FlyNet: A Platform to Support Scientific Workflows from the Edge to the Core for UAV Applications

Eric Lyons, Hakan Saplakoglu, Michael Zink
{elyons|hsaplakoglu|mzink}@umass.edu
University of Massachusetts Amherst

Komal Thareja, Anirban Mandal
{kthare10|anirban}@renci.org
RENCI, University of North Carolina at Chapel Hill

Chengyi Qu, Songjie Wang, Prasad Calyam
cqy78@mail.missouri.edu,{wangso|calyamp}@missouri.edu
University of Missouri Columbia

George Papadimitriou, Ryan Tanaka, Ewa Deelman
{georgpap|tanaka|deelman}@isi.edu
ISI, University of Southern California

ABSTRACT

Many Internet of Things (IoT) applications require compute resources that cannot be provided by the devices themselves. At the same time, processing of the data generated by IoT devices often has to be performed in real- or near real-time. Examples of such scenarios are autonomous vehicles in the form of cars and drones where the processing of observational data (e.g., video feeds) needs to be performed expeditiously to allow for safe operation. To support the computational needs and timeliness requirements of such applications it is essential to include suitable edge resources to execute these applications. In this paper, we present our FlyNet architecture which has the goal to provide a new platform to support workflows that include applications executing at the network edge, at the computing core, and leverage deeply programmable networks. We discuss the challenges associated with provisioning such networking and compute infrastructure on demand, tailored to IoT application workflows. We describe a strategy to leverage the end-to-end integrated infrastructure that covers all points in the spectrum of response latency for application processing. We present our prototype implementation of the architecture and evaluate its performance for the case of drone video analytics workflows with varying computational requirements.

CCS CONCEPTS

• **Computer systems organization → Sensor networks**; **Cloud computing**.

UCC'21, December 6–9, 2021, Leicester, United Kingdom

ACM ISBN 978-1-4503-8564-0/21/12…$15.00
https://doi.org/10.1145/3468737.3494098

1 INTRODUCTION

In the past decade, we have seen an unprecedented increase in data driven research leading to data-intensive applications, which are one of the pillars of computational science. This increase in data driven research is caused by a series of factors. First, through advances in sensor technology, instrumentation, and the proliferation of sensor-rich devices, a vast variety of data sets have been generated, with a significant portion of them being publicly available. Second, national and international research facilities like the Large Hadron Collider (LHC) [34], the Laser Interferometer Gravitational-Wave Observatory (LIGO) [18], and networks of telescopes like the Event Horizon Telescope (EHT) [5], continue to drive fundamental research.

One class of data-intensive applications, which has recently received significant attention are applications that rely on autonomous (e.g. self-driving cars) and semi-autonomous systems (e.g. drones) that are dynamic, data-driven and data-intensive — the so called Dynamic Data Driven Application Systems (DDDAS) [8]. In this paper, we focus on supporting applications that provide safe and efficient use of Unpiloted Aerial Vehicles (UAV), e.g. drones for information or payload delivery. We have selected these applications since UAVs have become both the subject of research and are also used by researchers to provide scientific observations and deliver time-critical payloads.

UAV applications pose several challenges. They often have diverse and dynamic requirements for network and compute resources. Many UAV applications, e.g. the application described in Section 2, require optimized and adaptive data movement from data repositories and instruments to an end-to-end edge-to-core computing infrastructure that allows for the processing of the data at the edge and on core computing resources (which may also host key data sets). Some UAV applications demand a range of processing capabilities with stringent latency requirements under several constraints (e.g. energy, etc.). In this case, the required processing could be "spread out" over a spectrum of available resources, from the edge (lightweight and immediate analysis) to the core (heavyweight analysis that takes significant time). Additionally, the applications are sensitive to network latency for data movement and for sending control information to the edge devices. These applications also require complex workflows to collect, process, analyze, and store the generated data. Such workflows utilize an arbitrary set of networking, compute, and storage resources and can perform dynamic computational offloading in the edge-to-core continuum of resources. Provisioning

resources for these workflows and planning and executing them is a complex task in itself, which requires the orchestration of resources in a timely, robust, and efficient manner.

In this paper, we present a new platform, FlyNet, that has the goal to *support data-driven scientific research* and addresses some of the above challenges. FlyNet supports UAV applications from the *network edge* to the *compute core* with *programmable networks*. It provides researchers with unprecedented flexibility for tailoring workflows to the resource requirements of their DDDAS IoT applications.

FlyNet builds on our previous work [20] that demonstrated how a workflow management system and a dynamic resource provisioning entity can work together to support data-driven science applications. In particular, we have shown how we can dynamically provision network links to deliver weather radar data to dynamically provisioned computational resources. This system [20, 21] was put in place to support the CASA [25] severe weather forecast and warning system in the Dallas Fort Worth area.

The work presented makes the following contributions:

- **An overarching architecture for supporting UAV applications.** FlyNet's design supports data-driven applications that require features like (ultra) low latency and in-network processing by enabling the provisioning of compute resources at the edge, within the network (in-network), and core computing resources. This enables a new generation of science applications through end-to-end resource integration.
- **Provisioning and Deployment Services.** We implement a set of services that instantiate that architecture by providing mechanisms for automated resource provisioning and service deployment.
- **Novel Application.** We present the design and implementation of a novel drone application that performs video analytics, offloading the analytics to the nearest edge server in-flight and leverages the FlyNet architecture.
- **Proof-of-concept deployment.** We present a proof-of-concept deployment of the FlyNet architecture for the drone video analytics application in actual testbeds.
- **Evaluation.** We present results from a functional evaluation of the drone video analytics application running on FlyNet.

There are a number of novel aspects of Flynet. Unlike existing, proprietary, industry edge-to-cloud architectures like Intel's FlexRAN [1], which provides software support for applications to simplify the use of the radio access and edge resources, FlyNet provides an open approach, which is agnostic of the target hardware architecture and is compatible with existing research testbeds. Additionally, FlyNet, as demonstrated through our target application, enables a dynamic, measurement-based resource selection approach with the goal to guarantee certain metrics (e.g., latency, throughput) to the application. Unlike existing approaches for computational offloading [43], which mainly focus on the usage of drones as edge compute resources for IoT devices, Flynet addresses applications where drones themselves are data producers (IoT devices) that can benefit from edge computing resources.

The paper is organized as follows. Section 2 presents the unique challenges for UAV applications and Section 3 discusses the architectural design for FlyNet, which addresses these challenges. Section 4 presents the individual components of the FlyNet architecture, which we designed, implemented and used for the evaluation. Section 5 presents a multi-faceted evaluation of our system with an exemplar UAV video analytics application employing different architectural choices and network conditions. We present related work in Section 6 and conclude the paper in Section 7.

