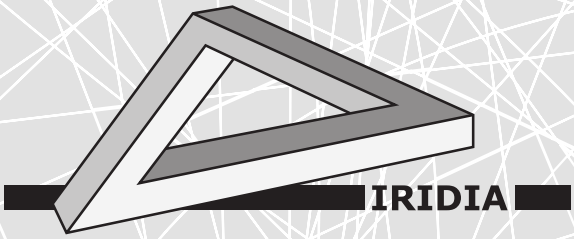

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation

V. Strobel and M. Dorigo

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2018-009

May 2018

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2018-009

Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation

Volker Strobel[1] and Marco Dorigo[1]

[1]IRIDIA, Université libre de Bruxelles, Belgium
{vstrobel, mdorigo}@ulb.ac.be

Abstract

In this work, we demonstrate how a blockchain can be used as shared knowledge medium, computing platform, and reputation management system in robot swarms. We show that a swarm of robots can collectively determine the relative frequency of environmental features via blockchain-based smart contracts (decentralized protocols executed via blockchain technology). Consensus in the swarm is achieved in a fully decentralized way without the need of an external observer. Via the presented reputation management system residing on the blockchain, the impact of malfunctioning robots (Byzantine robots) can be managed. We conducted three experiments using a robot swarm simulator, showing (i) the feasibility of the approach, (ii) the trade-off between blockchain size and accuracy, and (iii) the suitability of the blockchain as a reputation management system.

Keywords

Byzantine robot fault-tolerance, blockchain technology, reputation management system, robot swarms

1 Introduction

In swarm robotics research, it is often assumed that robots do not have access to shared knowledge. This is mainly due to three reasons: (i) it could be unfeasible to set up the infrastructure for such a shared knowledge system; e.g., if the robots are in a remote area and scattered throughout a large physical space, (ii) the shared knowledge system could represent an unacceptable single point of failure, and (iii) it might be computationally too complex to process all incoming and outgoing data in a single system. By sharing knowledge, however, it may become easier to determine whether consensus has been reached or to aggregate the information of the individual robots (the shared knowledge system could, for example, calculate the mean value of the sensor readings of the single robots). Hence, decisions could be based on a shared view of the world, possibly simplifying several swarm robotics tasks.

In this paper, we show that blockchain technology can provide the infrastructure for setting up a shared knowledge system in a robot swarm (see Figure 1 for a high-level overview of the approach). The blockchain is a peer-to-peer database and computing system, originally devised for the digital currency Bitcoin [13].

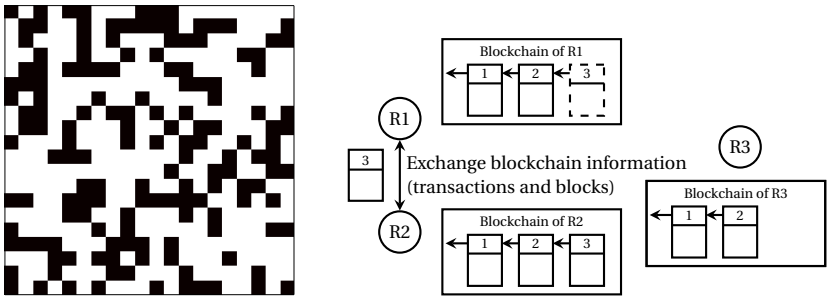

Figure 1: *Left*: Example of a generated floor (2×2 m^2) covered with 400 black and white tiles. The robots collectively estimate the relative frequency of the black tiles. *Right*: In this simplified example, the robots R1, R2, and R3 sense the features of the floor. Each robot Ri locally stores a copy of the blockchain. R1 and R2 are within communication range, whereas R3 cannot communicate with another robot. The robots R1 and R2 exchange their blockchain information: R2's blockchain is longer than R1's (R2 has solved block 3 by performing a computational mining puzzle). Since robots agree on the longest chain, robot R2 sends its latest block to R1. The blockchain contains transactions (sensor measurements in this work).

We argue that via blockchain technology, one can address drawbacks of having a shared knowledge system. For example, even if the robots are scattered throughout space, information in the blockchain can get aggregated once the robots are in communication range again. The blockchain does not pose a single point of failure due to its consensus protocol and its distributed character (replications of the data among the peers of the network). To reduce the computational complexity, it is important to decide which information should be processed via the blockchain and which information can be processed locally by the individual robots.

