Energy-aware Dynamic Computation Offloading for Video Analytics in Multi-UAV Systems

Jeromy Yu*, Aditya Vandanapu†, Chengyi Qu‡, Songjie Wang§ and Prasad Calyam§
*Department of ECE, Purdue University
†Department of CS, University of Illinois at Chicago
‡§ Departmant of ECE, University of Missouri - Columbia
Email: *yu816@purdue.edu, †avanda7@uic.edu, ‡cqy78@mail.missouri.edu, §wangso, calyamp@missouri.edu

***Abstract*—Multi-Unmanned Aerial Vehicle (UAV) systems with high-resolution cameras have been found to be useful for operations such as disaster management and smart farming. These systems feature Flying Ad-Hoc Networks (FANETs) that connect the computation edge with UAVs and a Ground Control Station (GCS) through air-to-ground network links. Leveraging the edge computation resources effectively with energy-awareness, and dealing with intermittent failures of FANET links are the major challenges in supporting video processing applications. In this paper, we propose a novel energy-aware dynamic computation offloading scheme for UAV systems, which provides the ability to intelligently share tasks among individual UAVs and allows for parallel execution of tasks while evenly distributing energy consumption. Intelligence gathering is performed using machine learning to create resource consumption profiles for a given set of video processing tasks prior to scheduling. Our scheme handles the problem of computation offloading tasks as a job-shop scheduling problem where we aim to minimize the total energy consumption in the edge resources while minimizing video processing times to meet application requirements. Our experimental results show our energy-aware dynamic offloading scheme enables lower processing time for low drone-to-ground server ratios and consumes less energy when compared to other offloading schemes. Notably, these results also hold in various other multi-UAV scenarios involving largely different number of detected objects.**

***Index Terms*—Multi-access Edge Computing, Flying Ad-Hoc Networks, Function-Centric Computing, UAV-based edge computing, Job Shop Scheduling**

I. Introduction

Autonomous Unmanned Aerial Vehicles (UAVs), also as known as drones, have been widely used in a large number of scenarios over the past decade. Their ability to carry different type of payloads such as sensors, high-resolution cameras and even high performance micro-processors provide unique capabilities for scientists and engineers [1]. In large scale applications, e.g., wide-area farms, highway traffic controls and mass-rallies management, a cluster or a network of UAVs can co-operate on intensive sensor/video data collection, streaming, processing, analytics, and object recognition/tracking. This has led to a new field of study called "Flying Ad hoc Networks" (FANETs) that integrate UAVs and Internet of Things (IoT) as communicating entities in high-demanding and large-scale applications [2]. FANETs support distributed wireless networks that solve the communication range restriction problem by allowing communication among the UAVs without the need for infrastructure [3]. There is no doubt that with the proper collaboration and coordination of multiple UAVS in the FANET, the system would far exceed the capacity and capabilities of single UAV systems.

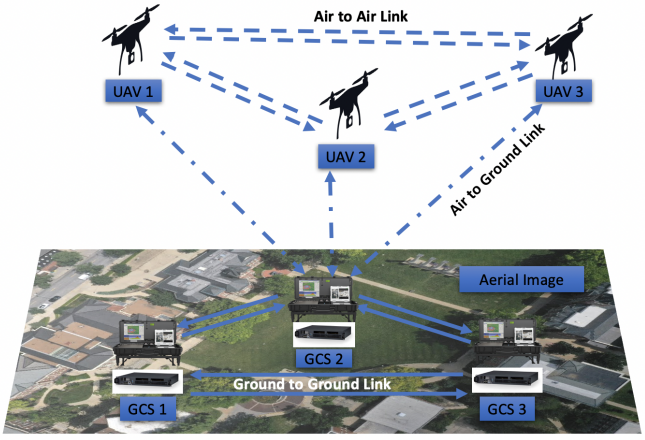


Fig. 1: FANET setup including multiple drones, ground control stations (GCSs) with edge server hosting; connected via wireless (dashes) and wired network (solid lines). The drones work together while recording the same geo-location at different angles and process the video locally onboard or they offload computation tasks to the GCSs.

Although there are several advantages of using FANETs over the single UAV-to-Ground Control Station (GCS) system architectures, several challenges emerge when implementing FANETs. Notable challenges particularly arise in systems with different requirements, such as total processing time, energy consumption, efficient communication and coordination between UAVs-UAVs and UAVs-GCSs based on mobility [4]. In addition, in recent drone-based computer vision applications, some visual data processing functions involve computation-intensive analysis of video streams. It is not practical for drones to handle these computations if visual data processing needs to be performed on-board. In this case, partial computation offloading or Function-Centric Computing [5] (FCC) strategies can applied on the edge servers and FANET setup.

