BLOCKCHAIN BASED DECENTRALIZED DATA MANAGEMENT MODEL FOR SWARM UAV SYSTEMS

SÜRÜ İHA SİSTEMLERİ İÇİN BLOK ZİNCİRİ TEMELLİ MERKEZİYETSİZ VERİ YÖNETİM MODELİ

GÖKBERK AÇIKGÖZ

PROF. DR. SUAT ÖZDEMİR

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Computer Engineering

June 2023

ABSTRACT

BLOCKCHAIN BASED DECENTRALIZED DATA MANAGEMENT MODEL FOR SWARM UAV SYSTEMS

Gökberk AÇIKGÖZ

Master of Science , Computer Engineering

Supervisor: Prof. Dr. Suat ÖZDEMİR

June 2023, 93 pages

The rapid development of Unmanned Aerial Vehicles (UAVs) and the Internet of Things (IoT) has given rise to new possibilities for decentralized communication and data sharing among connected devices. Swarm UAV systems, in particular, require robust and secure data management solutions to ensure safety, continuity, and efficient collaboration. Blockchain technology, known for its decentralized nature, offers a promising solution to address these challenges. This study proposes a novel data management model that integrates blockchain technology into UAV systems, with a special focus on swarm UAV systems. By leveraging the decentralized structure of blockchain, the proposed system provides enhanced robustness against tampering and irreversibility, ensuring secure data transfer between UAVs and decentralized processing. The model is specifically designed to facilitate mission plan and command upload operations, allowing for the distribution of critical information to all UAVs in the system. Additionally, the proposed model enables efficient task transfer between nodes without direct communication. Since inputs for required tasks are continuously updated in the blockchain, upon task creation, inputs, task type, and other relevant task information can be shared via the blockchain within the system. This eliminates the necessity for numerous message transfers among the connected UAVs, streamlining communication and

collaboration within the swarm. Through the implementation of this blockchain-based data management model, we aim to contribute to the ongoing research and development of secure and efficient swarm UAV systems, paving the way for more advanced and reliable applications in various domains.

Keywords: Blockchain, UAV, IoT, Swarm, Smart Contracts Blockchain, UAV, IoT, Swarm, Smart Contracts

ÖZET

SÜRÜ İHA SİSTEMLERİ İÇİN BLOK ZİNCİRİ TEMELLİ MERKEZİYETSİZ VERİ YÖNETİM MODELİ

Gökberk AÇIKGÖZ

Yüksek Lisans, Bilgisayar Mühendisliği

Danışman: Prof. Dr. Suat ÖZDEMİR

Haziran 2023, 93 sayfa

İnsansız Hava Araçları (İHA) ve Nesnelerin İnterneti (IoT) alanındaki hızlı gelişmeler, bağlantılı cihazlar arasında merkezi olmayan iletişim ve veri paylaşımı için yeni olanaklar sunmaktadır. Özellikle sürü İHA sistemleri, güvenlik, süreklilik ve etkili işbirliği sağlamak için sağlam ve güvenli veri yönetimi çözümlerine ihtiyaç duymaktadır. Blockchain teknolojisi, merkezi olmayan yapısı ile bu zorluklara çözüm sunan umut verici bir teknolojidir. Bu çalışma, özellikle sürü İHA sistemleri üzerinde yoğunlaşarak, blockchain teknolojisini İHA sistemlerine entegre eden yeni bir veri yönetimi modeli önermektedir. Blockchain'in merkezi olmayan yapısından yararlanarak, önerilen sistem, İHA'lar arasında güvenli veri transferi ve merkezi olmayan işlem sağlamak için manipülasyona karşı güçlendirilmiş ve geri alınamaz bir yapı sunar. Model, özellikle görev planı ve komut yükleme işlemlerini kolaylaştırmak üzere tasarlanmış olup, sisteme dahil olan tüm İHA'lara kritik bilgilerin dağıtılmasına olanak tanır. Ayrıca, önerilen model düğümler arasında doğrudan iletişim gerektirmeden etkili görev transferi sağlar. Gerekli görevler için girdiler sürekli olarak blockchain'de güncellendiğinden, görev oluşturulduğunda girdiler, görev türü ve diğer ilgili görev bilgileri sistemin içinde blockchain aracılığıyla paylaşılabilir. Bu, bağlantılı İHA'lar arasında çok sayıda mesaj transferi ihtiyacını ortadan kaldırarak, sürü

içinde iletişimi ve işbirliğini hızlandırır. Bu blockchain tabanlı veri yönetimi modelinin uygulanmasıyla, güvenli ve verimli sürü İHA sistemlerinin araştırma ve geliştirmesine katkıda bulunmayı ve çeşitli alanlarda daha ileri düzey ve güvenilir uygulamalar için zemin hazırlamayı amaçlamaktayız.

Keywords: Blok Zinciri, İHA, IoT, Sürü, Akıllı Kontratlar

ACKNOWLEDGEMENTS

I wish to express my profound gratitude to my supervisor, Prof. Dr. Suat OZDEMIR, who has been a beacon of guidance throughout this academic voyage. His wisdom, patience, and persistent encouragement were pivotal in shaping this thesis.

My heartfelt appreciation extends to my mentor, Asst. Prof. Feyza YILDIRIM OKAY. Her consistent engagement and profound expertise enriched my thesis journey immeasurably. I am also deeply grateful to Assoc. Prof. Adnan OZSOY, whose constructive feedback and unflinching support were invaluable.

A special note of thanks goes to Turkish Aerospace for their generous provision of resources and cooperation. I am also grateful to my superiors and my colleagues, Ihsan YAYLA and Umut Serkan YAVUZ, whose support was vital in bringing this research to fruition.

My deepest acknowledgments go to my family and friends, who have been a bedrock of support throughout this endeavor. Their steadfast love and faith in my abilities were the fuel that kept me going, even in challenging times. A special tribute goes to my parents, Mustafa ACIKGOZ and Candan ACIKGOZ, who have been the lighthouses guiding my path. Their unwavering love and constant encouragement have been my strength.

Lastly, I am profoundly grateful to my girlfriend, Canan AKDAS, whose love, understanding, and unyielding support have been the pillars on which I leaned throughout this journey. Her endless patience and encouragement have been invaluable, and for her love, I am eternally thankful.

CONTENTS

Page

TABLES

FIGURES

ABBREVIATIONS

UAV	:	**U**nmanned **A**erial **V**ehicle
IoT	:	**I**nternet **o**f **T**hings
GCS	:	**G**round **C**ontrol **S**tation
SATCOM	:	**SAT**ellite **COM**munication
LOS	:	**L**ine **o**f **S**ight
MEC	:	**M**obile **E**dge **C**omputing
FCC	:	**F**light **C**ontrol **C**omputer
EO-IR	:	**E**lectro **O**ptic - **I**nfra**R**ed

1. INTRODUCTION

A growing application of IoT technology involves the utilization of Unmanned Aerial Vehicles (UAVs) as IoT devices. Nowadays, UAVs are employed in various applications, including military operations, package deliveries, agriculture, aerial photography, search and rescue missions, and scientific research, among others. The fundamental idea behind UAVs is to design an aircraft that can fly autonomously or semi-autonomously without requiring human pilots in control. UAVs are often used in situations where it may be too dangerous, difficult, or impractical for a human pilot to operate an aircraft [5].

UAVs are versatile in their capabilities and can be equipped with various payloads of differing types and weights. Since UAVs have no human pilots on board, they are operated remotely from Ground Control Stations (GCS). The payload capabilities of UAVs can be increased through the use of various GCS applications. Hence, the development of GCS technology is crucial for UAVs to integrate with other technologies. Establishing reliable communication between the UAV and the GCS is a major challenge, and there are many communication systems available with varying degrees of security. Especially in military applications, ensuring secure communication is especially important to prevent data hijacking, system takeover, or the unauthorized disclosure of classified information [6].

Blockchain is a decentralized technology that provides secure and anonymous data storage for IoT [7]. Blockchain technology offers a transparent, safe, and always operational method of storing data through a network of data nodes registered to a chain, with each system holding a copy of the same chain. The unique structure of the blockchain ensures data integrity, and any inappropriate changes to the data will render the block invalid, thereby making the remaining part of the chain invalid. Current researches in blockchain technology focus on developing new cryptocurrencies with advanced blockchain technologies, decentralized apps, and the concept of Web 3.0.

Swarm UAV systems have gained popularity in recent years due to their potential to achieve tasks more efficiently and timely than single UAVs. However, managing and coordinating

a swarm of UAVs is a challenging task, particularly in communication and data sharing. These systems require a reliable and secure communication infrastructure that provides seamless connectivity between UAVs and GCSs and between different UAVs in the network. Furthermore, swarm UAV systems must also ensure that the data generated by UAVs is available to all relevant parties while protecting the data from unauthorized access or tampering [8]. Blockchain technology can meet the requirements of swarm UAV systems, as it offers secure and reliable data acquisition, which is one of the key features of this technology.

In swarm UAVs, distributing flight information, mission plans, and commands between swarm UAVs is crucial to prevent the failure of a single point of control, such as a master drone. However, communication interruptions due to environmental factors can pose a challenge, requiring a flawless transmission scheme. Additionally, coordinating the operations of different UAVs in a swarm can also be challenging due to the limited capabilities and varying payloads of UAVs. This can further exacerbate issues when specific tasks may require more inputs from different sensors with heavy computations. Designing a proper system for task transferring among swarm UAVs needs to compete with given challenges and issues to ensure successful task completion. Blockchain-based models can be developed to overcome these challenges.

The integration of blockchain into swarm UAV systems could potentially enable the formation of merged swarms, wherein multiple UAV swarms collaborate and operate cohesively. By leveraging the inherent features of blockchain, such as decentralized consensus, immutability, and secure transaction validation, the connection of additional GCSs to the blockchain network may offer a means of achieving transparent and secure communication and coordination among different swarm units. This integration may enable efficient resource allocation, task assignment, and information sharing, thereby enhancing the collective capabilities and operational scope of swarm-based UAV systems. However, further research is required to investigate the technical feasibility, scalability, and potential challenges associated with this approach, in order to ascertain its viability and effectiveness in real-world applications.

1.1. Scope of the Thesis

This thesis mainly focuses on a data management model for swarm UAV systems, which leverage blockchain technology to facilitate the management of operations and ensure peer-to- peer communication, and enable decentralized data sharing among UAVs in the network. Utilizing a blockchain-based architecture, our proposed model provides an expandable a platform that allows for the easy integration of new swarm nodes, UAVs, and GCSs into the network. We focus on integrating blockchain into swarm systems to provide reliable, and immutable data acquisition for aviation applications. We analyze the advantages and challenges of using blockchains in UAVs.

With the advancements in blockchain technology, the proposed model is distinguished from existing studies by focusing on distributing mission plans and commands to UAVs in a swarm, enabling faster integration of new UAVs and eliminating the need for a master drone. Additionally, the system utilizes blockchain for automatic task assignment and data collection, simplifying the complex messaging process by querying the entire UAV network.

The thesis has focused on primary operations that are standard for all UAV systems that could be managed by blockchain and which are not suitable for being handled by blockchain by measuring the model created. Primary operations are Mission Plan Upload, Command Upload, Payload Information Retrieval, UAV Status Tracking. Additionally, Mission State and Task Transfer operations will be included in the model to check if those operations suit blockchain.

1.2. Contributions

This research covers these deficiencies by proposing a novel, simple and efficient approach. The main contributions of this paper can be summarized as follows:

- The model enables data sharing for all UAVs connected. Subscribing as a node to the blockchain system provides a connection to the swarm without any complex network configuration to receive data roaming among UAVs in the system.

- The model especially provides mission plan and command retrieval from a decentralized system without any need for a complex data transfer scheme.
- The model provides efficient task transfer between nodes without communicating with each other. Since inputs for required tasks are continuously up to date in blockchain, on task creation, inputs, task type, and other task information can be shared via blockchain in the system, which removes the necessity for many message transfers among UAVs connected.

1.3. Organization

The organization of the thesis is as follows:

- Chapter 1 presents our motivation, contributions, and the scope of the thesis.
- Chapter 2 provides background information about UAVs and Blockchain.
- Chapter 3 gives related work about Blockchain on UAVs.
- Chapter 4 introduces details, flows, and contents of the model.
- Chapter 5 demonstrates measurements and transactions around the model.
- Chapter 6 states the thesis summary and possible future directions.

2. BACKGROUND INFORMATION

2.1. Unmanned Aerial Systems

Unmanned aerial systems are complete systems that include and control UAVs. UAVs can be classified based on various design criteria, such as size, wing type, and usage area. In this study, UAVs are classified into two types based on their design: rotary-winged UAVs and fixed-winged UAVs.

Rotary-winged UAVs, commonly called drones, are favored in closed areas due to their small size and high maneuverability. These drones have limited endurance due to the high power consumption required for mid-air stability, with batteries as their primary power source. Despite their endurance limitations, rotary-winged UAVs can perform a variety of applications, such as search and rescue missions, surveillance, mapping, and delivery services. Additionally, their ability to operate in swarm systems, where multiple drones work together towards a common goal, allows for more efficient operations in certain applications. However, communication and coordination between drones are crucial for efficient swarm operation.

In contrast, fixed-winged UAVs have higher endurance and payload capacity than rotary-winged UAVs, making them suitable for military operations and continuous surveillance. Control surfaces control them and can carry payloads such as EO-IR cameras, SATCOM modules, and military equipment. The endurance of a fixed-winged UAV varies based on aerodynamics, weight, and engine fuel consumption. However, these UAVs are generally more expensive to operate due to their reliance on gasoline-based fuels and have lower movement capabilities than rotary-winged UAVs.

Overall, the classification of UAVs into rotary-winged and fixed-winged types allows for a better understanding of the applications and limitations of each type. This knowledge can aid in selecting the appropriate UAV for a given application and promoting the efficient and safe operation of UAVs. Figure 2.1 provides examples of different UAVs and payloads.

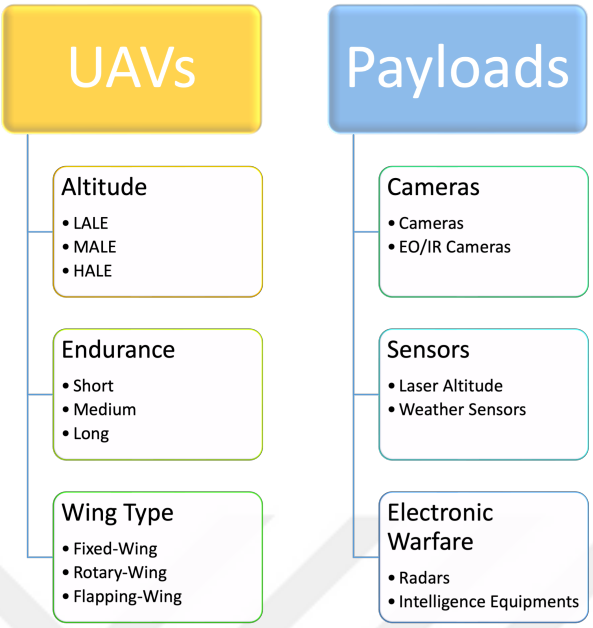

Figure 2.1 UAVs and Payloads

2.1.1. Rotary-Winged UAVs

Rotary-wing UAVs are often preferred in closed areas such as warehouses due to their small size and high maneuverability. Additionally, their vertical takeoff and landing capabilities make them safer for use in rural areas than fixed-wing UAVs. These drones can hover in very low altitudes, even holding their position in the air at altitudes less than 1ft, making them precise in their movements. These features make rotary-winged drones popular among general civilian users and companies for their various uses.

While rotary-winged UAVs have the advantage of small size and high movement capability, they have a lower endurance than fixed-winged UAVs. This endurance loss is mainly due to the high power consumption required to keep the drone in mid-air, with varying power consumption during takeoff and landing phases. As a result, the capabilities of rotary-winged UAVs are limited to shorter distances and lower altitudes. These drones typically use electricity as their power source, which presents a challenge regarding battery size and endurance. If battery size is increased to improve endurance, the drone's weight increases, resulting in higher power consumption.

Moreover, rotary-winged drones have issues with communication modules; with their small size, rotary-winged drones cannot carry big payloads or avionic components. That makes communication range limited for this type of UAV.

Due to their high maneuverability, rotary-winged UAVs can be used in swarm systems, where multiple drones work together towards a common goal. This can be achieved by having all the drones in the swarm perform the same task or by dividing the tasks among the drones.

2.1.2. Fixed-Wing UAVs

Fixed-wing UAVs, on the other hand, have lower movement capabilities but higher endurance and payload capacity. They can carry heavier loads at altitudes ranging from 1,000 to 50,000 feet. Unlike rotary-winged UAVs, the movement of fixed-wing UAVs is controlled by control surfaces without the need to adjust engine thrust, making them similar to commercial aircraft in their flight system.

Due to their larger size, fixed-wing UAVs can carry heavy equipment and larger avionics. Popular payloads for fixed-wing UAVs include EO-IR cameras, which can zoom in from 5-10km distances and have features such as IR cameras. For communication, a SATCOM module can expand the communication distance to the entire world. For military purposes, payloads such as missiles, bombs, or electronic warfare equipment can be installed on a fixed-wing UAV.

Their endurance differs from type to type. The aerodynamics of UAV, weight, and engine fuel consumption make endurance vary for the main reasons. A fixed-wing UAV's endurance can expand to 120 hours below the stratosphere. If it's a stratospheric UAV, endurance up to 25 days is possible, [9]. Long endurance makes fixed-wing UAVs feasible for military operations or continuous surveillance of an area, preferably a border or emergency zone. In hard terrain, UAVs come out for support in different ways, like surveillance or radio relay to communicate with an area.

Fixed-wing UAVs are generally more expensive due to their reliance on gasoline-based fuels. Their low movement capability makes them unsuitable for certain missions that rotary-wing UAVs are better suited for. On the other hand, they have the advantage of higher endurance and payload capacity. Fixed-wing UAVs are capable of carrying a variety of payloads, including data acquisition hardware equipped with complex signal processors and avionics.

2.1.3. UAV Operations

Unmanned Aerial Vehicle (UAV) systems operate with several main subjects for pilot or user interaction. Given that UAVs fly without an onboard pilot, the entirety of the flight operation must be executed from a Ground Control Station (GCS). A UAV system encompasses a GCS and one or more UAVs connected to the GCS. The GCS must maintain a robust connection with the UAVs, as numerous critical operations must be executed without data loss.

UAV communication is facilitated via uplink and downlink connections, essential for effective air-ground communication. These connections employ several protocols, including MAVLink, MQTT, STANAG-4586, and STANAG-7023 messaging protocols. These protocols are widely used for military and non-military purposes, encapsulating data under the transmission protocol and instituting their flow controls. The downlink direction refers to data transmission from the UAV to the GCS, while the uplink direction indicates a message from the GCS to the UAV. Loss of communication is referred to as link loss.

Operations can be classified based on their criticalness. Flight-critical data or operations are typically considered critical due to their potential impact on the UAV's operation. The most critical operations for UAVs generally are those related to flight, as any issue with flight-related operations can potentially result in a crash, given the absence of an onboard pilot. The priority is to ensure that flight-critical data is reliably received from the UAV and can be accurately displayed by operators in the GCS without any issues, such as delay or invalid data. Most protocols employ checksum methods for data validation to mitigate the risks associated with invalid data, making these protocols widely used in civil and military UAV systems.

