Information Autonomy: Self-Adaptive Information Management for Edge-Assisted Autonomous UAV Systems

Davide Callegaro*, Sabur Baidya*, Gowri S. Ramachandran†, Bhaskar Krishnamachari† and Marco Levorato*
*Computer Science Department, University of California, Irvine
{dcallega, sbaidya, levorato}@uci.edu
†USC Viterbi School of Engineering, University of Southern California, Los Angeles, USA
{gsramach,bkrishna}@usc.edu

***Abstract*—Making Unmanned Aerial Vehicles (UAV) fully autonomous faces many challenges, some of which are connected to the inherent limitations of their on-board resources, such as energy supply, sensing capabilities, wireless characteristics, and computational power. The sensing, communication, and computation Internet of Things (IoT) infrastructure surrounding the UAVs can mitigate such limitations. However, external traffic dynamics, signal propagation, and other poignant characteristics of the IoT infrastructure make it an extremely dynamic and incoherent environment, especially in urban scenarios, thus challenging the use of IoT resources for mission-critical UAV applications. Herein, the concept of information autonomy is introduced to extend autonomy to encompass how information-related tasks are handled in this challenging scenario. In this paper, we motivate the need for "Information Autonomy" based on our observations from real-world experiments and present a self-adaptive framework for edge-assisted UAV applications. Through our preliminary evaluation, we show that our "Information Autonomy" framework is capable of handling uncertainties autonomously during run-time.**

I. Introduction

On-board resources available to commercial Unmanned Aerial Vehicles (UAV) are inherently limited by the airborne nature of these devices. Such constraints affect key subsystems, such as sensors and computing platforms, which are instrumental for prolonged and autonomous operations. A relevant trend in mobile devices, whose application has been recently extended to include UAVs, is to use resources from the Internet of Things (IoT) and edge infrastructure surrounding the UAVs to enhance their performance [1], [2]. In particular: *a)* wireless cellular networks can be used to extend the communication range of the UAVs and connect them to other edge devices or the internet core [3]; *b)* data streams from ground sensors or devices can supplement those from on-board sensors to extend the information available to the UAVs [4]; and *c)* signal processing tasks can be offloaded to compute-capable devices residing at the network edge, that is, edge servers [5].

Although most of the discussion provided in this paper applies to all the three cases listed above, herein we focus on edge offloading – case *c)*, often referred to edge computing [4] – due to its potential to enable a high degree of operational autonomy to those highly constrained devices. The advantages of edge computing applied to UAV systems are rather intuitive:

This work was partially supported by the NSF under grant IIS-1724331 and DARPA under grant HR00111910001.

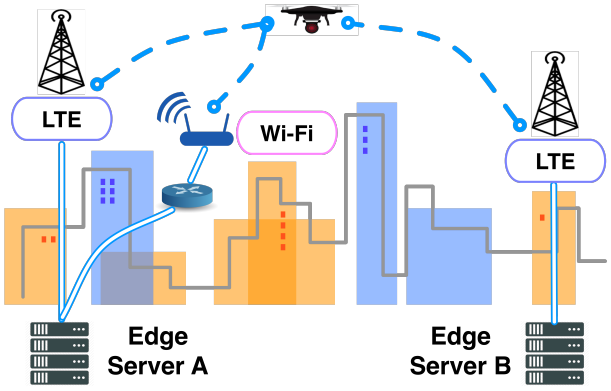

Fig. 1: The scenario considered in this paper: an autonomous UAV leverages the surrounding IoT infrastructure to improve mission performance. Specifically, we focus on offloading computing tasks to edge servers through wireless links.

a powerful machine with an unlimited energy supply connected to a UAV through a one hop wireless link can effectively run complex algorithms beyond the reach of on-board UAV computing platforms and deliver the outcome to the UAV, thus enabling advanced – autonomous – learning, control and planning. The edge server can run simpler algorithms in a shorter time, thus improving the reaction time of the UAV to external stimuli. Finally, we remark that continuous computation requires a substantial amount of energy, a precious resource that should be parsimoniously used to avoid degradation of mission lifetime. By taking over compute-intense processes, the edge server can significantly reduce energy consumption of the UAV.

