raspi-config", follow the sequence **Interfacing Options -> I2C -> Yes** in the displayed menu, and reboot the SBC to make the changes effective.

   3. Verify that the sensor is detected Installing the software *i2c-tools* in the SBC, and executing the command "sudo i2cdetect -y 1". If so, a grid should appear indicating the position where the sensor is detected.

   4. Install the appropriate software libraries to allow the SBC reading and sending the data provided by the sensor. In particular, this experiment leverages on the *RPi.bme280*[32] and *paho-mqtt*[33] Python libraries.

8. Using the mobile application of the SUAV, take off the aerial vehicle that hosts the *Access Point VNF*, and position it to provide wireless coverage to the SBC with the sensor.

   NOTE: The flight of the NFV-capable SUAVs are independent from the operational behaviour of the network service, which is able to operate whether the SUAVs are flying or in a state of repose to mitigate battery consumption. Thus, the step 4.8 is optional.

9. Attach the SBC in charge of reading the data collected by the sensor to the Wi-Fi wireless access point provided by the *Access Point VNF*). After a successful attachment, a wireless network path will be enabled from the sensor to the *MQTT Gateway VNF*.

10. Start the transmission of sensed data, running the command "python3 /home/ubuntu/ sensorDataTransmission.py -a <MQTT-GW-IP>" in the SBC that incorporates the sensor (<MQTT-GW-IP> is the IP address obtained in the step 4.6.).

11. Access the web GUI provided by the *IoT server VNF* to check the correct real-time reception of the sensed data. To that end, check the IP address of the IoT Server VNF with the command "osm vnf-list", and type the following Uniform Resource Locator (URL) in a web-browser: http://<VNF-IP>:3001, where <VNF-IP> is the IP address of the *IoT server VNF*. Then, click on the **Sensors Data**

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

Collection button of the **Home** tab, and verify the real-time update of the graphs included in the dashboard as data are received.

NOTE: To be able to access to the URL mentioned in the step 4.12, the device with the web-browser attempting to reach that resource must be connected to the NFV ecosystem and have IP connectivity with the *IoT Server VNF*. The VPN service can be also used for this purpose.

12. Wait for an appropriate period of time to obtain representative results of the execution of the smart farming service. Then, collect the data stored in the *IoT server VNF* for further analysis. Considering that the sensor included in this experiment provides temperature, humidity and pressure readings every 5 seconds, the service in the experiment run for a period of 10 minutes, resulting in 180 samples of sensed data (60 for each meteorological value type).

13. Access the database of the *IoT Server VNF* to retrieve the sensed data for further analysis. To this purpose, execute the command "id_database=$(sudo docker ps | grep 'influxdb:' | cut -d ' ' -f 1)" on the IoT Server VNF, and then "sudo docker exec -it $id_database bash"

14. Export the data to a comma-separated value (CSV) file, running the command "influx -database 'mainflux' -execute "SELECT * FROM messages WHERE \"name\" = '<data>' " -format csv > /tmp/<filename>.csv". Modify the parameter <data> to select which type of sensed data is to be export with the "temperature", "humidity" or "pressure", and set the <filename> parameter to choose a name for the output file that will keep the results.

15. Save the data files generated in the previous step for later representation (see Representative Results section) and verification of proper operation of the smart farming service.

Representative Results

After carefully following the protocol to incorporate a new site to the central platform and run one network service to validate its proper functionality, **Figure 6** depicts a screenshot of the open-vpn-monitor tool. It can be observed how the new site is using the VPN for all its communications, showing how its communications follow the VPN to allow this data exchange and, in consequence, the correct addition of the new site to the VPN service.

As depicted in **Figure 3**, the network service is delivering information from a sensor located in a remote infrastructure to the server located in the central site. In addition, **Figure 7** displays the successful deployment of the network service from the OSM web GUI, showing how the experiment can be properly instantiated in the new remote infrastructure from the MANO stack located within the central site. Moreover, the time required in the experiment to complete the deployment of the service is around eight minutes. This value, along with the time needed to on-board the service descriptors into the orchestration platform (about 9 seconds, with 1.3 seconds per descriptor, considering both the NS and each VNF descriptors), enable to satisfy the Key Performance Indicator (KPI) of 90 minutes for the service creation time, as indicated by the 5G Infrastructure Public Private Partnership[34]. In this context, the work presented in Vidal et al.[9] includes an in-depth analysis of the service creation time with multiple sites using the presented protocol.

