ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2023.0426

IMPROVING BYZANTINE FAULT TOLERANCE IN SWARM ROBOTICS COLLECTIVE DECISION-MAKING SCENARIO VIA A NEW BLOCKCHAIN CONSENSUS ALGORITHM

Theviyanthan Krishnamohan
Department of Computer Science and Engineering, University of Westminster, United Kingdom

Abstract

Swarm robotics applies concepts of swarm intelligence to robotics. Discrete consensus achievement is one of the major behaviors found in swarm robotics. Various algorithms have been developed for discrete consensus achievement. However, existing discrete consensus achievement algorithms are vulnerable to Byzantine robots. Blockchain has been successfully used to mitigate the negative effect of Byzantine robots. Nevertheless, since the blockchain solution uses the Proof-of-Work blockchain consensus algorithm, it is vulnerable to the 51% attack. Besides, the swarm also takes longer to achieve consensus. This research proposes a novel blockchain consensus algorithm called Proof-of-Identity—which uses a private-public key pair and a swarm controller—to create a dynamically permissioned blockchain that would negate the 51%-attack problem associated with the Proof-of-Work algorithm while also reducing the consensus time. This proposed solution was tested against the classical solution and the existing blockchain solution using the collective perception scenario. Test results show that the Proof-of-Identity algorithm prevents the 51%-attack problem while improving the consensus time in comparison to the existing blockchain solution without affecting the exit probability.

Keywords:

Blockchain, Swarm Robotics, Proof Of Identity, Proof Of Work, Blockchain Consensus Algorithm, Collective Perception

1. INTRODUCTION

Swarm robotics uses multiple, simple robots to collectively solve real-life problems. Collective decision-making is one of the applications of swarm robotics. In collective decision-making, robots in a swarm try to collectively come to a consensus on one particular decision. Consensus achievement is a type of collective decision-making scenario where robots collectively choose one among several choices.

Several strategies exist to solve consensus achievement scenarios. However, such solutions are vulnerable to Byzantine robots. Blockchain-based solutions were developed to provide protection against Byzantine robots. However, blockchain introduced a new Byzantine problem in the form of the 51% attack. Further, these solutions also performed poorly in comparison to the existing solutions. Such issues with the blockchain-based solutions can be zeroed down to the Proof-of-Work (PoW) blockchain consensus algorithm used.

This paper proposes a novel blockchain consensus algorithm called Proof of Identity (PoI) to provide improved Byzantine fault tolerance to consensus achievement strategies in swarm robotics. Through performance and security testing, this study shows that the PoI algorithm offers immunity against the 51% attack while improving performance.

This paper first discusses swarm robotics before providing a primer on blockchain. Then, existing classical and blockchain-based solutions are explored. Subsequently, the methodology of the solution is discussed by explaining the PoI algorithm and the benchmarking tool that was developed. Finally, the experiment setup and test results are expounded before the findings are discussed and the conclusion is presented.

2. SWARM ROBOTICS

Swarm robotics applies concepts of swarm intelligence to robotics in order to solve problems that single, monolithic, or multi-agent robots cannot solve.

Swarm intelligence is heavily inspired by biological systems found in nature such as ant colonies, bee colonies, bird flocking, and bacterial growth. These systems solve complex problems via the coordination of simple individuals. A good example of this is insect societies that contain simple and homogenous individuals that find the best route to a source by communicating using pheromones without centralization or synchronization [1].

Swarm robotics can be formally defined as "the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment" [2].

2.1 CLASSIFICATION OF SWARM ROBOTICS

Brambilla et al. [3] classify the existing works into two major taxonomies, viz. methods and collective behaviors.

The methods taxonomy is based on the methods used to design swarm robotics systems. The collective-behaviors taxonomy is based on the basic problem-solving behaviors of swarms.

Collective behaviors are divided into four main groups: spatially organizing behaviors, navigation behaviors, collective decision-making behaviors, and other collective behaviors.

This research deals with collective decision-making behaviors. Collective decision-making is having a swarm agree on a certain decision. This can be divided into consensus achievement and task allocation. Consensus achievement is choosing one option among several others while task allocation is distributing different tasks among robots. This research focuses on consensus-achievement behavior.

3. BLOCKCHAIN

Blockchain was invented to decentralize monetary systems through a distributed ledger. However, over time, blockchain has

started to be used to create decentralized applications as well [4] [5].