To explain data types that are crucial included in our model, stated as follows:

- **UAV Status:** UAV Status is a downlink information package that includes the current location of the aircraft, acceleration, and angles in axes, link health, fuel data, etc. In contrast, UAV States transfer flight-critical data to real-time aircraft in real-time. This data is crucial because the pilot operates aircraft regarding those parameters. Uplink packages of that information are retrieved by controllers and sent to aircraft with the highest importance to provide real pilot cockpit inputs to aircraft directly. The uplink package is not considered in our model.
- **Mission Status:** Mission Status is also a downlink package that informs the pilot of the current state of UAV inside a mission plan. Since the mission plan is a big bundle, values in the package slightly change in time.
- **Payload Information:** Payload information is also a downlink package that delivers payload-specific information from UAV to GCS. Payloads are very differentiable, a payload can be a temperature sensor, EO-IR camera, etc. Payload data is critical for operators since camera payloads become an eye for operators. Payloads make UAVs operate for a purpose or many purposes. Such that, payloads and their information are important for UAV systems and one of base packages in air-ground communication.
- **Mission Plan Upload:** Mission plan upload is an uplink package that sends a mission plan to UAV from GCS. A mission plan in unmanned aerial vehicle (UAV) systems outlines the strategic approach and objectives for a UAV operation. It encompasses various key elements to ensure a successful mission. Firstly, the plan defines the mission's purpose, such as reconnaissance, surveillance, or delivery. It specifies the area of interest, target locations, and desired outcomes. The plan also includes pre-flight preparations, including airspace clearance, weather analysis, and equipment readiness checks. Additionally, it outlines the UAV's flight path, altitude, and waypoints to optimize efficiency and safety. The plan incorporates risk assessment and contingency strategies to address potential hazards or emergencies. It also

details communication protocols, data collection parameters, and real-time procedures. Finally, the mission plan includes post-flight activities, such as data analysis, reporting, and maintenance. By comprehensively addressing these elements, a well-structured mission plan ensures the effective and reliable execution of UAV operations. Mission plan upload is very crucial for UAVs. Lack of a mission plan generally causes UAV flight makes it risky; if pilot control becomes discarded due to any problem such as link loss, UAV should continue its flight regarding its mission plan. Mission plan upload operation is generally a one-time operation but in the middle of a flight, the mission plan can be uploaded again to update the mission plan or change the mission plan inside the UAV.

- **Command Upload:** Command upload is an uplink package that sends a command to UAV from GCS. Command can be considered as a part of mission plan that command includes location information and payload-specific operation, a flight-specific operation to execute a command. The pilot performs command execution and uploads in case of necessity.

- **Task Transfer:** Task transfer stands for objectives that create of the task, owning a task, and completion of a task. Task transfer means if an UAV system is not capable of processing payload information or needs other information to process data, UAV creates a task and sends it to a capable system. When data is processed, the output of data is sent back to the creator of the job.

2.2. Blockchain Technology

Blockchain structures can be classified into public, consortium, and private. Public blockchains allow anyone to access the chain and see all transactions, often used in cryptocurrencies. In contrast, Consortium blockchains have limited access to multiple groups or organizations. Finally, Private blockchains limit access to a group or organization of users, and transactions are only visible to them. Access to these blockchains is often invitation-based [1].

A summary of their capabilities and features is presented in Figure **??**.

Property	Public blockchain	Consortium blockchain	Private blockchain
Consensus determination	All miners	Selected set of nodes	Within one organization
Read permission	Public	Public or restricted	Public or restricted
Immutability level	Almost impossible to tamper	Could be tampered	Could be tampered
Efficiency (use of resources)	Low	High	High
Centralization	No	Partial	Yes
Consensus process	Permissionless	Needs permission	Needs permission

Figure 2.2 Types of blockchains.[1]

2.3. Layered Architecture of Blockchain

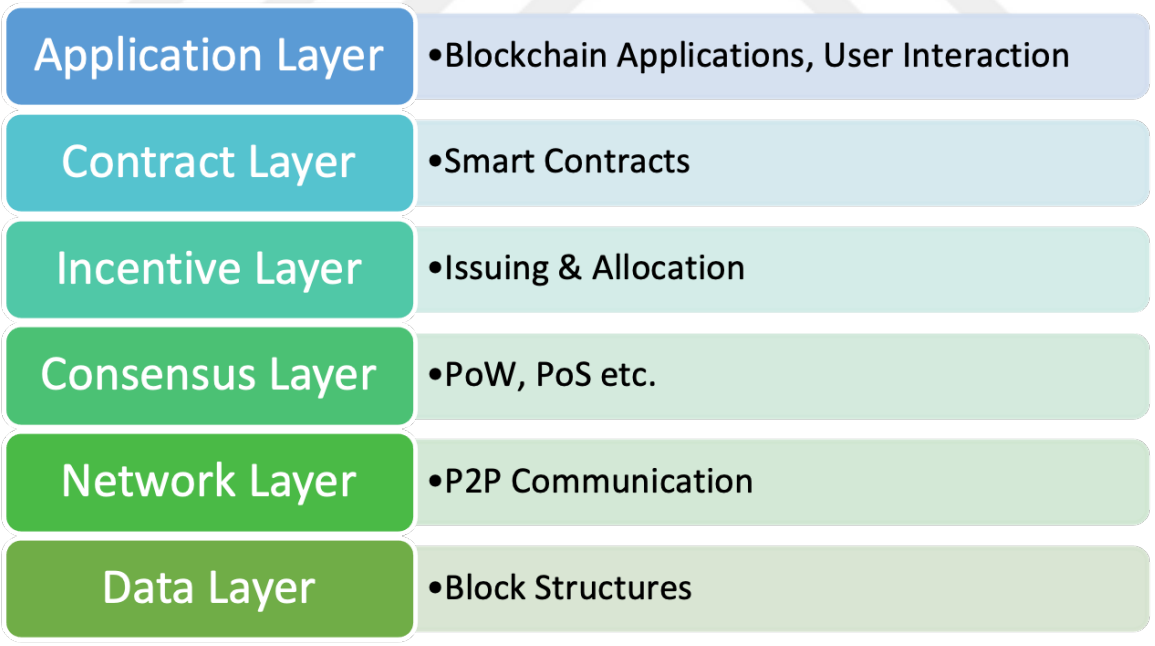

Figure 2.3 Blockchain in layers

Blockchain architecture can be understood through a layered approach [10] which is given in Figure 2.3, with the *Data Layer* containing core elements such as data blocks, hashes,

and Merkle trees. The *Network Layer* handles communication and verification mechanisms. In contrast, the *Consensus Layer* involves protocols for accepting or rejecting blocks and is implemented through various consensus algorithms such as Proof of Work (PoW) [11], Practical Byzantine Fault Tolerance (PBFT) [12], Proof of Stake (PoS)[13], Tendermint [14], Ripple [15]. The *Incentive layer* provides rewards for mining operations. The *Contract Layer* holds information about deployed smart contracts; detailed information is given under *Ethereum and Smart Contracts section*. The *Application Layer* allows users to interact with the blockchain through various libraries and models, such as those utilized in Ethereum [10].

The architecture of blockchain can be analyzed through a layered approach [10] which is given in Figure 2.3. The first layer is called the *Data Layer* and contains core elements of blockchain structure such as data blocks, hashes, and Merkle trees. The Merkle root is where changes in a block occur with each transaction over the Merkle tree. The *Network Layer* follows the data layer, which includes communication, verification mechanisms, and a peer-to-peer network. This layer is responsible for distributing, forwarding, and authenticating blockchain transactions. One of the most critical layers of the blockchain is the *Consensus Layer*. It involves protocols to accept or reject which blocks are created by whom and who is responsible for obtaining new blocks. There are various implementations of consensus protocols, such as Proof of Work (PoW) [11], Practical Byzantine Fault Tolerance (PBFT) [12], Proof of Stake (PoS)[13], Tendermint [14], Ripple [15], among others. Smart contracts, applications, and programs come out over these layers, the layer in which we utilize blockchain technology as end-users. *Incentive layer* provides a concurrency mechanism for blockchains. This layer also provides prizes for mining operations. *Contract Layer* holds information about loaded smart contracts deployed. Detailed information is given under *Ethereum and Smart Contracts section* which are scripts and algorithms used for flexibly programming the blockchain system. Specific programming languages, such as Solidity and Serpent, have been developed to enable manipulation of the blockchain with features like flow controls and other expandable functionalities. The *Application Layer* of blockchain technology allows users to interact with the blockchain through various Web3 and blockchain interactive libraries and models, making it possible for them to use blockchain

technology without being aware of it. Numerous examples of applications operate at this layer [10].

Regarding Figure 2.3, the first four layers of blockchain, containing the Data Layer, Network Layer, Consensus Layer, and Incentive Layer, form the basis of blockchain. Since Bitcoin was introduced, these four layers have been used for the initial blockchain architecture.

2.4. Blockchain Components and Characteristics

Blockchain, like any other computer structure, consists of multiple components. As seen in Figure 2.4, the core components of blockchain operate through several phases.

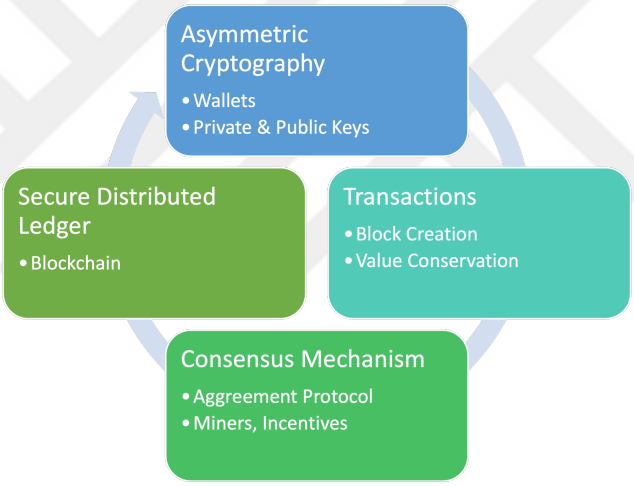

Figure 2.4 Core components and phases of blockchain

Although blockchain appears to be a complex system for holding data, its complexity provides enhanced security and the ability to operate decentralized. In addition, blockchain's capabilities are not limited to cryptocurrencies. Recent developments such as Non-Fungible Tokens (NFTs) and Metaverse projects highlight the diverse range of potential blockchain applications. Our paper focuses explicitly on the usage of blockchain in the UAV and IoT fields. With its desirable characteristics, particularly regarding security and decentralization, blockchain has become an appealing option for various technological domains.

As seen in Figure 2.5, we have summarized the characteristics of blockchain below [8]:

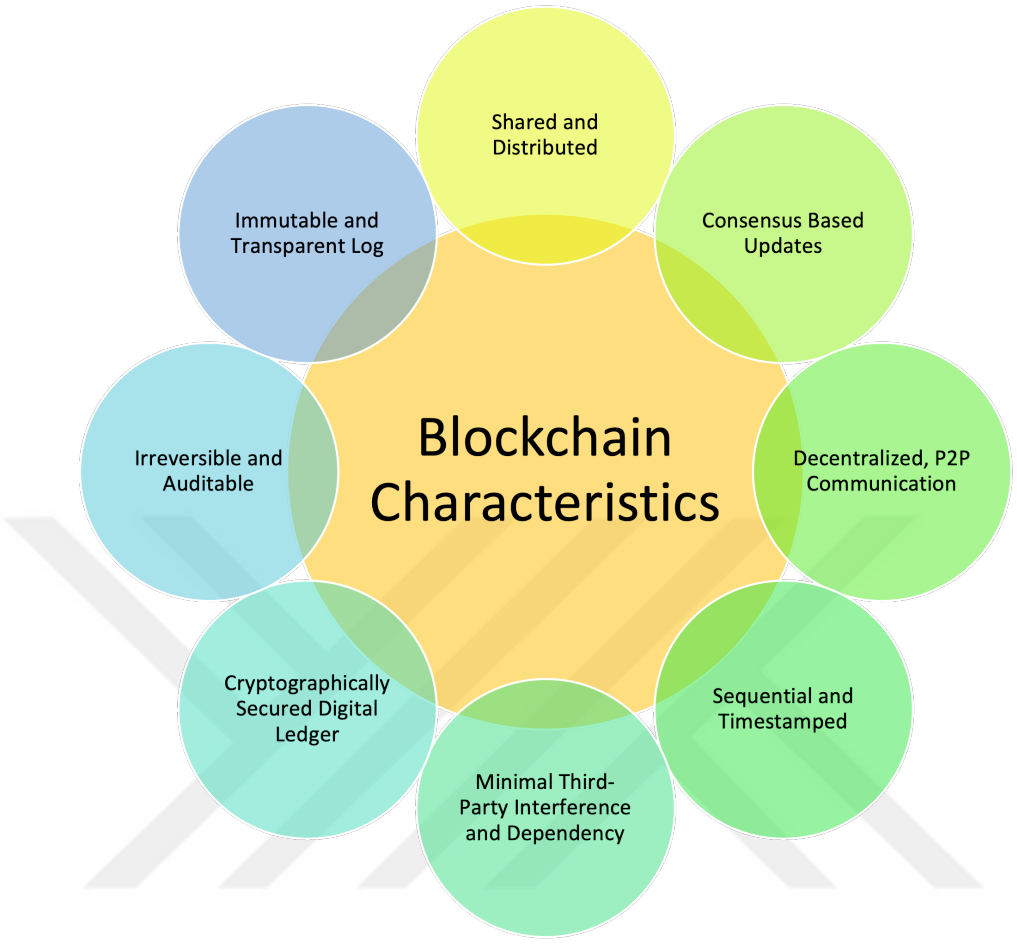

Figure 2.5 Characteristics of Blockchain

- *Shared and Distributed*: The data in a blockchain is distributed among nodes in the network, with every node having up-to-date information. This characteristic ensures the redundancy and reliability of the system.

- *Consensus-Based Updating*: Updates to the blockchain require the approval of a consortium of nodes, ensuring the system's security and integrity by preventing tampering.

- *Sequential and Timestamped*: Each block in the blockchain has a timestamp, and new blocks are added to the end of the chain in sequential order.

- *Minimized Third-Party Interference and Dependency*: The blockchain is an independent decentralized network that can be accessed from any device and has its

own programming language under the Ethereum Virtual Machine. This feature reduces dependency on third-party systems.

- *Cryptographically Secured Digital Ledger*: Asymmetric cryptography methods are used for all transactions, making it difficult to make invalid or tampered transactions.
- *Irreversible and Auditable*: Transactions in the blockchain cannot be edited or deleted, ensuring that the transaction history is preserved safely and securely.
- *Immutable and Transparent Log*: The blockchain is immutable, meaning that block data cannot be changed once recorded. All operations can be viewed in public blockchains, increasing transparency and confidence in the system.
- *Decentralized, P2P Communication*: Decentralization enables an application to operate on multiple nodes in an extensible network, making it more resilient and responding faster under heavy loads.

2.5. Advantages of Blockchain

Due to the inherent characteristics of blockchain, the developed system can easily expand by adding new nodes to the network. All information stored in the blockchain is automatically shared with connected nodes, and all transactions are immutable and secure. Blockchain also logs all data with its structure, which is important for sharing all data with others through a simple transaction. Decentralization makes the system robust in case of possible problems in nodes. Each node is created independently, so powerful and less powerful systems can run together on the same blockchain. For instance, heavy tasks such as mining or signing operations can be performed by powerful nodes, while light nodes such as fog-ends or IoT devices can make simple transactions. Supporting blockchain with specialized endpoints can create an IoT processing environment for edge systems, where heavy nodes can be considered servers or edge nodes.

2.6. Ethereum and Smart Contracts

Ethereum, developed after Bitcoin, introduced a different validation method and data storage technique. Unlike Bitcoin, Ethereum uses code blocks for data storage, enabling the execution of code on the blockchain and decentralized processing. Ethereum also introduced the concept of smart contracts, which are code snippets stored and executed on the blockchain. A gas fee is required to deploy or interact with smart contracts, which varies based on execution time and job size. Smart contracts are written in Solidity, a programming language specialized for the Ethereum Virtual Machine (EVM). They are accessible to all wallets on the chain, allowing for the development and use of protocols and models already deployed into the network.

Ethereum was developed after Bitcoin, using different validation methods and a distinct data storage technique. In contrast to Bitcoin, Ethereum uses code blocks for data storage. While Bitcoin holds transactions, Ethereum combines transactions with code. This unique structure enables Ethereum to execute code on the blockchain, leading to decentralized processing. Ethereum also introduced the concept of smart contracts used to perform predefined operations. These contracts are run by nodes in the Ethereum Virtual Machine (EVM), marking a new era in blockchain technology.

Smart contracts are code snippets stored and executed on the blockchain. Since smart contracts contain runnable code, executing heavy code can strain the blockchain. The transaction sender must pay a gas fee for the operation to deploy or interact with smart contracts. The gas fee varies based on the execution time and the job site. For example, loops with large amounts and data storage for big objects require more gas than simple operations. Smart contracts are typically written in Solidity, a programming language specialized for the Ethereum Virtual Machine (EVM), which holds and executes smart contracts. A contract is accessible to all wallets on the chain and can be used as a library for other contracts. This flexibility allows for developing and operating protocols and models already deployed into the network.

Go-Ethereum is an application layer designed to create, run and manage blockchain operations. It allows for the easy creation of private Ethereum nodes. It provides several functionalities, such as initialization, account creation and management, connectivity with other nodes, and monitoring of specific items on the blockchain.

Truffle is another application that enables interaction with smart contracts. It can compile Solidity contracts and identify errors in them. Truffle makes compiled contracts and contract ABI accessible. Truffle can connect to the blockchain with a simple configuration and deploy smart contracts using a specified account. Using Truffle simplifies the development and deployment of contracts. The combination of smart contracts and IoT devices in Ethereum creates a secure, decentralized data transfer and storage solution. The usage of Ethereum for IoT devices also enables nodes to run on different systems. It allows for connecting systems that can process data or interact with an Ethereum node as a user with no workload.

Smart contracts are used for data storage. Data may be stored separately for swarm systems and may store for everyday use. Other nodes and endpoints are using the stored data. A private Ethereum network will be utilized for UAV data, such as mission plans, flight, and payload data. A task transfer scheme will be informed with more detail in our model.

3. RELATED WORK

The literature review conducted for this thesis extensively covered research related to the integration of IoT and edge computing with blockchain technology. Several studies were analyzed to examine the potential benefits and challenges associated with combining these technologies. The review focused on understanding how IoT and edge computing can be effectively integrated with blockchain to enhance the capabilities of UAV systems. Additionally, it explored the concept of GCS as edge systems for in-situ data processing and operation. By synthesizing the existing academic knowledge on IoT, edge computing, and blockchain, the literature review provides a solid foundation for our thesis, establishing the academic context and supporting the relevance of our research in the field of UAV systems.

Alladi et al. [16] provided a review contains information about six major usages of Application of Blockchains with UAVs. The paper covers each usage with the motivation, role of blockchain, system, and challenges under the influence of usage. In the end, the paper gives 2 emerging topics as broader perspectives. The important section of this paper for us is ”UAV Networks for Edge Computing” which include assuming UAVs as Mobile Edge Computing system.

Islam et al. [17] introduced a secure data collection scheme and collected data from IoT devices with a UAV.They stored this data in blockchain at the mobile edge computing server.

Xu et al. [18] with their work provided a schema for resource pricing and allocation to solve this problem in edge networks. They formulated the problem as a Stackelberg game where the leader is ECS and the followers are UAVs. They have shown that ECS can achieve optimal edge computing resource pricing where resources can be optimally allocated among UAVs. They solved the security problem in a schema with blockchain.

Lagkas et al. [19] reviewed the latest UAV application areas enabled by IoT and 5G. Analyzed the sensor requirements, and overviewed the solutions for fleet management over aerial networking, privacy, and security challenges. Then they proposed a framework of IoT

architecture that supports and enables these technologies on UAVs. Their framework enables the protection of UAVs as flying things in a collaborative network environment.