Based on the above discussion, edge computing, and in general infrastructure assistance to UAV systems, appears to be an extremely promising component of future UAV systems (see Fig. 1). However, there are several technical challenges to overcome to fruitfully and reliably apply this paradigm to a mission-critical system such as autonomous UAVs. A crucial aspect of the infrastructure assisting the UAVs is that it is shared by a multitude of devices and services. This, together with the characteristics of signal propagation, creates an extremely dynamic environment, where the time needed to transfer data to the edge server, or the time to complete the processing task are highly variable, and governed by extremely complex temporal and spatial random processes. As a result, blindly trusting infrastructure assistance may lead to severe performance degradation.

We contend that to fully harness the benefits of infrastructure assistance while preserving reliable flight and mission control the notion of "autonomy" needs to be extended from pure mission control to include an advanced layer of intelligence making decisions on how information is handled within the complex UAV-infrastructure system. We refer to this layer as "information autonomy". We remark that information autonomy could boost the performance of many distributed, and constrained, mission-critical systems. We make our discussion specific to UAVs due to their extreme characteristics in terms of limitations, mobility and complexity of operations.

Herein, we provide a first description of an architecture realizing information autonomy, and a detailed discussion on the many challenges present in the definition of key modules such as state acquisition/tracking, prediction and decision making. The discussion is supported by illustrative results from detailed simulation and real-world experiences and implementations.

The rest of the paper is organized as follows. Section II describes the operational scenario and discusses the lessons learned from real-world experiments. In Section III, we present the motivation and need for information autonomy in the context of UAV systems. Section IV presents the fundamental structure of information autonomy and emphasizes the key challenges in its realization. Section V presents the proposed "Information Autonomy" architecture. We present the preliminary results in Section VI. Finally, Section VII concludes the paper.

II. Lessons Learned from Real-world Experiments on UAV Systems

Figure 1 illustrates the autonomous infrastructure-assisted UAV system at the center of this paper. In the edge computing scenario, the UAV is connected to one or more edge servers, positioned at the network edge for handling resource-intensive computations, through available wireless access networks, such as Long-Term Evolution (LTE) or Wi-Fi. Edge servers are preferred over cloud infrastructure due to their low latency.

The edge server would take over the analysis, and possibly the control, operations of the profiling-analysis-control pipeline typically seen in UAV applications. In such Edge-based computation pipeline, the input generated by the sensor and the output from the modules at the edge server, needs to be wirelessly transferred to and from the edge server. Importantly, as compute-intense tasks often require a large amount of energy, offloading typically reduces energy consumption at the UAV, even when considering the additional energy needed to transmit the data to the edge server.

For a successful completion of a mission, all the hardware components of UAV have to continuously generate new knowledge for the control algorithm in a dependable and energy-efficient manner. Unfortunately, it is difficult to estimate the resources needed for sensing, actuation, communication, and control before the mission (run-time) since the stability of the drone is impacted by various environmental, system, and network components. In the rest of this section, we present the lessons learned from our real-world experiments.

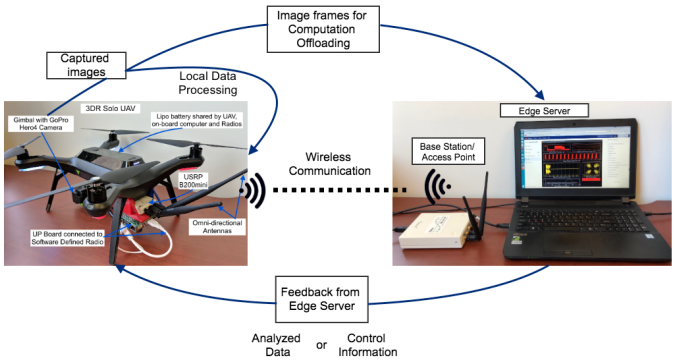

Fig. 2: Experimental setup for UAVs with on-board computing and communication equipements.