A ledger is a chain of blocks that stores transactions. A private-public key pair is used to perform transactions. All nodes in a blockchain network get a copy of this ledger [6].

A transactor sends money to a recipient by using the recipient's public key. The transaction is signed using the transactor's private key. A transactor must have already received the money to be able to send it. This is verified by checking if there are transactions in the chain that are addressed to the transactor's public key.

To prevent double spending, the order of transactions should be recorded. So, transactions are packed into blocks and the blocks are chained together using hashes. This makes the order immutable. The blocks are generated through a process called mining. The nodes that generate blocks are called miners.

Miners compete to generate the next block. The winner is decided by a consensus algorithm. PoW is the most popular consensus algorithm at present. This algorithm decides the winner by checking if the hash value of a block is less than a specified value. The difficulty of mining a block can be adjusted by lowering or raising this value. Miners add a nonce value to their block to try to produce a block with a hash value below the specified value.

Producing the right hash value is done through trial and error. This work takes CPU time. The right hash value serves as proof of the miner's work. Thus, this algorithm is called Proof of Work. To modify the order of blocks, the work done since that block has to be repeated. This is expensive, thus, making the blockchain immutable.

4. RELATED WORK

4.1 CLASSICAL APPROACH

Valentini, Brambilla, et al. [7] introduced the collective perception scenario to test three different consensus-achievement strategies. In this scenario, the swarm tried to find the color of the majority of the tiles in a square grid that had black and white tiles. This scenario had two states, namely the exploration state and the dissemination state, and these were tantamount to the waggle dance of the bee populations [8].

Robots start with an opinion when the experiment is started. This opinion is about the color of the majority of the tiles. In the exploration state, the robots explore their environments through random walk and rotations for a random amount of time. If a robot detects an obstacle within 30cm, then it turns in the opposite direction and continues its motion. In the meantime, the robots scan the color of the floor using their ground sensors. The quality pi of an opinion *i*, where $i \in \{a, b\}$ (*a* corresponds to black and *b* to white), is defined as the amount of time the robot detected the color of its opinion (t_i) over the amount of time the robot spent in the exploration state (t).

$$p_i = t_i/t \quad (1)$$

After the exploration state, robots switch to the dissemination state. During this state, while performing random walk and rotations, robots also broadcast their opinion to their neighbors. Towards the end of this state, robots update their opinion with the best opinion. The consensus strategy used decides how the best opinion is chosen.

4.1.1 Direct Modulation of Majority-based Decision (DMMD):

When this strategy is used, a robot remains in the dissemination state for a random amount of time proportional to the quality of its opinion. This allows a robot with a higher quality opinion to broadcast its opinion to a lot of neighbors.

During the dissemination state, robots also receive the opinions of their neighboring robots. By the end of this state, the robots choose the opinion of the majority of their neighboring robots as their own and begin the next cycle. [9][10].

4.1.2 Direct Modulation of Voter-based Decision (DMVD):

The DMVD strategy differs from the DMMD only in its decision-making mechanism. Just like DMMD, DMVD also modulates its dissemination time using the quality of its opinion.

However, when DMVD is used, robots choose the opinion of a random neighbor as their own [11].

4.1.3 Direct Comparison (DC):

Unlike in DMMD and DMVD, the dissemination time is not modulated in DC. Instead, the dissemination time is randomly chosen. Besides, the robots broadcast the quality of their opinion in addition to their opinion. Towards the end, robots compare the quality of their opinion with that of a random neighbor and choose the greater of the two as their opinion [7].

Consensus is achieved when all the robots end up with the same opinion.

4.2 BLOCKCHAIN APPROACH

Strobel et al. [12] attempted to solve the Byzantine problem in the classical DMMD, DMVD, and DC strategies using blockchain. The authors found that the classical solutions faltered when faulty or malicious robots kept broadcasting the wrong opinion and they showed that blockchain could make these strategies immune to Byzantine robots.

In the blockchain approach, the exploration state was the same as it was in the classical approach. However, in the dissemination state, instead of broadcasting their opinion, robots voted using the smart contract. A vote was cast every 5 ticks (10 ticks made a second), so the higher the quality, the higher the number of votes was.

After voting, robots executed the decision-making strategy by calling the smart contract. When DMMD was used, the opinions of two pseudorandom robots were chosen and the opinion of the majority was chosen as the best opinion. When DMVD was used, the opinion of a pseudorandom robot was chosen as the best opinion.

When DC was used, robots passed both their opinion and its quality to the smart contract and picked the opinion of the higher quality between its own opinion and that of a pseudorandom robot.

