# A Measurement Study on Edge Computing for Autonomous UAVs

Davide Callegaro
University of California, Irvine
dcallega@uci.edu

Sabur Baidya
University of California, Irvine
sbaidya@uci.edu

Marco Levorato
University of California, Irvine
levorato@uci.edu

## ABSTRACT

The ability to execute complex signal processing and machine learning tasks in real-time is the core of autonomy. In airborne devices such as Unmanned Aerial Vehicles (UAV), the hardware limitations imposed by the weight constraint make the continuous execution of these algorithms challenging. Edge and fog computing can mitigate such limitations and boost the system and mission-level performance of the UAVs. However, due to the UAVs motion characteristics and complex dynamics of urban environments, the performance of pipelines using interconnected, rather than onboard, resources can quickly degrade. Motivated by the development of Hydra, an architecture for the establishment of flexible sensing-analysis-control pipelines over autonomous airborne systems, this paper reports a preliminary measurement study on the performance of computing task offloading on available network technologies in this class of applications and systems.

## KEYWORDS

Unmanned Aerial Vehicles; Edge Computing; Fog Computing; Autonomous Systems

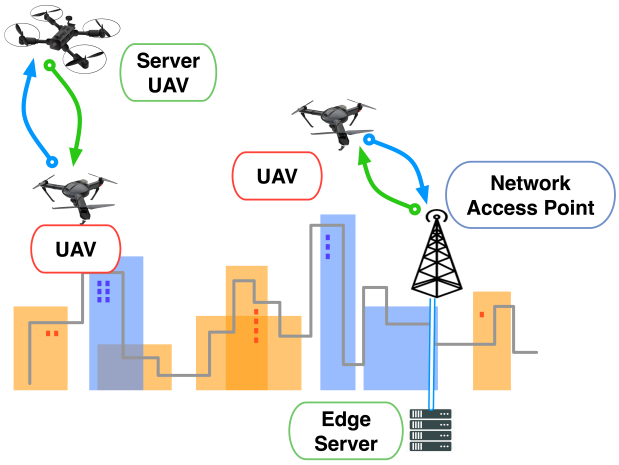


**Figure 1: Scenario considered in this paper: ground and airborne devices provide assistance to UAVs by taking over compute-intense tasks.**

## 1 INTRODUCTION

The ability to observe and analyze the surrounding environment to inform decision making is instrumental to achieve fully autonomous operations. Possibly complex signal processing and machine learning algorithms transform sensor feeds into control in real-time. For instance, recent contributions present Deep Neural Networks (DNN) architectures used to control navigation based on a video input. For instance, the DNN presented in [6] supports UAV navigation in city environments. However, the execution of sensing-processing-control pipelines supporting the autonomous operations of Unmanned Aerial Vehicles (UAV) faces technical challenges due to the inherent limitations of these airborne platforms. Traditional UAV design principles, where the whole pipeline is performed onboard an individual UAV may result in reduced functionalities and mission performance, a delayed response to stimuli due to large capture-to-control time, and a shortened mission time due to the additional energy consumption. The onboard execution of higher level functions, such as online learning, and general state-of-the-art algorithms supporting general mission objectives – *e.g.*, object detection – is a technological challenge to overcome.

Assistance from the infrastructure or other devices in the vicinity can mitigate such limitations. In line with the recent edge and fog computing paradigms [1], UAVs could offload compute-intense tasks to, more capable, interconnected devices, thus saving energy and possibly enhancing their ability to respond to external stimuli. However, while executing the whole sensing-analysis-control pipeline onboard results into a predictable capture-to-output delay, offloading introduces a considerable degree of uncertainty. In fact, as information is transferred over a wireless link and processed at a remote device, the state of these two components influences the total time from the acquisition of the information to the availability of control at the UAV. Hardly predictable and often unknown parameters such as channel gain, interference, network load and server load all influence the statistics of total delay. Even when all these parameters are fixed, complex protocol interactions and environment features induce patterns that are difficult to control or characterize.