This material is based upon work supported by the National Science Foundation under Award Number: CNS-1647182 and the Army Research Lab under Award Numbers: W911NF1820285 and W911NF1910181. 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 or the Army Research Lab.

Figure 1 shows an example of a FANET setup in a live video streaming scenario. In this setup, multiple drones capture videos of the same location at the same time, but from different angles. The drones are connected to each other, as well as to a ground control station (GCS) with an edge server via wireless connectivity (e.g., Wifi, Radio-over-IP). This wireless connectivity also provides the drones with the capability to communicate with peer drones. In addition, multiple GCSs on the ground are connected to each other and a core cloud via high-speed wired networks. When capturing live video, the drones will either locally execute the video processing tasks on their own hardware (on-board), or partially offload the tasks to the GCSs with edge servers.

In this paper, we present a novel *energy-aware computation offloading scheme* that minimizes the total energy consumption in the edge devices while achieving video processing time that fulfills the requirements of an application that uses the FANET setup shown in Figure 1. This scheme features the FCC computing paradigm [5] by decomposing a video analytics application into functions/tasks that can be individually deployed onto either drones or edge servers on GCS to maximize performance while reducing energy consumption on the drones. The contributions of this paper can be summarized as follows:

- We detail a novel energy-aware computation offloading scheme, which features the FCC paradigm, to optimize the trade-offs in energy, processing latency, and task scheduling time among different computing architectures used in different video processing applications.
- We describe a drone video analytics application pipeline that supports FCC and can be decomposed into into a chain of microservice functions. The functions can be deployed by knowledge of the computing capacity thresholds on the drone or edge servers, communicating via RESTful APIs.
- We simulate a multi-UAV/GCS environment that spans multiple network environments and utilizes the state-of-the-art edge resources on the system to evaluate different offloading strategies and show FCC benefits with realistic application settings.

The remainder of the paper is organized as follows: Section II presents related work. In Section III, we discuss our multi-UAV system model in general, and describe our drone/Edge system control modules. In Section IV, we detail our novel intelligent energy-aware computation offloading scheme for real-time drone video analytics. Section V describes our testbed-based evaluation methodology, performance metrics and results. Section VI concludes the paper.

II. Related Work

Flying Ad-Hoc Networks (FANETs). FANETs are used to overcome the challenges of communication in a multi-UAV environment. Authors in [6] propose a system where UAVs can connect with each other rather than having all the UAVs connect to a single GCS or a satellite. Based on advanced routing protocols, even if connection among one of the drones to the GCS is interrupted or disconnected, communication may still be possible [7]. Thus, in our FANET model, we assume the setup to be a complete graph where all the drones are connected to each other as well as connected to an edge server that hosts situational-awareness dashboards with geolocation services [8], [9] at the GCS. We use a FANET setup because we want the ability of a drone to offload functions either to nearby drones or to a distant GCS.

Computation Offloading on Multi-UAV Systems. Computation Offloading is the act of transferring computationally intensive tasks or functions to other platforms. In our case, the drones have the option of transferring to another drone or the GCS. Low power devices trying to execute computation-intensive tasks will often consume more energy than what is preferred by the user. Energy-awareness is especially important for drones since the development of battery capacity has stagnated, and we want to have the drones flying in the air for as long as possible for data collection [10]. Computation offloading overcomes such limitations of low-power devices such as drones [11]. With computation offloading, energy consumption of the drones can be minimized as shown in [5], without compromising the user quality of experience at the application level [12]–[14]. In our proposed scheme, functions will be offloaded from the drones to the edge as a way to minimize energy consumption of the drones as long as the delay constraint is met. In addition, drones will offload functions to other drones to evenly distribute the energy consumption of the drones.

Function-Centric Computing (FCC). A FCC paradigm involves decomposing applications into microservice functions that can be individually deployed onto edge resources [5]. Advantages to this architecture are that specific functions in a larger application that are computation-intensive and would cause large energy consumption can be offloaded. In the prior work [15], the authors used FCC for data collection from IoT devices at the edge of a network and for data processing at the edge and cloud infrastructure. For our work, we use FCC to decompose a frame-processing application for object detection. Our decoupled application will contain four functions: F1 – pre-processing, F2 – object detection, F3 – feature extraction, and F4 – object registration. All four of these functions can be offloaded to either other drones or a GCS-connected edge server.