Strobel et al. [13] employed exogenous fault detection to identify Byzantine robots. A vote from a robot was rejected if it was based on an outdated opinion or if the blockchain versions were different. An outdated opinion is an opinion that has not been updated during the last 25 blocks. Besides, robots could cast

a maximum of 50 votes when DMMD and DMVD were used and only one vote when DC was used.

Even though Strobel et al. (2018) solved the Byzantine problem using this approach, consensus time was found to be higher when compared to the classical approaches. This was because of the PoW consensus algorithm.

Additionally, since PoW is resource-intensive, it is not suitable to run on simple robotics devices. Moreover, PoW introduced a new Byzantine problem in the form of the 51% attack, which meant that the Byzantine problem was not completely resolved.

The PoW algorithm can be compromised by a node or a group of nodes with a hash rate in excess of 50% of the total hash rate of the network [14]. This attack is known as the 51% attack and the solution of Strobel et al. (2018) is vulnerable to it.

5. METHODOLOGY

5.1 PROOF OF IDENTITY (POI)

The PoI algorithm allows only authorized nodes to mine blocks and thus, creates a permissioned blockchain. However, in contrast to the typical Proof-of-Authority (PoA) algorithms, the authorized nodes are not declared before the blockchain is run [15]. To allow new miners into the network during runtime, the PoI algorithm introduces a novel swarm controller that uses a private-public key pair to sign authorized miners. This allows PoI to create dynamically permissioned blockchains.

When the swarm controller is spun up, a private-public key pair is generated. To add a new miner, the miner first sends its coinbase to the swarm controller. The swarm controller signs the coinbase with its private key and returns its signature. The miner also obtains the swarm controller's public key.

When mining a block, a miner adds its signature to the header of the block and seals it. When verifying blocks, the verifying node decrypts the signature of the block with the public key of the swarm controller and checks if the decrypted value is equal to the coinbase of the miner. If the values match, then the authenticity of the miner can be affirmed.

This solves the 51% attack threat because no node, however powerful it might be, cannot compromise the network if it is not authorized by the swarm controller. At the same time, since the algorithm does not involve producing the right block through trial and error, the performance concerns are also rectified.

