FlyNet: Drones on the Horizon

Alicia Esquivel Morel, Chengyi Qu, Prasad Calyam
University of Missouri - Columbia
{ace6qv, cqy78}@mail.missouri.edu, pcalyam@missouri.edu

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

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

George Papadimitriou, Ewa Deelman
University of Southern California
{georgpap, deelman}@isi.edu

***Abstract*—Over the past few years, due to the boom of advances in image processing, edge computing, and wireless networking, Unpiloted Aerial Vehicles (UAVs), often referred to as drones, have become an important enabler to support a wide variety of scientific applications, ranging from environmental monitoring, disaster response, wildfire monitoring, to the survey of archaeological sites. In this article, we present the FlyNet platform, which extends an existing workflow management system to support and manage scientific workflows. FlyNet enables automated resource allocation, workflow instrumentation, and network service support to facilitate researchers in their goal to analyze data for new scientific discoveries. In addition, FlyNet provides network services management to support QoS for efficient data transport between edge devices, edge servers, and the cloud.**

***Index Terms*—Edge and Cloud Computing, Workflows, UAVs**

I. Introduction

Drones are literally on the horizon. Unpiloted Aerial Vehicles (UAVs) (often referred to as drones) are now supporting a wide range of scientific applications, ranging from environmental monitoring, disaster response, and wildfire monitoring, to the survey of archeological sites. The success of these applications heavily depends on the ability to efficiently manage and analyze large volumes of data generated by drones. This is where scientific workflow support comes into play, providing researchers with the tools and techniques to better manage and analyze their data. In this context, scientific workflows can be characterized as a series of processes that are executed in a specific order to analyze the data generated by drones. Examples include the processing and analysis of video, imagery, and other sensor data. By using workflow management systems for scientific UAV applications, researchers can create data management and analysis processes with the goal to efficiently and effectively extract insights and new knowledge from the collected data.

In parallel, there has been an evolution of the cloud computing paradigm with the advent of edge computing, providing researchers with the opportunity to span their workflows across the edge-to-cloud spectrum based on the resource needs of their scientific applications. To streamline data management based on application requirements, resources across the spectrum need to be appropriately allocated. Unfortunately, selecting the appropriate set of resources for a specific scientific workflow is often a challenge for domain scientists who are not experts in distributed computer systems.

FlyNet introduces a platform to support scientific workflows from the edge to the core for UAV and other edge-to-cloud applications by automating the processes of resource allocation, workflow implementation, and network service support to support researchers in their goal to analyze data for new scientific discoveries.

II. FlyNet System Architecture

The FlyNet architecture (shown in Figure 1), supports the composition of end-to-end (edge-to-core) workflows capable of supporting scientific UAV and other edge-to-cloud applications.

A. Edge-to-Core Infrastructure

The edge-to-core infrastructure depicted at the bottom of Figure 1 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 to support the increasing scale of IoT applications, 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 workloads like deep learning models).

As the latency on the spectrum increases, processing packets and turning them around using in-network computing capabilities (either compute resources or specialized programmable hardware deployed in the network core) can be envisioned. 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

