2023 International Conference on Unmanned Aircraft Systems (ICUAS)
June 6-9, 2023 | Lazarski University, Warsaw, Poland

2023 International Conference on Unmanned Aircraft Systems (ICUAS) | 979-8-3503-1037-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICUAS57906.2023.10156100

UAV-based Networked Airborne Computing Simulator and Testbed Design and Implementation

Baoqian Wang[1] Junfei Xie[2] Ke Ma[1] Yan Wan[3]

***Abstract*—The integration of onboard computing capabilities with unmanned aerial vehicles (UAV) has gained significant attention in recent years as part of mobile computing paradigms such as mobile edge computing (MEC), fog computing, and mobile cloud computing. To enhance the performance of airborne computing, networked airborne computing (NAC) aims to interconnect UAVs through direct flight-to-flight links, with UAVs sharing resources with each other. However, despite the growing interest in NAC and UAV-based computing, existing studies rely heavily on numerical simulations for performance evaluation and lack realistic simulators and hardware testbeds. To fill this gap, this paper presents the development of two NAC platforms: a realistic simulator based on ROS and Gazebo, and a hardware testbed with multiple UAVs communicating and sharing computing resources. Through simulation and real flight tests with two computation applications, we evaluate the platforms and examine the impact of mobility on NAC performance. Our findings offer valuable insights into NAC and provide guidance for future advancements.**

***Keywords*—Unmanned Aerial Vehicles, Networked Airborne Computing, Simulator Design, Testbed Development.**

I. Introduction

Unmanned aerial vehicles (UAVs), when equipped with onboard computing capabilities, can function as servers to provide computing services to ground users [1]. Advanced onboard computing capabilities can also enhance many UAV functions such as positioning and image processing, hence benefiting a wide range of applications such as surveillance [2], search and rescue [3], remote sensing [4], and traffic monitoring [5]. These exciting advantages have motivated numerous researchers to investigate the integration of UAVs into the multi-access edge computing (MEC), fog computing, or mobile cloud computing paradigms [1], [6], [7].

As small UAVs are restricted in payload capacity, the onboard computing capability of individual UAVs is limited. To address this limitation, networked airborne computing (NAC) [8] arises that aims to maximize airborne computing resources through inter-vehicle resource sharing via direct flight-to-flight communication links. To enable NAC, a novel distributed computing framework for static UAV networks was introduced in [9], [10]. It optimizes resource allocation and utilizes coding techniques to achieve efficient and robust airborne computing. To address scenarios where UAVs may move during computing, a reinforcement learning-based algorithm for jointly optimizing computation load allocation and UAV trajectory was introduced in [11]. Besides theoretical investigation, a hardware platform was introduced in [12] to facilitate airborne computing research. This platform is equipped with NVIDIA Jetson TX2 of high computing power and also supports broadband wireless communication that can facilitate network-based UAV research. Nevertheless, its design is centered on a single UAV, and inter-vehicle resource sharing is not supported. Moreover, its communication module, equipped with a directional antenna, was designed and tested only for point-to-point communications. Nevertheless, NAC may involve point-to-multipoint or multipoint-to-multipoint communications.

In existing studies on UAV-based computing [1], [13]–[16], performance evaluation was typically conducted through simulations with UAV movement, communication, and computing behaviors described using mathematical models. While mathematical model-based simulations offer the advantage of being inexpensive and easy to deploy, their underlying models, due to simplicity, may not accurately reflect the intricate behavior of real UAV systems. Hardware tests have been limited to single or dual UAVs. For example, [17] investigated the application of airborne computing for computer vision and machine learning, and developed a UAV-based MEC system with two UAVs to analyze communication and computation delays for video streaming between the two UAVs.

Although the absence of realistic testbeds for NAC research remains, various testbeds have been created to aid research in UAV control, communication, and networking [18]–[24]. For instance, [18] created a testbed to investigate the communication and control in UAV swarms. [19] introduced a UAV hardware testbed for control and information acquisition. To support robotics applications, such as collision avoidance, [20] built a UAV simulator that runs on the cloud using Docker. In the realm of UAV communication, [21] presented a new evolved packet core (EPC) that can be placed directly on UAV to establish UAV-based LTE networks. They evaluated its performance in a two-UAV LTE network, which resulted in improved client connectivity. A ROS/Gazebo simulator was designed in [25] to emulate wireless communication among networked UAVs. [22] developed a directional antenna-based broadband and long-range communication system for UAV-to-UAV communication, which has the ability to automatically adjust antenna directions for optimizing communication performance

[1] Baoqian Wang and Ke Ma are with the Department of Electrical and Computer Engineering, University of California, San Diego and San Diego State University, La Jolla, CA, 92093 (e-mail: bawang@ucsd.edu).

[2] Junfei Xie is with the Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA, 92182 (e-mail: jxie4@sdsu.edu).

[3] Yan Wan is with the Department of Electrical Engineering, University of Texas at Arlington, Arlington, TX 76019 (e-mail: yan.wan@uta.edu).

