González, Luis F.; Vidal, Iván; Valera, Francisco; Sánchez-Agüero, Víctor. (2021) A Comparative Study of Virtual Infrastructure Management Solutions for UAV Networks. *Proceedings of the 7th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (Dronet'21), June 24, 2021, Virtual, WI, USA. ACM, New York, USA*.

DOI: https://doi.org/10.1145/3469259.3470486

A Comparative Study of Virtual Infrastructure Management Solutions for UAV Networks

Luis F. Gonzalez
luisfgon@it.uc3m.es
Universidad Carlos III de Madrid
Leganés, Madrid, Spain

Ivan Vidal
Francisco Valera
[ividal,fvalera]@it.uc3m.es
Universidad Carlos III de Madrid
Leganés, Madrid, Spain

Victor Sanchez-Aguero
victor.sanchez@imdea.org
IMDEA Networks Institute
Universidad Carlos III de Madrid
Leganés, Madrid, Spain

ABSTRACT

The promising combination of Unmanned Aerial Vehicles (UAVs) with network virtualisation technologies has positively shown many advantages enabling the deployment of communication services over aerial networks, that is, networks conformed by a set of interconnected UAVs. However, this synergy may certainly involve diverse challenges that must be carefully considered. In this respect, this paper compares some of the most common virtual infrastructure management solutions that could potentially be used to deal with virtualised payloads over aerial networks, identifying their main strength and limitations. The paper also presents a preliminary exploration on the utilisation of the Kubernetes virtual infrastructure management platform to support value-added services over UAV networks, showing off its potential as a suitable platform to this purpose.

CCS CONCEPTS

• **Networks** → *Ad hoc networks*; **Network management**.

KEYWORDS

UAV networks, VIM, OpenStack, K8s, Fog05

ACM Reference Format:
Luis F. Gonzalez, Ivan Vidal, Francisco Valera, and Victor Sanchez-Aguero. 2021. A Comparative Study of Virtual Infrastructure Management Solutions for UAV Networks. In *The 7th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet) (Dronet '21), June 24, 2021, Virtual, WI, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3469259.3470486

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have been regarded as one of the technologies that will shape the future vehicular technology thanks to their almost limitless mobility. Taking advantage of their increased ubiquity, these devices may implement new functionalities that other vehicles would struggle to provide, either by using their own default systems or by on-boarding additional equipment. Some well-known use cases of UAV technology can be seen in examples like aerial photography, precision agriculture and military surveillance applications [4]. However, UAVs also have the potential to act as service carriers for certain applications that cannot be provided by UAVs themselves, but rather their payload, as these aircrafts are able to on-board certain types of communication equipment to create networks between several units by using wireless technologies, e.g., using WiFi in order to establish an ad-hoc network between aircrafts or create a WiFi Access Point to allow other UAVs and/or user-end equipment to connect to the network.

Unfortunately, most networking services require specialised equipment for their proper functionality. Usually, this equipment is expensive to deploy and maintain, and also their size and weight could be an issue when on-boarding them in some situations. Therefore, there has been an increasing interest in the combination of UAV technology together with Network Function Virtualisation (NFV) in order to allow the deployment of network functionalities over aerial networks. NFV is a paradigm that aims at the softwarisation of physical network functions, allowing their deployment into generic hardware equipment with enough computational, storage and networking capabilities. Using these processes, service providers are not forced to rely on specialised hardware equipment for its provision, reducing development and maintenance costs as well as easing the development and deployment of network services. Moreover, NFV could assist telco providers and vertical industries since it would transform UAVs into programmable infrastructures that could flexibly support multiple services. A prominent example of this combination can be seen in [6], where authors present an airborne computing platform design using NFV running on the payloads of the aircrafts in order to enhance UAV functionalities.

Although it is true that this prospect has a lot of potential, current NFV management solutions might not be suited to deal with issues that resource-constrained environments such as the ones that can be included on UAVs can introduce (both in terms of resource availability and communication availability), which we analysed in detail in our previous work [8]. In this work, we compare some of the most prominent open-source virtual infrastructure management solutions in order to see how they may deal with the main challenges that aerial networks may rise. Afterwards, we accomplish the practical deployment of a virtual infrastructure management platform in a test environment based on Kubernetes, providing initial insights of its effectiveness to support the flexible deployment of services on UAV networks.