A. Run-time Uncertainties of Edge-assisted UAV Applications

We have carried out a set of experiments involving drones and edge devices following the setup as shown in Figure 2 to understand the performance and the resource constraints of edge-assisted UAV applications. The UAV is equipped with small on-board computer connected with WiFi dongle and USRP B200mini for LTE interface. The edge server is realized on a Laptop with high computational capability, that is also connected with WiFi and USRP B210 for LTE interfaces. We tested a UAV mission of target tracking based on image classification.

From our real-world experiments conducted in the outdoor, we have identified the following uncertainties in managing the edge-assisted UAV applications:

- $\mathcal{U}$**1. Environmental Uncertainty** Weather pattern such as wind speed, temperature, and humidity influences the control algorithm since the drone needs to stabilize itself under harsh conditions by controlling its engines. Due to the unpredictable nature of the weather, it is hard to allocate resource budget for control operations prior to the mission.
- $\mathcal{U}$**2. System Uncertainty** For surveillance or other edge-assisted UAV applications, performance depends on the computation capacity of the drone, and the energy budget available for various operations including computation, communication, sensing, and storage (e.g., buffer).
- $\mathcal{U}$**3. Network Uncertainty** Wireless communication is susceptible to interference, hence the edge-assisted UAV applications relying on remote edge servers have to carefully manage the network resources including the bandwidth capacity of the wireless link and the energy budget needed to operate the radio hardware.

The above challenges highlight the need to introduce "Information Autonomy" to react and adapt to the system, network, and environmental uncertainties. Application goals have to be updated dynamically during the mission time to maximize the mission duration. In the next section, we formulate problem goals with respect to the application scenario.

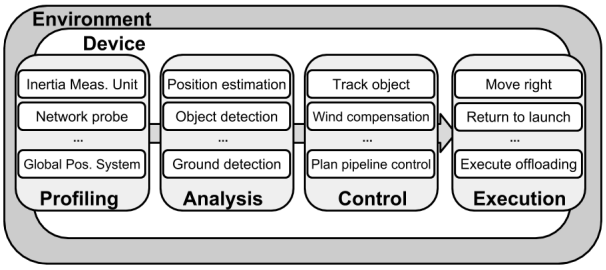

Fig. 3: Profiling-Analysis-Control Pipeline for Mission Autonomy.

III. Problem Formulation and Motivation

We consider an UAV autonomously operating to fulfill a mission. Here, we do not specifically define the mission, but, instead, assume that autonomous operations necessitates sensor input to be transformed into short- and/or long-term decisions. For instance, video input from on-board cameras can be processed to determine navigation, or to plan the mission. We further assume that our objective is to assure the mission completion, therefore maximizing mission time, while maintaining quality of operation, as discussed below.

Figure 3 depicts a basic diagram of the modules involved in the transformation from input vector $\mathbf{u}_t$ to the output vector $\mathbf{y}_{t+\Delta}$ given a context or state $\boldsymbol{\phi}_t$, where t is the time at which the sensor input is acquired. The input is first processed in the analysis module to extract relevant features. For instance, in a video-based navigation application, the input is a video frame, and the output of the analysis module, generated by an object detection algorithm, is a set of labeled bounding boxes. The features produced by the analysis module are sent to the control module, which generates the final output $\mathbf{y}_{t+\Delta}$. In this particular example, the output are motion and navigational commands.

Formally, we denote the transformation using the following:

$$\mathbf{y}_{t+\Delta} = f(\mathbf{u}_t, \boldsymbol{\phi}_t). \tag{1}$$

Note that the output is generated at time $t+\Delta$, where Δ is a random variable corresponding to the time needed for the transformation. We refer to Δ as the *capture-to-output* delay.