Algorithm 1: Computing model in NAC

// **Master:**
Broadcast $\mathbf{X}$ to workers.
for $i = 1 : N$ **do**
  Listen to the channel and collect result $f_i(\mathbf{X})$ from worker i.
end
Calculate and output final result using $\{f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_N(\mathbf{X})\}$
// **Worker** i**:**
Listen to the channel and receive $\mathbf{X}$ from the master.
Compute $f_i(\mathbf{X})$ and send results back to the master.

in unknown communication environments. [23] proposed a new UAV network design using mmWave for Gigabit speed communication and evaluated it on a hardware testbed. Additionally, [24] created a UAV simulator to investigate the communication security issue.

Main contributions: In this paper, we aim to address the research gap regarding the scarcity of realistic testbeds for NAC research. To this end, two NAC platforms are designed and implemented. One is a realistic simulator created using ROS (Robot Operating System) [26] and Gazebo [27]. Another is a hardware testbed composed by multiple UAVs with computing and inter-vehicle resource sharing capabilities. Moreover, we conduct various simulations and real flight tests with two computation applications to examine the impact of UAV mobility on NAC. The obtained insights are vital for progressing NAC research by uncovering the barriers to achieving high-performance NAC. They also provide guidance for future enhancements of the proposed NAC platforms.

The rest of the paper is organized as follows. Sec. II presents the computing model for UAV-based NAC. Sec. III and Sec. IV describe the designs for the simulator and hardware testbed, respectively. The simulation and flight test results are presented in Sec. V and Sec. VI, respectively. Finally, Sec. VII concludes the paper.

II. Computing Model

In this section, we describe how a computation task is completed collaboratively by UAVs in NAC. Consider a NAC system with $N+1$ UAVs, where N UAVs are the workers and one UAV serves as the master. Suppose the master UAV needs to complete a decomposable computation task $\mathbf{f}(\mathbf{X})$, where $\mathbf{X} \in \mathbb{R}^{m\times n}$ is the input to the task. To enhance computational efficiency, the master partitions the task into N subtasks represented as $\{f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_N(\mathbf{X}))\}$, and assigns each worker i with subtask $f_i(\mathbf{X})$. During execution, the master shares the input $\mathbf{X}$ with all the workers, which then calculate their respective subtasks and send the results back to the master. The master then combines the results to produce the final output of the task $\mathbf{f}(\mathbf{X})$. Algorithm 1 summarizes the task execution procedure.

In subsequent simulation and experimental studies, we select two computation tasks to be executed using the above computing model. They are 1) matrix multiplication and 2) linear regression, both of which are fundamental components for more complicated computations such as those in machine learning applications [28], [29]. Here we briefly describe each of the tasks.

To perform a matrix multiplication task $\mathbf{AB}$, where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a pre-stored matrix and $\mathbf{B} \in \mathbb{R}^{n\times l}$ is the input, the master equally divides $\mathbf{A}$ row-wise into N submatrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$. Suppose each submatrix $\mathbf{A}_i$ is pre-stored in the corresponding worker i. The master distributes $\mathbf{B}$ to each worker, which then computes $\mathbf{A}_1\mathbf{B}, \mathbf{A}_2\mathbf{B}, \ldots, \mathbf{A}_N\mathbf{B}$, respectively, and returns the results to the master. The master then aggregates the received results $[\mathbf{A}_1\mathbf{B}, \mathbf{A}_2\mathbf{B}, \ldots, \mathbf{A}_N\mathbf{B}]$ to obtain the final result $\mathbf{AB}$.

For linear regression, we use a real-world data set collected in Montesinho park [30] as input to train a forest fire detector. The goal is to predict burned areas given meteorological information by training a linear regressor using gradient descent. The data set consists of 517 meteorological data samples, each containing information such as location in the park, time, temperature, humidity, wind speed, and the burned area of the forest. To train the linear regressor in NAC, the data set is divided into N equal parts, with each part stored in a worker. The master starts by initializing the parameters of the linear regressor and updating them iteratively. During each iteration, the master sends the current parameters to each worker, which calculates the gradients using their local data and returns the gradients to the master. The master then updates the parameters using the received gradients.

III. Simulator Design

This section presents a ROS- and Gazebo-based simulator designed to provide a realistic simulation environment for NAC research. The simulator (see Fig. 1) consists of five modules: UAV hardware module, controller module, visualization module, wireless communication module, and computation module. These modules communicate through ROS topics [31]. The functionalities of each module and their associated ROS topics are described as follows.

A. UAV Hardware Module

The UAV hardware module simulates the physical attributes of a UAV including its frame, motor, and sensors. Users can configure UAV's frame such as base size, weight, shape, arm length, etc. using the configuration file based on the application needs. Additionally, users can select different types of motors to simulate UAVs of different sizes and aerodynamics.

ROS offers plugins for various sensors that can be attached to UAV to track its states, such as position, velocity, and orientation. In our simulator, each UAV is equipped with a GPS sensor for localization, a magnetic sensor for orientation, an IMU sensor for acceleration, and a barometer sensor for flight height measurement, etc. Moreover, each UAV includes a Wi-Fi receiver and Wi-Fi transmitter [25] to simulate wireless communication between UAVs. Users can also add additional sensors as needed.

