INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2026, VOL. 72, NO. 4, PP. 1-6
Manuscript received June 02, 2026; revised October 2026. doi: 10.24425/ijet.2026.6325

Blockchain-based application model for a communication-relay-controlled drone swarm

Andrii Bondariev, Ivan Horbatyi, and Ivan Maksymiv

***Abstract*—A model of the mechanism for returning drones to a network swarm, which allows ensuring the reliability of received commands and the correctness of transitions between system states by using a distributed event log with cryptographic verification is proposed and validated via simulation.**

***Keywords*—UAV; communication; relay drone; blockchain; Matlab**

I. Introduction

ONE of the key challenges of a modern autonomous interaction system for unmanned aerial vehicles (UAVs) is to ensure stable collective behavior in conditions of limited or intermittent interference-resistant communication [1]. Temporary loss of communication with individual drones due to terrain or electronic warfare (EW) makes it impossible to synchronize the command and the current state. A UAV that goes offline, even for a short time, carries the risk of executing outdated commands, re-assignment, or erroneous interaction with other devices, which reduces the effectiveness of the swarm or leads to its destabilization. Traditional resynchronization methods are installed on trusted nodes or servers with centralized data storage [2], which creates vulnerabilities associated with a substitute command, the creation of fake network elements, or the compromise of relay systems. Therefore, the use of distributed event logs (Distributed Ledger), which make it possible to verify commands regardless of the source of their transmission, is promising. In this paper, a mathematical model of UAV return to compliance via a repeater with cryptographic verification of the event log based on a blockchain-like structure is purposed. The principle of log formalization, the synchronization algorithm after loss of connection were shown, the structural model of interaction between UAV, swarm and relay drone was described and the model verification and application in Matlab were carried out.

II. Overview

Modern approaches to the organization of UAV swarm management involve decentralized decision-making models, where control is performed without a centralized control point. Such swarms are able to autonomously coordinate spatial placement, target distribution and routes, perform reconnaissance, destruction or related functions (retransmission, detection of electronic warfare).

However, the main technical problem is the stability of data exchange in conditions of deterioration of radio communication as a result, in particular, of the influence of radio interference devices. Under such conditions, significant delays, interruptions of communication or data distortion can cause:

- conflicts during the performance of assigned tasks;
- deterioration of interaction between drones;
- complete loss of control over part of the swarm.

Therefore, there is a need for a mechanism that ensures synchronous updating of system states, especially for devices that have been outside the network for some time. One of the promising solutions is the implementation of blockchain-like structures on relay drones.

A feature of using blockchain for these tasks is that a drone swarm is a dynamic system, so the verification mechanism should be as resource-efficient as possible and, as a result, may contain certain compromises to ensure high communication speed and reduce decision-making time.

One of the potential mechanisms for implementing blockchain verification of information exchange between swarm components is Permissioned Proof-of-Authority (PoA) and Practical Byzantine Fault Tolerance (PBFT).

Permissioned Proof-of-Authority is a consensus mechanism for private ledgers, in which the role of validators is performed by previously known and identified nodes (authorities). In the PoA scheme, blocks are created by a limited set of validators in a connected mode (round-robin or leader election), after which other nodes only validate the signature and add the block to the local registry. This architecture allows for low write duration and high throughput, making PoA suitable for resource-constrained edge nodes (e.g., relay drones). However, PoA is permissioned and relies on trust in validators, so it is advisable to combine PoA with multi-sig mechanisms or periodic anchoring to increase compromise resistance [3, 4].

Practical Byzantine Fault Tolerance (PBFT) is a replication algorithm that provides consensus in the presence of Byzantine nodes under the condition ($n>3f$) (the system can withstand up to (f) malicious nodes). PBFT implements a sequence of pre-prepare, prepare, and commit phases. A quorum of messages should be obtained in each phase to accept a block. PBFT provides formal guarantees of correctness and stability, but has a quadratic communication complexity ($O(n^2)$) and therefore is appropriate primarily for small permissioned networks (for example, between relay drones). In “drones ↔ relay drones”

Andrii Bondariev and Ivan Maksymiv are with Lviv Polytechnic National University, Ukraine (e-mail: andrii.p.bondariev@lpnu.ua, ivan.p.maksymiv@lpnu.ua).