In previous work [19], we provided a proof-of-concept for addressing security issues in robot swarms via blockchain technology using the blockchain implementation *Ethereum* [2, 23]. The goal of the previous research was to show that blockchain technology can provide a security layer on top of existing algorithms and we implemented variations of existing approaches using blockchain technology. We showed that an existing collective-decision making approach fails as soon as one robot does not behave according to the protocol. In contrast, the blockchain-based approach presented there is able to identify these Byzantine robots and exclude them from the swarm.

However, the implementation of these variations reduced the design space to algorithms originally designed for approaches without blockchain technology. In contrast, in this work, we present a new algorithm tailored to blockchain technology, exploiting the blockchain's possibilities. To this end, we use the same experimental setup as in our previous research: an environment in which the floor is covered with white and black tiles, modeled with the robot swarm simulator ARGoS [14]. However, instead of solely determining if there are more black or white tiles (i.e., a binary decision task), in the present work, the swarm's goal is to determine the relative frequency of black tiles (expressed as a value between 0.0 and 1.0)—a collective *estimation* scenario which yields significantly more information and may be more desired in real-world deployments. As soon as the uncertainty in the estimated relative frequency is below a threshold, the experiment is stopped in a fully decentralized way using the blockchain. Additionally, we show how to identify Byzantine robots (robots performing

"arbitrarily faulty or malicious behavior" [19]) via a simple reputation management system.

Besides its function as shared knowledge medium, the blockchain serves as audit log and records all crucial actions—sensor readings in this work—in a tamper-proof way. The data can, hence, be easily analyzed during or after the completion of an experiment. The log may be important for digital forensics, monetarization, and trust in the outcome of an experiment. Attacks, such as replay attacks or Sybil attacks are inherently prevented by using blockchain technology.

We demonstrate the blockchain's capabilities in three experiments (see Section 5). Experiment 1 provides a proof-of-concept of the feasibility of the shared knowledge approach; Experiment 2 shows the trade-off between blockchain size and accuracy; and Experiment 3 demonstrates how Byzantine robots can be managed via a reputation management system.

The remainder of this paper is structured as follows. Section 2 summarizes related work. Section 3 discusses which requirements must be met to use blockchain technology with robot swarms. Section 4 describes the novel blockchain-tailored algorithm and the setup of our experiments. Section 5 presents the results of our experiments. Section 6 discusses the results and provides our conclusions.

2 Related Work

Many swarm robotics tasks require the swarm to make a collective decision [21]. Collective decision-making tasks are divided into consensus achievement problems and task allocation problems [1]. Consensus achievement problems are then further divided into *discrete* and *continuous* problems [21]. Discrete problems can be formalized as best-of-n problems, where the swarm has to agree upon a choice among a finite set of n choices. In continuous problems, in contrast, the swarm agrees upon a choice among an infinite set of continuous choices. Examples of discrete problems are path selection [12], site selection [15], and collective perception [20]. Examples of continuous problems are collective motion [6], spatial aggregation [18], and collective estimation (as studied in this work). However, all of these problems were studied in secure laboratory settings and do not address the performance of the approaches in the context of Byzantine robots.

At the outset of swarm robotics research, robot swarms were assumed to be fault-tolerant by design, due to the large number and redundancy of the robots units [10]. The first survey on security issues in robot swarms was presented in [9]. The authors identify *tampered swarm members or failing sensors, attacked or noisy communication channels*, and *loss of availability* as the main threats to robot swarms. The detection of defective robots is, for example, addressed in [5, 4]. Exogenous security issues, such as attacks on the swarm, were only studied in later work. For example, in [24], a reputation management system is developed where each robot keeps a trust level about every other robot based on an agreed upon protocol. A comparison of a swarm's robustness to attacker strategies in a collective navigation scenario is studied in [17]. The required connectivity for achieving consensus when malicious robots are present, is determined in [8]. A resilient consensus protocol for agents with a dynamic network topology is presented in [16].

The idea of using blockchain technology for addressing security challenges in robot swarms was first proposed in [3]. The author describes several features of blockchain technology that could provide decisive advantages in real-world deployments (e.g., secure communication, data logging, and consensus agreement). In [19], this theoretical notion was supported with an implementation and proof-of-concept of how to manage Byzantine robots via blockchain technology in a discrete consensus achievement problem.

3 Blockchain technology for robot swarms

In this section, we briefly summarize characteristics of blockchain technology and describe modifications and requirements for using blockchain technology in swarm robotics applications. In [19], we describe the fundamentals of blockchain technology in more detail. Further information about blockchain technology and smart contracts in general can be found in [13, 2].