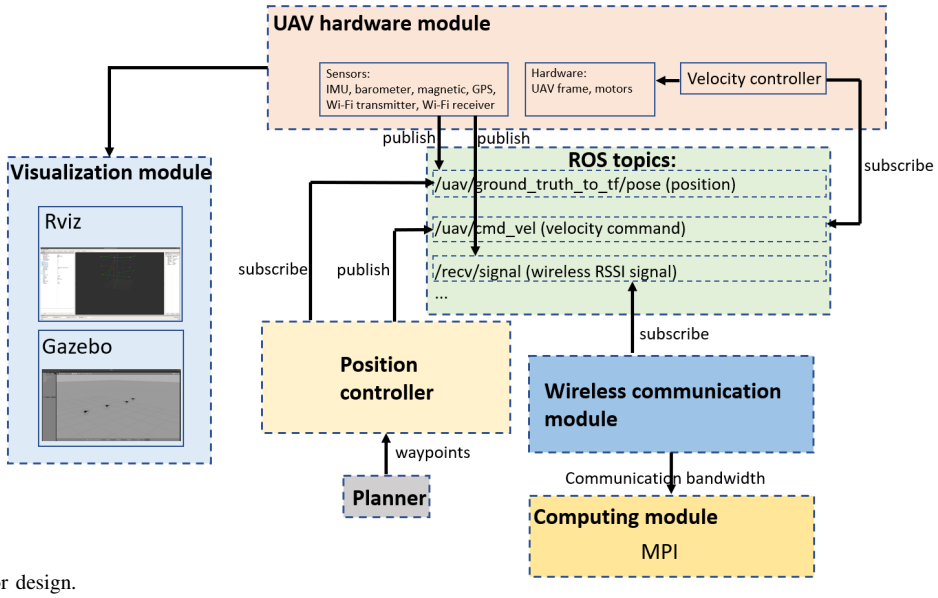


Fig. 1. NAC simulator design.

B. Controller Module

The controller module controls UAV's movement to follow the planned waypoints. This is achieved by using two controllers: velocity controller and position controller. Particularly, the position controller calculates the desired velocities based on the planned waypoints and inputs the desired velocities into the velocity controller. The velocity controller, provided by ROS plugin, then generates control commands to drive the UAV to reach the desired velocities. The calculation of the desired velocities is performed using the following equations:

$$\begin{aligned} \hat{v}_x &= k_x(x - x_g) \\ \hat{v}_y &= k_y(y - y_g) \\ \hat{v}_z &= k_z(z - z_g) \end{aligned} \tag{1}$$

where $\hat{v}_x, \hat{v}_y, \hat{v}_z$ are the desired velocities in three directions. k_x, k_y, k_z are the gains of the position controller. (x, y, z) and (x_g, y_g, z_g) are UAV's current position and the desired position, respectively.

C. Visualization Module

The visualization module allows users to monitor the status of UAVs during NAC simulations in real time. This is achieved using Gazebo [27], which provides a realistic simulation environment and an interactive graphic interface. Moreover, by using Rviz [32], the data captured by UAV's sensors, such as trajectories and images captured by the on-board camera, can be displayed. Our simulator provides the real-time visualization of waypoints and UAV trajectories.

D. Wireless Communication Module

The communication module simulates the wireless communication between UAVs, each equipped with a Wi-Fi transmitter and receiver. The received signal strength (RSS) at the Wi-Fi receiver, denoted by $S(dBm)$, is calculated using the well-known Hata-Okumura model [33], [34] by the following equation:

$$S = S_t + P_L, \tag{2}$$

where $S_t(dBm)$ is the transmission power and $P_L(dB)$ is path loss given by

$$P_L = D + E\log_{10}(d) + F. \tag{3}$$

In the above equation, $d(km)$ is the distance between two UAVs. The values of D, E, and F depend on the transmission frequency, antenna heights, and environment types. In particular, D and E are represented by

$$\begin{aligned} D &= 69.55 + 26.16log_{10}(f_c) - 13.82log_{10}(h_b) - a(h_m) \\ E &= 44.9 - 6.55log_{10}(h_b) \end{aligned} \tag{4}$$

where f_c is the frequency of transmission in MHz (valid range: $150MHz - 1500MHz$), h_b is the effective height of the transmitter in meters (valid range: $30m - 200m$), h_m is the effective height of the receiver (valid range: $1m - 10m$), and $a(h_m)$ is the mobile antenna height correction factor, which is a function that depends on the environment. F is a factor to correct formulas for open rural and suburban areas. Given RSS S, the maximum data rate, denoted by C (bps), can then be calculated according to Shannon's Theory:

$$C = W\log_2(1 + \frac{10^{\frac{(S-30)}{10}}}{N_0}), \tag{5}$$

where W (Hz) is the communication bandwidth and N_0 (Watts) is the noise power.