2 ANALYSIS OF EXISTING SOLUTIONS

As it has been introduced, the main objective pursued by the combination of both NFV and UAV technologies is to support the agile

Dronet '21, June 24, 2021, Virtual, WI, USA

ACM ISBN 978-1-4503-8599-2/21/06…$15.00
https://doi.org/10.1145/3469259.3470486

and flexible deployment of network functions over a delimited geographic area. Still, it is important to properly address the potential shortcomings of this approach. In our previous work [8], we already identified and analysed the most relevant challenges related with the orchestration of NFV services in resource-constrained devices: *(i)* the limited mission lifetime of aerial nodes, since UAVs will need to be frequently replaced due to battery constraints; *(ii)* the intermittent availability of control communications, as battery replacements and wireless disruptions (e.g., fade-away or information loss) can affect the links established between aircrafts, which may potentially limit management communications towards specific UAVs; *(iii)* the limited payload capacity of aerial nodes, due to the typical size and weight restrictions of UAVs; *(iv)* low performance of transport-layer protocols for control communications, since TCP-oriented solutions might not behave properly in environments with constant communication breakdowns; and *(v)* lack of enhanced policies for the placement of softwarised services over aerial networks, since most management solutions do not provide service distribution mechanisms based on aerial networks characteristics.

Any platform aiming to deploy and manage Virtualised Network Functions (VNFs) over UAV networks should ideally take into account these issues in order to increase their performance and effectiveness. However, this prospect has not been fully explored yet, and most of the available solutions are oriented towards cloud environments where powerful servers and full connectivity through Gb/s Ethernet links are prevalent. In consequence, the direct utilisation of these solutions might introduce some issues that can negatively impact aerial networks since they lack the resources available at cloud environments. In addition, each VNF platform has its own advantages that, after a careful analysis, could be the first stepping stone to develop a fully specialised manager in the future for aerial platforms (and in general resource constrain environments).

With this objective in mind, this section identifies a set of well-known open-source solutions that could be considered to deploy virtual functions on resource-constrained UAV platforms. To support our analysis, we define a reference scenario where a set of representative services is deployed over a UAV network. For each virtual infrastructure management solution, we identify their main strengths and weaknesses when implementing the reference scenario. Table 1 presents a summary of our analysis.

2.1 Reference scenario

Figure 1 illustrates our reference scenario, which uses a number of UAVs and a Ground Control Station (GCS) to deploy two independent services: an emergency call service, where users are able to call for instance for rescues in remote areas; and a wildlife monitoring service, where a UAV with a camera is able to film certain species from afar.

The emergency service includes a WiFi access point, which enables users to connect their mobile phones and establish emergency calls, and an IP telephony server. The wildlife monitoring service includes a recording function that can also be programmed to analyse the captured images and trigger alerts. It also encompasses a database, which can store the recorded videos for further analysis. Due to distance limitations of wireless communications, an intermediary UAV executes a forwarding function, supporting the exchange of traffic with the GCS. In the scenario, the components of both services are deployed as virtual functions on the UAVs and the GCS. To this purpose, the scenario includes a virtual infrastructure management solution, capable of adapting the UAVs functionality to support multiple mission objectives. In the following, we present different alternatives that could potentially be used to implement such solution: OpenStack, Kubernetes, k3s, and fog05.

2.2 OpenStack

OpenStack is an open-source platform that aims to control large pools of computation, storage and networking resources in order to manage distributed computing infrastructures [14]. OpenStack can be used as a Virtualised Infrastructure Manager (VIM) so as to orchestrate an NFV Infrastructure, assigning the necessary resources to the softwarised functions. This action is performed by a set of services in charge of different functionalities such as the control of computing systems (*Nova*), registration and management of Virtual Machine (VM) images (*Glance*), or networking management (*Neutron*). There are a handful of services available for a variety of purposes, but not all of them are necessary for a fully-functioning OpenStack platform, providing this solution with a high degree of flexibility that could be beneficial for resource-constrained environments, as some of its services could be discarded in favour of establishing a more lightweight solution able to work in nodes with smaller computing capabilities.

