2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3315-4393-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IROS60139.2025.11247266

Analysis and Mitigation of Inconsistencies in Blockchain-Enabled Robot Swarms

Giada Simionato[1†], Volker Strobel[2†], Mario G.C.A. Cimino[1], and Marco Dorigo[2]

***Abstract*— Recent research has demonstrated that blockchain-enabled robot swarms—where robots coordinate using blockchain technology—can secure robot swarms by neutralizing malicious and malfunctioning robots. This security is achieved through blockchain technology's consistency properties. However, prior work addressed malfunctions at the information level, that is, it studied how to neutralize robots that stored information in the blockchain that did not correspond to the real-world state (i.e., it studied the oracle problem). In contrast, this study focuses on inconsistencies at the blockchain protocol level. We analyze how network partitions, which may arise from robots' local-only communication capabilities, malfunctioning hardware, or external attacks, can lead to inconsistent information in a robot swarm. In order to mitigate these disruptions, we propose a decentralized approach to detect partitions and a corresponding response. We study our approach in a swarm robotics simulator, where we demonstrate its effectiveness in reducing blockchain inconsistencies.**

I. INTRODUCTION

Swarm robotics studies how multiple simple robots can coordinate to complete tasks that are beyond the capability of individual robots [1], [2]. Through decentralized control and local interactions, it seeks to ensure flexibility, scalability, and robustness in multi-robot systems. Although robot swarms were often assumed to be resistant to the influence of Byzantine robots—robots showing discrepancies between their intended behavior and their actual behavior—this assumption has been proven wrong [3].

Recent work has established that blockchain technology [4], [5] can secure robot swarms by effectively neutralizing Byzantine robots [3], [6], [7]. Besides identifying Byzantine robots, such blockchain-enabled robot swarms also have the potential to enable robot economies, improve data consistency, and incorporate rules for open robot swarms that decide which robots may join or must leave the system [8]. In blockchain-enabled robot swarms, each robot acts as a node in a peer-to-peer blockchain network, maintained exclusively by the robots, where coordination is achieved through blockchain-based smart contracts—programming code executed on data stored in the blockchain. These smart contracts enable the decentralized network to verify and agree on computations while governing how robots interact, coordinate, and respond to different situations. The robots share and store information by creating blockchain transactions, which smart contracts then process according to predefined rules and logic.

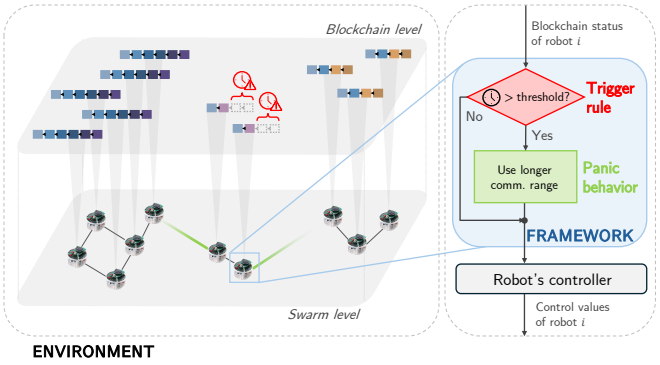

Fig. 1. In our setup, each robot performs a random walk while acting as a node in a blockchain network maintained by the robots. The random movement and local-only communication of the robots can lead to network partitioning, which causes inconsistencies in blockchain data due to delays in block propagation. When the framework detects that a robot's blockchain data is potentially inconsistent (based on the latest block's timestamp, as shown by the two robots in the center), it triggers a panic behavior: the robot temporarily increases its communication range to increase the probability of synchronizing with the latest blockchain data.

To perform the required tasks, the robot swarms move and disperse throughout their environment. Given their limited communication capabilities, only nearby robots can interact, leading to the formation of clusters, or *partitions*, of connected robots within the swarm [9], [10]. These blockchain-enabled robots independently maintain their version of the blockchain, proposing and validating new blocks only with robots that are within their communication range. As a result, each partition develops its own blockchain version, which remains consistent among the robots within that partition but differs from those of other partitions. This *inconsistency* in blockchain states across partitions can lead to conflicts, such as data discrepancies or overwriting, when robots with different chains reconnect and synchronize. The longer the partitions persist, the greater the inconsistencies become. The issue becomes especially critical when the swarms must act on blockchain data rather than merely store it—acting on inconsistent information can severely compromise the mission of the swarm.

Previous research has addressed Byzantine faults at the *smart contract level*—specifically focusing on the oracle problem of verifying that blockchain-stored information corresponds to real-world conditions [11]—where such faults can only be neutralized if the security properties of the

† These authors contributed equally.
[1]Department of Information Engineering, University of Pisa, Pisa 56122, Italy. giada.simionato@phd.unipi.it, mario.cimino@unipi.it
[2]IRIDIA, Université Libre de Bruxelles, Brussels, Belgium volker.strobel@ulb.be, mdorigo@ulb.ac.be

underlying blockchain protocol are respected. However, in the event of swarm partitioning, this may no longer hold, undermining the advantages of integrating blockchain technology into the swarm. In light of this, we study Byzantine faults at the *blockchain protocol level*, particularly analyzing how swarm partitioning and local communication can lead to inconsistencies in a blockchain's information state, compromising both the liveness and security of blockchain-enabled robot swarms.

