A Blockchain-Controlled Physical Robot Swarm Communicating via an Ad-Hoc Network

Alexandre Pacheco, Volker Strobel(✉), and Marco Dorigo

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
{alexandre.melo.pacheco,vstrobel,mdorigo}@ulb.ac.be

Abstract. We present a robot swarm composed of Pi-puck robots that maintain a blockchain network. The blockchain serves as security layer to neutralize Byzantine robots (faulty, malfunctioning, or malicious robots). In the context of this work, we implemented a framework for high-throughput communication using a decentralized mobile ad-hoc network. This work serves as a building block for secure real-world deployments of robot swarms. Our results show that the use of a blockchain is feasible and warranted in embodied robot swarm deployments.

1 Introduction

In real-world deployments, robot swarms will face a multitude of security challenges that are rarely taken into consideration in the swarm robotics field [7]. In particular, the presence of *Byzantine* robots, that is, malfunctioning, faulty, or malicious robots, might lead to a discrepancy between the *intended* and the *actual* behavior of a swarm. In a recent article, Strobel et al. [16] deliver a comprehensive proof-of-concept for a blockchain-based approach that greatly limits, in a fully decentralized manner, the impact of Byzantine robots on the robot swarm behavior.

The field of Blockchain Technology was initiated through the digital currency Bitcoin [12], in which the blockchain serves as a decentralized *ledger* for storing financial transactions. Later blockchain frameworks, such as Ethereum [2] (used in this work), extended the capability of blockchains to be decentralized *computing platforms*. Put simply, in the Ethereum blockchain the network participants can run programming code on the blockchain and agree on the outcome of the programs, without the need for supervision or mutual trust. It is precisely these decentralized programs that have proven to be useful in robot swarms, where a blockchain can serve as a secure decentralized coordinator and database [14–17]. To the best of our knowledge, all existing multi-robot systems that use blockchain and smart contracts technology have been demonstrated on simulated robots: the present paper is the first successful implementation of this technology in a physical robot swarm.

The replication of simulation results with physically embodied robots is crucial to convince the robotics community of the feasibility of blockchain and smart

M. Dorigo et al. (Eds.): ANTS 2020, LNCS 12421, pp. 3–15, 2020.
https://doi.org/10.1007/978-3-030-60376-2_1

contracts technology as a tool for solving security issues in robot swarms. However, moving from simulated to real robots is not straightforward and involves, among other things, the choice of adequate robot platforms, communication protocols, and blockchain frameworks. Unfortunately, in previous works addressing the topic, many questions whose answer would be of paramount importance for a successful real robot implementation were not addressed. Examples are:

- which lightweight robot platform can provide the requirements for running a blockchain framework?
- which blockchain consensus protocols are appropriate for robot swarms?
- which communication infrastructure should be used?

In this work we give a first answer to the above questions by presenting an experimental setup consisting of Pi-puck robots [10] which maintain a proof-of-authority blockchain network and communicate through a mobile ad-hoc network. The chosen robot platform—the Pi-puck—is a reasonable choice due to the low cost and easy availability of the Pi-pucks, and to their support for Linux that allows the easy installation of the blockchain software—Ethereum in our case. The choice of the proof-of-authority protocol is motivated by its low computational cost that allows for running it efficiently on the Pi-puck processor. Finally, we chose to implement a mobile ad-hoc network as communication infrastructure. This choice is consistent both with the standard swarm robotics requirements of *decentralized* and *local/peer-to-peer* communication (see e.g., [1,4,6]), and with the throughput required for blockchain synchronization.

The field of blockchain-based swarm robotics was set out in 2016 by [3]; the paper describes several use cases for blockchain-controlled robot swarms. Since 2018 there has been a number of simulation results for blockchain applications in robots such as: the achievement of consensus in robot swarms in the presence of Byzantine robots [15]; the improvement of communications and performance in industrial robots [5]; the formation of coalitions in cyber-physical systems [9]; the management of collaboration in heterogeneous multirobot systems [14]; the secure collection of data from robots [20]; and path planning in multi-robot systems [11]. Hence, research addressing the application of blockchain technology to robotics is an active research area and, as such, there is a high demand for platforms to run blockchain experiments on physical robots.