Ivan Horbatyi is with Lviv Polytechnic National University, Ukraine (e-mail: giv@polynet.lviv.ua).

© The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/), which permits use, distribution, and reproduction in any medium, provided that the Article is properly cited.

systems, PBFT is appropriate to be used between relay drones, leaving the drones to play the role of light clients that verify signatures or Merkle proofs. [5, 6].

Taking into account these features and the specified blockchain construction mechanisms, a mathematical model of drone synchronization in a "swarm" network using relay drones was created.

III. Mathematical Model

This section presents a formal mathematical model of the "reconnaissance swarm – relay drones" system with a distributed ledger. The goal is to obtain analytical expressions for the probability of correct drone recovery after loss of communication, synchronization time, and channel load estimation during recovery. The mathematical model is constructed in such a way that it can be directly used in simulation modeling using software tools (Matlab) or for estimating Proof of Concept (PoC) parameters.

Let us imagine a typical situation when one of the drones in the swarm loses communication with the network, then restores communication. During the period of absence of the UAV in the network, its configuration has changed and the drone needs to update data about the current situation (Fig. 1).

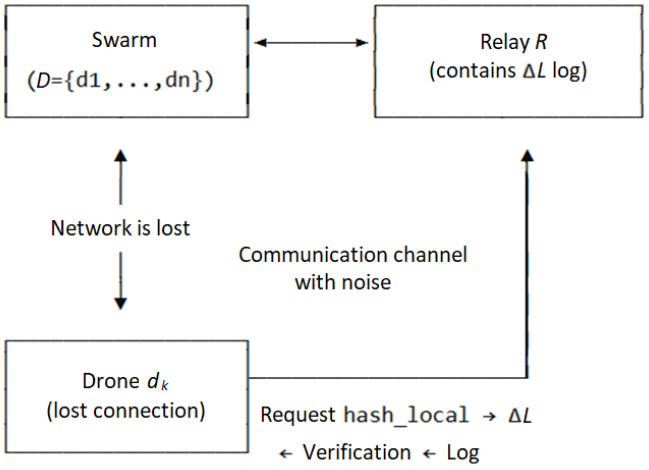


Fig. 1. Block diagram of a relay drones-based swarm communication system

Definitions:

$D = \{d_1, \ldots, d_n\}$ – a set of drones in a swarm, n – swarm size.
$R = \{r_1, \ldots, r_m\}$ – a set of relay drones. In a simple model $m \geq 1$.
L – distributed event log (ledger).
Tx_i – transaction (event) with index i.
B_j – a block in a chain containing a set of transactions.
$H(B_j)$ – block hash (B_j).
$hash(\cdot)$ – cryptographic hash function (e.g. SHA-256).
$sig_k(\cdot)$ – digital signature of the drone/node k.
P_l – the probability of loss of communication (during EW) for a specific drone in a period of time.
T_r – connection recovery time (local random variable, with parameter λ recovery intensity).
ΔL – the volume of missed events (number of transactions or blocks) that need to be transmitted to the drone during synchronization.
s_{tx} – average size of one transaction in bytes.
s_b – average block size (bytes). If a series of blocks are being transmitted, the total size $|\Delta L| \cdot s_{tx}$ or $|\Delta B| \cdot s_b$.
C – available channel bandwidth between the repeater and the drone (bytes/s) at the time of recovery.
t_{ver} – average signature/hash verification time for one transaction on drone hardware (sec).
α – protocol overhead rate (protocol headers, re-request, CRC etc.), $(0 < \alpha < 1)$.

Remarks:

1. The formation of transactions in the swarm is implemented as a Poisson process with intensity μ (events/sec).

2. Repeaters are more reliable nodes because they are more resilient and have more power than typical drones – meaning they retain short-term history L_{cache} .

3. Consensus is implemented as Permissioned PoA or PBFT-similar protocol between repeaters. For drones, signature verification and hash chain verification are sufficient. Therefore, the consensus duration is taken into account as a fixed value in the model (or related to the number of rlays m).

4. The channels between drones and repeaters are characterized by average C and the probability of packet loss p (depends from EW). For simplicity, independent packet losses are assumed.