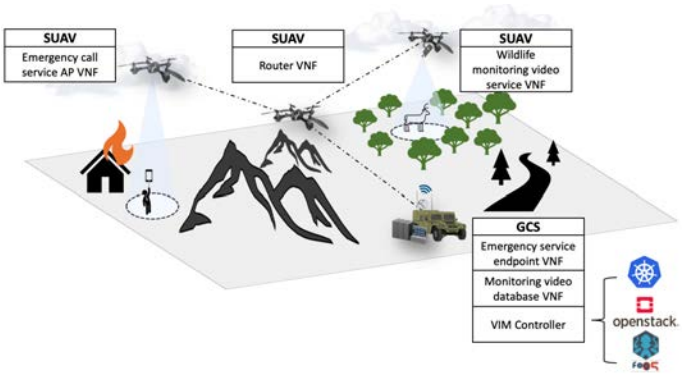

Figure 1: Reference scenario with a UAV network offering two virtualised network services

However, taking into account the limited lifetime and capacity of aerial nodes, OpenStack is still considered to be a resource-demanding solution, (its minimal installation requires at least 2 CPUs, 8GBs of RAM and 100 GB of storage), which could be an obstacle for its deployment over UAV networks if payloads used by the aircrafts are not able to meet its requirements. It must be noted that OpenStack is able to support container virtualisation. Containers are lightweight virtualisation units because they require less resources for their execution and maintenance. This approach towards function virtualisation could be beneficial to save battery lifetime and computational resources.

OpenStack follows a centric model approach where a node, the "controller", generates most of the management traffic that needs

to be sent to the rest of the compute nodes of the infrastructure (i.e., the UAVs), for example during a service instantiation, after modifying the resource allocation for a virtual function, etc. In environments where communications might be unavailable at certain moments in time, like UAV networks, relying on a central entity can introduce some problems: when an aircraft goes offline, other parts of the network might be affected as well. This situation occurs if a set of UAVs rely on the missing aircraft to connect with the controller, which in turn could induce OpenStack to consider the rest of nodes offline as well (when in reality, it is just a temporary issue). Our previous analysis in [7] suggests that OpenStack is able to deal with this kind of cutoffs as long as they have a limited duration (not exceeding 15 minutes), since the VIM will neither discard the compute nodes nor schedule new VNFs on other compute nodes, resuming management operations once they are reachable again. Hence, OpenStack is able to deal with these disconnections, although not in an optimal way as it leaves some aircrafts without management for some time before connectivity can be resumed.

OpenStack allows the flexible development and implementation of new plug-ins, so new services that could provide novel functionalities to the platform while taking advantage UAVs mobility can be developed, such as a service to instantiate VMs/containers based on parameters like location or battery lifetime.

Although it may seem a trivial matter at first, the Neutron service from OpenStack provides all the required networking capabilities necessary to establish communications at the link layer or/and the IP layer between VNFs, allowing the attachment of new interfaces to VMs/containers as well as creating and managing networks and subnetworks for VNF interconnection. This is an essential functionality for almost all networking services, including the one shown in Figure 1, since it is necessary to forward packets from/to different VNFs, and not all VIM solutions provide an implementation by default, or they are severely restricted.

Despite its shortcomings, OpenStack has been regarded as one viable VIM solution in resource-constrained environments, which motivated our research exploring its usage in realistic Small UAV (SUAV) scenarios. In particular, in [5] we propose a smart farming use case by combining cloud technologies and a UAV network. In [3], we studied the utilisation of OpenStack to support the execution of a functional telephony service on a remote area, using a cloud platform of UAVs.

2.3 Kubernetes

Kubernetes [2] (also known as K8s) is a portable, extensible open-source system used for the automated deployment, management and scaling of containerised applications and services. It works exclusively with container technology, allowing the deployment of applications based on containers like Docker [1]. These containers could be used to execute VNFs, allowing them to take advantage of the benefits of containers (lower resources required for execution and maintenance) on aerial networks.