If the function f is complex, the delay Δ might be large, thus increasing the reaction time of the UAV to input stimuli and possibly degrading mission performance. Moreover, compute intense transformations may require a large amount of energy E to be completed. *Our objective, thus, is to to use infrastructure-assistance to reduce as much as possible Δ, while also minimize energy consumption.*

In the edge-based pipeline, the capture-to-output time, Δ, is a function of many variables describing the environment. Let's decompose the delay of the edge-based pipeline Δ^e as follows:

$$\Delta^e = \Delta^e_{\text{data}} + \Delta^e_{\text{computing}} + \Delta^e_{\text{output}}, \tag{2}$$

where Δ^e_{data}, $\Delta^e_{\text{computing}}$, and Δ^e_{output} are the delay to transport the data to the edge server, the processing time at the edge server and the time to transport the output back to the UAV, respectively.

Intuitively, the components associated with the wireless transfer of data, *i.e.*, Δ^e_{data} and Δ^e_{output}, are a function of the used technology (*e.g.*, LTE and Wi-Fi) channel characteristics (UAV-base station distance, fading and shadowing), and network load. The processing component of the delay, that is, $\Delta^e_{\text{computing}}$, is highly dependent on the server load. Importantly, channel gain, and network and server load are highly variable, and have complex spatio-temporal distributions that play an important role in building information autonomy.

A. Need for Information Autonomy

To study the dynamics of Edge-assisted UAV applications, we used an integrated UAV-network simulator – FlyNetsim [6] which allows the application developers to experiment with simulated UAVs, wireless network(s), and edge server. The UAVs receive control messages over widely used MAVLink [7] protocol and exchange telemetry information for specific computation tasks and closed-loop controls. Different from other integrated UAV-network simulators, FlyNetSim is capable of supporting a wide range of application scenarios in terms of UAV controls, e.g., multi-UAV swarms, transporting data streams and telemetry-based controls; also it can emulate a range of network conditions, including multiple heterogeneous networks, device-to-device ad-hoc communications, different types of mobility models and also multipath and multi-hop communications. FlyNetSim simulator also is more efficient compared to others in terms of scalability and computational resource usage and hence, can support the complex tasks of information autonomy framework. Using this simulator, we simulate the real-world network conditions with varying load and mobility to reinforce our conjecture about the variability in response time of the system.

In Figure 4 we provide temporal traces generated using FlyNetSim [6]. The traces show Δ_{data} over time in our object detection setup, where the UAV transfers a picture to the edge server using Wi-Fi as a function of the number of competing wireless nodes and their transmission rates. It can be observed that as the traffic generated by each node increases (see Figure 3a), not only the average delay increases, but the temporal variations across subsequent frames become more apparent. This is due to the complex interactions between the streams induced by the transmission and transport layer protocols, which manifest especially as the sum traffic approaches the maximum link capacity. Similarly, if we increase the number of competing nodes (see Figure 3b), we will reach a point in which packet failures or timeouts will induce large variations. The bottom plot (see Figure 3c) in the picture shows the abrupt variations in the delay Δ_{data} induced by random motion of the UAV when adaptive rate is used. Importantly, the underlying conditions, channel gain, number of nodes, task load, vary over time and space, presenting different trends for different context.

From the considerations, it is clear that offloading may not be consistently beneficial, at least from the point of view of