Formalization of events and block structure

The transaction (event) has the form:

$$Tx = \langle t, cmd, origin, sig_{origin}, meta \rangle, \quad (1)$$

where t – time stamp, cmd – command/state, $origin$ – initiator identifier, $sig_{origin} = sig_{origin}(hash(Tx))$ – digital signature, $meta$ – additional data (coordinates, role, priority).

Transactions/blocks hash chain:

$$H_i = hash(Tx_i \,\|\, H_{i-1}) \quad (2)$$

or in a block:

$$H(B_j) = hash(header(B_j) \,\|\, H(B_{j-1})). \quad (3)$$

Drone d_k has a local last known hash at some point ($H_{local} = H_p$). After the channel is restored, it queries the repeater and receives a sequence of blocks $\{B_{p+1}, \ldots, B_q\}$ or equivalent sequence of transactions ΔL.

Probability of correct recovery ($P_{success}$)

Let's identify two main factors that influence successful recovery:

1. Will it be possible to deliver ΔL through channel (bandwidth and packet loss problem)?

2. Are signatures/hashes verified (only verified transactions will be accepted)?

Let the total size ΔL in bytes – $S_\Delta = |\Delta L| \cdot s_{tx}$ (or $|\Delta B| \cdot s_b$). Transmission time excluding overhead (protocol overhead, computational overhead, etc.):

$$t_{tx} = S_\Delta / C. \quad (4)$$

Including overhead costs α:

$$t_{tx}^{eff} = \frac{S_\Delta}{C} \cdot 1 + \alpha \quad (5)$$

The parameter of transmission failure due to EW/noise represents the conditional probability that during transmission at least one key packet will be lost and will not be retransmitted successfully within the allowable time window τ. To simplify the model, if retransmission mechanisms (ARQ) are used with efficiency η ($0< \eta \leq 1$), equivalent probability of transmission success ΔL:

$$P_{tx_succ} = e^{-\beta\, t_{tx}^{eff}}, \quad (6)$$

where the parameter β characterizes the intensity of channel degradation (related to p, the intensity of EW); this choice of the exponential form gives convenient analytical properties.

In parallel, signature verification has a probability of success P_{ver}, which depends on hardware errors. Since the probability of hardware error is negligible, it can be considered $P_{ver} \approx 1\text{-}\varepsilon$, where ε – very small value (verification error/hardware failure). Hence the overall probability of correct recovery:

$$P_{success} = P_{tx_succ} \cdot P_{ver}. \quad (7)$$

By substituting P_{tx_succ}:

$$P_{success} = e^{-\beta \frac{S_\Delta}{C}(1+\alpha)} \cdot (1-\varepsilon). \quad (7)$$

Therefore, $P_{success}$ decreases exponentially with increasing ΔL or decreasing C. Decreasing α (protocol optimization) and increasing C increase the probability of correct recovery.

Expected full return time (Latency)

The total drone return time to active tasks consists of three components:

$$T_{return} = T_{handshake} + t_{tx}^{eff} + t_{ver_total}, \quad (8)$$

where

$T_{handshake}$ – handshake/authentication time (usually a small constant, includes certificate verification and key authentication).

$t_{tx}^{eff} = (S_\Delta/C)(1+\alpha)$ – transmission time (defined above).

$t_{ver_total} = |\Delta L| \cdot t_{ver}$ – total time to verify all transactions (or blocks).

Therefore,

$$T_{return} = T_{handshake} + (S_\Delta/C)(1+\alpha) + |\Delta L| \cdot t_{ver}. \quad (9)$$

If parallel verification hardware acceleration is used then t_{ver_total} can be modeled as $(|\Delta L| \cdot t_{ver})/g$, where g – parallelism factor.

Missed event count model ΔL

If events are generated by a Poisson process with intensity μ (events/sec) and the drone lost connection for a while T_r, therefore the mathematical expectation of the number of missed events:

$$M[|\Delta L|] = \mu M[T_r]. \quad (10)$$

If T_r exponentially distributed with parameter λ (i.e. average time $M[T_r] = 1/\lambda$), then:

$$M[|\Delta L|] = \mu/\lambda. \quad (11)$$

Average data volume:

$$M[S_\Delta] = s_{tx} \cdot M[|\Delta L|] = \mu s_{tx}/\lambda. \quad (12)$$

Substituting into the expression for $P_{success}$, we obtain the average probability of recovery in terms of intensities:

$$M[P_{success}] = exp\left(-\beta \frac{\mu s_{tx}}{\lambda\, C}(1+\alpha)\right) \cdot (1-\varepsilon). \quad (13)$$

This expression allows us to evaluate key dependencies:

- increasing the frequency of events μ reduces the probability of success (higher ΔL);
- faster recovery ($\lambda \uparrow$) increases success;
- higher throughput C – positively affects the probability of success.