E. Computing Module

The computing module simulates the distributed computing process using Message Passing Interface (MPI) [35], which provides system interfaces for users to program parallel computing applications and supports various programming languages. In our simulator, each process represents a UAV that runs computation tasks in parallel and

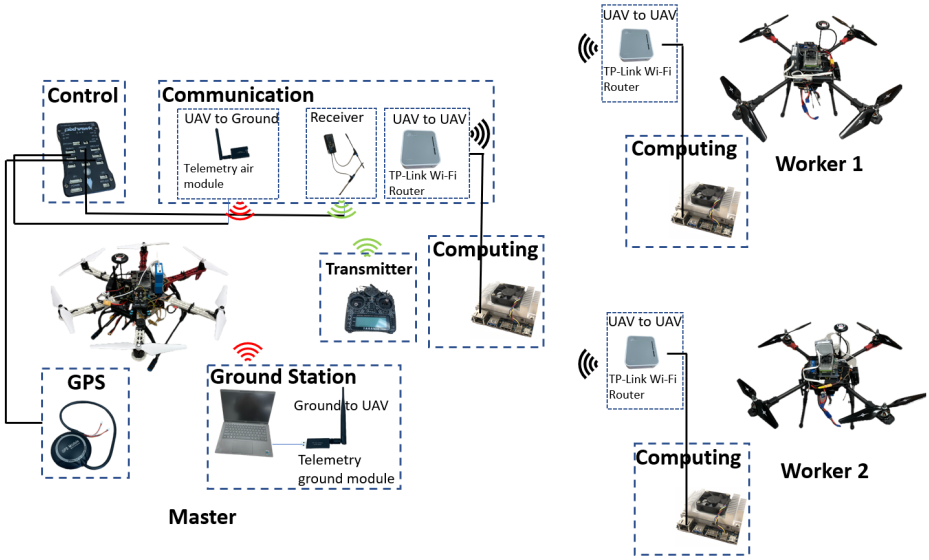


Fig. 2. NAC hardware testbed design.

communicates with each other via MPI. Each process uses the states of UAV to calculate the data rate using (5). We calculate the delay as $\frac{I}{C}$ (s), where $I(bits)$ is the size of data being transmitted. This delay is introduced manually using the $time.sleep()$ function in Python before sending data via MPI.

F. ROS Topics

In our simulator, communication between modules is achieved through ROS topics as shown in Fig. 1. Each module can either subscribe to a topic to receive messages or publish messages to a topic. For example, GPS publishes UAV's position to the topic $/uav/ground\_truth\_to\_tf/pose$. The position controller subscribes to this topic to receive the UAV's current position. After calculating the desired velocities, the position controller then publishes these values to the topic $/uav/cmd\_vel$, which is subscribed by the UAV's velocity controller. Moreover, the Wi-Fi receiver publishes the RSS signals to the topic $/uav/recv\_signal$, which is subscribed by the communication module to calculate the data rate.

IV. Hardware Testbed Design

In this section, we present the design of a NAC hardware testbed, consisting of three UAVs, two constructed using Tarot 650 Quadrotor frames and one using a DJI F550 frame (see Fig. 2). The Tarot quadrotors serve as workers, while the F550 serves as the master. In the following, we describe the hardware design from four aspects: computing, communication, flight control, and power management. For each aspect, we detail its functionalities and the hardware devices used to achieve them.

A. Computing

The computing unit on each UAV is a NVIDIA Jetson TX2 [36], which boasts a powerful 6-core CPU and 256-core NVIDIA Pascal GPU, and has 8GB memory. Despite its compact size (50mm×87mm) and lightweight (85g), Jetson TX2 requires a development board to operate and the original one from NVIDIA is too large (17cm×17cm) to be placed on the UAV. To overcome this, we utilize a compact carrier board developed in our previous works [12]. To enable computing resource sharing among UAVs, we connect Jetson TX2 to the same Wi-Fi network as described in the following subsection. This allows UAVs to perform collaborative computing using MPI.

B. Communication

As illustrated in Fig. 2, there are three types of communications in NAC including UAV-to-UAV, UAV-to-Ground/Ground-to-UAV, and Transmitter-to-Receiver. The features and enabling techniques for each communication type are described as follows.

1) UAV-to-UAV Communication: The UAV-to-UAV communication is used to exchange data between UAVs, such as data needed for computation and UAV's state information. Our testbed achieves this through a TP-Link Wi-Fi router network. In particular, one TP-Link Wi-Fi router is set up as an access point on the master while the remaining TP-Link Wi-Fi routers on the workers are configured as clients connecting to the master. This allows the master to communicate with both workers simultaneously. Each router is connected to the Jetson TX2 via an Ethernet port.

2) UAV-to-Ground/Ground-to-UAV Communication: The UAV-to-Ground/Ground-to-UAV communication is used to monitor the states of UAV such as its position, rotation, battery status, etc. It is enabled by the FPVDrone 500MW Radio Telemetry Kit 915 Mhz Air and the Ground data transmit module. The air data transmit module is connected to the Pixhawk telemetry port and the ground module is connected to a laptop USB port. In our design, we use the QGroundControl software [37] installed on the ground station, which is a laptop, to interact with the UAV such as commanding it to take off or land. We can also create

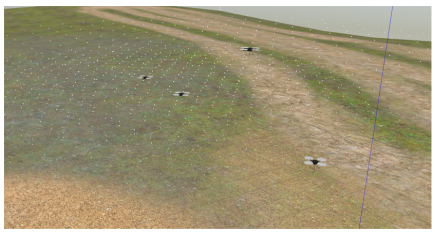


Fig. 3. Visualization of the simulation environment with Gazebo.