2 MOTIVATING APPLICATION: DRONE VIDEO ANALYTICS

Before we describe the overall FlyNet system, we present our target UAV science application to illustrate the challenges such applications pose to the available network and compute infrastructure.

We consider a UAV video analytics application, where a drone is flying at low altitude over a city simultaneously recording video footage of the area. The video footage can be used for a variety of applications like traffic jam detection, recognition of humans, or the identification of storm damage. Real-time video analytics can assist in such scenarios, helping guide the drone or helping users (e.g., emergency responders) react quickly to events (since automated detections can be much faster than human analysis). Figure 1 shows a computer vision application that provides situational awareness in real-time (e.g., recognizing and counting humans/vehicles) in order to support disaster response or smart city traffic management. In particular, we focus on drones equipped with high-resolution cameras, that are needed for operations such as disaster management, traffic management, remote sensing, border surveillance, and smart farming.

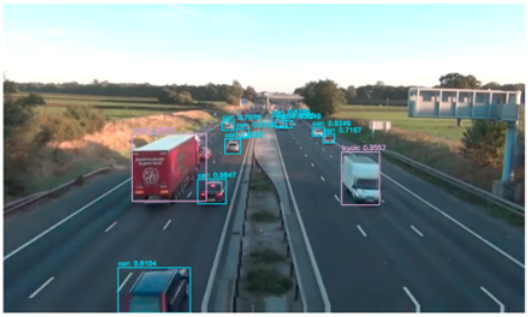

Figure 1: Computer vision application for analysis and classification of moving objects (e.g., cars, trucks, etc.) observed by a drone.

Our target drone video analytics scenario is shown in Figure 2, where a drone communicates via Wi-Fi, 4G/LTE or 5G networks (indicated by circles) with candidate edge processing nodes (green and yellow boxes) that are distributed across the region. The diagonal green line indicates the path of the drone and the black arrow indicates its current position. During such a flight a drone will be entering and leaving the coverage area of cell towers and, in some cases, might be able to communicate with more than one. In addition, several candidate edge nodes might be available for the execution of the video analytics application.

To use such drones effectively, several challenges need to be addressed in terms of provisioning available edge/cloud computation resources to satisfy the needs of drone video analytics applications [32, 42]. First of all, limited compute and power resources on board a drone require that processing be offloaded to more powerful compute resources like edge or core cloud resources. Second, to keep

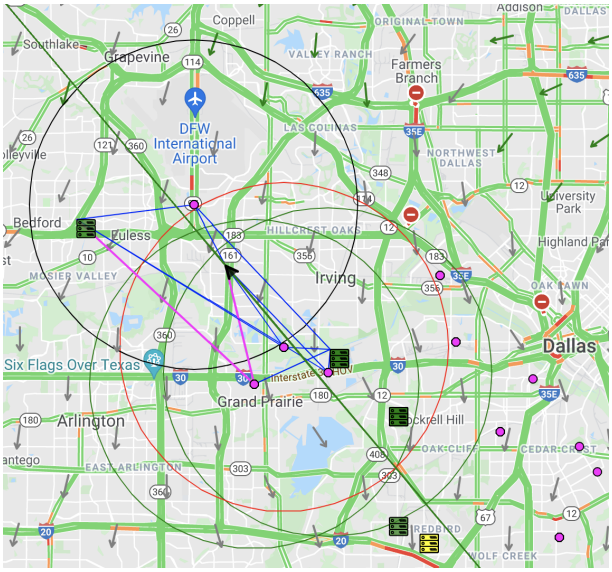

Figure 2: FlyNet scenario for drone video analytics. The circles in this image show the coverage area of cell towers and the boxes show base stations (edge nodes). The green lines indicate flight paths for drones.

latency at a minimum, the edge compute resources should be close to the physical location of the drone. Since a drone can change its location significantly over the lifetime of an application the computation offloaded to the edge might have to be migrated between edge serves in order to keep the latency as low a possible.

To meet these exemplar challenges, we designed FlyNet as an Mobile Edge Cloud (MEC)-based system [36] that provides: (i) dynamic edge/cloud computation offloading, and (ii) mobility management [14] across the edge-to-core latency spectrum.

3 FLYNET SYSTEM ARCHITECTURE

In this section, we present the FlyNet architecture, as shown in Figure 3, which is needed to compose an end-to-end (edge-to-core) infrastructure capable of supporting UAV applications.

3.1 Edge-to-Core Infrastructure

The edge-to-core infrastructure depicted at the bottom of Figure 3 covers all points in the spectrum of response latency for application processing - the *latency spectrum*. While some processing needs to be performed on the devices and the network edge, some computations need to be performed in-network and some can be offloaded to core computing resources "far" from the edge devices.

There are several categories in this latency spectrum - *edge devices, edge servers, in-network,* and *core computing*. While edge devices provide minimum latency for response times, they have limited computational capabilities and/or power constraints. Thus, on-board resources are often not sufficient to support the UAV application processing needs. Edge servers or nodes that comprise an edge computing infrastructure have more computational power and fast turnaround time, but support only limited scales of computation (e.g. they might be able to run very lightweight algorithms, but not data and compute intensive codes like deep learning models).

As the latency on the spectrum increases, one can envision processing packets and turning them around using in-network computing capabilities (either compute resources or specialized programmable hardware deployed in the network core). This will reduce latency compared to cases where data has to be transmitted all the way to the computing core. For UAV data processing that needs substantially more computational resources (e.g. GPUs for training machine learning models for object detection), data needs to travel all the way to core cloud resources. This incurs the maximum latency with the benefit that high processing power can be utilized.

The current FlyNet edge-to-core infrastructure consists of edge devices (i.e., drones capable of obtaining high-resolution video data), edge servers - small and large - capable of edge processing and hosting data sets, edge servers connected to cloud platforms (e.g. Chameleon [17]), high-performance traditional Layer2 networks, and core cloud resources (ExoGENI [6] and Chameleon) with substantial computational power.

3.2 FlyNet Services: Resource Provisioning

To realize the overall FlyNet architecture, we use a network-centric platform called Mobius [20, 23, 24, 26] with support for provisioning programmable cyberinfrastructure comprised of ExoGENI [6] and Chameleon [10, 17] cloud testbeds. Mobius makes it easier for applications to provision and manage the appropriate infrastructure resources for their execution. In addition, external data repositories can be stitched into workflows by using Layer2 circuits (Internet2, ESnet) and SDN overlay networks [29].

