UAV Swarm Exploration With Byzantine Fault Tolerance

Zhentang Liao[1], Lihua Zhang[1, 2, 3] and Zhiyan Dong[1, 2, 3]
[1]Institute of AI & Robotics, Fudan University, Shanghai, China
[2]Shanghai Engineering Research Center of AI & Robotics
[3]Engineering Research Center of AI & Robotics, Ministry of Education
Email: ztliao20@fudan.edu.cn; dongzhiyan@fudan.edu.cn

2021 China Automation Congress (CAC) | 978-1-6654-2647-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/CAC53003.2021.9727874

***Abstract*—Swarm robots need Byzantine fault tolerance to ensure their tasks can still be complete even if malfunctioning or malicious robots exist. Nevertheless, advancement in implementing Byzantine fault tolerance is not suitable for the UAV swarm exploration task. This study aims to contribute to this growing area of research. We presented an implementation of Byzantine fault tolerance for swarm robots, proposing a theoretically more effective and flexible approach than previous approaches. We integrated a multi-agent foraging algorithm with a BFT consensus algorithm for the UAV swarm exploration task, providing a method that can be generalized to similar multi-agent cooperative tasks. We developed a ROS-based framework to incorporate such algorithms into swarm robots, facilitating robotics development and testing. Empirical results of the UAV swarm exploration task imply our approach's advantages.**

I. Introduction

The Unmanned Aerial Vehicle (UAV) becomes increasingly widely used as it gets cheaper and more capable. Although individual UAVs only have limited computing resources and communicate with their neighbors, a swarm of UAVs can communicate with each other and make decisions intelligently as a group. Deploying UAVs in various scenarios has been a research hot-spot, for example, exploring a complex environment to support the search and rescue mission. [1] Autonomous UAVs can move between obstacles without a pre-acquired map and gather environmental information to complete their exploring task. Such tasks can be modeled as multi-agent foraging problems, and solutions usually require agents to exchange their information.

Task completion of UAVs relies on information exchange without Byzantine failures. Byzantine failures are used to describe robots' unintended or inconsistent behavior. [2] A Byzantine agent may not report after it finds the target, or it may maliciously claim to have found the target though it has not. A Byzantine agent may also send messages to make others believe it is located somewhere other than its actual location. Byzantine failures are caused by tampered messages or link interruption because agents or channels can be broken or attacked. Considering the limited wireless transmission range and unstable network topology of the Mobile Ad-hoc NETworks (MANETs), risks are higher when massive fast-moving UAVs compose a swarm or working conditions are poor. If Byzantine fault tolerance has not been implemented, UAVs cannot complete their tasks in most cases.

Implementing Byzantine fault tolerance for swarm robots has been of considerable interest in recent years, but advancement so far is not suitable for the UAV swarm exploration task. Byzantine fault tolerance should be considered in all stages of robotic development, from laboratory designs to real-world applications. [3] Because when related issues indeed arise in real-world applications, time-consuming redesign or replacement of existing design is inevitable. In typical asynchronous networks such as MANETs, Byzantine fault tolerance can be achieved using a consensus mechanism. Consensus can be non-strictly divided into two categories, BFT Consensus and Nakamoto Consensus. [4] The former category is a traditional topic in distributed systems, and the latter is used as a fundamental infrastructure in blockchain systems. Prior studies used consensus mechanisms to implement Byzantine fault tolerance for swarm robots, but they have not exploited the potential of multi-agent cooperative tasks.

Previous work investigated extensively on using consensus mechanisms in some tasks for swarm robots. However, few lessons can be drawn since those tasks differed from the UAV swarm exploration task in computational complexity. For example, distributed gathering algorithms that ensure all robots are gathered simultaneously on a plane have been widely investigated. [5]–[8] The algorithms need to successfully gather regular agents independently of the behavior of Byzantine agents. If a swarm of n agents performs the algorithms, the gathering task has a $O(m)$ complexity while the UAV swarm exploration task has an $O(m/n)$ complexity. Because in the gathering task, each agent has to calculate its path from the current position to the goal. However, in the UAV swarm exploration task, visited areas usually do not need to visit again, and it is enough for one agent to find the target. Other studies mainly investigated tasks in the same computational complexity as the gathering task, such as receiving control commands from multiple base stations,

This work was supported by grants from the Department of Science and Technology of Guangdong Province (No. 2019A1515110352), the Open Project of Jihua Lab Project (No. X190021TB194), Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0103), Science and Technology Commission of Shanghai Municipality (No. 19511132000), Ningbo Science and Technology Bureau (No. 2020Z073), the Project of Guangxi University Young and Middle-aged Teachers Scientific Research Basic Ability Improvement (No. 2021KY0052), and the Project of Guangxi Normal University Major Scientific and Technological Achievements Transformation and Cultivation (No. 2020PY001).

unifying sensor readings on environmental information, and synchronous beeping without communication. [9]–[11] Using consensus mechanisms in tasks with different computational complexity are left as a gap in these studies.