III. System Model

In this section, we introduce our energy-aware dynamic computation offloading system model featuring multiple UAVs as shown in Figure 2. Our system model consists of multiple UAVs with embedded computation resources, an Execution Control Module middleware, and a GCSs group with edge server computation capabilities.

A. Drone Fleet with Camera and Computation Resources Embedded and GCS group with Edge Server Computation

The first part of our system model's application involves the communication and co-ordination of multiple UAVs. Each

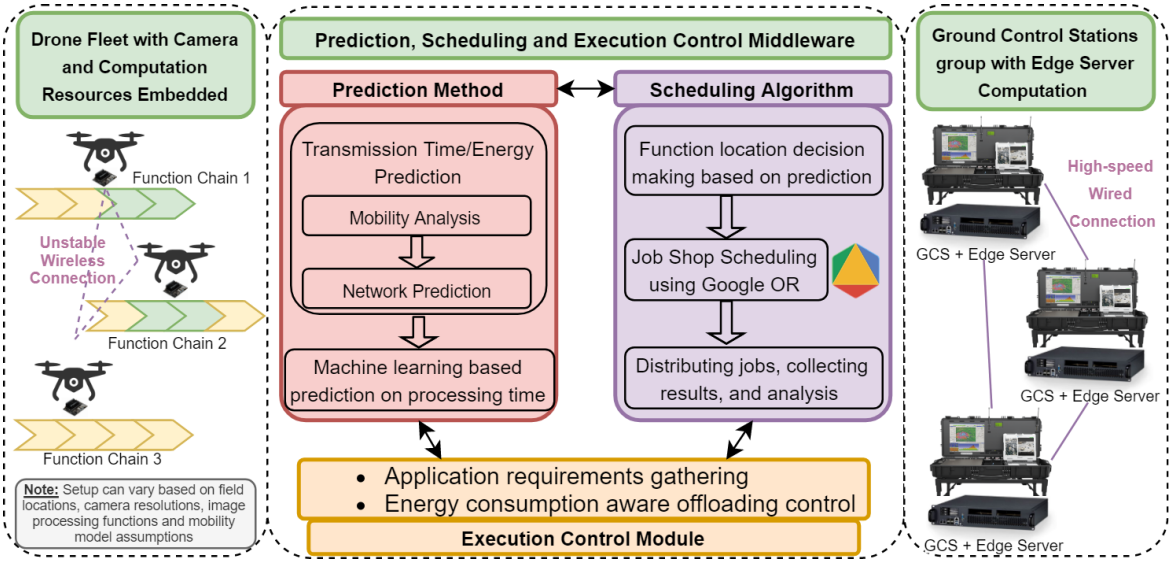


Fig. 2: Energy-aware Dynamic Computation Offloading System Model - our prediction method and scheduling algorithm can be part of the middleware to orchestrate data processing between the UAVs and the Ground Control Stations.

of the UAVs are assumed to have a camera (with varying quality and resolution) to record scene video, and the drones have the computation ability to execute the functions in the corresponding image processing pipeline. In addition, we not only assume that the camera resolutions may be different, but that the setup will vary on the field in which it is used as per the application demands. This is an important assumption because the drone(s) needs to decide whether or not to offload the image processing functions, depending on its network connection quality (stable vs. unstable wireless connection with available network bandwidth bottelenecks). Another dominant factor is the mobility model determined by the probability parameter that indicates whether the drone is near or far away from a given edge server. The next part of our system application model assumes the Ground Control Stations are connected with a wired high-speed network connection. This assumption is essential for executing the highly computation-intensive tasks, that cannot be executed on the drone due to the issue that it would consume the most amount of energy from the drone. In our system model, we consider a setup of the multiple UAVs with one GCS connected to an edge computation server for the execution and offloading of the object detection application functions.