Motivated by the development of Hydra, an architecture for resilient collaborative computing within distributed airborne and ground systems, this paper reports an empirical study on the performance and characteristics of edge computing for this class of applications. To perform the study, we developed a modular sensing-analysis-control architecture supporting the offloading of general computation tasks. The architecture allows the distribution of modules over wirelessly interconnected systems comprising airborne and ground resources. A case-study centered on object tracking was developed where the UAV's navigation is based on the bounding box outputted by the algorithm. Intuitively, the time between the capture of a picture and the generation of the control (capture-to-control time) determines the ability of the UAV to follow a target

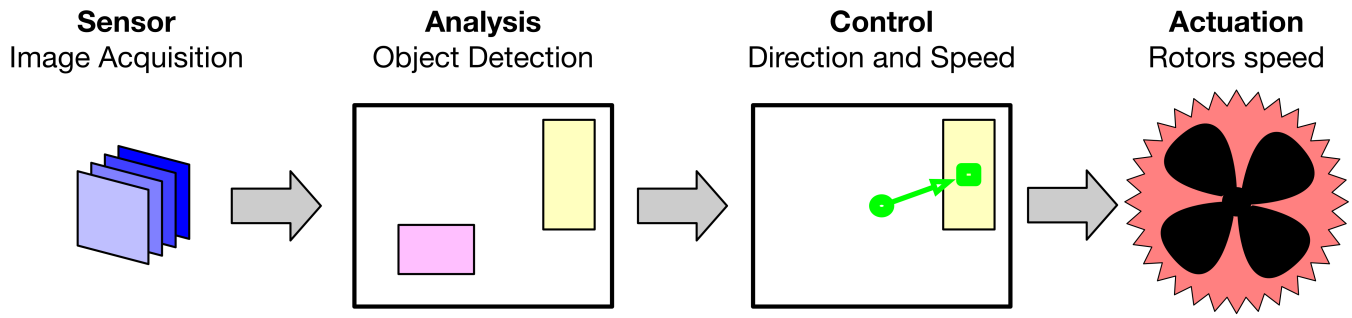


**Figure 2: Application considered in the experiments: an autonomous UAV captures images from an onboard camera. An object detection algorithm is used to detect objects of interest. The bounding box is used to derive speed and direction and control the rotors.**

object. The measurement study in this paper reports the spatio-temporal characteristics of the capture-to-control time in a real-world platform for two key communication technologies: Wi-Fi and LTE. The latter is emulated using Software Defined Radios – both at the UAV and edge server – and the srsLTE [2] open-source software.

The rest of the paper is organized as follows. Section 2 describes the application considered in this paper and makes some initial considerations in the implications of edge or fog computing. In Section 3, we present the experimental platform used to derive the results discussed in Section 4. Section 5 makes some final remarks on the experience reported in this paper, and Section 6 acknowledges the sources of support.

## 2 APPLICATION AND PRELIMINARY DISCUSSION

We make our discussion specific to a relatively simple, but representative, case-study, where the task of the UAV is that of tracking objects in a predetermined class *e.g.*, pedestrians or vehicles. Figure 2 depicts the schematics of the application. The UAV acquires images at a constant rate. Each image is analyzed using an object detection algorithm to detect classes of objects of interest. The bounding box of a selected object detected by the algorithm is used to determine the motion of the UAV. Specifically, a control function computes the vector from the center of the image to the center of the corresponding bounding box. The vector is, then, converted into speed and direction commands.

We define the time from image capture to the emission of the control *capture-to-control delay*. Intuitively, minimizing the capture-to-output delay is critical to achieve a fast response of the UAV to movements of the target object and an effective tracking. The main bottleneck of the sensing-analysis-control pipeline is the analysis algorithm. State of the art object detection is realized via sophisticate algorithms, such as Deep Neural Networks (DNN). An example is RetinaNet [5], which features Pyramid Network backbone on top of a feedforward ResNet architecture (50 to 100 layers depending on the version). These DNNs provide a high accuracy both in terms of classification and precision of the bounding box, thus enabling accurate navigation in the considered application.

