FlyPaw: Optimized Route Planning for Scientific UAV Missions

Andrew Grote[§] Eric Lyons[§], Komal Thareja[*], George Papadimitriou[†], Ewa Deelman[†], Anirban Mandal[*], Prasad Calyam[‡], Michael Zink[§]

[*]RENCI, University of North Carolina at Chapel Hill, NC, USA [†]Information Sciences Institute, University of Southern California, CA, USA [‡]Electrical Engineering and Computer Science Department, University of Missouri Columbia, USA [§]Electrical and Computer Engineering Department, University of Massachusetts Amherst, MA, USA

2023 IEEE 19th International Conference on e-Science (e-Science) | 979-8-3503-2223-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/E-SCIENCE58273.2023.10254831

***Abstract*—Many Internet of Things (IoT) applications require compute resources that cannot be provided by the devices themselves. On the other hand, processing of the data generated by IoT devices and sensors often has to be performed in real- or near real-time, i.e., with stringent latency requirements in constrained environments (e.g., intermittent network connectivity and limited power envelopes). Examples of such scenarios are autonomous vehicles in the form of cars and drones where the processing and analysis of observational data (e.g., video feeds) need to be performed expeditiously to allow for safe operation of the vehicles and to deliver the results in a timely fashion to the stakeholders of the mission. To support the compute and timeliness requirements of such applications, it is essential to include suitable edge resources to process these workflows, and to develop an end-to-end system that can route the vehicles dynamically and process and deliver mission-critical data and analyzed results. In this paper, we develop and evaluate a dynamic scheduling approach that considers complex tradeoffs between real-time constraints, network availability, and latency sensitivity of the mission. We devise an optimized route planning and data transmission schedule for drone flights. The scheduling algorithm is encapsulated in a novel end-to-end architecture (FlyPaw) and an associated adaptive drone mission control system, which enables deployment and management of an integrated cyberphysical system (CPS) – from real drone testbed to base stations to edge-to-cloud resources. The planning algorithm takes into account measured network communication characteristics, estimated uncertainties of future data link connectivity, and data timeliness requirements of the mission to prioritize candidate decision tree solutions based on a risk metric derived from Sharpe's ratio. Our results show that for given task sets, Net Time to Retrieve, our metric describing the time required to perform end-to-end collection and downstream processing of data, can be significantly reduced compared to other naive approaches. The theoretical improvement provided by our algorithm over other naive approaches is dependent on several factors — task locations, network connectivity, processing times and available resources, and is bounded by the duration of the drone flight.**

***Index Terms*—edge computing, network-centric platform, cloud and edge resource provisioning, network management, workflow automation, Unpiloted Aerial Vehicle (UAV) systems, drone video analytics**

I. Introduction

We live in an era of data-driven research, which has been fueled by the success of the Internet of Things (IoT) and its many applications. This wide-ranging proliferation of sensors (from smart thermostats in homes to networks of telescopes [1]) results in large amounts of data. Data streams from IoT sensors have become so vast that transmitting them to and processing them only at central sites (potentially cloud data centers) is not feasible. As a result, many applications rely on a combination of edge and cloud computing resources.

In contrast to cloud computing, edge computing offers smaller amounts of computing resources with the benefit of low latency communication and the possibility of aggregating sensor data streams before passing information on to core cloud data centers. These characteristics make edge computing attractive to data-driven science applications that make use of sensors and require high-volume data processing and low latency.

The combination of sensing, edge computing, and cloud computing resources requires the creation and management of complex workflows to support scientific and commercial, data-driven applications. This includes the processing, analyzing, and storing of data, where workflows utilize an arbitrary set of networking, computing, and storage resources. Planning and executing such workflows is a complex task, which not only requires the orchestration of resources in a timely, robust, and efficient manner but also has to be optimized for performance, energy, resilience and other unique attributes of these sensor data-driven applications.

This paper focuses on edge-to-core workflows for Unpiloted Aerial Vehicles (UAV), a particular class of sensor data-driven applications. We have selected these applications since UAVs have become either the subject of research themselves or are used by researchers to provide scientific observations, ranging from environmental monitoring, disaster response, and wildfire monitoring, to the survey of archeological sites [2]–[5]. UAV-based applications also represent a category that is challenged by intermittent network connectivity and highly fluctuating throughput, stringent power consumption requirements, in addition to often being mission critical.

In this paper, we present an approach that takes these challenges into account to *devise a combined route planning and data transmission scheme for UAVs, which is cognizant of the objectives of the overall data collection task and available network and computing resources.* Our work is motivated by an optimization problem that determines the best UAV flight path for a specific mission, under the constraints of limited power/connectivity and the timely delivery of data. In this work, we address this challenge by developing a dynamic scheduling approach that is informed by active network performance measurements.

In scenarios in which UAVs are used to provide real-time data (e.g., video footage of a natural disaster), there is a trade-off that has to be taken into account when planning the route of the vehicle. While a mission plan might require a drone (a particular type of UAV) to intersect with specific waypoints in a specific order, the latter might be changed if it improves the overall delay between data capture, transmission, and processing. This might lead to non-

intuitive routes, where a drone backtracks to a certain waypoint where there is high confidence in available network connectivity. Such behavior would be based on the fact that backtracking will result sooner in re-establishing network connectivity than continuing on the originally planned route. Obviously, such backtracking comes with increased power consumption and an increase in overall mission execution time. *In this paper, we develop and evaluate a dynamic scheduling approach that considers these complex, dynamic, inter-dependent, often conflicting factors to determine an optimized route plan and data transmission schedule for UAV flights.*

The approach presented in this paper is based on our previous work on FlyNet [6], which extends an existing workflow management system [7] by automated resource provisioning in the edge-to-cloud continuum [8]–[10], workflow instrumentation [9], and network service support [11]. The work presented in this paper builds on these capabilities and innovates in a number of important areas. Overall, this paper makes the following contributions:

- **FlyPaw architecture.** It presents *a novel end-to-end architecture (FlyPaw) and an associated adaptive UAV mission control system, which enables deployment and management of a complex cyberphysical system (CPS) - from drones to base stations to edge-to-cloud resources* by leveraging a real drone testbed — AERPAW [12], supported by the National Science Foundation Platforms for Advanced Wireless Research (PAWR) initiative [13]. This novel architecture enables drone route planning under the consideration of mission-critical tasks and network communication characteristics.
- **Multi-objective, dynamic UAV route planning optimization algorithm.** We present and evaluate a new algorithm that performs *in-flight route planning for a UAV-based sensing system, which is driven by the uncertainty of future data link connectivity*. Based on this information, the algorithm can modify the path of the UAV in-flight to meet application-critical deadlines. This route planning algorithm takes into account measured network communication characteristics, estimated uncertainties of future data link connectivity, and data timeliness requirements of the mission to prioritize candidate decision tree solutions based on a risk metric derived from the Sharpe's ratio [14]. Our results show that for given task sets, the *Net Time to Retrieve*, our metric describing the time required to perform end-to- end collection and downstream processing of data, can be significantly reduced.