Several researchers have investigated tasks similar to the UAV swarm exploration task for swarm robots. However, their studies were limited to the computational complexity of different problems. The search problem of finding a target on a circle has been considered. [12] The search was completed when a regular agent found the target and other regular agents confirmed the finding. Agents were randomly placed on the circle's circumference and could only move along it. The worst-case search completion time was investigated. Another search problem on a circle's plane has also been considered, and algorithms to minimize the search time has been designed. [13] Lack of considering implementation of consensus mechanism in these studies would make applying their methods to other problems be difficult.

Recent progress focuses on developing blockchain-based smart contracts for swarm robots. However, this approach needs demanding computational and network resources to run a blockchain system. A secure swarm coordination framework via smart contracts has been proposed based on the interface between the simulator ARGoS and the blockchain framework Ethereum. [2], [3] Several consensus algorithms have been compared in the framework. The feasibility of deploying blockchain systems and running smart contracts on robots has been demonstrated using a swarm composed of Pi-puck robots. [14] Though using blockchain systems' inherited consensus mechanisms to ensure task-specific data consistency is feasible, it would be unnecessary to deploy blockchain systems on resources-limited embedded systems such as UAVs.

There have been studies highlighting the use of consensus mechanisms considering network topology or architecture for swarm robots. However, this approach often produces unnecessary communication costs since it is not task-specific. On the data link layer, the PBFT-based consensus decision-making procedure has been used to select an appropriate MAC protocol, and the consensus was based on data packets transmitted on the shared channel. [15] On the network layer, a routing protocol has been designed to handle route detection and fault avoidance, and the consensus was reached through fault-free paths. [16] A trusted networking framework using consensus packet as control signal has been proposed, including network architecture, protocol stack, and algorithms. [17] Communicating using these protocols or networking frameworks would not be effective because consensus mechanisms need to process information much more than task-specific data.

In this paper, we presented an implementation of Byzantine fault tolerance for swarm robots, proposing a theoretically more effective and flexible approach than previous approaches. Comparing to other approaches that use blockchain systems or consider network properties, resources consumption has been minimized because we only processed task-specific data. In contrast to studies that focus on the computational complexity of their problem, we provided a flexible approach to use consensus mechanisms in similar multi-agent cooperative tasks. We integrated a multi-agent foraging algorithm with a BFT consensus algorithm for the UAV swarm exploration task, providing a method that can be generalized to similar multi-agent cooperative tasks. We developed a ROS-based framework to incorporate such algorithms into swarm robots, facilitating robotics development and testing.

The remainder of this paper is structured as follows. Section II describes the developed framework and the chosen algorithms. Section III describes the setup of simulated experiments and presents the empirical results. Section IV discusses our findings and concludes our work.

II. Methods

We consider a fixed set of $n = 3k + 1$ agents, indexed by $i \in [n]$ where $[n] = 1, \ldots, n$. There is a set $K \subset [n]$ of up to $k = |K|$ Byzantine agents, and the remaining ones are regular agents. UAVs as agents compose a swarm and perform our chosen algorithms to search a complex environment for one target. The task can only be completed in finite time with probability 1 if Byzantine fault tolerance has been achieved. One regular agent receives a message from another regular agent if and only if the latter sent that message to the former. All messages can be delivered within a known bound time.

We adapted the c-marking multi-agent foraging algorithm for the swarm to find one target in a complex environment. [18] The c-marking multi-agent foraging algorithm is essentially a parameter-free distributed and asynchronous version of the wavefront algorithm, which can be regarded as finding the shortest paths on a graph using breadth-first search. [19] In the perspective of our chosen multi-agent algorithm, agents take action to move in the environment periodically and individually. So the number of rounds of moving can be used as a performance metric, fewer rounds for a swarm to find the target indicating better task completion.