We focus our analysis on blockchain-enabled robot swarms using the Proof-of-Authority (PoA) consensus protocol, as this was used in most previous research and was shown to be suitable for execution on real robots [7], [12]. In addition to examining how partitions introduce inconsistencies in blockchain-enabled robot swarms, we analyze their impact on this widely used consensus protocol. Partitioning can disrupt PoA, potentially halting or delaying blockchain growth, which in turn prevents the timely dissemination of new information. This issue is especially critical for time-sensitive tasks, such as victim localization in search-and-rescue missions.

To address the challenges posed by partitions, robots require mechanisms to detect and resolve severe partitioning. To this end, we propose a framework that can be integrated into existing blockchain-enabled robot swarms to reduce the likelihood of inconsistencies and deadlocks in the blockchain. The framework operates on each robot individually, making it easy to add to the swarm controller, and consists of two modules: *(i)* the *trigger rules*, a set of rules that assess partition severity based on blockchain status and initiate a change of behaviors, and *(ii)* the *panic behaviors*, which adjust the robot's control values to encourage partition reconciliation. Figure 1 illustrates how the framework works.

We evaluated the framework's performance using the ARGoS robot swarm simulator [13], demonstrating how it mitigates the negative impact of partitions on the robots' blockchains. The framework effectively improves consistency, thereby enhancing the overall security of the swarms.

II. BLOCKCHAIN PRELIMINARIES

Blockchain technology enables peer-to-peer networks to achieve consensus on data without the need for mutual trust among the peers (nodes) of the network [4]. Specifically, a blockchain is a decentralized database consisting of a sequence of *blocks*, starting with the common *genesis* block. Each block contains *transactions*, created by nodes, which store arbitrary data, and a cryptographic hash pointer to its previous block in the sequence. Therefore, altering any block invalidates all subsequent ones, making the blockchain relatively tamper-proof thanks to the longest-chain rule (see below).

Blocks are added to the blockchain based on a *consensus protocol*. In the first developed consensus protocol, based on Proof-of-Work (PoW), some nodes (miners) were required to solve cryptographic puzzles using computational power to validate blocks [4]. In this work, we focus on the significantly less computationally expensive Proof-of-Authority, where a

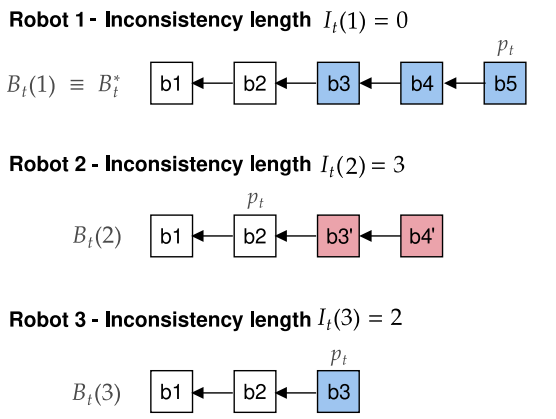

Fig. 2. The three robots have different blockchain versions. Up to block b2, all three robots have the same blockchain. With the third blocks (b3 and b3'), the blockchains diverge. Robot 1 does not have an inconsistency because it has the longest blockchain in this example. Robot 2 has an inconsistency of 3 because its common blockchain with the longest blockchain is three blocks shorter than the longest blockchain. Robot 3 has an inconsistency of length 2 because its out-of-sync blockchain is two blocks shorter than the longest blockchain of Robot 1.

selected group of N nodes (sealers) create blocks after a configured time period, called *block period* [14]. After sealing a block, a node must wait for $\lfloor \frac{N}{2} \rfloor$ additional blocks before sealing another one. A preferred sealer—that is, a sealer who is given priority to propose the next block—is chosen in a round-robin fashion for each block. While the preferred sealer has the first opportunity to propose a block, other sealers can also submit block proposals if the preferred sealer fails to do so within a specified time frame.

When multiple sealers create blocks simultaneously, conflicting blockchain versions (forks) can arise, causing *inconsistencies* due to conflicting transactions or their different order. To resolve forks, nodes compare blockchain versions using the *longest-chain rule*. This rule assigns a difficulty of 1 to blocks from non-preferred sealers and 2 to those from preferred sealers. The chain with the higher total difficulty (called the longest chain) is accepted, while the other is discarded. Due to the possibility of forks, a new block is not *final* and may be discarded based on the longest-chain rule. To address this, users (or robots) wait for a certain number of subsequent blocks, denoted as *confirmations*, before considering a block final. The user selects this number, balancing two factors: more confirmations reduce the likelihood of a block being discarded but also introduce additional delay [4].

