Unmanned Systems, Vol. 0, No. 0 (2013) 1–15

Exploring Networked Airborne Computing: A Comprehensive Approach with Advanced Simulator and Hardware Testbed

Haomeng Zhang[a,b,*], Baoqian Wang[a,b,†], Ruitao Wu[a,‡], Junfei Xie[a,§], Yan Wan[c,¶], Shengli Fu[d,‖], Kejie Lu[e,**]

[a] *San Diego State University, San Diego, CA, 92182*
[b] *University of California, San Diego, La Jolla, CA, 92093*
[c] *University of Texas at Arlington, Arlington, TX 76019*
[d] *University of North Texas, Denton, TX 76201*
[e] *University of Puerto Rico at Mayagüez, Mayagüez, Puerto Rico 00681*

In recent years, networked airborne computing (NAC) has emerged as a promising paradigm because it can leverage the collaborative capabilities of unmanned aerial vehicles (UAVs) for distributed computing tasks. Despite the burgeoning interests in NAC and UAV-based computing, many existing studies depend on over-simplified simulations for performance evaluation. This reliance has led to a gap in our understanding of NAC's true potential and challenges. To fill this gap, this paper presents a comprehensive approach: the creation of a realistic simulator and a novel hardware testbed. The simulator, developed using ROS and Gazebo, emulates networked UAVs, focusing on resource-sharing and distributed computing capabilities. This tool offers a cost-effective, scalable, and adaptable environment, making it ideal for preliminary investigations across a myriad of real-world scenarios. In parallel, our hardware testbed comprises multiple quadrotors, each equipped with a Pixhawk control unit, a Raspberry Pi computing module, a real-time kinematic (RTK) positioning system, and multiple communication units. Through extensive simulations and hardware tests, we delve into the key determinants of NAC performance, such as computation task size, number of UAVs, communication quality, and UAV mobility. Our findings not only underscore the inherent challenges in optimizing NAC performance but also provide pivotal insights for future enhancements. These insights encompass refining the simulator, reducing computation overheads, and equipping the hardware testbed with cutting-edge communication devices.

Keywords: Unmanned Aerial Vehicles; Networked Airborne Computing; Simulator Design; Testbed Development.

1. Introduction

Unmanned aerial vehicles (UAVs), when equipped with onboard computing capabilities, can transcend their traditional roles, acting as aerial servers to cater to ground users.[1] Such advanced computing not only augments core UAV functionalities like precise positioning and efficient image processing but also paves the way for a myriad of applications. These encompass surveillance,[2] search and rescue operations,[3] remote sensing,[4] and traffic monitoring.[5] Given these compelling benefits, there has been a burgeoning interest among researchers to seamlessly integrate UAVs into contemporary computing paradigms such as multi-access edge computing (MEC), fog computing, and mobile cloud computing.[1,6,7]

While small UAVs are known for their agility and adaptability, their constrained payload capacity often limits their onboard computing capabilities. To address this, our team introduced the concept of networked airborne computing (NAC).[8] NAC is designed to harness and amplify airborne computing resources by promoting inter-vehicle resource sharing through direct flight-to-flight communi-

*Ph.D. student, Department of Electrical and Computer Engineering. Email:hzhang3986@sdsu.edu.
†Ph.D. candidate, Department of Electrical and Computer Engineering. Email: bwang4848@sdsu.edu. The first two authors contribute equally.
‡M.S. student, Department of Computer Science. Email: rwu9937@sdsu.edu.
§Associate professor, Department of Electrical and Computer Engineering. Email: jxie4@sdsu.edu. Corresponding Author.
¶Distinguished Professor, Department of Electrical Engineering. Email: yan.wan@uta.edu.
‖Professor, Department of Electrical Engineering. Email: shengli.fu@unt.edu.
**Professor, Department of Computer Science and Engineering, Email: kejie.lu@upr.edu.

cation links. Leading the charge in this domain, our research[9,10] unveiled a novel distributed computing framework for static UAV networks. This framework integrates resource allocation with advanced coding techniques, ensuring a robust and efficient airborne computing experience. Recognizing the dynamic nature of UAV operations, we further extended our research to accommodate UAVs in motion, developing a reinforcement learning-based algorithm[11] that jointly optimizes computation load allocation and UAV trajectories. To bridge the gap between theory and practice, we also pioneered a dedicated hardware platform,[12] which is equipped with the powerful NVIDIA Jetson TX2 and supports high-bandwidth wireless communication, laying a solid foundation for future UAV network research. While this platform has been instrumental in advancing UAV computing, it was primarily designed with a focus on individual UAV operations, 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.

Shifting our focus to contributions from other research teams, a plethora of studies on UAV-based computing has been conducted in the last few years.[1,13–16] Despite the importance of these studies, we notice that most existing studies rely on mathematical model-based simulations for performance evaluation. These simulations, though economical and convenient, encapsulate UAV dynamics, communication, and computational behaviors within rudimentary mathematical constructs. Such abstractions, despite their appeal, might not always resonate with the intricate behaviors observed in real-world UAV deployments. Moreover, when venturing into the realm of testing real UAVs, the breadth of these studies seems somewhat constrained, often tethered to single or dual UAV configurations. For instance, the study presented in[17] explored the potential of airborne computing in the domains of computer vision and machine learning, in which the authors developed a UAV-based MEC system with a pair of UAVs to analyze communication and computational delays during inter-UAV video streaming.

While there remains a noticeable gap in the availability of realistic testbeds tailored for NAC research, the UAV research community has made significant strides in developing testbeds aimed at UAV control, communication, and networking.[18–24] For instance, Diller *et al.*[18] devised a testbed specifically to probe into the intricacies of communication and control within UAV swarms. Pereira *et al.*[19] took a different route, unveiling a UAV hardware testbed centered around control and information acquisition. Schmittle *et al.*,[20] catering to robotics applications like collision avoidance, introduced a cloud-based UAV simulator leveraging Docker. For UAV communication, Moradi *et al.*[21] showcased an evolved packet core (EPC) tailored to be mounted directly onto tethered UAVs, facilitating the establishment of UAV-based long-term evolution (LTE) networks. Their two-UAV LTE network experiment yielded promising results, notably enhancing client connectivity during handoff. Moon *et al.*[25] developed a Robot Operating System (ROS) and Gazebo-based simulator to emulate wireless communication dynamics among networked UAVs. Li *et al.*[22] ventured into the realm of UAV-to-UAV (U2U) communication, pioneering a directional antenna-based system capable of self-adjusting its antenna orientations to optimize communication in unpredictable environments. Sheshadri *et al.*[23] put forth a novel UAV network design harnessing mmWave technology, aiming for Gigabit-speed communication, and validated their design using a hardware testbed.