The remainder of the paper is organized as follows. Sect. II introduces the end-to-end FlyPaw architecture and theassociated drone control system. In Sect. III, we present the time optimized planning routine and a thorough evaluation of the system. We present the related work in Sect. IV. Sect. V concludes the paper.

II. FlyPaw Architecture

A. AERPAW

The Aerial Experimentation and Research Platform for Advanced Wireless (AERPAW) [12] is a National Science Foundation supported research testbed at North Carolina State University funded under the Platforms for Advanced Wireless Research (PAWR) initiative [13]. The AERPAW team developed a system that enables researchers to leverage semi-autonomously controlled drone and ground-based rover vehicles, field deployed compute resources, and several types of software defined radios including 4G LTE and 5G transmitters and receivers. AERPAW has also developed an emulator for software implementation and testing in advance of submission for live flights with real vehicles. In partnership with RENCI at the University of North Carolina Chapel Hill, AERPAW has created a management platform for users to register and create their own experiments, including the ability to share results and even code with other system users. The AERPAW system allows for many types of notional experiments utilizing different hardware and software, and also permits users to take control of the vehicle and compute systems to perform their own experiments. To ensure legal compliance with FAA regulations, AERPAW provides a software operator oversight layer that ensures vehicles cannot be instructed to perform unsafe or illegal actions such as flying too high or landing too fast, and can always be manually instructed to return home. Aside from that, researchers are given the freedom to design and execute their experiments as they wish.

AERPAW's canonical experiments utilize preplanned flight trajectories designed using the QGroundControl software system for remote vehicle management [15]. Vehicles proceed in a largely scripted manner with background processes running continuously or through a cron job scheduler. In the first external user-performed experiment using the AERPAW system, in an effort to characterize the limits of the 4G LTE software defined radio network, we executed a live flight around the perimeter of the test field, periodically making command line-based *iPerf* network performance measurements. In our experiment, the vehicle exceeded the range limit of the 4G LTE network a short time after take-off and did not regain it until the perimeter flight was nearly complete. Although the vehicle was able to fly its path correctly in spite of the lost datalink, no application layer data could be exchanged over the bulk of the flight duration. This experiment provided insight into the limits of the LTE software defined radio, and importantly, led us to design a system where the air vehicle can dynamically account for situations where the datalink is working and where it is degraded or lost, with an on board plan to handle both scenarios.

B. The FlyPaw Software System

FlyPaw is the FlyNet project's Python-based software architecture for AERPAW, open sourced and available to researchers [16]. FlyPaw is intended to provide programmatic interfaces to vehicle control and a template to integrate application processes within a state machine framework. The system facilitates semi-autonomous operations and task performance by the AERPAW UAV by accounting for safety checks and dynamic states common to many experiments. The framework is divided over multiple platforms, with modules running on the mobile vehicle, on the base station, and optionally on edge and cloud computing infrastructure. It establishes a simple UDP-based communication protocol between air and ground, along with a set of class objects for standard exchanges relating to vehicle status, telemetry, network characterization, and mission requests. It automatically performs preflight and in-flight safety checks to prevent unsafe and unauthorized operations within the test domain, and allows users to focus on implementing their specific missions, state definitions, and integrating application wrappers.

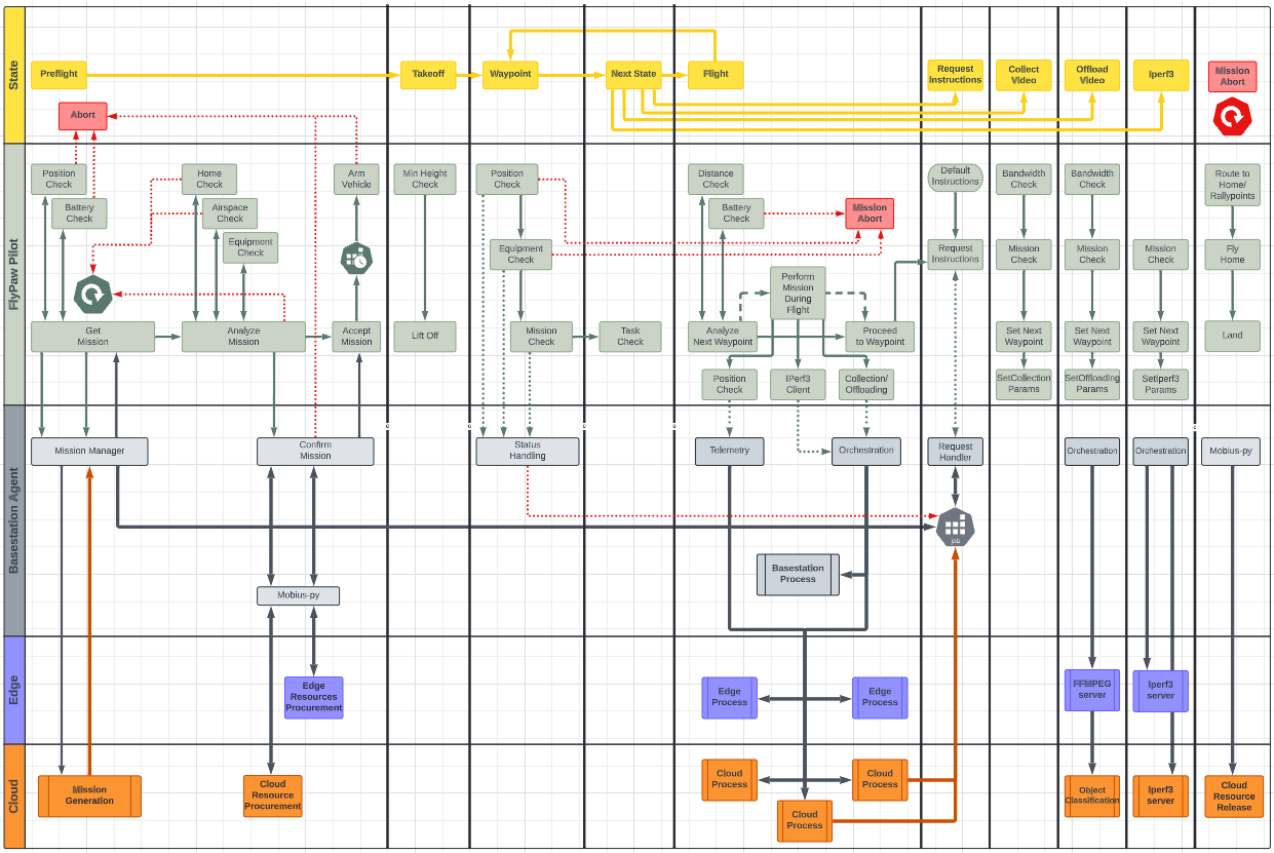

Fig. 1: Simplified representation of the FlyPaw state machine deployed from edge to cloud. Sample applications for a given mission relating to bandwidth measurement and video collection and streaming are depicted.