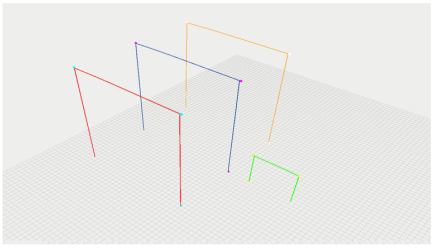


Fig. 4. Visualization of the UAVs' pre-planned waypoints and paths with Rviz.

different missions such as waypoint following and upload them to the flight controller.

3) Transmitter-to-Receiver Communication: The transmitter-to-receiver communication is used to control the UAV. The transmitter has joysticks that allow manual control of throttle, yaw, pitch, and roll angles. The UAV's flight modes can also be changed by adjusting the switches, such as the Manual control mode, Mission mode for completing pre-programmed missions, and Hold mode for maintaining altitude.

C. Flight Control

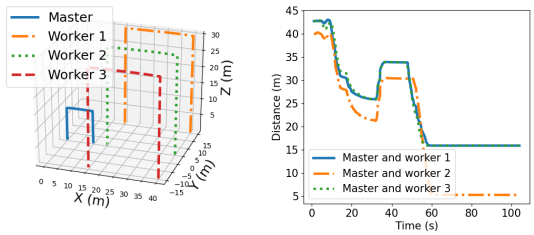
Each UAV is equipped with a Pixhawk flight controller, which is capable of controlling the UAV based on manual inputs or following planned waypoints with the aid of GPS signals. Both the receiver and the telemetry air module are connected to the Pixhawk.

D. Power Management

Each UAV is equipped with two 4S Lipo batteries having 14.8 Voltage to provide power to all its subsystems. The first battery is designated to power the computing and communication subsystems while the second battery is used to power the flight control and propulsion subsystems of the UAV. The flight time of UAV with the battery fully charged is around 15 minutes and the operation time of the computing and communication unit with the battery fully charged is around 45 minutes.

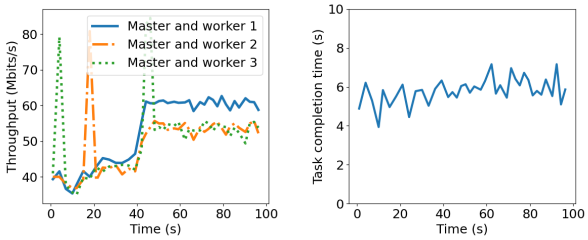
V. Simulation Studies

In this section, we simulate NAC using the designed ROS- and Gazebo-based simulator with UAVs collaboratively completing the matrix multiplication task. We examine the impact of UAV mobility by comparing results from two scenarios: the *moving scenario* where UAVs move continuously during task execution and the *static scenario* where the UAVs remain stationary during task execution. The detailed simulation configurations and results are presented as follows.



(a) (b)

Fig. 5. a) Trajectories of the four UAVs and b) distances between the master UAV and worker UAVs in the simulation.



(a) (b)

Fig. 6. a) Throughput between the master and worker UAVs and b) completion time for a single iteration of matrix multiplication in the moving scenario.

A. Simulation Configurations

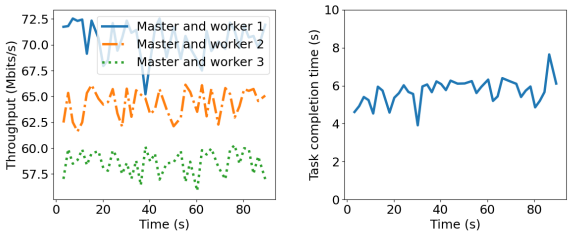
We simulate a NAC system with four UAVs. The computation task considered is matrix multiplication as described in Sec. II. The matrices A and B are randomly generated and have a dimension of 100×10000 and 10000×1, respectively. This matrix multiplication task is conducted repeatedly for 40 iterations. The parameters of the position controller described in (1) are set to $k_x = 0.15, k_y = 0.15, k_z = 1$. The parameters of the wireless communication module as described in (2), (4) and (5) are set to $S_t = 1200MHz, f_c = 1200MHz, W = 1200MHz, N_0 = 1.1 \times 10^{-35}$ watts. In the moving scenario, UAVs are controlled to follow pre-planned waypoints (see Figs. 3-4 as an illustration). In the static scenario, UAVs are placed on the ground.

B. Simulation Results

The simulation results, including the throughput and computation time, for both the moving and static scenarios, are presented as follows.

1) Moving Scenario: The trajectories of the four UAVs in the moving scenario are shown in Fig. 5(a). Each UAV takes off from the ground, flies straightly at a constant altitude, and finally lands. The master UAV flies a shorter distance than the worker UAVs to vary the distances between them. Fig. 5(b) shows the distances between the master and worker UAVs. As we can see, the distances between the master and workers decrease during the first 35s, remain stable for 15s, and then decrease again until landing at around 60s.

As shown in Fig. 6(a), the varying distances between the master and workers result in changes in the throughputs. The throughputs increase over time as the distances decrease. It is also noted that there are significant variations in the throughputs due to UAVs' mobility. Fig. 6(b) shows the completion time for a single iteration of matrix multiplication, which has an average value of around 5s.



(a) (b)

Fig. 7. a) Throughput between the master and workers and b) completion time for a single iteration of matrix multiplication in the static scenario.