However, the execution of such complex algorithms on constrained embedded devices may be simply unfeasible, or result into excessive capture-to-control delay and energy consumption. Simplified versions of these DNNs reduce computational complexity at the price of a degraded performance. Yolo [7] is an example of this kind of approach, a highly optimized software which degrades the accuracy of the output, especially in terms of bounding box precision. Versions of these DNNs specifically tailored to mobile devices are also available [3, 4] corresponding to various points in the complexity-accuracy tradeoff.

Offloading these compute-intense tasks of the pipeline to interconnected compute-capable devices, such as edge servers, can significantly reduce the capture-to-control delay and decrease energy intake. However, offloading requires to wirelessly transfer the signal produced by the onboard sensors and the output remotely produced. In the considered application scenario, the captured images from the onboard sensor are transferred at regular intervals to the server, which executes the DNN (analysis) and computes the control commands (control) which are then sent to the UAV.

As mentioned earlier, offloading introduces uncertainty:

- **Wireless link:** The continuous flow of images and control messages is transported over a wireless link, whose short-term capacity is influenced by time-varying factors such as channel gain, interference, network load, and fine-grain protocol interactions.
- **Server:** The time frame of execution of the task at the server is a function of time-varying load, as well as of resource allocation strategies.

The objective of this paper is that of analyzing the capture-to-control delay patterns over different communication technologies in a real-world platform.

## 3 EXPERIMENTAL PLATFORM

The experimental platform we developed realizes the application described in the previous section over a distributed wirelessly interconnected system. In the following, we describe the hardware and software components.

### 3.1 Hydra and Hardware

We briefly describe the overall system our development effort is aiming at to provide some context to this study. HyDRA – Resilient computing for Heterogeneous Autonomous Devices – is a middleware architecture enabling the dynamic migration within

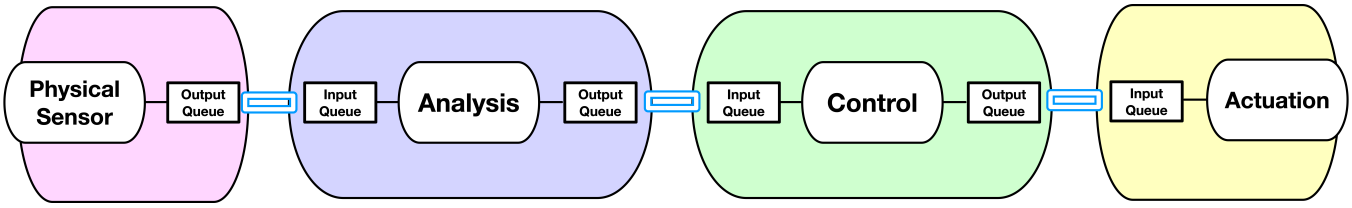


**Figure 3: Modular architecture developed for the experiments. The modules wrap functions adding input/output functions for routing and filtering.**

interconnected systems of modules composing pipelines for autonomy. The objective is to maintain a capture-to-control delay as low as possible, while minimizing the amount of channel and computation resources used. The system is hierarchical and composed of three layers: UAV-users, UAV-servers and ground servers.

We aim to explore various processing and communication options. Therefore, we equip the UAVs and ground servers with an embedded device and a Software Defined Radio (SDR).