III. RELATED WORK

Several studies employ blockchain technology to secure robotics applications [15], [16], [17], [18]. For instance, [6] proposes an information market where robots trade data via blockchain transactions, preventing malicious actors from influencing swarm decisions through smart contracts. In [12], blockchain-based coordination rules enhance swarm foraging, while [19] demonstrates consensus in swarms despite Byzantine robots. These applications naturally cause a swarm to disperse into locally connected clusters that maintain separate blockchain versions. Inconsistencies between these

blockchain versions become critical when the swarm must act on information rather than merely collect it. Acting too soon risks relying on inconsistent data, while acting too late causes delays, which are problematic in time-sensitive scenarios.

In distributed systems, the CAP theorem [20] asserts that at most two of the following three properties can be achieved at once: consistency (all honest nodes agree on the same block sequence), availability (nodes have access to all data needed to validate transactions and maintain consensus), and partition tolerance (the system continues operating correctly despite network partitions) [21], [22]. As of today, most studies in blockchain-enabled swarm robotics have focused on enhancing partition tolerance while ensuring availability [23], whereas the consistency property has been less studied. An exception is [24] that, to improve consistency, leverages the concept of Extended Virtual Synchrony (EVS) [25]. EVS provides agreement on the blockchain status only between nodes belonging to the same cluster, i.e., those that are in communication. When nodes belonging to different clusters connect, their local blockchains merge into a Directed Acyclic Graph (DAG)-based ledger called SwarmDAG, preserving all collected data. Similarly, [26] introduces Blockgraph, a DAG-based blockchain for mobile ad-hoc networks, featuring consensus, block management, and group management modules to ensure resilience to network partitions. Other works, such as [27], [28], employ IOTA, a DAG-based distributed ledger [29], to create a partition-tolerant and Byzantine-tolerant system. [27] utilizes IOTA's smart contracts for vision-based anomaly detection in multi-robot environments, while [28] integrates IOTA with ROS 2 for mapping in intermittently connected networks.

While DAG-based solutions improve partition tolerance, they have drawbacks: *(i)* inconsistent data due to unresolved conflicts; *(ii)* growing inconsistencies as clusters lack incentives to merge; *(iii)* no common "final" block, making consensus and action difficult; and *(iv)* excessive data retention, unsuitable for resource-limited agents. Our approach uses a linear blockchain and encourages partition reconciliation, ensuring consistency and actionable information.

An alternative approach is PeloPartition [30] that creates separate blockchain branches during network partitions, assigning miners to consensus groups and allowing only partitioned members to sign blocks. Though it resolves conflicts upon merging, it supports only one split and merge at a time, making it unsuitable for swarm robotics. Similarly, Komorebi [31], a DAG-based Byzantine fault-tolerant protocol, improves partition tolerance by dividing the blockchain into shards. However, since shards can still communicate, it fails in scenarios where nodes are physically isolated.

Ensuring consistency is especially critical when malicious agents deliberately create inconsistencies to execute 51% attacks, where a single entity controls most nodes, enabling blockchain takeover, double-spending, and data forgery. Various methods aim to reduce these inconsistencies and prevent such attacks [32], [33]. For instance, [34] proposes randomly selecting the node that verifies new blocks, while [35] suggests choosing a random group of miners for block proposals. Other approaches accept only blocks from different miners than the previous proposers [36] or increase chain difficulty based on miner activity [37]. Consensus rate also plays a key role, with [38] allowing users to set it and [39] requiring 60% node agreement. Other work reduces inconsistencies by optimizing blockchain parameters [40]; however, it relies on PoW, which is unsuitable for swarm robotics. These consistency-focused solutions assume connected networks. In partitioned swarms, they become impractical, as the selected nodes may not be physically present, causing block production to stall or face significant delays.

Existing blockchain-enabled swarm robotics solutions ignore the risks of agents operating on different blockchain versions, while consistency-focused approaches fail in partitioned networks. This work analyzes inconsistencies in a basic swarm scenario, highlighting their significance and proposing a framework to prevent severe inconsistencies while maintaining partition tolerance. Unlike previous methods that sacrifice either consistency or partition tolerance, our framework dynamically adjusts based on task requirements, making it adaptable to a broader range of applications.

IV. METHODS

The fundamental rationale for using a blockchain is to maintain consistent and tamper-proof information in a peer-to-peer network. Information inconsistencies—situations where different nodes maintain different information—represent one of the most critical challenges in blockchain implementations. Indeed, many security vulnerabilities in blockchains (e.g., double-spend attacks) have inconsistent information as an underlying prerequisite. In this paper, we analyze blockchain security and performance by examining the occurrence of these inconsistencies.

Consistency is particularly crucial in swarm robotics applications where robots must modify their behavior based on information stored in the blockchain. For example, in a search-and-rescue scenario, robots might need to vote via a smart contract on which victims should be rescued first. If the blockchain contains forks, different forks might contain conflicting votes: one fork could indicate person A should be rescued first, whereas another could indicate person B. This could lead to a situation where no rescue operation is successfully completed because robots are split between both locations without sufficient numbers at either to transport victims to a safe place. Similarly, if a robot is out-of-sync (i.e., has not received new blocks for an extended period), it might not receive any information about victims' locations at all.