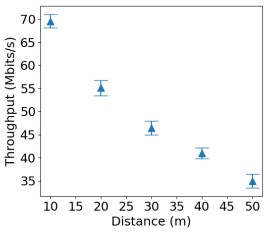


Fig. 8. Throughput between two UAVs at various distances in simulations.

2) Static Scenario: In the static scenario, the distances between the master and worker UAVs are set to 9.43m, 12.80m, and 17m. Fig. 7(a) depicts the throughputs between UAVs. Compared to Fig. 6(a), we can see that the throughputs between the static UAVs have smaller variances, resulting in more stable communication between the UAVs. The time for completing a single iteration of matrix multiplication in the static scenario is shown in Fig. 7(b).

To further understand the relationship between distance and communication performance, we vary the distance between two UAVs and measure the throughput. The results, which include the mean values and standard deviations of 100 data samples, are shown in Fig. 8. As we can see, the throughput decreases quickly as the distance between the UAVs increases. This graph demonstrates the crucial impact of UAV mobility in NAC.

VI. Real Flight Tests

In this section, we conduct real flight tests at San Diego State University (SDSU) sport field as shown in Fig. 9 to evaluate the NAC hardware testbed for performing linear regression. The experiments include both moving and static scenarios to assess the impact of UAV mobility on the system performance. The configurations and results of the experiments are described in detail below.

A. Experiment Configurations

In the flight tests, each UAV in the moving scenario follows pre-planned waypoints with a flying speed of 0.67 m/s and a constant altitude of 3.04m. In the static scenario, all UAVs remain stationary on the ground. The computation task performed by the UAVs in both scenarios is to train a linear regressor for forest fire detection as described in Sec. II. The throughput between two UAVs is measured using $Iperf3$ [38].

Fig. 9. Flight test of NAC hardware testbed with three UAVs at the San Diego State University (SDSU) sport field

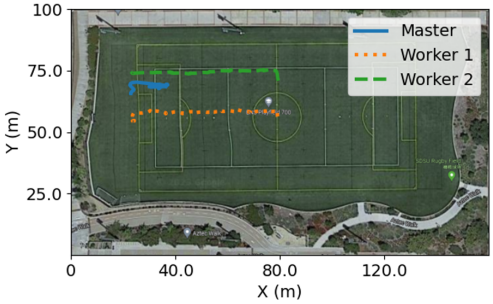


Fig. 10. Trajectories of the three UAVs.

B. Experiment Results

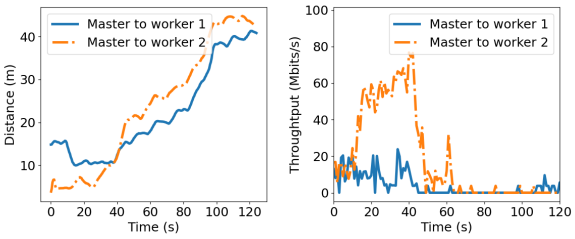
1) Moving Scenario: We extract the positions of UAVs from the flight logs and calculate the distance between the master UAV and the two worker UAVs. The trajectories of the UAVs are displayed in Fig. 10. The two workers follow straight lines in parallel, while the master flies a short distance in between them and then hovers until the completion of the task.

The distances between master UAV and worker UAVs are shown in Fig. 11(a). The corresponding throughputs between UAVs are depicted in Fig. 11(b). It can be observed that within the first $40s$, the throughput between master and worker 2 increases even though the distance between them remains relatively stable. This variation in the throughput may be attributed to the impact of UAV mobility and wireless communication interference. Moreover, the throughput between master and worker 2 is much larger than that between master and worker 1 during the first 40s, due to the shorter distance between them. After this point, both throughputs decrease drastically as the distances continuously increase and approach zero when the distances reach around 20m. The training terminates after $75s$ due to communication loss caused by increasing distances.

Fig. 12. presents the impact of distance on communication performance. As expected, the throughput decreases as the distance between the UAVs increases. The maximum communication distance between two UAVs is around 25m as indicated in the figure.

2) Static Scenario: To further evaluate the impact of mobility on the communication and computation performance, we keep all UAVs stationary on the ground and re-run the forest fire detection application. This allows us to compare and contrast the results with those obtained from the moving scenario.

In this scenario, we set the distances between the master



(a) (b)

Fig. 11. a) Distance and b) throughput between the master UAV and worker UAVs in the moving scenario.

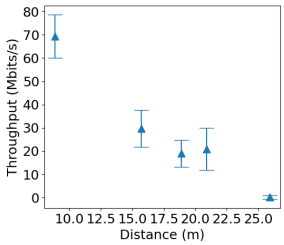


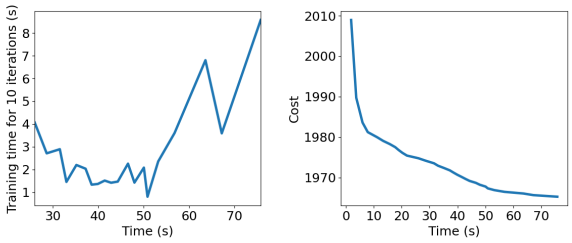
Fig. 12. Throughput between two UAVs at various distances in flight tests.

and worker 1 and 2 to 9.8m and 8.9m, respectively. The throughputs between the UAVs are shown in Fig. 14(a). Compared to the throughputs when UAVs are moving as shown in Fig. 11(b), we can see that, during the first 40s, the throughput between the master and worker 2 is lower in the static case due to the larger distance. However, the throughput variance is smaller when the UAVs are static.