K8s follows a very similar model to OpenStack. Every infrastructure (called a cluster) is divided into nodes, with two different types according to their functionality: computation nodes (workers), which execute the applications workload, and a control plane (controller), in charge of managing the workers and their applications. K8s is predominantly a cloud-oriented solution. Therefore, just like OpenStack, it is considered computationally resource-demanding, requiring at least 2GB of RAM and 2 CPUs per node, making its implementation difficult in environments with limited computational resources. Moreover, K8s does not natively have a full networking implementation for container communication, nor it provides native IP addressing for containers. Instead, it relies on third party plug-in solutions to implement such functionality by defining a series of requirements that every implementation must follow. These limitations can potentially harm the development and deployment of network services over UAV networks, requiring an extensive analysis of existing solutions to determine if they could provide the networking functionalities required in our reference scenario, while respecting the impositions of K8s.

However, K8s provides a set of advantages that aids the management of container workloads in comparison with other virtualisation platforms. Its setup and node aggregation are very easy to perform, as it can automatically configure every new node based on the configurations defined at the controller, and it can also spread modifications across a full cluster by simply modifying its configuration in the controller. On the other hand, K8s uses a set of tools that aims at decreasing service downtime through a set of management functionalities that could be useful in aerial environments, particularly: automated roll-outs and rollbacks, as K8s allows to modify container applications on real time, updating each application with newer versions by replacing old containers with new ones. Moreover, K8s allows to return to a previous state or version if needed (e.g., a failure with a new update). An aerial infrastructure could benefit from this characteristic to dynamically modify services running at the aircrafts (e.g., assign more container replicas when demand increases); Self-healing properties, since when a container fails, K8s will try to replace it (either in the original node or in a different one) in order to bring the application up again as soon as possible. This property could be useful in a UAV network, since aircrafts will constantly have to replace batteries, bringing down containers that build up services. Having the ability to automatically redistribute those containers into new/other aircrafts can greatly improve service provisioning.

As stated before, K8s follows a model centred around a controller node. In consequence, it shares the same core issue with OpenStack: loosing communications with one aircraft might affect a portion of the network, leaving a zone temporarily without orchestration. Fortunately, K8s provides flexibility to deal with intermittently available nodes: once a controller is not able to reach a node, it will be marked as absent. After a timer of 5 minutes (set by default) it will schedule all of its containers into other candidates based on either pre-determined instructions or their affinity (i.e., based on their computational resources available), scheduling the termination of the containers in the absent node. This timer can be modified to suit the infrastructure needs, for example defining a timer that may vary based on the status of an aerial network, or disabling it completely. Hence, K8s can deal with absent nodes better than OpenStack, since it is able to automatically migrate failing containers into new units (unlike OpenStack that cannot do this). Still, its mechanisms have their own limitations due to its centralised model, which can leave some nodes temporarily without management.

2.4 Other solutions: K3s & Fog05

Due to the high computational requirements of the aforementioned orchestration solutions, other ones have been proposed in order to reduce their implementation footprint for its usage over resource constrained devices, which would allow UAVs to on board payloads with reduced size and weight. The most prominent efforts for this task are K3s [11] and Fog05[13].

On the one hand, K3s is an open-source lightweight fully compliant Kubernetes distribution with a footprint of around 100 MB, sharing most of the advantages and disadvantages of K8s, while having a considerably reduced size, which would greatly benefit UAV networks. On the other hand, the open-source Eclipse Fog05 fog infrastructure manager aims at providing a decentralised infrastructure for provisioning and managing compute, storage, communication and I/O resources available anywhere across the network. With a very reduced footprint and wire overhead, Fog05 is able to spread the management information throughput the network without relying on a centralised entity, so the management tasks could be "offloaded" to the compute nodes of a portion of an aerial network if it was affected by a temporary link disruption, allowing to always have some form of management over an aerial network (which is impossible to achieve using centralised approaches).

These two solutions are still under development, so their implementation over aerial networks is scarce. Nonetheless, their characteristics allows them to be regarded as potentially outstanding platforms to orchestrate container workloads in aerial networks once they have further matured (specially in the case of Fog05 due to its recent conception).