Such inconsistencies can occur accidentally, particularly due to network delays caused by partitioning (where the robot swarm is divided into disconnected sub-swarms, preventing information flow between them) or by low bandwidth. They can also result from malicious activity, such as jamming attacks or selfish mining.

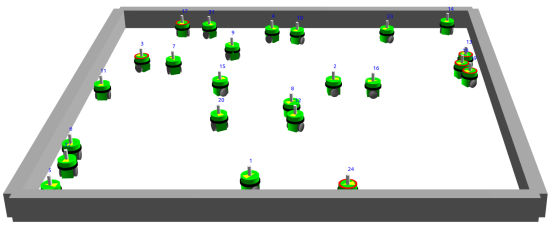

Fig. 3. In our experiments, 8, 16, or 24 robots perform a random walk in a square environment of dimension $1.9 \times 1.9\,\mathrm{m}^2$. The communication range is set to 0.1 m, leading to slow information propagation.

A. Formal definition of blockchain inconsistencies

Let z denote the number of confirmations a block b has received (i.e., z is the number of successive blocks extending b), and let z^* denote the minimum number of confirmations that are required for a block to be considered *final*. The choice of z^* yields a trade-off between speed and security: higher values create delays but make it less likely that robots act on inconsistent information. Let d be the block period of the PoA consensus protocol, set by the swarm designer (the default value in PoA is $d = 15$ s, which we also use in this work). Let $B_t(r)$ denote the blockchain of a robot r at time t and let B_t^* denote the longest blockchain at time t in the swarm.

A robot r is considered to be experiencing a severe inconsistency if, at any timestep t, $B_t(r)$ is a fork or out-of-sync blockchain that differs by at least z^* confirmations from the longest blockchain in the swarm (for a graphical representation, see Figure 2).

In order to calculate this difference, let p_t be the last common block of $B_t(r)$ and B_t^*. The inconsistency length the robot r is experiencing at this timestep t is then $I_t(r) = \text{length}(B_t^*) - \text{block\_number}(p_t)$. If different longest blockchains B_t^* exist, the blockchain versions are compared with the longest chain with which they share the most blocks (excluding a robot's own blockchain). In order to calculate the probability that robots act on consistent information (i.e., the probability that they do not experience severe inconsistencies), we measure the percentage of timestamps t over all robots r for which it is true that $I_t(r) \leq z^*$.

In addition, we measure the *block reception time* in seconds for different values of z^*: for each block in a robot's blockchain, this is the time elapsed until it receives z^* subsequent blocks (i.e., z^* block confirmations).

B. Experimental setup

Our experiments are conducted in the ARGoS robot swarm simulator [13], using Pi-puck robots (previous research has demonstrated that physical Pi-puck robots can successfully maintain blockchain networks [7], [12]). The arena is a square environment of 3.61 m^2, surrounded by walls (see Figure 3) and the robots perform a random walk—we use this setup as it has been used for blockchain-enabled swarm robotics experiments in previous research [7]. Each experiment runs for 1,800 seconds, and we perform 10 repetitions per experiment configuration.

Each robot in the robot swarm functions as a blockchain node and block producer in Toychain, a lightweight blockchain framework specifically designed for swarm robotics experiments [41], [42]. Toychain supports the Proof-of-Authority consensus protocol, simulating Ethereum's Clique implementation [14]. Its clock synchronization feature ensures temporal consistency between the simulation clock in ARGoS and blockchain operations.

This setup allows us to obtain simulation results faster than real-time. Moreover, because Toychain's PoA closely models Ethereum's PoA, these results can be transferred to real-world applications by leveraging the mature and well-established Ethereum framework [5].

V. INCONSISTENCIES ANALYSIS

In this work, we argue that swarm partitioning negatively impacts information propagation among the robots' blockchains. To support this claim, we analyze the likelihood of inconsistencies and the time required to generate and disseminate blockchain blocks.

Figure 4a shows the probability of acting on consistent data for different numbers of confirmations z^* and for various swarm sizes. This plot was obtained by computing the cumulative percentage of time during which robots experience certain inconsistency lengths in our experiments. Figure 4a could then be used as a decision-making tool for swarm designers. Specifically, if a scenario requires a certain percentage of robots to act on consistent information, a swarm designer could select the appropriate number of required confirmations z^*. For example, in a swarm of 16 robots, achieving approximately 80 % probability that robots act on consistent information would necessitate selecting $z^* = 5$ as the minimum required number of confirmations.

The swarm size correlates with the frequency of inconsistencies: larger swarms exhibit higher inconsistency rates because they have more sealers than smaller swarms and therefore they generate blocks more frequently. Consequently, in smaller swarms, a lower value of z^* can be chosen to achieve the same probability that robots act on consistent information.

Despite this relationship, the time required for a specific percentage of robots to confirm the data remains relatively similar across swarm sizes. For example, to get approximately 80 % probability to act on consistent information, one would need to select 3 confirmations in a 8-robot swarm and 6 confirmations in a 24-robot swarm. To receive 3 confirmations, it is expected to take 100 seconds in a 8-robot swarm, and to receive 6 confirmations, it is expected to take 120 seconds in a 24-robot swarm (see Figure 4b).