We further analyze the computation performance by showing the training time for 10 iterations and the training cost in Fig. 14(b) and Fig. 14(c), respectively. The results indicate that the task completion time is generally smaller in the static case compared to the case when UAVs are moving due to shorter inter-vehicle distances as shown in Fig. 13(a). Moreover, the training completes 640 iterations within $200s$ when UAVs are static, while only 310 iterations are completed in the moving case within $75s$. This is due to the fact that, in the moving scenario, the training terminates when workers move out of the master's communication range, which occurs at 75s. As more training iterations are performed, the training cost is also smaller in the static case as shown in Fig. 13(b) and Fig. 14(c).

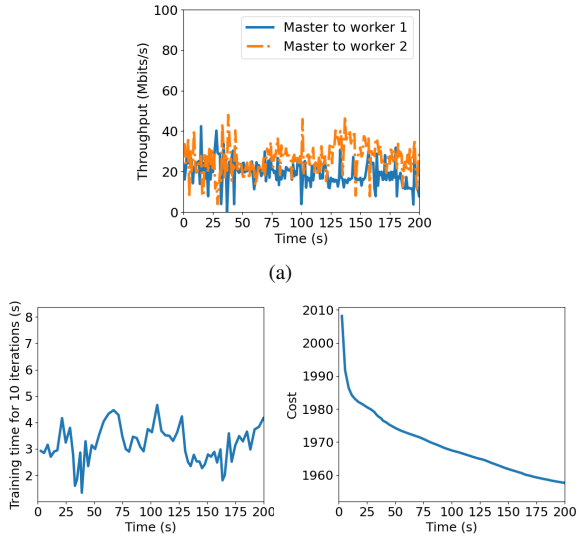
VII. Discussions and Conclusions

The simulation and flight tests reveal challenges in achieving high-performance NAC. In particular, UAV mobility results in significant communication variance, which can lead to communication bottlenecks that affect robust airborne computing. The flight tests also demonstrate the limitations of standard Wi-Fi routers with omni-directional antennas. They have limited communication range and do not provide sufficient communication throughput for real-time applications. To address these limitations, our future work will focus on improving the UAV-to-UAV communication range and throughput through the use of phased array antennas and mmWave 5G technology [23]. Additionally, we will implement coding techniques [10] to improve the robustness of NAC to uncertain system disturbances and apply



(a) (b)

Fig. 13. a) Time for completing 10 training iterations and b) the associated training cost in the moving scenario.



(a)

(b) (c)

Fig. 14. a) Throughput between the master UAV and worker UAVs in the static scenario; b) Time for completing 10 training iterations and c) the associated training cost in the static scenario.

the computation load allocation framework we previously developed [11] to optimize computation resource allocation.

In conclusion, this paper introduces a realistic simulator and hardware testbed for NAC research. The simulator, built using ROS and Gazebo, emulates networked UAVs with inter-vehicle resource sharing and distributed computing capabilities and can simulate various realistic environments. The hardware testbed consists of three quadrotors each with a Pixhawk control unit, Jetson TX2 computing unit, and telemetry radio and TP-Link Wi-Fi router for communication. The simulation and real flight tests on two computation applications showed that UAV mobility has a direct impact on the throughput, resulting in degraded and unstable communication performance as distance increases.

Acknowledgement

We would like to thank National Science Foundation under Grants CAREER-2048266 and CAREER-1714519 for the support of this work.

References

[1] H. Chang, Y. Chen, B. Zhang, and D. Doermann, "Multi-uav mobile edge computing and path planning platform based on reinforcement learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.

[2] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek, "Autonomous uav surveillance in complex urban environments," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2. IEEE, 2009, pp. 82–85.

[3] J. Scherer, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan, V. Vukadinovic, C. Bettstetter, H. Hellwagner, and B. Rinner, "An autonomous multi-uav system for search and rescue," in *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, 2015, pp. 33–38.

[4] J. Everaerts *et al.*, "The use of unmanned aerial vehicles (uavs) for remote sensing and mapping," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, no. 2008, pp. 1187–1192, 2008.

[5] M. Elloumi, R. Dhaou, B. Escrig, H. Idoudi, and L. A. Saidane, "Monitoring road traffic with a uav-based system," in *2018 IEEE wireless communications and networking conference (WCNC)*. IEEE, 2018, pp. 1–6.

[6] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud, "Uavfog: A uav-based fog computing for internet of things," in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2017, pp. 1–8.

[7] S. Jeong, O. Simeone, and J. Kang, "Mobile cloud computing with a uav-mounted cloudlet: optimal bit allocation for communication and computation," *Iet Communications*, vol. 11, no. 7, pp. 969–974, 2017.

[8] K. Lu, J. Xie, Y. Wan, and S. Fu, "Toward uav-based airborne computing," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 172–179, 2019.