Mobius supports multi-clouds and automated network provisioning to connect the clouds. It leverages the Ahab API [24] to provision VMs on ExoGENI and the *jclouds* API, which supports OpenStack based clouds, to provision Bare Metal (BM) nodes and VMs from Chameleon. Mobius leverages the ExoGENI Layer2 network provisioning API to stitch provisioned resources (potentially across clouds), and to connect external edge resources to stitchports [20], which essentially are named attachment points enabling direct Layer2 connections to resources outside the ExoGENI federation. Users, applications, and workflow management systems interact with Mobius using a REST API for provisioning resources and deploying services (Section 3.3).

Future provisioning support will include integrating in-network computing resources (e.g. with P4 [9]), as they become available on the FABRIC [7] network infrastructure. Mobius also supports provisioning resources from public clouds like Amazon AWS, and connecting them to the rest of the provisioned infrastructure with Cloud Connect [2, 3]. We have not leveraged this feature for the work in this paper.

3.3 FlyNet Services: Service Deployment

Container setup and orchestration. Since we envision that edge servers will be shared by more than one application the FlyNet architecture supports a container-based application deployment approach by using *KubeEdge* [40], which provides container orchestration at the edge. This containerized approach provides FlyNet with the required flexibility for workflows that support drone-based applications. The use of containers adds the additional benefit of simplified deployments of applications on edge nodes and supports migration of applications between edge nodes. The latter is an important requirement of drone-based applications, where the distance and thus the

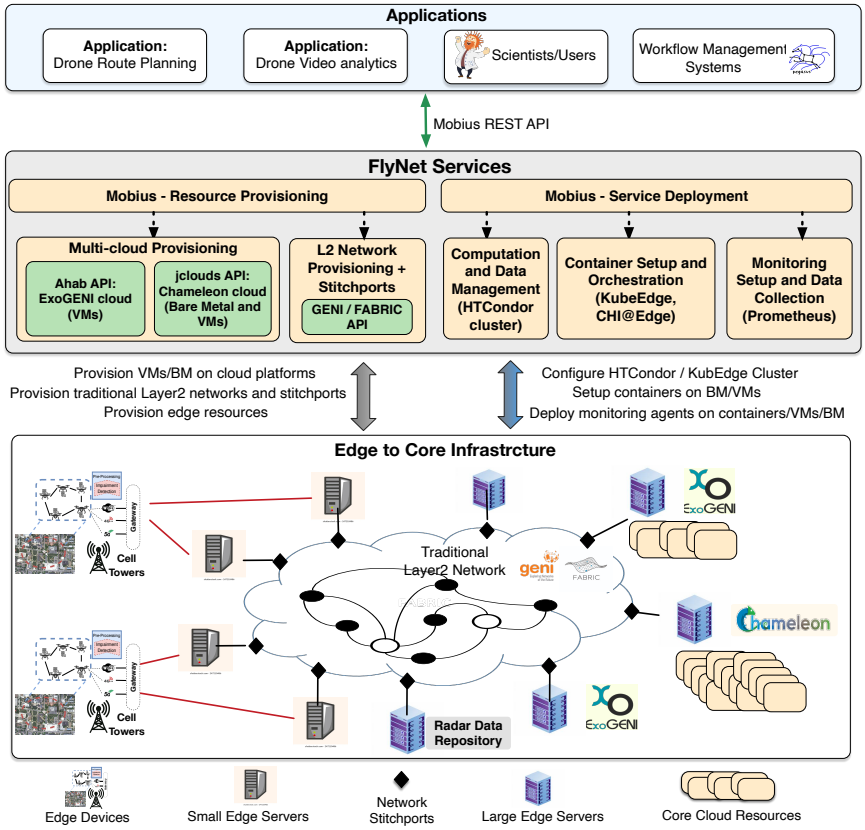

Figure 3: FlyNet System architecture showing how applications can leverage edge to core infrastructure via FlyNet services.

resulting latency between a drone and an edge node might become too large for effective and safe operations. In this case, migrating the application to a different edge node that is closer to the drone is critical. To support FlyNet, we extended Mobius to automatically deploy a container orchestration service using KubeEdge, which automatically instantiates KubeEdge clusters on the provisioned nodes. In order to support bare metal container orchestration on the edge resources, as on the Chameleon edge resources - *CHI@Edge* [11], Mobius takes advantage of the REST API [27] to provision the containers.

Computation and data management services. Mobius services also allow applications and workflow systems to deploy HTCondor [37] clusters - HTCondor Master/scheduler and HTCondor workers - on the provisioned resources selected from (potentially) multiple cloud platforms (ExoGENI and Chameleon), so that workflow/application tasks can be readily scheduled and executed. Mobius automates configurations for the networks, IP addresses, setup of the daemons and makes it easier for scientists and applications to use the provisioned infrastructure.

Monitoring setup and data collection - Prometheus. Mobius also automatically deploys Prometheus [4] monitoring agents on the provisioned resources - containers/VMs/BM. These agents monitor different resource metrics, e.g. CPU loads, continuously and stream the measurements to a central Prometheus server. The Prometheus server aggregates all the monitoring time series data from the agents and exposes an API for applications. The applications can query on the observed performance attributes of the resources and make key decisions for resource management. Such monitoring data is critical for edge resource selection (described in Section 5.3).

4 EVALUATION SETUP

4.1 Scenario

To facilitate a multifaced evaluation of the FlyNet platform, which is difficult with real drones, we have *simulated* drone flights that replay formerly recorded in-flight video. During the flight, the drone controller has to decide which cell towers the drone should communicate with. Additionally the controller needs to decide on which edge node(s) the video analytics algorithm should be executed. We use network emulation software on the simulated drone to test our video analytics application with throughput and latency settings representative of current (4G) and future (5G) cellular networks [41]. The video feed from the drone can also be forwarded from the edge to the core cloud. There, the data is stored longer-term with the goal to integrate it with the larger data set that is then used to train the machine learning-based video analytics algorithms. This part of the application is performed in the core cloud since latency is not a critical factor and more powerful compute resources can and need to be used. The infrastructure and software components required to execute the scenario are shown in Fig. 4 and Fig. 5 and explained in detail in the following sections.