B. Execution Control Middleware with Prediction method and Scheduling Algorithm

The second part of our system model's application involves the Execution Control Middleware that is responsible for making sure that the application requirements are all met through data collection with the necessary (minimal) energy consumption that is achieved by controlling how/where the functions will be offloaded. The middleware essentially consists of two parts: our proposed novel Prediction Method (i) and a Scheduling Algorithm (ii). The prediction method consists of the Transmission Time/Energy Prediction, and the Machine Learning Prediction - based on the processing and transmission times from the previous frames. For Transmission Time and Energy Prediction, we analyze what is the best mobility model that we could utilize based on the available network environment. After collecting the transmission times and processing times in conjunction with the corresponding energy predictions, we use several machine learning models to make a time prediction based on the processing completion, and the transmission time needed per frame. In the Scheduling Algorithm, we utilize the prediction method results and determine where the function should be processed on the drone or the GCS-edge server. Our scheduling algorithm uses the popular Job Shop Scheduling to schedule the tasks on multiple machines. Finally, we collect the time it took to execute all the tasks and compute the consumed energy for later comparison and evaluation.

IV. Prediction Method and Scheduling Algorithm

In this section, we detail the two major parts of our novel energy-aware dynamic computation offloading scheme i.e., the prediction method and the scheduling algorithm. These two parts allow for creation of a decision making scheme to not only facilitate trade-offs in energy vs. optimal scheduling time factors of dynamic decision making on multi-drone to GCS, but also aid in decisions pertaining to the pertinent data computation architecture, i.e., selection of either drone, edge server or FCC for data processing.

A. Prediction Method

1) Mobility Analysis: We use three mobility models in this work shown in Figure 3, based on a literature survey [16]: (i) Gauss-Markov Mobility Model, (ii) Random Way Point Mobility Model, and (iii) Mission Plan Based Mobility Model. By using these mobility models, we can analyze the network environment in terms of how well the connection quality varies

in a FANET setup. The Gauss Markov Model simulates the swarm behavior and it shows a probability parameter for the mobility. The probability parameter is defined as ρ and if ρ = 0, it will be using the Random Way Point model, in which the drones move at different directions and speeds randomly in the projected area. Equation 1 represents the speed as well as the direction calculations of a drone:

$$v_n = \rho v_{n-1} + (1-\rho)\bar{v} + \sqrt{(1-\alpha^2)v_{x_{n-1}}} \quad (1)$$

These UAVs are independent from the other UAVs as they have different speeds and directions. However, if ρ = 1, then these drones will use the Mission Plan based model. These drones have a projected plan on where the UAVs destination will be at, and these drones will be programmed to have a pre-determined path without manually having a user operating the UAVs paths. The pre-programming not only helps a UAV to move towards a destination, but it can also direct it to move away from the projected area as well as desired by the application demands. Also, the starting and ending point are randomly selected because the UAV needs to know its plan in the given projected area for the flight. Furthermore, the flight time and velocity are fixed numbers to determine how long the UAV will stay in the sky, and how fast it covers a given area for a planned mission puropose.

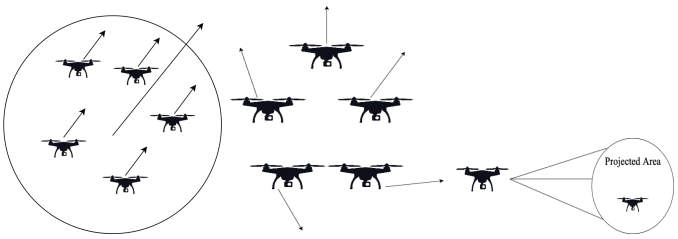


Fig. 3: Three major Drone Mobility Models we consider: (a) Gauss-Markov Mobility Model, b) Random Way Point Mobility Model, and (c) Mission Plan based Mobility Model.

2) Machine Learning Prediction: Once we gather our transmission and processing master dataset files (with a .csv extension) considering the mobility models, we use machine learning models to predict the time on how well the dataset performs on various models. In addition to that, we use machine learning to also predict the time for completion of the four functions (F1 – pre-processing, F2 – object detection, F3 – feature extraction, and F4 – object registration) in each of the frames in the dataset. The four machine learning models that we use for data processing are from the Sci-Kit Learn Tool [17] are as follows: (i) Kernel-Ridge Regression, (ii) SVR-RBF, (iii) Gaussian-Process Regression, and (iv) Random Forest Regression. Training Time, Prediction Latency, RMSE (Root Mean Square Error), and MAE (Mean Absolute Error) are generated using the Sci-Kit Learn Tool for each model as shown in Table I.