The remainder of this paper is structured as follows. Section 2 describes the robot hardware and control routines, the used blockchain software, and the setup of the experiments. Section 3 presents the experimental results. Finally, Sect. 4 concludes the paper and indicates directions for future research.

2 Methods

2.1 Experimental Scenario

In this paper, we consider a scenario where a robot swarm is given the task to determine the fraction of white tiles in a checkerboard environment (Fig. 1).

The difficulty of this task can be increased by increasing the number of Byzantine robots that distribute false information, or by decreasing the size of the swarm, which leads to lower coverage of the map and lower network connectivity. Our goal is to study how a blockchain—and its relevant components, such as smart contracts, cryptotokens and protocols for consensus—can be used to counteract the negative influence of Byzantine robots when a swarm is trying to achieve swarm wide consensus. As this is the first implementation of a physical robot swarm that uses blockchain and smart contracts technology, we provide guidelines for the establishment of such a system, and insights into some details of its operation.

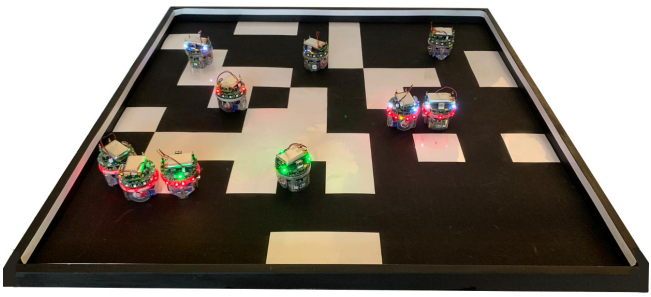

Fig. 1. The Pi-puck robots move in an $1 \times 1\,\mathrm{m}^2$ arena covered by 68 black and 32 white tiles. The robots' goal is to determine the fraction of white tiles by using their ground sensors. A part of the swarm act as Byzantine robots that disseminate wrong estimates. Using their LEDs, the robots communicate events—such as the receipt of a new block or the receipt of the consensus signal—to the experimenter for visual analysis.

The Robot Platform. The experiments are conducted using a swarm of up to $N = 10$ Pi-puck robots [10]. Pi-pucks are e-puck robots extended by a Raspberry Pi Zero W single board computer. Compared to the e-puck robot, the extension board improves the robots' communication capabilities and computational power. The Raspberry Pi Zero W has a 1 GHz processor with 512 MB of RAM. Hence, the use of the Raspberry Pi Zero W extension board allows for the implementation of more complex algorithms compared to previous e-puck robot versions.

The Arena. The arena has a dimension of $1 \times 1\,\mathrm{m}^2$. Its plywood floor is covered with 68 black and 32 white square tiles; therefore, the fraction of white tiles is 0.32. Each tile is $10 \times 10\,\mathrm{cm}^2$. The arena is bounded on each side by a wooden barrier which can be detected by the robots' obstacle avoidance sensors.

2.2 Control Routines

In the following, a high-level overview of the software that is executed on each robot is given. A more in-depth description is given in a separate technical report [13], in which we provide a detailed description of all the steps needed to set up and replicate our experiments using the Pi-puck robot.