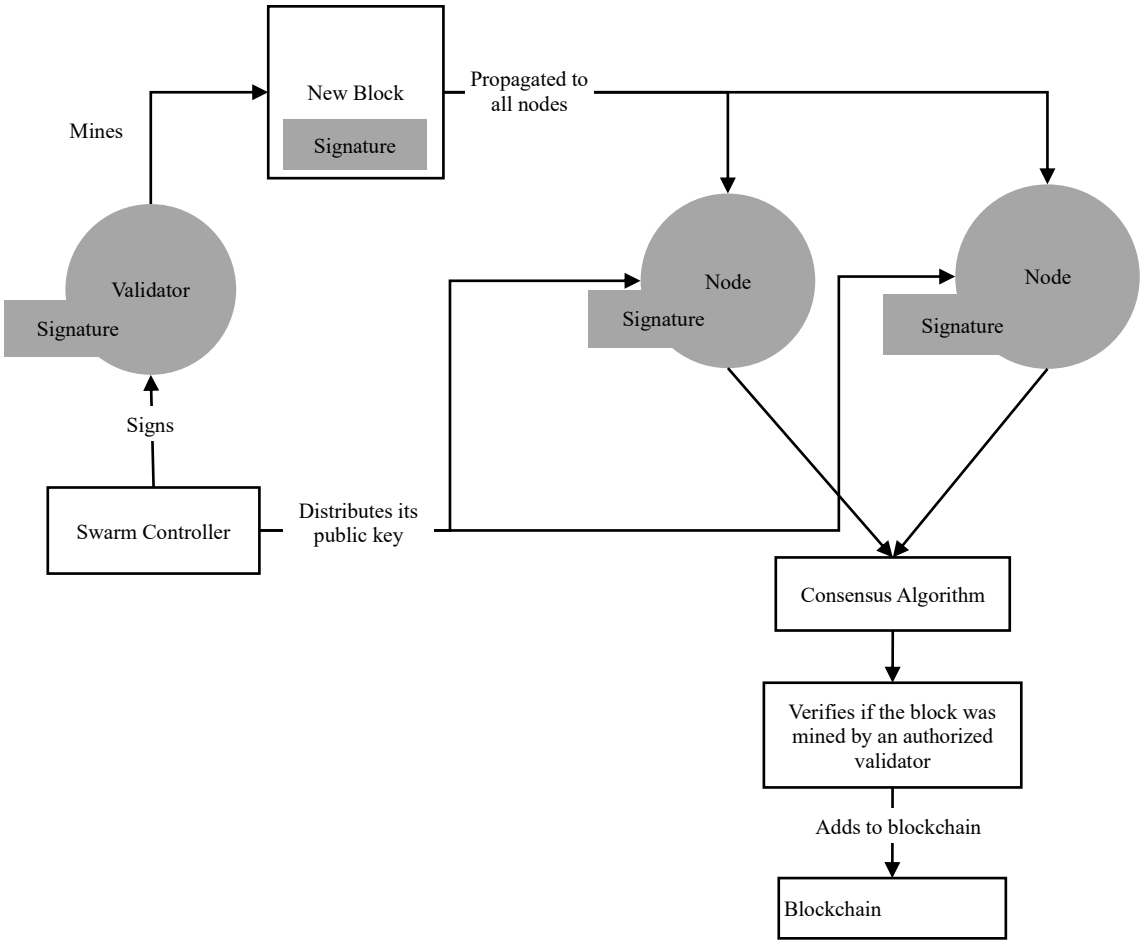

Fig.1. A diagrammatic representation of the PoI algorithm

5.2 BENCHMARKING TOOL

The benchmarking tool was developed to benchmark the performance of the PoI algorithm using the collective perception scenario on top of the benchmarking tool developed by Valentini, Brambilla, et al. (2016), and Strobel et al. (2018). This benchmarking tool improves the existing tool by introducing a live dashboard to carry out experiments, a database to store

experiment data, and a service layer to facilitate communication between the dashboard and the simulator.

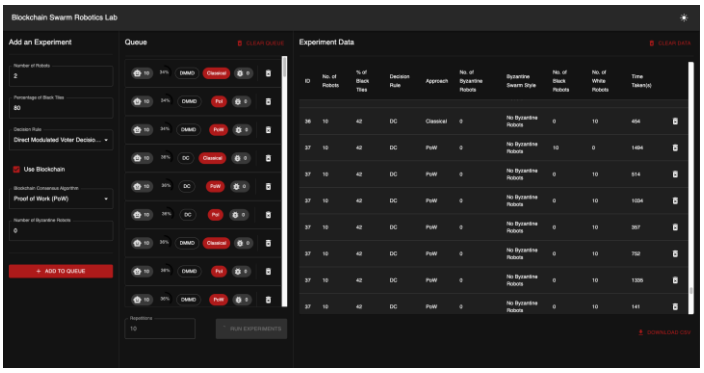

Fig.2. The user interface of the benchmarking tool

5.2.1 Architecture of the Benchmarking Tool:

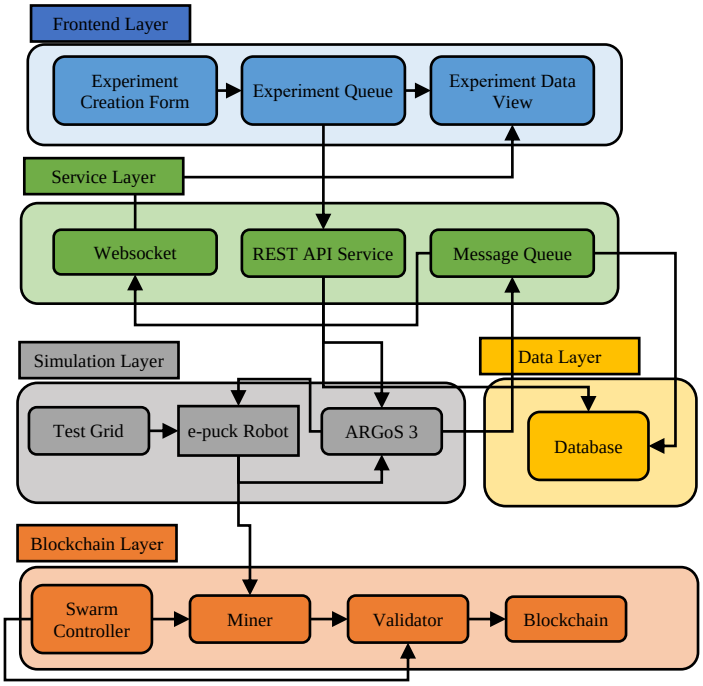

Fig.3. Layered architecture of the benchmarking tool

The architecture of the prototype consists of the frontend layer, service layer, simulator layer, and blockchain layer. The frontend layer provides the user of this prototype with a user interface to interact with the prototype. The service layer sits in between the frontend layer and the simulator layer and provides the necessary APIs to the frontend layer to communicate with the simulator layer. The simulator layer interacts with the blockchain layer to solve the collective perception scenario using the smart contract deployed in the blockchain. The forthcoming section discusses these layers and the modules belonging to them elaborately.