Relay system load and scalability assessment

If in the system simultaneously k drones are restored (queue for synchronization), then the total input traffic to the relay drone is:

$$S_{tot} = \sum_{i=1}^{k} S_{\Delta,i} \quad (14)$$

Time until the relay drone will service the queue (assuming serial service):

$$T_{queue} = \sum_{i=1}^{k} \frac{S_{\Delta,i}}{C_r}(1+\alpha_r), \quad (15)$$

where C_r – effective relay bandwidth per client, α_r – its overhead costs.

Assuming the same characteristics for drones $S_{\Delta,i} \equiv \bar{S}_\Delta$, then this time will be:

$$T_{queue} = k \cdot \frac{\bar{S}_\Delta}{C_r}(1+\alpha_r). \quad (16)$$

To avoid significant latency, it is desirable to cut off simultaneous servicing of too large k and introduce prioritization (e.g., score based – critical drones).

So, for systems with large n, practical solutions include:

- increasing the number of relay drones m (reducing the k for one relay drone);
- differential updates (transmitting only headers/transactions of the critical level);
- using Bloom-filter / Merkle-proofs to quickly determine the desired transactions, which significantly reduces $S\Delta$.

Hacking Resistance

One way to influence the drone network is to spoof the log through compromised relays or rogue nodes that report false ΔL_s. Therefore, to minimize such influence, the following conditions must be met:

1. Mandatory cryptographic signature verification – the transaction is accepted only if the signature is valid.
2. Checking the hash chain $H(B_j)$ – sequence integrity.
3. Requiring multi-signatures or qualifications from multiple relays (threshold signatures).

Let us denote f – the fraction of fake (broken) repeaters among all authorized nodes. If PoA with m relay drones is used, then forgery requires that the attacker control at least τ repeaters, where τ is a threshold that depends on the specific protocol (for example, in PBFT $\tau = (m+1)/3$ for fault tolerance f).

Therefore, the probability of compromise in a random hack is:

$$P_{comp} = \sum_{i=\tau}^{m} \binom{m}{i} f^i (1-f)^{m-i}. \quad (17)$$

From this model we can conclude that for system reliability it is desirable to provide P_{comp} as small as possible (for example $< 10^{-6}$). This value depends on both the number of repeaters and the acceptable risk of losing a portion f of the total number of relay drones.

Analytical Model Validation

Due to the lack of open experimental datasets for combat or quasi-combat scenarios with the influence of electronic warfare, model validation is carried out using a combined approach, which includes:

- Analytical verification of the correctness of the model.
- Comparison of model results with known characteristics of real UAV networks given in modern scientific publications.

Evidence from a review of blockchain integration in drone networks shows that using mechanisms such as Practical Byzantine Fault Tolerance (PBFT) can maintain a consistent ledger in dynamic networks with moving nodes, although performance and scalability depend on the number of nodes and channel parameters; delays remain stable up to a certain limit of the number of nodes, after which they increase [7].

The DTPBFT (Dynamic, High-Confidence PBFT for UAV Swarm) study showed experimental data: at 200 nodes, the algorithm-maintained consensus for ~0.24 (sec) and demonstrated high resistance to some malicious nodes (> 1/3), which is consistent with theoretical expectations of BFT mechanisms for permissioned networks [8].

These results support the hypothesis that mechanisms similar to those proposed in our model (PoA/PBFT hybrids) can be effective in real-world scenarios with a large number of nodes.

IV. Simulation-Based Validation of the Mathematical Model

To further verify the adequacy of the proposed analytical model and evaluate system robustness under realistic conditions, a computer simulation was conducted in the MATLAB [9] environment.

The simulation aims to validate analytical expressions for probability of successful recovery $P_{success}$ and expected return time T_{return}.