The control for each robot is composed of five high-level routines that are executed in parallel at different frequencies:

- *Random-walk with obstacle avoidance* (frequency: 10 Hz): at each step the robot can perform: a) straight movement, b) rotation on site, or c) obstacle avoidance. If the robot's infrared (IR) sensors detect an obstacle, phase c) is selected in order to prevent collisions; otherwise, the robot alternates between the random-walk phases a) and b); the duration of each phase is sampled from an exponential distribution (we use the same parameters as [19]).
- *Estimation* (frequency: 1 Hz): the robots calculate a local estimate of the fraction of white tiles in the environment by dividing the number of white ground sensor readings by the total number of readings; to reduce noise, the sensors sample the floor at a rate 20 Hz and the average of these samples is used in this routine.
- *Peering* (frequency: 3 Hz): each robot uses its range-and-bearing IR actuators/sensors to simultaneously transmit its ID, and listen for other IDs within a fixed range (approx. 10 cm). After an ID is received, the robot executes a TCP request to obtain the Enode—a unique identifier of a blockchain node, used for the peering calls. After 2 s without receiving close range IR messages, a peer is removed from the blockchain and all information regarding that peer is deleted.
- *Local estimate dissemination* (frequency: $\frac{1}{45}$ Hz): Every 45 s, a robot sends its local estimate to the smart contract where local estimates are stored, aggregated, and refined to generate a *shared* estimate.
- *Block sealing* (frequency: varying—see below): On each robot, an instance of the Ethereum software *geth* is executed during the entire course of the experiment. In order to create new blocks on the blockchain, each robot acts as a sealer in this background process.

With the exception of the Ethereum client, which is implemented in the Go language [2], we implemented each control routine in Python[1]; in order to run the routines in parallel and guarantee the specified frequencies, the multi-threading package is used.

2.3 Blockchain Technology

For fundamentals of blockchain technology, we refer the reader to [16, Section 2] and to the original papers on Bitcoin [12] and Ethereum [2]. Here, we limit ourselves to the description of those aspects of blockchain technology that are most relevant for our setup—namely, the used *proof-of-authority* consensus protocol, as well as the concepts of *smart contract* and *cryptotoken*.

[1] Project repository: https://github.com/teksander/geth-pi-pucks.

Consensus Protocol. The decentralized nature of blockchains can result in conflicting situations (e.g., a different order of transactions in different versions of the blockchain). To resolve these conflicts and agree on a common order of transactions, a consensus protocol is needed. In this work, we use proof-of-authority (see [18] for the full specifications), an alternative to the original and most commonly used blockchain consensus protocol known as proof-of-work [12]. In contrast to the computationally expensive proof-of-work, proof-of-authority requires a majority of preselected nodes (i.e., in this work, a majority of robots) to agree on the state of the blockchain database. In the proof-of-authority protocol there are two kinds of blockchain nodes: normal blockchain nodes and *sealer* nodes, which are analogous to *miners* in proof-of-work and that are able to create new blocks by signing them. For each block, there is a preferred sealer, chosen in a round-robin fashion. If the preferred sealer signs the block, it is called an *in-turn* signature, if another sealer signs it, it is called an *out-of-turn* signature. The sealers can sign new blocks anytime they want, but in order for a new block to be valid:

- the timestamp of the new block must be at least $t = 15\,\mathrm{s}$ after the previous block (also known as the block time);
- a sealer can only sign one block in $\lfloor \frac{N}{2} \rfloor + 1$ blocks (to guarantee majority voting);
- a sealer must create a correct signature using its private key and sign the hash of the current block.

As soon as a sealer has signed a block, it disseminates the block in the network. The other nodes verify the signature and the validity of the block. The nodes agree on the *strongest* chain, that is the chain with the highest difficulty. The difficulty for in-turn signature is 2, and the one for out-of-turn signature is 1.

The major advantage of proof-of-authority is that it does not depend on solving computationally complex mathematical puzzles (as the standard proof-of-work-based consensus protocol does). Even though in this work our swarms contain a fixed number of robots (from 5 to 10), proof-of-authority also works with swarms with varying number of robots by either keeping a core of trusted sealers or adding and removing sealers based on majority vote.

Blockchain-Based Smart Contract. A blockchain-based smart contract is a piece of programming code that is stored on the blockchain. The smart contract encapsulates functions and variables, and participants of the blockchain network are able to alter its state by sending transactions to its functions. Blockchains additionally store the amount of "cryptotokens"—that is, immutable shares of a digital currency—that each participant possesses (see below).