- **UAV-User:** A lightweight 3DR Solo UAV is equipped with a gimbal and a GoPro camera. A 3D printed structure is added to support an Up Board board, an Ettus Research USRP B200mini and a Lithium High Voltage (11.4V/5.2 Ah) battery. The default on-board computer (IMX6) on the 3DR solo is only used to expose the Telemetry2 ports of the Pixhawk flight controller to be connected to UPboard via USB serial. The UP Board runs Ubuntu 16.04 OS to support all the on-board computing software and also the software suite for radio communications.
- **UAV-Server:** Large UAV International X6S *UAVs* are equipped with a Jetson NVIDIA TX2 and an Ettus Research USRP B210. The Jetson TX2 board has an NVIDIA Pascal-family GPU for high performance computation, and runs Ubuntu 16.04 with Kernel 4.4.
- **Ground Station - Server:** Ground edge servers are realized using a laptop with Intel Core i7-7700HQ CPU (8 cores, 2.8 GHz) and 16 GB RAM connected to the access point or base station.

All the devices are also equipped with a Wi-Fi dongle to enable Wi-Fi communication capabilities.

## 3.2 Software

Fig. 3 shows the modular pipeline developed to allow the distribution of analysis and control functions over the interconnected system. Each module encapsulates a function, and includes an input and an output queue.

The modules logically abstract the functions composing the pipeline, isolating them from adjacent operations. Modules are atomic and can be interchanged easily, and even dynamically. The modularization is necessary to achieve multi-threaded parallelization in-device Naturally, modules that interact with the environment do not have "data" input (*e.g.* Sensor) or output (*e.g.*, Actuation).

The input and output queues provide flexibility to the flow of information throughout the pipeline. We remark that in our study some input and output queues and interconnected through sockets latched to the Wi-Fi or LTE stack. Queueing is critically necessary due to uncertainty in the time needed to traverse modules. For instance, as the capacity of the link connecting the UAV-user to a server changes, the timing of image forwarding may increase, so that queueing is necessary. Herein, we set the queue size to 1, and implement an aggressive preemptive policy at all modules, where new captured images, analysis output and control take the place of older one stored in the queue. This allows the minimization of the accumulation delay. Logging for the evaluation of performance metrics, such as the capture-to-control delay, is performed at the input and output queues. Specifically, the capture-to-control delay is the time interval delimited by the insertion of the data point into the Sensor's output queue and the insertion of the control message into the respective output queue.

The full HyDRA architecture includes dynamic routing within multiple pipelines and a logic controlling the routing paths based on system logs.

For LTE communications, the user UAVs are configured as User Equipment (UE) using srsLTE version 18.03 compliant with 3GPP release 8. We use srsLTE version 18.12 configured as eNodeB and core network (EPC) at the airborne and ground servers.

# 4 EXPERIMENTAL RESULTS

## 4.1 Setup

Figure 4 shows the setup of the UAV with the hardware components used for communication, computation and control. The experiments were conducted in a large outdoor environment. The UAV and ground server were connected using Wi-Fi (through Wi-Fi dongles) or LTE (through programmable radios implementing srsLTE).

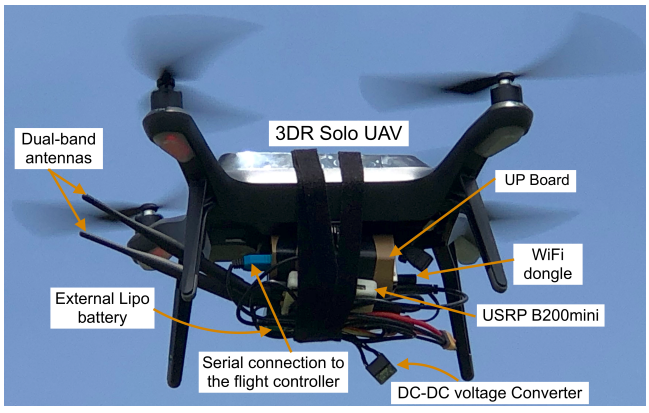


**Figure 4: The UAV setup used for outdoor experiments.**

| Parameters | Value |
|---|---|
| WiFi Standards | IEEE 802.11b and 802.11n |
| WiFi Bandwidth | 20 MHz |
| WiFi Band | Channel 1 (2.412 GHz) |
| LTE Downlink EARFCN | 3400 |
| LTE Bandwidth | 10 MHz |
| LTE Antenna Model | SISO |
| LTE UE Tx antenna gain | 110 dB |
| LTE eNodeB Rx antenna gain | 40 dB |
| LTE RLC Mode | Acknowledgement (RLC AM) |
| LTE MAC Scheduler | Round Robin |
| TCP Congestion Control | Cubic |