A blockchain is a tamper-proof decentralized system used as database and computing platform. Originally devised for the digital currency Bitcoin, later adaptations used the underlying technology for other specific, non-financial applications (e.g., voting, identity management, and so on). In 2014, the *Ethereum* framework was released, which allows for running arbitrary Turing-complete applications via blockchain technology (blockchain-based smart contracts). A blockchain-based smart contract is a container that encapsulates variables and functions executed via blockchain technology. We use *Ethereum* via *geth* (a tool for running an Ethereum node, implemented in the Go programming language) in our implementations.

The participants (robots in this work) of a blockchain network locally keep a copy of the blockchain. They create transactions and distribute them among their peers. The data of the transactions is then used as input to the functions of smart contracts. A transaction contains the signature of the address belonging to the sender. Therefore, it can be unambiguously assigned to a specific participant (robot) and an attacker cannot create transactions under a false identity.

The participants can have different blockchain versions. For example, during the experiments conducted in the scope of this research, the information written in the blockchain differs among the robots, since the robots are not always close enough to communicate with each other. Ethereum achieves consensus by agreeing on the longest blockchain, that is, the one that required the highest Proof-of-Work (PoW). PoW requires the participants to solve a computational puzzle (finding a hash value below a target value using transactions as input to the hash function). The process of solving this puzzle is called *mining*; the number of hashes a participants can compute per second is stated by its hash power. This ensures that writing information into the blockchain is computationally expensive. Via the PoW-based consensus protocol, the different blockchain versions of the robot can be resolved.[1]

It depends on the context whether a PoW-based consensus protocol is adequate. If no intruder is able to enter the swarm (e.g., due to a permissioned blockchain system with an access control layer), PoW is suitable: as long as the majority of nodes is honest, the data in the blockchain can be trusted. If, however, an intruder can outperform the hash power of the remaining robots (51 % attack), it can change the order of the transactions and decide whether or not transactions should be included in the blockchain. To summarize, when using PoW, the information written in the blockchain can be trusted as long as no powerful attacker can compromise the network. Additionally, even if robots break during the experiment, their data is securely stored in the blockchain.

Using available blockchain protocols like Ethereum, robots need to meet certain requirements in terms of communication, processing, and storage. The size of one transaction is around 160 Bytes. In order to communicate with each other, robots should be able to send and receive some kB/s, otherwise, they may not be able to synchronize their blockchain states in an adequate amount of time. We studied the development of the size of the block-

[1] The longest blockchain, that is, the one that required the higher PoW, gets accepted as the true blockchain, while the shorter blockchains get discarded. Transactions that were in the shorter blockchains but are not yet in the longest blockchain can become part of a block added later to the blockchain.

chain over time in our experiments. During our experiments, the blockchain grew up to 7.5 MB, a size which could be stored on state-of-the-art robots in swarm robotics (e.g., the e-puck robot platform [11]) without difficulty.

We use an auxiliary node for publishing the smart contract to the blockchain at the beginning of each experimental run. The auxiliary node then mines the smart contract and distributes the contract address among the robots (i.e., the contract is written into the configuration file of the robots).

Additionally, the robots need *ether* (Ethereum's cryptocurrency, i.e. immutable tokens stored in the blockchain ledger) to be able to send transactions to the blockchain. To this end, a custom genesis block (i.e., the first block in the blockchain) is used, which allocates ether to the addresses associated with the robots. In this paper, we allocate 100 ether to each robot, which ensures that there are no limitations on the amount of transactions the robots can send to the blockchain.

In order to adapt the mining difficulty to the limited computational of the (simulated) robots and to keep it at a constant rate (in order not to introduce an additional variable) mining difficulty of Ethereum was set to a fixed value.

4 Methods

4.1 Experimental setup

In the experiments, the swarm's goal is to estimate the relative frequency of black tiles in an environment where the floor is covered with B black and W white tiles, $B+W=400$. The task difficulty (ρ_b^*) is given by the ratio of black to white cells: $\rho_b^* = \frac{B}{W}$ (in all our experiments, 'white' is the most frequent tile color, so that $0 \leq \rho_b^* < 1$). The positions of the black and white tiles and the positions of the robots are randomly chosen at the beginning of each experimental run. In each time-step, a robot determines if it is above a black or a white tile via its ground sensor.