In our machine learning use for the data processing, we are primarily concerned about the training time and RMSE metrics. The reason for the training time consideration is that - we want to see how the dataset competes among other machine learning models for how long it takes to process the dataset files i.e., the processing and transmission times. The other reason for the RMSE consideration is that - we want to explore how concentrated our data is, and how flawed the standard deviation predictions are when using the various machine learning models. For transmission time results, we find that the best machine learning model that had best performance in the training time is the Kernel-Ridge Regression model. Additionally, for the processing time results, the best machine learning model performance was seen in the SVR-RBF.

TABLE I: Machine Learning model results on our datasets.

Model Type	Transmission Time	Processing Time
Kernel-Ridge Regression	**0.120±0.00362**	0.004±0.00272
SVR-RBF	0.099±0.01620	**0.068±0.00252**
Gaussian-Process Regression	2.170±0.00100	1.962±0.00000
Random Forest Regression	0.205±0.05400	0.156±0.05400

B. Scheduling Algorithm

1) Decision Making: To design the decision making policies of our computation offloading scheduler, we created a set of heuristics based on best practices and a set of field experiments. For functions F1 and F4, it was found that the transmission energy to offload to the GCS was always greater than the processing energy to execute F1 locally (i.e., 1.173 J vs. 3.762 J).

To determine whether to offload F2 or not, is best determined by the network condition (i.e., upload speed) and the number of F2 tasks on the queue. F2 would be offloaded to the GCS until the case where the queue on the GCS is filled up. The number of tasks allowed on the queue of the GCS is best determined by the condition of the air-to-ground network. The higher the upload speed, the more tasks that will be allowed on the queue of the ground, and the reverse is true. A slower network speed implies that fewer number of tasks will be allowed on the queue. The reason for this is because - for slow network speeds, the transmission energy increases because it is tied to the transmission time. If a transmission takes long enough, eventually the transmission energy will surpass the local processing energy consumption. The network condition in our heuristics is based on the predictions obtained by the mobility models chosen, and how many tasks are allowed on the queue of the GCS at a given network condition is determined by the machine learning predictions of processing time and energy consumption.

Moreover, processing energy for F3 is tied to the number of objects detected in F2. When the number of objects detected is greater than 10, we found that the processing energy to perform F3 is greater than the energy to transmit the task to the GCS. In cases such as these, we offload F3 to the GCS.

2) Job Shop Scheduling: We treat the problem of offloading functions to the drones as a job shop scheduling problem. The job shop scheduling problem is a problem that occurs when multiple jobs are processed on multiple machines in

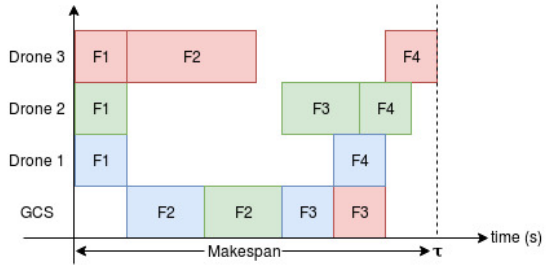


Fig. 4: An example Job Shop Scheduling algorithm schedule with one GCS and three drones. Each function chain must be executed sequentially from F1 to F4, and functions F1 and F4 must be executed on the drones themselves. Blue represents a function chain belonging to drone 1, green represents a function chain belonging to drone 2, and red represents a function chain belonging to drone 3. τ represents the optimal scheduling time or the scheduling makespan.

which each job has multiple tasks that must be done sequentially [18]. The heuristics established from the decision making (listed in previous sub-section) serve as constraints for the job shop scheduling algorithm. Since we are dealing with video streams, it is important that the frames are processed sequentially. We adopted and extended the job shop algorithm that is openly available from the Google Optimization toolkit (i.e, OR-Tools). Using this algorithm, we distribute the jobs among the machines and allow for calculation of the optimal scheduling time as well as the energy consumption rate of the drones to execute all tasks of all jobs on all machines. An example schedule generated by our algorithm is shown in Figure 4, where certain functions of a drone's function chain are offloaded to the GCS while others are processed locally on the drones. Following the constraints set in the decision making based on heuristics, F1 and F4 are always processed on the respective drones; F2 is determined from the machine learning predictions, and F3 is reliant on the results of F2.

V. Performance Evaluation

In this section, we first describe the experiment setup and the data collection experiments. Following this, we present the results discussion along with the salient findings.