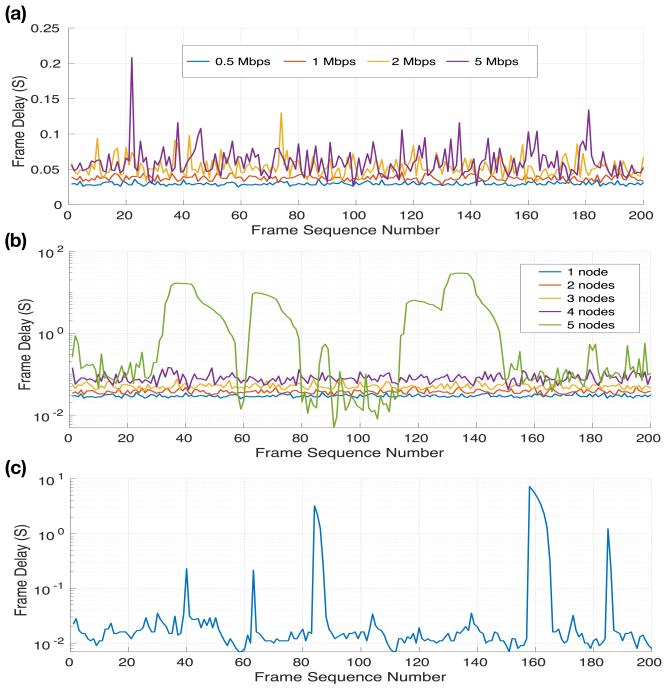

Fig. 4: Frame delay w.r.t. varying network loads & mobility.

capture-to-output delay. Moreover, the dynamics of capture-to-output delay highly depend on the characteristics of the surrounding environment. A simple reasoning and modeling on the key factors – network rate and server load – can be found in our recent paper [8].

Related Work: Information Autonomy problem presented in this paper is a form of Multi-Criteria Decision Making (MCDM) problem [9]. The configuration space for the Information Autonomy consists of multiple parameters that control the wireless performance, processor efficiency, energy consumption, and response time of the application with the goal of maximizing the mission time. Existing self-adaptation approaches such as Model At Run-time [10], [11] and control theory methods [12] can be applied to achieve information autonomy for Edge-assisted UAV systems. However, such approaches are not optimal and may have to be carefully tuned to reduce "capture-to-control" delay.

IV. Information Autonomy

Our simulation results in Section III-A show that a predefined offloading strategy would fail to harness the performance boost granted by infrastructure-assistance, or would possibly even degrade mission performance. We contend that due to the complexity of the involved dynamics, UAVs will need to implement a form of autonomy encompassing this relevant aspect of the system. We refer to this component of autonomy as *information autonomy*. Recall that the mission of information autonomy is to minimize the capture-to-output time Δ by selecting the best processing pipeline, either local or distributed over the infrastructure. As illustrated in Fig. 1, the UAV may have multiple options in terms of network and edge server.

Structurally, the pipeline to realize information autonomy is analogous to that shown in Fig. 3. However, the modules need to be specialized to the rather difficult problem of managing information in the composite UAV-infrastructure system.

A. Profiling

In most autonomous systems, this part of the pipeline is composed of sensors directly collecting signals from the environment. In the context of information autonomy, the Profiling module is assigned the difficult task of extracting information, a signal, from the infrastructure enabling the estimation and/or prediction of the state of entire communication-processing sections of the infrastructure. We identify two main strategies, which we refer to as inquiry and probing.

Inquiry: The sensing module sends direct inquiries to the infrastructure. For instance, the module can obtain from the network the expected transmission rate, and the number of current tasks – being executed or queued – from the edge server. This method presents two main disadvantages. First, the infrastructure would need to implement protocols to accept requests and compose replies. Additionally, in many network technologies the use of a channel would induce complex interactions with other active data streams, altering the available resource.

Probing: A simpler and possibly effective option is to introduce forms of probing, where the sensing module would "test" available pipelines by issuing tasks over the resource. Probing does not require any modification to the infrastructure, but incurs a drawback: the probe will use network and server resources, and for this reason probe size has to be finely tuned. This approach allows an objective observation of the state of the pipelines, but imposes a considerable burden to the infrastructure.[1]

A possible strategy to reduce the burden on the system is to reduce the size or frequency of the tasks sent within the probes. However, a reduced size or frequency of the probes may reduce the amount of information gathered on the state of the pipeline, as a small probe may "traverse" the system unaltered, whereas a full task may induce complex effects.