In summary, if a scenario requires a high percentage of robots to act on consistent information, a swarm designer must first be aware that such a situation requires waiting for a certain number of block confirmations. In addition, the designer might need to select a relatively high number of confirmations, resulting in undesirable delays in task execution. Therefore, in blockchain-enabled robot swarms,

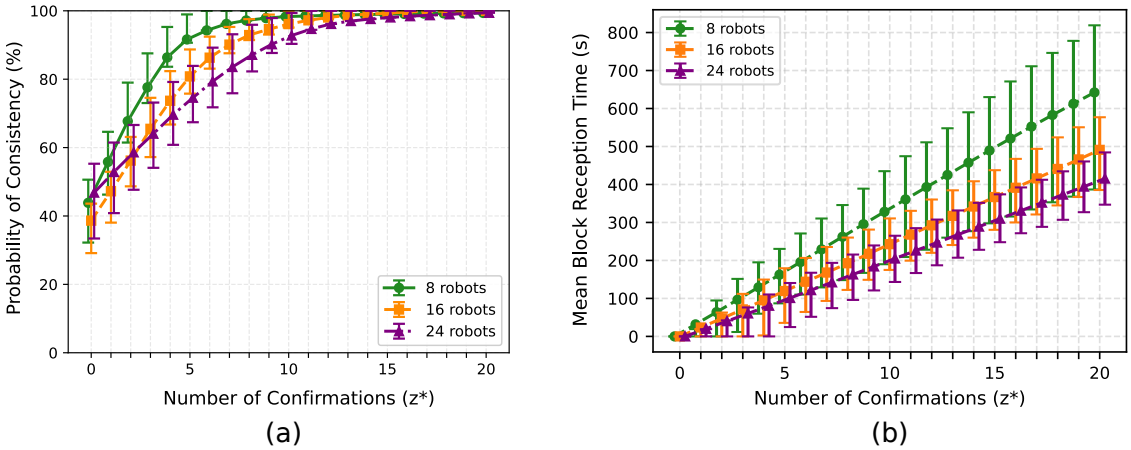

Fig. 4. Inconsistencies analysis (10 repetitions per swarm size, error bars show interquartile ranges). (a) In a larger swarm, there are more inconsistencies, therefore one needs to select a higher number of confirmations z^* to reach the desired probability of acting on consistent information. (b) With a larger swarm, the mean block reception time decreases. Therefore, even though such swarms are experiencing more inconsistencies, these inconsistencies are also expected to be resolved faster.

there exists a fundamental trade-off between maintaining information consistency and ensuring timely task execution.

VI. THE PROPOSED APPROACH

To reduce the development of inconsistencies and mitigate their drawbacks (see Section V), we introduce a framework that complements the standard operation of blockchain-enabled robot swarms. This framework consists of two modules: the *trigger rule* and the *panic behavior*.

For each robot, the trigger rule continuously monitors the status of its blockchain, detecting whether prolonged network partitioning has significantly delayed or halted block production or reception. Potential partitions are identified based on the fact that a decrease in network connectivity lowers the probability of finding a suitable sealer. This leads to a greater expected difference between the actual time it takes to create a block and the configured block period (see Section II). As a result, the time elapsed since the most recent block's timestamp can serve as an estimate of partition severity. To capture this, the trigger rule continuously monitors this elapsed time and compares it against a predefined threshold, set in this work to three times the configured block period: $3d = 3 \times 15\,\text{s} = 45\,\text{s}$. If the elapsed time exceeds this safety threshold, the robot initiates a change in behavior.

This change, called panic behavior (the second module), adjusts the robot's control parameters to improve its connection to the swarm, thus promoting the synchronization of its blockchain. Specifically, in this work, we implement a temporary increase in the robot's communication range from 0.1 m to 0.5 m, simulating the transition from a low-power, short-range communication technology to a high-power, long-range one. This strategy increases the likelihood of establishing communication with peers that possess more recent blockchain data (enabling the robot to synchronize its local blockchain) by leveraging the fact that the rate of robot encounters is proportional to the communication range [43], [44]. Once blockchain synchronization is performed, the trigger rule is no longer met and the robot reverts to its normal communication range of 0.1 m. This low communication range saves energy, as longer-range communication consumes significantly more power, which is a bottleneck for robots with limited battery capacity.

A. Results and Discussion

To validate the proposed framework, we conducted experiments with a single swarm size (16 robots) to maintain a concise comparison. These experiments confirmed that our framework significantly improves consistency among local blockchains. As shown in Figure 5a, for all blockchain confirmation values z^*, our approach achieves a higher probability of consistency compared to the original blockchain-enabled robot swarms. For instance, with $z^* = 4$, the likelihood of robots acting on consistent information increases from approximately 74 % in the original setting to approximately 84 % with our framework.