The complete implementation of the presented approach is hosted on GitHub[2]. $N=20$ e-puck robots [11] are used in the robot swarm simulator ARGoS [14] (version 3.0.0-beta48) with the plugin ARGoS-epuck [7]. ARGoS uses discrete time-steps (ten per second in our experiments) to simulate the actions of the robots. The experiments were conducted on a computer cluster with 32 cores. A private Ethereum network is used for the experiments and each robot is allocated 100 ether[3] at the beginning of each experimental run using a custom genesis block. For each robot, the Ethereum implementation *geth* is executed using one thread.

The robots move in the environment using a random walk routine. Each robot mines (i.e., it performs the Proof-of-Work) from the start to the end of an experimental run. In order to simulate the communication of real robots, simulated robots have the ability to connect to each other's Ethereum processes if their distance is smaller than 50 cm; they can then exchange their blockchain information (blocks and transactions).

The shared knowledge in the blockchain is provided via a blockchain-based smart contract. Each robot works in exploration *phases*. The duration of each exploration phase is 30 seconds. A robot calculates the ratio between the number of black tiles and the total amount of tiles it sensed in this exploration phase: $\hat{\rho}_b(i,m) = \frac{\hat{B}_{i,m}}{\hat{B}_{i,m}+\hat{W}_{i,m}} \in [0,1]$ (we use (i,m) and the subscript notation $_{i,m}$ for variables of a robot i in its mth exploration phase). The blockchain-based smart contract provides a function called `vote`. It accepts a value be-

[2] `https://github.com/Pold87/blockchain-estimation`

[3] Ethereum's cryptocurrency, which is required for sending transactions

tween 0.0 and 1.0 as an argument[4]. At the end of each exploration phase, a robot interacts with the function `vote` by sending a transaction with $x_{i,m}$ as input; non-Byzantine robots send $x_{i,m} = \hat{\rho}_b(i,m)$, while Byzantine robots send $x_{i,m} = 0.0$.

Every time a robot sends a `vote` transaction to the smart contract, the mean $\bar{x}_t$ and standard error of the mean $s_{\bar{x}} = \frac{s}{\sqrt{n}}$ are stored and calculated via the smart contract. These statistics are updated via a single pass *online algorithm* to reduce their computational requirements [22]. Using this online algorithm, the mean is updated via $\bar{x}_{t+1} := \bar{x}_t + \frac{1}{n+1}(x_{i,m} - \bar{x}_t)$. The sum of squares of differences from the current mean $M_{2,t+1} := M_{2,t} + (x_{i,m} - \bar{x}_t)(x_{i,m} - \bar{x}_{t+1})$ and the sample variance $s^2_{t+1} := \frac{M_{2,t+1}}{n}$ are used in an intermediate step to calculate the standard error of the mean $s_{\bar{x},t+1} := \frac{s_{t+1}}{\sqrt{n+1}}$.

The standard error of the mean is used as uncertainty metric; as soon as $s_{\bar{x},t}$ drops below a threshold τ, the blockchain event `consensusReached` is set to `true`. At the end of each exploration phase, a robot queries the status of the event `consensusReached`. If the value is `true`, the robot stops to explore the environment and only continues the random walk. As soon as all N robots received `true` from the `consensusReached` event, the experimental run is stopped and the total time (consensus time) is recorded.

4.2 Byzantine robots and reputation management

In Experiment 3 (Section 5.3), we study the influence of Byzantine robots. While there are many possible Byzantine failures, in this paper, a Byzantine robot sends a quality estimate of $x_{i,m} = 0.0$ in all exploration phases m independent of its actual sensor readings.

To be able to identify these Byzantine robots, a reputation management system is implemented via the smart contract (i.e., the reputation of the individual robots is stored and managed via the blockchain). The idea of the reputation management system is that a robot with a properly functioning sensor is likely to increase its reputation over the course of an experimental run, while a Byzantine robot's reputation value is likely to decrease.

Every time a robot i sends a `vote` transaction (with $x_{i,m}$ as input) at the end of its exploration phase, its reputation $r_{i,t}$ gets updated. To this end, the absolute difference between the value sent by the robot and the mean of all sent values of all robots ($\bar{x}_t$) stored in the smart contract is calculated ($|\Delta x| = |x_{i,m} - \bar{x}_t|$). This difference is then used to update the robot's reputation value (a small difference will increase a robot's reputation, while a large difference will decrease its reputation): $r_{i,t+1} := r_{i,t} - u(|\Delta x|) = r_{i,t} - a|\Delta x|^2 + b$. The variables a and b are design variables that determine the shape and offset of u, respectively (we used $a = 4$, $b = 0.3$ see Figure 2). For $u(|\Delta x|) > 0$, a robot's reputation increases, otherwise the reputation decreases. With $a = 4$, $b = 0.3$, the reputation increases, if $|\Delta x| < 0.273$. At the beginning of each experimental run ($t = 0$), each robot i gets assigned the initial reputation value $r_{i,0} = 1.0$.