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

Figure 8 displays the data collected from the sensor, including the values of humidity, temperature and pressure respectively. These samples correspond to all data sent from the sensor to a remote server located in 5TONIC, where these values are stored in a database. All these data demonstrate that the platform is able to deploy practical network services after the inclusion of a new infrastructure, as well as to correctly enable communications between sites.

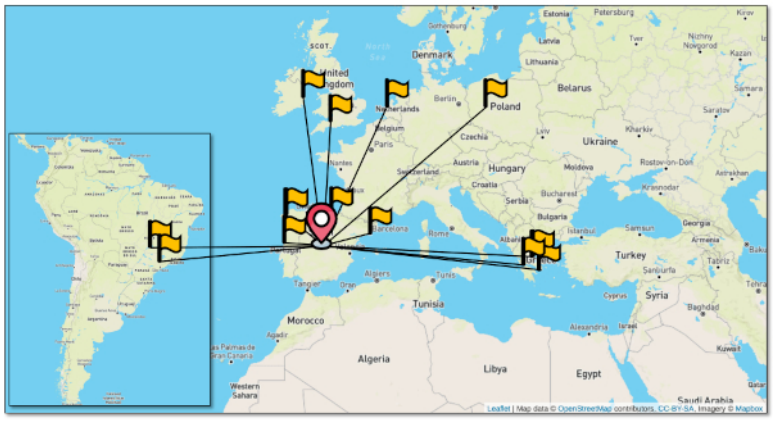


Figure 1: VPN service site distribution. Distribution of the VPN service through the platform and their link connectivity (all passing through 5TONIC). Please click here to view a larger version of this figure.

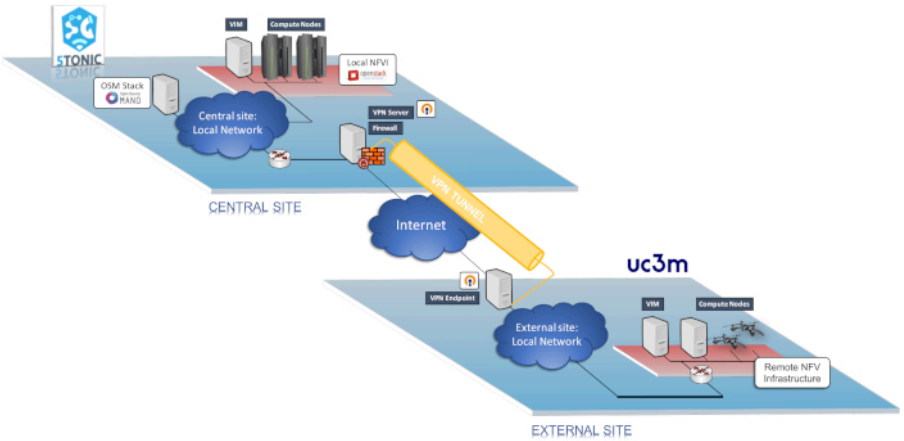


Figure 2. Overview of the platform and VPN service. This figure shows all elements of the platform: the central location, along with its NFV Infrastructure, the VPN service and a new infrastructure aggregated to the system. It also includes the connections between its elements. Please click here to view a larger version of this figure.