**Table 1: Experiment Parameters used for WiFi and LTE**

The movements of the UAV are fully autonomous based on a programmed mission on the flight controller. In order to characterize the application over capture-to-control delay over space and time, we disabled the function allowing the UAV to follow an object and, instead, programmed a predefined flight plan. Specifically, the UAV moves autonomously in steps to reach a specific distance and hovers for 1 minute at each distance.

We focus on a topology with a single UAV and ground server. The ground server runs on the same laptop that is configured as AP or base station, thus ensuring minimal delay between reception and processing. The communication parameters used for WiFi and LTE experiments are reported in the table 1.

In order to measure the impact of external traffic on the delay patterns, we set a second stationary user in the vicinity of the access point (AP) or the base station and generate periodic large data frames with different data rate over TCP. We remark that the area was not isolated from external communications and interference.

In the experiments, the sensor module acquires images (480×640 RGB pixels, compressed using JPEG to 26 KB). The analysis module executes a lightweight DNN for object detection (SSD-lite mobilenet v2 2018-09). Herein, we limit our study to pipelines originating from UAV-users and positioning analysis at the ground station-servers. The execution of the DNN model takes approximately 1 and 0.07 s on the Up Board and ground server, respectively.

## 4.2 Results

Fig. 5 shows the temporal trace of the capture-to-control delay (blue line) using IEEE 802.11n obtained as the UAV flies away from the ground station. The orange line reports the distance from the ground station based on the received telemetry. It can be seen that when the distance is sufficiently small, the high throughput of the IEEE 802.11n grants a total delay of approximately 0.1 s, composed of approximately 0.07 s to execute object detection at the ground station and 0.03 s to transport the image and the control over wireless. As the distance increases both the average delay and its variance grow until disconnection occurs at ∼ 35 m.

Fig. 12 shows the same trace – this time with the UAV approaching the ground station – when another node is active in the system producing 15 Mbps. We observe a generally increased delay floor (approximately 0.15 s average) and a much higher overall delay variance. High peaks of delay may correspond to TCP timeouts

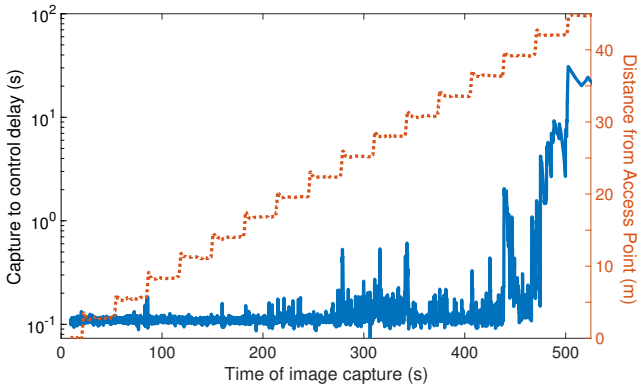


**Figure 5: Capture-to-control delay over WiFi 802.11n. In this flight, the UAV-User moves away from the ground base station in free space.**