In this paper, we aim to bridge the existing research gap concerning the lack of realistic testbeds tailored for NAC research. Our main contributions are summarized as follows:

(1) *Realistic NAC Simulator Leveraging ROS[26] and Gazebo.*[27] Distinct from existing ROS and Gazebo-based UAV simulators, our simulator seamlessly integrates the Message Passing Interface (MPI) to emulate distributed computing across UAV computing clusters. This simulator paves the way for cost-efficient preliminary NAC investigations and offers a platform to assess scenarios that pose challenges for hardware testbed configurations, especially those spanning large-scale environments.
(2) *Hardware Testbed Fortified with Networked Airborne Computing Prowess.* This testbed comprises multiple quadrotors, each equipped with computing, communication, and inter-vehicle resource-sharing functionalities. It serves as a conduit for precise evaluations of NAC performance under real flight conditions.
(3) *In-depth Analysis of Determinants Influencing NAC Performance.* A plethora of experiments, executed using the aforementioned simulator and hardware testbed, delve into the pivotal determinants of NAC performance. These encompass computation task size, number of UAVs, U2U communication, and UAV mobility. By implementing diverse computation tasks and distributed computing paradigms, we glean insights into their ramifications, thereby illuminating the challenges in optimizing NAC performance. These revelations also chart the course for future refinements in our NAC testbed designs.
(4) *Comparative Study of State-of-the-Art Communication Apparatuses.* Recognizing the indispensable role of U2U communication in NAC, we compare the performance of three communication devices, each with a unique communication capability. The devices under investigation comprise: 1) TP-Link TL-WR902AC[28] with an omnidirectional antenna, operating in the 2.4GHz band, 2) Ubiquity Loco M5[29] with a directional antenna, working in the 5GHz band, and 3) MikroTik wAP 60G[30] fortified with phased-array antennas, functioning in the 60GHz band. The outcomes of this comparison provide invaluable insights into the selection of communication techniques that are crucial to NAC

performance.

(5) *Real-World Application in Forest Fire Detection.* Our hardware testbed is utilized to actualize a forest fire detection mechanism using federated learning, underscoring its pragmatic utility in tangible scenarios.

The rest of the paper is organized as follows. Sec. 2 introduces the system model for UAV-based NAC. Sec. 3 and Sec. 4 describe the designs for the simulator and hardware testbed, respectively. The simulation and hardware test results are presented in Sec. 5 and Sec. 6, respectively. Sec. 7 then discusses the insights gained from the experiments. Finally, Sec. 8 concludes the paper and discusses future work.

2. System Model

In this section, we first describe how a computation task is completed collaboratively by UAVs in NAC. We then introduce the main computation tasks and schemes that will be evaluated in this study.

2.1. *Computing System*

Consider a NAC system with $N+1$ UAVs, where $N \in \mathbb{Z}^+$ 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 $\mathbf{X}$ 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 calculates their respective subtasks and sends 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.

Algorithm 1 NAC Computing Model

Master:

1: Broadcast $\mathbf{X}$ to workers.
2: **for** $i = 1 : N$ **do**
3: Listen to the channel and collect result $f_i(\mathbf{X})$ from worker i.
4: Calculate and output the final result using $\{f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_N(\mathbf{X})\}$.

Worker i:

5: Listen to the channel and receive $\mathbf{X}$ from the master.
6: Compute $f_i(\mathbf{X})$ and send results back to the master.

2.2. *Computing Schemes*

We primarily use matrix multiplication as the computation task for conducting comprehensive performance evaluations and analyzing the key impact factors in NAC. Matrix multiplication is chosen as it is straightforward to configure and serves as a fundamental building block for more complex computations, including those commonly found in machine learning applications.[31,32]

Consider a matrix multiplication task $\mathbf{AX}$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a pre-stored matrix and $\mathbf{X} \in \mathbb{R}^{n \times l}$ is the input. To compute the output $\mathbf{Y} = \mathbf{AX}$, different distributed computing schemes can be applied. In this study, we implement the following two fundamental distributed computing schemes.

(1) *Uncoded Distributed Computing:* In the traditional uncoded distributed computing scheme,[10] the matrix $\mathbf{A}$ is divided equally row-wise into N submatrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$, where each submatrix $\mathbf{A}_i$ is pre-stored in the corresponding worker i. Upon receiving the input $\mathbf{X}$, the master distributes $\mathbf{X}$ to each worker i, which then computes $\mathbf{A}_i\mathbf{X}$ and returns the results to the master. Finally, the master aggregates the received results to obtain the final output $\mathbf{Y} = [\mathbf{A}_1\mathbf{X}, \mathbf{A}_2\mathbf{X}, \ldots, \mathbf{A}_N\mathbf{X}]$.

(2) *Coded Distributed Computing:* The uncoded scheme described above may not perform well in mobile computing systems characterized by significant uncertain system disturbances, especially in NAC systems with high node mobility that can lead to frequent topology changes. To address this issue, various coded distributed computing schemes[10,33,34] have been developed to enhance system resilience and improve computational efficiency. The key idea is to introduce redundancies into the computation process by leveraging the principles of coding theory. To provide a comparison to the uncoded scheme, we also implement a classical coded scheme[10] in the simulations. In this scheme, prior to task execution, the matrix $\mathbf{A}$ is first encoded into a larger matrix $\hat{\mathbf{A}} = \mathbf{HA}$ with more rows, where $\mathbf{H} \in \mathbb{R}^{q \times m}$ is the encoding matrix with any m rows being full rank and $q > m$. This encoded matrix $\hat{\mathbf{A}}$ is then divided into N submatrices $\hat{\mathbf{A}}_1, \hat{\mathbf{A}}_2, \ldots, \hat{\mathbf{A}}_N$, with each submatrix $\hat{\mathbf{A}}_i$ pre-stored in a worker i. Upon receiving the input $\mathbf{X}$, each worker i multiplies $\hat{\mathbf{A}}_i$ with $\mathbf{X}$ and returns the computed result to the master. When the master receives a sufficient amount of results, i.e., when the number of received rows of inner product results exceeds the value of m, it proceeds to compute the final output by $\mathbf{Y} = \hat{H}_b^{-1}\hat{\mathbf{Y}}_b$, where $\hat{\mathbf{Y}}_b \in \mathbb{R}^{m \times l}$ denotes the aggregated results the master has received. $\hat{H}_b \in \mathbb{R}^{x \times m}$ is a submatrix of $\mathbf{H}$ corresponding to $\hat{\mathbf{Y}}_b$.

To showcase the practicality of the designed NAC testbed, we also implement and evaluate a real learning-enabled application using federated learning[35] in Sec. 6.3. Following the procedure outlined in Algorithm 1, in federated learning, the master broadcasts the parameters of the machine learning model to be trained (input $\mathbf{X}$) at the beginning of each iteration. Each worker then utilizes the received model parameters and its local data, which can

be considered a subset of the entire training dataset, to compute a model update (e.g., gradients). The model updates from all workers are sent to the master, which then aggregates these updates to calculate new model parameters. This procedure is repeated iteratively for a predefined number of iterations or until convergence.

3. 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 core modules: UAV hardware module, controller module, wireless communication module, computing module, and visualization module. These modules interact through ROS topics.[36] The functionalities of each module and their associated ROS topics are described as follows.

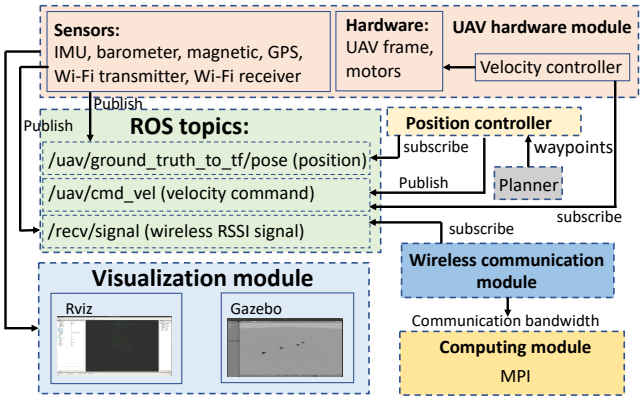


Fig. 1. NAC simulator design.

3.1. *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 its base size, weight, shape, and arm length, 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. In our simulator, we install the hector quadrotor UAV package[37] to emulate quadrotor UAV systems.