The smart contract used in our research has four functions with which the robots can interact:

1. `sendEstimate(localEstimate)`: this function enables the robots to store their local estimates on the blockchain. In order to store an estimate, robots

have to send 40 ether. *Ether* is a scarce cryptotoken, and the fact that sending transactions requires ether effectively limits the number of transactions a robot can send;
2. `askForUBI()`: by sending a transaction to this function, robots can make a request for the universal basic income (see below for a description);
3. `getEstimate()`: this function returns the aggregated estimate of the fraction of white tiles, as determined by the blockchain-based smart contract;
4. `hasConverged()`: this function checks if the smart contract has reached convergence on an estimate (i.e., it determines if the absolute difference between the previous and the current value of the shared estimate is smaller than $\tau = 0.01$), in which case it returns 'true';
5. `registerRobot()` this function is called by each robot at least once, and is required before a robot is allowed to use any other function of the smart contract. This function allows using the same smart contract independently of the number of robots N.

The flow of information in the smart contract works as follows. After a robot registers itself, it can begin sending transactions that contain local estimates. Sent local estimates are stored in a list of proposals in the smart contract. As soon as N proposals are received, the smart contract performs a simple outlier detection, where all proposals with an absolute difference to the current blockchain estimate larger than $\delta = 0.2$ are discarded (except for the very first N proposals that are all accepted). The accepted proposals are used to update the estimate in the blockchain, which is the arithmetic mean of all accepted proposals. All robots that sent an accepted proposal get back their 40 ether plus a bonus consisting of a share of the non-repaid ethers of the discarded proposals.

Cryptotokens. In order to store their local estimate in the blockchain, robots send `sendEstimate` transactions accompanied by a fixed amount of *cryptotokens*. Cryptotokens are an immutable and scarce asset which is stored on a blockchain ledger. A digital asset with these properties is a key component to limiting the number of transactions robots can send, and thus prevent Sybil attacks. Robots can obtain tokens in two ways:

- by being reimbursed when sending accepted proposals (that is, by sending useful information);
- by receiving the universal basic income (UBI).

The UBI is an economy mechanism we established within the smart contract to allow the fair distribution of tokens between the robots. It functions as follows: at block numbers which are a power of 2 (i.e., in the blocks $2, 4, 8, \cdots$), the smart contract grants 20 ether to each robot in the swarm. This exponential scheme makes sure that in the beginning of an experiment every robot receives enough ether to be able to send its local estimate; however, over time sending useful information becomes the main means to receive additional ethers and to be able to continue participating in the experiment. Using this scheme, we can take advantage of the immutability and scarcity of blockchain cryptotokens to filter Byzantine robots out and limit their influence on the smart contract estimate.

2.4 Ad-Hoc Network

In order to exploit the Wi-Fi communication abilities of the Raspberry Pi Zero W without compromising the decentralization of the robot swarm, we establish a Mobile Ad-hoc Mesh Network using the B.A.T.M.A.N. routing protocol [8]. The advantage of such a network is that it does not rely on any central hubs (such as routers or master servers) nor does it assume global connectivity. Instead, each node participates in routing by forwarding data of other nodes.

In our experiments, the communication range of the ad-hoc network is additionally constrained by the range-and-bearing (RAB) board of the robots. We do this to enforce the swarm robotics core assumption of local communication, or, otherwise, the small arena size would lead to global communication. By tuning the power allocated to the RAB board it is possible to physically limit the communication range to approximately 5 cm. Robots broadcast the last 8 bits of their IP address in this fashion, and once an exchange has taken place, the connection to this IP address is established via the Ad-hoc network. Then, the robots exchange their enodes (an enode is a unique identifier for each Ethereum node) using TCP, in order to connect their Ethereum nodes and begin synchronization of the blockchain. From the moment a robot stops receiving signals from the RAB of a blockchain peer, a 2-s grace period is started after which the peer is removed.