[9] B. Wang, J. Xie, K. Lu, Y. Wan, and S. Fu, "Coding for heterogeneous uav-based networked airborne computing," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[10] ——, "On batch-processing based coded computing for heterogeneous distributed computing systems," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, pp. 2438–2454, 2021.

[11] ——, "Learning and batch-processing based coded computation with mobility awareness for networked airborne computing," *IEEE Transactions on Vehicular Technology*, 2022.

[12] B. Wang, J. Xie, S. Li, Y. Wan, Y. Gu, S. Fu, and K. Lu, "Computing in the air: An open airborne computing platform," *IET Communications*, vol. 14, no. 15, pp. 2410–2419, 2020.

[13] A. Asheralieva and D. Niyato, "Hierarchical game-theoretic and reinforcement learning framework for computational offloading in uav-enabled mobile edge computing networks with multiple service providers," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8753–8769, 2019.

[14] H. Wang, H. Ke, and W. Sun, "Unmanned-aerial-vehicle-assisted computation offloading for mobile edge computing based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 180 784–180 798, 2020.

[15] H. Zhou, Z. Wang, G. Min, and H. Zhang, "Uav-aided computation offloading in mobile edge computing networks: a stackelberg game approach," *IEEE Internet of Things Journal*, 2022.

[16] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 73–84, 2020.

[17] A. Douklias, L. Karagiannidis, F. Misichroni, and A. Amditis, "Design and implementation of a uav-based airborne computing platform for computer vision and machine learning applications," *Sensors*, vol. 22, no. 5, p. 2049, 2022.

[18] J. Diller, P. Hall, C. Schanker, K. Ung, P. Belous, P. Russell, and Q. Han, "Iccswarm: A framework for integrated communication and control in uav swarms," in *Proceedings of the Eighth Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, 2022, pp. 1–6.

[19] E. Pereira, K. Hedrick, and R. Sengupta, "The c3uv testbed for collaborative control and information acquisition using uavs," in *2013 American Control Conference*. IEEE, 2013, pp. 1466–1471.

[20] M. Schmittle, A. Lukina, L. Vacek, J. Das, C. P. Buskirk, S. Rees, J. Sztipanovits, R. Grosu, and V. Kumar, "Openuav: A uav testbed for the cps and robotics community," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 130–139.

[21] M. Moradi, K. Sundaresan, E. Chai, S. Rangarajan, and Z. M. Mao, "Skycore: Moving core to the edge for untethered and reliable uav-based lte networks," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 35–49.

[22] S. Li, C. He, M. Liu, Y. Wan, Y. Gu, J. Xie, S. Fu, and K. Lu, "Design and implementation of aerial communication using directional antennas: learning control in unknown communication environments," *IET Control Theory & Applications*, vol. 13, no. 17, pp. 2906–2916, 2019.

[23] R. K. Sheshadri, E. Chai, K. Sundaresan, and S. Rangarajan, "Skyhaul: An autonomous gigabit network fabric in the sky," *arXiv preprint arXiv:2006.11307*, 2020.

[24] A. Y. Javaid, W. Sun, and M. Alam, "Uavsim: A simulation testbed for unmanned aerial vehicle network cyber security analysis," in *2013 ieee globecom workshops (gc wkshps)*. IEEE, 2013, pp. 1432–1436.

[25] S. Moon, J. J. Bird, S. Borenstein, and E. W. Frew, "A gazebo/ros-based communication-realistic simulator for networked suas," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1819–1827.

[26] ROS, "Robot operating system (ros)," https://www.ros.org/, 2023, accessed: 2023-01-14.

[27] GAZEBO, "Gazebo," https://gazebosim.org/home, 2023, accessed: 2023-01-14.

[28] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[29] C. Wu and J. Z. Yu, "Evaluation of linear regression techniques for atmospheric applications: the importance of appropriate weighting," *Atmospheric Measurement Techniques*, vol. 11, no. 2, pp. 1233–1250, 2018.

[30] U. M. L. Repository, "Forest fires data set," https://archive.ics.uci.edu/ml/datasets/forest+fires, 2023, accessed: 2023-01-14.

[31] ROS, "Ros topic," http://wiki.ros.org/Topics, 2023, accessed: 2023-01-14.

[32] Rviz, "Rviz," http://wiki.ros.org/rviz, 2023, accessed: 2023-01-14.

[33] M. Hata, "Empirical formula for propagation loss in land mobile radio services," *IEEE transactions on Vehicular Technology*, vol. 29, no. 3, pp. 317–325, 1980.

[34] Y. Okumura, "Field strength and its variability in vhf and uhf land-mobile radio service," *Rev. Electr. Commun. Lab.*, vol. 16, pp. 825–873, 1968.

[35] "MPI," Accessed: Feb. 04, 2023. [Online]. Available: https://ds.cs.luc.edu/mpi/mpi.html

[36] "Jetson TX2," Accessed: Feb. 04, 2023. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/

[37] "Q Ground Control," Accessed: Feb. 04, 2023. [Online]. Available: http://qgroundcontrol.com/

[38] "Iperf 3," Accessed: Feb. 04, 2023. [Online]. Available: https://iperf.fr/iperf-download.php