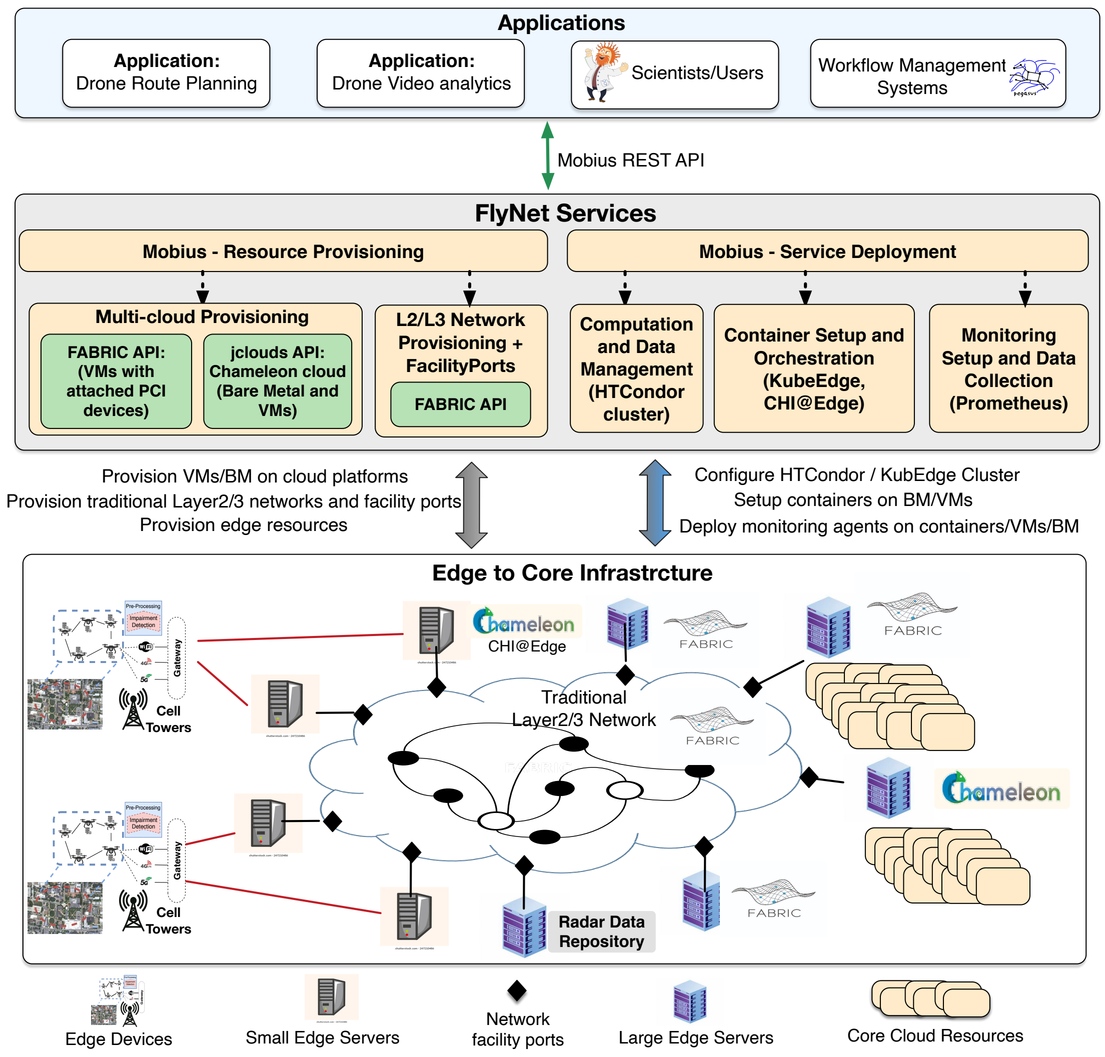
Applications
Application: Drone Route Planning
Application: Drone Video analytics
Scientists/Users
Workflow Management Systems
Mobius REST API
FlyNet Services
Mobius - Resource Provisioning
Mobius - Service Deployment
Multi-cloud Provisioning
FABRIC API: (VMs with attached PCI devices)
jclouds API: Chameleon cloud (Bare Metal and VMs)
L2/L3 Network Provisioning + FacilityPorts
FABRIC API
Computation and Data Management (HTCondor cluster)
Container Setup and Orchestration (KubeEdge, CHI@Edge)
Monitoring Setup and Data Collection (Prometheus)
Provision VMs/BM on cloud platforms
Provision traditional Layer2/3 networks and facility ports
Provision edge resources
Configure HTCondor / KubEdge Cluster
Setup containers on BM/VMs
Deploy monitoring agents on containers/VMs/BM
Edge to Core Infrastrcture
CHI@Edge
FABRIC
Traditional Layer2/3 Network
Cell Towers
Gateway
Radar Data Repository
Edge Devices
Small Edge Servers
Network facility ports
Large Edge Servers
Core Cloud Resources

Fig. 1. FlyNet System architecture showing how applications can leverage edge to core infrastructure via FlyNet services.