Also, it analyzes the impact of communication degradation (parameter β), swarm size n, event intensity μ and evaluates the system resilience to node failures and Byzantine behavior.

Simulation Model Description

A discrete-time simulation model was implemented, where each drone is represented as an autonomous agent with state transitions:

- Active – connected to the swarm;
- Disconnected – temporarily lost connection;
- Recovering – synchronizing with relay drones;
- Rejoined – successfully authenticated and synchronized.

The following processes were simulated:

1) Disconnection process – modeled as a stochastic process with probability P_l.
2) Recovery process – recovery time T_r was generated according to an exponential distribution with parameter λ.
3) Event generation – transactions were generated according to a Poisson process with intensity μ.
4) Data synchronization – the number of missed transactions:
$$| \Delta L | \sim \text{Poisson}(\mu T_r) \quad (18)$$
5) Transmission process – transmission success was simulated using:
$$P_{tx_succ} = e^{-\beta t_{tx}^{eff}} \quad (19)$$
6) Consensus modeling – PoA (constant delay) and PBFT (delay increases with number of relay nodes m, approximated as T_{PBFT})

The Monte Carlo method was used to simulate stochastic processes in the proposed model, including recovery time, event generation, and transmission success. Due to the probabilistic nature and nonlinear dependencies of these processes, analytical evaluation is insufficient, making Monte Carlo simulation an appropriate tool for model validation and performance analysis.

The parameter values used in the simulation correspond to typical ranges observed in UAV communication and embedded systems. Specifically, transaction size was estimated based on typical blockchain payload structures (150–300 bytes), protocol overhead was assumed within 10-30%, and verification latency corresponds to lightweight cryptographic operations on embedded hardware (Fig. 2). Channel degradation parameter β was treated as an abstract representation of interference intensity and is varied within a realistic range to analyze system sensitivity.

The 3D surface plot illustrates the combined influence of channel degradation and throughput on the recovery success probability. The results demonstrate that increasing throughput does not eliminate the impact of interference but reduces exposure time, thereby improving recovery probability. A critical region can be observed where moderate degradation and insufficient bandwidth lead to a sharp drop in system performance (Fig. 3).

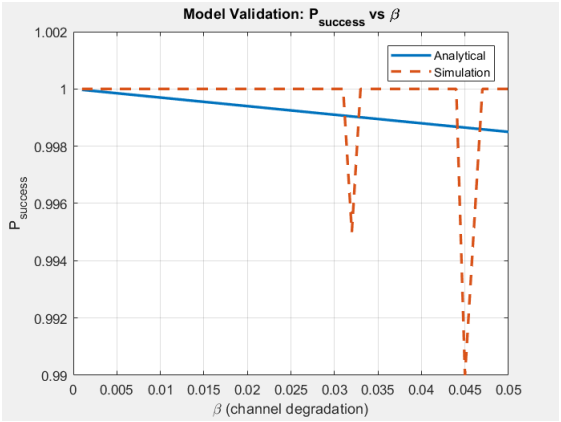


Fig. 2. Probability of successful recovery $P_{success}$ versus channel degradation parameter β: analytical (solid) and simulation (dashed)

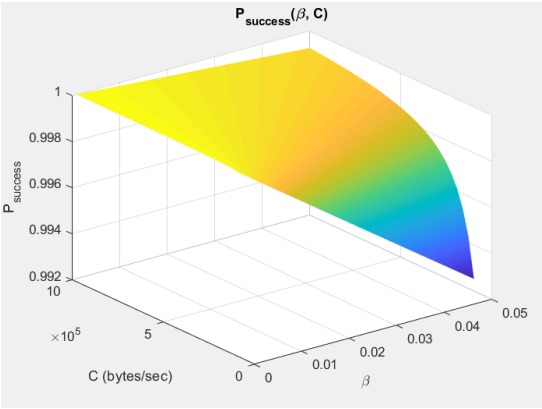


Fig. 3. Combined impact of channel degradation β and throughput C on the probability of successful recovery $P_{success}$