Once the reputation value of a robot drops below a threshold $r_{min} = 0.0$, its votes are ignored for the rest of the experimental run.

The reputation value $r_{i,t+1}$ of a robot is used as weight in the calculation of the weighted mean $\bar{x}_{t+1} := \bar{x}_t + \frac{r_{i,t+1}}{\mathscr{R}}(x_{i,m} - \bar{x}_t)$, where $\mathscr{R} = \sum_i \sum_t r_{i,t}$ is the total sum of reputation values.

4.3 Metrics

The following metrics are used to determine the performance of our approach:

- Mean absolute error $\mathrm{MAE} = |\rho_b - \bar{x}_{*,\infty}|$: this statistic is the absolute value of the differ-

[4] Smart contracts in Ethereum accept and handle integer values only. Therefore, in the actual implementation, we multiply all values by 10^6.

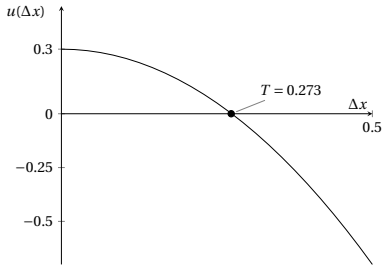

Figure 2: The figure shows how the difference between the value sent by the robot and the mean written in the blockchain ($|\Delta x| = |x_{i,m} - \bar{x}_t|$) influences the update of the reputation value $u(\Delta x) = \Delta r = r_{i,m+1} - r_{i,m}$. If $|\Delta x|$ is smaller than $T = 0.273$, the robot's reputation increases, otherwise the reputation decreases.

ence between actual relative frequency of black tiles (ρ_b, set by the experimenter) and the collective swarm estimate ($\bar{x}_{*,\infty}$) at the end of an experimental run. To obtain the collective swarm estimate, one robot is randomly chosen, and the mean as written in this robot's blockchain is used.

- Consensus time T_N: this statistic is the time in seconds until all robots have received the stop signal.
- Blockchain size $\mathrm{BC}_{\mathrm{MB}}$: this statistic indicates the blockchain size in MB of one randomly chosen robot, determined at the end of each experimental run.
- Exit probability E_N: this statistic states the ratio of runs in which the robots correctly determined that 'white' is the most frequent tile color. For its calculation, the *discrete* collective swarm estimate is used: this value is 1 if the collective estimate is smaller than 0.5 (i.e., the robots correctly determined that there are more white than black tiles) and 0 otherwise.

5 Experiments

5.1 Experiment 1 – Increasing the relative frequency of black cells

The goal of Experiment 1 is to provide a proof-of-concept of the shared knowledge approach; we determine if the shared knowledge system can be used to estimate the relative frequency of the color of tiles. We vary the relative frequency of black tiles (ρ_b) to test the approach. In this experiment, the reputation management system is not used (i.e., the weights are fixed: $\forall m : r_{i,m} = 1.0$).

The actual vs. predicted plot shows that the median value of the collective swarm estimate $\bar{x}_{*,\infty}$ is very close to the actual values ρ_b (dashed lined in Figure 3 top left): the variability is small. A low mean absolute error of approximately 0.01 was achieved for all values of ρ_b. The measured statistics are independent of ρ_b and, hence, independent of the difficulty of the task. The median value of the consensus time is 300 seconds. The median blockchain size is 0.6 MB. The exit probability is $E_N = 1.0$ for all values of ρ_b, except for $\rho_b = 0.48$, where it slightly decreases to 0.95. The high and stable E_N is in contrast to the modulation-based collective decision-making approaches studied in [20, 19], where E_N drops with higher difficulty levels.