A. Experiment Setup and Data Collection

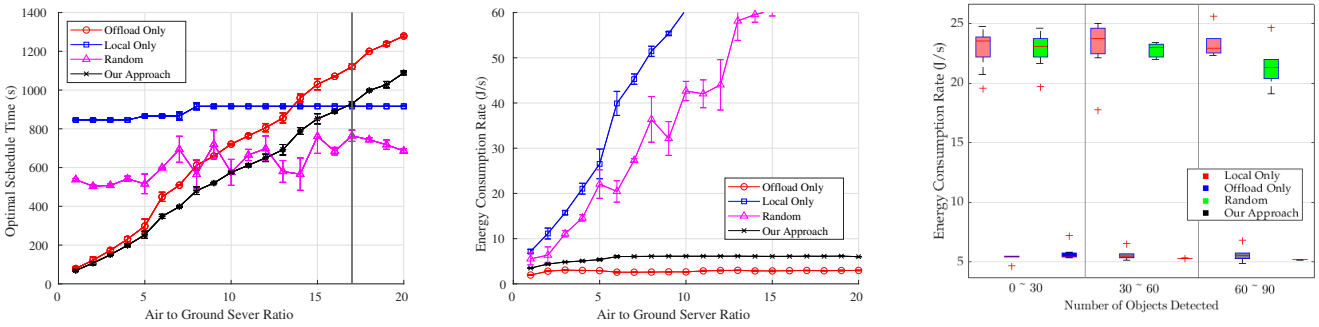
In our experiment setup, we first focus on the mobility model that represents the movement of the nodes. The main role of the mobility model is to implement a realistic environment and to evaluate network parameters in different geo-spatial locations. We specifically use the Gauss Markov Mobility Model to simulate the UAV behavior in a swarm as detailed in [19]. We assume that each UAV travels to the involved area and then returns to the GCS. Initially, by applying the Gauss Markov Mobility Model, each UAV is assigned a current speed and direction. During fixed intervals of time, movement occurs by updating the speed and direction of each of the UAVs. Specifically, values of speed and direction at a moment n is calculated on the basis of the values of speed and direction at moment n-1. The Gauss-Markov Mobility model inherently can easily eliminate sudden stops and sharp turns by allowing past velocities (and directions) to influence future velocities (as well as directions) [20]. Based on literature survey [16], we set up our memory level parameter α to be 0.85 which indicates the Gauss-Markov Model has some memory. The velocity at current time slot depends upon both its velocity at time n-1 and a new Gaussian random variable. Since the α value is nearer to 1, the current velocity is more likely to be influenced by its previous velocity, which is also reasonable for the settings of a FANET.

As part of the experiments to measure the transmission and processing energy consumption with regards to image resolution and the number of objects detected, we used the VisDrone2019 dataset [21]. This dataset is made up of multiple live streams that were recorded by the Lab of Machine Learning and Data Mining in Tianjin University, China. In the dataset, several live streams were recorded at the same time and same location by different drones at different angles of view of the same location. For each live stream, we determine whether the live stream was densely or sparsely populated with objects. Note that we defined objects as pedestrians for our study purposes. Thus, a live stream with many people is dense, while a live stream with few people is sparse. The dataset we considered has three different resolutions (1344x756, 1902x1071, 2688x1322) with 50 images per resolution in 20 datasets of livestream recordings. We process each live stream through an impage processing pipeline implemented as a Python script. The pipeline includes a pre-trained model designed for pedestrian detection running on the Tensorflow framework. The object detection function is able to detect how many pedestrians there are for each frame of the live stream. The function provides information about the resolution of the live stream along with the total energy it takes to process a given live stream.

For transmission energy data collection, a computation environment was created with a desktop serving as the GCS connected to an edge server. A Nvidia Jetson Nano was used as a drone device connected to the GCS-edge. The Nvidia Jetson Nano that we used has a CPU with clock speed of 1428 MHz, GPU with clock speed of 33 MHz, and 547 MB of free memory. The CPUs we used have the following configurations: Intel(R) Core(TM) i5-6200U CPU that has a clock speed of 2401 MHz and 1633 MB of free space, and the Intel(R) Core(TM) i5-3470 CPU which has a CPU clock speed of 3370 MHz, with GPU Xeon E3-1200 v2 with GPU clock speed of 33 MHz, and 5049 MB of free memory. The wireless network chips we used include the Intel 8265NGW, Edimax EW-7811Un, and Cypress CYW43455.

B. Results Discussion

Table II shows the features that were collected during our experiments, including networking hardware, processing capabilities of the experiment machines, and attributes of the data to be offloaded. We obtained the transmission energy results based on the calculations given in [10]. In addition, we calculated the energy consumption for processing the dataset using the various hardware settings mentioned above in the previous sub-section.