C. Architecture

Underlying the FlyPaw architecture is the notion of a mission manager agent running on a local ground-based computer called the base station, and one or more UAVs that represent free-agent workers, ready to perform a flight mission. Once activated, UAVs persistently initiate requests to the base station until they are assigned a mission. At present, researchers can create missions by defining a series of waypoints in the open-source software suite QGroundControl [15], which creates a *planfile*. With the aid of a Python script provided by FlyPaw, users can augment the planfile with associated tasks at specific waypoints. Tasks in the planfile must have associated user-defined logic blocks within the system so that the drone knows how to execute these. For example, a task requiring an iPerf client call must be able to reference the Python-wrapped function for making iPerf client calls, or know how to make an iPerf client system call, including any necessary parameters and requirements. While it is largely up to the researcher to integrate their application specific functions within the system, the linkage with the task manager to given tasks is simplified, and defining when they run – be it preflight, after takeoff, inflight upon arrival at a waypoint, inflight between waypoints, upon mission abort, or at landing – is laid out in various state modules within the system. Figure 1 depicts our distributed framework with pre-flight and in-flight states and select applications.

While FlyPaw is designed for users to create custom experiments, it is their responsibility to appropriately implement the functions. However, we believe that there is a shared aspect in how and where these tasks are communicated to the UAV. At present, FlyPaw includes function classes for commonly used applications such as iPerf, ping, and ffmpeg, as well as a customized image transfer module over UDP. Upon receipt of a mission, the UAV automatically populates a task queue, which is an ordered instruction sequence consisting of both vehicle and application level commands. The task queue is flexible and allows for modifications in flight in response to new information or if tasks cannot be executed as initially planned.

In order to facilitate complete workflows, users have the option to request edge and cloud resources specific to a particular mission type. These resources will be allocated and configured during the base station's mission initialization sequence using the Mobius resource provisioning system [9]. Modules are provided for in-line resource procurement on FABRIC [17] and Chameleon [18] testbeds. The base station FlyPaw module includes Prometheus [19] and automatically configures external compute resources so that compute and network status information can be queried for explicit load balancing. Additionally, FlyPaw automatically sets up the base station as a forwarding gateway and configures routes on the UAV such that it can communicate with edge and cloud servers over its wireless datalink. At this time, cloud accessibility is only available within the sophisticated AERPAW emulated system and not for live flights, as a security precaution relating to the automatic control of the drone.

Once resources have successfully been allocated and have come online, and the UAV has performed safety checks (GPS working, battery charged, etc.) and mission specific checks (camera present, waypoints within range, etc.), the vehicle is armed and takeoff is initiated. Once the UAV commences its mission, it regularly establishes

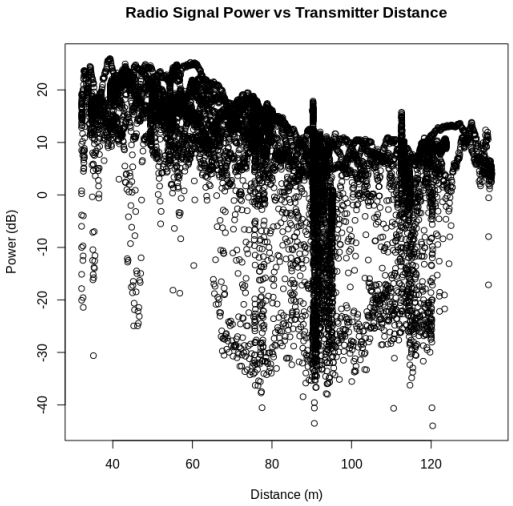

Fig. 2: A plot of 4G LTE received signal power as a function of distance from the transmitter. Over 10,500 samples depicted, showing wide variation of received power, even at same distances.

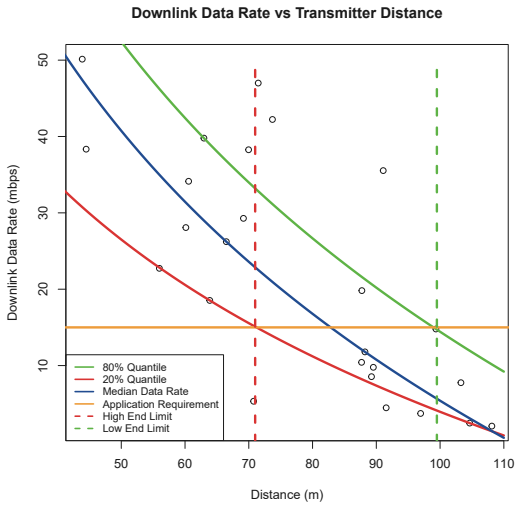

Fig. 3: Scatterplot of distance vs iPerf measured downlink data rates over srsRAN 4G LTE link during FlyPaw controlled live experiment. Quantile regression is performed showing the median estimated data rate, and 80% (green) and 20% (red) confidence indexes. A sample application requirement (orange) is depicted, here 15mbps for 1440p streaming video, with vertical dashed lines showing the distance interval over which uncertainty must be taken into account.

communication with the base station by sending UDP messages in an attempt to transmit its telemetry data. Upon receiving the message, the base station acknowledges it as the drone awaits the response. If a response is received, the UAV marks in its memory that radio connectivity was successful in that specific location. While this doesn't guarantee future connectivity at that location, it instills confidence and a connectivity map is generated, offering the UAV the option to return to those locations if needed to communicate with the base station.

Due to the unpredictable nature of communication between the UAV and base station, the FlyPaw architecture is specifically designed to enable missions to operate independently of the status of the datalink. Figure 2 depicts a channel sounding scatterplot that illustrates the significant deviations in received power even at equivalent distances from the transmitter. This underscores the importance of the drone to be able to autonomously make decisions in situations where communication with the base station is unattainable. FlyPaw's state machine is defined to handle these contingencies on a custom basis so the UAV has a default mechanism, either predefined or dynamic, to proceed or abort regardless of the connectivity with the ground. It is necessary to mention that connectivity with the UAV for our purposes refers to the datalink connection and not the control channel. For legal compliance, UAVs must always stay in range of command and control signals, such that a manual operator can, at any time, issue a "land" or "return home" command. However, that is only an oversight layer as the vehicle is otherwise entirely controlled via the datalink or with onboard processes. FlyPaw incorporates the AERPAW provided *aerpawlib* Python library [20] and uses MAVlink, the Micro Air Vehicle Communication Protocol [21] for programmatic control of the vehicle. FlyPaw provides logs from some independent modules of the overall flight system, including the flight telemetry, the measured data rates, the state transitions within the state machine architecture, as well as for a few select image transfer and image processing applications. Also included are Python scripts to merge independent logs, including native AERPAW logs relating to wireless radio connectivity, into common files for convenient timeseries and scatterplot analysis.