Throughout the experiments, robots spent only a small fraction of their time satisfying the trigger rule and executing the panic behavior (mean: 7.3 %, standard deviation: 6.5 %). This demonstrates the efficiency of our framework in preserving normal operations while effectively resolving inconsistencies when needed.

As illustrated in Figure 5b, incorporating the panic behavior also decreases the average reception time for new blocks. This indicates that our approach not only reduces the number of confirmations z^* required for a given probability of acting on consistent information but also shortens the time needed to obtain these confirmations. In the previous example with $z^* = 4$, the expected time to receive these four blocks decreases from 97 s to 81 s on average.

Our findings allow swarm designers to select appropriate block confirmation values z^* based on the consistency requirements of a given mission. For instance, in our 16-robot setup, if a task must be executed within a maximum delay of 200 seconds, one can expect to receive 10 confir-

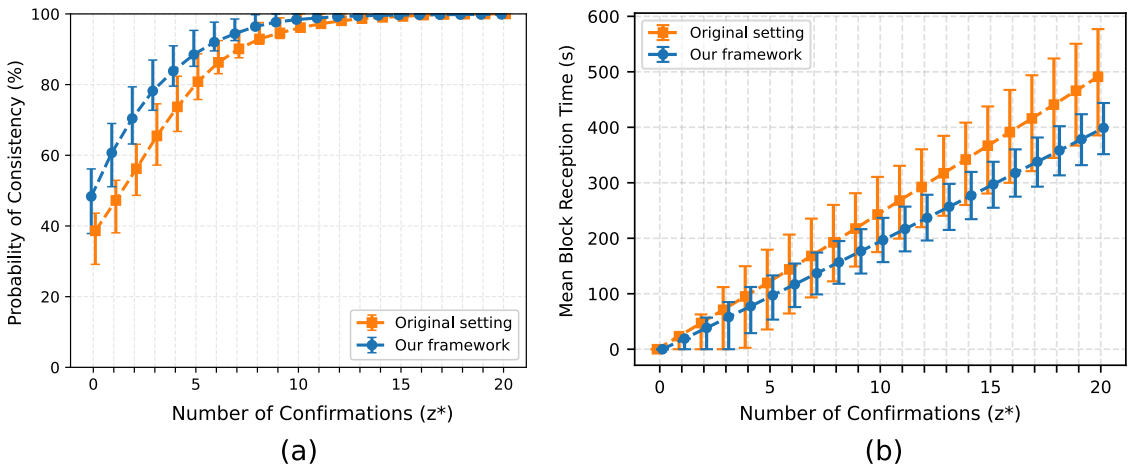

Fig. 5. Comparison of inconsistency occurrence in the original setting and with our framework (16 robots, 10 repetitions, error bars show interquartile ranges). (a) Consistency probability as a function of confirmation value z^*. This analysis enables the selection of the appropriate confirmation value z^* for a desired consistency probability (e.g., for approximately 60 % consistency probability, one needs to select $z^* = 1$ with our framework versus $z^* = 3$ with the original setting). (b) The mean block reception time decreases when using our framework, leading to a faster resolution of inconsistencies.

mations within this timeframe, ensuring a 97% probability of consistent information across the swarm. By enhancing the probability of robots acting on consistent information in blockchain-enabled swarms, our work provides a framework for predicting swarm behavior. In doing so, we address a critical challenge often overlooked in previous research: the fact that blockchain security properties hold only when the underlying inconsistencies of the blockchain protocol are properly accounted for.

VII. CONCLUSIONS

In this work, we questioned the security properties of blockchain-enabled robot swarms when required to act on information stored in the blockchain. Inconsistencies in local blockchains, caused by swarm partitioning, can significantly disrupt the swarm's mission. Using a simple scenario, we analyzed how partitioning affects the overall blockchain consistency, quantified the severity of these inconsistencies, and estimated the time required to resolve them. We demonstrated that employing blockchain frameworks without waiting for block confirmations can lead to decisions based on highly inconsistent data, undermining the very purpose of using blockchain in robot swarms. Furthermore, we introduced a novel framework consisting of trigger rules and panic behaviors, designed to reduce the occurrence of inconsistencies and accelerate their resolution. Experimental results confirmed the effectiveness of our approach in increasing the overall consistency and speeding up data dissemination, all while minimizing the disruption to the swarm behavior.

Our study focused on a challenging scenario with a very low communication range. Future work will explore different ranges and refine the trigger rule, which currently detects only out-of-sync blockchains. In real-world applications, increasing the communication range (e.g., switching from Bluetooth to Wi-Fi) as a panic behavior may lead to higher energy consumption. To minimize the battery drain, panic behaviors should be optimized based on robot capabilities and mission requirements. Alternative behaviors might include homing or flocking. In future work, we aim to develop a comprehensive framework by studying varied communication ranges, advanced trigger rules, alternative panic behaviors, and diverse movement patterns, ensuring robot swarms can truly benefit from the security offered by blockchain technology.