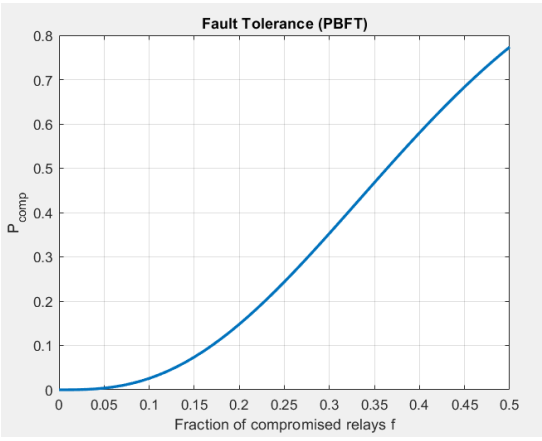


Fig. 4. Probability of system compromise P_{comp} versus fraction of malicious relay nodes f in a PBFT consensus model

The fault tolerance analysis was performed by evaluating the probability of system compromise P_{comp} for varying fractions of malicious relay nodes f (Fig. 4). For each value of f, the probability that the number of compromised nodes exceeds the PBFT threshold is computed using the binomial distribution. The threshold was defined as $\tau = \lceil (m+1)/3 \rceil$, where m - the number of relay nodes. The resulting probability was obtained by summing over all cases where the number of compromised nodes was greater than or equal to τ.

The conducted simulation study confirms the adequacy of the proposed mathematical model and demonstrates its applicability for analyzing UAV swarm synchronization under adverse conditions. The results show close agreement between analytical and Monte Carlo estimates, validating the correctness of the derived expressions. Furthermore, the fault tolerance analysis highlights the threshold behavior of PBFT-based systems, while the parametric study reveals the trade-off between channel degradation and throughput. Overall, the simulation framework provides an effective tool for evaluating system performance and robustness in realistic operating scenarios.

V. Parametric Analysis of Swarm Size and Relay Node Density

To further evaluate the scalability and practical applicability of the proposed model, a parametric analysis of the UAV swarm was conducted with respect to the number of drones in the swarm and the number of relay nodes. The goal of this analysis is to assess how system performance, particularly the recovery latency of a disconnected drone, depends on the network size and relay infrastructure density.

In real-world UAV swarm deployments, the number of active drones may vary significantly depending on the mission scenario, while the number of relay nodes is typically limited due to hardware cost, energy consumption, and deployment constraints. Therefore, it is important to understand how these parameters affect synchronization efficiency and whether the system remains stable under increased load.

The analysis is based on the previously derived analytical model for recovery time (9) where the number of missed events $|\Delta L|$ depends on the event generation intensity and the recovery time. In this extended model, the effective event intensity is assumed to increase proportionally with the swarm size, reflecting the fact that a larger number of drones generates more events in the distributed ledger.

To incorporate the impact of relay node load, an additional scaling factor is introduced. Since relay drones are responsible for serving multiple clients simultaneously, the effective load per relay node increases with the ratio n/m where n is the number of drones and m is the number of relay nodes. This leads to increased communication delay due to queueing and bandwidth sharing effects. In the simplified model, this is captured by a linear load factor applied to the recovery time.

The simulation was implemented in Matlab and evaluates the recovery time across a range of swarm sizes and relay node configurations. The number of drones n varies from 10 to 200, while the number of relay nodes m takes values from a predefined set (e.g., 3, 5, 7, and 10). For each configuration, the recovery time is calculated based on the analytical expressions, adjusted by the relay load factor.

The obtained results (Fig 5) demonstrate several important trends. First, the recovery time increases with the number of drones in the swarm, which is expected due to the higher volume of synchronization data and increased communication load. Second, increasing the number of relay nodes significantly mitigates this effect, leading to lower recovery latency even for large swarm sizes. This confirms that relay node density is a critical parameter for maintaining system performance.