3 APPLICATION OF KUBERNETES IN UAV NETWORKS

Even though K8s has been raising as one of the most popular cloud-management solutions for container workloads management, there have not been significant advances on its deployment over aerial networks, even though it could provide several advantages in comparison to other orchestration solutions. In consequence, this paper will introduce a first exploration of it appropriateness to support the automated deployment of services in our reference scenario of Figure 1.

For this exploratory work, we have deployed a K8s cluster. The cluster has been conformed by two Mini-iTx machines with 4 available CPUs and 8 GBs of RAM with Ubuntu 20.04, which were used as controller and "relay" nodes. A single board computer, Raspberry Pi 3B Model (RPi) with 4 CPU cores and 4 GBs of RAM running Ubuntu 20.04, was also included in the K8s cluster since due to its size and weight characteristics it is the typical device that we on-board on a small size UAVs. The installation of K8s has been done using kubeadm [15], due to its compliance with the Kubernetes conformance tests, amd64 architectures (used for the Mini-ITx machines) and arm architectures (RPi).

As commented, K8s does not provide a complete networking solution. It only defines a set of guidelines to communicate containers (from this point on they will be named *pods* to use K8s terminology) within a cluster. Particularly, all pods in a cluster must be able to directly reach every other pod in the cluster. Hence, it is essential to select an appropriate set of K8s plugins that, being compatible with the infrastructure, still provide the necessary functionalities to establish independent point-to-point links between *pods* (some of the most prominent networking solutions for K8s, e.g., Flannel [10]), do not support the creation of such links). In our case, as a first step to support a complete networking solution in K8s, we have implemented the open-source plugin Calico [18] network policy enforcer.

Calico has all the tools required to provide addressing and network connectivity between pods, as well as to enforce network policies regarding pod communications (e.g., enabling or preventing specific data flows). The latter is essential to prevent unauthorised entities from interfering legitimate communications on pod-to-pod links. Internally, Calico supports the exchange of traffic among pods using IP tunnels (e.g., VxLAN [9]). This is transparent to pods, which are simply provided with a network interface and an IP address. Unfortunately, IP tunnelling mechanisms introduce overhead, requiring further extra traffic processing that may introduce delays.

Despite its benefits, Calico is not enough to establish the point-to-point links shown in our reference scenario. This is because it relies on proxying techniques to guarantee the "abstraction" of pod communications. That is, Calico uses a software agent at every compute node to route traffic towards other compute nodes. Consequently, pods will be able to communicate at the IP layer, but they will not be able to establish a direct communication at the link-layer, which is required to establish a point-to-point connections.

The solution that has been followed to address this issue is the creation of point-to-point links among pods using VxLANs, and has been deployed using a VxLAN controller for K8s [16]. With this approach, a VxLAN interface is created at each pod on the point-to-point link. Traffic sent through one of the VxLAN interfaces is tunnelled and sent through the main interface of the pod. This way, Calico routes this traffic towards the other pod on the link, which receives the encapsulated traffic through its regular interface. After decapsulation, it is delivered to the pod through the VxLAN interface, this way creating the abstraction of a point-to-point link among the pods.

Of course, using additional VxLANs over Calico will again increase the communication overhead, thus reducing the available bandwidth. Other solutions have tried to support enhanced communication models in K8s, with the objective of closing the gap between K8s and NFV platforms. Network Service Mesh (NSM) [12] is an open-source solution that builds on top of an existing networking platform (e.g., Flannel or Calico) and establishes "dedicated virtual links" between pods, assisting with complex networking scenarios that native plug-ins are not able to implement. The open-source OVN4NFV K8s plug-in [17] allows the dynamic creation of multiple types of sub-networks (overlay, VxLAN, etc.) for cloud workloads. These could potentially offer networking solutions for K8s over aerial networks. Nevertheless, due to their complexity, lack of maturity, and their cloud-oriented approach, a careful study is still needed to assess their effectiveness on intermittently available UAV environments.