D. FlyPaw Live Flight Experiment

In March of 2022, the FlyNet team conducted the first user-created live flight test in the AERPAW testbed, with the vehicle semi-autonomously controlled through the FlyPaw platform. The test was simple, with the vehicle taking off and proceeding through a series of 27 waypoints at 30m above ground level, at each one making iPerf measurements to the base station over TCP and logging the achieved data rate and number of retransmissions. Data link connectivity was provided via 4G LTE and the srsRAN software defined radio suite [22]. Figure 3 depicts iPerf measured bandwidth results as a function of distance from the eNB anntenna. The live flight test demonstrated the capability to successfully hand off FlyPaw based experiments designed by the FlyNet team members on the emulated system, over to AERPAW staff to deploy on the UAV, the locally situated base station, and using the software defined radio. The UAV performed all preflight checks, took off, flew its intended path, achieved its tasks, landed safely, and the experiment logs were collected successfully.

III. Time Optimized Planning Routine

A. Overview

Here we introduce an algorithmic approach to in-flight route planning for a time-critical single UAV mission-oriented sensor system [23] relating to real-time image collection and processing. At its core, the *time-optimized planning* algorithm, henceforth "TOP", is dynamically controlling a single UAV to perform user-assigned tasks as fast as possible. It is designed to react to a state in which the drone cannot complete a task as planned, specifically a task held up by a lack of network connectivity, rendering it unable to offload data for further processing at the location at which it was collected. The technique presented in this paper is generally applicable to problems in which minimizing the "Age of Information" is an important component of application utility, and where wireless network coverage

(WiFi, LTE, point-to-point radios, or 5G) cannot be uniformly relied upon throughout the flight path. The term Age of Information (AoI) is used to describe the freshness or staleness of data. As stated by Liu et al. [24], when it comes to mission critical data, a lower AoI will result in higher confidence in decisions made based on the collected information. As example, we consider the well-researched, highly important use cases relating to wildland fire monitoring [3]–[5] and real-time object detection [25]. Drones are typically sensitive to weight and energy restrictions and resource-intensive data processing workflows are likely to occur on edge servers or cloud-deployed resources on the ground. The image collection is often only the first step of a requested task, as stakeholder benefit does not take effect until the data is transmitted and the results of the downstream analysis are generated and provided back to users. In the wildland fire case, these workflows include the detection of flames [3] and smoke [26], often using convolutional neural networks (CNN) and other computer vision techniques. Positive detections may also trigger advection models making use of meteorological information and ground fuels to model fire propagation [27]. These are potentially time-consuming, hardware-dependent operations.

TOP attempts to minimize the overall time required to complete workflows, which is a combination of gathering, offloading, and performing secondary processing on the given set of independent image collection tasks in the face of uncertain network connectivity. For this scenario, we make two notable assumptions: i) the processing of each individual frame or sequence holds incremental value and does not require the completion of all tasks before a certain level of utility is attained, ii) the vehicle is bound to flight tracks that it can traverse in both directions, but does not have full access to unrestricted airspace. These limitations constrain the set of possible solutions for specific mission objectives and, in many scenarios, represent a realistic model of flight operations. For instance, in cases of wildland fires, drones usually need to steer clear of areas with ongoing firefighting operations. Similarly, in object detection applications, vehicles may be required to avoid flying over densely populated or other sensitive areas.

TOP considers as input the estimated makespans of downstream post-processing workflows, and whether workflows can be executed in parallel or must be run serially. Processing criteria can significantly affect the best collection strategy. For example, if tasks must be individually processed in sequence, and each processing task is time-consuming, it becomes potentially advantageous to offload data throughout the flight and perform processing while data collection is still ongoing, rather than creating a processing backlog. Researchers can define static values representing their workflow makespans, and the number of available threads as input to TOP, which impacts the overall solution.

Furthermore, TOP evaluates the uncertainty of network connectivity along the path and incorporates this information into the set of potential solutions. In our experimental setup –using the AERPAW emulator and the FlyPaw architecture– the vehicle transmits telemetry data via UDP over an LTE software-defined radio link [22] and creates an internal map of where it received an acknowledgement from the base station. In addition, the vehicle intermittently halts and conducts TCP or UDP iPerf measurements to gather bandwidth estimates. This procedure enables TOP to assess the level of confidence that the vehicle will be capable of transmitting data at a projected rate from that specific position in the future. However, estimating the likelihood of connectivity in areas where previous measurements have not been made is a challenge. Although physical equations can be used to model connectivity and data rates as a function of distance, transmit power, antenna gain, frequency, and the number of users, as shown in Figure 2, even the received signal power is highly variable at fixed distances from the transmitter. Consequently, we do not attempt to estimate future network connectivity to a base station based on these physical parameters but rely on either the experimentally derived measurements within the AERPAW test domain [12], [28], or by running a series of iPerf measurements as shown in Figure 4 if available in suitable numbers for simple modeling. If network connectivity across the airspace were well understood in advance, the algorithm could be run in the preflight stage and achieve a near-optimal solution. We further discuss how connectivity confidence plays into the solution set subsequently.

B. Algorithm Execution

TOP is designed to be executed in-flight at 'haltpoints', which we define as places where a data collection task is requested (see Figure 4), but cannot be completed due to a lack of connectivity at that location. Upon reaching a task location, the image is collected and by default, an attempt is made to immediately offload the data to the ground-based processing resources. However, if that connection attempt does not succeed, this becomes a haltpoint, which triggers an iteration of TOP to determine the best course of action.

The available options for the drone at a haltpoint are few, given our adherence to a tightly controlled airspace where vehicles are bound to predefined tracks. The options are i) to "hold"– to queue the data and proceed along the given route until the next projected opportunity to transmit the data in the presence of connectivity presents itself, or ii) to "block"– to turn back and return to a previous location where connectivity is expected to allow execution of the present task, before resuming subsequent tasks. By definition, flight plan modifications that result in blocking (to achieve a given task) delay the vehicle's arrival time at future task locations. Although it may seem counterintuitive, this can actually lead to a shortened overall completion time, or makespan, for all tasks by more evenly distributing downstream processing tasks.

Another possibility for applications that prioritize rapid processing upon collection is to minimize AoI. In this case, blocking can improve the collection strategy, so long as the utility gain for a given task is not outweighed by the implied delay for future tasks.