ROS offers plugins for various sensors that can be attached to the 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, a barometer sensor for flight height measurement, etc. Moreover, each UAV includes a Wi-Fi receiver and transmitter[25] to simulate wireless communication between two UAVs. Of course, users can also add additional sensors based on their needs.

3.2. *Controller Module*

The controller module governs the movement of the UAV. In our simulator, the UAVs are configured to follow preplanned waypoints. This is achieved by using two controllers: velocity controller and position controller. 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.

3.3. *Wireless Communication Module*

The communication module[37] simulates the wireless communication between two UAVs, each equipped with a Wi-Fi transmitter and receiver. The received signal strength (RSS) at the Wi-Fi receiver, represented as S (dBm), is determined using the well-known Hata-Okumura model.[38,39] This model estimates the RSS 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 (m) 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 defined as

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

where f_c is the transmission frequency 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 [40]:

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

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

3.4. *Computing Module*

The computing module simulates the distributed computing process using the Message Passing Interface (MPI),[41] which provides system interfaces for users to program parallel computing applications and supports various programming languages. In our simulator, each UAV runs as an individual thread assigned by MPI to execute computation tasks in parallel on a multi-core machine. Functions such as `send()`, `iSend()`, `recv()`, `iRecv()` are used to transmit data between the master and workers. Since the data transmission occurs instantaneously within the same machine, to simulate the data transmission delay between two UAVs, each thread obtains the states of the UAVs and calculates the data rate C using (5). The transmission delay is then calculated as $\frac{I}{C}$ (s), where I (bits) is the size of the data being transmitted. This delay is manually introduced using the `time.sleep()` function in Python before sending data via MPI. The `Barrier()` function is used to synchronize the process of all UAVs at the beginning of each matrix multiplication task.

3.5. *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,[42] the data captured by UAV's sensors, such as trajectories and images captured by the onboard camera, can be displayed. Our simulator provides the real-time visualization of waypoints and UAV trajectories.

3.6. *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.

4. 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). In the following, we describe the hardware design from five aspects: computing, communication, localization, flight control, and power management. For each aspect, we detail its functionalities and the hardware devices used to achieve them.

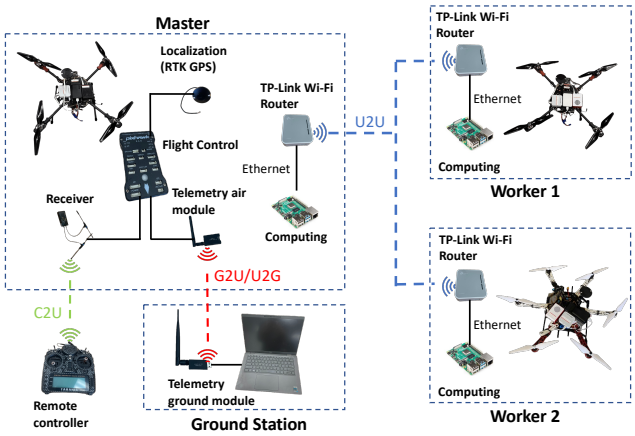


Fig. 2. NAC hardware testbed design.

4.1. *Computing*

The execution of distributed computing tasks is achieved by a Raspberry Pi 4 installed on each UAV, which contains a Quad-core Cortex-A72(ARM v8) processor running at 1.8GHz and 4GB LPDDR4-3200 SDRAM. It has a weight of approximately 46g and a dimension of $3.94 \times 2.76 \times 1.18$ inches, making it compact and lightweight for UAV applications. In terms of wireless connectivity, it supports both 2.4GHz and 5.0GHz IEEE 802.11B/g/n/ac standards. To enable computing resource sharing among UAVs, we connect Raspberry Pis to the same network as described in the following subsection and implement MPI on each device. This allows UAVs to perform collaborative computing using MPI.

4.2. *Communication*

As illustrated in Fig. 2, there are three types of communications involved in our testbed including UAV-to-UAV (U2U), Ground-to-UAV/UAV-to-Ground (G2U/U2G), and Controller-to-UAV (C2U) communications. The features and enabling techniques for each communication type are described as follows.

4.2.1. *U2U Communication*

The U2U 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 Wi-Fi router network using three TP-Link TL-WR902AC[28] routers. This router is equipped with an omnidirectional

antenna and can operate at both the 2.4GHz and 5GHz frequency bands. It offers three modes including access point (AP) mode, client mode, and range extender mode. To establish the NAC network, we set one router to AP mode and connect it to the Raspberry Pi on the master UAV via an Ethernet port. The remaining two routers are set to the client mode and connected to the worker UAVs. Moreover, all routers are configured to use the default WLAN service set identifier (SSID) of the AP, enabling point-to-multipoint communication between the master and the two workers. All routers operate at the 2.4GHz band. The WLAN IP is assigned randomly by Dynamic Host Configuration Protocol (DHCP) for all three UAVs.

In addition to the 2.4GHz omnidirectional antenna-based U2U communication, we also implement and evaluate another two advanced communication techniques:

(1) *5GHz directional antenna-based U2U communication*: We achieve this by using the Ubiquiti Nanostation Loco M5[29] that operates at the 5GHz band with a default antenna gain of 13dBi. It also offers three modes including AP mode, client mode, and wireless distribution system mode. For setting up the NAC network, we use three Loco M5 devices. One is connected to the master and operates in the AP mode. The others are connected to the workers and operate in the client mode. To establish point-to-multipoint connections, we configure all Loco M5 to use the same WLAN SSID. The channel width is set to 40MHz. Additionally, we assign a static IP for the WLAN.
(2) *60GHz phased-array antenna-based U2U communication*: This is achieved by using the MikroTik wAP 60G,[30] which contains 96 antenna elements and operates at the 60GHz band. It offers three modes including AP bridge mode (for point-to-point connections), bridge mode (for point-to-multipoint connections), and station-bridge mode (for client nodes). Similarly, we use three wAP 60G devices to set up the NAC network. One is connected to the master, operating in the AP bridge mode, while the other two are connected to the workers, operating in the station-bridge mode. All devices are configured to use the same WLAN SSID. The channel frequency is set to the default value 58.24GHz. The WLAN IP is randomly selected by DHCP for all three UAVs.

4.2.2. *G2U/U2G Communication*

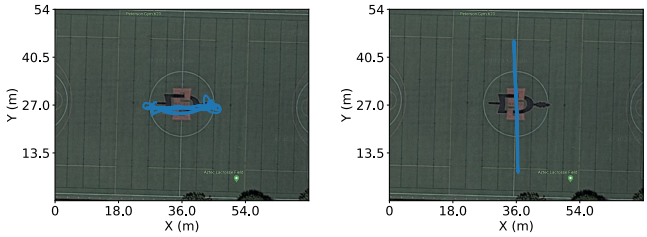
The G2U/U2G communication is used to monitor the state of UAV such as its position, rotation, battery status, etc. on the ground. It is enabled by the FPVDrone 500MW Radio Telemetry Kit 915 MHz Air and the Ground data transmit module, which operate at the 915MHz band. The air data transmit module is connected to the Pixhawk telemetry port and the ground module is connected to the ground station, which is a laptop. In our design, we use the QGroundControl software[43] installed on the ground station to interact with the UAV such as commanding it to perform different missions like waypoint following.