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

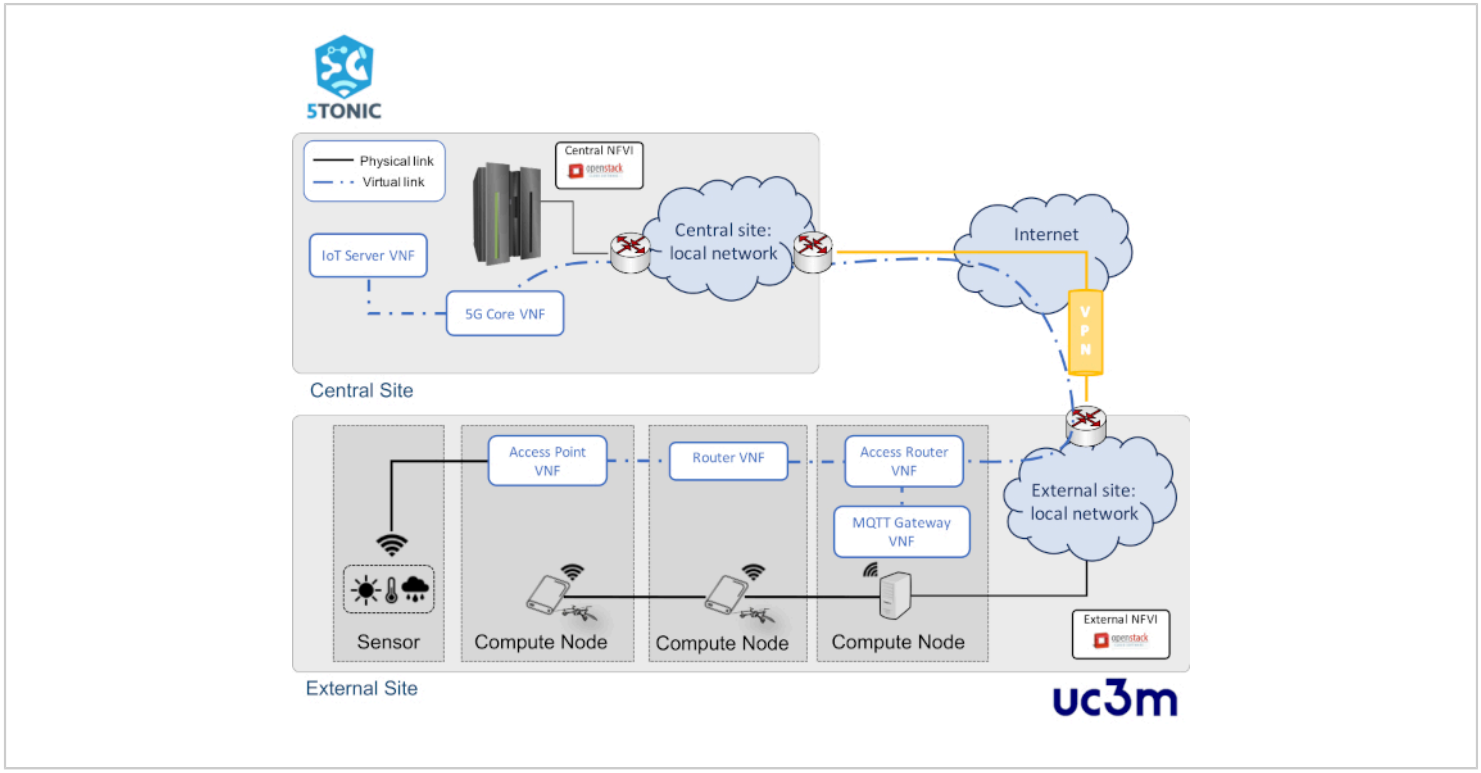


Figure 3: Overview of the network service. It depicts the elements involved in the network service, its distribution and its logical, and networking, connectivity. Please click here to view a larger version of this figure.