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.

B. *FlyNet Services: Resource Provisioning*

To implement this overall architecture, FlyNet uses a network-centric platform called Mobius [1] with support for provisioning programmable cyberinfrastructure comprised of FABRIC [2] and Chameleon Cloud [3] testbeds. Mobius makes it easier for applications to provision and manage the appropriate infrastructure resources for their execution. It supports multi-clouds and automated network provisioning to connect the clouds. It leverages the *jclouds* API, which supports OpenStack based clouds, to provision Bare Metal (BM) nodes and VMs from Chameleon. It uses the FABRIC FABlib API [4] to (a) provision VMs from FABRIC with directly attached PCI devices – GPUs, Network Cards, NVMe drives, FPGAs, and (b) to provision Layer 2/3 networks and facility ports [5] for connecting different FABRIC Core and Edge nodes with external infrastructure. Users, applications, and workflow management systems interact with Mobius using a REST API for provisioning resources and deploying services (Section II-C).

C. *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* [6], 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 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 resulting latency between a drone and an edge node might become too large for effective and safe operations. In that 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* [7], Mobius takes advantage of the REST API [8] to provision the containers.

Computation and data management services. Mobius services also allow applications and workflow systems to deploy HTCondor [9] clusters - HTCondor Master/scheduler and HTCondor workers - on the provisioned resources selected from (potentially) multiple cloud platforms (FABRIC and Chameleon), such 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 [10] 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.

III. Edge-to-Cloud Workflow Orchestration

A. Challenges of Edge-to-Cloud Execution

Edge-to-cloud computing environments make it possible for applications and systems to capitalize on the desirable advantages offered by both computing paradigms: faster response times, data locality, cost savings at the edge, scalability, high availability, and reliability provided by the cloud. Effectively utilizing both computing paradigms within such a complex execution environment for a given application presents a number of challenges. First, available resources and their states need to be visible in order to make scheduling decisions. Some environments with IoT devices may experience churn due to limited power and network connection. This is especially the case for UAVs that might come in and out of communication range when executing a mission. Second, scheduling decisions must be made. When running in the cloud, both compute and data movement costs may need to be considered. Incorporating the edge may involve taking into consideration energy consumption, limited compute capacity, and storage constraints. In addition to scheduling decisions, there may be resource provisioning decisions that can be made to better accommodate varying levels of expected load. Such provisioning can happen at the edge, for example in a cloudlet or on idle edge devices. Third, software systems must be in place to execute computations at both ends and automatically handle failures when they occur. Finally, the ability to capture fine-grained performance metrics or provenance data is indispensable to optimizing executions on an edge-to-cloud continuum.

B. Edge-to-Cloud Workflow System Design

In order to orchestrate workflows that span edge and cloud resources, FlyNet uses the Pegasus Workflow Management System [11] Pegasus has a number of key features that make it a particularly good candidate to provide the automation needed to span the edge-to-cloud continuum. Most importantly, it has the notion of an abstract workflow. This is a workflow description that is resource independent and captures the workflow at the science level: the codes used for the computations, the data needed, and generated by the workflow tasks. Pegasus takes this abstract workflow description and maps it to the available resources, generating the necessary resource-dependent scripts for job submission and adding the necessary data movement between jobs by invoking appropriate data transfer protocols. These resource-specific scripts produced by Pegasus form the executable workflow that is then passed to HTCondor's DAGMan [12] for execution.

Pegasus' architecture and the use of proven and versatile technologies such as HTCondor allowed us now to extend the workflows to the edge. HTCondor can run on any edge or cloud resource running Linux, macOS, or Windows, creating a hybrid edge-cloud infrastructure. In order to match jobs specifically with edge or cloud resources, we added an additional attribute, which indicates whether or not that resource was an edge or cloud resource. During workflow generation, jobs can be annotated with the type of resources they should be matched with. During execution, HTCondor takes into account this requirement in addition to other job requirements and matches the job with the appropriate resources.