(a) (b) (c)

Fig. 5: Offload only, local execution (drone) only, random approach and our approach results of: (a) increasing drone to ground ratio (Drone:GCS) and optimal schedule time (s), (b) Drone:GCS and average energy per drone (Joules), and (c) number of objects detected and Energy Consumption Rate (Joule/s) box plots (experiments assume a Drone:GCS of 5).

TABLE II: Collected features from our experiment dataset.

Transmission Dataset	**Processing Dataset**
Data Attributes:	
Video Resolution (width × height)	Video Resolution (width × height)
Hardware Parameters:	
Wireless Data Rate (Mb/s)	CPU Cloud Speed (MHz)
Number of Spatial Streams	RAM (GB)
Code rate (Mb/s)	Free Memory (MB)
Number of bits per symbol	Hard Drive Capacity (MB)
Mobility Models:	
Gauss-Markov, Random Way Point, Mission Plan based	
Result Parameters:	
Transmission Energy (J)	Processing Energy (J)

1) Our knowledge-based offloading scheme outperformed local-execution in terms of scheduling time for low drone-to-ground-server ratios: As shown in Figure 5(a), our approach follows a similar trajectory to offload all of the functions to the ground (i.e., the GCS connected to the edge server). The difference is that our approach always outperforms the offload-only scheme. We found that our offloading scheme resulted in up to 15% better scheduling makespan than using the offload-only scheme. Random offloading is sporadic with no definable correlation and contains a far greater degree of variance compared to the other offloading schemes. The optimal scheduling time when processing is performed only on the done locally is very consistent, even when the drone to ground server ratio changes. This is because of the parallel execution of the tasks done on the drones, with each drone processing its own video stream without ever offloading to the ground. Therefore, there is no build up of the queue on the ground. Our approach consistently outperformed simply processing the tasks locally on the drones until the drone to ground ratio reached about $17:1$. This phenomenon happens because as more drones are added to the server with each drone recording a live stream, more tasks will naturally be offloaded to the ground. Incidentally for large ratios, the queue on the ground will grow very large, meaning that it will take a long time for all tasks on the queue to be completed.

2) Our approach outperformed random offloading and only local execution in energy consumption: From Figure 5(b), we can see that the energy consumption rate grows at a very fast rate as the air to ground server ratio increases. Random offloading also increases at a fast rate. However, the results are still somewhat sporadic since the energy consumption rate at a higher air to ground ratio will sometimes be lower than the energy consumption rate for a lower air to ground ratio. Our approach consistently produced a lower energy consumption rate than only local execution and random offloading, but it was still slightly more than the energy consumption rate of the offload only scheme. This is reasonable because the only energy consumption counted for the offload only scheme is the transmission energy. That being said, the difference in energy consumption rate between our approach and the offloading scheme is fairly marginal. Considering that our approach performs up to 15% better than the offload only scheme in terms of scheduling time, we conclude that our approach creates a good trade-off between both time and energy considerations in completing the tasks execution.

3) Our approach works for various application scenarios involving multi-UAV systems: The results in Figure 5(c) show that the number of objects detected in a video stream does not have a major impact on the energy consumption rate, regardless of the schedule offloading scheme. This finding suggests that our heuristics-based offloading scheme will work in various application scenarios regardless of the number of target objects in an area being explored for environmental situational awareness. For various application scenarios, such as smart farming where the number of target objects such as e.g., ripe fruits may be small in quantity, or for traffic flow monitoring applications where there may be a medium number of automobiles on the freeway, or for a disaster responses in an area with debris everywhere after a man-made or natural disaster incident, our approach would meet the corresponding application demands for drone video analytics.

VI. Conclusion

In this paper, we presented a novel energy-aware Function-Centric Computing offloading scheme that would allow for parallel execution to evenly distribute and minimize computation energy of the drones while maintaining a reasonable delay. As part of our scheme development, we considered

mobility of drones and dynamically changing network environments, and used machine learning to predict processing times. Furthermore, our proposed dynamic computation offloading approach successfully used a job shop scheduling to minimize the optimal scheduling times (i.e., makespan) and energy consumption. As part of our performance evaluation, we simulated a multi-UAV system featuring a FANET and a VisDrone2019 dataset to show benefits of Function-Centric Computing by considering a trade-off between energy consumption rate and optimal scheduling time. Finally, we demonstrated the benefits of our approach for different application scenarios of multi-UAV systems and the FANET setups such as: smart farming, traffic flow monitoring in transportation systems and disaster response after a man-made or natural disaster incident.