We used the Chained HotStuff BFT consensus algorithm as the consensus mechanism. [20] The Chained HotStuff BFT algorithm uses a three-phase commitment scheme, and it can pipeline the commitment in a chain to achieve almost-linear message complexity. We assumed that the MANETs could form a connected graph at least once in each round to simplify simulated experiments. All agents were able to communicate directly or through other relaying agents to take part in reaching consensus. With the pipeline style of our chosen consensus algorithm, the consensus can be reached in one round on average.

We used a map to integrate our chosen algorithms. Every agent had a replica of the map that the multi-agent algorithm took as input, and changes of the replica were based on the consensus algorithm. The map, an occupancy grid map represented by a vector of coordinates and tags, contained global positions of agents and objects. The origin of the map's coordinate system was defined near UAVs' starting point, calibrated by their Simultaneous Localization and Mapping (SLAM) output.

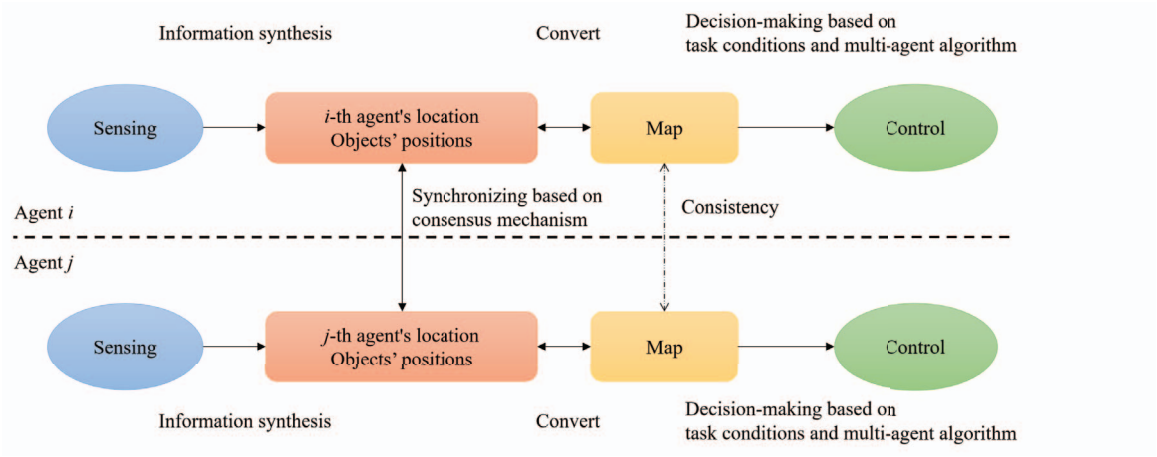


Fig. 1. Planning architecture of our framework.

We developed a ROS-based framework to incorporate algorithms into swarm robots. An autonomous UAV usually has an on-board computer and a flight controller. They can use ROS and PX4 to control other hardware on the UAV, including visual system, wireless transmission system, propulsion system, and power supply system. [21], [22] ROS is a set of software libraries and tools for creating robot applications, and PX4 is a control software for UAV and other unmanned vehicles. ROS running on the on-board computer provides interfaces for our framework to communicate with PX4 running on the flight controller. ROS and PX4 were initially designed for working in embedded systems, but they can also be deployed on computers together as a simulation platform. Because of the minimized sim-to-real gap in PX4 Software-In-The-Loop simulation, algorithms that work properly in simulation can be directly transferred to real UAV or other real robots with ROS-interface controllers. Our framework used a modified version of the 3DR Iris multi-copter in the physics-based Gazebo simulator. [23]

Our framework included software and hardware support. The software was a modularized application consisting of 3 modules: 1) State module for synthesizing sensor output and synchronizing different agents' information based on the consensus algorithm; 2) Action module for generating control input based on task conditions and the multi-agent algorithm; 3) Device module for using hardware. The software was running with ROS under Ubuntu on the on-board computer. The hardware support contained multi-copters that conform to Pixhawk Standard, a design specification and guideline for PX4 compatible products. Hardware supports for other robots with ROS-interface controllers can be added easily in the Device module.