Drone. In our scenario, we envision relatively small drones that are equipped with a video camera and some very minimal compute capability. Drones communicate with cell towers to transmit their video

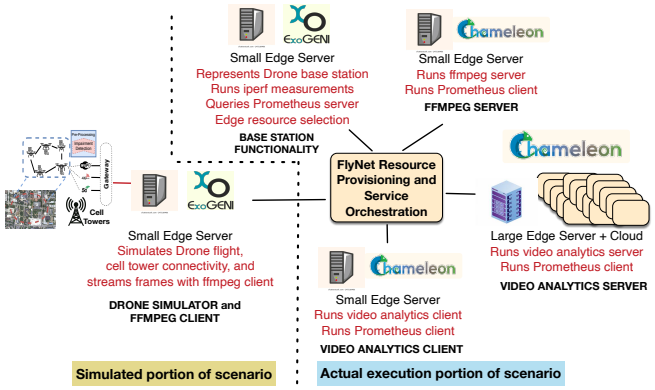

Figure 4: FlyNet evaluation scenario deployment. The simulated parts of our setup are shown left of the dotted line, while the right side shows the actual executed parts.

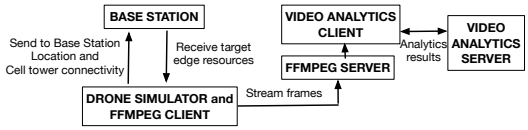

Figure 5: Evaluation scenario software components.

footage to either edge compute or core cloud compute resources. Their operations are limited by the power capacity that is provided by their batteries.

Cell Tower. A cell tower's only functionality in this scenario is to serve as the interface between the wireless network (for communication with drones) and the wired network (for communication with edge and cloud compute resources).

Base Station. The base station provides mobility management services for the drones. It collects information about a drone's location, current network conditions, and the current load of edge servers. Based on this information it can determine on what edge server the video analytics (e.g. object detection) should be performed and on which path the data from the drone should be transmitted to the edge computing node. The base station has to constantly monitor the drone's position and the network and edge computing load to change the location where computation is performed and/or the path between drone and edge. While the base station service can be located anywhere in the network, it is beneficial to place it on a device that is in the same region as the drone and the edge servers to avoid any additional latency. In this paper, we place the base station service on the same edge node as the video analytics process.

Edge Server(s). This is either a single compute node or a small cluster of compute nodes with high-bandwidth connectivity and gridded power. It provides compute capacity that is significantly higher than what is available on the drone but also significantly lower than the resources offered in the core cloud.

Core Cloud. This component represents a compute cloud that consists of one or more data centers. The core cloud offers substantial compute capabilities including specialized processing entities like GPGPUs and FPGAs. In our scenario, we make use of resources offered by the Chameleon testbed [17] as our core cloud platform.

4.2 Drone Flight Simulation and Video Playback

This initial evaluation of our approach does not include the deployment of actual drones and is based on flight simulation and the emulation of varying network conditions and video playback from the drone to the edge.

Drone Flight Simulation. To determine the simulated flight path of a drone, we define a start and end point. Based on that information our system currently determines a direct, straight-line path between source and destination. Cell towers are randomly placed in the region where the flight takes place and during a flight a drone detects the ones it can use for communications.

Video Playback. For video playback, a previously recorded video is stored at the node that simulates the drone. On that node, the *ffmpeg* tool is used to stream the video in real time to the edge node. To simulate the real-time streaming of the video frames they are sent according to the time stamps that were added to the frames when they were originally generated.

Application Containers. To support the scenario, we create two docker containers at the edge. One container runs the receiving end of *ffmpeg* and stores the received frames in a shared file system. The second container runs the video analytics application [30]. Video data for this application is streamed from the (simulated) drone to the first container. The decision to run the receiver of the video stream and the video analytics application in two separate containers allows for more flexibility with respect to resource allocation. For example, several containers executing the video analytics application can be initiated to parallelize this process, if and when required.

Resource Provisioning. To support the scenario, FlyNet provisions nodes from Chameleon and ExoGENI that emulate edge nodes (Figure 4). It performs automated provisioning of the edge computing nodes and the node(s) simulating drones and implementing base station capabilities on Chameleon and ExoGENI clouds, respectively. Furthermore, FlyNet orchestrates the entire setup including the provisioning of compute resources from ExoGENI and Chameleon, stitching the Layer2 networks between the cloud testbeds, and connecting them into the stitchport for the edge nodes.

4.3 Network Emulation

During the simulated drone flight, we generate a randomized mesh of cell towers such that at any given time the drone has five towers that it can communicate with. For the experiment we have used a 10km radius for 4G LTE networks. Each simulated tower remains static until the drone exits its 10km radius, at which point it is replaced with a new tower.

We use drone height and distance from the cell tower as rough proxy for reference signal receive power (RSRP), as the two are shown to be somewhat correlated [35]. We generate a signal parameter similar to cell phone 'bars', equating to representative RSRP values between -110 and -50dbm. We refer to estimates in the literature to vary network latency round trip times between 60 and 90ms [38] and uplink rates between 1 and 10Mbps based on the realistic RSRP [41]. In practice median upload rates may tend to fall on the lower edge of this range.

4.4 Resource Selection

The simulated drone sends cell tower information to the base station. The base station is aware of available edge resources and generates

a weighted network graph with nodes for the drone, the cell towers, and the edge resources. The weights on the graph include round trip time, bandwidth to the cell tower, and load on the edge resource. Using Dijkstra's algorithm, the base station determines the most optimal network path and appropriate edge node for the drone to use. By generalizing all network elements of the system into a graph, the network can be scaled up with more layers, nodes, and/or weight parameters, and continue to use the same framework.

Upon completion of the decision process the base station first selects the intended edge resource it will send data to, such that the server side of *ffmpeg* can be instantiated on that resource with KubeEdge. After receiving confirmation it indicates to the drone which path and which edge resource to use along with the estimated latencies and uplink rates. The *tc* utility is used to emulate the 4G wireles network characteristics on the link between drone and edge in advance of the initiation of the video streaming process.

5 EVALUATION

We present a two part evaluation: 1) a functional evaluation of the FlyNet-deployed system showing the types of monitoring information we can collect and how this information is used to derive application-relevant metrics, and 2) the application's use of the metrics for making decisions about computation offloading.