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

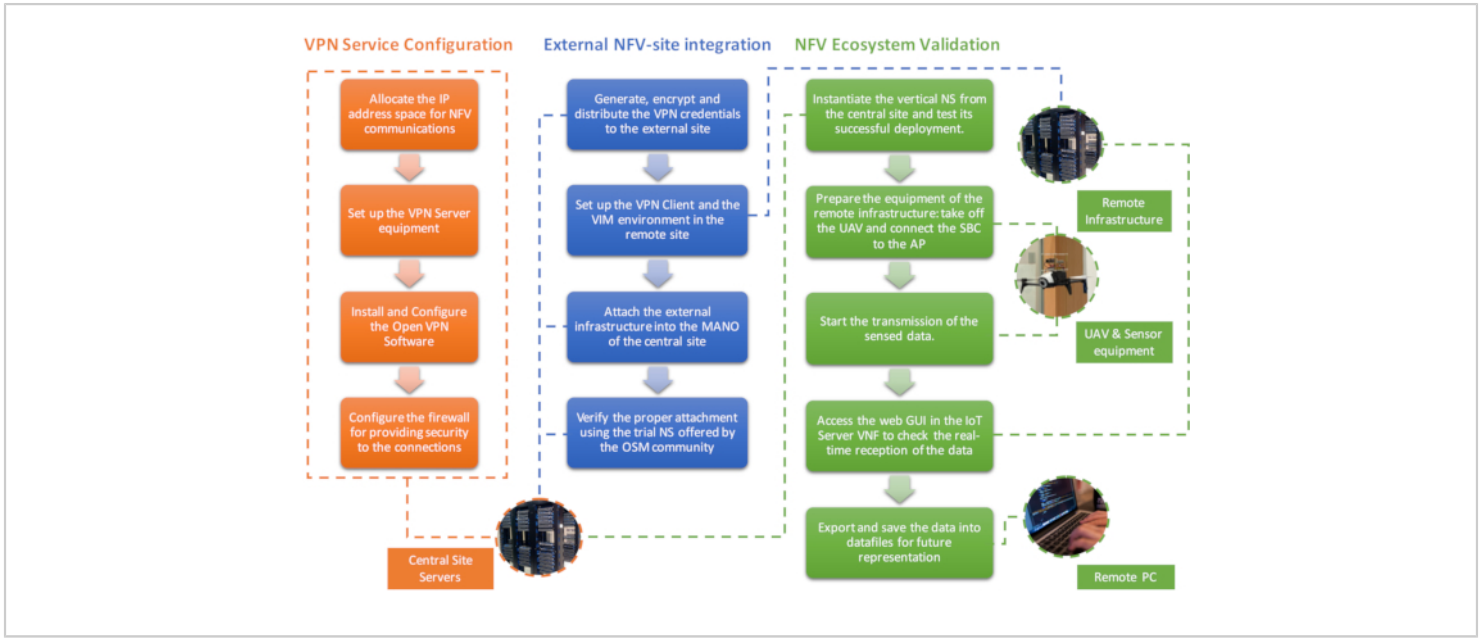
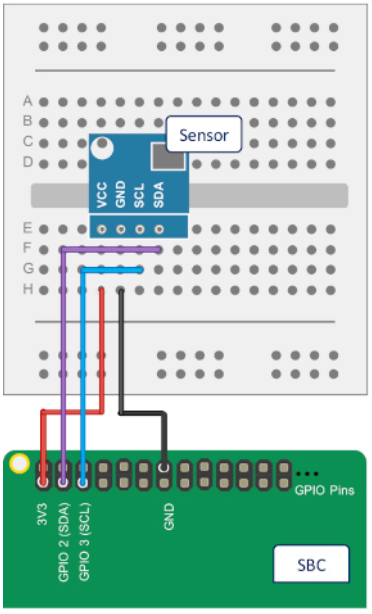


Figure 4: Protocol Workflows. Each column represents one section of the protocol, where every action performed is described, its logical connection between them and the component in charge of its execution. Please click here to view a larger version of this figure.

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License



Sensor Board Pins	SBC GPIO Pins
VCC	3V3
GND	GND
SCL	GPIO 3 (SCL)
SDA	GPIO 2 (SDA)

Figure 5: Pin configuration scheme. Diagram representing how to make the physical connections between the board pins of the sensors and the GPIO pins of the SBC that incorporates that sensor. Please click here to view a larger version of this figure.

Figure 6: OpenVPN-monitor snapshot. The picture shows that the aggregated infrastructure is connected to the VPN service, including some of its details regarding its connection. Moreover, the figure also depicts additional connections belonging to other remote infrastructures. Please click here to view a larger version of this figure.