When replanning at a haltpoint, TOP must take into account the possibility that future task points may also be haltpoints. The binary nature of this lends itself to the creation of decision trees for evaluation purposes. Using the given uncertainty model, TOP probabilistically estimates the likelihood that each subsequent task will be a haltpoint, and creates the solutions where connectivity is present and where it is not. Algorithm 1 describes the recursive algorithm sequence. The solution in which we assume that connectivity is not present results in a recursive iteration of TOP and a fork in the decision tree. Figure 6 depicts such a tree, with the net task set execution time depicted as a time series. Each decision tree member is associated with an overall confidence metric and measures of time, both described in more detail below. The preferred weighting of these parameters is application specific and exposed to the researcher within FlyPaw. Algorithm solutions are

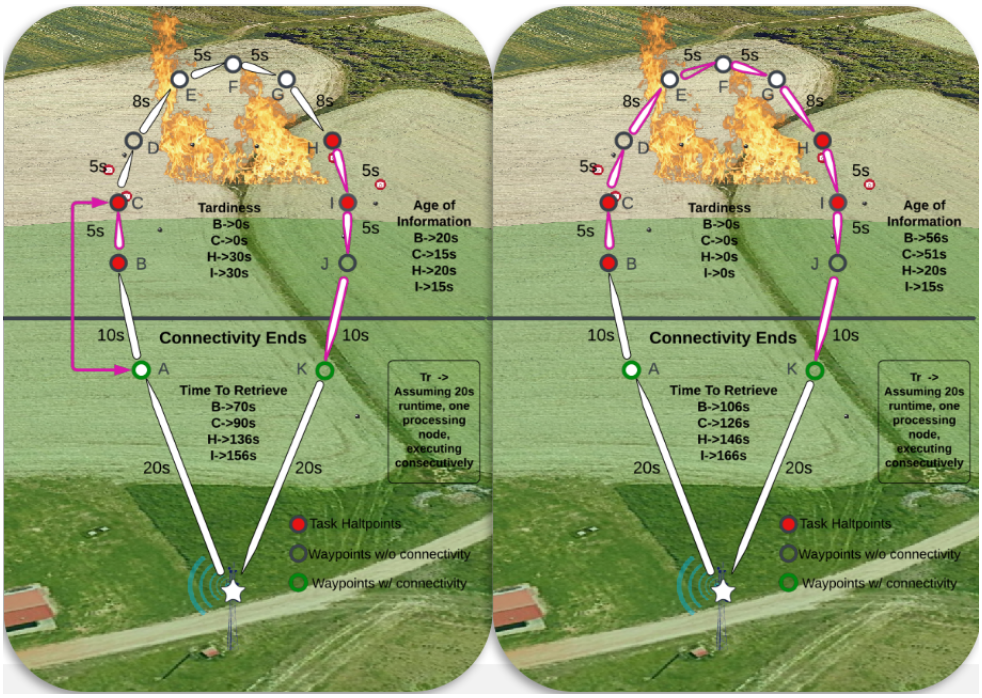

Fig. 4: Exemplary mockup of image collection mission around a simulated fire in the AERPAW testbed.

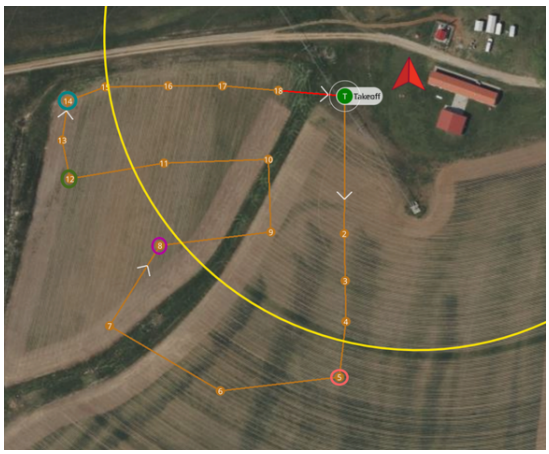

Fig. 5: A simple test mission displayed on QGroundControl. Yellow line represents the connectivity boundary limit. Image collection points here shown with colored outlines are all haltpoints.

determined according to weighted preference. TOP runs at present as an exhaustive, brute force algorithm, considering all possible solutions in the decision tree, with O(n^2) complexity, scaling with the number of tasks. We recognize that a very large task set would potentially require TOP to be revised to reduce the search space.

It should be mentioned that any in-flight path changes must be allowed within the physical limitations of the vehicle. This is supported by the FlyPaw architecture, which provides a level of oversight with respect to battery consumption and the rejection of flight plans that would risk exceeding capacity. At present, we do not impose penalties on flights for the increased resource utilization resulting from blocking solutions, as long as the flight and flight tasks can still be successfully completed. However, the total distance flown could be taken into account as an indicator of resource consumption.

Algorithm 1 Pseudo code to generate predictive analysis tree.

if current state's TaskQ is empty **then**
  return ▷ a solution has been reached
else
  Block on Halting node with halted state N_H,returning N_B. ▷ N_B contains a $TaskQ$, with this task located at the head of the queue for priority
  while $TaskQ$ is not empty **do** ▷ using node N_{C_b} initialized with state from N_B
    if action can be completed given the state of N_{C_b} **then**
      Simulate Action on Virtual Drone popping this task from the $TaskQ$
      Adopt a new node for this executed action, set N_{C_b} to this new node
    else
      Recursively Call Haltpoint on N_{C_b}
    end if
  end while
  Hold on Halting node with halted state N_H and halting task, returning unhalted node N_W.
  while TaskQ is not empty **do** ▷ using node N_{C_h} initialized with state from N_W
    if action can be completed given the state of N_{C_h} **then**
      Simulate Action on Virtual Drone popping this task from the TaskQ
      Adopt a new node for this executed action, set N_{C_h} to this new node
    else
      Recursively Call Haltpoint on N_{C_h}
    end if
  end while
end if

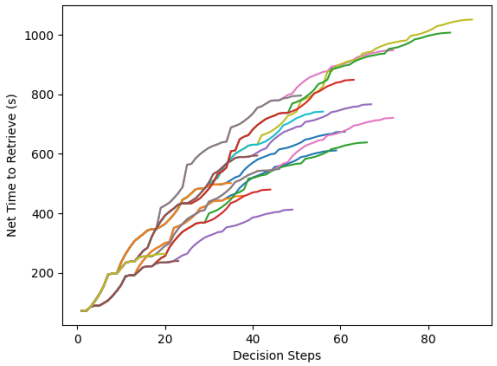

Fig. 6: A sample TOP decision tree depicting the time for net task completion of test mission shown in Figure 5. Each line shown represents a sequence of block and hold and transmit solutions for each mission task. The best solution is to Block, i.e., to turn around, at the first haltpoint task location (point 5), and thereafter to Hold, i.e., to proceed, at the next 3 haltpoint locations (points 8, 12, 14), as indicated by the bottom line in the graph.