Li et al. [20] proposed solutions for Machine Type Communication Devices achieve problems in network services if IoT infrastructure gets destroyed. To provide solutions, they used UAVs, blockchain, and mobile edge computing to ensure data transmission, security, and reliability in damaged M2M networks. They also proposed a joint optimization framework to maximize both the data communication capacity and throughput of blockchain systems.

Sharma et al. [21] focused on ultra-reliable communication in MEC. They used drones as on-demand nodes for caching. They presented a novel neural-blockchain-based drone caching approach, designed to ensure ultra-reliability and provide a flat architecture. The neural model fortifies an efficient transport mechanism since blockchain maintains high reliability amongst the peers involved in the communications.

Khan et al. [22] provided a data management and monitoring framework with a blockchain-aware framework. Their main interest is about analyze blockchain storage changes. Data flows through a fog network. They used smart contracts for analyzing fog-cloud-based stored information. After analysis ran for stored and new data, using differences will yield the surface changes.

Aloqaily et al. [23] covered research guidelines for 5G UAV Networks with parameters speed, reliability, and security. Their targets are smart cities, they used drones as a service. With the aid of blockchain, UAVs can act as mobile access points, routing entities, or resource providers in a decentralized manner.

Luo et al. [24] created a new architecture for automatically offloading user tasks in MEC scenarios for challenges about MEC services coverage limitations and problems in the audit trail of which MEC processed the data. In their solution, they use drones to dynamically cache data generated from IoT devices and send them to MEC servers in the private blockchain network.

Chao et al. [25] proposed a UAV network for transmitting air traffic data. For security and network architecture, they used blockchain.

Kuzmin et al. [26] provided a concept for UAVs using blockchain. Each UAV in their proposed UAVNet is a Blockchain node, has onboard functionality for creating and reading transactions from the block, as well as communication tools for exchanging transactions with other UAVs.

Raj et al. [27] utilized the blockchain to pick up people's health data which is saved on the furthest server system by unmanned aerial vehicle (UAV) is introduced. UAV communicates with the body sensor hives (BSH) low-powered and safely using a token. UAV, which solves HD's password in favor of a known key, makes an identity validation fitting system that has two steps. If this is confirmed, data can transmit to the furthest server, and the proposed healthcare methodology can be analyzed with some methods, such as making a simulation and observing the work's efficiency and performance. At the end of the analysis, the proposed work assists well in a secure environment.

The following table represents the literature review in format with topics, method summary, and summary of works stated above. Table 3.1 includes works primarily on blockchain-supported IoT systems. Most of the works include data acquisition schemas and data transmission via blockchain. For task transfer, works with edge computing give background information.

Table 3.1 Related Works of Blockchain Assisted Systems

Reference	Name	Topic	Purpose	Key Finding
[16]	Applications of Blockchain in Unmanned Aerial Vehicles: A Review	UAV Networks, Security and Privacy, Blockchain, IoT	Blockchain Applications	Paper contains information about 6 major usages of Applications of Blockchains.
[17]	BUAV: A blockchain based secure UAV-assisted data acquisition scheme in Internet of Things	Blockchain, IoT, UAV, Mobile Edge Computing, Security	Data acquisition with blockchain.	Secure data collection scheme.
[18]	Edge Computing Resource Allocation for Unmanned Aerial Vehicle Assisted Mobile Network With Blockchain Applications	Edge Computing, IoT, Blockchain, UAV, Resource Pricing, Resource Allocation.	Edge Computing with Blockchain.	Achieve optimal edge computing resource pricing among UAVs secured with blockchain.
[19]	UAV IoT Framework Views and Challenges: Towards Protecting Drones as “Things”	Security, Privacy, UAV, IoT.	Distributed Fleet Management	Proposed a framework of IoT architecture that supports and enables aerial networking, privacy, and security on UAVs.
[20]	UAV-Assisted Data Transmission in Blockchain-Enabled M2M Communications with Mobile Edge Computing	Mobile Edge Computing, UAV, IoT, Blockchain, Data Transmission, M2M.	UAV Data Transmission	Making data transmission for damaged networks with UAVs secured with blockchain.
[21]	Neural-blockchain-based ultra-reliable caching for edge-enabled UAV networks	Caching, Edge Computing, UAVs, Drones, Ultra reliability, Blockchain, Neural Networks.	Reliability on UAV-Supported Networks	Using neural blockchain for caching on UAVs.
[22]	Blockchain-Aware Distributed Dynamic Monitoring: A Smart Contract for Fog-Based Drone Management in Land Surface Changes	Blockchain, Smart Contracts, Fog-Cloud Computing, UAV, Remote Sensing, IoT, Urban Landing.	Data Management on UAV Fog Nodes	Blockchain-supported data management framework.
[23]	Design Guidelines for Blockchain-Assisted 5G-UAV Networks	UAV, Blockchain, 5G, QoS	Research Directions for 5G UAV Networks	With the aid of blockchain, UAVs can act as mobile access points, routing entities, or resource providers in a decentralized manner.
[24]	Blockchain-Based Task Offloading in Drone-Aided Mobile Edge Computing	Blockchain, UAV, Mobile Edge Computing, Task Offloading	Task Offloading	Using UAVs to collect tasks from IoT with blockchain.
[25]	UAV Traffic Information Exchange Network	UAV, Blockchain, UAV Networks	Network Architecture with Blockchain	UAV network for transmitting air traffic data with security and network on blockchain.
[26]	Blockchain-base structures for a secure and operate network of semi-autonomous Unmanned Aerial Vehicles	UAV, UAV Networks, Blockchain, Integrity, Consensus Mechanisms, Cyber Security	Blockchain Supported UAV communication.	Concept for UAVs using blockchain for communication.
[27]	Security Enhanced Blockchain based Unmanned Aerial Vehicle Health Monitoring System	UAV, Blockchain, Healthcare, IoT, Security, Body Sensors	UAV Data Transmission	Collecting data from health services with UAV’s support of blockchain and transferring.

Integrating blockchain technology with UAVs has been a topic of interest for several studies and research works aimed at enhancing their efficiency and overall performance. These studies have shown that blockchain technology can improve the functionality and capabilities of UAV systems. Different operations in UAV systems can be handled or supported with the help of blockchain technology. Recent studies have examined the potential applications of blockchain technology in UAV systems, which serve as a basis for our work in this domain. In contrast to the given works following works are much more related to our model.

In blockchain technologies, a more complex structure is required beyond that of Bitcoin, which is solely designed for financial transactions. Ethereum is a robust and dependable blockchain-based software platform that enables more complicated operations. In a review of Ethereum by Zou et al. [28], the capabilities of the Ethereum network, explanations of smart contracts, the architecture of Ethereum, and sample application scenarios, are investigated.

Guan et al. [29] presented blockchain technology as a decentralized remedy for UAV-enabled mobile edge computing. They proposed a blockchain-based architecture within UAVs to establish a model of mutual trust, impartiality, transparency, and steadiness as a contingency solution in case the ground network fails. The integration of blockchain technology allows for the transparent, immutable recording of device computing capacity, task allocation, and task execution processes. They employed smart contracts to ensure the publicity of algorithms.

Islam et al. [30] proposed a scheme for secure data collection from IoT devices using a UAV swarm, in which the collected data is stored on the nearest mobile edge server via a smart contract included in the blockchain. To ensure effective communication, the UAVs self-define Theication and create a shared key to facilitate communication before initiating the data acquisition mission. After completing the mission, the server creates a block and requests permission from other validators. The data is added to the blockchain only after receiving consent from all validators.

Different than existing studies in the literature, our system primarily focuses on distributing mission plans and commands to UAVs in a swarm. By utilizing blockchain technology, the

system can achieve faster integration of new UAVs and eliminates the need for a master drone, as all nodes share the same command or mission plan. Moreover, the system also includes a task transfer mechanism using blockchain, enabling automatic task assignment and simplifying data collection through querying the entire UAV network for necessary data.

4. PROPOSED METHOD

This section explains a new data management model using blockchain technology for swarm UAV systems. The proposed model offers a blockchain-based architecture to manage data sharing, command and mission plan uploading, and task transferring between swarm nodes and GCSs. The model provides sharing commands and mission plan for all nodes in the swarm system without the need for any master node in the system. Task transferring operations are useful for heavy jobs where swarm node performance is not enough. Data-sharing operations will support task transferring and provide a mission-tracking endpoint for GCSs.

Common communication protocols used in UAVs, such as MAVLink and STANAG, are capable of fulfilling those operations. However, enhancing those operations with blockchain automatically synchronizes all data, processes, and transactions between nodes, making all operations irreversible, logged, and robust against tampering. In UAV systems, strong tamper protection provided by blockchain is crucial. Automatic synchronization makes the system available to share data between nodes in a given format in smart contracts without updating any communication scheme.

Figure 4.1 illustrates our proposed model where all nodes are connected with the blockchain, the proposed model's general flow is given, and the processes are visualized. The model is developed for 802.x communication assumed as already deployed in UAVs and GCSs. The physical structure of the 802.x protocol may vary between endpoints. GCSs can be connected via cables, while UAVs can communicate with GCSs via RF, LOS, SATCOM, 5G, and other methods. Moreover, swarm communication can also be provided with a relay from a UAV. In the model, all communication infrastructure is assumed as deployed and capable of 802.x communication. The communication scheme also shows swarm nodes connected with a mesh.

To explain communication details, 802.x protocol is used along with common transfer protocols such as UDP and TCP. UDP is used for broadcast messages mainly for discovery,

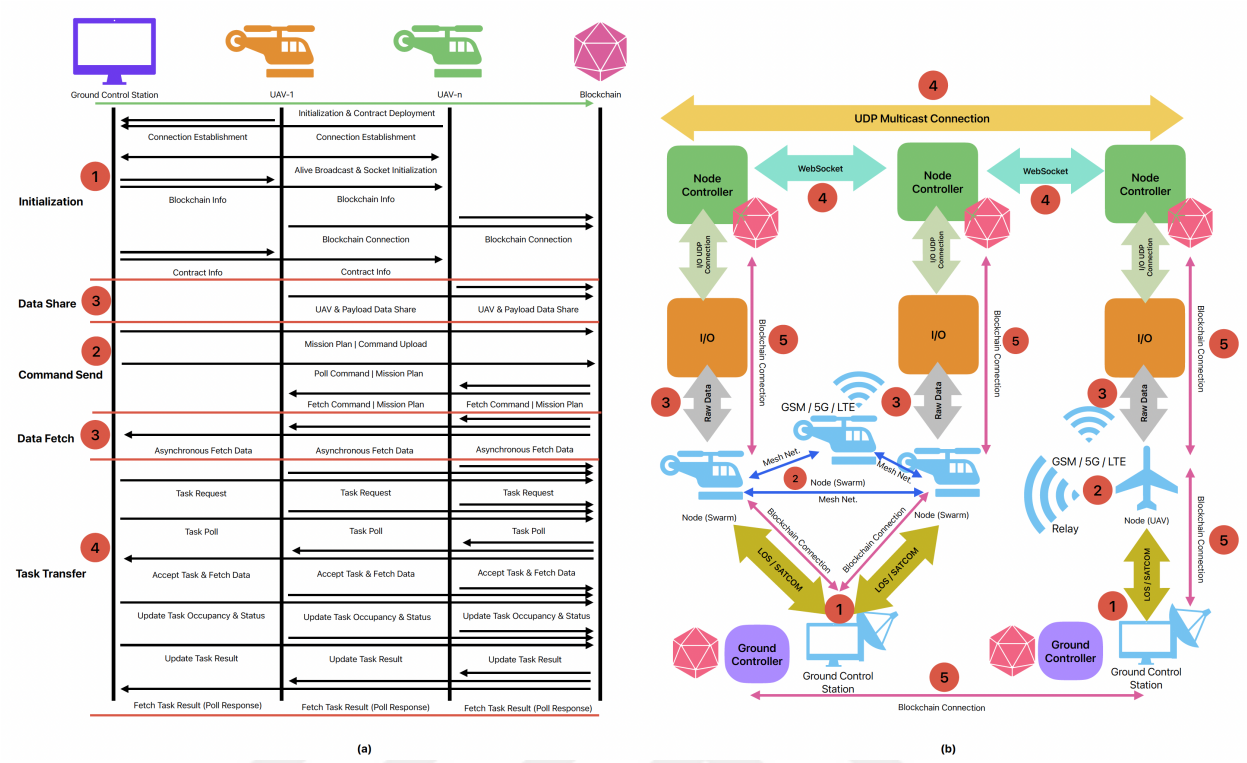

Figure 4.1 The Proposed Blockchain-based Decentralized Model for Swarm UAVs a) Execution Orders of Operations, b) Communication Scheme

while TCP is employed in WebSockets between nodes connected with mesh communication created with information received from broadcast messages. It provides communication for notification messages and complex data transfers. Message payloads will be in JSON format to make parse operations easier.

Figure 4.1 (a) shows the flow of operations with directions. The modules responsible for handling the operations indicated in Figure 4.1 (a) are described in the communication scheme depicted in Figure 4.1 (b). Directions, sources, and targets of data are given in Figure 4.1 (a) with process order. The green UAV is considered as any other UAV in the system. Modules in the communication scheme hold structures and manage such operations given below.

Every operation given in the schema will be provided in details in the following sections.

- *Blockchain & Smart Contracts (Pink):* Mission Plan, Command, UAV States, Mission Track States, Payload Data, Tasks
- *Node Controllers (Green):* Blockchain Interaction, UDP Messaging, WebSocket Messaging, Flow Control, Running a Node, I/O Receiver, Request Management
- *I/O Module (Orange):* I/O Modules, Sensors, Preprocessors
- *Ground Controller (Purple):* Data Viewer, Mission Tracker, Mission Planner, Blockchain Initialization

In Figure 4.1, the following sub-sections give information about the main parts in the flow diagram with execution order which is shown in Figure 4.1(a) with components given in Figure 4.1(b). Our proposed model consists of four main parts of operations, which are the *Initialization* phase, *Data Share & Fetch* operations, *Command and Mission Plan Upload* operations, *Task Transfer*.

4.1. Structures and Modules

To explain the model, in this section, we break down each module and structure with their relations and connections between them. In common, all modules use configurations to discriminate their operational capabilities and create interfaces using parameters in configuration with 3rd party applications or between modules.

The system depends on many connection modules and corresponding blockchain structures for messages received from the I/O module. Modules are responsible for controlling the flow of operations, handling and sending messages, and making interactions with blockchain.

We have three configurations which are:

- Main Node Configuration: Configuration includes connection details, node details, and contract addresses for blockchain.

- Genesis File: Configuration for genesis block of blockchain.
- Truffle Configuration File: Configuration for Truffle that makes deployment of smart contracts.


```
{
 "uavId": "G000",
 "nodeName": "Node",
 "mode": "prod",
 "isGCS": true,
 "paths": {
  "basePath": "./",
  "nodePath": "Node_G000",
  "trufflePath": "./truffle/"
 },
 "config": {
  "networkInterface": "lo0",
  "ip": "127.0.0.1",
  "port": 30330,
  "httpPort": 30331,
  "ws_port": 30332,
  "multicastAddress": "230.100.100.100",
  "multicastPort": 30303
 },
 "chain_config": {
  "chainId": 2223141,
  "identity": "uavTestNet",
  "defaultAccount": "0x0a8Dc09CaC642ecC3Ce602e7F45a2a7416D07965",
  "etherbase": "0x0a8Dc09CaC642ecC3Ce602e7F45a2a7416D07965",
  "etherbase_seed": "G000",
  "etherbase_privkey": "0x8775675b15c8b6ec0f65f2a6a4e44e7cedc2f9b21cc19160fc2ff9ba25d9d48a"
 },
 "nodeInfo": {
  "enode":
"enode://a9eab3d3439776d1c38d83114728cd23dce546b3001358d3c10e883869b8ea7f992fefc67b6bb6704789dfcea142502d7c8d
e268d5977e375b4afdab90c663ed@127.0.0.1:30330?discport=0"
 },
 "contracts": {
  "MPStates": "0xA5057b0B2181cbB0e97e372122D735AaB0013eC0",
  "UAVStates": "0xE80293d4D6fEe43F3E5f4fDCcf1C41032B1582C8",
  "TaskContract": "0xA83cad023523725b1934c0a919eB7d3347B9eA8a",
  "CommandContract": "0xBe692BC4F92Db081E4479a6E773279E36fF3dd6c",
  "MissionPlanContract": "0x9F8E637fbdCD125f8345CeaF1Afee82020bF730c",
  "PayloadData": "0x7150A74E3D733FAd778ecAaDDb23F03fae6BDc2e"
 },
 "timerPeriods": {
  "missionPlan": 5000,
  "command": 5000,
  "task": 5000,
  "alive": 1000,
  "sysAvailability": 5000
 }
}
```


Figure 4.2 Main Node Configuration

The main configuration structure is given in Figure 4.2. Fields in the configuration are stated below:

- *uavId* discriminates between nodes; GCS and UAV use the same field. Used as an address for nodes.
- *isGCS* is necessary for choosing a node for GCS. GCS node deploys and distributes contracts and addresses to nodes. GCS nodes run miners.

- *Ip and Port* parameters used for blockchains RPC communication socket.
- *httpPort* is used for the HTTP connection socket of blockchain.
- *wsport* is used for websocket connections.
- *multicastAddress and multicastPort* is used for registering to broadcast UDP messages.
- *chainId and identity* is used for initializing blockchain. chainId is also used in the genesis file.
- *defaultAccount and etherbase* stands for blockchain address of an initialized node, automatically assigned after initialization.
- *seed and etherbaseprivkey* values are derived from seed for a created account. uavId is used as a seed. Values are automatically assigned.
- *enode* is a connection address for blockchain to make nodes as peers. Automatically assigned on initialization.
- *Contracts* are automatically filled while the deployment of each contract with addresses. GCS fills fields on deployment. UAV nodes receive deployed addresses from GCS via message.

Genesis configuration structure is given in Figure 4.3. Fields in the configuration are stated below:

- *chainId* is used to discriminate blockchain network.
- *difficulty* value is used for setting mining difficulty to find nonce value by miners.
- *gasLimit* is used to dedicate the maximum gas payment amount for transactions.
- *alloc* field is filled by generated initial account with given ETH balance in gwei. Allocated account given in Figure 4.4

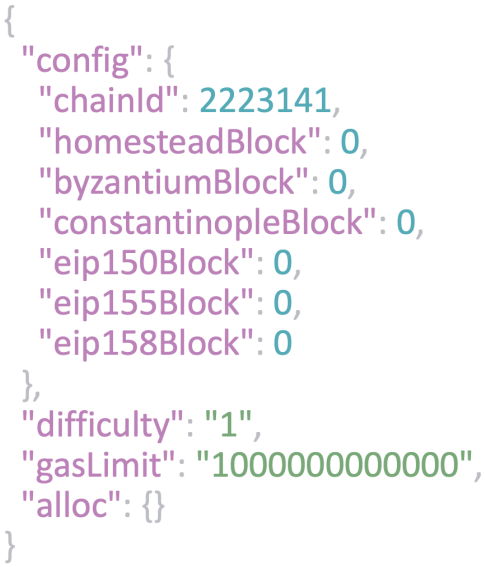

Figure 4.3 Genesis Block Configuration

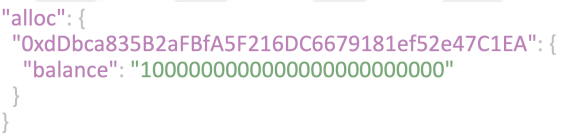

Figure 4.4 Genesis Account Allocation

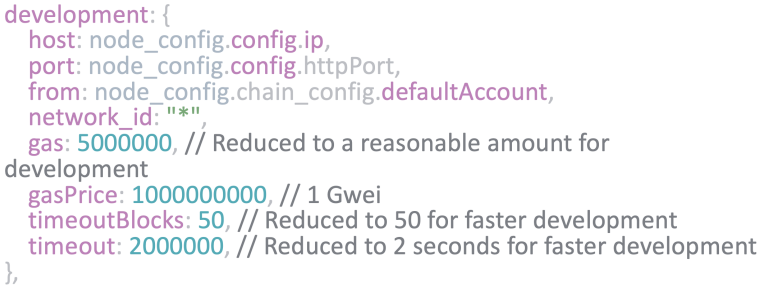

Figure 4.5 Truffle Contract Configuration

Genesis account allocation is given in Figure 4.4.