4.2.3. *C2U Communication*

This communication link is used for transmitting control signals from the remote controller to the UAV, which operates at the 2.4GHz band. The remote controller 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. In the flight tests presented in Sec. 6, the remote controller primarily serves to initiate (and terminate if needed) the waypoint following missions, where waypoints are generated using the QGound Control software and uploaded to the UAV prior to mission execution.

4.3. *Localization*

Each UAV is equipped with a Here3 Real-Time Kinematic (RTK) module[44] for localization, which has dimensions of 68mm×68mm×16mm, a weight of 48.8g, and an update rate of 8Hz. The complete Here3 RTK system comprises two main components: a stationary *reference station* positioned on the ground with a known coordinate and mobile *rovers* mounted on each UAV. The reference station continuously collects signals from multiple Global Navigation Satellite System (GNSS) satellites, determining signal errors by comparing actual satellite positions with those inferred from received signals. These errors are then transmitted as correction data to the rovers. Utilizing this correction data, the rovers can improve their position calculations derived from received satellite signals to achieve a centimeter-level accuracy. This is a significant improvement over the meter-level accuracy of conventional GPS modules.[45] Our flight tests (see Fig. 3) show that precise positioning significantly enhances the navigation performance of the UAVs during waypoint following missions.



(a) (b)

Fig. 3. UAV trajectories: (a) with conventional GPS vs. (b) with Here3 RTK.

4.4. *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 RTK signals. Both the receiver and the telemetry air module are connected to the Pixhawk.

4.5. *Power Management*

Each UAV is equipped with a 4S Lipo battery, which operates at 14.8V. This battery is specifically used to power the flight control and propulsion subsystems of the UAV. With the battery fully charged, the UAV has an approximate flight time of 15 minutes. To power the Raspberry Pi, we use a 5V 3A portable battery with a USB-C connector. To power the TP-Link Wi-Fi router, we utilize a separate portable battery with an output of 5V and 2.5A. This battery is connected to the router using a Micro USB wire.

5. Simulation Studies

In this section, we simulate NAC using the designed ROS and Gazebo-based simulator with UAVs collaboratively completing matrix multiplication tasks. We specifically focus on investigating the impact of four key factors on NAC performance, including task size, number of workers, U2U communication distance, and topology changes induced by UAV mobility. Both uncoded and coded schemes are implemented to gain a better understanding of the NAC system.

5.1. *Simulation Configurations*

Four simulations are designed to examine each impact factor. In all simulations, the parameters of the wireless communication module as described in (2), (4) and (5) are set to $S_t = 1200\text{MHz}$, $f_c = 1200\text{MHz}$, $W = 1200\text{MHz}$, $N_0 = 1.1 \times 10^{-35}$watts. The gains of the position controller in (1) are set to $k_x = 0.15, k_y = 0.15, k_z = 1$, which are selected to emulate the kinematic attributes of UAVs as configured in our hardware tests. The parameter q in the coded scheme is set to $q = 1.5m$ in all simulations except for Simulation Four, where it is configured as $q = 2m$ to gain higher resilience. In both coded and uncoded schemes, the workload is equally divided among the workers as they have the same computing power. Detailed configurations and the results of each simulation are presented in the subsequent subsections. In Fig. 4, we show an illustration of the simulation environment. All simulations are performed on a Precision 7865 Tower workstation, featuring the AMD Ryzen threadripper pro 5965wx CPU that contains 24 cores and 48 threads. The machine is equipped with 32GB of memory and runs on a 64-bits Ubuntu 20.04.5 LTS operating system.

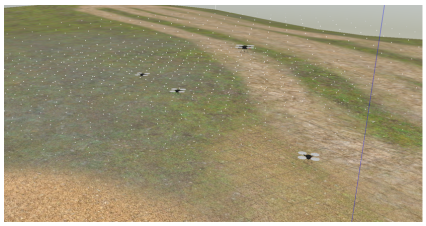
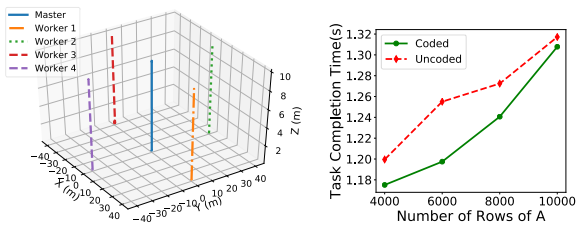


Fig. 4. Visualization of the simulation environment with Gazebo.

5.2. *Simulation One: Impact of Task Size*

In the first simulation, we investigate the impact of the task size on the NAC performance by varying the size of matrix $\mathbf{A}$. Specifically, we maintain the size of the input matrix $\mathbf{X}$ at 2000×1 and fix the number of columns of matrix $\mathbf{A}$ to 2000. The number of rows of matrix $\mathbf{A}$, i.e., m, is varied between 4000 and 10000. Five UAVs, including one master and four workers, are deployed which form a NAC system. The master UAV is surrounded by the workers, which are equally distanced and are all 40m away from the master. After taking off, all UAVs maintain a constant altitude of 10m and hover persistently throughout the entire task execution phase, as depicted in Fig. 5(a). We then let the NAC system repeatedly perform the matrix multiplication task 10 times (referred to as *iterations*). The average task completion time is recorded and shown in Fig. 5(b).



(a) (b)

Fig. 5. Results of Simulation One: (a) UAV trajectories and (b) average task completion time of different computing schemes at various task sizes.

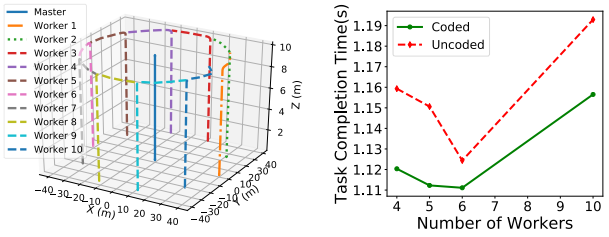
As anticipated, the task completion time grows as the task size increases. Notably, the coded scheme consistently outperforms the uncoded scheme. This is because in the coded scheme, the introduction of redundancies enhances the system's resilience to uncertainties such that network disturbances or computing node slowdowns have minimal impact on the overall system performance.

5.3. *Simulation Two: Impact of Number of Workers*

In the second simulation, we investigate the impact of the number of workers on the NAC performance. We employ the same configuration as before, with the workers evenly

positioned around the master and at a fixed distance of 40m from the master. All UAVs maintain a steady altitude of 10m after taking off. However, in contrast to the previous simulation, we let the workers move around the master, as illustrated in Fig. 6(a). To make the workers move at approximately the same speed, we interpolate the circle with a radius of 40m and with the master at the center to obtain 30 evenly spaced waypoints. By following these waypoints, each worker moves at an average speed of 0.36 m/s. For the computation task, we set the size of the matrix $\mathbf{A}$ and input $\mathbf{X}$ to 1000×2000 and 2000, respectively.

Fig. 6(b) displays the average task completion time as performed by the NAC system, measured over 10 iterations. The task completion times are presented for various numbers of workers involved in the computation. As we can see, as more workers share the workload, the task completion time decreases. However, it is worth noting that the performance degrades when the number of workers is large. This phenomenon occurs due to the additional time required for data transmission between the master and workers. As more workers are involved in sharing the workload, the data transmission time may exceed the computation time of the workers. Consequently, communication plays a major role in influencing the overall performance. As the number of workers, from which the master must wait for results, increases, any rise in transmission delay may lead to a degradation of the overall performance.