B. Analysis

The Analysis module transforms the output of the Profiling module – for instance a vector of delays obtained using probes – into features of the sensed pipeline state. Intuitively, as the objective of information autonomy is to select the best pipeline, the features should allow prediction of future performance of available pipelines. Taking as example the traces shown in Section III-A, the Analysis module could attempt to predict future delays or extract properties of their distribution to inform pipeline selection.

A major problem in this part of the information autonomy is the temporal scale of analysis and prediction. The module

[1] Note that if the full task is sent over multiple – independent – pipelines, the UAV can use the output generated in the smallest time, thus reducing the impact of unexpected variations through diversity.

could extract a long-term state of the pipeline, corresponding for instance to the number of active nodes and their traffic in our illustrative example. Tracking such state would roughly correspond to mapping samples to a long-term distribution of end-to-end delays, and characterizing their components to plan the appropriate action. Alternatively, the module could maintain a running predictor, continuously predicting future delays from a set of samples. This latter approach allows faster reaction to the variations that exist within the logical states, thus possibly achieving better performance compared to the former approach.

C. Control

The Control module transforms the features produced by the Analysis module into decisions. In this context, decisions control the activation/deactivation of pipelines, as well as probing strategies. For instance, a perceived degradation in the performance of the currently used pipeline, the Control module could activate probing over available pipelines. Minimizing Δ, the capture-to-control delay requires continuous tuning of system parameters, such as sampling frequency, offloading strategy and analysis functions. Note that multiple pipelines could be kept active during transitions to avoid mission interruptions.

V. Architecture

In developing a solution for Information Autonomy for UAV systems, we use the well-known Monitoring-Analysis-Plan-Execution-Knowledge (MAPE-K) [13] architecture, and tailor our framework (see Figure 5) to circumvent, at least in part, the run-time uncertainties as described in the previous sections.

Consider an autonomous system with sensors s_i producing the data point $(t_j, y_i(t_j))$, tuple corresponding to the value read at time t_j. In order to combine such information in complex data types we introduce the notion of Virtual Sensors (VS), where the $n-th$ VS is represented by a sequence $\text{VS}_n^\delta = \{r_1, ..., r_k\}$ of one or more readings from sensors $\{s_i\}_{i=1,...,m}$. The parameter δ is the time interval between new sequences are produced. Every sequence is the concatenation of several sub-sequences corresponding to data from the same sensor. One important challenge is to use algorithms that combine these multimodal data, merging data produced at different time instants. Note that from this mapping we can determine the minimum update frequency of sensors $\{s_i\}_{i=1,...,m}$, and optimize energy consumption.

When a new sequence is produced by VS_n^δ, it is encapsulated in a Data Block (DB) that includes some metadata, e.g., type, before submitting it to central Buffer (B), implemented as a priority queue, with priority function $\mathcal{P}$. The highest priority information in the buffer is then redirected to the next stage using policy $\mathcal{F}$, that consists of a map associating every DB of type τ to the function $f_a(\cdot)$, and then, *a)* triggers a State update $X(t) \rightarrow X(t')$ of the device and *b)* enriches the DB with a selection of state variables $\chi(t') \subseteq X(t')$ as it appears at time t'.

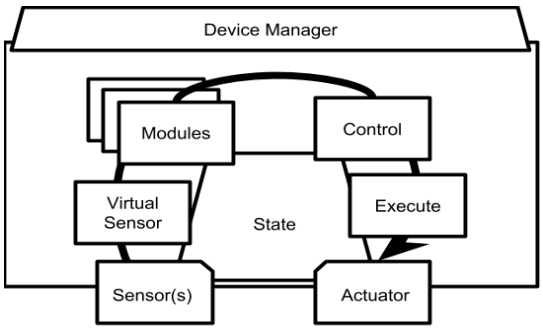

Fig. 5: Graphical representation of the proposed framework and its control flow

We model the data analysis operations as a concatenation of functions $f_a \circ f_\circ ... \circ f_z(\cdot)$, after each of which a DB is submitted to the buffer B and the priority of such pipeline is re-evaluated. Since the state is explicitly part of the input DB, and all the state changes are applied when a DB is redirected after passing through the buffer, each function is by design isolated and stateless, can be easily offloaded to other devices. These functions are part of the Modules, that include even other pre-control functionalities, such as outgoing communications.