In order to validate this design, the experiments were specified to emulate the wildlife monitoring service depicted in the reference scenario. A VNF in the UAV sends an HD live-feed video into another VNF present in the GCS, where a playback tool will be used to watch the live-stream. In the experiment this has been

Table 1: Platforms comparison for dealing with virtualised application management in aerial networks

	OpenStack	Kubernetes/K8s	K3s	Fog05
Limited lifetime of compute nodes	Instance recovery provided by an external plugin. No default migration mechanism available.	Provides a native solution for migrating failing containers. Programmable behaviour relying on computational resources.		Currently allows the migration of some types of containers manually. No default migration for failing containers.
Intermittent availability of control communications	Centralised model. Link disruptions can leave some nodes without management.	Centralised model. Link disruptions can leave several nodes without management.		Distributed model. Management can be available across an aerial network despite communication intermittencies.
Limited capacity of compute nodes	Resource demanding solution. Allows the instantiation of both VM and container based applications.	Resource demanding solution, but lighter than OpenStack. Uses container technology for deploying virtualised workloads.	Reduced footprint compared with K8s (100MB). Uses container technology for deploying virtualised workloads.	Reduced footprint and wire overhead. Allows the instantiation of both VM and container based applications.
Networking provision and implementation	Provides a full networking solution through Neutron.	Relies on one external plugin to provide networking. Imposes a set of restrictions on its behaviour.	Relies on a single external plugin to provide networking (Flannel). Limited variety of external plugins available.	Uses the Zenoh protocol for data discovery, distribution and network setup. Provides full networking for applications in an infrastructure.
Enhanced policies for VNF placements	Optimises service instantiation based only in computational resources. Does not take into account aerial characteristics.	Flexible platform to determine service placement through node affinity. Does not take into account aerial characteristics.		In the future, the platform could take into account characteristics of aerial networks. Still under development.

emulated with Iperf3, establishing a traffic flow between the RPi and the controller in a set of cases: direct communication without VxLAN, direct communication with VxLAN, and communication though different VxLANs using a relay node for traffic routing. For each scenario, two traffic flows, one for each battery of tests, were established: one TCP flow at the maximum possible rate to obtain the available bandwidth on the channel, and one UDP flow with 5Mbps in order to emulate the HD video generation at 720p (as 5Mbps is the average transmission rate used for HD video at 720p). Each one of these flows lasted 90s, and were repeated 20 times for every scenario to obtain measurements regarding their throughput (in Mbps). The objective of this validation is to establish the impact of using a "double VxLAN" model to communicate pods against a "single VxLAN" approach, as well as to determine the viability of providing an HD video service over the aerial network.

The average results for the TCP tests on each case were the following: 39.06 Mbps with no VxLAN, 35.74 Mbps using a single VxLAN and 34.77 Mbps using a third node as traffic relay. These showcase that as expected, direct communication between pods provide higher throughput when no additional VxLAN is present due to the additional overhead data processing that nodes introduce. However, this performance degradation (around the 10% mark) is not enough to severely limit the available bandwidth over the link, making its usage a viable solution in aerial networks since applications will still have sufficient bandwidth for data communications (around 35 Mbps in our case), which is enough to place several of them over a shared link. Nevertheless, it is still a sub-optimal approach, since some bandwidth is lost due to the additional header processing, and realistic scenarios would have a more limited bandwidth since it is a shared medium. Further analysis should be performed in a future study.

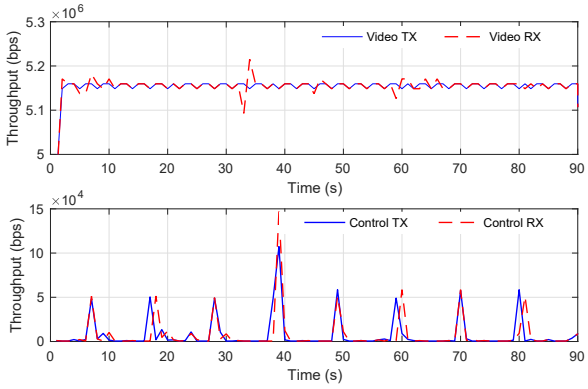

Figure 2: Video emulation throughput: data & control traffic

Afterwards, for the second battery of tests, i.e., tests with UDP flows, all scenarios averaged 5 Mbps, because there is enough channel availability to distribute an HD video from the RPi to the controller through an intermediate relay node and as expected no bandwidth loss was shown during the tests. As it can be seen in Fig 2, corresponding to one of the test runs using an intermediate node, video distribution was constant throughout the whole test. The picture also shows the management traffic exchanged between the K8s controller and the compute node. We can observe that management traffic is exchanged in small bursts over time that, overall, do not produce a significant impact in the network due to their reduced bandwidth requirements.