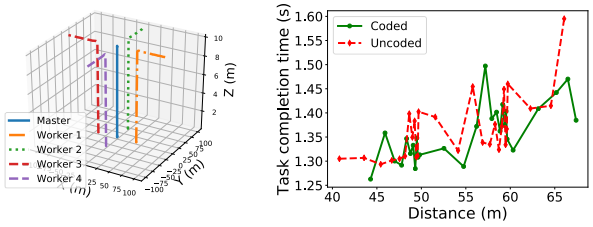


(a) (b)

Fig. 6. Results of Simulation Two: (a) UAV trajectories when $N = 10$ and (b) average task completion time of different computing schemes at various numbers of workers.

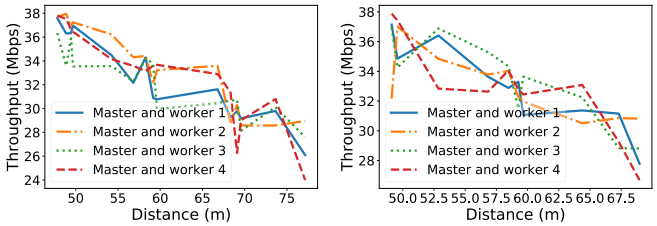
5.4. *Simulation Three: Impact of U2U Communication Distance*

In the third simulation, we investigate the impact of the U2U communication distance on the NAC performance. Similar to the first simulation, we construct a NAC system with five UAVs, comprised of one master and four workers. The workers are positioned evenly around the master. All UAVs fly at the same altitude of 10m during task execution. To study the impact of the U2U communication distance, we let all workers gradually move away from the master, as depicted in Fig. 7(a). The distance between each worker and the master UAV is initially set to 40m, and then incrementally increased to a final distance of 100m. While the workers are in motion, the NAC system is assigned to execute 30 matrix multiple tasks. Each task is configured with matrices $\mathbf{A} \in \mathbb{R}^{10000\times2000}$ and $\mathbf{X} \in \mathbb{R}^{2000\times1}$. Note that the NAC system is able to complete all assigned tasks before the workers reach their final positions.



(a) (b)

Fig. 7. Results of Simulation Three: (a) UAV trajectories, and (b) task completion of different computing schemes at various master-worker distances.



(a) (b)

Fig. 8. Results of Simulation Three: throughputs of different master-worker communication links for the (a) coded and (b) uncoded computing scheme.

Fig. 7 shows the time taken to complete each of the 30 matrix multiplication tasks versus the corresponding average master-worker distance at the moment of task completion. It can be observed that as the distance increases, the task completion time for both computing schemes tends to increase, indicating the degradation in the NAC performance. This is attributed to the reduction of the U2U throughputs, as depicted in Fig. 8. Comparing the two computing schemes (see Fig. 7), it is evident that the coded scheme exhibits greater resilience to throughput fluctuations and reductions compared to the uncoded scheme. This is mainly attributed to the nature of the coded scheme, where the master can utilize early-arriving results from high-throughput links for computing the final output. Consequently, fluctuations in throughput have a reduced impact on the overall performance. In contrast, the uncoded scheme relies on results from all workers, making it highly susceptible to any transmission delays.

5.5. *Simulation Four: Impact of Topology Change*

In a multi-UAV network, UAVs may leave or join the network during task execution, leading to alterations in the network's topology. To examine the impact of such network topology changes on NAC performance, we adopt the same configuration as Simulation Two with the number of workers set to $N = 10$. We then let some of the workers

leave the network by moving away from the master. Once the distance between a worker and the master exceeds 50m, it is considered out of the master's communication range, and its results can no longer be received by the master.

In Fig. 9(a), we illustrate the trajectories of the UAVs when two workers leave the network. Fig. 9(b) shows the average task completion time of the coded scheme when different numbers of workers leave the network. Additionally, Fig. 9(c) plots the corresponding task completion time for each iteration as some workers move away from the master. Notably, the results of the uncoded scheme are not presented in the figure as it fails to complete all the tasks in all cases. This is due to its dependency on the results from all workers to compute the final output. Therefore, when any worker moves out of the master's communication range, the absence of that worker's result renders the uncoded scheme ineffective. As illustrated in Fig. 9(d), in the case where there are two workers leaving the NAC network, the uncoded scheme fails to execute the last four iterations due to disconnections between the master and any of the two workers. However, as shown in Fig. 9(b)-9(c), although topology changes degrade the computation efficiency, the coded scheme is still able to complete all tasks efficiently, demonstrating its high resilience to topology changes. The specific number of workers leaving the coded scheme can tolerate depends on the level of redundancies incorporated, which is governed by the parameter q. In this simulation, with $q = 2m$, the coded scheme can tolerate up to 5 workers leaving the network.

Also of interest, the average task completion time, as depicted in Figs. 9(b)-9(c), does not necessarily increase when more UAVs leave the network. This is because, with the use of the coded scheme, task completion occurs once the master UAV receives sufficient results, specifically, those from any 5 workers. Since all workers possess identical computing capabilities and are assigned identical workloads, the time required for each worker to complete its assigned task is approximately uniform. This implies that the overall task completion time aligns with the time taken by each worker to complete its assigned task. Consequently, as long as there is an adequate number of workers, specifically 5, within the communication range of the master UAV, the task completion time remains resilient and doesn't deteriorate as more workers depart from the network. The variations in time shown in Figs. 9(c)-9(d) can be attributed to uncertain system disturbances, such as communication delays and slowdowns experienced by workers. For a more comprehensive understanding of the coded scheme, we recommend referring to references.[10,11]

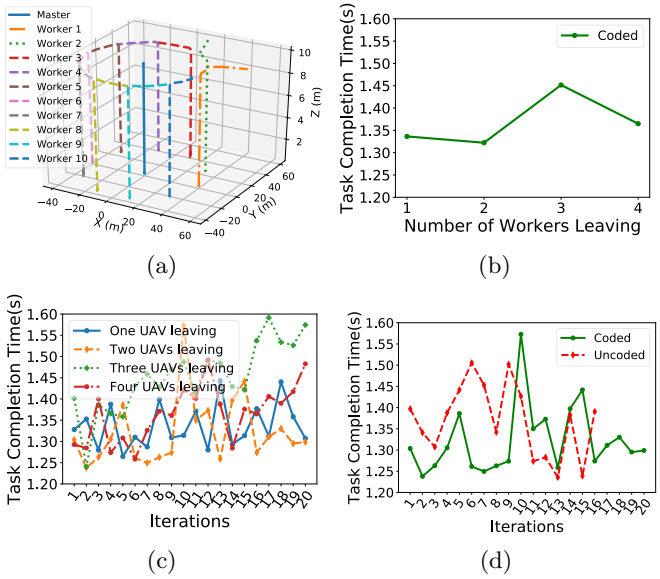


Fig. 9. Results of Simulation Four: (a) UAV trajectories in the scenario where two workers leave the network, (b) average task completion time of the coded scheme with varying numbers of workers leaving the network, (c) task completion time of different iterations for the coded scheme in different scenarios, and (d) task completion time of different iterations for different computing schemes in the scenario where two workers leave the network.

6. Hardware Tests

In this section, we utilize the designed NAC testbed to conduct hardware tests, providing us with additional insights into the NAC systems. Our primary objective through these tests is twofold. Firstly, we seek to gain a more comprehensive understanding of two crucial factors that have a great impact on NAC performance: U2U communication and UAV mobility. Secondly, we aim to showcase the practical application of the testbed by demonstrating its effectiveness in the context of forest fire detection. We conduct the outdoor flight tests at the sport fields at San Diego State University (SDSU) as illustrated in Fig. 10.