Genesis account allocation is given in Figure 4.5. Fields in the configuration are stated below:

- *host, port and from* are filled from a main configuration with blockchains RPC socket and from is defaultAccount that node's address to deploy contracts.

- Other parameters are necessary for contract transactions, states minimum gas and default gas sent for transactions.

Those configurations are necessary for connections between node controllers, blockchain initialization, and blockchain interaction, in further sections, usage of those parameters are explained in detail.

4.1.1. Blockchain Structures

Since the model shares data via blockchain, some parameters are shared in blockchain with connected nodes. To provide that, smart contracts for different data types were developed to interact with blockchain. Many different functions are used to share and fetch data. Functions can set all parameters at once, get all parameters at once or set and get parameters partially. There are six smart contracts used for data sharing and management which are: UAV States Contract, Mission States Contract, Payload Information Contract, Mission Plan Contract, Command Contract, Task Contract. In Mission Plan Contract and Command Contract, structures are shared instances for all UAVs in swarm, nodes checks if they are assigned to created command or mission plan.

4.1.1.1. UAV States Contract UAV States include flight critical information fields in Table 4.1. Information on contract may be populated. However, we choose the most necessary information that should be shared between UAVs and GCSs are included in contract. Other flight-critical informations such as axial accelerations and axial angles, are discarded.

The location struct given has int256 type values due to the lack of floating point numbers in blockchain. Latitude and Longitude values are set by multiplying the received location value in degrees with 10^{10}. When any value is requested from those fields, the node controller converts the received value by dividing it by 10^{10}.

UAV States		
Field	**Type**	**Description**
FlightStatus	enum	Current flight state of UAV. Such as NAVIGATION, HOLD, etc.
Location	struct	Current location of UAV in geospatial coordinates in degrees.
Location.latitude	int256	Current latitude of UAV in geospatial coordinates in degrees.
Location.longitude	int256	Current longitude of UAV in geospatial coordinates in degrees.
Location.altitude	int256	Current mean sea level altitude of UAV in meters.
uavCount	uint8	Current UAV count in the swarm.
UAVStateStruct	struct	UAV State struct.
UAVStateStruct.uavNo	uint8	Order of related UAV.
UAVStateStruct.uavId	string	uavId of related UAV.
UAVStateStruct.flightMode	FlightStatus	Current flight state of related UAV.
UAVStateStruct.location	Location	Current location of related UAV.
UAVStateStruct.linkHealthPercentage	uint8	Current link health percentage of related UAV.
UAVStateStruct.batteryPercentage	uint8	Current battery percentage of related UAV.
uavs	mapping($address \rightarrow UAVStateStruct$)	Maps UAV node addresses to UAVStateStructs that matches UAV nodes with their UAVStateStruct.

Table 4.1 UAV States Contract Fields

4.1.1.2. Mission States Contract Mission States include information about the current mission state of UAV. Mission state provides information about mission plan execution, which mission plan is loaded, which waypoint is current, which is next and other parameters. Fields are given in Table 4.2.

Mission States		
Field	**Type**	**Description**
FlightStatus	enum	Current flight state of UAV. Such as NAVIGATION, HOLD etc.
Formation	enum	Current formation of swarm which UAV in. Such as LINE, ECHELON etc.
PayloadType	enum	Current type of payload used on UAV. Such as CAMERA, LASER, etc.
PayloadAction	enum	Current action of the payload of UAV. Such as ACTIVATE, DEACTIVATE, etc.
Payload	struct	Current payload of UAV.
Payload.payloadType	PayloadType	Type of payload.
Payload.payloadAction	PayloadAction	Action of payload.
MPState	struct	Mission State struct.
MPState.mpid	uint16	ID of mission plan loaded.
MPState.address	address	address of related UAV.
MPState.uavNo	uint8	Order of related UAV.
MPState.uavId	string	uavId of related UAV.
MPState.currentWpId	uint16	Current Waypoint ID of related UAV.
MPState.nextWpId	uint16	Next Waypoint ID of related UAV.
MPState.flightStatus	FlightStatus	Current flight state of related UAV.
MPState.formation	Formation	Current formation of swarm which UAV in.
MPState.payload	Payload	Current payload of UAV
mpstates	mapping($address \rightarrow MPState$)	Maps UAV node addresses to MPState that matches UAV nodes with their MPState.

Table 4.2 Mission States Contract Fields

4.1.1.3. Payload Information Contract Payload Information includes payload-centric data that is stored in a serialized format with unpacking string and payload type. Node controllers have unpacked structures of payload data. Since payloads need integration, nodes

can easily alterated to parse different payload data. However, to store payload data, we use a structure given in Table 4.3.

Payload Information		
Field	**Type**	**Description**
PayloadType	enum	Current type of payload used on UAV. Such as CAMERA, LASER etc.
PayloadData	struct	Current payload of UAV.
PayloadData.payloadType	PayloadType	Type of payload.
PayloadData.data	string	Serialized data received by payload.
PayloadData.unpackString	string	Unpack string for serialized data.
mpstates	mapping($address \rightarrow PayloadData$)	Maps UAV node addresses to PayloadData that matches UAV nodes with their PayloadData.

Table 4.3 Payload Information Contract Fields

4.1.1.4. Mission Plan Contract Mission Plan contract is a key feature of the model. Blockchain is very efficient for decentralized mission plan upload for swarm systems due to swarm must operate regardless of a master UAV in the swarm. Mission plan is shared with all UAVs with which UAVs are assigned to it. UAV nodes periodically check the mission plan entity to follow changes in the mission plan. Fields of Mission Plan Contract is given in Table 4.4.

Mission Plan		
Field	**Type**	**Description**
FlightPattern	enum	Flight pattern for UAVs. Such as CIRCLE, RACETRACK, etc.
Formation	enum	Current formation of the swarm which UAV is in. Such as LINE, ECHELON, etc.
PayloadType	enum	Current type of payload which mission plan needs. Such as CAMERA, LASER, etc.
PayloadAction	enum	Current action of payload which mission plan needs. Such as ACTIVATE, DEACTIVATE etc.
Payload	struct	Payload to use.
Payload.payloadType	PayloadType	Type of payload.
Payload.payloadAction	PayloadAction	Action of payload.
Location	struct	Current location of waypoint in geospatial coordinates in degrees.
Location.latitude	int256	Latitude of waypoint in geospatial coordinates in degrees.
Location.longitude	int256	Longitude of waypoint in geospatial coordinates in degrees.
Location.altitude	int256	Mean sea level altitude of waypoint in meters.
Waypoint	struct	Structure with location and uav actions in location.
Waypoint.wpid	uint16	Identifier for waypoint.
Waypoint.location	Location	Location of the waypoint.
Waypoint.flightPattern	FlightPattern	Flight pattern that executed on waypoint.
Waypoint.formation	Formation	Formation that executed on waypoint.
Waypoint.payload	Payload	Payload and action of payload that executed on waypoint.
Waypoint.assignedUavs	address[]	Node addresses of UAVs assigned to waypoint.
MissionPlan.mpid	uint16	Identifier for mission plan.
MissionPlan.waypointCount	uint16	Count of waypoints in mission plan. Used for validation reasons.
MissionPlan.waypoints	Waypoint[]	Waypoints that included to mission plan.
MissionPlan.assignedUavs	address[]	Node addresses of UAVs assigned to mission plan.
missionPlan	MissionPlan	Mission plan loaded to swarm system.

Table 4.4 Mission Plan Contract Fields

The location struct given has int256 type values due to the lack of floating point numbers in blockchain. Latitude and Longitude values are set by multiplying the received location value

in degrees with 10^{10}. When any value is requested from those fields, the node controller converts the received value by dividing it by 10^{10}.

Mission plan contract has special functions for waypoint assignment. Assign to waypoint and unassign from waypoint functions makes UAVs assign themselves or being assigned by the operator in an automatized way by clearing or populating assigned sections of the waypoint. Those special functions make automatization on blockchain and make usage easier.

4.1.1.5. Command Contract Similar to mission plan, commands represent one action in a location. We can consider commands as a single waypoint of a mission plan. Despite mission plans are big objects that include many waypoints and actions, commands are sent to the system frequently to accomplish different immediate actions. Fields of Command Contract is given in Table 4.5.

Command		
Field	**Type**	**Description**
FlightPattern	enum	Flight pattern for UAVs. Such as CIRCLE, RACETRACK, etc.
Formation	enum	Current formation of swarm which UAV in. Such as LINE, ECHELON etc.
PayloadType	enum	Current type of payload which command needs. Such as CAMERA, LASER etc.
PayloadAction	enum	Current action of payload which command needs. Such as ACTIVATE, DEACTIVATE etc.
Payload	struct	Payload to use.
Payload.payloadType	PayloadType	Type of payload.
Payload.payloadAction	PayloadAction	Action of payload.
Location	struct	Current location of command in geospatial coordinates in degrees.
Location.latitude	int256	Latitude of command in geospatial coordinates in degrees.
Location.longitude	int256	Longitude of command in geospatial coordinates in degrees.
Location.altitude	int256	Mean sea level altitude of command in meters.
Command	struct	Structure with location and uav actions in location.
Command.commandId	uint16	Identifier for command.
Command.location	Location	Location of the command.
Command.flightPattern	FlightPattern	Flight pattern that executed on command.
Command.formation	Formation	Formation that executed on command.
Command.payload	Payload	Payload and action of payload that executed on command.
Command.assignedUavs	address[]	Node addresses of UAVs assigned to command.
command	Command	Command loaded to swarm system.

Table 4.5 Command Contract Fields

The location struct given has int256 type values due to the lack of floating point numbers in blockchain. Latitude and Longitude values are set by multiplying the received location value in degrees with 10^{10}. When any value is requested from those fields, the node controller converts the received value by dividing it by 10^{10}.

4.1.1.6. Task Contract Task is a structure that includes input data, process type, and output data. Task structure is created to transfer tasks between swarm nodes and computation nodes. Computation nodes are more powerful systems that are available to process data faster than swarm node can. Task contract is responsible for the creation and assignment of tasks. Fields of Task Contract is given in Table 4.6.

Task		
Field	**Type**	**Description**
TaskType	enum	Types of task available on system. Such as FIREDETECT, OBJECTCOUNT etc.
TaskState	enum	Current state of task. CREATED, INPROGRESS, COMPLETED.
Availability	enum	Availability of computation nodes. AVAILABLE, NOTAVAILABLE
ProcessCapabilities	enum	Process capabilities of computation nodes. Such as IMAGEPROCESS, DATASUMARIZE etc.
DataStruct	struct	Structure to store task data.
DataStruct.data	string	Serialized data.
DataStruct.unpackString	string	Unpack string for serialized data.
SystemAvailability	struct	Availability status of computation nodes.
SystemAvailability.computationalPower	uint8	Available load measure that computational node has.
SystemAvailability.availability	Availability	Availability of computational node if node is busy.
SystemAvailability.availability	ProcessCapabilities	Process capabilities of computational node.
Task	struct	Structure to hold task data.
Task.inputData	DataStruct	Input data for being processed.
Task.outputData	DataStruct	Output data which is process output.
Task.taskState	TaskState	State of task.
Task.taskType	TaskType	Type of task.
Task.owner	address	Node address of UAV that created task.
systemsAvailable	mapping($address \rightarrow SystemAvailability$)	Maps UAV node addresses to SystemAvailability that matches Computational nodes updates their status frequently.
tasks	mapping($address \rightarrow Task$)	Maps UAV node addresses to Task that matches UAV nodes with their created Tasks.

Table 4.6 Task Contract Fields

Task contract include 3 special functions without getters and setters which are createTask, ownTask, and completeTask. Without setting parameters in task separately; createTask creates a task for the sender address in tasks map with state CREATED, ownTask checks if task is owned by another node and in progress then if task is not owned owns task with setting owner field in task with its own address then sets state INPROGRESS, finally completeTask writes task output then sets the state to COMPLETED and set owner as empty address. Those special functions make automatization on blockchain and make usage easier.

4.1.2. Modules

In the start of section Figure 4.1 summarizes the flow of model. Modules in the figure are one of the most important parts of our model. Modules are responsible for automatized actions such as messaging, message-triggered events, initialization, check timers and interaction with FCC. Mainly we can handle our models in 2 parts which are I/O module and node controller module. I/O modules handle communication between FCC and node controllers.

Node controllers manage node initialization, broadcast messages, websockets, blockchain operations, and timers. Those two module types are explained in detail in the following sections.

Figure 4.6 represents connections in UAV, FCC, and modules between themselves and other avionic devices. Sensors and links produce raw data for FCC. Those data are sent to our model runner with a serial connection which generally is RS-232 or Ethernet.

The link module given is considered as manager of all connections including air-ground communication and machine-to-machine communication. The provided link may differ with an avionic structure which can be LOS, SATCOM, or 5G. Figure 4.6 represents the main structures that are most common in UAV systems only. Different and detailed avionic architectures should be applied between modules.

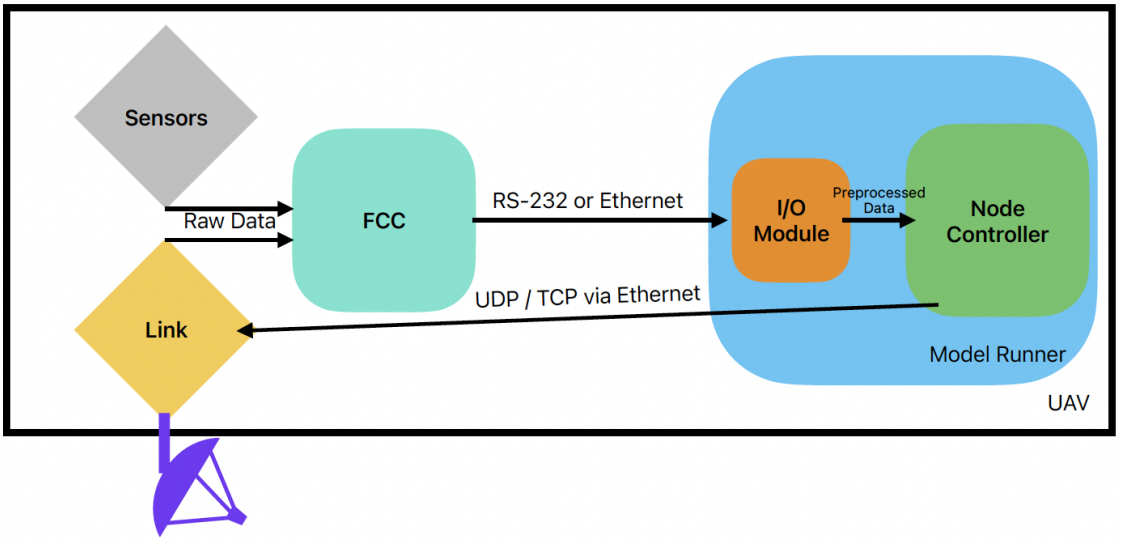

Figure 4.6 Module Connections

4.1.2.1. I/O Module Since node controllers operate at higher levels than FCC, I/O module mainly handles necessary data from FCC and converts it into a shareable format for blockchain. For experiments, I/O module has an unicast UDP listener that receives messages from a simulation script.

UAV States parameters, mission state, and payload communication must be received from FCC. Including the model into FCC directly needs low-level implementations and may cause FCC possible malfunctions. In regulations under DO-178-C, FCC is considered as level 1 (critical) system and has very strict rules for development. To handle risks, this model must run outside FCC using parameters received from FCC.

Simulation is a Python script that sends dummy data to our I/O module via UDP. I/O module receives data in a message format which is explained under messaging structures section.

I/O module handles communication, parses messages, and distributes messages to the node controller for operations triggered by messages. Flow demonstration of how I/O module stands for given in Figure 4.6.

4.1.2.2. Node Controller Module Node controller can be considered as our main module. Node controller controls messaging, program flow, and blockchain operations. All of the operations given in the following sections are executed by node controllers.

Node controller receives messages from I/O module and link module. The node controller is able to send messages directly into other nodes including GCS and other UAVs in the swarm. Node controller controls blockchain with creation, start/stop, execute commands for connections, and running miners. Node controller also manages contract deployment operations. Node controller developed in Node.js environment for dependency causes for Ethereum network interaction.

Node controller handles blockchain operations in 2 parts, initiator part interacts with Geth and Truffle for running a blockchain node and deploying contracts. Geth is a third-party application that handles blockchain processes. Truffle is another third-party application that manages and deploys contracts into a network. Node controller uses them in the order given in Figure 4.7. GCS node starts miners also to handle transactions.

Blockchain controller module uses Ethers.js library to interact with blockchain with requests. Ethers handle abi encoding, decoding, and creating connections to blockchain in many

different methods, which in the model used JSON-RPC provider, to send and manage transactions. Blockchain controller is an interface to discriminate complex operations of blockchain with useful simple operations. In a development environment, every operation is handled by objects and functions, and blockchain is abstracted near to total.

Node controller creates a multicast UDP socket to send and receive alive messages. Broadcasting alive messages makes nodes aware of other nodes in the network. Alive messages include brief information about the node to create websocket socket. By receiving alive message for the first time from a node web socket is created. And all connected nodes create a websocket to the sender node.

Operations and connections of a node controller are given in Figure 4.7 below.

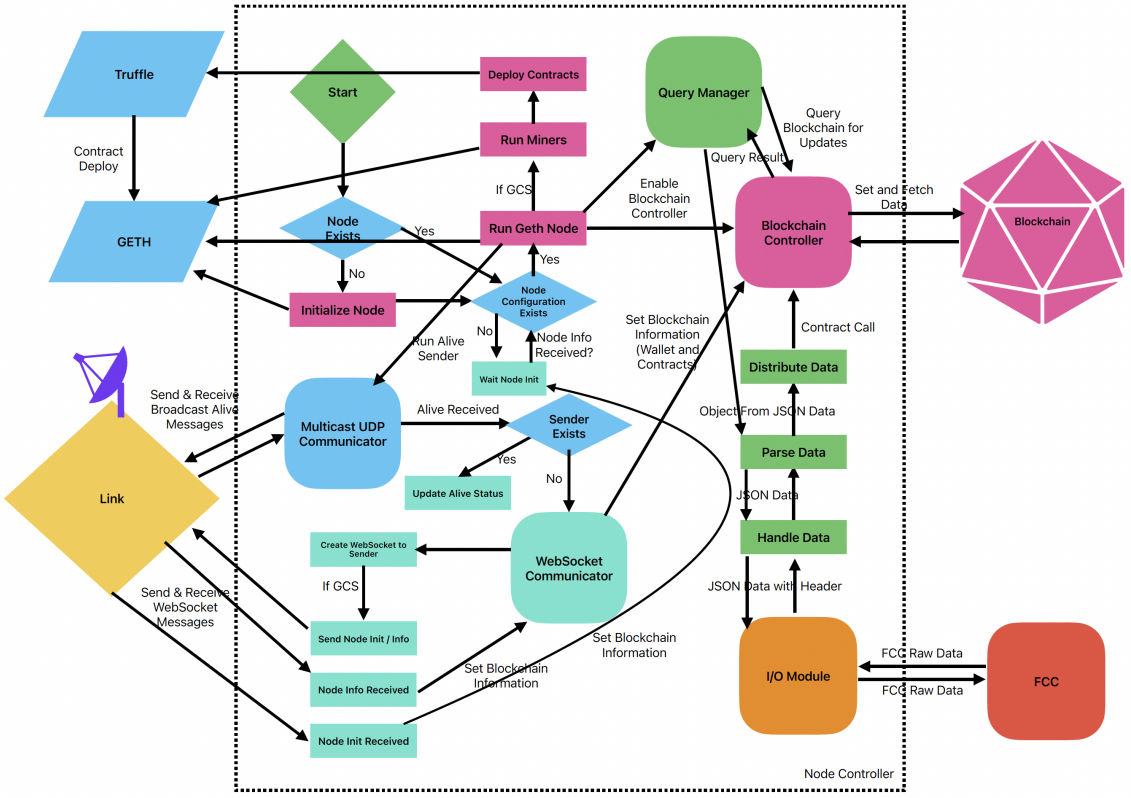

Figure 4.7 Node Controller

Websockets stand for sending messages with a reliable connection. Since websockets are implemented over TCP, sending messages from websockets is considered reliable.