ACKNOWLEDGMENTS

G. Simionato and M.G.C.A. Cimino acknowledge support from the Italian Ministry of Education and Research (MIUR) in the framework of the FoReLab project (Departments of Excellence) and the Italian Ministry of Enterprises and Made in Italy, in the framework of the "Agreements for Innovation" Projet "4DDS - 4D Drone Swarms" Ref. no. F/310097/01-04/X56. V. Strobel and M. Dorigo acknowledge support from the Belgian F.R.S.-FNRS, of which they are a Postdoctoral Researcher and a Research Director respectively.

REFERENCES

[1] H. Hamann, *Swarm Robotics: A Formal Approach*. Springer, 2018.
[2] M. Dorigo, G. Theraulaz, and V. Trianni, "Swarm robotics: Past, present and future," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1152–1165, 2021.
[3] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Managing Byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*. IFAAMAS, 2018, pp. 541–549.
[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf, Accessed November 9, 2023.
[5] V. Buterin, "A next-generation smart contract and decentralized application platform. Ethereum project white paper." 2014, https://ethereum.org/en/whitepaper/, Accessed November 9, 2023.
[6] L. Van Calck, A. Pacheco, V. Strobel, M. Dorigo, and A. Reina, "A blockchain-based information market to incentivise cooperation in swarms of self-interested robots," *Scientific Reports*, vol. 13, no. 1, p. 20417, 2023.
[7] V. Strobel, A. Pacheco, and M. Dorigo, "Robot swarms neutralize harmful Byzantine robots using a blockchain-based token economy," *Science Robotics*, vol. 8, no. 79, p. eabm4636, 2023.

[8] M. Dorigo, A. Pacheco, A. Reina, and V. Strobel, "Blockchain technology for mobile multi-robot systems," *Nature Reviews Electrical Engineering*, vol. 1, no. 4, pp. 264–274, 2024.

[9] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292–321, 2016.

[10] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.

[11] H. Zhao, A. Pacheco, V. Strobel, A. Reina, X. Liu, G. Dudek, and M. Dorigo, "A generic framework for Byzantine-tolerant consensus achievement in robot swarms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2023)*. IEEE Press, 2023, pp. 8839–8846.

[12] A. Pacheco, V. Strobel, A. Reina, and M. Dorigo, "Real-time coordination of a foraging robot swarm using blockchain smart contracts," in *Swarm Intelligence – Proceedings of ANTS 2022 – Thirteenth International Conference*, ser. Lecture Notes in Computer Science, vol. 13491. Springer, 2022, pp. 196–208.

[13] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[14] P. Szilágyi, "EIP 225: Clique proof-of-authority consensus protocol," 2017, Accessed May 10, 2020. [Online]. Available: https://eips.ethereum.org/EIPS/eip-225

[15] J. Grey, O. Seneviratne, and I. Godage, "Blockchain-based mechanism for robotic cooperation through incentives: Prototype application in warehouse automation," in *Proceedings of the 2021 IEEE International Conference on Blockchain (Blockchain 2021)*. IEEE Press, 2021, pp. 597–604.

[16] M. G. Santos De Campos, C. P. Chanel, C. Chauffaut, and J. Lacan, "Towards a blockchain-based multi-UAV surveillance system," *Frontiers in Robotics and AI*, vol. 8, p. 557692, 2021.

[17] A. Kapitonov, S. Lonshakov, A. Krupenkin, and I. Berman, "Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of UAVs," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE Press, 2017, pp. 84–89.

[18] H. Gupta, V. Strobel, A. Pacheco, E. Ferrante, E. Natalizio, and M. Dorigo, "Group-level behavioral switch in a robot swarm using blockchain," in *Swarm Intelligence – Proceedings of ANTS 2024 – Fourteenth International Conference*, ser. Lecture Notes in Computer Science, vol. 14987. Springer, 2024, pp. 98–111.

[19] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Blockchain technology secures robot swarms: A comparison of consensus protocols and their resilience to Byzantine robots," *Frontiers in Robotics and AI*, vol. 7, p. 54, 2020.

[20] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002.

[21] H. Wang, H. Li, Q. Ye, P. Lu, Y. Yang, P. H. J. Chong, X. Chu, Q. Lv, and A. Smahi, "A physical topology for optimizing partition tolerance in consortium blockchains to reach CAP guarantee bound," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 9, p. e4820, 2023.

[22] G. R. Carrara, L. M. Burle, D. S. Medeiros, C. V. N. de Albuquerque, and D. M. Mattos, "Consistency, availability, and partition tolerance in blockchain: a survey on the consensus mechanism over peer-to-peer networking," *Annals of Telecommunications*, vol. 75, pp. 163–174, 2020.

[23] J. P. Queralta, F. Keramat, S. Salimi, L. Fu, X. Yu, and T. Westerlund, "Blockchain and emerging distributed ledger technologies for decentralized multi-robot systems," *Current Robotics Reports*, vol. 4, no. 3, pp. 43–54, 2023.

[24] J. A. Tran, G. S. Ramachandran, P. M. Shah, C. B. Danilov, R. A. Santiago, and B. Krishnamachari, "SwarmDAG: A partition tolerant distributed ledger protocol for swarm robotics," *Ledger*, vol. 4, 2019.