Future work could involve measuring the performance of our scheme under different mobility conditions (e.g., assuming drones being stationary or moving on a pre-determined path or random path) and using real-time network feedback and GCS load conditions as additional parameters to further optimize the video processing task scheduling for an application scenario.

REFERENCES

[1] B. Dickson, "When the cloud is swamped, it's edge computing, ai to the rescue," Accessed August 2019. [Online]. Available: https://www.pcmag.com/article/360311/when-the-cloud-is-swamped-its-edge-computing-ai-to-the-re

[2] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (fanets): A survey," *Elsevier Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[3] I. Bekmezci, I. Sen, and E. Erkalkan, "Flying ad hoc networks (fanet) test bed implementation," in *2015 7th International Conference on Recent Advances in Space Technologies (RAST)*. IEEE, 2015, pp. 665–668.

[4] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.

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

[6] A. Guillen-Perez and M.-D. Cano, "Flying ad hoc networks: A new domain for network communications," *Sensors*, vol. 18, no. 10, p. 3571, 2018.

[7] H. Yang and Z. Liu, "An optimization routing protocol for fanets," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 120, Accessed August 2019. [Online]. Available: https://doi.org/10.1186/s13638-019-1442-0

[8] J. Gillis, P. Calyam, A. Bartels, M. Popescu, S. Barnes, J. Doty, D. Higbee, and S. Ahmad, "Panacea's glass: Mobile cloud framework for communication in mass casualty disaster triage," in *MobileCloud*. IEEE, 2015.

[9] M. Vassell, O. Apperson, P. Calyam, J. Gillis, and S. Ahmad, "Intelligent dashboard for augmented reality based incident command response co-ordination," in *Consumer Communications Networking Conference (CCNC)*. IEEE, 2016.

[10] B. Dab, N. Aitsaadi, and R. Langar, "Q-learning algorithm for joint computation offloading and resource allocation in edge cloud," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 45–52.

[11] N. H. Motlagh, M. Bagaa, and T. Taleb, "Uav-based iot platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, February 2017.

[12] P. Calyam, M. Haffner, E. Ekici, and C.-G. Lee, "Measuring interaction qoe in internet videoconferencing," in *IEEE/IFIP Management of Multimedia and Mobile Networks and Services (MMNS)*. IEEE, 2007.

[13] A. Sukhov, P. Calyam, W. Daly, and A. Ilin, "Towards an analytical model for characterizing behavior of high-speed vvoip applications," *Computational Methods in Science and Technology Journal*, vol. 11, no. 2, 2005.

[14] P. Calyam, P. Chandrasekaran, G. Trueb, N. Howes, R. Ramnath, D. Yu, Y. Liu, L. Xiong, and D. Yang, "Multi-resolution multimedia qoe models for iptv applications," *International Journal of Digital Multimedia Broadcasting (IJDMB)*, 2011.

[15] D. Chemodanov, R. Gargees, B. Morago, P. Rengarajan, P. Calyam, Z. Oraibi, Y. Duan, G. Seetharam, and K. Ş. Palaniappan, "Flying ad-hoc networks (fanets): A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 1, pp. 182–197, 2016.

[16] F. Bai and A. Helmy, "Chapter 1 a survey of mobility models in wireless adhoc networks," Available: https://www.cise.ufl.edu/ helmy/papers/Survey-Mobility-Chapter-1.pdf, Accessed August 2019 [Online].

[17] "scikit-learn 0.21.2 documentation." [Online]. Available: https://scikit-learn.org/stable, Accessed August 2019.

[18] "The job shop problem — or-tools — google developers." [Online]. Available: https://developers.google.com/optimization/scheduling, Accessed August 2019.

[19] K. Kumari, B. Sah, and S. Maakar, "A survey: different mobility model for fanet," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5, no. 6, 2015.

[20] D. A. Korneev, A. V. Leonov, and G. A. Litvinov, "Estimation of mini-uavs network parameters for search and rescue operation scenario with gauss-markov mobility model," in *2018 IEEE Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, July 2018, pp. 1–7.

[21] P. Zhu, L. Wen, X. Bian, L. Haibin, and Q. Hu, "Vision meets drones: A challenge," *arXiv preprint arXiv:1804.07437*, 2018.