2.5 Experiment Setup and Evaluation

Initialization and Termination. At the start of each experimental run, the robots are randomly distributed in the arena by the experimenter. Then, all robots connect to the Ethereum process of a bootstrap node (a desktop PC) and wait for a signal to start executing the parallel routines of Sect. 2.2. The experimenter sends the start signal from the bootstrap node, which consists of broadcasting a transaction containing the smart contract. An experimental run is stopped after all robots have received 'true' when querying `hasConverged()`, at which time they turn on their green LEDs.

Independent Variables. Each experiment may differ in (i) the total number of robots in the swarm; or (ii) the number of Byzantine robots.

(i) Swarm Size: Changing the swarm size allows for analyzing our platform in terms of two key features of robot swarms: *scalability*, i.e., the ability of the system to maintain or improve performance as the swarm size increases; and *partition-tolerance*, i.e., the ability of the system to reach consensus when there is reduced network connectivity (in this case, induced by a more sparse distribution of robots in the arena). As mentioned before, the dynamic addition and removal of block sealers is a feature in proof-of-authority; however, it has not been exploited in this work, and instead, all sealers are included on the genesis block in each experiment.

(ii) Byzantine Robots: To study the performance of our approach for increasing numbers of Byzantine robots, we model a Byzantine robot as a robot that disables its ground sensor, keeps a local estimate $\hat{\rho} = 0.0$, and sends this faulty estimate to the smart contract. This failure mode is well-motivated by our tests with physical robots and can occur in several situations: (1) a robot gets stuck on a tile during the course of the experiments, for example, due to a broken motor; (2) a robot's ground sensor does not have the correct distance from the floor, for example, due to a loose screw; (3) the communication to the ground sensor is broken, for example, due to a crash of the I^2C communication protocol; or (4) the robot is controlled by a malicious entity that tries to work against the goal of the swarm. Byzantine robots are selected randomly by the experimenter at the start of the experiment.

Table 1. Overview of the experiments and their parameters

No.	Experiment name	Swarm size	# Byzantine robots
1	Increasing Byzantines	10	$0, 1, 2, 3, 4$
2	Increasing swarm size (no Byzantine robots)	$5, 6, 7, 8, 9, 10$	0
3	Increasing swarm size (20% Byzantine robots)	$5, 10$	20%

Metrics. The performance of our approach is evaluated by comparing the error between the actual fraction ρ of black tiles to the blockchain estimate of a randomly selected robot at the end of each run; and the time required for all robots to receive the consensus signal. In addition, we record the size of the blockchain in MB and we use it to draw conclusions regarding the scalability of the approach.

3 Results

In order to evaluate the presented approach with physical robots, we conduct three experiments (Table 1). For each setting of each experiment we conduct 10 repetitions.

3.1 Experiment 1: Increasing Byzantines

The first experiment studies the impact of Byzantine robots in a swarm of fixed size $N = 10$. The number of Byzantines is increased from 0 to 5. Our hypothesis is that the approach has the lowest absolute error when no Byzantines are part of the swarm. We expect the Byzantines to have little effect up to a crucial point where they have a strong adverse effect on the estimate.

Results and Short Discussion. Figure 2 shows the results obtained. The Byzantine robots have a small impact when their number is between 0 and 3 (median of absolute error <5%) because their estimates are rejected by the smart contract and therefore they eventually run out of cryptotokens. As soon as four Byzantines are part of the swarm, the median error becomes significantly larger as Byzantines begin to collect rewards and therefore to have a stronger influence on the estimate. We also observe a high variability that is due to the fact that Byzantine robots may or may not become the dominant party—the estimate may therefore sway in either direction: reality, or zero. The estimate variability decreases with five Byzantines because in this case the Byzantines become dominant as they always send the same 0% estimate, and are therefore able to consistently collect rewards and steer the estimate towards the wrong value.