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

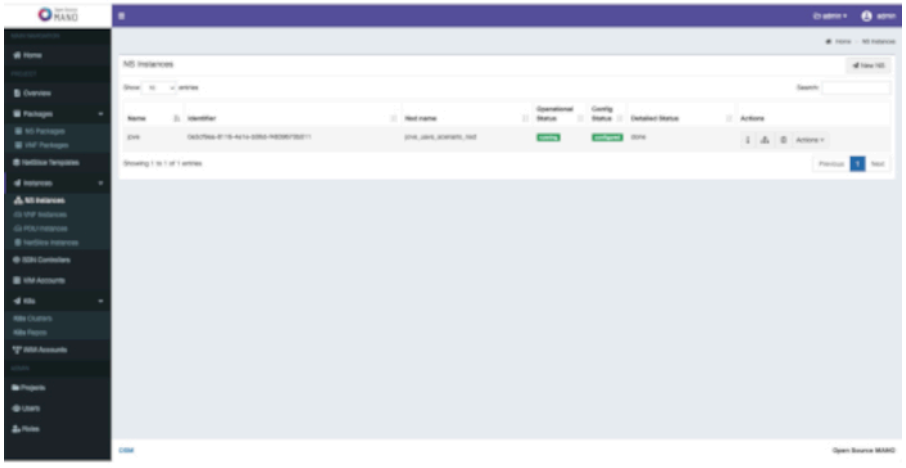


Figure 7: OSM NS deployment status. OSM graphical interface, showing the successful deployment of the test network service in the remote infrastructure. Please click here to view a larger version of this figure.

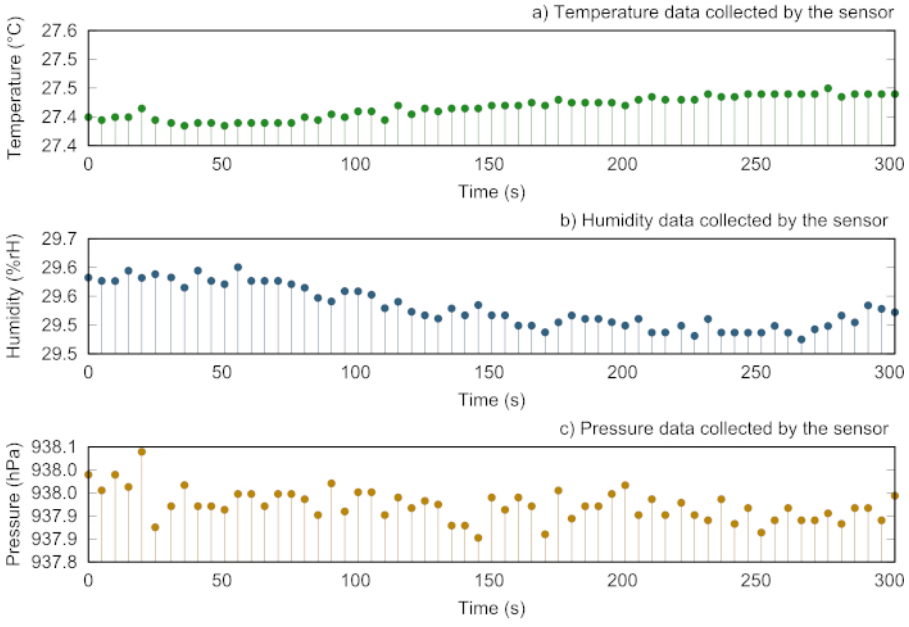


Figure 8: Representative analysis of the data collected by the sensor. (A) Illustration of the temperature data periodically collected by the sensor every 5 seconds. (B) Graphical representation of the humidity data collected by the sensor every 5 seconds. (C) Visual depict of the pressure data collected by the sensor every 5 seconds. Please click here to view a larger version of this figure.

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

Discussion

One of the most important aspects of the previously described protocol is its outstanding flexibility to incorporate new computational infrastructures to an NFV ecosystem, regardless of their distribution in terms of geographical location (as long as bandwidth and latency of the network communications with remote sites supports it). This is possible through a VPN-based overlay network architecture, which enables the establishment of a virtual link to connect remote sites to the central premises of the NFV ecosystem. This approach enables the provision of an effective and secure channel to support the NFV and data communications among sites of an NFV ecosystem, reducing the probability of external parties accessing and/or modifying sensitive information regarding NFV orchestration processes and data from deployed services. In this context, the protocol also describes a specific methodology to share securely the VPN credentials with the external sites that will enable the integration of new infrastructures. The protocol has been exemplified using the NFV ecosystem made available at 5TONIC by Universidad Carlos III de Madrid, Telefónica, and IMDEA Networks Institute, although it is generic to be utilized in other NFV environments satisfying the prior requisites mentioned in step 1 of this protocol.