5.1 Monitoring Information

Underpinning our scenario is the notion of multiple edge devices serving as candidate locations for which to offload a batch of computation at a given time. Given that these devices may be heterogeneous in nature, possibly shared between users, and residing on different networks at different physical locations, we collect live state information from each device to inform our edge node selection process. There is an assumption that the monitoring data is reasonably continuous in time, such that metrics obtained a short time beforehand are representative at execution. For our video analytics application, we collect the following information about the processing resources:

- Network bandwidth: A key consideration for streaming high resolution video data is the network bandwidth. An airborne drone at any given time may be within range of multiple cell towers that could provide communications. As described in Sect. 4.3, we simulate a cell network mesh and assign representative bandwidths to each such path option. This represents the first hop from the drone. Given that we are using real processing systems for our evaluation, we deploy *iperf3* servers on each candidate node. Every iteration of the drone flight sends a one second bandwidth test from the base station to each device to serve as a proxy for the available bandwidth more generally associated with the device, excluding the drone connectivity which will often be the bottleneck.
- Average Compute Load: CPU load estimates may take on considerable importance when using a shared device, and/or for compute heavy jobs such as video analytics with neural networks. To measure such we use Prometheus (Section 3.3), which provides a REST interface allowing the system to request information for average CPU utilization of the edge node in the last one minute.
- Additional Queries: We collect state information from Prometheus to ensure that the node is online and active. Prometheus can also monitor disk utilization, memory usage, and with a plugin, GPU usage, however we did not collect these for our scenario, but would be able to if required for future workflows.

5.2 Application Metrics

There are a number of metrics that enable the characterization of the UAV application and help determine whether quality of service (QoS) is being met. Here we collect monitoring information and use it to calculate metrics such as frame rate, network bandwidth, processing time, and accuracy needed for the application to deliver the needed QoS. This information is then used to determine the best network path and edge server to use by the drone for computation offloading.

Metrics:

- Frame Rate: This metric is measured in frames per second (FPS) and can be applied to characterize the performance of networking and processing. In the case of streaming video data, the frame rate can be used to express the capability of the network. In terms of video analytics, this metric can be used to evaluate if an algorithm can process a video stream at a certain frame rate, i.e., in real time.
- Processing time: This metric will measure the time it takes to create detections by the video analytics application. This will allow us to evaluate under which conditions this process can be executed in real time.
- Detection Accuracy: We use the number of object detections per frame and compare it with the ground truth to determine the detection accuracy in terms of confidence scores. These confidence score values are then used to compare the quality of object detections among different video resolutions in our evaluation experiments.

5.3 Network Path and Edge Compute Selection

As described in Sect. 4.1, the resources required for the video analytics workflow can be selected from several network links and edge compute nodes. For example, the UAV might be in the coverage area of several cell towers, and from each tower, there are paths to multiple candidate edge nodes. Simply using the shortest path to the closest edge node will not always be the best choice. Available bandwidth might be lower on the shortest path than on alternative paths or the selected edge node might have a high competing network and/or compute load, be temporarily unavailable, or lack required hardware for an application. Both simulated and measured resource metrics described in 5.1 are used as input to the edge compute selection module. At each iteration of the drone flight update, new values are generated and queried and fed into the utility calculation along with the application requirements. We experimented with two utility functions: one giving more weight to the load at the edge and one prioritizing the bandwidth to the edge. In the case of streaming video frames, prioritizing the available bandwidth to the edge device over the load results in a 30 percent improvement in frames per second transmission as can be seen in Fig. 6. This is the case because video streaming is not a heavily compute intensive process. For other applications, alternative utility functions may be more appropriate. The FlyNet framework and system enables this type of experimentation.

5.4 Architectural Choices

During flight, the camera on the drone generates video footage. Based on the capabilities of the video camera and encoder settings a video

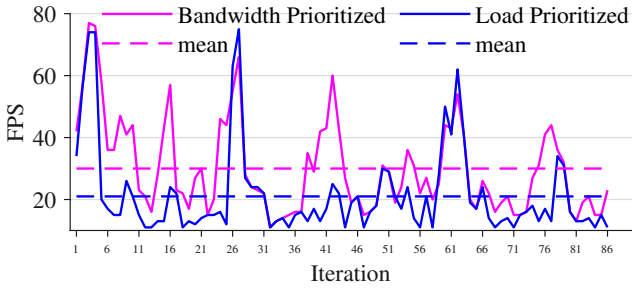

Figure 6: The impact of the FlyNet utility function in improving application performance. By prioritizing available bandwidth associated with particular cell towers and available edge worker nodes, we improve ffmpeg transmission rates of a batch of compressed frames from an mpg video over TCP.

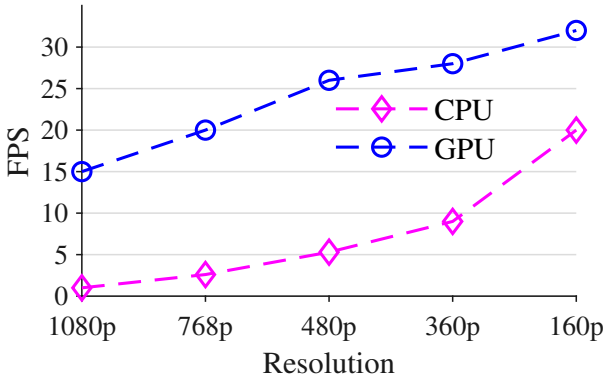

Figure 7: The impact of video resolution and processing unit type (CPU vs. GPU) on video processing performance in frames per second (FPS).

stream of varying resolutions (160p, 320p, 480p, etc.) and frame rates (10 fps, 30 fps, 60 fps, etc.) can be created. In our first analysis for the drone video analytics workflow, we evaluate its performance based on the processor type and network conditions. Such information is required to inform the allocation of underlying resources for a particular workflow. For example, a video analytics application might require a specific resolution or frame rate to achieve a minimum detection accuracy. Based on that information the necessary compute and networking resources can be allocated.

For this experiment the video is streamed via *ffmpeg* from a node that emulates a drone to a node that hosts the Docker container in which the video analytics application runs. While the frame resolution is varied we observe the frame rate at which the application can process the incoming data. Figure 7 shows that the frame rate is mainly impacted by the resolution of the video frames and the processing unit used (CPU vs. GPU) to execute the application. Thus, if a certain frame rate is required, this can either be achieved by running the application on a GPU with higher resolution frames or on a CPU with lower resolution frames. These results also inform the resource selection described in Sect. 4.1. For example, if alternative edge nodes are available and the free capacity of these nodes is known, the one that matches the application requirements best can be selected.