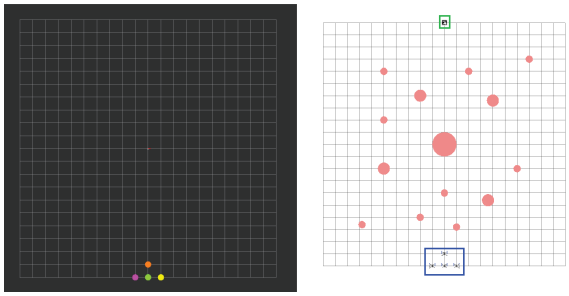
The planning architecture of our framework is depicted in Fig. 1, and agents' behaviors in each round are listed below. UAVs would constantly take action until they triggered some other task conditions, for instance, returning to recharge when power is not enough. UAVs used their visual system to detect objects, including other UAVs and the target, and they used sensor fusion to locate themselves on the map. Objects' positions in broadcasting messages were outputted from SLAM for cross-validation. Our framework used the Real-Time Appearance-Based Mapping as SLAM and the OpenCV for object detection. [24]

1) Each agent located itself in the environment, detected objects within visual range, and broadcasted object detection outcomes with environmental characteristics to its neighbors.
2) Each agent updated its map with received messages using the consensus algorithm. If one regular agent found the target and other regular agents confirmed the outcome, agents reported the outcome.
3) Each agent used the multi-agent algorithm to decide its moving direction in the next round and moved on.

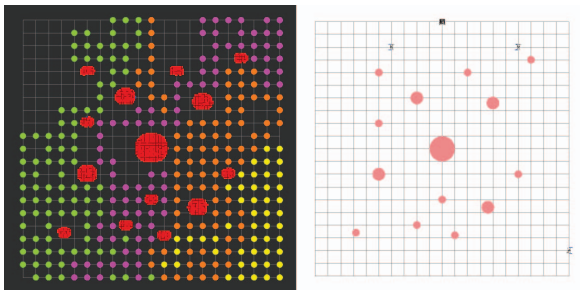
III. Results

To provide a proof-of-concept, we simulated situations where a fixed number of Byzantine agents are known, either with or without consensus mechanisms. The type of specific agent might change in each round, but the number of agents of different types remained the same. Consensus mechanisms in the asynchronous network can only guarantee a swarm of $3k+1$ agents to achieve Byzantine fault tolerance with at most k Byzantine agents and no less than $2k+1$ regular agents. So we used a swarm of 4 agents with 0 or 1 Byzantine agent as a minimum viable condition for simulated experiments.

We calculated the rounds of completing the UAV swarm exploration task 30 times in each of the two environments, either with or without obstacles. The task was completed when agents found the target and reported their outcome in situations with consensus mechanisms. In situations without consensus mechanisms, the task was recorded as completed when every agent found the target at least once. Situations were identical when no Byzantine agents existed, whether with consensus mechanisms or not. If the round had exceeded the worst case of traversing the environment, the task would record as failed in a given large number.



(a) Task began in the environment with obstacles.



(b) Task completed in the environment with obstacles.

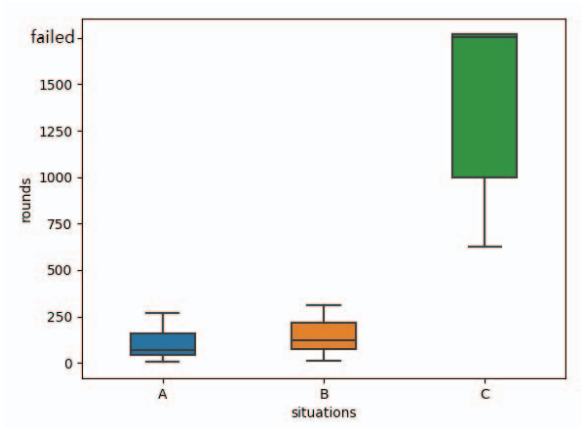
Fig. 2. Visualized example of simulated experiments. The map on the left and the environment on the right. In (a), obstacles in red, UAVs in the blue box, and QR code in the green box. In (b), The crosses on the map were marked in different colors when UAVs passed.

The visualized example of a simulated experiment is shown in Fig. 2. UAVs started from a fixed point to find one randomly placed target, a downward-looking visible only QR code on the ground. In the area of a 20 m $\times$ 20 m square, UAVs moved 1 m in four directions parallel to the grid's edges on the map in each round.