In addition, it is worth emphasizing the exclusive utilization of open-source tools and software for the protocol implementation. Notwithstanding the potentially beneficial functionalities that could be offered by different proprietary solutions (e.g., *Fortinet*[35]), the use of open-source developments has facilitated the integration of all elements encompassed by the protocol due to their inherent characteristics such as cost effectiveness, an extensive software support provided by the open-source community, and a high level of reliability, just to name a few of them. Moreover, the utilization of open-source technologies can also promote synergies between components of similar nature. For instance, in order to monitor the VPN connection status for the clients using the platform, the VPN service implemented throughout the protocol could rely on the open-vpn monitor tool[36] (a python-based monitoring tool capable of interoperating with OpenVPN servers).

On the other hand, the protocol specification considers the instantiation of networking services across different sites for validation purposes. In this regard, it is important to highlight that the deployment of services on a given site is subject to the availability of compute, storage and network resources at the site, as well as of specialized equipment that might be needed to perform the deployment (e.g., NFV-enabled SUAVs). This is not a limitation of the protocol, and should be taken into account by stakeholders interested in reproducing the experiment described in this paper.

Moreover, it should be noted that the time required to carry out the deployment of network services highly depends on several factors such as the network path between the orchestrator and the different VIMs, the performance of data communications between the VIM and its managed computational nodes, and also in the intrinsic nature of these computational nodes (not only because of their available computing resources, but also the technologies incorporated to conduct the virtualization of network functions).

Finally, and given the outstanding performance that this platform and its VPN service had on the European projects and collaborative works where it has been used so far (e.g., 5GINFIRE, 5GRANGE or 5GCity, mentioned in the introduction of this document), it will be regarded as an important element in emerging European projects where Universidad Carlos III de Madrid, Telefónica, and IMDEA

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

Networks Institute participate, such as the Horizon 2020 LABYRINTH, or national projects, like TRUE-5G.

Disclosures

The authors have nothing to disclose.

Acknowledgments

This work was partially supported by the European H2020 LABYRINTH project (grant agreement H2020-MG-2019-TwoStages-861696), and by the TRUE5G project (PID2019-108713RB-C52PID2019-108713RB-C52 / AEI / 10.13039/501100011033) funded by the Spanish National Research Agency. In addition, the work of Borja Nogales, Ivan Vidal and Diego R. Lopez has partially been supported by the European H2020 5G-VINNI project (grant agreement number 815279). Finally, the authors thank Alejandro Rodríguez García for his support during the realization of this work.

References

1. Gupta, A., Jha, R. K. A Survey of 5G Network: Architecture and Emerging Technologies. *IEEE Access.* **3**, 1206-1232 (2015).
2. Yu, H., Lee, H. Jeon, H. What is 5G? Emerging 5G Mobile Services and Network Requirements. *Sustainability.* **9**, 1848 (2017).
3. Yi, B., Wang, X., Li, K., Huang, M. A comprehensive survey of network function virtualization. *Computer Networks.* **133**, 212-262 (2018).
4. 5TONIC. *An Open Research and Innovation Laboratory Focusing on 5G Technologies.* Available online: https://www.5tonic.org. last access on 15 October (2020).
5. ETSI GS NFV 002. Network Functions Virtualization: Architectural Framework. *ETSI, V1.2.1.* (2014).
6. 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 15 October (2020).
7. Silva, A. P. et al. 5GinFIRE: An end-to-end open5G vertical network function ecosystem. *Ad Hoc Networks.* **93**, 101895 (2019).
8. 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).
9. Vidal, I. et al. Multi-Site NFV Testbed for Experimentation With SUAV-Based 5G Vertical Services. *IEEE Access.* **8**, 111522-111535 (2020).
10. Nogales, B., Sanchez-Aguero, V., Vidal, I., Valera, F. Adaptable and automated small uav deployments via virtualization. *Sensors.* **18** (12), 4116 (2018).
11. Gonzalez, L. F. et al. Transport-Layer Limitations for NFV Orchestration in Resource-Constrained Aerial Networks. *Sensors.* **19** (23), 5220 (2019).
12. Sanchez-Aguero, V., Valera, F., Nogales, B., Gonzalez, L. F., Vidal, I. VENUE: Virtualized Environment for multi-UAV network emulation. *IEEE Access.* **7**, 154659-154671 (2019).
13. Kalogiros, C. et al. The potential of 5G experimentation-as-a-service paradigm for operators and vertical industries: the case of 5G-VINNI facility. *IEEE 2nd 5G World Forum (5GWF), Dresden, Germany. 347-352.* (2019).
14. Ordonez-Lucena, J., Tranoris, C., Rodrigues, J., Contreras, L. M. Cross-domain Slice Orchestration for Advanced Vertical Trials in a Multi-Vendor 5G

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

Facility. *2020 European Conference on Networks and Communications (EuCNC), Dubrovnik, Croatia. 40-45.* (2020).