To support data movement operations workflows are configured to use remote transfer protocols such as HTTP and SCP, and local file system operations. These are managed by the *pegasus-transfer* utility. Pegasus-transfer is invoked for each job to handle staging in input data and staging out output data. For jobs that are scheduled on locations where input data already resides, *symlinks* are used by *pegasus-transfer* to avoid unnecessary data movements and reduce overall disk usage. One notable advantage of *pegasus-transfer* is that data movement operations are decoupled from the jobs themselves. For example, a change in the locations of initial input files would only require a workflow-specific configuration change with Pegasus.

C. Workflow Evaluation

For the evaluation, we used a drone application and two other edge-to-cloud workflows. We use these applications to demonstrate the feasibility of our approach and the benefits of using an infrastructure that provides resources across the edge-to-cloud continuum.

Typical UAV Workflow. This workflow [13] was developed to represent data aggregation and analytics applications, which run in edge-to-cloud environments. For such applications, initial input data is derived at the edge from multiple instruments such as cameras and sensors mounted on drones. Each input goes through pre-processing steps before being aggregated by a single job that outputs the final result.

Wind Workflow. The Wind workflow [1], [14] is designed to identify areas of maximum observed wind magnitudes from a network of overlapping Doppler weather radars. Single radar files in polarimetric format, from a total of seven radars, are regridded into a common coordinate system. At a centralized location, the workflow periodically takes any available scans collected over a given time interval and creates a new file in a latitude/longitude projection representing the highest winds that have been observed during the time period.

Orcasound Workflow. Orcasound [15] is a community-driven project that leverages hydrophone sensors deployed in three locations in the state of Washington (San Juan Island, Point Bush, and Port Townsend) in order to study Orca whales in the Pacific Northwest region. The Orcasound Pegasus workflow [16] processes the hydrophone data of one or more sensors in batches for each timestamp and converts them to a WAV format. Using the WAV output, the workflow creates spectrogram images that are stored in the final output location. Furthermore, using a pre-trained Orcasound model developed by the community, the workflow scans the WAV files to identify potential sounds produced by the orcas.

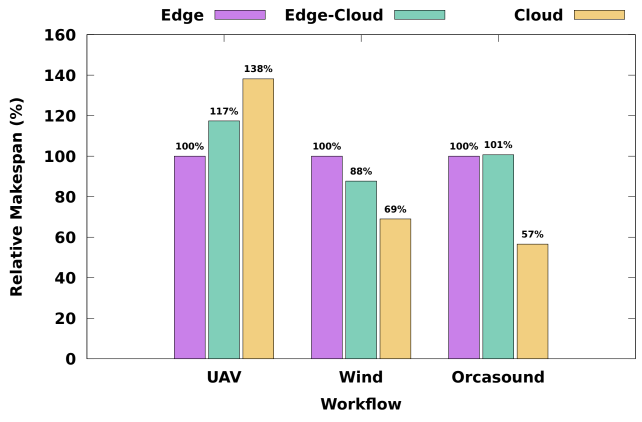

Fig. 2. Workflow makespans for ten runs of each of the three workflows under different scenarios.

Edge-to-Cloud Evaluation To evaluate our approach we executed each of the three workflows in edge-only, edge-to-cloud, and cloud-only scenarios. We emulated an edge-to-cloud scenario and provisioned nodes on both Chameleon sites at TACC and UChicago. At TACC, we deployed our cloud site, where we assumed we could get unlimited resources, and at UChicago we used Docker to deploy our edge nodes and limit their processing capabilities [17].

In Fig. 2 we present the average makespan for ten runs of each of the three workflows under the different scenarios, as a percentage to the edge scenario. As can be seen, the wall clock time (makespan) for each of the three workflows is different for the three execution environments. While the typical UAV workflow performs best in an edge-only environment, the Wind and Orcasound workflows perform best in the cloud-only environment.