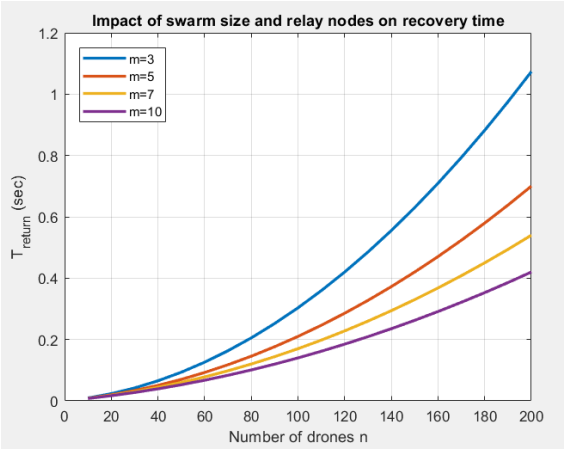


Fig 5. The impact of drone swarm size and number of relay nodes on the drone recovery time

Additionally, the results indicate the presence of a quasi-linear growth region for moderate swarm sizes, followed by a more rapid increase in recovery time when the relay nodes

become overloaded. This behavior highlights the importance of proper system dimensioning and supports the need for adaptive relay deployment strategies.

Overall, the parametric analysis complements the previously obtained results by providing insights into system scalability and resource allocation. It demonstrates that the proposed model can be effectively used not only for performance evaluation but also for guiding the design of UAV swarm communication architectures under realistic constraints.

CONCLUSIONS

The paper proposes a mathematical model for implementing the process of synchronization and verifying requests in a swarm drone network followed by returning UAVs to the swarm using a repeater with cryptographic verification of the event log based on a blockchain-like structure.

For the first time the Proof-of-Authority (PoA) and Practical Byzantine Fault Tolerance (PBFT) were used as hybrid blockchain mechanism for autonomous drone swarm.

A Monte Carlo simulation was implemented in MATLAB to validate the analytical expressions of the proposed model. The simulation reproduces stochastic processes of disconnection, recovery, and data synchronization. The obtained results demonstrate close agreement between analytical and simulated values, with deviation not exceeding 5%, confirming the adequacy of the model.

To evaluate the scalability of the proposed system, a parametric analysis was conducted by varying the number of drones in the swarm and the number of relay nodes. The recovery time was calculated based on the analytical model, taking into account increased load on relay nodes as the swarm size grows.

The results show that increasing the number of relay nodes significantly improves system performance, especially for large swarms. Conversely, insufficient relay density leads to increased synchronization delays due to higher communication load per relay node.

This analysis demonstrates the importance of balancing swarm size and relay infrastructure in order to maintain acceptable recovery latency.

REFERENCES

[1] Bobalo, Y., Horbatyi, I. “Interference-Protected Radio Communication System Using a Repeater”, Conference Proceedings 2024 IEEE 17th International Conference on Advanced Trends in Radioelectronics Telecommunications and Computer Engineering Tcset 2024, 2024, pp. 133–136. doi: 10.1109/TCSET64720.2024.10755662

[2] Z. Wang, J. Li, C. Liu, Y. Xu, and X. Tan, “A decentralized decision-making algorithm of UAV swarm with information fusion strategy,” Expert Systems with Applications, vol. 236, 2023. doi: 10.1016/j.eswa.2023.121306

[3] Q. Wang et al., “Exploring Unfairness on Proof of Authority,” arXiv preprint arXiv:2203.03008, 2022. [Online]. Available: https://arxiv.org/abs/2203.03008

[4] M. A. Manolache, “Decision Making using the Blockchain Proof of Authority,” Procedia Computer Science, vol. 216, pp. 123–130, 2022.

[5] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” in 3rd Symposium on Operating Systems Design and Implementation (OSDI ’99), New Orleans, LA, USA, Feb. 1999.

[6] C. Berger, H. P. Reiser, A. Bessani, “Making Reads in BFT State Machine Replication Fast, Linearizable, and Live,” arXiv preprint, Jul. 2021.

[7] E. Aloquaily et al., “Blockchain Integration in UAV Networks: Performance Metrics and Analysis,” Sensors, vol. 24, 23, 2024, Art. no. 7813, doi:10.3390/s24237813.

[8] Pengbin Han et al., “DTPBFT: A dynamic and highly trusted blockchain consensus algorithm for UAV swarm,” Computer Networks, vol. 250, 2024, Art. no. 110602, doi:10.1016/j.comnet.2024.110602

[9] Serhii Altunin, Andrii Bondariev, Ivan Maksymiv “Simulation of Artillery Ballistic Station”, Proceedings - IEEE 17th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv – 2024. – P. 68-71. doi:10.1109/TCSET64720.2024.10755634