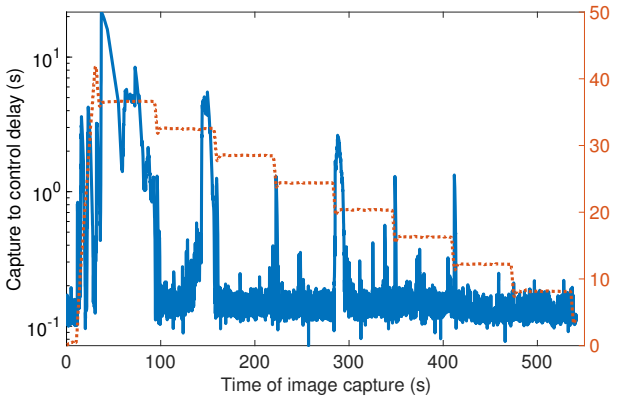


**Figure 6: Capture-to-control delay over WiFi 802.11n. In this flight, the UAV-User moves toward the ground base station in free space. An external node produces traffic at 15 Mbps.**

due to the interference load. Interestingly, sharp spikes are also present in correspondence with UAV acceleration from a stationary position. Similar, but less prominent, spikes could be noted in the previously shown trace, with a possibly faster "absorption" from the system due to the typically smaller delay. Although we do not have an explanation supported by conclusive evidence, the spikes could be generated by antenna misalignment when the UAV tilts to accelerate, or temporary large channel estimation error.

Fig. 7 shows the Cumulative Density Function (CDF) of the delay for different traffic rates of the external node. The CDF is calculated by taking the average over each second, and then computing the CDF over the obtained delay. This to have a more fair weight in the CDF As expected, a general degradation of the delay as contention increases is present. Interestingly, the contention scenarios in the 0 – 10 Mbps range share a step-like shape of the CDF, where the key difference is the probability of high-delay spikes. Higher volumes of the contending data stream reduce the steepness of the step, introducing mid-range delays.

Fig. 8 compares the CDF achieved when using IEEE 801.11n and b. The lower maximum throughput offered by the latter is apparent. With no external traffic, the IEEE 801.11b suffers a small

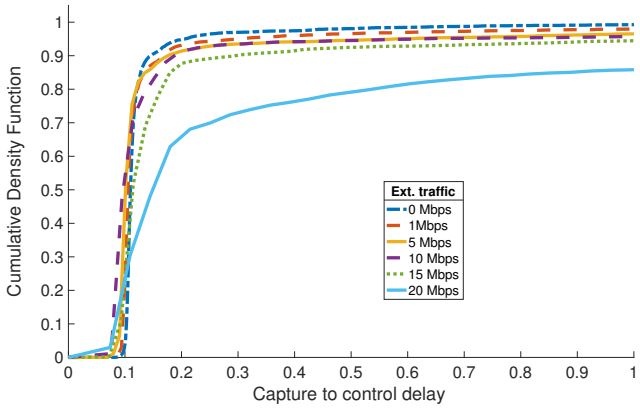


**Figure 7: Cumulative density function of the capture-to-control delay for different volumes of external traffic.**

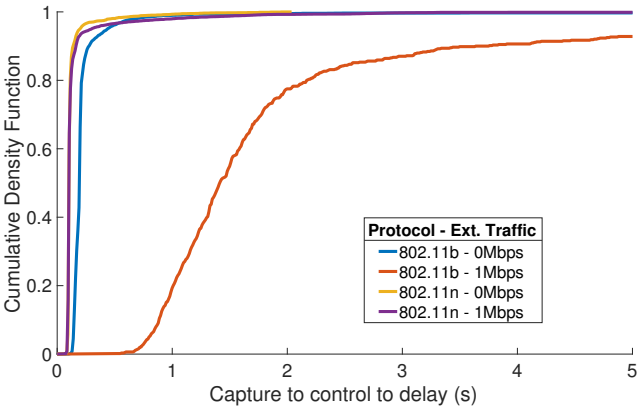


**Figure 8: Cumulative density function of the capture-to-control delay for different volumes of external traffic and IEEE 801.11 versions .**

degradation in the minimum delay compared to n, due to the fact that the communication time is a small part of the total delay. However, 1 Mbps of external traffic are sufficient to dramatically increase the delay, introducing a substantial fraction of delays in the $1-2$s range.

We show in Fig. 9, which depicts mean and variance of capture-to-control delay for IEEE 802.11n as a function of the external traffic, that not only the mean of the delay increases, but also its variance.