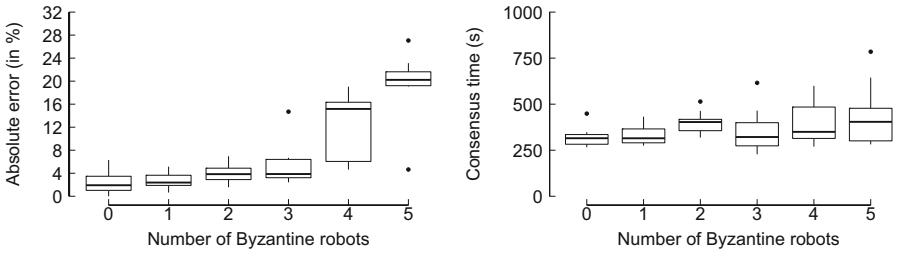

Fig. 2. Experiment 1 – Increasing Byzantines (10 robots in total, 0 to 5 Byzantines). *Left*: The median of the absolute error stays below 5% for 0 to 3 Byzantine robots. With 4 and 5 Byzantine robots the estimate error becomes much larger as the Byzantine robots are able to steer the estimate towards the wrong value. *Right*: There are no statistically significant differences in the consensus time when the number of Byzantine robots increases, even though the variability tends to increase.

3.2 Experiment 2: Increasing Swarm Size (No Byzantine Robots)

In Experiment 2 we investigate to what extent the size of the swarm has an influence on the consensus time as well as on the blockchain size. To this end, we increase the swarm size from 5 to 10 robots.

Results and Short Discussion. The median of the absolute error is below 5% for all swarm sizes and independent of the swarm size (results not shown). As expected, the consensus time decreases with an increasing swarm size (Fig. 3, *left*). The consensus time is influenced by several variables, such as the number of transactions and the average connectivity of the network, which is higher when there are more robots distributed in the arena. The blockchain size grows linearly in time. To obtain the growth rate for the different swarm sizes, a linear regression was performed using time as a predictor of blockchain size (Fig. 3, *right*). The larger the swarm size, the more transactions are created, thus the faster the blockchain size grows.

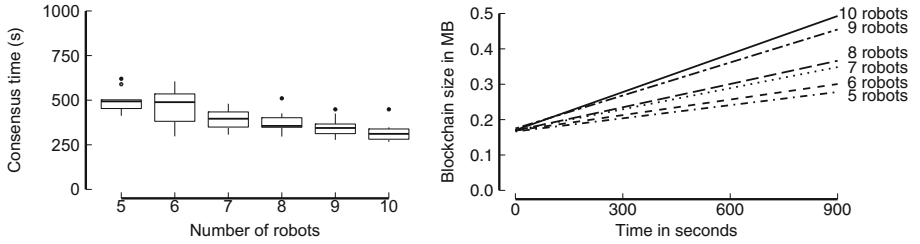

Fig. 3. Experiment 2 – Increasing swarm size (5 to 10 robots, no Byzantine Robots). The consensus time (*left*) decreases approximately linearly with the number of robots. The blockchain size (*right*) increases linearly over time (values obtained by linear regression for each swarm size).

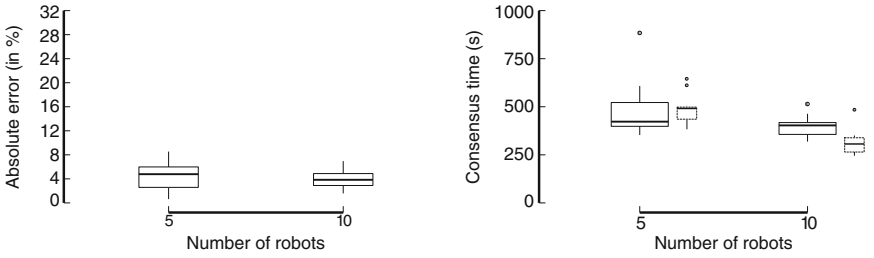

Fig. 4. Experiment 3 – Increasing swarm size (5 or 10 robots, 20% Byzantine robots). *Left*: The absolute error is not influenced by the number of robots. *Right*: With 5 robots the consensus time with (large boxplots) or without (small boxplots) Byzantine robots is not significantly different; with 10 robots the presence of Byzantines significantly increase the consensus time.