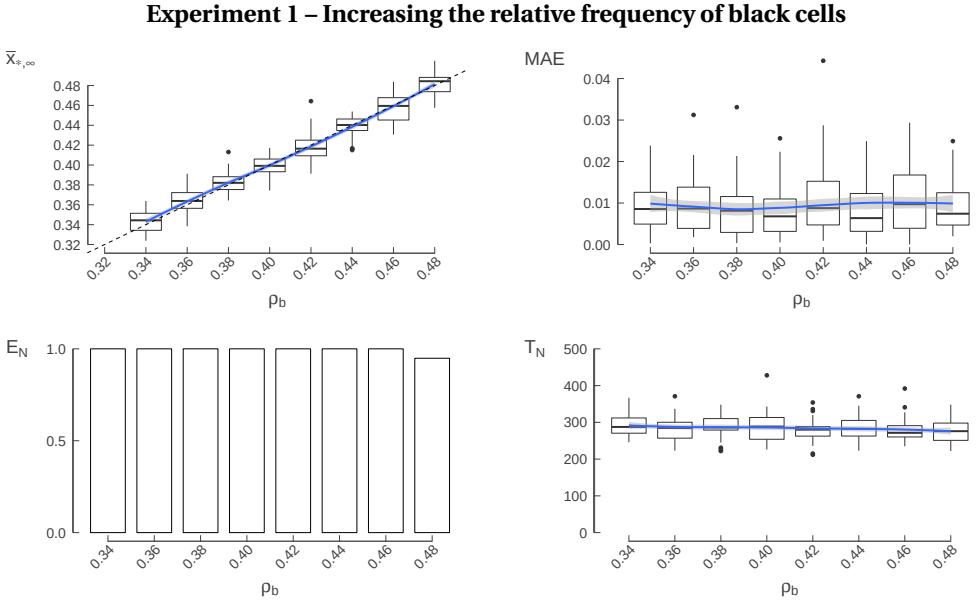

Figure 3: The collective swarm estimate $\bar{x}_{*,\infty}$ is accurate for all relative frequencies of black tiles ρ_b: the mean absolute error (MAE) is low. The discretized collective swarm estimate yields a perfect exit probability E_N of 1.0 for all $\rho_b < 0.48$, with a slight decrease to 0.95 at $\rho_b = 0.48$. The consensus time T_N is independent of ρ_b. The blue curves are fitted via local regression, the gray areas represent the 95 % confidence interval. 30 repetitions of the experiment were executed for each value of ρ_b.

5.2 Experiment 2 – Influence of the threshold τ

In Experiment 2, we study the influence of the threshold parameter τ on (i) the mean absolute error, (ii) the size of the blockchain at the end of one experimental run, and (iii) the consensus time. A fixed relative frequency of black tiles was used ($\rho_b = 0.40$).

The results show that τ may be used for trading off memory and speed with accuracy and variability (Figure 4). Using the smallest threshold $\tau = 0.006$, the blockchain size can grow up to around 7.5 MB with a median consensus time of 780 seconds. The small threshold value leads to a low mean absolute error (≈ 0.006) with a low variability. With increasing τ, the blockchain size decreases, requiring only 335 kB with $\tau = 0.030$; the consensus time decreases to 175 seconds at $\tau = 0.030$. However, increasing τ results in a higher mean absolute error (≈ 0.017 at the highest setting), a higher variability of the error term, and a higher probability of outliers.

5.3 Experiment 3 – Reputation management system and influence of Byzantine robots

In Experiment 3, we study the influence of the number k of Byzantine robots on a non-secure and a secure approach. Using the non-secure approach, the reputation value $r_{i,m}$ is fixed for all i and m. Using the secure approach, $r_{i,m}$ gets updated as described in Section 4.2. The relative frequency of black cells is fixed at $\rho_b = 0.40$. The length of one experimental run was set to 450 seconds in order to control differences due to varying consensus times.