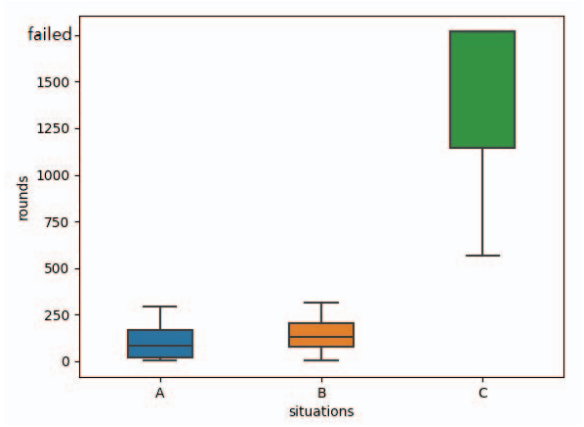
The results of simulated experiments are shown in Fig. 3. As can be seen, in situations with consensus mechanisms, most of the number of rounds when Byzantine agents existed (situation A) is less than the maximum number of rounds when Byzantine agents did not exist (situation B). Besides, the numbers of rounds in situations with consensus mechanisms are far less than the average number of rounds when Byzantine agents existed without consensus mechanisms (situation C), which nearly failed. The distribution of rounds indicates that Byzantine fault tolerance was achieved using our approach since the task would fail in most cases otherwise. It can also be seen that the average number of rounds when Byzantine agents existed (situation A) is less than the one when Byzantine agents did not exist (situation B). Because if Byzantine agents were not Byzantine, they could have contributed to complete the task using our chosen multi-agent algorithm.

IV. Discussion

Empirical results of the UAV swarm exploration task imply our approach's advantages. Prior studies have not implemented Byzantine fault tolerance for swarm robots in similar multi-agent cooperative tasks. We have not yet compared our frame-



(a) Results in the environment without obstacles.



(b) Results in the environment with obstacles.

Fig. 3. Results of simulated experiments. In (a) and (b), situation A represents that Byzantine agents did not exist, situation B represents that Byzantine agents existed with consensus mechanisms, and situation C represents that Byzantine agents existed without consensus mechanisms.

work with other approaches that use blockchain systems or consider network properties by implementing on the same hardware or in the same network, which is on schedule. However, it is evident that our approach consumes much fewer hardware resources or less network throughput than other approaches since only task-specific data were processed. The difference will become more significant as more UAVs join a swarm.

Our chosen algorithms are not limited to the proof-of-concept. Our chosen consensus algorithm also supports the situation when the number of agents of different types is unknown or change dynamically. For agents other than the one that found the target, the stochastic property of our chosen multi-agent algorithm provided opportunities to find the target again and confirm the outcome themselves, which has been used in the situation without consensus mechanisms. Suppose other algorithms were used, the task might be unable to be completed in a given time since regular agents could be maliciously led to be far away from the target or consensus could not be reached.

For the generalization of our approach, the map or the algo-

rithms can be modified or replaced to satisfy the needs of other similar multi-agent cooperative tasks. While the simulation used multi-copters, our chosen multi-agent algorithm supports different mobile robots, such as inspection robots, Unmanned Ground Vehicles, and Unmanned Submarine Vehicles. Fixed-wing is supported by adjusting the size of grids on the map or adding motion primitive. The developed framework is still being improved to be open source and keep pace with other ROS-based UAV frameworks to facilitate robotics development and testing. [25]–[27] The code to reproduce the empirical results is available in [28].

In conclusion, the implementation of Byzantine fault tolerance for swarm robots was presented using our approach. A multi-agent foraging algorithm and a BFT consensus algorithm were integrated for the UAV swarm exploration task. Such algorithms can be incorporated into swarm robots using our developed ROS-based framework.

REFERENCES

[1] K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, vol. 4, no. 35, 2019.

[2] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Blockchain Technology Secures Robot Swarms: A Comparison of Consensus Protocols and Their Resilience to Byzantine Robots," *Frontiers in Robotics and AI*, vol. 7, p. 54, 2020.

[3] V. Strobel, E. Ferrer, and M. Dorigo, "Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 541–549.

[4] Z. Ren, H. Xiang, Z. Zhou, N. Wang, and H. Jin, "AlphaBlock: An Evaluation Framework for Blockchain Consensus Algorithms," in *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*. Virtual Event Hong Kong: ACM, 2021, pp. 17–22.

[5] N. Agmon and D. Peleg, "Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 56–82, 2006.

[6] X. Défago, M. Gradinariu, S. Messika, and P. Raipin-Parvédy, "Fault-Tolerant and Self-stabilizing Mobile Robots Gathering," in *Distributed Computing*, ser. Lecture Notes in Computer Science, S. Dolev, Ed. Springer Berlin Heidelberg, 2006, vol. 4167, pp. 46–60.