Websocket connection is being used for sending wallet details to connected UAV nodes and sending contract addresses. Although UAV nodes cannot create wallets for themselves and all of them must be up to date for contracts, the GCS node deploys and distributes contract addresses to nodes via websocket. Wallet creation is limited by GCS for security issues. GCS creates wallets with an initial ETH balance.

4.1.3. Messaging Structures

Messages are easily explained as custom network packets. Over the transmission layer, a custom header is added to encapsulation. UDP and TCP structures are directly used to provide network architecture. Custom headers differ by package. In the model, 2 different packages are used which are broadcast messages with UDP and Websockets with TCP. Websockets are an application of HTTP. When using websockets, custom headers are added to HTTP payload section.

4.1.3.1. Multicast Broadcast Messages Alive messages are sent by a multicast socket in node controller. Since multicast messaging is used for broadcasting a message, sending alive messages to all nodes connected to the network is provided by a multicast socket. Alive messages should be smaller to prevent network congestion regarding high transmission period. Alive messages are sent much more than other messages in our model and make every node aware of other nodes. UDP sockets are predefined in nodes.

Alive Message		
Field	**Data**	**Description**
type	alive	Type of message
uavId	Config.uavId	uavId of Sender
timestamp	UNIX Timestamp	Timestamp in milliseconds from 1/1/1970
ip	IP Address	Ip address of sender from current network interface

Table 4.7 Structure of Alive Message

Alive messages trigger websocket creation to provide a higher-level reliable connection between nodes. Receiver nodes automatically create a websocket to the sender. If there

is already a socket created for the sender, the receiver node does not make any updates on websocket connection. Using this method on websocket creation automatically creates a mesh between nodes.

Especially, GCS checks alive statuses of nodes to make assignments, unassingment to mission plans, and update the order of UAV in the swarm. This provides a swarm system to operate regardless of holding other UAVs orders in formations. This operation should be used for swarm formation management. The structure of alive message is given in Figure 4.7 below. The message structure is serialized in JSON format to UDP packet.

4.1.3.2. WebSocket Messages Websocket is a communication method using HTTP protocol over TCP. In the model, we use websockets to send important information between nodes due to the reliability of TCP. In nodes, websockets work as a mesh between nodes so that all nodes are able to send information to each other.

We have a more complex structure than alive messages on websocket messages given in Figure 4.8. The message structure is serialized in JSON format to an HTTP packet.

WebSocket Message		
Field	**Data**	**Description**
type	Public, Private	Sets which message receivers to consider message
recipients	Array of uavId's	Receiver UAV's on private type messages
header	N/A	Structure for header
header.timestamp	UNIX Timestamp	Timestamp in milliseconds from 1/1/1970
header.uavId	Config.uavId	uavId of Sender
header.ip	IP Address	Ip address of sender from current network interface
header.wsport	Config.wsport	Websocket port of sender
header.messageType	NODEINFO, NODEINIT	Type of websocket message
data	N/A	Payload in JSON format for messages

Table 4.8 Structure of WebSocket Message

In websocket messages, we use a recipient and type field to provide broadcast a message with public type or send a message to a node or group of nodes. There are two types of websocket messages implemented in the model which are node info and node init messages.

Node init message is an initializer for UAV nodes. UAV nodes do not create an account for their own use and do not deploy contracts to the blockchain. To initialize blockchain in UAV

nodes, a node init message has to be received. Node init message includes chainId, network name, enode information, and contract addresses. With that information received UAV node can run its node and tries to connect to the blockchain network using enode information.

Node info message transfers a wallet to UAV node from GCS. GCS is authorized with an initial account with a total ETH balance in the network. The GCS node is responsible to create a wallet, fill the wallet balance and transfer it to a UAV node with a node info message. After a UAV node receives a node info message, it is enabled to make transactions using the wallet. Node info message also includes a private key of the created account.

4.2. Operations

The provided model is responsible for handling different data of UAVs and GCSs. UAV State, Mission States, Payload Information, Mission Plan, Command, and Task are data types that model responsible to hold and distribute, edit, process the contents of data types.

To classify operations into 3 types, UAV State, Mission State, and Payload Information are periodically received data types from FCC. The model is responsible to share those data between UAV nodes and GCSs. UAV nodes share their data with a map of their addresses to UAV State structs defined in the UAV States contract. Mission States are mapped with UAV addresses to Mission State structs in the Mission States contract and payload information is mapped with UAV addresses to Payload Information structs in the Payload Information contract. Consequently, the GCS node queries those maps in contracts to fetch data shared by UAVs in the swarm to provide visual feedback for operators. To explain those operations we name them as Data Share and Fetch operations.

Another type of operations is Mission Plan and Command upload. As easily understood by its name it includes mission plan and command uploads. In this type of operations, GCS sends data to the blockchain and UAVs periodically check the contents of related smart contracts to fetch changes. Mission plan is held in the Mission Plan contract with one mission plan object, and command is held in the Command contract with one command. In contrast to data share operations, a single struct for all UAVs created in contracts. However, reverse

to map of addresses to structs, each struct has assigned uav's array that holds addresses of UAVs which are assigned to that struct.

The final type is Task Transfer. Task contract holds a map of tasks created by nodes with their addresses. Every connected node updates another map in the contract that holds system availability with processing capabilities and processing availability of each node. Each node continuously updates its own system availability status. That operation allows nodes to create, own and process tasks with sharing the result of task inside the Task struct.

Flows of given operations are summarized in Figure 4.8 below with schema. Details in schema are explained in the sections below.

4.2.1. Data Share and Fetch Operations

The data-sharing operation begins when the node receives input/output (I/O) data from the I/O module. The I/O module functions as the Flight Control Computer (FCC) in UAVs and communicates with all data sources on the UAV, including EGI (Embedded GPS/INS) and payloads. It then sends the collected data to the node controller via a unicast UDP socket. When the node controller receives data during an optimal period, it uploads the latest flight data, including the aircraft's location, link health, and battery status, to the smart contract. The smart contract address is received during initialization.

The mission status, including flight mode, waypoint information, and mission information, is received from the Flight Control Computer (FCC) using the same connections, then sent to the smart contract address designated for mission status.

Because payloads can differ from each other, payload data is stored as a byte array in the contract, along with payload types and an unpacking string to provide a generic structure for all payloads. The node controller periodically receives payload data from the I/O module and uploads it to the blockchain.

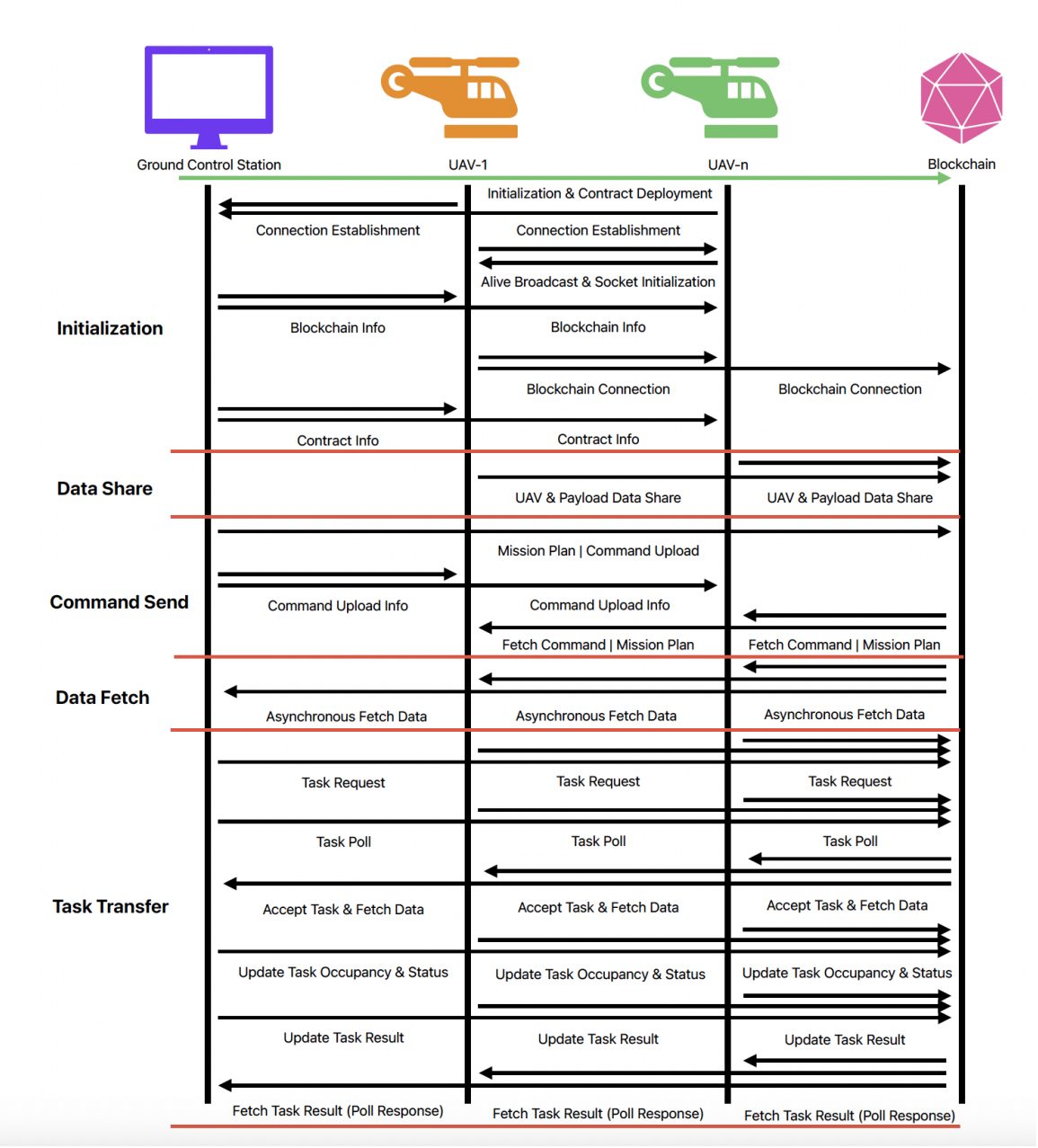

Figure 4.8 Flow of operations

The data fetch operations are on-demand operations. To retrieve data from the blockchain, the requester fetches data from related smart contract addresses by calling the 'get' functions in the smart contract. The GCS periodically fetches data from the blockchain to provide pilot tracking data.

4.2.2. Mission Plan and Command Upload

When requested by the pilot, a prepared mission plan or command can be uploaded to the smart contract, which holds the mission plan structure. Any change to the mission plan is stored with a unique hash code, which is calculated using all fields of the mission plan. Hashcode is calculated with all fields of a mission plan. Mission plan and command upload are provided by GCS and the contracts have authorization for GCS addresses. Although UAVs can fetch data from the contract, only the GCSs can upload plans and commands to the contract.

The proposed model involves multiple interconnected UAVs that periodically check hashcodes to identify any changes in their assigned mission plans and commands. If the hashcode in the blockchain differs from the local hashcode, the UAV automatically retrieves the new mission plan from the blockchain. Similarly, command IDs are checked periodically and updated if they have changed. Mission plans and command entities are dedicated to specific UAVs using their unique IDs, and each UAV only checks mission plans and commands associated with its own ID to obtain its designated commands or mission plans. The use of hashcodes to verify the authenticity of mission plans and commands is a strong security practice in this context.

UAVs are able to make assignations of themselves under waypoints with functions that only allow UAVs to assign/unassign with their addresses.

4.2.3. Task Transfer

The task transfer structure includes the task type, requester, task information containing inputs and outputs, owner assignment, completion status, and result. Inputs refer to the data sources for the given task, while outputs refer to the predicted values or the return type for the task. The task owner node can access all the data required for processing from the inputs and payload data stored on the blockchain.

The task transfer flow is as follows: a requester creates a task and adds it to the task pool. Nodes periodically check the tasks in the task pool as well as other nodes. If a node finds a task that matches its own capabilities, it checks other nodes to see if there is another node with higher computational availability. If another node has higher computational availability, the node does not assign the task to itself. Instead, the node with the highest computational availability assigns the task to itself and sets the owner information with its address. Computational availability is a metric calculated using processing capability and current load. The task owner updates the owner information on the task and starts updating the completion status with milestones. Once the task is completed, the task requester can receive the result of the processing.

4.2.4. Node Initialization

The initialization phase of a swarm is given in the first phase in Figure 4.8. Following paragraphs explain the initialization phase in GCS and UAV nodes.

4.2.4.1. GCS Node Initialization We can explain GCS initialization with the steps given below:

1. GCS creates a genesis block and runs a single node, initializing the Ethereum Virtual Machine (EVM).
2. New genesis block is created with an Ethereum balance to provide the initial ETH amount to the system.
3. The GCS authorizes the created account for signing and building and deploying smart contracts to the Ethereum chain.
4. addresses of the smart contracts are locally stored to enable sending the contract names and addresses to connected UAVs.

5. Start to wait for alive messages from UAV nodes. When GCS receives an alive message from a UAV node for the first time, GCS creates an account for UAV using uavId of the sender as seed.

6. GCS sends Node Init and Node Info to alive message sender to start initialization in the UAV node.

4.2.4.2. UAV Node Initialization We can explain GCS initialization with the steps given below:

1. UAV creates a genesis block and runs a single node, initializing the Ethereum Virtual Machine (EVM).

2. UAV starts to transmit alive messages.

3. When GCS responds to first alive message transmitted, UAV received node init.

4. With node init received, UAV sets chainId parameter and connects to the blockchain with enode information. Contract addresses are also set in configuration with information received from GCS.

5. Node info also received from GCS that includes wallet information for node created with the seed of uavId. After receiving wallet information UAV starts to make transactions with wallet received.

5. EXPERIMENTAL RESULTS

Experiments were made for evaluating the model to determine which operations are eligible to be handled by blockchain. All operations are executed to prove if blockchain can handle an operation and collect information while transactions are executed to make performance measurements. Experiment details are provided as follows:

- As the experiment environment, node ran with 16GB Ram and Intel Core i7 2.6Ghz 6 core, MacBook Pro.
- Data is collected from GCS node.
- Messages from the I/O module are sent with a period of 5 seconds.
- Mission Plan and Command messages are sent on demand.
- As FCC, a simulator script that transmits data UAV State, Mission State, and Payload Information periodically.
- Initial balance set as 100.000 ETH on genesis block.
- Different functions that set specific fields in the contract is called per (message period / 10). On that function calls the primary setter is not initiated.
- Experiments made with 16, 8, 4, 1 miner counts.
- Average execution time for data collection is 15 Minutes.
- Getter functions are called with message periods and specific field getters called per (message period / 10). On that function calls the main getter is not initiated.

We begin with a proof of successful transactions.

5.1. Transactions

This section shows empty versions of contract structures given in the method section with their filled values. Some transactions to contract addresses are also provided in this section. Shared figures prove that those operations are applicable for data-sharing reasons. However, we need performance evaluation results to determine eligibility for swarm UAV systems. Experiments are designated for determining which operations are eligible or not for swarm UAV systems with reasons.

Figures provided collected data by querying the network in the initial state and after transactions.

5.1.1. UAV States Contract

UAV States contract is set successfully with all fields and structs. Enum values are shown with integer values corresponding to enums. UAV States struct has a fixed size. The location struct given has int256 type values due to blockchain's lack of floating point numbers. Latitude and Longitude values are set by multiplying the received location value in degrees with 10^{10}. When any value is requested from those fields, the node controller converts the obtained value by dividing it by 10^{10}. Fields of the contract are given in Figure 5.1.

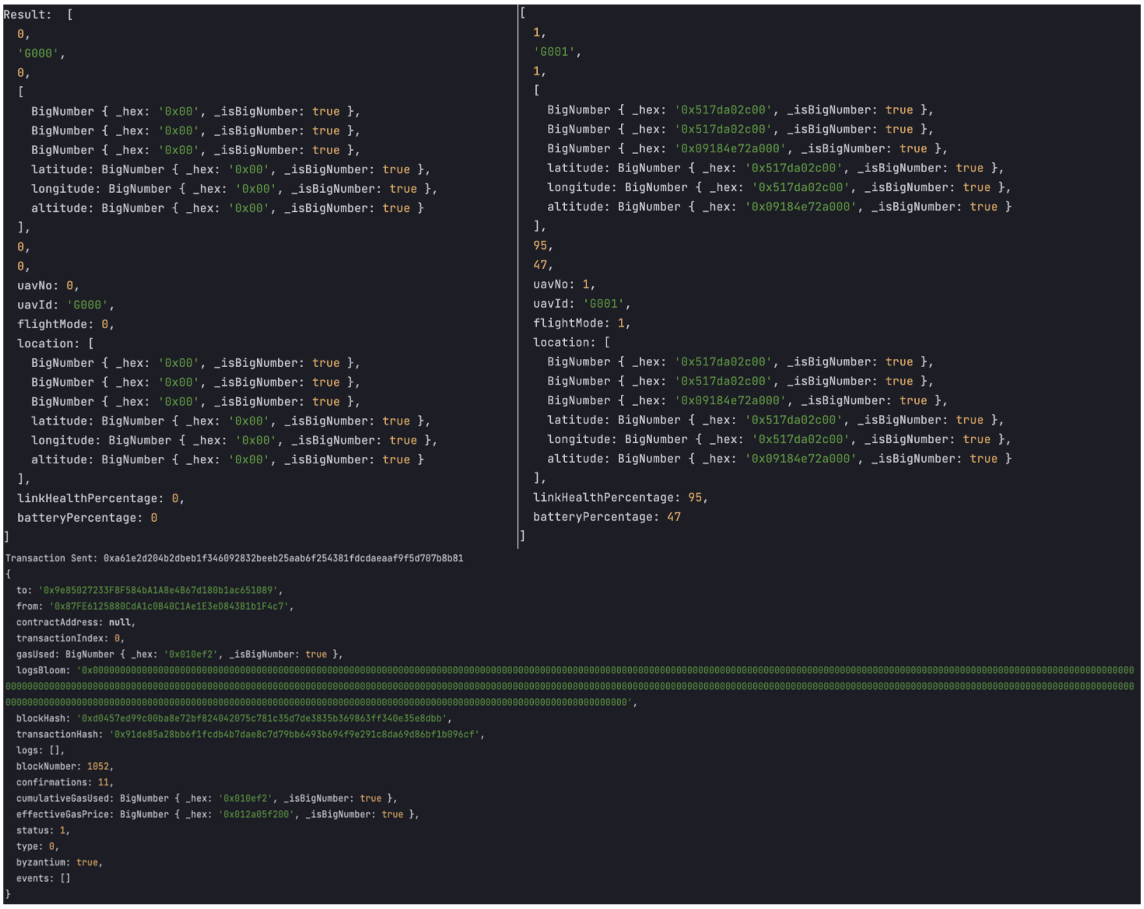

Figure 5.1 Successful set operation on UAV States Contract

5.1.2. Mission States Contract

Mission States contract is set successfully with all fields and structs. Enum values are shown with integer values corresponding to enums. Mission States struct has a fixed size. Fields of contract are given in Figure 5.2.

```
Result:  [
  0,
  '0x0000000000000000000000000000000000000000',
  0,
  '',
  0,
  0,
  0,
  0,
  [ 0, 0, payloadType: 0, payloadStatus: 0 ],
  mpid: 0,
  uavaddress: '0x0000000000000000000000000000000000000000',
  uavno: 0,
  uavId: '',
  currentWpId: 0,
  nextWpId: 0,
  flightStatus: 0,
  formation: 0,
  payload: [ 0, 0, payloadType: 0, payloadStatus: 0 ]
]
```

```
[
  0,
  '0x87FE6125880CdA1c0B40C1Ae1E3eD843B1b1F4c7',
  1,
  'G001',
  101,
  102,
  1,
  2,
  [ 1, 1, payloadType: 1, payloadStatus: 1 ],
  mpid: 0,
  uavaddress: '0x87FE6125880CdA1c0B40C1Ae1E3eD843B1b1F4c7',
  uavno: 1,
  uavId: 'G001',
  currentWpId: 101,
  nextWpId: 102,
  flightStatus: 1,
  formation: 2,
  payload: [ 1, 1, payloadType: 1, payloadStatus: 1 ]
]
```


```
{
  to: '0xa65D80F74b755396d22212649068b44f053845ab',
  from: '0x87FE6125880CdA1c0B40C1Ae1E3eD843B1b1F4c7',
  contractAddress: null,
  transactionIndex: 0,
  gasUsed: BigNumber { _hex: '0x01414e', _isBigNumber: true },
  logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
  blockHash: '0xecb6eed2078b5051d4f89ad9dd9c5936edfea059186d1c6cfc75b182da99117b',
  transactionHash: '0xa61e2d204b2dbeb1f346092832beeb25aab6f254381fdcdaeaaf9f5d707b8b81',
  logs: [],
  blockNumber: 1060,
  confirmations: 3,
  cumulativeGasUsed: BigNumber { _hex: '0x01414e', _isBigNumber: true },
  effectiveGasPrice: BigNumber { _hex: '0x012a05f200', _isBigNumber: true },
  status: 1,
  type: 0,
  byzantium: true,
  events: []
}
```