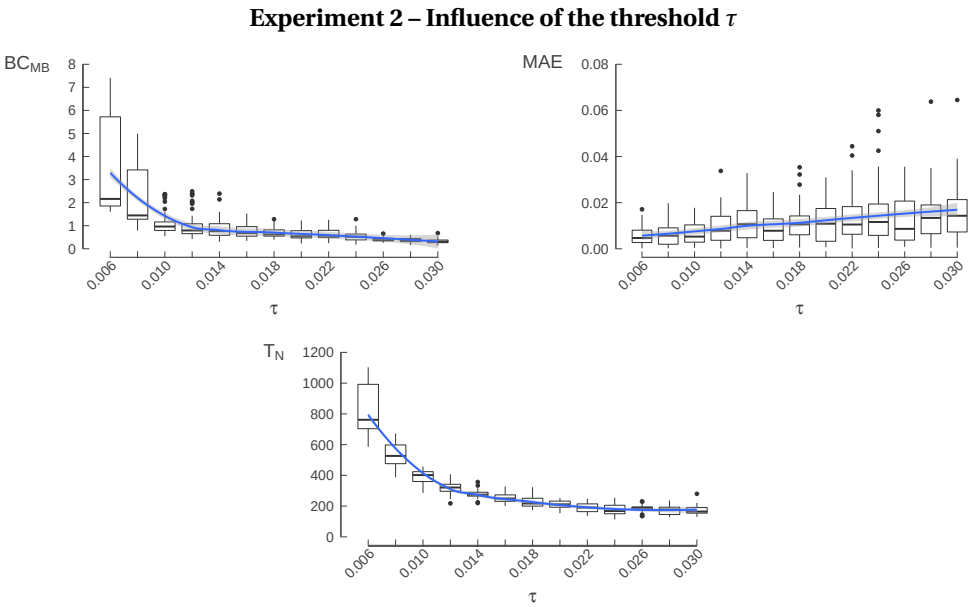

Figure 4: The uncertainty threshold τ achieves a trade-off between blockchain size BC_{MB}/consensus time T_N and mean absolute error MAE. The blue curves are fitted via local regression, the gray areas represent the 95 % confidence interval. 30 repetitions of the experiment were executed for each value of τ.

The comparison between the non-secure and secure approaches shows the advantage of the reputation management system (Figure 5). While the performance of the non-secure approach linearly deteriorates when increasing the number k of Byzantine robots, the secure approach is stable for smaller values of k. However, it can present a stronger deterioration when k is large (that is, when almost half of the robots in the swarm are Byzantine robots).

6 Discussion and Conclusions

In this paper, we demonstrated a blockchain-based shared knowledge system for robot swarms. The approach yields short consensus times and a high accuracy—even in the presence of a few Byzantine robots, which are managed using a reputation system. The blockchain stores critical events during an experiment and then can be used for detailed insights into the course of the events after the experiment is finished.

In a permissionless blockchain network, new blockchain addresses (pseudo-identities) can be generated at any time. However, using the presented reputation management system, there is an incentive to keep an address to achieve a higher reputation over time. In future work, we will refine the reputation management system and study further Byzantine failures.

We argue that having a medium of shared knowledge can possibly facilitate the implementation of several swarm robotics algorithms and can pave the way for novel swarm robotics applications (e.g., computationally lightweight machine learning algorithms). The use of blockchain technology as decentralized tamper-proof reputation management system may be useful in a wide range of tasks.

Experiment 3 – Reputation management system and influence of Byzantine robots: non-secure approach (left) vs. secure approach (right)

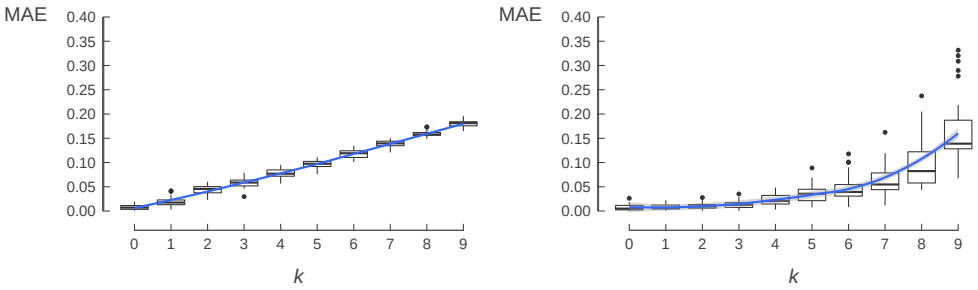

Figure 5: While the mean absolute error (MAE) increases linearly with the number of Byzantine robots k (robots that send a quality estimate of $x_{i,m} = 1.0$, independent of their sensor readings) when using the non-secure approach (left), the secure approach (right) remains largely unaffected when a few Byzantine robots are part of the swarm. The blue curves are fitted via local regression, the gray areas represent the 95 % confidence interval. 30 repetitions of the experiment were executed for each value of k.

However, a shared knowledge medium introduces additional computational and memory requirements. Therefore, it may not be suitable for all swarm robotics tasks and systems. The additional requirements depend on the number of participants in the network and the amount of information that is sent to the blockchain. Therefore, it is important to determine which information is security-relevant and should be stored on the blockchain, and which information can be locally processed by the robots.

In future work, we will transfer the blockchain system to physical e-puck robots and study the possibilities of consensus protocols tailored to robot swarms.

7 Acknowledgements

Volker Strobel and Marco Dorigo acknowledge support from the Belgian F.R.S.-FNRS and from the FLAG-ERA project RoboCom++.