A control mechanism follows the data analysis and allows us to plan (as in MAPE) an action to modify the current state towards the desired one. This can consist in steering the drone in a direction or change one of the control parameters, such as the frame rate. The execution module follows, allowing us to implement the change, interacting with drivers of physical components or applying the execution required.

Let's now observe the behaviour of the listed components in a simple scenario involving one UAV and an edge server running the same software architecture. Consider a $VS_{0.1}^1 = [r_0, r_1, r_2]$, producing a new sample every $0.1s$, containing three readings, where r_0 is a frame and r_1, r_2 are the most recent data points available from accelerometer and gyroscope. First a Data Block is produced by the virtual sensor and enqueued into the buffer B. When it is served, function f_a is applied, producing a new DB containing the same frame rotated to straighten the horizon. The produced DB is again enqueued in B and ready to be redirected to a pedestrian detection function. Suppose the link with the edge is fairly strong, and therefore the mapping rule $\mathcal{F}$ calls an offloading function, that sends the DB to a connected edge server. Now the DB is received at the ES, where the same architecture is running. A virtual sensor is listening for incoming DBs, and when receives the data, it enqueues the DB in the local buffer. Thereafter the DB is redirected using the map $\mathcal{F}_E$, that is computed taking into account the edge characteristics and state. The result is then forwarded to the communication module again, now sending to the UAV the resulting DB containing the estimated position of the object to track. The DB is now concatenated with some mission parameters, before being forwarded to the control module. In this case we use the state information attached to decide if the detected object must be tracked or avoided. Finally, the Execute module converts the control DB into an actionable

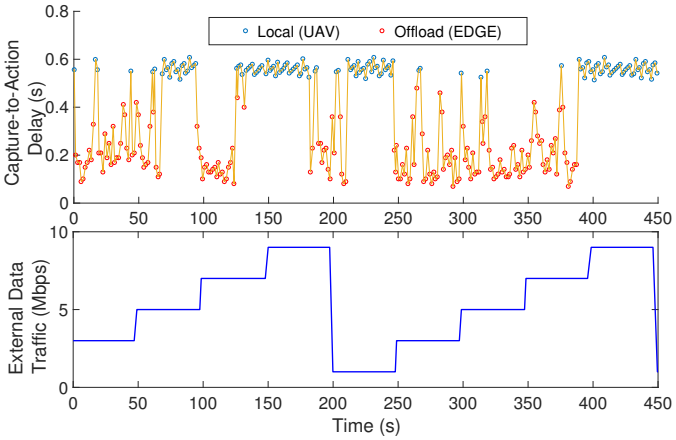

Fig. 6: Temporal variation of frame capture-to-action delay based on instantaneous offloading decisions w.r.t. variations in external traffic load.

packet that is sent to the motor's driver changing the rotors' speed and performing the required maneuvers.

VI. Preliminary Results

We experimented with an application scenario where a UAV captures video for object detection based on which it performs control action to follow a target object. The object detection is done by Haar-cascade classifier which can be computed either by the on-board computer or by the edge server depending on the resource availability. Using a preliminary version of the system described in Section V, and data collected during field experiments, we have been able to observe the adaptation of the system to a competing traffic scenarios.