1. Front-end Layer

This layer consists of the Graphical User Interface (GUI) that a user will be using to interact with the prototype. It consists of the following modules:

- *Experiment Creation Form*: This is a form that allows a user to configure the parameters of the experiment such as the number of robots, the decision rule to be used, the percentage of black and white tiles, the number of Byzantine robots and the approach to be used.
- *Experiment Queue*: Since, to benchmark different solutions, a user may need to run experiments in batches, experiments created using the Experiment Creation Form are added to this queue. This queue allows users to delete experiments that are later deemed unnecessary, specify the number of times each experiment should be repeated, and provides a button to start running the experiments in the queue.
- *Experiment Data View*: This view shows the result of each experiment live as it is completed in a tabular format. This view also allows the user to download the results as a Comma-Separated-Values (CSV) file. Moreover, this view also shows a progress bar to give the user an idea about how many experiments have been completed and how many more remain.

2. The Service Layer

This layer allows the frontend layer to communicate with the simulator layer by providing the necessary APIs. The configurations of the experiment entered through the frontend layer are fed to the simulator via this layer. This layer also communicates the results of the experiment from the simulator layer to the frontend layer. The modules contained in this layer are as follows:

- *REST API Service*: This provides REST API services to be consumed by the frontend layer. Users can configure experiments, start experiments and get experiment results using these REST API services. The experiment configurations sent to this service by the frontend are also persisted in a database in the data layer.
- *Websocket*: This allows live experiment results to be streamed to the frontend layer so that users can view the experiment results in a GUI that gets updated automatically.
- *Message Queue*: This is used to capture the experiment results from the simulator layer. This allows process-to-process communication between the server and the simulator. The experiment results in the message queue are also persisted in a database in the data layer.

3. The Simulator Layer

This is the layer where the experiments are run. This layer gets the experiment configuration from the service layer, runs the experiments, and communicates the results of the experiments back to the service layer using the message queue. This layer consists of the following modules:

- *Test Grid*: This is the environment in which the robots will operate on. This is a 200 × 200cm2 grid consisting of 10 × 10cm2 tiles of colors black and white. The ratio between the number of black and white tiles is configurable. Moreover, this grid is bounded by walls that can be detected by the robots to avoid collisions.
- *e-puck Robot*: This is a small robot with a footprint of 7cm2 that is used to sense the color of the tiles and to take part in the consensus achievement task to find the color of the majority of the tiles. When blockchain is used, this robot also acts as a miner.

- *ARGoS* 3: This is the simulator that controls the robots. This simulator runs the robots on the test grid and finds out if a consensus has been reached or not. Apart from this, the simulator also gathers evaluation metrics such as the exit probability and consensus time and communicates them to the service layer.

4. The Blockchain Layer

The blockchain layer consists of the blockchain, the mining nodes, the validators, and the swarm controller. The e-puck robots in the simulator layer publish their opinion to the blockchain and receive updated opinions from the smart contract running on the blockchain. The functionality of the modules in this layer is discussed below.

- *Swarm Controller*: This module signs the coinbase of the miners with its own private key and distributes its public key to the miners. This allows the PoI algorithm to create a dynamically permissioned blockchain.
- *Miner*: The e-puck robots also act as miners who mine blocks to be added to the blockchain. When the robots publish their opinions as transactions, the miners verify these transactions and add them to a new block before sealing them with the signature.
- *Validator*: The e-puck robots also act as validators. The validators validate the blocks mined by the miners before adding them to their blockchain. The blocks are validated by verifying the signature found in the blocks using their coinbase and the public key of the swarm controller.
- *Blockchain*: The smart contract that runs the decision rule algorithm is deployed in the blockchain.

5. The Data Layer

The data layer consists of a database that is used to persist the experiment results so that this data can be later serialized into a different format or used as it is for data analysis. Aside from this, experiment configurations sent by the frontend to the REST API service are also persisted in the database.