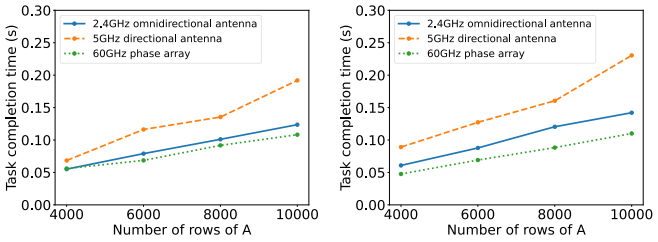
Fig. 10. Conducting flight tests using the NAC hardware testbed at the SDSU sport field.

6.1. *Comparison of U2U Communication Techniques*

Recognizing the vital role of U2U communication links within the NAC system, we compare three cutting-edge communication techniques for establishing the U2U communications. Through a comprehensive comparison, we thoroughly examine their performance and impact on the NAC performance.

To assess and compare their performance, we conduct a series of experiments in both indoor and outdoor settings. In the indoor scenario, UAVs are positioned in close proximity to each other. In the outdoor scenario, the two worker UAVs are near each other, with each worker located about 24m away from the master. Given the payload limitations of our current UAV platform, lifting the relatively large and heavy MikroTik wAP device is not currently feasible[a]. As a result, to ensure fairness, the UAVs are placed on the ground in all experiments. In each experiment, we let the NAC testbed repeatedly compute the matrix multiplication task 10 times and record the average time to complete each task. The matrices are randomly generated, with $\mathbf{X} \in \mathbb{R}^{2000\times 1}$ and the size of $\mathbf{A}$ varied. The uncoded distributed computing scheme is implemented to perform the computations.

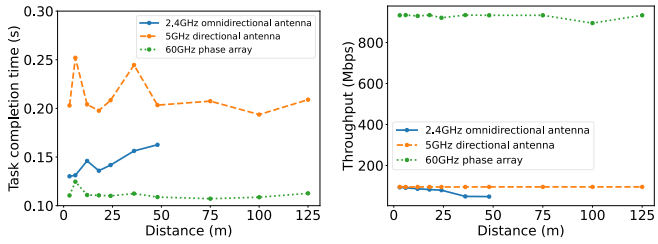
Fig. 11 shows the results obtained by using different communication devices in both indoor and outdoor settings. Comparing the two scenarios, the obtained results demonstrate consistency. Particularly, in both scenarios, using the 60GHz phased-array antenna-based device consistently yields the best performance. Interestingly, the 2.4GHz omnidirectional antenna-based device exhibits superior performance compared to the 5GHz directional antenna-based device. To gain a deeper understanding of this phenomenon, we conduct additional experiments in the outdoor environment. Specifically, we vary the distance between the master and the workers while keeping the distance between the two workers constant. This allows us to evaluate the performance of different communication devices under varying distances.



(a) (b)

Fig. 11. Average task completion time achieved using different communication devices for various task sizes in the (a) indoor and (b) outdoor scenarios.

The results are presented in Fig. 12(a), showing the average task completion time, and in Fig. 12(b), demonstrating the average throughput of the master-worker link, where the throughput is measured using Iperf3.[46] A notable observation is the absence of results for the 2.4GHz omnidirectional antenna-based device when the master-worker distance exceeds 48m. The reason behind this is that, in our experiments, the device often fails to establish reliable communication links beyond this distance, demonstrating its restricted transmission range. In contrast, our earlier experiments have demonstrated that the 5GHz directional antenna-based device can communicate over a long distance, reaching up to 5km while sustaining a throughput of 2Mbps.[22] Additionally, the 60GHz phased-array antenna-based device achieves a throughput that surpasses 800Mbps for communication distances of up to 250m. As the distance extends to 300m, the throughput decreases to around 400Mbps. Beyond the 350m mark, establishing a reliable connection becomes unfeasible.



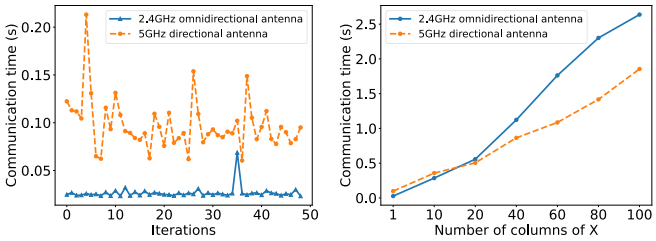
(a) (b)

Fig. 12. (a) Average task completion time and (b) throughput achieved using different communication devices for various master-worker distances.

Another noteworthy observation is that the 5GHz directional antenna-based device achieves a higher throughput (see Fig. 12(b)) compared to the 2.4GHz omnidirectional antenna-based device. However, interestingly, this higher throughput didn't result in improved computation performance, as depicted in Fig. 12(a). This phenomenon is likely attributed to the larger overhead required by the directional antenna-based device during the data transmission initiation process. To validate this hypothesis, we measure the time dedicated to computations and subtract it from the task completion time to obtain the communication time, which primarily includes the data transmission time and the time required to initiate the data transmission process. Fig. 13 shows the communication time recorded for running the matrix multiplication task with $\mathbf{A} \in \mathbb{R}^{10000\times 2000}$ and $\mathbf{X} \in \mathbb{R}^{2000\times 1}$ when the master-worker distance is set to 24m. The task is repetitively performed 50 times (iterations). This figure validates that the 5GHz device indeed requires more time to initiate the data transmission process compared to the 2.4GHz device. This is evident from the longer communication time of the 5GHz device, accompanied by its shorter data transmission time as indicated by its higher throughput (see Fig. 13(a)). However, it is anticipated that this limitation will be compensated for by its higher throughput when more data are transmit-

[a]This limitation will be addressed in the future by using larger UAV frames for increased payload capacity.

ted. To demonstrate this, we increase the data transmission size by expanding the number of columns in $\mathbf{X}$. Fig. 13(b) displays the average communication time across 50 iterations for the two communication devices. As we can see, the 5GHz device outperforms the 2.4GHz when the data transmission size surpasses a certain threshold.



(a) (b)

Fig. 13. Comparison of 2.4GHz and 5GHz devices in terms of (a) the communication time at different iterations when the number of columns of $\mathbf{X}$ is set to 1 and (b) the average communication time at different numbers of columns of $\mathbf{X}$.

6.2. *Impact of UAV Speed*

In this experiment, we aim to delve deeper into the impact of UAV mobility, particularly focusing on speed, on NAC performance. It is noted that the simulator's capabilities are inadequate for this study due to its simplified wireless communication module, which does not take into consideration the speed of UAVs. Consequently, the simulator does not reflect the potential impact of UAV speed on NAC performance.

To make UAV speed the only variable, we let the three UAVs fly in parallel, ensuring a roughly constant master-worker distance. Two worker UAVs fly on the two sides of the master UAV, maintaining an identical speed to that of the master. All UAVs fly back and forth between two ends of their predefined paths. The trajectories of these three UAVs are visualized in Fig. 14. Whenever the UAVs arrive at one end of their pre-defined paths, they pause at that location for 10 seconds and then resume their movement to the other end at a different speed. During the flight, the UAVs collaboratively execute the matrix multiplication task repeatedly.