We also use FlyNetSim [14] software simulator to simulate a network with five nodes: a UAV and an Edge Server, and other three nodes competing for the same WiFi channel. Fig. 6 shows the effect of the interfering traffic on the capture-to-action delay, the time interval between when the production of the frame at the virtual sensor, and the beginning of the execution of the maneuver. In this case our goal is to increase the performance, i.e. minimizing the end-to-end delay Δ. We accomplish this through a simple policy that uses the same object detection algorithm both at the UAV and the ES, selecting for each frame the first control result that reached the executor. It can be observed that in this case, given the deterministic nature of the local processing, the resulting end to end delay of UAV is always around $0.6s$, bounding the worst case, and guaranteeing increase in performance.

We can observe important variations and complex behaviours of the resulting capture-to-control delay. For example, notice the effect of the preceding high traffic over the capture to control delay after $t = 200s$, even though the competing nodes are using low amount of communication resources between $[200, 250]s$. These preliminary results justify how in highly dynamic environments show complex behaviors due to intricacies in system protocols and their variability and hence, an autonomous system must implement some form of information autonomy to handle information-related tasks.

In our ongoing work, we are developing object detection and RF localization applications using the "Information Autonomy" architecture presented in this paper to validate the effectiveness of our framework further.

VII. Conclusions

In this paper, we have addressed the challenges of edge assisted autonomous systems in dynamic environment scenarios and proposed a distributed framework for information autonomy to accomplish the mission overcoming the constraints. The preliminary version of the framework is demonstrated with an object detection based UAV mission, and preliminary results from the implementation exhibit the importance of autonomous decisions to achieve the goal for the mission-critical application. In our future work, the framework will be extended with more parameters for a fine-grained response of the information autonomy system and optimization of the resource usage.

References

[1] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, no. 5, pp. 2795–2808, 2016.

[2] J. Dick, C. Phillips, S. H. Mortazavi, and E. de Lara, "High speed object tracking using edge computing," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 26.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[4] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2831347.2831354

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *arXiv preprint arXiv:1702.05309*, 2017.

[6] S. Baidya, Z. Shaikh, and M. Levorato, "FlyNetSim: An Open Source Synchronized UAV Network Simulator based on ns-3 and Ardupilot," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, 2018, pp. 37–45.

[7] L. Meier, J. Camacho, B. Godbolt, J. Goppert, L. Heng, M. Lizarraga *et al.*, "Mavlink: Micro Air Vehicle Communication Protocol," *Online]. Tillgänglig: http://qgroundcontrol. org/mavlink/start.[Hämtad 2014-05-22]*, 2013.

[8] D. Callegaro and M. Levorato, "Optimal computation offloading in Edge-Assisted UAV systems," in *2018 IEEE Global Communications Conference: Selected Areas in Communications: Tactile Internet (Globecom2018 SAC TI)*, Abu Dhabi, United Arab Emirates, Dec. 2018.

[9] S. Greco, J. Figueira, and M. Ehrgott, *Multiple criteria decision analysis*. Springer, 2016.

[10] D. Weyns, M. U. Iftikhar, D. Hughes, and N. Matthys, "Applying architecture-based adaptation to automate the management of internet-of-things," in *Software Architecture - 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24-28, 2018, Proceedings*, 2018, pp. 49–67. [Online]. Available: https://doi.org/10.1007/978-3-030-00761-4_4

[11] L. Garcia Paucar and N. Bencomo, "Runtime models based on dynamic decision networks: enhancing the decision-making in the domain of ambient assisted living applications," in *MRT 2016 - Models@run.time*, ser. CEUR workshop proceedings, S. Götz, N. Bencomo, K. Bellman, and G. Blair, Eds. CEUR-WS.org, 11 2016, pp. 9–17.

[12] S. Shevtsov, M. Berekmeri, D. Weyns, and M. Maggio, "Control-theoretical software adaptation: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 44, no. 8, pp. 784–810, Aug 2018.

[13] A. Computing *et al.*, "An architectural blueprint for autonomic computing," *IBM White Paper*, vol. 31, pp. 1–6, 2006.

[14] S. Baidya, Z. Shaikh, and M. Levorato, "Flynetsim: Flying and networking simulator," https://github.com/saburhb/FlyNetSim, 2018.