Fig. 10 shows the average number of frames per second processed by the system. The markers correspond to different traffic rates and IEEE 802.11 versions (n in red and b in green). We remark that while the emission rate of frames from the sensor is fixed, the preemptive queueing policy may eliminate frames from any queue, thus subsampling the generated sequence as it is transformed. The high sensitivity of IEEE 802.11b to external traffic is apparent, making this protocol unsuitable to establish stable pipelines. Due to its high throughput, the IEEE 802.11n provides stable $8-10$ frames per second until degradation occurs at around 20 Mbps. We observe that the on-board processing pipeline would sustain about 1 frame per second.

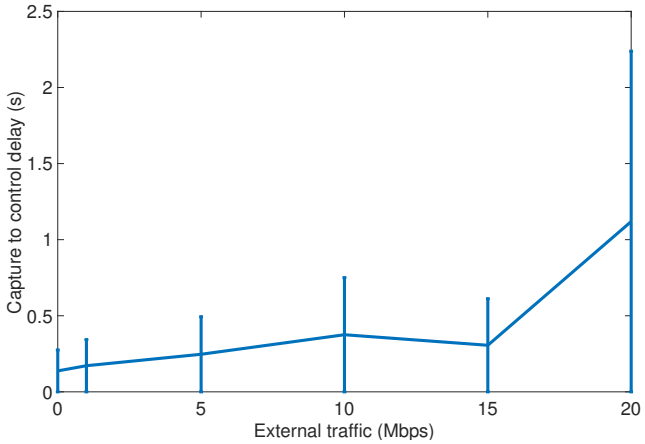


**Figure 9: Average and standard deviation of capture to control delay, taken over a sequence of flying experiments, each with a constant external traffic rate. Distance from Access Point between 0-35 m.**

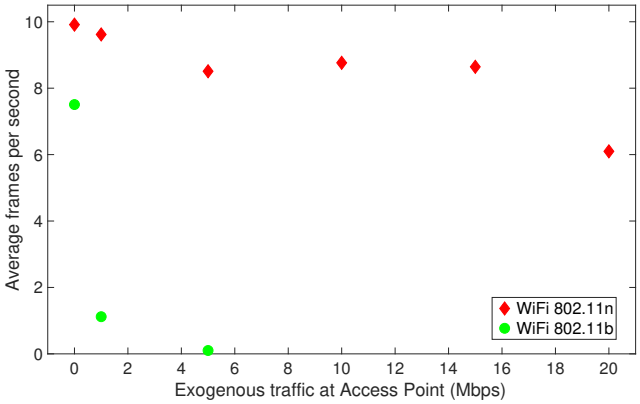


**Figure 10: Average number of frames per second processed by the system for different IEEE 802.11 protocols and external traffic rates.**

Fig. 11 shows the mean and variance of delay achieved when using LTE over SDRs as a function of the distance. The blue and red lines correspond to stationary measurements on the ground and actual flight. Interestingly, it can be observed a substantial difference between ground measurements and flight. This may be caused by several factors including antenna emission patterns, UAV motion, and a larger interference from active LTE base stations perceived at a higher altitude. We observe that the SDRs have power constraints which degrade the system performance compared to actual cellular systems, this limits the range to tens of meters – vs hundreds of meters in commercial LTE systems. The trace reported in Fig. 12 shows a rapidly increasing number of delay spikes as the distance between the UAV and the ground server increases. We note that the delay floor is at about 0.15 s, and disconnection occurs at 20 m. We remark that experiments are not performed in an isolated environment, and some high-delay sections may be caused by other active LTE emitters. Future investigations will use