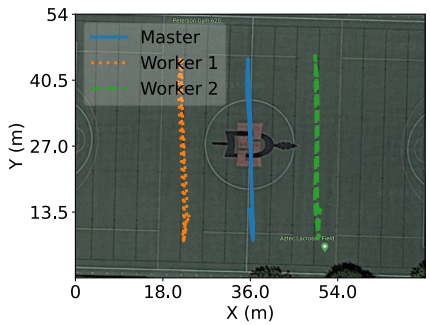
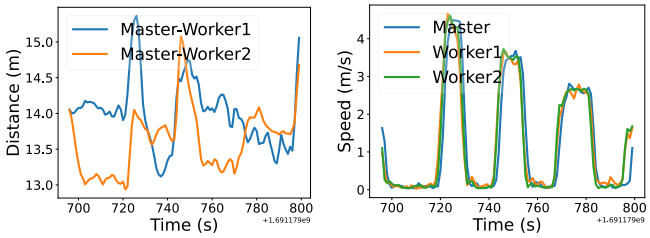


Fig. 14. UAV trajectories.

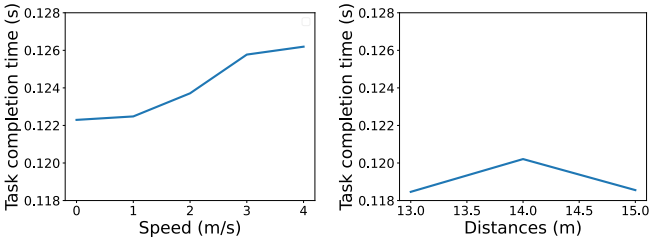
Fig. 15(a) shows the distances between the master and each worker calculated using the RTK coordinates recorded by the Pixhawk flight controller. As we can see, the master-worker distances remain relatively consistent, fluctuating only slightly within a narrow range of 13m to 16m throughout the test. Fig. 15(b) plots the speeds of each UAV calculated from the velocity data recorded by the flight controller. Note that the flat segments observed at the valleys of the speed curves indicate the time intervals during which the UAVs are hovering in place. The peaks of the speed curves occur when the UAVs approach the center of their predefined paths.



(a) (b)

Fig. 15. (a) Distances between the master and each worker, and (b) speeds of each UAV.

To illustrate the impact of UAV speed on NAC performance, we calculate the average task completion time across different UAV speeds. Specifically, we take the average of task completion times from iterations associated with similar speed values of the master UAV, ensuring that the speed difference does not exceed 1m/s. The results are shown in Fig. 16(b). As we can see, the task completion time increases with the ascent of UAV speed, indicating that higher UAV speeds will degrade NAC performance. This phenomenon could be attributed to the increased chance of data loss when UAVs move at higher speeds. To ensure that the performance degradation is not a result of an increased master-worker distance, we also plot in Fig. 16(b) the average task completion across different master-worker distances. The mean values are calculated by averaging over iterations associated with similar distance values between the master and worker 1. As shown in this figure, no significant influence of master-worker distance on NAC performance can be observed, compared to the impact of UAV speed.



(a) (b)

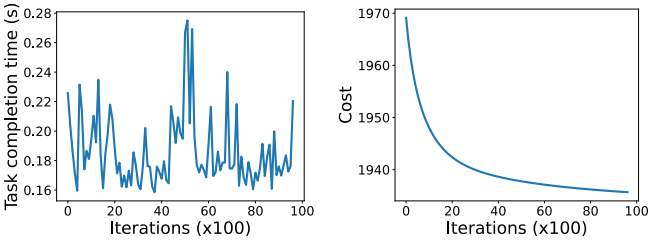
Fig. 16. Average task completion time at different (a) master speeds and (b) distances between the master and worker 1.

6.3. *Application in Forest Fire Detection*

To demonstrate the practicality of the designed NAC testbed, we also implement and evaluate a real application: forest fire detection. Particularly, we use a real-world dataset collected in Montesinho park[47] as input to train a forest fire detector. The goal is to predict burned areas given meteorological information by training a linear regressor using federated learning. The dataset consists of 517 meteorological data samples, each containing 13 features including location in the park, time, temperature, humidity, wind speed, amount of rainfall, and the burned area of the forest.

To train the linear regressor over the NAC testbed, the dataset is divided into $N = 2$ equal parts, with each part pre-stored in a worker. The training process starts with the master initializing the parameters of the linear regressor, which then updates these parameters 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. Throughout the entire training process, the three UAVs maintain a hovering position in the air, with the master being around 14m away from the workers.

Fig. 17(a) shows the time required to execute every 100 training iterations and Fig. 17(b) depicts the trajectory of the cost function's value after every 100 iterations. From Fig. 17(b), we can observe that the cost converges at around 10000 iterations, which takes about 18.51s. The average training time for each iteration is about 0.00185s.



(a) (b)

Fig. 17. (a) Time per completing 100 training iterations and (b) the associated training cost after every 100 iterations.

7. Discussions

The simulations and hardware tests reveal challenges in achieving high-performance NAC. When executing a decomposable computation task over an NAC system, several factors must be considered simultaneously when designing distributed computing schemes to achieve high computational efficiency and high resilience to uncertain system disturbances. These factors encompass task size, the distance between the master and workers, the number of workers involved, UAV mobility, and the communication technique used for U2U communications. It's important to note that UAV mobility influences not just the network topology but also the quality of U2U communications, a facet often overlooked by existing studies.[48–50]

There are several other important factors to consider depending on the application needs and the characteristics of the NAC platform. For example, the battery capacity of the UAV becomes important for applications that appreciate extended flight durations. Hence, energy consumption should be taken into account when designing distributed computing schemes. Moreover, UAVs may vary in terms of computing power or storage space, which should also be considered. The impacts of these two factors are not studied in this work due to the constraints of our existing simulator and hardware testbed. However, they can be easily enhanced to accommodate the requirements of such investigations. Particularly, an energy consumption model can be incorporated into the simulator to enable the measurement of energy consumed by the UAVs during computing, communication, and maneuvering. The computing power of the UAVs in the simulator can be varied by implementing appropriate limitations on the processes, such as installing the cpulimit module on Linux systems to regulate resource usage. For the hardware testbed, the energy consumption of the UAV can be measured using suitable tools such as the USB power meter. Additionally, we can deploy different single-board computers with different computing power and/or storage capacities to investigate heterogeneous scenarios.

The hardware tests also demonstrate the limitations of standard Wi-Fi routers used in the current testbed, which are equipped with omnidirectional antennas. They have a restricted communication range and provide a limited communication throughput. The directional antennas can largely extend the communication range, but their effectiveness is limited to a fixed direction. To achieve optimal performance, proper antenna direction control mechanisms are required to align the antennas at the transmitter and receiver.[22] As a result, they are not the ideal choice for point-to-multipoint or multipoint-to-multipoint communications. The phased-array antennas can potentially overcome these limitations and enable long-distance and broadband multipoint-to-multipoint communications. The MikroTik wAP 60G, which was tested in this study, shows promise as a candidate to partially achieve this. However, further designs and testing are necessary to successfully deploy it on the UAV platform and achieve robust multi-device communications. Moreover, as the device operates at the 60GHz band, its communication range is limited, with a maximum distance of around 350m according to our tests.

In comparison to the hardware testbed, the designed ROS and Gazebo-based simulator offers a range of benefits, including low-cost, low-risk, scalable, and flexible. However, we acknowledge the presence of one major limitation associated with the simulator, which is the high overhead introduced by running MPI and ROS together. This may be caused by the resource contention between MPI and ROS. We notice that when executing the simulator, ROS occupies a majority of the CPU resources, resulting in decreased

efficiency of MPI compared to running it alone. Moreover, the high overhead may also be caused by the differences in communication models of MPI and ROS, resulting in the need for coordination between the two systems, which incurs additional overhead. In addition, since the simulator relies on models, like the wireless communication model in (5), it cannot fully replicate the real hardware systems. Consequently, relying solely on it proves inadequate for attaining a comprehensive understanding of NAC and investigating all of its impact factors, such as the UAV speed.