Additionally, in Fig. 3 we present the average time the workflows spent transferring data over the wide area network, as a percentage of the edge scenario. This figure provides some insights as to why the cloud-only scenario does not perform the best in all cases. The UAV workflow was designed to favor the edge-only scenario and without any computation at the edge, the workflow is forced to spend 30 times more on WAN transfers, negating any increase in compute power the cloud offers. On the other hand, the Wind and Orcasound workflows still have to spend about 4-times and 2-times more on WAN transfers respectively, but the speed up these workflows are getting from the cloud resources is enough to improve their overall makespans (Fig. 2).

Overall, these results show the benefits and flexibility this approach provides. Without any additional development, Pegasus can map the workflows to edge and/or cloud resources, enabling optimizations under constraints utilizing different trade-offs (e.g., shorter makespan versus more network utilization).

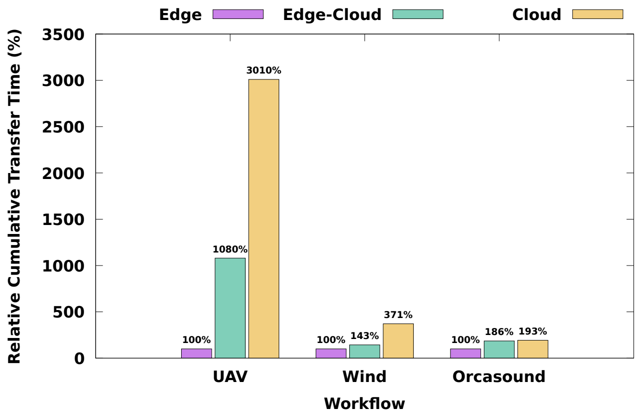

Fig. 3. Cumulative time spent on transferring data over WAN.

IV. Network Services for Edge-to-Cloud Workflows

The edge-to-cloud orchestration presented in Sect. III shows the benefits of being able to explore the trade-off between compute time, data transmission time, and queueing delays for different workflows. In addition to this workflow orchestration, we also investigate how network services that are based on programmable data planes can efficiently manage the transmission of data in the edge-to-cloud continuum. Such network services are an important component in the FlyNet architecture since they support efficient data transport between edge devices, edge servers, and the cloud. Figure 4 shows an example scenario for search and rescue operations, which requires efficient transmission of video footage to adequate compute resources.

The advent of programmable data planes provides In-band Telemetry (INT) capabilities that address network resource usage, identify resource contention, and provide detailed visibility into the network infrastructure. Based on these capabilities, INT can be used to enable network Quality of Service (QoS) assuring that workflows receive the required network service.

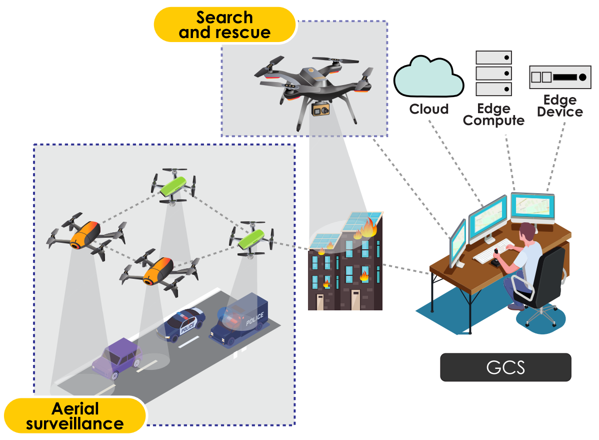

Fig. 4. Unmanned Aerial Vehicles can be utilized for a wide variety of applications such as e.g., search-and-rescue, and aerial surveillance. Challenges for network services management need to be overcome to guarantee satisfactory performance of network-edge based applications such as video delivery.

A. *In-band Network Telemetry*

INT-based Packet processors (e.g., P4 [18]) enable the generation of monitoring data. In contrast to existing approaches INT based on P4 allows for the collection of network metrics (delay, jitter, BW, etc.) on a per-hop basis. Thus, QoS-related issues with a specific link can be pinpointed to a specific segment of the path allowing network services to address these issues with the goal to maintain the required QoS.