5.5 Resolution & Detection Confidence

While lowering the resolution of a video is certainly an option to achieve a specific end-to-end frame rate (see Sect. 5.4), the impact in terms of detection confidence has to be taken into consideration. We conducted an experiment (solely on the processing side of the workflow) in which the image resolution of the individual frames ingested by the video analytics algorithm was varied from 160p up to 1080p. This experiment was conducted on an Intel(R) Xeon(R) CPU E5-2670 v3 at 2.30GHz in an x86_64 architecture, 48 CPUs per node, 125GB total memory, and equipped with 3x Nvidia V100 GPUs for running video analytics on the Darknet deep neural network. As a metric, we used the confidence with which the algorithm detected a certain type of object. Fig. 8 shows that in the 1080p case actual traffic lights are detected as traffic lights with a confidence of more than 95%. Further analysis of Fig. 8 shows that the confidence depends on the object type (car, truck, traffic light, person) and the image resolution. The results also show that, apart from the exception of the truck object, the confidence of a detection stays relatively stable down to an image resolution of 480p. This coincides with the results from Sect. 5.4, where a frame rate of 25 fps or higher can be maintained at 480p resolution if video analytics is performed on a GPU.

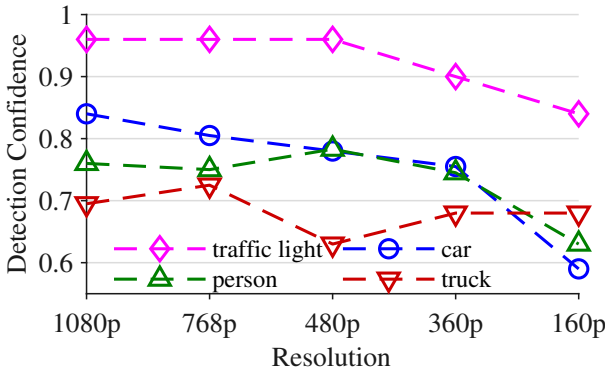

Figure 8: The impact of resolution on detection confidence.

5.6 Video Streaming Under Various Network Bandwidths

During the drone video streaming process, it is critical that the available bandwidth on the path between the drone and the edge is sufficient to allow data transmission at an appropriate frame rate. Not meeting this requirement can negatively impact video processing and object detection and tracking. To identify the ideal bandwidth required to stream videos encoded in different resolutions we perform the following evaluation. A video with a fixed frame rate (25 FPS) is streamed from the drone to the edge while the available bandwidth is set to eight different values. At the streaming client (edge node), we evaluate the resulting FPS based on the available bandwidth. As shown in Fig. 9, the frame rate of 25 fps can be sustained if the available bandwith is greater than 1 Mbps.

From Figure 9, we can also see that the network bandwidth has a significant influence on the streaming quality represented by the FPS values. Specifically, when the bandwidth drops below 2Mb/s for the 1080p video resolution, 1Mb/s for the 768p resolutions, and 500Kb/s for all the other lower resolutions, the video streaming is hugely

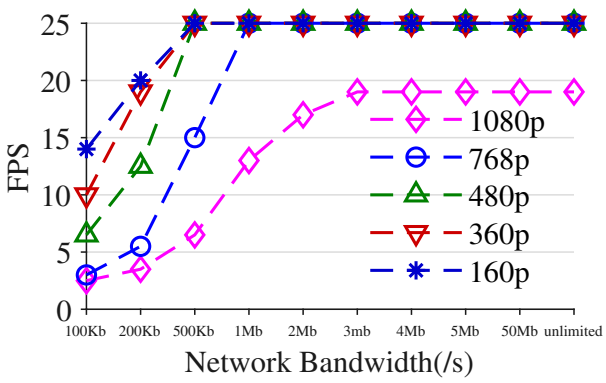

Figure 9: The impact of available bandwidth on frame rates at different video resolutions

impaired, and thus the FPS received on the edge node decreases significantly. As the network bandwidth drops even lower, so does the frame rate. From the results, we can see that a network bandwidth at 1Mb/s is sufficient for streaming drone videos in most resolutions, i.e., ranging from 160p to 768p, to achieve the native FPS of the original video, which in this case is 25. However, in the case of 1080p resolution we did notice that we were not able to achieve the native 25 FPS, even in the case of the maximum link bandwidth (which is ~ 24Gb/s in Chameleon cloud network). We attribute this result to the much larger video frame size (in terms of storage), which requires more time to be written to the local file system of the edge node.

5.7 Network Loss

Packet loss affects all networks, however, it is particularly common in wireless network links, where packet losses can occur more frequently due to interference, low signal-to-noise-ratio, and other effects. Since at least the first hop in the end-to-end path between a drone and the respective receiver will be a wireless link we performed an experiment where packet loss was induced. In this scenario, we used *tc* to induce a worst case of 10% packet loss during the transmission of a 1080p video. To evaluate the impact of packet loss we compare the number of detections when we have a lossless transmission with one that has a 10% packet loss. Fig. 10 shows unambiguous results. While the number of detections is decreasing for most objects in the case of packet loss, the number of detections for the type "truck" is increasing. In the latter case, we conjecture that it is an effect caused by mislabeling. This means, that actual objects like cars and buses get mislabeled as trucks. In future work, we plan to repeat this evaluation with labeled data to be able to clearly distinguish between mislabeling and missing an object completely.

6 RELATED WORK

Edge Computing. Recent work is focusing on the support of IoT applications through edge computing, where computation can be offloaded from the device to compute resource located at the edge of the network (which is much closer to the devices than any cloud resource). A comprehensive overview on resource management for edge computing with a focus on IoT support is presented by Hong and Varghese [16]. While edge computing is a very broad research area, we focus on the related work that emphasizes on the use of edge computing to support drone-based applications. Zhou et al. [43]

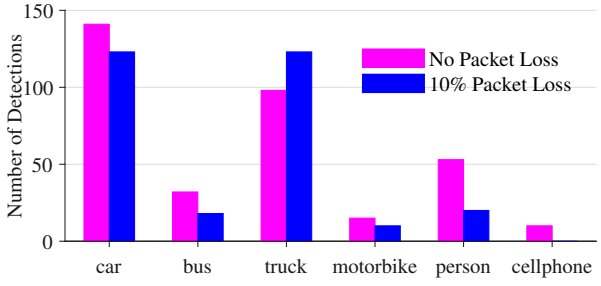

Figure 10: Impact of packet loss on number of detections.

present a survey on computational offloading for drone-based applications. This survey shows that a significant portion of the existing work focuses on using drones as part of the network infrastructure to offload data from IoT devices at remote locations. The work presented in this paper is different since drones act also as data sources (video data) and significant computational resources are used to process the data. More closely related to our work is the work presented by Hayat et al. [15]. In their work, drones generate video data which is used for navigation. Their study analyzes the trade-off between different computation models (drone, hybrid, edge), which shows the offloading is only beneficial if the compute power of the edge node is significantly higher than the one of the drone. In recent work, we conducted preliminary studies to understand the benefits of a policy-based offloading scheme [12, 30]. In these studies, we used an object tracking application similar to the one in Figure 1 involving real-time video analytics in geo-distributed areas to meet user quality of experience (QoE) expectations.