5.2.2 The Functioning of the Benchmarking Tool:

The Fig.3 shows the data flow diagram that shows how data flows between different components of the benchmarking tool. Accordingly, it can be seen that a user first inputs the experiment configuration to the frontend app, which is then sent to the REST API service. This data is persisted in a database while being fed into the ARGoS 3 simulator. The simulator then configures the e-puck robots using this configuration data.

The e-puck robots sense the color of the tiles in the test grid and transact their opinion about the color to the blockchain miners. The miners verify these transactions, pack them into blocks and broadcast them to the validators. The validators validate these blocks and add them to their blockchain. The blockchain smart contract runs the decision rules and updates the e-puck robots with the new opinion. The ARGoS 3 simulator reads the opinions of the robots to decide if a consensus has been reached.

Once consensus is reached, the evaluation metrics of the experiment are pushed to the message queue. These metrics are persisted in the database and emitted to the frontend using a WebSocket so that the user can view the data live.

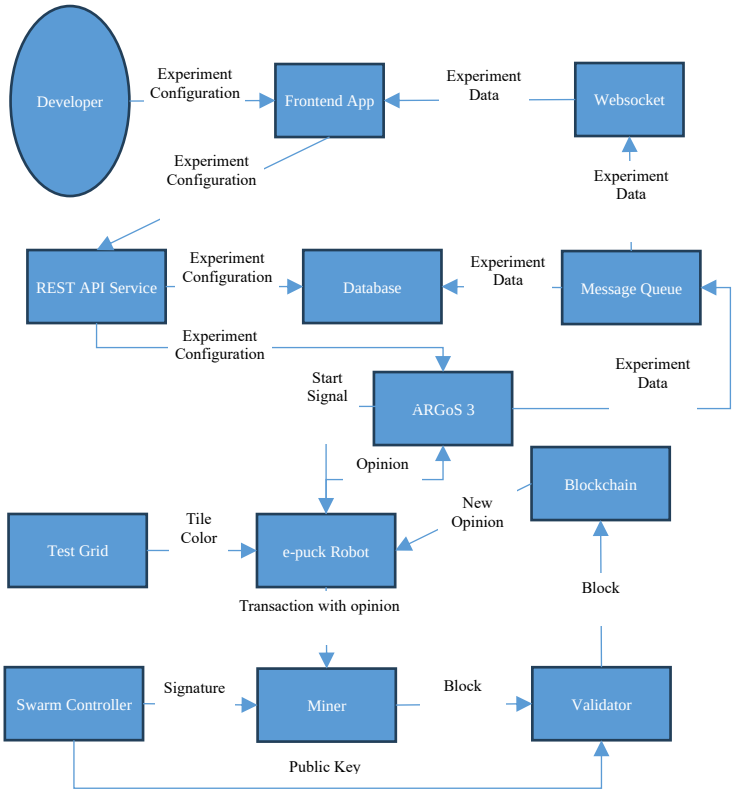

Fig.4. The data flow diagram of the benchmarking tool

6. TESTING

6.1 COLLECTIVE PERCEPTION EXPERIMENT

The collective perception scenario was used to benchmark the research prototype. The collective perception scenario involves a fixed number of robots coming to a consensus on the color of most tiles in a grid.

For the purposes of this research, 10 e-puck robots were deployed in a 200cm x 200cm grid bounded by four walls. The grid had 400 tiles each of area 10cm x 10cm. The tiles were either black or white in color and the ratio between the number of black and white tiles determined the difficulty of the challenge. The difficulty of the challenge is given by the following equation:

$$\rho_b = b/w \quad (2)$$

where:

ρ_b is the difficulty of choosing white as the best opinion

b is the percentage of black tiles

w is the percentage of white tiles

At the beginning of the experiment, one half of the robots started with the opinion black, and the other half started with the opinion white. During the experiments, robots changed their opinions based on the decision rules used and the experiment ended when all robots had the same opinion. In the experiments performed, white was always the color of most of the tiles. This was done to ensure the results of these experiments could be compared to those of the existing research works.

The experiments were executed in discreet time steps called ticks with 10 ticks forming a second. During the experiment, two

robots could communicate with one another only when the distance between them was under 50cm.

The experiments had the following configurable parameters, and experiments were run for every value of each of these parameters. The parameters and the values they took are given in Table.1.