3.3 Experiment 3: Increasing Swarm Size (20% Byzantines)

In the third and final experiment, we study how different swarm sizes deal with a fixed fraction of 20% Byzantine robots. We perform Experiment 3 exclusively with 5 and 10 robots, because only these two swarm sizes permit to have exactly 20% Byzantine robots.

Results and Short Discussion. Figure 4 (left) shows that, as without Byzantines, the absolute error is independent of the swarm size. Figure 4 (right) shows that, when comparing the results to Experiment 2, with 5 robots the consensus time with or without Byzantines is not significantly different, while with 10 robots the presence of Byzantines significantly increases the consensus time.

4 Conclusions

In our research we are interested in developing robot swarms that present a high level of security against the possible presence of Byzantine robots.

The blockchain protocol uses a set of technologies to generate secure and tamper-proof knowledge shared by a network of mutually untrusting agents (robots in our case): public-key cryptography, digital signatures, consensus protocols, decentralized databases, and smart contracts. By using these technologies within the blockchain protocol it becomes possible to protect a robot swarm from Sybil attacks [16] and to reduce the influence that Byzantine robots can have on the overall swarm behavior. Additionally, it makes it possible to let the swarm reach consensus about the overall status of the system as stored in a decentralized ledger that can also be used as a tamper-proof register of events, accessible during, or after, an experiment.

While we had already demonstrated the feasibility of using the blockchain protocol in a robot swarm in previous work [15–17], this was done only in simulation. One important question when moving to real robots is whether the computations and communications required by the blockchain protocol are still feasible in a system composed of agents (the robots) that have limited computational power and only local communication (i.e., they can only communicate with neighbour robots)—as opposed to standard implementations of the blockchain protocol where the individual nodes are powerful computers that are fully connected to each other.

In this paper we have demonstrated the first example of a physical robot swarm that uses the blockchain protocol and smart contracts. In particular, we have showed that it is possible to do so in a swarm of not-so-powerful robots using a low-cost Raspberry Pi Zero W as onboard computer. To get these results, we have used the proof-of-authority protocol and our results show that the Ethereum client *geth* running on our robots uses, regardless of experimental parameters, about 13.7% of the available Raspberry Pi's CPU power.

Our results show that the blockchain approach is feasible also in terms of data storage: the size of the blockchain data folder in our robots grows linearly over time, and remains under 0.5 MB at the end of a 15-min run (Fig. 3). Therefore, the same experiment could be executed for about one year using a 16 GB SD card as data storage.

In conclusion, the reader should note that our goal for this article was to show that our previous results obtained in simulation would carry over to a swarm of real robots and consequently that the use of a blockchain is warranted for real-world deployment of secure robot swarms. We did not, however, intend to provide the most efficient possible implementation, nor fine tune parameters in order to achieve the best possible results: these aspects are left for future work.

Acknowledgements. Alexandre Pacheco acknowledges support via a fellowship from the Faculty of Applied Sciences of the Université Libre de Bruxelles. Volker Strobel and Marco Dorigo acknowledge support from the Belgian F.R.S.-FNRS, of which they are a Research Fellow and a Research Director respectively.