C. Data Timeliness

The product of TOP is a decision tree of possible waypoint sequences and outcomes. From this set of solutions, we can estimate the distance, total trip time, uncertainty, and timeliness of each solution. Data timeliness can be expressed as a funcition of the Age of Information, (AoI), and Time to Retrieve (Tr). AoI represents the time elapsed from the moment data is gathered ($t_{collect}$) to when its associated processing is completed ($t_{processed}$).

$$AoI = t_{processed} - t_{collect}$$

From the definition of AoI, once collected, data starts to become stale. Minimizing AoI may be particularly important when a tight coupling of observation and reaction is necessary for satisfactory task completion, as it prioritizes rapid completion of tasks after data collection. To minimize AoI for a specific data item, it is crucial to prioritize the timely delivery of that information to the relevant stakeholder. Simultaneously, there is a notion of tardiness to collect data for an event. Time spent before an image is captured accrues on a per task basis. Therefore, efforts to minimize AoI of one task have the potential to generate tardiness for subsequent tasks, thereby degrading the relevancy of the image to the mission. Per task tardiness is referred to as the Time to Capture ($t_{capture}$), which can be expressed as the time elapsed from the mission start time, or, preferably, as the additional delay from the minimum time that the task could be captured from the mission start. By using delay, we effectively normalize for distance from the start point such that tardiness can be evaluated on a relative basis. Using delay also represents the direct consequence of decisions to block, as a strategy of always holding at haltpoints represents by definition the minimum $t_{capture}$ for all tasks given our current assumptions. Therefore:

$$t_{capture} = t_{collect} - t_{collect_{min}}$$

Per task Time to Retrieve is then

$$Tr = t_{capture} + AoI$$

The evaluation of timeliness on a full mission basis can be measured as the net task sum of the AoI, of the Tr, or a weighted

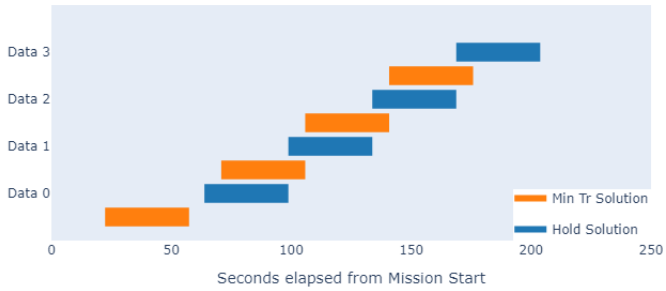

Fig. 7: Gantt Chart depicting the start and end times processing times for two solutions from the mission shown in Figure 5. The best solution of blocking at the first haltpoint results in processing occurring while the mission is ongoing and reduces backlog. For this evaluation, we assume a single machine, sequential processing pipeline, and 35-second runtime. This is consistent with some Darknet YOLO processing times on CPU devices.

combination of the two. Minimizing net Tr is the fastest time that all tasks could be completed. Figure 4 is a simple notional example of how blocking can improve net Tr and. It shows an exemplary mockup of an image collection mission around a simulated fire in the AERPAW testbed, considering a vehicle path strategy and video analytics post processing. The solution minimizing Time to Retrieve is shown on left, depicting a hold/block/hold/hold approach. On the right, is the naive solution, proceeding along the loop continuously. Given a single server for image processing with 20s runtime, this notional demonstration of TOP results in 6% faster completion of all tasks, ~17% net Time to Retrieve reduction, and ~51% net Age of Information reduction. Figures 5, 6, and 7 depict a real experiment performed on the AERPAW testbed, similarly depicting net Tr improvement by blocking. However as mentioned previously, each solution is associated with a projected probability of success and the fastest solution may be improbable for lack of connectivity.

D. Estimation of Uncertainty

In evaluating haltpoint solutions with an attempt to make an optimal decision, risk must be considered. At present, risk primarily represents the uncertainty associated with future data link connectivity along the flight path and at later task points. Risk models might also eventually consider physical risk to the drone itself, or risk of mission failure should researchers be able to define and quantify the circumstances that lead to such. We envision that these could take the form of maximum allowed time for mission completion, or perhaps the integrated flight time over a portion of route deemed more dangerous for some reason.

Within the AERPAW testbed, our uncertainty estimates are generated in a two-fold manner. For points we have not yet been to, we estimate the likelihood of a usable connection as a function of distance, using either previously collected datasets characterizing connectivity and data rate, and/or by using measured connectivity that is generated in flight as the drone transmits telemetry data and makes iPerf measurements. Quantile regression analysis is performed to determine the conditional probability distributions as shown in Figure 3. Accuracy is sensitive to the number of samples, particularly if the researcher chooses to use in flight measurements exclusively. We prefer to use the a priori measurements within the AERPAW testbed for this reason. However, as shown in Figure 2,

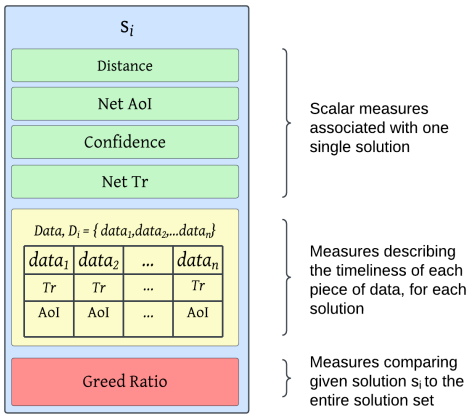

Fig. 8: A visual representation of parameters associated with the algorithm solution.

it is evident that components of connectivity can exhibit significant variations even at similar distances, indicating that this estimate is inherently imperfect. The FlyPaw repository contains code to perform quantile regression in R and Python and to merge iPerf and telemetry logs to estimate connectivity changes as a function of distance.

Attempting to create a generalized approach to connection certainty for arbitrary airspace is a difficult problem and beyond the scope of this work.

This empirical description of network uncertainty for points the vehicle has not yet been is complemented by iPerf measurements taken during the mission. As the vehicle progresses through the mission, FlyPaw periodically makes note of the bandwidth at given locations using calls to iPerf. If connectivity is present and an iPerf measurement indicates suitable bandwidth we automatically assign a high probability of connectivity should we return to that point (0.95), deferring to the measurements over the function of distance. Similarly, if no connectivity was present at that location, we assume a similarly low probability of connectivity at that point (0.05) in the future.

The uncertainty of a given solution is determined by multiplying the connection uncertainties of each transmit task included within the solution. We observe that the fastest possible solution may well be a low-confidence solution depending on the location of the assumed transmissions, making it quite possibly a less-than-optimal choice.