Table.1. The experiment parameters and their values

Parameter	Values
Difficulty	0.52, 0.56, 0.61, 0.67, 0.72, 0.79, 0.85, 0.92
Decision Rules	DMMD, DMVD, DC
Approach	Classical, Proof of Work (PoW), PoI

The following metrics were used for benchmarking:

- *Exit probability*: The number of correct consensus decisions over the total number of runs.
- *Consensus time*: The time taken by a swarm to achieve the correct consensus.

Tests were run for the following approaches:

- Classical: The original approach used by Valentini et al. (2016).
- *PoW*: The blockchain-based approach used by Strobel, Ferrer and Dorigo (2018) using the PoW consensus algorithm.
- PoI: The blockchain-based approach used by Strobel, Ferrer and Dorigo (2018) using the PoI consensus algorithm.

Thus, altogether, 72 different types of experiments were planned. To avoid random errors and variations, each type of experiment was repeated 10 times. Consequently, 720 experiments were run in total.

6.2 EXPERIMENT SETUP

The experiments were run on a virtual machine running on a macOS host. The details of the virtual machine and the host machine are furnished in Table.2.

Table.2. Experiment setup

	Component	Model/Type/Capacity
Virtual Machine	CPU	ARM64
	Operating System	Debian 11.3.0
	RAM	14GB
	Hypervisor	QEMU 7.0 ARM Virtual Machine
Host Machines	CPU	Apple M1 Pro
	Operating System	macOS Monterey
	RAM	16GB (Unified memory)

6.3 TEST RESULTS

The Fig.5 shows the exit probability obtained for the three decision rules using the classical, PoW, and PoI approaches on a column graph.

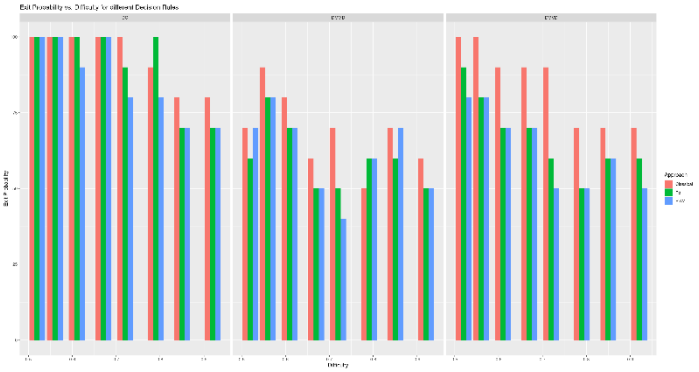

Fig.5. Exit probability for different decision rules and approaches

The Fig.5 shows the consensus time obtained for the three decision rules using the three different approaches on a box plot.

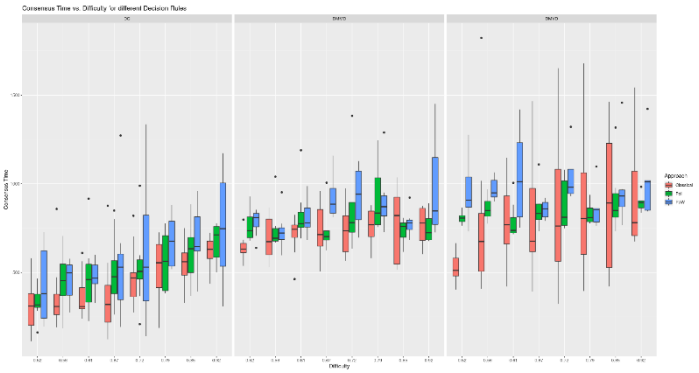

Fig.6. Consensus time for different decision rules and approaches

The results obtained for the classical as well as the PoW approaches were mostly consistent with the findings of Strobel, Ferrer and Dorigo (2018). The DC decision rule with the classical approach showed the highest exit probability under all but one difficulty settings. The DC decision rule along with the classical approach also produced the fastest consensus time.

The DMVD decision rule with the classical approach showed a steady decline in exit probability with the rise in difficulty whereas the DMMD rule, though showed an overall decline, had comparatively more variability. Overall, as far as the exit probability was concerned, the blockchain approach performed worse than the classical approach in comparison. Both the PoI and PoW approaches showed greater parity even though the PoI performed marginally better under certain circumstances.

The consensus time of both the classical approach and the blockchain approaches steadily increased with difficulty for the DC decision rule. However, even though the classical approach showed a similar steady increase for both the DMMD and DMVD decision rules, the consensus time of the blockchain approaches was largely disaffected by the difficulty. This observation is consistent with that of Strobel, Ferrer and Dorigo (2018).