[25] L. E. Moser, Y. Amir, P. M. Melliar-Smith, and D. A. Agarwal, "Extended virtual synchrony," in *14th International Conference on Distributed Computing Systems*. IEEE, 1994, pp. 56–65.

[26] D. Cordova, A. Laube, G. Pujolle, *et al.*, "Blockgraph: A blockchain for mobile ad hoc networks," in *Proceedings of the 4th Cyber Security in Networking Conference (CSNet 2020)*. IEEE, 2020, pp. 1–8.

[27] S. Salimpour, F. Keramat, J. P. Queralta, and T. Westerlund, "Decentralized vision-based byzantine agent detection in multi-robot systems with IOTA smart contracts," in *International Symposium on Foundations and Practice of Security*. Springer, 2022, pp. 322–337.

[28] F. Keramat, J. P. Queralta, and T. Westerlund, "Partition-tolerant and byzantine-tolerant decision making for distributed robotic systems with IOTA and ROS2," *IEEE Internet of Things Journal*, vol. 10, no. 14, pp. 12 985–12 998, 2023.

[29] W. F. Silvano and R. Marcelino, "IOTA tangle: A cryptocurrency to communicate internet-of-things data," *Future generation computer systems*, vol. 112, pp. 307–319, 2020.

[30] J. Fang, F. Habibi, K. Bruhwiler, F. Alshammari, A. Singh, Y. Zhou, and F. Nawab, "Pelopartition: Improving blockchain resilience to network partitioning," in *IEEE International Conference on Blockchain (Blockchain 2022)*. IEEE, 2022, pp. 274–281.

[31] S. Peng, Y. Liu, J. Chen, J. He, and Y. Wang, "Komorebi: A DAG-based asynchronous BFT consensus via sharding," in *2023 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2023, pp. 1221–1227.

[32] K. Nicolas, Y. Wang, G. C. Giakos, B. Wei, and H. Shen, "Blockchain system defensive overview for double-spend and selfish mining attacks: A systematic approach," *IEEE Access*, vol. 9, pp. 3838–3857, 2020.

[33] C. Badertscher, Y. Lu, and V. Zikas, "A rational protocol treatment of 51% attacks," in *Proceedings of 41st Annual International Cryptology Conference (Advances in Cryptology – CRYPTO 2021)*. Springer, 2021, pp. 3–32.

[34] Nayancy, S. Dutta, and S. Chakraborty, "Lightweight blockchain approach to reduce double-spend and 51% attacks on proof-of-work," *Intelligent Data Analysis*, vol. 28, no. 5, pp. 1309–1319, 2024.

[35] J. Bae and H. Lim, "Random mining group selection to prevent 51% attacks on bitcoin," in *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W 2018)*. IEEE, 2018, pp. 81–82.

[36] S. M. Babur, S. U. R. Khan, J. Yang, Y.-L. Chen, C. S. Ku, and L. Y. Por, "Preventing 51% attack by using consecutive block limits in bitcoin," *IEEE Access*, pp. 77 852–77 869, 2024.

[37] X. Yang, Y. Chen, and X. Chen, "Effective scheme against 51% attack on proof-of-work blockchain with history weighted information," in *IEEE International Conference on Blockchain (Blockchain 2019)*. IEEE, 2019, pp. 261–265.

[38] H.-s. Moon, J. Song, H. Shin, and J. Jang, "Home IoT device management blockchain platform using smart contracts and a countermeasure against 51% attacks," in *Proceedings of the 2022 4th Asia Pacific Information Technology Conference*, 2022, pp. 191–195.

[39] S. Sayeed and H. Marco-Gisbert, "Assessing blockchain consensus and security mechanisms against the 51% attack," *Applied Sciences*, vol. 9, no. 9, p. 1788, 2019.

[40] R. Shrestha and S. Y. Nam, "Regional blockchain for vehicular networks to prevent 51% attacks," *IEEE Access*, vol. 7, pp. 95 033–95 045, 2019.

[41] A. Pacheco, U. Denis, R. Zakir, V. Strobel, A. Reina, and M. Dorigo, "Toychain: A simple blockchain for research in swarm robotics," 2024, arXiv preprint:2407.06630 [cs.RO].

[42] A. Moroncelli, A. Pacheco, V. Strobel, P.-Y. Lajoie, M. Dorigo, and A. Reina, "Byzantine fault detection in Swarm-SLAM using blockchain and geometric constraints," in *Swarm Intelligence – Proceedings of ANTS 2024 – Fourteenth International Conference*, ser. Lecture Notes in Computer Science, vol. 14987. Springer, 2024, pp. 42–56.

[43] N. Correll and A. Martinoli, "Modeling and designing self-organized aggregation in a swarm of miniature robots," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 615–626, 2011.

[44] D. T. Gillespie, "A rigorous derivation of the chemical master equation," *Physica A: Statistical Mechanics and its Applications*, vol. 188, no. 1-3, pp. 404–425, 1992.