[7] J. Czyzowicz, R. Killick, E. Kranakis, D. Krizanc, and O. Morales-Ponce, "Gathering in the plane of location-aware robots in the presence of spies," *Theoretical Computer Science*, vol. 836, pp. 94–109, 2020.

[8] X. Défago, M. Potop-Butucaru, and P. Raipin-Parvédy, "Self-stabilizing gathering of mobile robots under crash or Byzantine faults," *Distributed Computing*, vol. 33, no. 5, pp. 393–421, 2020.

[9] R. Guerraoui and A. Maurer, "Byzantine Fireflies," in *Distributed Computing*, ser. Lecture Notes in Computer Science, Y. Moses, Ed. Springer Berlin Heidelberg, 2015, vol. 9363, pp. 47–59.

[10] W. Xu, M. Wegner, L. Wolf, and R. Kapitza, "Byzantine Agreement Service for Cooperative Wireless Embedded Systems," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2017, pp. 10–15.

[11] N. Grigoropoulos, M. Koutsoubelias, and S. Lalis, "Byzantine fault tolerance for centrally coordinated missions with unmanned vehicles," in *Proceedings of the 17th ACM International Conference on Computing Frontiers*. ACM, 2020, pp. 165–173.

[12] K. Georgiou, E. Kranakis, N. Leonardos, A. Pagourtzis, and I. Papaioannou, "Optimal Circle Search Despite the Presence of Faulty Robots," in *Algorithms for Sensor Systems*, ser. Lecture Notes in Computer Science, F. Dressler and C. Scheideler, Eds. Springer International Publishing, 2019, vol. 11931, pp. 192–205.

[13] J. Czyzowicz, M. Godon, E. Kranakis, and A. Labourel, "Group search of the plane with faulty robots," *Theoretical Computer Science*, vol. 792, pp. 69–84, 2019.

[14] A. Pacheco, V. Strobel, and M. Dorigo, "A Blockchain-Controlled Physical Robot Swarm Communicating via an Ad-Hoc Network," in *Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, T. Stützle, M. J. Blesa, C. Blum, H. Hamann, M. K. Heinrich, and V. Strobel, Eds. Springer International Publishing, 2020, vol. 12421, pp. 3–15.

[15] M. Zhang, C. Dong, and Y. Huang, "FS-MAC: An adaptive MAC protocol with fault-tolerant synchronous switching for FANETs," *IEEE Access*, vol. 7, pp. 80 602–80 613, 2019.

[16] B. Awerbuch, D. Holmer, and H. Rubens, "Swarm intelligence routing resilient to byzantine adversaries," in *International Zurich Seminar on Communications, 2004*. IEEE, 2004, pp. 160–163.

[17] X. Jian, P. Leng, Y. Wang, M. Alrashoud, and M. S. Hossain, "Blockchain-Empowered Trusted Networking for Unmanned Aerial Vehicles in the B5G Era," *IEEE Network*, vol. 35, no. 1, pp. 72–77, 2021.

[18] O. Simonin, F. Charpillet, and E. Thierry, "Revisiting wavefront construction with collective agents: An approach to foraging," *Swarm Intelligence*, vol. 8, no. 2, pp. 113–138, 2014.

[19] O. Zedadra, N. Jouandeau, H. Seridi, and G. Fortino, "Multi-Agent Foraging: State-of-the-art and research challenges," *Complex Adaptive Systems Modeling*, vol. 5, no. 1, p. 3, 2017.

[20] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT Consensus with Linearity and Responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. ACM, 2019, pp. 347–356.

[21] ROS Powering the world's robots. [Online]. Available: https://www.ros.org/

[22] Open Source Autopilot for Drones. PX4 Autopilot. [Online]. Available: https://px4.io/

[23] Gazebo. [Online]. Available: http://gazebosim.org/

[24] RTAB-Map Real-Time Appearance-Based Mapping. [Online]. Available: http://introlab.github.io/rtabmap/

[25] Amovlab, "Prometheus autonomous uav opensource project," https://github.com/amov-lab/Prometheus.

[26] Autonomous drones generalized autonomy aviation system. [Online]. Available: https://www.gaas.dev/

[27] K. Xiao, S. Tan, G. Wang, X. An, X. Wang, and X. Wang, "XTDrone: A Customizable Multi-rotor UAVs Simulation Platform," in *2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*. IEEE, 2020, pp. 55–61.

[28] Usewbft. [Online]. Available: https://gitee.com/usewbft/usewbft