E. Solution Evaluation

The optimal solution can vary greatly depending on priorities and weights. For each solution there is a set of specifications, including a set of time benchmarks for each data set collected –images for our initial applications. The solution specifications are currently defined as follows: **Distance flown, Net Age of Information, Net Time to Retrieve, and Solution Uncertainty**. Within each solution, each task has individual metrics scored, those being Age of Information, Time to Retrieve, Tardiness, and Uncertainty. Figure 8 depicts solution and data evaluation metrics.

With these fully evaluated, researchers can easily assign customized weighted parameters on solution set metrics, or prune out solutions through thresholds. In our evaluation, we decided to give full weight to Net Time To Retrieve, as this represents the fastest net time for the data to be processed and returned to users, but certain applications may have other requirements.

Weighing a solution's timeliness against the associated uncertainty requires consideration. To evaluate each solution, we compare them to the Naive solution. The naive solution corresponds to holding at every haltpoint, i.e., effectively continuing through the mission from start to end and sending data upon landing where network connectivity is guaranteed. The naive solution, given our assumptions, is the solution with zero tardiness. While this may be sub-optimal, it represents a simple solution that may occur for lack of a dynamic approach to data collection. In order to grade each solution's timeliness and adjust for uncertainty, we coin the parameter Greed Ratio (G_r), informed by Sharpe's Ratio [14]. Sharpe's Ratio is a measure of performance adjusted for risk. Typically it is used to examine a financial investment compared to the risk-free asset, defining the performance, adjusted for the risk.

$$S_a = \frac{E[Return_{RiskyAsset} - Return_{RiskFree}]}{\sigma_{RiskyAsset}}$$

Sharpe's ratio attempts to characterize the reward for making a risky decision and adjusting for said risk by taking the difference in returns and adjusting for risk with the variability of the Risky Asset. For a given solution i, here we propose measuring Risk Adjusted Timeliness using the Greed Ratio defined as follows:

$$G_r = \frac{Tr_{Naive} - Tr_i}{1 - ConnectionUncertainty_i}$$

A solution with a larger G_r compared to another solution is typically preferred, though for a given application one may prefer a higher certainty of connection or faster execution regardless of certainty. A negative G_r indicates that the naive solution has a lower net Tr than the given solution. Maximizing G_r helps prune out fast but highly uncertain Tr solutions, and slow but highly certain solutions. The user may also choose to introduce a threshold such that low confidence solutions are pruned out even if $Tr_i << Tr_{Naive}$.

IV. Related Work

Controlling and off-loading data from UAVs has been a very active research area over the past few years. Among the first, Pitre et al. [29] looked at route planning for joint search and track missions through path optimization based on an objective function. An early review by Triharminto et al. [30] presents several categories of route planning algorithms to intercept moving targets. In contrast, we currently assume that the targets for our UAV applications are not or only very slowly moving. More recently, Zhou et al. [31] present an approach based on reinforcement learning with the goal to improve the convergence speed of UAV route planning. The benefits of this approach are demonstrated through a simulation-based evaluation. In contrast to our approach, none of these works consider the (un)reliability of data communication networks as part of the route planning approach with respect to scientific workflows.

A comprehensive overview of the state of the art of UAV route planning is provided in Aggrawal and Kumar's survey article [32]. While the article considers network communication for drones, it does not incorporate the uncertainty of network communication as we do in the approach presented in this paper.

Liu et al. [24] and Hu et al. [33] both provide very relevant work regarding minimizing Age of Information (AoI). However, they do not include downstream processing considerations and the Time to Retrieve metrics. They make an attempt at generic network

characterization through physical radio propagation parameters. However, they assign constant values to bandwidth and noise power that we do not have available within the real AERPAW testbed. The work by Hu et al. [33] does not assume fixed trajectories making this a more complex path planning and energy optimization problem, but neither work takes into account connectivity uncertainty.

Ivancic et al. [34] evaluated the potential use of 4G LTE in commanding UAV traffic and how the performance of the LTE network affects the communication reliability and the application data capabilities. Ateya et al. [35] explored novel algorithms to offload data from drones keeping energy and latency in mind, while Kim et al. [36] explored offloading of computations in edge to cloud scenarios in an effort to optimize energy consumption of the UAVs. However, in their work, they do not consider dynamic routing decisions.

In our previous work [9], we demonstrated how a workflow management system and a dynamic resource provisioning component can work together to support data-driven science applications. Specifically, we have shown the capability to dynamically provision network links to transmit weather radar data to computational resources that are also dynamically provisioned. This system [9] was put in place to support the CASA [37] severe weather forecast and warning system in the Dallas Fort Worth area. We recently extended this data-driven science application to support workflows that include UAVs [6], including for dynamic path planning in free airspace around weather constraints [38]. We have further refined this architecture by proposing a new network service for data-driven workflows [11], which we have deployed in an actual programmable network testbed [17].

V. Conclusion

In this work, we have developed and evaluated an architecture and software suite, FlyPaw, which we believe can facilitate researcher experiments using AERPAW. FlyPaw promotes programmatic interfaces to the air and ground-based component systems of an AERPAW experiment with its Python-based state machine framework, allowing for dynamic decision-making and semi-autonomy, while taking into consideration uncertain data links, and flight safety parameters. It includes modules to allocate resources on NSF sponsored testbeds FABRIC and Chameleon Cloud, including the deployment of image processing workflows, automatic configuration and load monitoring with Prometheus, and network routing to seamlessly integrate the air-to-ground-to-cloud communications. FlyPaw includes an innovative and tuneable algorithm to optimize UAV route planning and task execution with considerations of the Age of Information, uncertain data link connectivity along the vehicle path, and downstream processing workflow makespans. It is important to note that our evaluation of FlyPaw has been limited to the AERPAW testbed, and our network uncertainty estimation relies on empirical data collected by ourselves and fellow researchers. Despite these limitations, we believe that the FlyPaw suite is extensible to generic UAV, base station, and edge server deployments, with optional connectivity to compute clouds. The FlyPaw architecture adheres to the common airframe control standard Mavlink and is interoperable with QGroundControl, both popular assets in the UAV community.

In future work, we plan to extend our network uncertainty estimation to better model previously unknown airspaces. This is a challenging problem as demonstrated by the highly variable power measurements that result from real-world interference, blockage, and propagation effects. AERPAW has recently introduced 5G data link options, which we plan to characterize across the testbed.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Award Number OAC-2018074, and utilizes the NSF AERPAW Platform (CNS-1939334). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] K. Akiyama *et al.*, "First m87 event horizon telescope results. iii. data processing and calibration," *The Astrophysical Journal. Letters*, vol. 875, no. 1, 4 2019.

[2] R. Dewan and K. F. Rahman, "A survey on applications of unmanned aerial vehicles (uavs)," in *Recent Innovations in Computing*, P. K. Singh, Y. Singh, J. K. Chhabra, Z. Illés, and C. Verma, Eds. Singapore: Springer Singapore, 2022, pp. 95–110.