Figure 5.2 Successful set operation on Mission States Contract

5.1.3. Payload Information Contract

The payload information contract is set successfully with all fields and structs. Enum values are shown with integer values corresponding to enums. Since the data field of the contract includes serialized information with different length, the payload information contract will contain different-sized payloads. Fields of the contract are given in Figure 5.3.

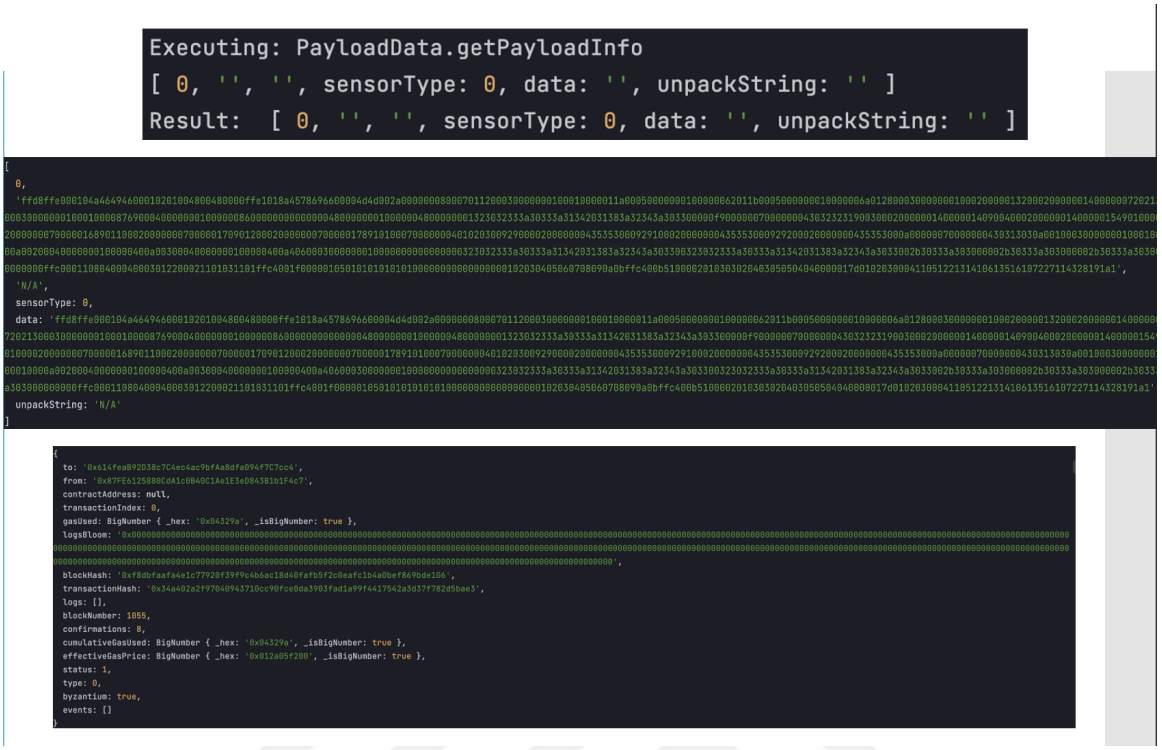

Figure 5.3 Successful set operation on Payload Information Contract

5.1.4. Mission Plan Contract

Mission States contract is set successfully with all fields and structs. Enum values are shown with integer values corresponding to enums. Mission plan struct has an array of Waypoint structures, making mission plan operations differ by size and gas measures. The location struct given has int256 type values due to the lack of floating point numbers in blockchain. Latitude and Longitude values are set by multiplying the received location value in degrees with 10^{10}. When any value is requested from those fields, the node controller converts the obtained value by dividing it by 10^{10}. Fields of the contract are given in Figure 5.4.

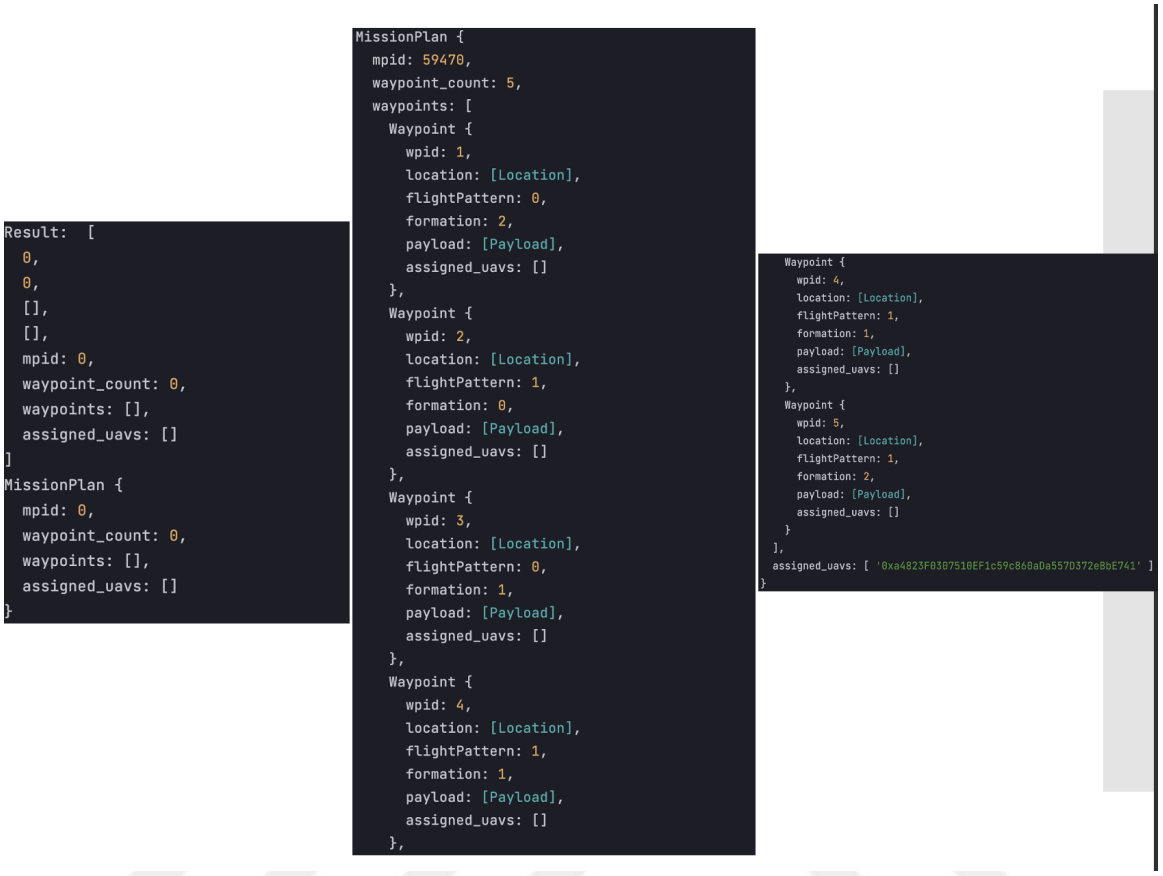

Figure 5.4 Successful set operation on Mission Plan Contract

5.1.5. Command Contract

The command contract is set successfully with all fields and structs. Enum values are shown with integer values corresponding to enums. Command struct has a fixed size; only UAV assignments change the size for address size. The location struct given has int256 type values due to blockchain's lack of floating point numbers. Latitude and Longitude values are set by multiplying the received location value in degrees with 10^{10}. When any value is requested from those fields, the node controller converts the obtained value by dividing it by 10^{10}. Fields of the contract are given in Figure 5.5.

Figure 5.5 Successful set operation on Command Contract

5.1.6. Task Contract

The command contract is set successfully with all fields and structs. Enum values are shown with integer values corresponding to enums. Task contract has different size due to the input and output data sections inside. A node owned by another node creates the task, and the creator node fetches the result. States are important for tasks.

Fields of an empty contract are given in Figure 5.6.

Figure 5.6 Empty Task

Fields of the created task are given in Figure 5.7. In the figure, the task is owned by a processor.

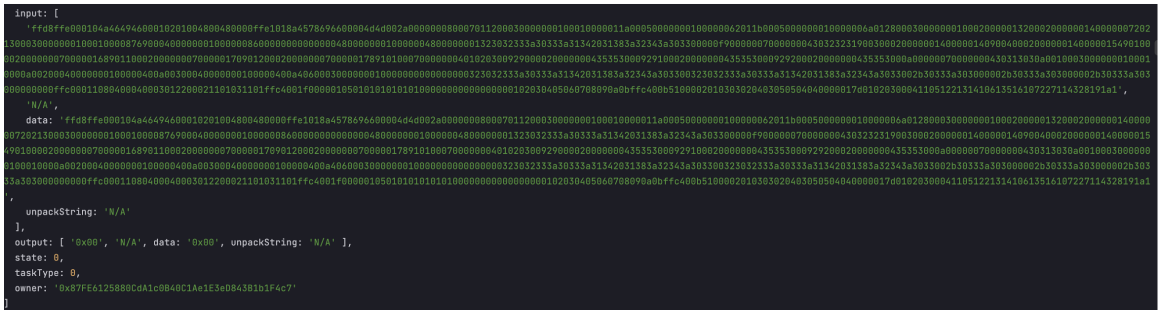

Figure 5.7 Created Task

Fields of completed task are given in Figure 5.8. The result field is set on the output section of the task structure.

Figure 5.8 Completed Task

5.2. Performance Evaluation

As seen in the section before, all contracts have interacted, and fields are set by the node controller with transactions. However, making change operations on blockchain needs to be processed. Miners process transactions and append them to the blockchain for a gas fee. Miners racing for the highest prize to collect the highest gas. At the same time, while executing operations, gas is consumed during execution. If the gas runs out before execution,

the miner cancels operations; otherwise, the miner processes the transaction if the gas is enough.

That is, given mining operations require gas for transactions, which takes time to process. Execution time and processing transactions in a limited time are essential for UAV systems. Operations such as mission plan upload and command upload are not periodic but need to be executed in a time margin.

Firstly, the contract transaction percentage is given in Figure 5.9. Contract call counts provide information about how much contract is open for congestion. Transactions to contract are added to the transaction pool in the network for functions has more priority; for those congested contracts, transactions are sent by higher gas to prioritize function calls. All figures given are collected from 8 miners' active experiments.

Figure 5.9 Transaction percentages for Contracts

Secondly, function execution counts are given in Figure 5.10. Getter functions have exact counts. All of them are called by GCS query to monitor blockchain. Call counts are essential due to functions with heavy load blocks miners until execution ends.

Figure 5.11 is provided to examine gas fees of function-based transactions. The setUAVState function is the setter for UAV States contract and is executed periodically to update UAV information in blockchain. The first execution has a higher gas fee due to the initial set that changes all empty values to a value that consumes higher gas. Fixed data size makes gas fees the same for all executions.

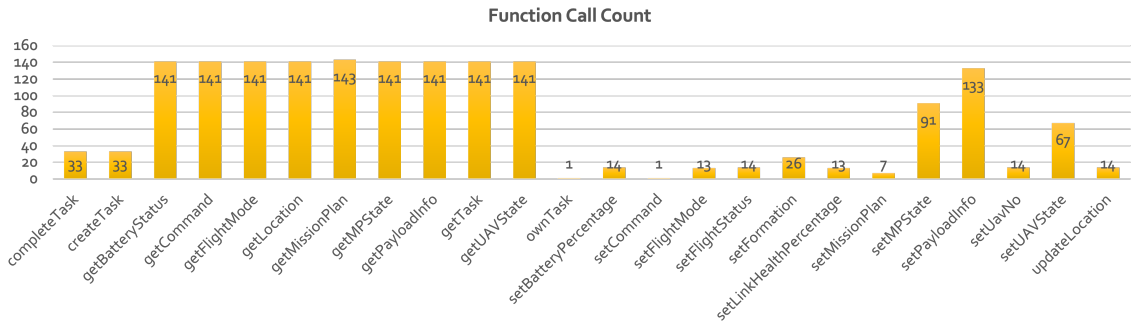

Figure 5.10 Execution counts of contracts

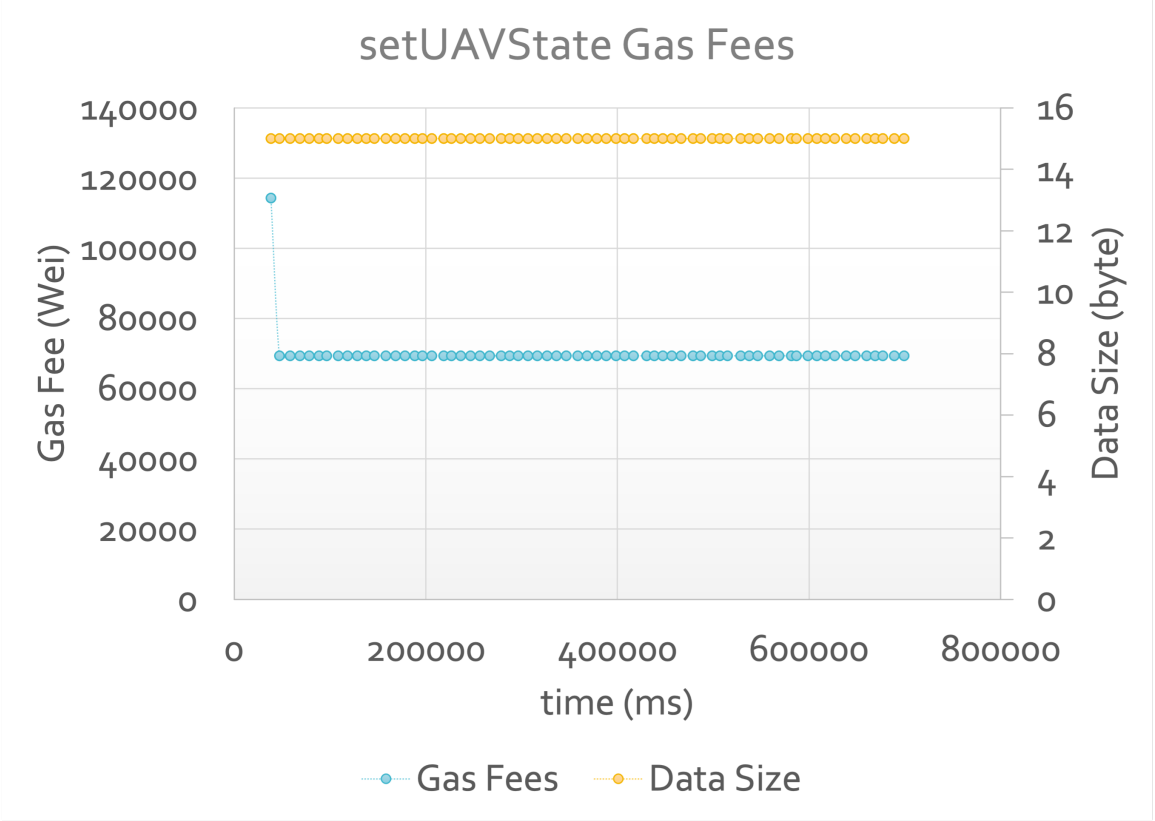

Figure 5.11 Gas Fees for setUAVState transaction

Gas fees and size of a periodic set Contract Field Mission Plan given in Figure 5.12. The initial set that changes all empty values to a value consumes higher gas. The different-sized contract calls change the gas amount consumed. The gas amount consumed represents how heavily it functions to get its transaction processed. Transactions with large data block miners with high gas and become the first target for miners. A polynomial curve in the plot correlates data size and gas fees.