However, unlike it was the case with the exit probability, the consensus time of the PoI approach showed a significant improvement in comparison to the PoW approach for all decision rules. Notwithstanding, the consensus time of the PoI approach was generally higher than that of the classical approach.

7. DISCUSSION

The findings of the tests were very similar to the findings of Strobel, Ferrer and Dorigo (2018). Generally, the classical approach had a better exit probability than both the PoI and PoW approaches. This is due to the limitation of the blockchain approach as explained by Strobel, Ferrer and Dorigo (2018). In the classical approach, duplicate opinions from the neighbors are discarded, while in the blockchain approach, no such implementation exists. The classical approach was also faster than the PoI and PoW approaches. This is due to the delays introduced by the mining process. However, the PoI approach was shown to be faster than the PoW approach. The test results showed that the PoI algorithm, developed to nullify the Byzantine robot issue introduced by the 51%-attack threat inherent to the PoW algorithm, made consensus achievement faster while not impacting the exit probability under most circumstances and slightly improving it under some.

8. CONCLUSION

This research work improved Byzantine fault tolerance in swarm robotics by addressing the 51%-attack issue found in the existing blockchain solution without compromising on the performance. Moreover, the developed solution was also shown to perform better than the existing blockchain solution improving the practical usability of blockchain-based solutions.

Besides, this research also created a web application to benchmark solutions to the collective perception scenario. This application will help future researchers benchmark their solutions in a lot more user-friendly manner.

REFERENCES

[1] G. Beni, "From Swarm Intelligence to Swarm Robotics", *Lecture Notes in Computer Science*, Vol. 3342, pp. 1-9, 2005.

[2] E. Şahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application", *Lecture Notes in Computer Science*, Vol. 3342, pp. 10-20, 2005.

[3] M. Brambilla, E. Ferrante, M. Birattari and M. Dorigo, "Swarm Robotics: A Review from the Swarm Engineering Perspective", *Swarm Intelligence*, Vol. 7, No. 1, pp. 1-41, 2013.

[4] M. Crosby, "BlockChain Technology: Beyond Bitcoin", Available at: http://scet.berkeley.edu/wp-content/uploads/AIR-2016-Blockchain.pdf, Accessed at 2016.

[5] T. Krishnamohan and K. Janarthanan, "BlockFlow: A Decentralized SDN Controller using Blockchain", *International Journal of Scientific and Research Publications*, Vol. 10, No. 3, pp. 1-14, 2020.

[6] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", Available at: www.bitcoin.org, Accessed at 2009.

[7] G. Valentini, D. Brambilla, H. Hamann and M. Dorigo, "Collective Perception of Environmental Features in a Robot Swarm", *Proceedings of International Conference on Swarm Intelligence*, pp. 65-76, 2016.

[8] K. Von Frisch, "*The Dance Language and Orientation of Bees*", Harvard University Press, 1993.

[9] G. Valentini, H. Hamann and M. Dorigo, "Efficient Decision-Making in a Self-Organizing Robot Swarm: On the Speed Versus Accuracy Trade-Off", *Proceedings of International Conference on Swarm Robotics*, pp. 1-8, 2021.

[10] G. Valentini, E. Ferrante, H. Hamann and M. Dorigo, "Collective Decision with 100 Kilobots: Speed versus Accuracy in Binary Discrimination Problems", *Autonomous Agents and Multi-Agent Systems*, Vol. 30, No. 3, pp. 553-580, 2016.

[11] G. Valentini, H. Hamann and M. Dorigo, "Self-Organized Collective Decision Making: The Weighted Voter Model", *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, pp. 45-52, 2014.

[12] V. Strobel, E.C. Ferrer and M. Dorigo, "Managing Byzantine Robots via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario", *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, pp. 1-13, 2021.

[13] A.L. Christensen, R. O'Grady and M. Dorigo, "From Fireflies to Fault-Tolerant Swarms of Robots", *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 4, pp. 754-766, 2009.

[14] N. Anita and M. Vijayalakshmi, "Blockchain Security Attack: A Brief Survey", Proceedings of International Conference on Computing, Communication and Networking Technologies, pp. 1-6, 2019.

[15] M.S. Ferdous, M.J.M. Chowdhury, M.A. Hoque and A. Colman, "Blockchain Consensus Algorithms: A Survey", Available: http://arxiv.org/abs/2001.07091, Accessed at 2021.