4 CONCLUSION AND FUTURE WORK

In this paper we made a comparison of several virtual infrastructure management solutions, analysing their strengths and limitations to

support the automated deployment of virtual functions and applications on UAV networks. With this analysis we established that K8s has some advantages over OpenStack in order to orchestrate VNFs over aerial networks. Furthermore, we have done a preliminary practical evaluation of K8s, using existing open-source plugins to support the abstraction of point-to-point among VNFs. This is a necessary condition to enable its utilisation in the deployment of NFV services on UAVs.

Our future work will include an in-depth analysis of the different networking solutions for K8s and their applicability in our reference scenario. We will also study other VIM solutions, given their potential to distribute management functionalities and deal with intermittently available nodes.

ACKNOWLEDGMENTS

This article has been supported by the H2020 Labyrinth project (grant agreement 861696) and the TRUE5G project funded by the Spanish National Research Agency (PID2019-108713RB-C52/AEI/10.13039/501100011033)

REFERENCES

[1] Docker. 2021. *Docker: Empowering App Development for Developers.* Retrieved June 8, 2021 from https://www.docker.com/

[2] Brendan Burns et al. 2019. *Kubernetes: up and running: dive into the future of infrastructure.* O'Reilly Media.

[3] Borja Nogales et al. 2019. Automated deployment of an Internet protocol telephony service on unmanned aerial vehicles using network functions virtualization. *JoVE (Journal of Visualized Experiments)* 153 (2019), e60425.

[4] Haichao Wang et al. 2018. Network-connected UAV communications: Potentials and challenges. *China Communications* 15, 12 (2018), 111–121. https://doi.org/10.12676/j.cc.2018.12.009

[5] Iván Vidal et al. 2020. A Multi-Site NFV Testbed for Experimentation With SUAV-Based 5G Vertical Services. *IEEE Access* 8 (2020), 111522–111535.

[6] Kejie Lu et al. 2019. Toward uav-based airborne computing. *IEEE Wireless Communications* 26, 6 (2019), 172–179.

[7] Luis F. Gonzalez et al. 2019. NFV orchestration on intermittently available SUAV platforms: challenges and hurdles. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* IEEE, 301–306.

[8] Luis F. Gonzalez et al. 2019. Transport-layer limitations for NFV orchestration in resource-constrained aerial networks. *Sensors* 19, 23 (2019), 5220.

[9] Mallik Mahalingam et al. 2014. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348. https://doi.org/10.17487/RFC7348

[10] Flannel. 2020. *Flannel.* Retrieved June 8, 2021 from https://github.com/flannel-io/flannel

[11] Cloud Native Computing Foundation. 2020. *K3s: Lightweight Kubernetes, the certified Kubernetes distribution built for IoT and Edge computing.* Retrieved June 8, 2021 from https://k3s.io

[12] Cloud Native Computing Foundation. 2020. *Network Service Mesh: The Hybrid/Multi-cloud IP Service Mesh.* Retrieved June 8, 2021 from https://networkservicemesh.io

[13] Eclipse Foundation. 2019. *Eclipse fog05.* Retrieved June 8, 2021 from https://fog05.io

[14] Open Infrastructure Foundation. 2021. *Openstack: The Most Widely Deployed Open Source Cloud Software in the World.* Retrieved June 8, 2021 from https://www.openstack.org

[15] The Linux Foundation. 2021. *Creating a cluster with kubeadm.* Retrieved June 8, 2021 from https://kubernetes.io

[16] Openvnf. 2019. Kube-vxlan-controller. Retrieved June 8, 2021 from https://github.com/openvnf/kube-vxlan-controller

[17] opnfv. 2020. OVN4NFV K8s Plugin - Network controller. Retrieved June 8, 2021 from https://github.com/opnfv/ovn4nfv-k8s-plugin

[18] Tigera. 2021. *Project Calico: What is Calico?* Retrieved June 8, 2021 from https://www.projectcalico.org