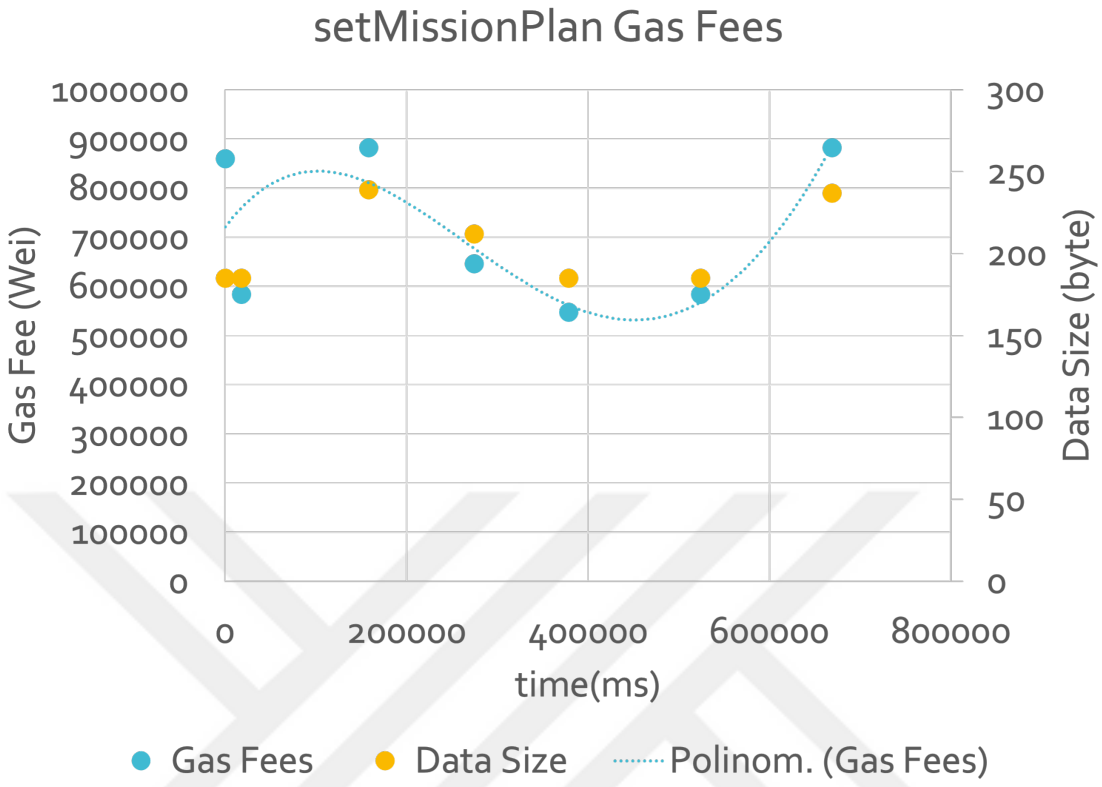

Figure 5.12 Gas Fees for setMissionPlan transaction

To examine congestion on the network, a data sample is provided in 5.13. In the figure, marked blue cells represent operations with high gas fees, and red cells show the congestion that high gas operation causes. Transactions with large data block miners with high gas and become the first target for miners. Blocking miners makes other transactions wait and causes congestion on blockchain networks. This congestion may cause timeouts for UAVs and cause late operations on critical operations. To prevent this gas amount sent with mission plans, commands are higher than periodic values to make miners choose them to process first. However, payload information has a significant amount of data and sets data periodically.

Figure 5.13 shows execution times for operations proving high values due to congestion.

In Figure 5.14 dots represent data size, and bars represent gas fees. The mission plan set operation has higher gas fees despite its lower data size than task and payload operations. The source of these results depends on the heavy operation while setting a mission plan.

Time (T0 + MS)	Function Name	Contract Name	Gas Fee (Wei)	Time Elapsed (ms)	Data Size (byte)
47451	setPayloadInfo	PayloadData	275098	2212	1030
47454	completeTask	TaskContract	77726	1480	11
47455	setUAVState	UAVStates	69362	3229	15
47460	setUavNo	MPStates	27333	5230	1
49074	getUAVState	UAVStates	0	48	0
49077	getFlightMode	UAVStates	0	44	0
49079	getBatteryStatus	UAVStates	0	50	0
49081	getLocation	UAVStates	0	50	0
49104	getMPState	MPStates	0	19	0
49158	getPayloadInfo	PayloadData	0	19	0
49202	getMissionPlan	MissionPlanContract	0	22	0
49338	getCommand	CommandContract	0	52	0
49358	getTask	TaskContract	0	32	0
50322	setPayloadInfo	PayloadData	275098	79	1030
51418	setMPState	MPStates	82254	4185	67
51421	setLinkHealthPercentage	UAVStates	27262	2191	2
52328	getMissionPlan	MissionPlanContract	0	91	0
54082	getBatteryStatus	UAVStates	0	52	0
54084	getFlightMode	UAVStates	0	48	0
54086	getUAVState	UAVStates	0	60	0
54093	getLocation	UAVStates	0	59	0
54136	getMPState	MPStates	0	37	0
54161	getPayloadInfo	PayloadData	0	21	0
54207	getMissionPlan	MissionPlanContract	0	26	0
54339	getCommand	CommandContract	0	48	0
54379	getTask	TaskContract	0	52	0
58398	createTask	TaskContract	813234	2444	1084
58398	setUAVState	UAVStates	69362	4164	15
58399	setPayloadInfo	PayloadData	275098	3151	1030

Figure 5.13 Congestion shown in data sample

Mission plan upload creates new waypoints and appends them to the mission plan from the data received. This operation causes high gas fees and execution time because it has a loop for traversing all waypoints. On other function executions, we can also see a correlation between gas and data size.

Each function call that alterates blockchain consumes gas and makes transactions. Getters query the blockchain to collect information that does not require a transaction and gas fee.

Figure 5.15 dots represent gas fee, and bars represent execution times. setPayloadInfo function consumes higher gas than many other functions due to its data size and claims priority with higher gas. Payload information values are set periodically, which causes congestion on the network that other functions with smaller data sizes. Additional functions with small gas fees lie after setting payload info in execution times that do not interact with heavier operations.

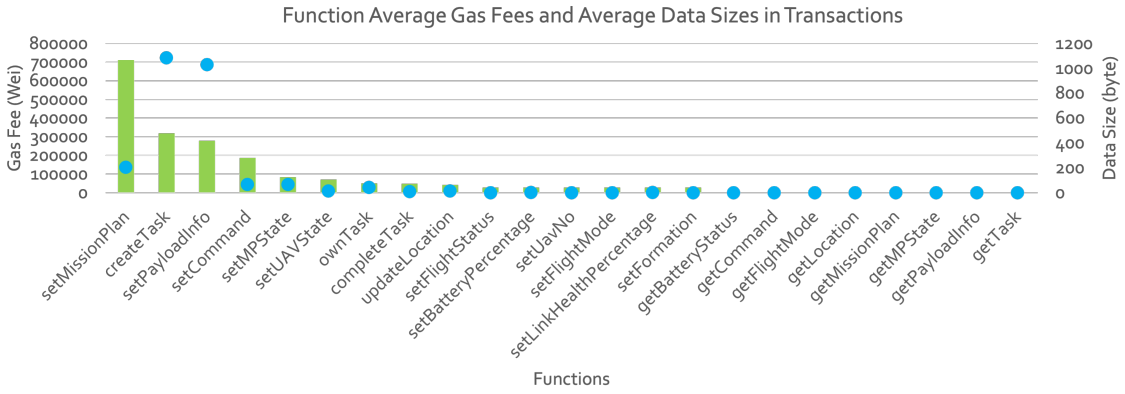

Figure 5.14 Gas fee consumptions, data sizes for function executions

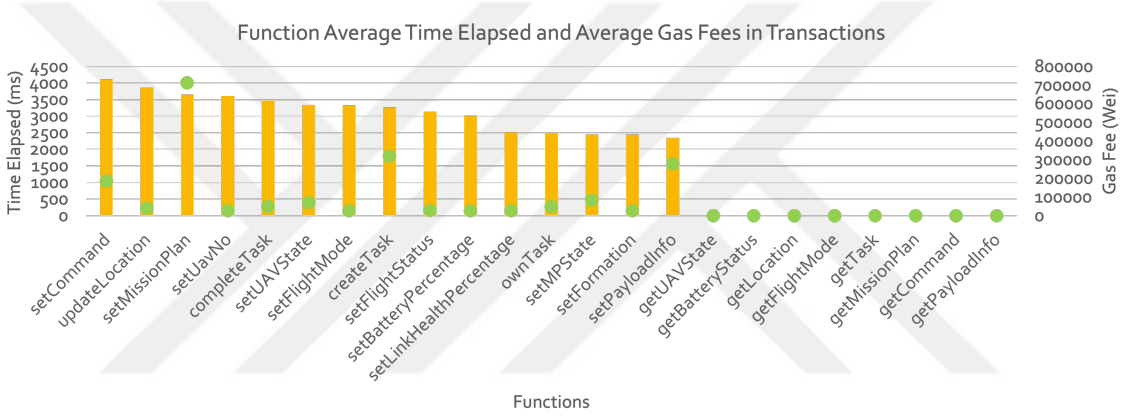

Figure 5.15 Gas fee consumptions, execution times for function executions

Gas fees are the same for different miner counts, as shown in Figure 5.16. Mission Plans gas fees are differentiable due to size changes.

Miners collect and process transactions into the same block on fewer miner counts. This makes querying faster, so getter times get shorter on fewer active miners. A bulk transaction processing example is given in Figure 5.17

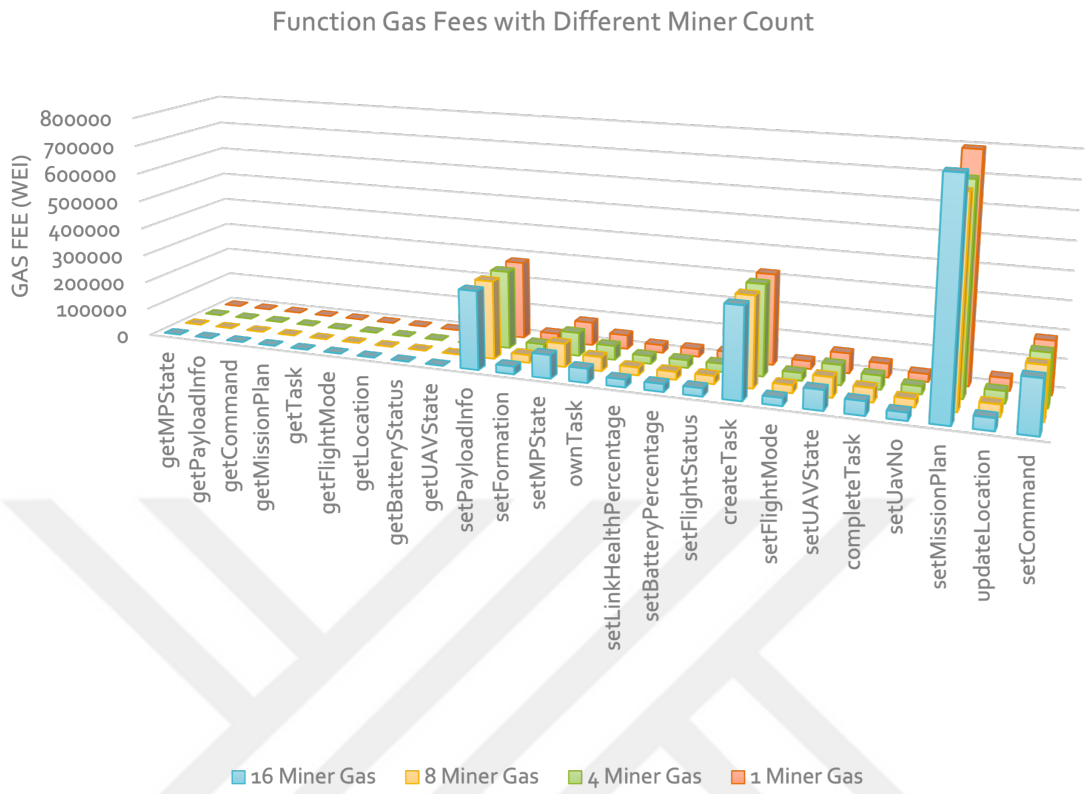

Figure 5.16 Gas fees with different miner counts

Figure 5.17 Bulk transaction processing on the same block

However, execution times differ by miner counts given in Figure 5.18. Gas fees are not changed due to the same operation, but time averages are raised regarding congestion.

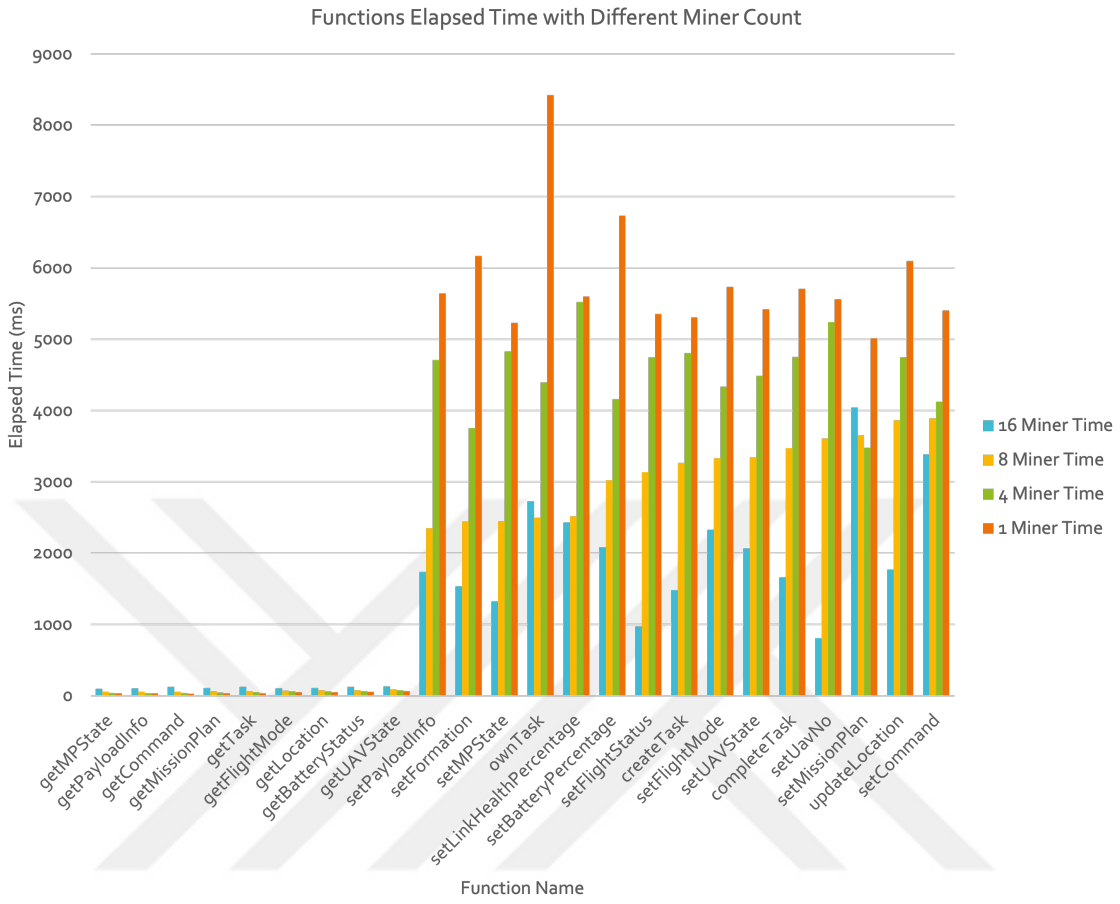

Figure 5.18 Execution times with different miner counts

5.3. Discussion

To compare our model with other blockchain-based data-sharing applications, several distinct factors can be considered. In the context of our model, which operates with UAVs, factors such as time and process load become essential considerations. Therefore, for an objective comparison of blockchain applications, this section will focus on important factors related to scalability and security.

Firstly, let us discuss security. Blockchain is widely regarded as a secure technology due to its inherent properties. Public blockchain systems are vulnerable to tampering and attacks, which raises concerns about data security. However, our model operates within closed networks, utilizing RF data links for connections. As a result, the system functions as a

closed network, minimizing the risk of unauthorized access. The Ground Control Stations (GCS) provide initial data to the connected UAVs, and without proper information and account credentials from the GCS, requesting access and making transactions is impossible. In the absence of encryption mechanisms, an intruder can only observe network traffic and gather information about the presence of UAVs. This inherent security design allows our model to bypass additional security measures and operate efficiently, which is particularly advantageous for UAV systems where speed is crucial.

The second factor to consider is scalability. Blockchain applications often face challenges when dealing with a large number of transactions. The high cost of providing miners to the network and the limited availability of miners can hinder the processing of numerous transactions. In our model, optimal operation is achieved with four miners per UAV. Even with heavy operations, such as periodical payload information sharing that may cause congestion on-demand, our model can process all transactions within a period of 5 seconds using four miners per UAV. This efficient processing time can be observed in Figure 5.18, which demonstrates the execution times of transactions.

Another significant advantage of our model lies in the use of a private Ethereum network. Operating on a private network reduces network traffic and minimizes latency compared to using the Ethereum mainnet. Synchronization times, transaction execution times, and gas fees tend to be higher on the Ethereum mainnet due to the large number of transactions awaiting processing and the higher difficulty level. The results obtained from different blockchain applications are illustrated in the following figures.

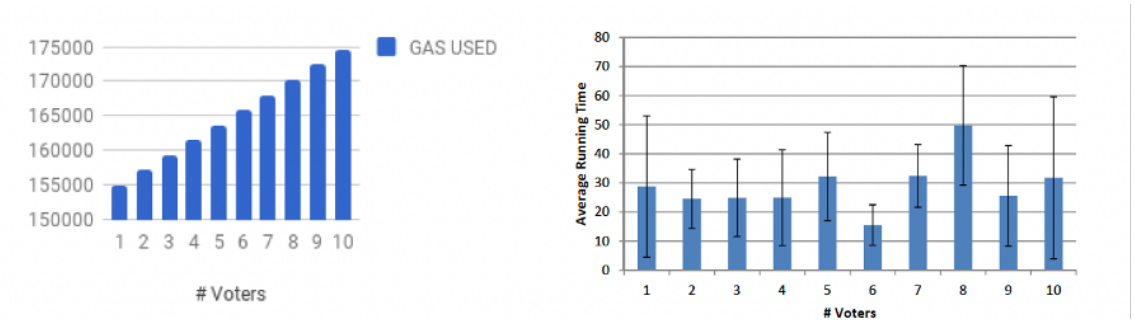

Figure 5.19 Gas Fees and Transaction Times of Adjudication Voting System a) Gas Consumption b) Average Running Time with Standard Deviation Error Bars[2]

In a comparative analysis, researchers proposed a novel alternative model, as depicted in Figure 5.19, for tracking, managing, and adjudicating data-sharing agreements using smart contracts and blockchain technology [2]. Their framework involved the generation of smart contracts based on legal parameters derived from data-sharing agreements. Through the use of blockchain technology, the system automatically enforced and adjudicated the terms outlined in these agreements. To ensure accountability, monetary penalties were implemented, with external auditors participating in secure voting to hold violators responsible.

The researchers conducted their experiments on the Ropsten testnet, a blockchain network. However, the results indicated an increase in total operation gas fees as the voter count and transaction count rose, as illustrated in Figure 5.19. Furthermore, the transaction processing times exceeded 20 seconds. In contrast, our model utilizes fewer gas fees for similar voting operations and completes them in under 5 seconds. This improved efficiency is attributed to our model's use of a private network, where only miners involved in the UAV system process the transactions.

By employing a private network and optimizing the transaction processing mechanism, our model provides a more efficient and cost-effective solution compared to the proposed alternative. The reduced gas fees and faster transaction processing times contribute to the overall effectiveness and practicality of our model in data-sharing agreements within the context of UAV systems.

Function	Gas	Cost	Time, s
register	108462	$0.14	9
requestUAV	192253	$0.25	18
changestate(true to false)	85455	$0.11	11
changestate(false to true)	40538	$0.05	11
confirm	33874	$0.04	11

Figure 5.20 Gas Fees and Transaction Times for Distributed Mobile Edge Computing [3]

In another study focusing on mobile edge computing, blockchain technology was employed with the objective of ensuring public accessibility to algorithms. The researchers utilized smart contracts in their work and presented simulation results demonstrating the reasonable and feasible resource consumption and time cost of their proposed scheme [3]. The Rinkeby network was employed for this research, and the gas fees associated with the operations were found to be more affordable compared to a voting system. Figure 5.20 illustrates the lower gas fees resulting from the simplicity of the operations. However, the time measures were higher when compared to our model.

In contrast, our model executes transactions with fewer gas fees, even for similar procedures that involve altering a single value on the blockchain. Furthermore, the time required for our model to perform these operations is significantly shorter and more suitable for UAV operations. This efficiency is achieved through the utilization of a private network, which employs dummy ETH balances and eliminates any monetary costs associated with the transactions.