15. OASIS. ISO/IEC 20922:2016 Information technology -- MQ Telemetry Transport (MQTT) v3.1.1. iso.org. *International Organization for Standardization.* (2016).
16. Mainflux. *An Open source IoT Platform Edge computing and Consulting services.* Available online: https://www.mainflux.com last access on 15 October (2020).
17. 3rd Generation Partnership Project. System architecture for the 5g system; stage 2. Technical Specification Group Services and System Aspects. *3GPP Technical Specification 23.501, version 16.2.0* .(2019).
18. *Open Source MANO Release SEVEN user-guide documentation.* Available online: https://osm.etsi.org/docs/user-guide last access on 15 October (2020).
19. OpenStack. *Open Source Software for Creating Private and Public Clouds.* Available online: https://www.openstack.org last access on 15 October (2020).
20. OpenStack release *Ocata Documentation.* Available online: https://docs.openstack.org/ocata last access on 15 October (2019).
21. OpenStack release *Ocata Installation Tutorial for Ubuntu.* Available online: https://docs.openstack.org/ocata/install-guide-ubuntu last access on 15 October (2019).
22. *Public Experiment Repository.* Available online: http://vm-images.netcom.it.uc3m.es/JoVE_2020/ last access on 15 October (2020).
23. OpenVPN. *A full-featured, open, and cost-effective VPN solution.* Available online: https://openvpn.net last access on 15 October (2020).
24. OpenVPN *How to Installation Guide.* Available online: https://openvpn.net/community-resources/how-to/#installing-openvpn last access on 15 October (2020).
25. Iptables. *A Linux kernel firewall implementation.* Available online: https://wiki.archlinux.org/index.php/Iptables last access on 15 October (2020).
26. OpenVIM. *An NFV VIM implementation contributed to the open source community project ETSI OSM.* Available online: https://osm.etsi.org/gitweb/?p=osm/openvim.git last access on 15 October (2020).
27. VMware Cloud Director. *A cloud service-delivery platform to operate and manage cloud-service businesses.* Available online: https://www.vmware.com/uk/products/cloud-director.html last access on 15 October (2020).
28. Amazon Web Services (AWS). *A broadly adopted cloud platform offering services from datacenters globally.* Available online: https://aws.amazon.com last access on 15 October (2020).
29. Microsoft Azure. *Microsoft cloud computing service for developing and managing services and applications through Microsoft-managed datacenters.* Available online: https://azure.microsoft.com/en-us last access on 15 October (2020).
30. Eclipse Foundation. *Eclipse fog05, The End-to-End Compute, Storage and Networking Virtualization solution.* Available online: https://fog05.io last access on 15 October (2020).
31. Nogales, B. et al. Automated Deployment of an Internet Protocol Telephony Service on Unmanned Aerial Vehicles Using Network Functions Virtualization. *Journal of Visualized Experiments.* (153), e60425 (2019).

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

32. RPi.bme280 0.2.3. *A Python library to drive BME280 sensor over I2C.* Available online: https://pypi.org/project/RPi.bme280/ last access on 15 October (2020).

33. Paho-mqtt 1.5.0. *A Python library implementing the MQTT client version 3.1.1.* Available online: https://pypi.org/project/paho-mqtt/ last access on 15 October (2020).

34. Public Private Partnership in Horizon 2020. Creating a Smart Ubiquitous Network for the Future Internet. *Advanced 5G Network Infrastructure for the Future Internet* .(2013).

35. Fortinet. *Deliver Network Security Digital Transformation.* Available online: https://www.fortinet.com last access on 15 October (2020).

36. Openvpn-monitor. *Open source tool to monitor the status of the service offered by an OpenVPN server.* Available online: https://github.com/furlongm/openvpn-monitor last access on 15 October (2020).

Copyright © 2021 JoVE Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License