To further illustrate, Figure 5 depicts an INT implementation. At each of the programmable P4 switches INT data in the form of the outgoing queue length is collected and added to the packets traversing the link. At the egress point, this metadata is removed from the packet (before it is forwarded to $h2$) and analyzed. Queue sizes above a certain threshold might indicate that the required QoS can no longer be supported along this path. In this case, network services can be invoked to actively manage the network (re-routing, limiting of other traffic) to further guarantee the required QoS.

B. *Network services control and workflow evaluation*

As shown in Fig. 1, the FlyNet architecture is designed to operate on advanced network infrastructures like FABRIC [19]. The availability of programmable network elements in FABRIC support INT scenarios as shown in Fig. 5. The benefits of this approach can be demonstrated by a scenario in which a swarm of drones sends video footage from a search-and-rescue operation. Through the combination of INT and Multi-hop Route Inspection (MRI) a control system can be created that is aware of the entire network topology between IoT devices (swarm of drones) scenario, edge servers, and the cloud. It allows the detection of congestion within that topology and can actively intervene to prevent it [20].

Figure 6 depicts a scenario in which aggregated video streams from a swarm of drones are transmitted from edge

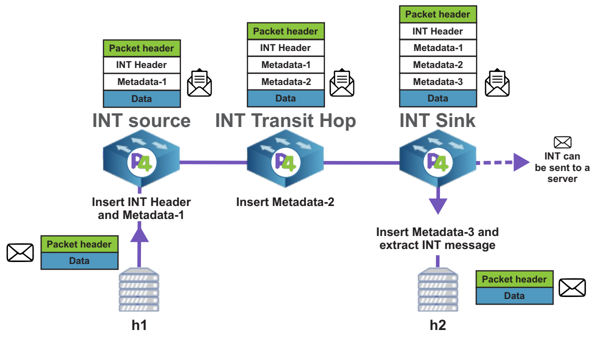

Fig. 5. Illustration of the application of INT where data is transmitted between hosts $h1$ to $h2$ using three programmable network switches; INT source, transit hop, and sink add headers to report the time spent in the outgoing queues across the network path.

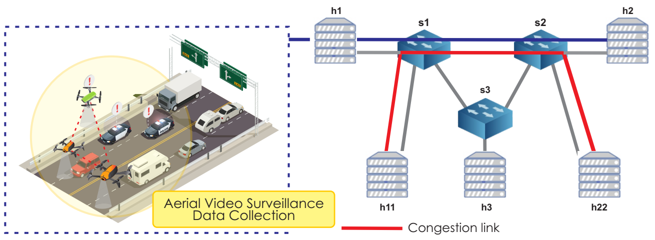

Fig. 6. Aerial video surveillance data collection use case scenario experiencing congestion bottlenecks without P4 programmable devices.

server $h1$ to cloud server $h2$ via $s1$ and $s2$. Due to competing traffic between $h11$ and $h22$ packet loss and delay can occur for the video stream. With the aid of INT the link on which this packet loss and delay occurs can be identified and MRI is invoked to re-route the competing traffic (from $h11$ to $h22$) via $s3$ mitigating the congestion on the $s1$ to $s2$ link.

As the results in Fig. 7 show, this INT-based network service (implemented via P4 in FABRIC) is able to guarantee QoS for the video streams generated by the swarm of drones. While there is significant packet loss when no INT is applied (cases 1 & 2), there is no packet loss when an INT-based network service is used (case 4).

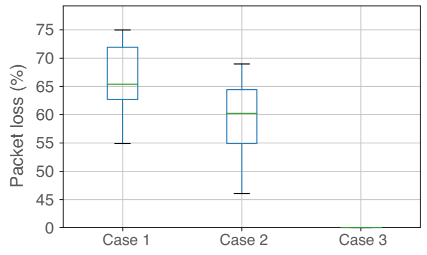

Fig. 7. Packet Loss measurements to show impact of increased congestion on the path between $s1$ and $s2$ with capacity of 200 Mbps for the cases: w/o P4 and Congestion of 800 Mbps (Case 1), w/o P4 and Congestion of 400 Mbps (Case 2), and with P4 (Case 3).