The approach presented in this paper is complementary to the works presented above since it focuses on automatically provisioning of edge and cloud resources and deploying the necessary services to easily generate the platform that drone applications depend on.

Resource provisioning and networking for science applications. There have been extensive survey papers [13, 19] in regards to provisioning IaaS cloud resource for scientific workflows. Wang et al. [39] propose an approach to build and run scientific workflows on a federation of clouds using Kepler and CometCloud. Moreover, there have been strategies for workflow systems to deploy virtual machines in the cloud with limited support for on-demand provisioning and elasticity, which are essential for the driving UAV applications in this work. Ostermann et al. [28] discussed a set of VM provisioning policies to acquire and release cloud resources for overflow grid jobs from workflows, and characterized the impact of those policies on execution time and overall cost. In prior work [20], we presented dynamic provisioning techniques that spawn resources based on compute elasticity using Mobius [24].

On the perspective of networking between the compute, storage and instrument sites, Macker et al. [22] describe workflow paradigms to address network edge workflow scenarios.

Our work presented in this paper differs from the above by presenting easy-to-use, on-demand resource provisioning mechanisms for data movement and edge and cloud compute provisioning for UAV applications. We provide dedicated network connections among multiple cloud provider sites, and leverage unique resource provisioning mechanisms that instantiate KubeEdge environments on clouds with advanced network setups.

Video Analytics. Drones that provide video footage are increasingly relying on video analytics applications that require HPC resources and real-time communications. Rao et al. [31] proposed an approach based on microservics that supports drone operators in performing video analytics. These are used to asses wide area scenes and help develop a plan of action. However, due to latency requirements and limited network bandwidth, UAV applications adaptively compress the data to strike a balance between overall analytics accuracy and bandwidth consumption, as demonstrated in several recent publications [30, 33]. Our proposed solution extends the idea of utilizing edge servers to enhance video analytics applications by applying complex learning-based video analytics such as object detection, categorizing, recognition and tracking.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we present the FlyNet system, which supports scientific workflows from the edge to the core for UAV applications. We identify the challenges that UAV applications pose to design and implement such a system. These challenges focus on the integration of edge computing resources, next generation networks, and core cloud resources, such that drone-related, scientific workflows can perform with very little latency and high reliability. We present an architecture that treats IoT devices (like drones) and edge servers as first-class citizens of scientific workflows. We then present the overall FlyNet system that performs automated resource provisioning and service deployment. Through evaluation of the system with an important application class - drone video analytics - we demonstrate the challenges the establishment of such application workflows face. The results show that the appropriate or inappropriate selection of resources can have significant impact on the application performance. These results clearly uncover a new challenge for the orchestration of scientific workflows such as: How can the resource selection for scientific workflows be sufficiently abstracted such that domain scientists can allocate an appropriate set of resources from the edge-to-core computing continuum to optimize their workflows?

In future work, we plan to continue to address these challenges. In particular, we will extend the edge compute resource allocation capability to be location aware such that researchers can request processing nodes in geographic proximity to their applications without needing to look up and specify sites in advance. In addition, we shall investigate the use of information-centric networking concepts to facilitate data transfers from the edge devices to the local resources without requiring known and dedicated IP addresses. With such an approach the data transfer model can be inverted. Rather than pushing video data for processing to a location that has to be determined upon creation of a workflow, processing nodes can query content (e.g., video frames) without having to address end systems. This may assist in addressing challenges associated with the migration to 5G where network handoffs may occur very quickly as cellular range decreases and the number of cellular towers increases.

ACKNOWLEDGMENT

This work is funded by NSF award OAC-2018074. Results in this paper were obtained using Chameleon and ExoGENI testbeds supported by NSF.

REFERENCES

[1] Intel flexran. https://github.com/intel/FlexRAN. Accessed: 2021-08-09.

[2] Internet2 Cloud Connect Production Service Announced, Capabilities Support Research, Scientific Collaboration, Academic Enterprise From Campus to the Cloud. https://www.internet2.edu/news/detail/17118/. Accessed: 2020-01-06.

[3] Networking for Cloud: Internet2 Webpage. https://www.internet2.edu/products-services/advanced-networking/networking-for-cloud/. Accessed: 2020-01-06.

[4] Prometheus. https://prometheus.io/docs/introduction/overview/.

[5] Akiyama, K. First m87 event horizon telescope results. iii. data processing and calibration. *The Astrophysical Journal. Letters 875*, 1 (4 2019).

[6] Baldin, I., Chase, J., Xin, Y., Mandal, A., Ruth, P., Castillo, C., Orlikowski, V., Heermann, C., and Mills, J. *ExoGENI: A Multi-Domain Infrastructure-as-a-Service Testbed.* 2016, pp. 279–315.

[7] Baldin, I., Nikolich, A., Griffioen, J., Monga, I. I. S., Wang, K.-C., Lehman, T., and Ruth, P. Fabric: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing 23*, 6 (2019), 38–47.

[8] Blasch, E., Ravela, S., and Aved, A. *Handbook of Dynamic Data Driven Applications Systems. [electronic resource].* Springer International Publishing, 2018.

[9] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and Walker, D. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev. 44*, 3 (July 2014), 87–95.

[10] Cevik, M., Ruth, P., Keahey, K., and Riteau, P. Wide-area software defined networking experiments using chameleon. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (April 2019).

[11] Chameleon Edge Resources. https://www.chameleoncloud.org/experiment/chiedge/.

[12] Chemodanov, D., Qu, C., Opeoluwa, O., Wang, S., and Calyam, P. Policy-based function-centric computation offloading for real-time drone video analytics. In *2019 IEEE LANMAN* (2019), IEEE.

[13] Galante, G., Erpen De Bona, L. C., Mury, A. R., Schulze, B., and Rosa Righi, R. An analysis of public clouds elasticity in the execution of scientific applications: A survey. *Journal of Grid Computing 14*, 2 (June 2016), 193–216.

[14] Grasso, C., and Schembra, G. A fleet of mec uavs to extend a 5g network slice for video monitoring with low-latency constraints. *Journal of Sensor and Actuator Networks 8*, 1 (2019).