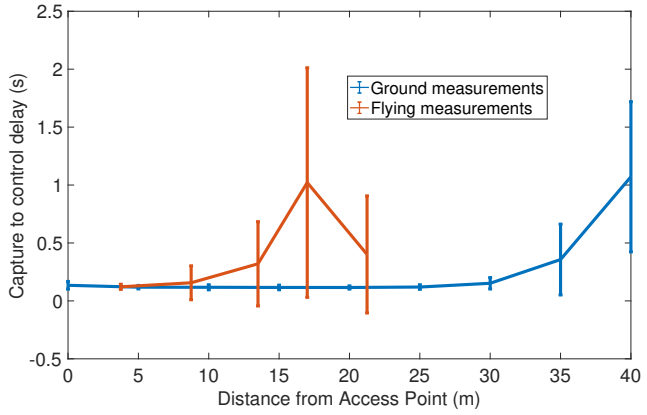


**Figure 11: Mean and standard deviation of capture-to-control delay using Software Defined Radios emulating an LTE network. The blue and red lines correspond to stationary measurements on the ground and actual flight.**

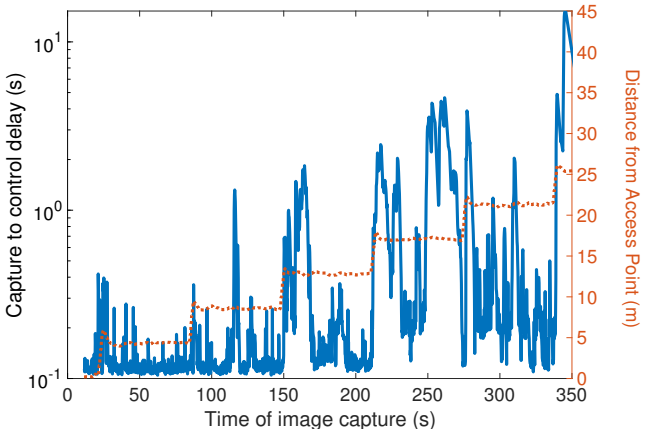


**Figure 12: Capture to control time of the distributed pipeline, over LTE with Software Defined Radios.**

other radios to measure the incoming energy in the used frequency band and correlate it with delay.

### 4.3 Discussion

The results showed in this section emphasized one critical aspect of offloading in this class of applications: wireless links from airborne devices are extremely unstable. Sensitivity to distance, physical orientation and exogeneous environmental conditions – such as other active data streams – was observed in the experiments. In Wi-Fi-based communications, interactions between Medium Access Control (MAC) and transport layer protocols of different devices increase delay and delay variance at the small and large scale of spatio-temporal trajectories. LTE offers a more stable multiplexing strategy. However, at least in the SDR-based setup used in our testbed, the delay patterns show a high variance, possibly motivated by physical layer factors such as uncontrolled inter-cell interference, antenna alignment and limited power.

Clearly, relying on one communication-computation loop to perform critical tasks can greatly harm the overall ability to reliably control the UAV. This further motivates our effort to develop a system capable of moving sensing-processing-control pipelines dynamically across resources in response to perceived changes in the environment.

## 5 CONCLUSIONS

Motivated by the development of Hydra, an architecture for resilient collaborative processing over hierarchical UAV systems, we presented a measurement study on performance achieved migrating computing tasks from an autonomous UAV to an interconnected server. We specifically focused on capture-to-control delay obtained by a system performing object detection-based navigation from an on-board camera feed. Our setup includes various communication options: IEEE 801.11b and n and LTE emulated on SDRs using the srsLTE open-source software. The collected measurements emphasize the instability of the links during flight, and the need for a further layer of intelligence to dynamically select the best available option.

## 6 ACKNOWLEDGEMENTS

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

## REFERENCES

[1] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 13–16.

[2] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. 2016. srsLTE: an open-source platform for LTE evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. ACM, 25–32.

[3] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[4] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. 2017. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7310–7311.

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[6] Antonio Loquercio, Ana I Maqueda, Carlos R del Blanco, and Davide Scaramuzza. 2018. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters* 3, 2 (2018), 1088–1095.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.