By leveraging a private network and optimizing the transaction process, our model provides a more cost-effective and time-efficient solution compared to the work focused on mobile edge computing. The reduced gas fees, coupled with faster transaction processing times, enhance the practicality and applicability of our model in the context of UAV systems.

In a different research endeavor, the focus was on addressing trust and authentication challenges in IoT networks by utilizing smart contracts for access control management. Multiple smart contracts, including the Access Control Contract (ACC), Register Contract (RC), and Judge Contract (JC), were employed to facilitate efficient access control management [4]. The researchers deployed these contracts and executed them on the Ethereum mainnet, which is the largest and primary network of Ethereum globally. However, it should be noted that executing transactions on the Ethereum mainnet can be time-consuming, with execution times ranging from minutes.

Figure 5.21 presents the measurement of function calls on the ACC contract functions, represented in gas units. On the Ethereum mainnet, the gas unit is denoted as Gwei,

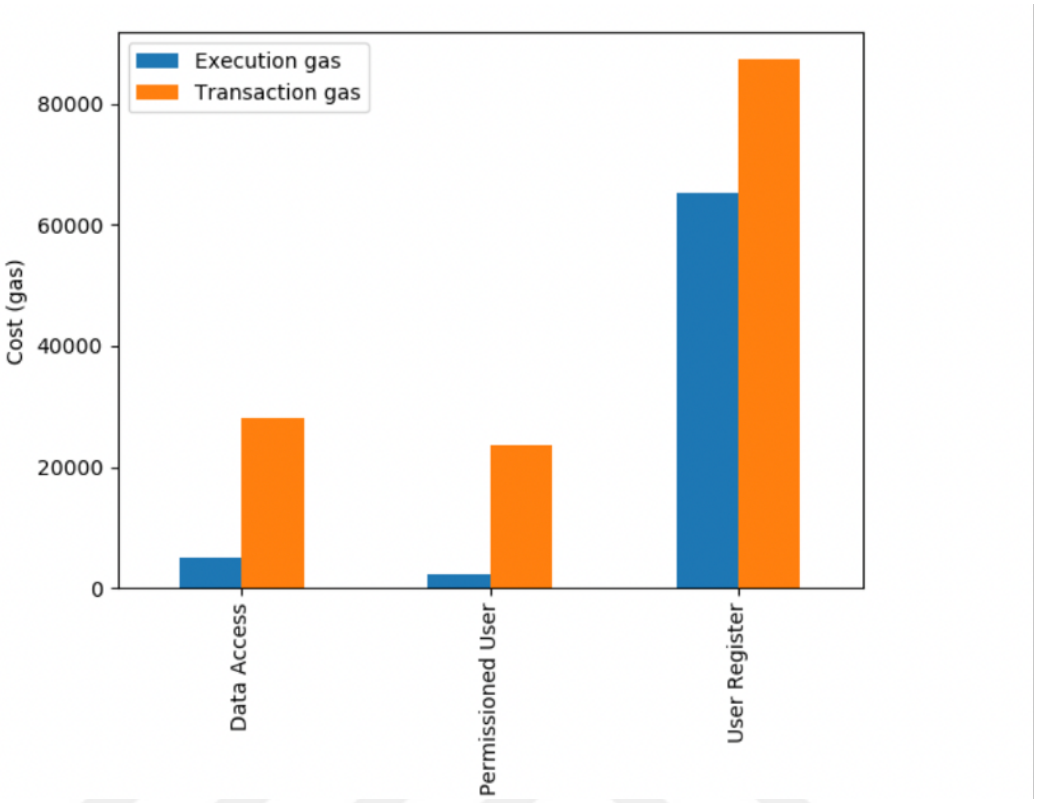

Figure 5.21 Gas Fees and Transaction Times of IoT Data Sharing System Integration for ACC Contract[4]

which signifies a billion times Wei. The figures displayed in Figure 5.21 indicate high fees associated with certain functions, such as 0.00178 ETH for the user registration function. The use of the Ethereum mainnet entails monetary implications due to these high gas fees. In contrast, our network employs dummy ETH balances that do not involve any monetary costs. Additionally, the execution times of our model are significantly shorter compared to the Ethereum mainnet.

By utilizing a private network and dummy ETH balances, our model addresses the limitations associated with trust and authentication in IoT networks in a more cost-effective and time-efficient manner. The absence of monetary costs and the reduced execution times further enhance the practicality and viability of our model within the context of UAV systems.

In conclusion, our model uses the advantage of a private Ethereum network by making miners work just for transactions that nodes make. The private network in our model is

created with a dummy ETH balance on initialization. All ETH used for transaction fees does not require any money. Moreover, using a private Ethereum network in a closed network creates an opportunity for bypassing security procedures for signing and validation making processing transactions faster. Dividing alteration functions to work with smaller amounts of data requires less gas and faster transactions in our model. With a glance at the works given, it's obvious to see the advantages of our model.

Further development of the model using Hyperledger with Practical Byzantine Fault Tolerance (PBFT) would enhance performance, scalability, and security manners. The utilization of Hyperledger with the PBFT consensus algorithm offers several advantages over standard smart contracts, particularly in terms of transaction times and overall system robustness. Firstly, a significant reduction in transaction times is achieved through Hyperledger and PBFT. The PBFT consensus algorithm enables rapid transaction finality, ensuring quick confirmation and availability of shared data within the swarm UAV system. This efficient consensus mechanism leads to faster transaction processing compared to standard smart contracts on public or permissionless blockchains, which often involve longer confirmation times and higher latency. Secondly, PBFT provides enhanced system robustness by offering strong fault tolerance against Byzantine failures. The consensus algorithm ensures that the swarm UAV system can reach an agreement even in the presence of malicious or faulty nodes. This resilience to Byzantine failures enhances the integrity and security of data sharing, as it mitigates the risk of data manipulation or compromise by malicious actors within the system. Lastly, Hyperledger's permissioned access and fine-grained control over data sharing provide additional advantages over standard smart contracts. The customization capabilities of Hyperledger enable tailored data-sharing mechanisms that can meet the specific privacy and security requirements of the swarm UAV system. This allows for a more controlled and secure environment, ensuring that sensitive mission-related data is only accessible to authorized participants.

In summary, the utilization of Hyperledger with the PBFT consensus algorithm presents several advantages over standard smart contracts. These include faster transaction times, enhanced system robustness through strong fault tolerance, and the ability to customize

data-sharing mechanisms to meet specific privacy and security requirements. These benefits contribute to the secure, efficient, and controlled exchange of mission-critical information within the swarm UAV system.

To take a glance at the future of provided model, the model should merge and divide swarms easily. With easy deployment, using simple operations, we can expand the swarm with other UAVs or other GCSs that mission plans deployed for all at once, assigning swarms should be easily done with the help of a provided model. Following scenarios will explain possible usages.

5.3.1. Scenario 1: Adding UAVs to Swarm Using Blockchain with Multiple GCSs

This scenario demonstrates a military operational environment. In this military operation, various fleets are deployed at different locations. For instance, intelligence systems are stationed at the intelligence team base, while armed UAVs are deployed at the military base. Pilots operate from a secure room within a fortified military building. The scenario involves the utilization of multiple GCSs and UAV fleets, employing blockchain-based communication.

In this scenario, the integration of blockchain technology is explored to facilitate the process of adding UAVs to a swarm that involves multiple GCSs with distinct operational capabilities. Specifically, two GCSs are considered, each having its own fleet of UAVs with specialized functionalities. The integration is achieved through the implementation of a mission plan contract on the blockchain network.

The mission plan contract serves as a smart contract that contains the necessary information regarding the mission objectives, waypoints, and assigned tasks for the UAVs within the swarm. By utilizing the transparency and immutability of the blockchain, the mission plan contract ensures that all participating GCSs have access to the same information, thereby enabling synchronized and coordinated operations.

Furthermore, the mission plan contract allows for the allocation of specific tasks and waypoints to the fleets of the different GCSs based on their distinct operational capabilities. For instance, one GCS fleet may possess UAVs equipped with radar and communication intelligence payloads, while the other fleet may consist of UAVs armed with weaponry. By leveraging the mission plan contract, the swarm can effectively distribute tasks and waypoints to optimize the utilization of the various capabilities present in the fleet, thereby enhancing the overall mission performance.

However, it is crucial to address certain challenges associated with this scenario. These challenges include ensuring secure communication and coordination among the GCSs, accommodating the varying operational characteristics and limitations of the different UAV fleets, and managing potential conflicts or overlaps in assigned tasks and waypoints. Consequently, a comprehensive analysis of these challenges is necessary to validate the feasibility and effectiveness of using blockchain technology to add multiple GCSs and their respective UAV fleets to a swarm, while ensuring efficient mission planning and execution.

5.3.2. Scenario 2: Enhancing Node Configuration and Introducing Fixed-Wing UAV as a Mobile GCS

This scenario depicts an environment that has recently been struck by a disaster. In order to commence search and rescue operations, UAVs play a crucial role, particularly equipped with IR cameras and sensors for effective search operations. Rotary-winged UAVs have the ability to descend to lower altitudes to detect any signs of life. However, conducting these operations with a deployed GCS poses challenges due to time limitations. On the other hand, deploying fixed-wing UAVs is relatively easier. The utilization of a fixed-wing UAV, capable of operating continuously for 2-3 days, as a mobile GCS relay facilitates seamless communication and coordination, thereby enhancing the operational capabilities of the swarm UAV system.

In this scenario, the focus is to enhance the node configuration within a swarm by introducing a new type of node that acts as a relay for contract addresses without deploying them. This

new node plays a crucial role in facilitating the initialization process of other swarm nodes and acts as a GCS for various operations. Additionally, a fixed-wing UAV is deployed to serve as a mobile GCS, enabling swarm operations in remote locations where a traditional GCS is unavailable.

The introduction of the relay node in the node configuration allows for the seamless addition of new nodes to the swarm. By relaying contract addresses without the need for immediate deployment, this node simplifies the process of integrating additional swarm nodes into the network. The relay node acts as an intermediary, providing necessary information and wallets to the swarm nodes, enabling their initialization and integration into the swarm. This approach leverages the decentralized and transparent nature of blockchain technology to ensure efficient and secure node configuration within the swarm.

Furthermore, a fixed-wing UAV is deployed as a mobile GCS to address scenarios where a physical GCS is not readily available. By utilizing the blockchain to store and distribute mission-related information, the fixed-wing UAV is capable of commanding and coordinating the swarm, even in remote locations lacking traditional GCS infrastructure. This enables the swarm to execute missions and maintain communication and synchronization using the information stored in the blockchain.

However, several challenges can be addressed in this scenario. These challenges include ensuring reliable and secure communication between the relay node, swarm nodes, and the fixed-wing UAV acting as a mobile GCS. Additionally, measures should be implemented to minimize transmission delays and optimize the mining process while the fixed-wing UAV is in operation. Furthermore, the integration of the fixed-wing UAV into the swarm requires careful consideration of flight dynamics, operational limitations, and potential conflicts with other UAVs in the swarm.

In conclusion, the introduction of a relay node in the node configuration and the deployment of a fixed-wing UAV as a mobile GCS demonstrate the potential for enhancing swarm operations in scenarios where traditional GCS infrastructure is limited or absent. By leveraging the capabilities of blockchain technology, swarm nodes can be efficiently

initialized and integrated, while the mobile GCS enables mission execution and coordination in remote locations. Nonetheless, further research and development are necessary to address the challenges associated with this scenario and validate its effectiveness in real-world swarm applications.

6. CONCLUSION

A new data management model that utilizes blockchain technology to enhance swarm UAV systems, which consist of multiple UAV devices controlled simultaneously for different missions. The model offers various data-sharing functions such as payload data, UAV data, mission planning, mission tracking, and task operations, and it ensures secure communication between swarm nodes without the need for a master UAV in the system. The proposed architecture allows GCSs or other UAVs to connect to the network and update data simultaneously, enabling the execution of mission plans in a decentralized way. The use of blockchain technology enhances system robustness against possible attacks and allows tracking of the mission state for all UAVs, along with sending other commands such as waypoint updates and data requests. With minimal development effort, the proposed model can be easily expanded to incorporate additional capabilities, making it a powerful and efficient solution for swarm UAV systems.

All contracts and functions are working. Since setter functions alter blockchain, setter functions use gas. Getter Functions are not using gas due to just querying the chain. Functions that alter a small portion of contracts use less gas. Transactions with large data block miners, with high gas becoming miners' first target. Blocking miners makes other transactions wait and causes congestion on blockchain network. Congestion may cause timeouts for UAVs and cause late operations on critical operations. To prevent late critical operation execution, the gas amount sent with mission plans and commands is higher than periodic values to make miners choose them to process first. Payload information has a big amount of data and sets data periodically. The gas amount consumed represents how heavy is functions to get its transaction processed. Gas fees are not changed due to the same operation, but time averages are raised regarding congestion. Miners collect and process transactions into the same block on fewer miner counts. This makes querying faster, so getter times get shorter on fewer active miners. 15 minutes total elapsed. The total price in Wei is 70.397.651. Wei / Hour for the system is about 281.590.604 Wei. 1 ETH could

process 3.5 Billion transactions. ETH amount will be enough for the system to operate for hours.

Critical operations need high responsiveness in less than 100 milliseconds for UAV systems. UAV States, which include location and crucial avionic data, can be considered as critical. Such that using UAV States for flight reasons is not suitable. However, it can be used for visualization.

Payload Information is included in Tasks in case of need. However, setting payload information on the chain causes congestion due to high gas prices and data amounts. Payload information may be set on-demand but not periodically.

Although given operations are asynchronous and generally triggered by user interaction, they do not need time limits, and all applications within the model: Mission Plan Upload, Command Upload, Mission States, and Task Transfer.

REFERENCES

[1] Blockchain architectures. https://mlsdev.com/blog/156-how-to-build-your-own-blockchain-architecture. Accessed: 23/12/2021.

[2] Harsh Desai, Kevin Liu, Murat Kantarcioglu, and Lalana Kagal. Adjudicating violations in data sharing agreements using smart contracts. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1553–1560. IEEE, **2018**.

[3] Zhenyu Guan, Hanzheng Lyu, Dawei Li, Yiming Hei, and Tongchen Wang. Blockchain: a distributed solution to uav-enabled mobile edge computing. *IET Communications*, 14(15):2420–2426, **2020**.

[4] Tanzeela Sultana, Ahmad Almogren, Mariam Akbar, Mansour Zuair, Ibrar Ullah, and Nadeem Javaid. Data sharing system integrating access control mechanism using blockchain-based smart contracts for iot devices. *Applied Sciences*, 10(2):488, **2020**.

[5] ML Cummings. Operator interaction with centralized versus decentralized uav architectures. *Handbook of Unmanned Aerial Vehicles*, pages 977–992, **2015**.

[6] İhsan Yayla. Application of stanag 4586 standard for turkish aerospace industries uav systems. In *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*, pages 1–22. IEEE, **2013**.

[7] Ruinian Li, Tianyi Song, Bo Mei, Hong Li, Xiuzhen Cheng, and Limin Sun. Blockchain for large-scale internet of things data storage and protection. *IEEE Transactions on Services Computing*, 12(5):762–771, **2018**.

[8] Deepak Puthal, Nisha Malik, Saraju P Mohanty, Elias Kougianos, and Gautam Das. Everything you wanted to know about the blockchain: Its promise, components, processes, and problems. *IEEE Consumer Electronics Magazine*, 7(4):6–14, **2018**.

[9] Uav ranges. `https://www.airforce-technology.com/features/featurethe-top-10-longest-range-unmanned-aerial-vehicles-uavs/`. Accessed: 22/12/2021.

[10] Shuai Wang, Yong Yuan, Xiao Wang, Juanjuan Li, Rui Qin, and Fei-Yue Wang. An overview of smart contract: architecture, applications, and future trends. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 108–113. IEEE, **2018**.

[11] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, **2008**.

[12] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186. **1999**.

[13] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19(1), **2012**.

[14] Jae Kwon. Tendermint: Consensus without mining. *Draft v. 0.6, fall*, 1(11), **2014**.

[15] David Schwartz, Noah Youngs, Arthur Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5(8):151, **2014**.

[16] Tejasvi Alladi, Vinay Chamola, Nishad Sahu, and Mohsen Guizani. Applications of blockchain in unmanned aerial vehicles: A review. *Vehicular Communications*, 23:100249, **2020**.

[17] Anik Islam and Soo Young Shin. Buav: A blockchain based secure uav-assisted data acquisition scheme in internet of things. *Journal of Communications and Networks*, 21(5):491–502, **2019**.

[18] Haitao Xu, Wentao Huang, Yunhui Zhou, Dongmei Yang, Ming Li, and Zhu Han. Edge computing resource allocation for unmanned aerial vehicle assisted mobile network with blockchain applications. *IEEE Transactions on Wireless Communications*, 20(5):3107–3121, **2021**.

[19] Thomas Lagkas, Vasileios Argyriou, Stamatia Bibi, and Panagiotis Sarigiannidis. Uav iot framework views and challenges: Towards protecting drones as "things". *Sensors*, 18(11):4015, **2018**.

[20] Meng Li, F Richard Yu, Pengbo Si, Ruizhe Yang, Zhuwei Wang, and Yanhua Zhang. Uav-assisted data transmission in blockchain-enabled m2m communications with mobile edge computing. *IEEE Network*, 34(6):242–249, **2020**.

[21] Vishal Sharma, Ilsun You, Dushantha Nalin K Jayakody, Daniel Gutierrez Reina, and Kim-Kwang Raymond Choo. Neural-blockchain-based ultrareliable caching for edge-enabled uav networks. *IEEE Transactions on Industrial Informatics*, 15(10):5723–5736, **2019**.

[22] Abdullah Ayub Khan, Zaffar Ahmed Shaikh, Asif Ali Laghari, Sami Bourouis, Asif Ali Wagan, and Ghulam Ali Alias Atif Ali. Blockchain-aware distributed dynamic monitoring: A smart contract for fog-based drone management in land surface changes. *Atmosphere*, 12(11):1525, **2021**.

[23] Moayad Aloqaily, Ouns Bouachir, Azzedine Boukerche, and Ismaeel Al Ridhawi. Design guidelines for blockchain-assisted 5g-uav networks. *IEEE Network*, 35(1):64–71, **2021**.

[24] Shuyun Luo, Hang Li, Zhenyu Wen, Bin Qian, Graham Morgan, Antonella Longo, Omer Rana, and Rajiv Ranjan. Blockchain-based task offloading in drone-aided mobile edge computing. *IEEE Network*, 35(1):124–129, **2021**.

[25] Hsun Chao, Apoorv Maheshwari, Varun Sudarsanan, Shashank Tamaskar, and Daniel A DeLaurentis. Uav traffic information exchange network. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3347. **2018**.

[26] Alexander Kuzmin and Evgeny Znak. Blockchain-base structures for a secure and operate network of semi-autonomous unmanned aerial vehicles. In *2018 IEEE International conference on service operations and logistics, and informatics (SOLI)*, pages 32–37. IEEE, **2018**.

[27] Jennifer S Raj. Security enhanced blockchain based unmanned aerial vehicle health monitoring system. *Journal of ISMAC*, 3(02):121–131, **2021**.

[28] Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach Dinh Le, Xin Xia, Yang Feng, Zhenyu Chen, and Baowen Xu. Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering*, 47(10):2084–2106, **2019**.

[29] Zhenyu Guan, Hanzheng Lyu, Dawei Li, Yiming Hei, and Tongchen Wang. Blockchain: A distributed solution to uav-enabled mobile edge computing. *IET Communications*, 14(15):2420–2426, **2020**.

[30] Anik Islam and Soo Young Shin. Bus: A blockchain-enabled data acquisition scheme with the assistance of uav swarm in internet of things. *IEEE Access*, 7:103231–103249, **2019**.