References

1. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intell. **7**(1), 1–41 (2013). https://doi.org/10.1007/s11721-012-0075-2
2. Buterin, V.: A next-generation smart contract and decentralized application platform. Ethereum project white paper. Technical report, Ethereum Foundation (2014). https://github.com/ethereum/wiki/wiki/White-Paper. Accessed 18 July 2019
3. Castelló Ferrer, E.: The blockchain: a new framework for robotic swarm systems. e-print (2016). arXiv:1608.00695v3
4. Dorigo, M., Birattari, M., Brambilla, M.: Swarm robotics. Scholarpedia **9**(1), 1463 (2014)
5. Fernandes, M., Alexandre, L.A.: Robotchain: using Tezos technology for robot event management. Ledger **4**(Suppl. 1) (2019). https://doi.org/10.5195/ledger.2019.175
6. Garattoni, L., Birattari, M.: Swarm robotics. In: Webster, J.G. (ed.) Wiley Encyclopedia of Electrical and Electronics Engineering. Wiley, Hoboken (2016)
7. Higgins, F., Tomlinson, A., Martin, K.M.: Survey on security challenges for swarm robotics. In: Proceedings of the Fifth International Conference on Autonomic and Autonomous Systems, pp. 307–312. IEEE Press (2009). https://doi.org/10.1109/ICAS.2009.62
8. Johnson, D., Ntlatlapa, N., Aichele, C.: A simple pragmatic approach to mesh routing using BATMAN. In: Proceedings of the 2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries (WCTID 2008) (2008)
9. Kashevnik, A., Teslya, N.: Blockchain-oriented coalition formation by CPS resources: ontological approach and case study. Electronics **7**, 66 (2018). https://doi.org/10.3390/electronics7050066
10. Millard, A.G., et al.: The Pi-puck extension board: a Raspberry Pi interface for the e-puck robot platform. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 741–748. IEEE Press (2017)
11. Mokhtar, A., Murphy, N., Bruton, J.: Blockchain-based multi-robot path planning. In: 2019 IEEE 5th World Forum on Internet of Things, pp. 584–589 (2019). https://doi.org/10.1109/WF-IoT.2019.8767340
12. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Technical report (2008). https://bitcoin.org/bitcoin.pdf. Accessed 11 Aug 2018
13. Pacheco, A., Strobel, V., Dorigo, M.: A framework for swarm robotics experimentation with Pi-puck robots and an Ethereum-based blockchain. Technical report TR/IRIDIA/2020-001, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2020)
14. Queralta, J.P., Westerlund, T.: Blockchain-powered collaboration in heterogeneous swarms of robots. e-print (2019). arXiv:1912.01711v2
15. Strobel, V., Castelló Ferrer, E., Dorigo, M.: Managing Byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. In: Dastani, M., Sukthankar, G., André, E., Koenig, S. (eds.) Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018), Richland, SC, USA , pp. 541–549. International Foundation for Autonomous Agents and Multiagent Systems (2018)

16. Strobel, V., Castelló Ferrer, E., Dorigo, M.: Blockchain technology secures robot swarms: a comparison of consensus protocols and their resilience to Byzantine robots. Front. Robot. AI **7**, 54 (2020). https://doi.org/10.3389/frobt.2020.00054
17. Strobel, V., Dorigo, M.: Blockchain technology for robot swarms: a shared knowledge and reputation management system for collective estimation. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Reina, A., Trianni, V. (eds.) Swarm Intelligence – Proceedings of ANTS 2018 – Eleventh International Conference. LNCS, vol. 11172, pp. 425–426. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00533-7
18. Szilágyi, P.: EIP 225: Clique proof-of-authority consensus protocol (2017). https://github.com/ethereum/EIPs/issues/225. Accessed 10 May 2020
19. Valentini, G., Brambilla, D., Hamann, H., Dorigo, M.: Collective perception of environmental features in a robot swarm. In: Dorigo, M., Birattari, M., Li, X., López-Ibáñez, M., Ohkura, K., Pinciroli, C., Stützle, T. (eds.) ANTS 2016. LNCS, vol. 9882, pp. 65–76. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44427-7_6
20. White, R., Caiazza, G., Cortesi, A., Cho, Y., Christensen, H.: Black block recorder: immutable black box logging for robots via blockchain. IEEE J. Robot. Autom. **4**, 3812–3819 (2019). https://doi.org/10.1109/LRA.2019.2928780