8. Conclusion and Future Work

This paper introduces a realistic simulator and hardware testbed with complementary capabilities to facilitate NAC research. The simulator, built using ROS and Gazebo, emulates networked UAVs with inter-vehicle resource sharing and distributed computing capabilities. It can simulate diverse real-world scenarios both at small and large scales with low cost, high scalability, and exceptional flexibility, making it suitable for initial investigations. The hardware testbed, on the other hand, comprises three quadrotors, each featuring a Pixhawk control unit, Raspberry Pi computing unit, RTK localization unit, and telemetry radio and TP-Link Wi-Fi router for communications. The simulations and hardware tests conducted with various computation tasks and distributed computing schemes shed light on the critical factors that can influence NAC performance, which include task size, number of UAVs, U2U communication, and UAV mobility. The results also reveal challenges in achieving high-performance NAC and offer insights that guide further improvements to both the simulator and testbed. In the future, we will enhance the simulator by incorporating additional features that accommodate diverse application requirements. Additionally, we will focus on mitigating the overhead that arises when running MPI and ROS concurrently. To enhance the hardware testbed, we will focus on deploying phased-array antennas on the UAV platform to improve the U2U communication performance. Additionally, we plan to augment the testbed by equipping it with additional sensors, such as cameras, to enable the evaluation of a broader spectrum of applications and scenarios.

Acknowledgement

We would like to thank National Science Foundation under Grants CAREER-2048266 and CCRI-2235157/2235158/2235159/2235160 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* **6**(3) (2022) 489–498.

[2] E. Semsch, M. Jakob, D. Pavlicek and M. Pechoucek, Autonomous uav surveillance in complex urban environments, *in Proceedings of 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, **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* **37**(2008) (2008) 1187–1192.

[5] M. Elloumi, R. Dhaou, B. Escrig, H. Idoudi and L. A. Saidane, Monitoring road traffic with a uav-based system, *in Proceedings of 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 Proceedings of 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)*, (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* **11**(7) (2017) 969–974.

[8] K. Lu, J. Xie, Y. Wan and S. Fu, Toward uav-based airborne computing, *IEEE Wireless Communications* **26**(6) (2019) 172–179.

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

[10] B. Wang, J. Xie, K. Lu, Y. Wan and S. Fu, On batch-processing based coded computing for heterogeneous distributed computing systems, *IEEE Transactions on Network Science and Engineering* **8**(3) (2021) 2438–2454.

[11] B. Wang, J. Xie, K. Lu, Y. Wan and S. Fu, Learning and batch-processing based coded computation with mobility awareness for networked airborne computing, *IEEE Transactions on Vehicular Technology* **72**(5) (2023) 6503 – 6517.

[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* **14**(15) (2020)

2410–2419.

[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* **6**(5) (2019) 8753–8769.

[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* **8** (2020) 180784–180798.

[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* **10**(8) (2022) 6622–6633.

[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* **7**(1) (2020) 73–84.

[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* **22**(5) (2022) p. 2049.

[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 Proceedings of 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 Proceedings of 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* **13**(17) (2019) 2906–2916.

[23] R. K. Sheshadri, E. Chai, K. Sundaresan and S. Rangarajan, Skyhaul: An autonomous gigabit network fabric in the sky (2020).

[24] A. Y. Javaid, W. Sun and M. Alam, Uavsim: A simulation testbed for unmanned aerial vehicle network cyber security analysis, *in Proceedings of 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 Proceedings of 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: 2023-02-04.

[27] GAZEBO, Gazebo `https://gazebosim.org/home`, (2023), Accessed: 2023-02-04.

[28] TP-Link, Tl-wr902ac `https://www.tp-link.com/us/home-networking/wifi-router/tl-wr902ac/`, (2023), Accessed: 2023-02-04.

[29] U. NETWORKS, Nanostation loco m5 `https://store.ui.com/us/en/products/locom5`, (2023), Accessed: 2023-06-23.

[30] wAP60G, wap60g `https://mikrotik.com/product/wap_60g`, (2023), Accessed: 2023-02-04.

[31] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos and K. Ramchandran, Speeding up distributed machine learning using codes, *IEEE Transactions on Information Theory* **64**(3) (2017) 1514–1529.

[32] C. Wu and J. Z. Yu, Evaluation of linear regression techniques for atmospheric applications: the importance of appropriate weighting, *Atmospheric Measurement Techniques* **11**(2) (2018) 1233–1250.

[33] A. Reisizadehmobarakeh, S. Prakash, R. Pedarsani and S. Avestimehr, Coded computation over heterogeneous clusters, *arXiv preprint arXiv:1701.05973* **2**(3) (2017).

[34] A. Reisizadeh, S. Prakash, R. Pedarsani and A. S. Avestimehr, Coded computation over heterogeneous clusters, *IEEE Transactions on Information Theory* **65**(7) (2019) 4227–4242.

[35] D. Wang, B. Wang, J. Zhang, K. Lu, J. Xie, Y. Wan and S. Fu, Cfl-hc: a coded federated learning framework for heterogeneous computing scenarios, *2021 IEEE Global Communications Conference (GLOBECOM)*, IEEE (2021), pp. 1–6.

[36] ROS, Ros topic `http://wiki.ros.org/Topics`, (2023), Accessed: 2023-02-04.

[37] U. model, Hector quadrotor uav model `https://github.com/tu-darmstadt-ros-pkg/hector_quadrotor`, (2023), Accessed: 2023-02-04.

[38] M. Hata, Empirical formula for propagation loss in land mobile radio services, *IEEE transactions on Vehicular Technology* **29**(3) (1980) 317–325.

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

[40] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach* (Elsevier, 2007).

[41] MPI `https://ds.cs.luc.edu/mpi/mpi.html`, (2023), Accessed: 2023-02-04.

[42] Rviz, Rviz `http://wiki.ros.org/rviz`, (2023), Accessed: 2023-02-04.

[43] Q Ground Control http://qgroundcontrol.com/, (2023), Accessed: 2023-02-04.

[44] HERE3, Here3 rtk gps https://docs.cubepilot.org/user-guides/here-3/here-3-manual, (2023), Accessed: 2023-08-28.

[45] GPS, Gps https://www.u-blox.com/en/product/neo-m8-series, (2023), Accessed: 2023-08-30.

[46] Iperf 3 https://iperf.fr/iperf-download.php, (2023), Accessed: 2023-02-04.

[47] U. M. L. Repository, Forest fires data set https://archive.ics.uci.edu/ml/datasets/forest+fires, (2023), Accessed: 2023-02-04.

[48] J. Xiong, H. Guo and J. Liu, Task offloading in uav-aided edge computing: Bit allocation and trajectory optimization, *IEEE Communications Letters* **23**(3) (2019) 538–541.

[49] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao and G. Y. Li, Joint offloading and trajectory design for uav-enabled mobile edge computing systems, *IEEE Internet of Things Journal* **6**(2) (2018) 1879–1892.

[50] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao and X. Shen, Energy-efficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization, *IEEE Transactions on Vehicular Technology* **69**(3) (2020) 3424–3438.