V. Conclusion

Unpiloted Aerial Vehicles (UAVs), often referred to as drones, have become an important enabler for a wide variety of scientific and societally-impactful applications. FlyNet supports these applications by providing automated resource allocation, workflow instrumentation, and network service management. It leverages the Pegasus workflow management system for supporting and managing scientific workflows spanning from the edge to the core cloud and Mobius, a resource provisioning system that can build a virtual edge-to-cloud platform. In combination with network services that are based on programmable network elements, FlyNet is able to allocate network and compute resources to optimize the execution of these UAV workflows. As a result, researchers can collect and efficiently analyze data, make scientific discoveries, or react to information coming from remote locations.

While we have created a platform that supports drone-based research, there are many research issues that still need to be addressed in the future. For example, the interdependency between data collection and offloading under uncertain network connectivity conditions has not been sufficiently studied. Resource provisioning, task scheduling, and fault recovery that takes into account a number of competing criteria including performance, reliability, and power are still challenging. We will address such research issues through the exploration of new algorithm design and experimentation with FlyNet on wireless testbeds like AERPAW [21].

In the future, we will utilize and extend the FlyNet platform to conduct new drone-based research – supporting new use cases like utilizing a network of drones for emergency management, using a network of edge computing systems to perform drone computations, and executing machine learning algorithms with varying computational requirements across the latency spectrum.

We also plan to harden, test, and expand its capabilities to make them available as part of the overall cyberinfrastructure ecosystem. This will allow scientists, engineers, and emergency managers to leverage FlyNet's capabilities for their work.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Award Numbers: CNS-1950873, CNS-1647182 and OAC-2018074. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] E. Lyons, G. Papadimitriou, C. Wang, K. Thareja, P. Ruth, J. Villalobos, I. Rodero, E. Deelman, M. Zink, and A. Mandal, "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.

[2] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.

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

[4] FABLib API, https://learn.fabric-testbed.net/knowledge-base/fablib-api/.

[5] Network Services in FABRIC, https://learn.fabric-testbed.net/knowledge-base/network-services-in-fabric/.

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

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

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

[9] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience." *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[10] "Prometheus," https://prometheus.io/docs/introduction/overview/.

[11] E. Deelman, K. Vahi, M. Rynge, R. Mayani, R. F. da Silva, G. Papadimitriou, and M. Livny, "The evolution of the pegasus workflow management software," *Computing in Science & Engineering*, vol. 21, no. 4, pp. 22–36, 2019.

[12] "The directed acyclic graph manager," https://research.cs.wisc.edu/htcondor/dagman/.

[13] R. Tanaka and G. Papadimitriou, "Pegasus synthetic edge workflow," Jan. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5889198

[14] G. Papadimitriou and S. C. Viswanath, "Pegasus casa wind workflow," Jan. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5889207

[15] "The orcasound project." [Online]. Available: https://www.orcasound.net/

[16] G. Papadimitriou, "Pegasus orcasound workflow," Jan. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5889225

[17] R. Tanaka, G. Papadimitriou, S. C. Viswanath, C. Wang, E. Lyons, K. Thareja, C. Qu, A. E. Morel, E. Deelman, A. Mandal, P. Calyam, and M. Zink, "Automating edge-to-cloud workflows for science: Traversing the edge-to-cloud continuum with pegasus," in *22nd IEEE International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2022, Taormina, Italy, May 16-19, 2022*. IEEE, 2022, pp. 826–833. [Online]. Available: https://doi.org/10.1109/CCGrid54584.2022.00098

[18] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[19] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.

[20] A. Esquivel Morel, P. Calyam, C. Qu, D. Gafurov, C. Wang, K. Thareja, A. Mandal, E. Lyons, M. Zink, G. Papadimitriou, and E. Deelman, "Network services management using programmable data planes for visual cloud computing," in *2023 International Conference on Computing, Networking and Communications (ICNC): Next Generation Networks and Internet Applications (ICNC'23 NGNI)*, Honolulu, USA, Feb. 2023.

[21] AERPAW: Aerial Experimentation and Research Platform for Advanced Wireless, https://aerpaw.org/.