[15] Hayat, S., Jung, R., Hellwagner, H., Bettstetter, C., Emini, D., and Schnieders, D. Edge computing in 5g for drone navigation: What to offload? *IEEE Robotics and Automation Letters 6*, 2 (2021), 2571–2578.

[16] Hong, C.-H., and Varghese, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Comput. Surv. 52*, 5 (Sept. 2019).

[17] Keahey, K., Anderson, J., Zhen, Z., Riteau, P., Ruth, P., Stanzione, D., Cevik, M., Colleran, J., Gunawi, H. S., Hammock, C., Mambretti, J., Barnes, A., Halbach, F., Rocha, A., and Stubbs, J. Lessons learned from the chameleon testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.

[18] LIGO Scientific Collaboration and Virgo Collaboration. GW150914: First results from the search for binary black hole coalescence with advanced LIGO. *Phys. Rev. D 93*, 12 (June 2016), 122003.

[19] Liu, J., Pacitti, E., Valduriez, P., and Mattoso, M. A survey of data-intensive scientific workflow management. *Journal of Grid Computing 13*, 4 (Dec. 2015), 457–493.

[20] Lyons, E., Papadimitriou, G., Wang, C., Thareja, K., Ruth, P., Villalobos, J., Rodero, I., Deelman, E., Zink, M., and Mandal, A. Toward a dynamic network-centric distributed cloud platform for scientific workflows: A case study for adaptive weather sensing. In *15th International Conference on eScience (eScience)* (2019), pp. 67–76. Funding Acknowledgments: NSF 1826997.

[21] Lyons, E., Westbrook, D., Grote, A., Papadimitriou, G., Thareja, K., Wang, C., Zink, M., Deelman, E., Mandal, A., and Ruth, P. An on-demand weather avoidance system for small aircraft flight path routing. In *Dynamic Data Driven Applications Systems - Third International Conference, DDDAS 2020, Boston, MA, USA, October 2-4, 2020, Proceedings* (2020), F. Darema, E. Blasch, S. Ravela, and A. Aved, Eds., vol. 12312 of *Lecture Notes in Computer Science*, Springer, pp. 311–319.

[22] Macker, J. P., and Taylor, I. Orchestration and analysis of decentralized workflows within heterogeneous networking infrastructures. *Future Generation Computer Systems 75* (2017).

[23] Mandal, A., Ruth, P., Baldin, I., Da Silva, R. F., and Deelman, E. Toward prioritization of data flows for scientific workflows using virtual software defined exchanges. In *2017 IEEE 13th International Conference on e-Science (e-Science)* (Oct 2017).

[24] Mandal, A., Ruth, P., Baldin, I., Xin, Y., Castillo, C., Juve, G., Rynge, M., Deelman, E., and Chase, J. Adapting scientific workflows on networked clouds using proactive introspection. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)* (Dec 2015), pp. 162–173.

[25] McLaughlin, D., Pepyne, D., Philips, B., Kurose, J., Zink, M., et al. Short-Wavelength technology and the potential for distributed networks of small radar systems. *Bull. Am. Meteorol. Soc. 90*, 12 (Dec. 2009), 1797–1817.

[26] Mobius Github Repository. https://github.com/RENCI-NRIG/Mobius.

[27] Openstack Containers API. https://docs.openstack.org/api-ref/application-container/.

[28] Ostermann, S., Prodan, R., and Fahringer, T. Dynamic cloud provisioning for scientific grid workflows. In *2010 11th IEEE/ACM International Conference on Grid Computing* (Oct 2010), pp. 97–104.

[29] Papadimitriou, G., Lyons, E., Wang, C., Thareja, K., Tanaka, R., Ruth, P., Villalobos, J. J., Rodero, I., Deelman, E., Zink, M., and Mandal, A. Application aware software defined flows of workflow ensembles. In *2020 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)* (2020), pp. 10–21.

[30] Qu, C., Wang, S., and Calyam, P. Dycoco: A dynamic computation offloading and control framework for drone video analytics. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)* (2019), IEEE, pp. 1–2.

[31] Ramisetty, R. R., Qu, C., Aktar, R., Wang, S., Calyam, P., and Palaniappan, K. Dynamic computation off-loading and control based on occlusion detection in drone video analytics. In *2020 ACM 21st International Conference on Distributed Computing and Networking (ICDCN)* (2020), ACM.

[32] Sahingoz, O. K. Networking models in flying ad-hoc networks (fanets): Concepts and challenges. *J. Intell. Robotics Syst. 74*, 1-2 (Apr. 2014), 513–527.

[33] Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., and Guizani, M. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access 7* (2019), 48572–48634.

[34] Silva, R. F. d., Rynge, M., Juve, G., Sfiligoi, I., Deelman, E., Letts, J., Würthwein, F., and Livny, M. Characterizing a high throughput computing workload: The compact muon solenoid (CMS) experiment at LHC. *Procedia Comput. Sci. 51* (Jan. 2015), 39–48.

[35] Simpson, O., and Sun, Y. Lte rsrp, rsrq, rssnr and local topography profile data for rf propagation planning and network optimization in an urban propagation environment. *Data in Brief 21* (2018), 1724–1737.

[36] Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., and Sabella, D. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials 19*, 3 (2017), 1657–1681.

[37] Thain, D., Tannenbaum, T., and Livny, M. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience 17*, 2-4 (2005), 323–356.

[38] Vodafone. Network-based drone airspace management trial report. Tech. rep., Vodafone, dec 2019.

[39] Wang, J., AbdelBaky, M., Diaz-Montes, J., Purawat, S., Parashar, M., and Altintas, I. Kepler + cometcloud: Dynamic scientific workflow execution on federated cloud resources. *Procedia Computer Science 80* (2016), 700 – 711. International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA.

[40] Xiong, Y., Sun, Y., Xing, L., and Huang, Y. Extend cloud to edge with kubeedge. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)* (2018), pp. 373–377.

[41] Yang, G., Lin, X., Li, Y., Cui, H., Xu, M., Wu, D., Ryden, H., and Redhwan, S. B. A telecom perspective on the internet of drones: From lte-advanced to 5g. *ArXiv abs/1803.11048* (2018).

[42] Zafar, W., and Muhammad Khan, B. Flying ad-hoc networks: Technological and social implications. *IEEE Technology and Society Magazine 35*, 2 (June 2016), 67–74.

[43] Zhou, F., Hu, R. Q., Li, Z., and Wang, Y. Mobile edge computing in unmanned aerial vehicle networks. *IEEE Wireless Communications 27*, 1 (2020), 140–146.