[3] A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P. J. Fulé, and E. Blasch, "Aerial imagery pile burn detection using deep learning: The flame dataset," *Computer Networks*, vol. 193, p. 108001, 2021.

[4] X. Chen, B. Hopkins, H. Wang, L. O'Neill, F. Afghah, A. Razi, P. Fulé, J. Coen, E. Rowell, and A. Watts, "Wildland fire detection and monitoring using a drone-collected rgb/ir image dataset," *IEEE Access*, pp. 1–1, 2022.

[5] R. S. Allison, J. M. Johnston, G. Craig, and S. Jennings, "Airborne optical and thermal remote sensing for wildfire detection and monitoring," *Sensors*, vol. 16, no. 8, 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/8/1310

[6] E. Lyons, H. Saplakoglu, M. Zink, K. Thareja, A. Mandal, C. Qu, S. Wang, P. Calyam, G. Papadimitriou, R. Tanaka, and E. Deelman, "Flynet: a platform to support scientific workflows from the edge to the core for uav applications," in *2021 IEEE/ACM 14th International Conference on Utility and Cloud Computing*, I. Brandic, R. Sakellariou, and J. Spillner, Eds. ACM, 2021. [Online]. Available: https://doi.org/10.1145/3468737.3494098

[7] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Gener. Comput. Syst.*, vol. 46, pp. 17–35, May 2015.

[8] R. Tanaka, G. Papadimitriou, S. C. Viswanath, C. Wang, E. Lyons, K. Thareja, C. Qu, A. Esquivel, 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 *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2022, pp. 826–833.

[9] 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.

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

[11] A. E. 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)*, 2023, pp. 130–136.

[12] A. Panicker, O. Ozdemir, M. L. Sichitiu, I. Guvenc, R. Dutta, V. Marojevic, and B. Floyd", "Aerpaw emulation overview and preliminary performance evaluation," *Computer Networks*, vol. 124, 2021.

[13] A. Gosain, "Platforms for advanced wireless research: Helping define a new edge computing paradigm," in *Proceedings of the 2018 on Technologies for the Wireless Edge Workshop*, ser. WirelessEdge '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 33. [Online]. Available: https://doi.org/10.1145/3266276.3266283

[14] W. F. Sharpe, "Mutual fund performance," *Journal of Business*, vol. 55, pp. 119–138, 01 1966.

[15] D. Gagne *et al.*, "mavlink/qgroundcontrol: v4.0.8," May 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3870845

[16] E. Adams, A. Grote, and K. Thareja, "Flypaw," 2022. [Online]. Available: https://github.com/FlyNet-NSF/flypaw

[17] 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.

[18] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, "Chameleon: a scalable production testbed for computer science research," in *Contemporary High Performance Computing: From Petascale toward Exascale*, 1st ed., ser. Chapman & Hall/CRC Computational Science, J. Vetter, Ed. Boca Raton, FL: CRC Press, May 2019, vol. 3, ch. 5, pp. 123–148.

[19] B. Rabenstein and J. Volz, "Prometheus: A next-generation monitoring system (talk)." Dublin: USENIX Association, May 2015.

[20] J. Kesler, "aerpawlib-vehicle-control," 2021. [Online]. Available: https://github.com/morzack/aerpawlib-vehicle-control

[21] L. Meier, "Mavlink," 2009. [Online]. Available: https://github.com/mavlink/mavlink

[22] S. R. Systems, "srsran," 2021. [Online]. Available: https://github.com/srsran/srsRAN_4G

[23] M. Ammari, *Mission-Oriented Sensor Networks and Systems: Art and Science Volume 1: Foundations: Volume 1: Foundations*, 01 2019.

[24] J. Liu, X. Wang, B. Bai, and H. Dai, "Age-optimal trajectory planning for uav-assisted data collection," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 553–558.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[26] X. Zheng, F. Chen, L. Lou, P. Cheng, and Y. Huang, "Real-time detection of full-scale forest fire smoke based on deep convolution neural network," *Remote Sensing*, vol. 14, no. 3, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/3/536

[27] J. Hodges and B. Lattimer, "Wildland fire spread modeling using convolutional neural networks," *Fire Technology*, vol. 55, 03 2019.

[28] S. J. Maeng, I. Guvenc, M. Sichitiu, R. Dutta, O. Ozdemir, and M. Mushi, "Aeriq: Sdr-based lte i/q measurement and analysis framework for air-to-ground propagation modeling," in *Proc. IEEE Aerospace Conference*, 2023, pp. 1–20.

[29] R. R. Pitre, X. R. Li, and R. Delbalzo, "Uav route planning for joint search and track missions—an information-value approach," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2551–2565, 2012.

[30] H. Himawan Triharminto, A. S. Prabuwono, T. B. Adji, N. A. Setiawan, and N. Y. Chong, "Uav dynamic path planning for intercepting of a moving target: A review," in *Intelligent Robotics Systems: Inspiring the NEXT*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 206–219.

[31] Y. Zhou, J. Shu, X. Zheng, H. Hao, and H. Song, "Real-time route planning of unmanned aerial vehicles based on improved soft actor-critic algorithm," *Frontiers in Neurorobotics*, vol. 16, 2022.

[32] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, vol. 149, pp. 270–299, 2020.

[33] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "Aoi-minimal trajectory planning and data collection in uav-assisted wireless powered iot networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1211–1223, 2021.

[34] W. D. Ivancic, R. J. Kerczewski, R. W. Murawski, K. Matheou, and A. N. Downey, "Flying drones beyond visual line of sight using 4g lte: Issues and concerns," in *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, 2019, pp. 1–13.

[35] A. A. Ashraf Ateya, A. Muthanna, R. Kirichek, M. Hammoudeh, and A. Koucheryavy, "Energy- and latency-aware hybrid offloading algorithm for uavs," *IEEE Access*, vol. 7, pp. 37 587–37 600, 2019.

[36] B. Kim, J. Jang, J. Jung, J. Han, J. Heo, and H. Min, "A computation offloading scheme for uav-edge cloud computing environments considering energy consumption fairness," *Drones*, vol. 7, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2504-446X/7/2/139

[37] D. McLaughlin, D. Pepyne, B. Philips, J. Kurose, M. Zink *et al.*, "Short-Wavelength technology and the potential for distributed networks of small radar systems," *Bull. Am. Meteorol. Soc.*, vol. 90, no. 12, pp. 1797–1817, Dec. 2009.

[38] E. Lyons, D. Westbrook, A. Grote, G. Papadimitriou, K. Thareja, C. Wang, M. Zink, E. Deelman, A. Mandal, and P. Ruth, "An on-demand weather avoidance system for small aircraft flight path routing," in *Dynamic Data Driven Applications Systems*, F. Darema, E. Blasch, S. Ravela, and A. Aved, Eds. Cham: Springer International Publishing, 2020, pp. 311–319.