References

[1] Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: A review from the swarm engineering perspective. Swarm Intelligence **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. (2014), `https://github.com/ethereum/wiki/wiki/White-Paper`

[3] Castelló Ferrer, E.: The blockchain: A new framework for robotic swarm systems. pre-print (2016), arXiv:1608.00695v3

[4] Christensen, A.L., O'Grady, R., Birattari, M., Dorigo, M.: Fault detection in autonomous robots based on fault injection and learning. Autonomous Robots **24**(1), 49–67 (2008). https://doi.org/http://dx.doi.org/10.1007/s10514-007-9060-9

[5] Christensen, A.L., O'Grady, R., Dorigo, M.: From fireflies to fault-tolerant swarms of robots. IEEE Transactions on Evolutionary Computation **13**(4), 754–766 (2009). https://doi.org/http://dx.doi.org/10.1109/TEVC.2009.2017516

[6] Ferrante, E., Turgut, A.E., Huepe, C., Stranieri, A., Pinciroli, C., Dorigo, M.: Self-organized flocking with a mobile robot swarm: a novel motion control method. Adaptive Behavior **20**(6), 460–477 (2012). https://doi.org/10.1177/1059712312462248

[7] Garattoni, L., Francesca, G., Brutschy, A., Pinciroli, C., Birattari, M.: Software infrastructure for E-puck (and TAM). Tech. Rep. 2015-004, IRIDIA, Université libre de Bruxelles (2015)

[8] Guerrero-Bonilla, L., Prorok, A., Kumar, V.: Formations for resilient robot teams. IEEE Robotics and Automation Letters **2**(2), 841–848 (Apr 2017). https://doi.org/10.1109/LRA.2017.2654550

[9] Higgins, F., Tomlinson, A., Martin, K.M.: Survey on security challenges for swarm robotics. In: Proc. Fifth Int. Conf. Autonomic and Autonomous Systems, pp. 307–312. IEEE Press (Apr 2009). https://doi.org/10.1109/ICAS.2009.62

[10] Millard, A.G., Timmis, J., Winfield, A.F.T.: Towards exogenous fault detection in swarm robotic systems. In: Towards Autonomous Robotic Systems - Proceedings of TAROS 2013 - 14th Annual Conference. Lecture Notes in Computer Science, vol. 8069, pp. 429–430. Springer (2014). https://doi.org/10.1007/978-3-662-43645-5_44

[11] Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Gonçalves, P.J.S., Torres, P.J.D., Alves, C.M.O. (eds.) Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions. vol. 1, pp. 59–65. IPCB: Instituto Politécnico de Castelo Branco (2009)

[12] Montes de Oca, M.A., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M., Dorigo, M.: Majority-rule opinion dynamics with differential latency: A mechanism for self-organized collective decision-making. Swarm Intelligence **5**(3–4), 305–327 (2011). https://doi.org/http://dx.doi.org/10.1007/s11721-011-0062-z

[13] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), `https://bitcoin.org/bitcoin.pdf`

[14] Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L.M., Dorigo, M.: ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intelligence **6**(4), 271–295 (2012). https://doi.org/http://dx.doi.org/10.1007/s11721-012-0072-5

[15] Reina, A., Dorigo, M., Trianni, V.: Collective decision making in distributed systems inspired by honeybees behaviour. In: Proceedings of 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), pp. 1421–1422. Int. Foundation for Autonomous Agents and Multiagent Systems (2014)

[16] Saldaña, D., Prorok, A., Sundaram, S., Campos, M.F.M., Kumar, V.: Resilient consensus for time-varying networks of dynamic agents. In: Proc. American Control Conf. (ACC), pp. 252–258. IEEE Press (May 2017). https://doi.org/10.23919/ACC.2017.7962962

[17] Sargeant, I., Tomlinson, A.: Maliciously manipulating a robotic swarm. In: Proc. of ESCS'16 – The 14th Intern. Conf. on Embedded Systems, Cyber-physical Systems, & Applications, pp. 122–128. CSREA Press (2016)

[18] Soysal, O., Sahin, E.: Probabilistic aggregation strategies in swarm robotic systems. In: Proceedings of 2005 IEEE Swarm Intelligence Symposium (SIS 2005). pp. 325–332 (June 2005). https://doi.org/10.1109/SIS.2005.1501639

[19] 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 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), p. to appear. Int. Foundation for Autonomous Agents and Multiagent Systems (2018)

[20] Valentini, G., Brambilla, D., Hamann, H., Dorigo, M.: Collective perception of environmental features in a robot swarm. In: Swarm Intelligence – Proceedings of ANTS 2016 – Tenth Inter-

national Conference. Lecture Notes in Computer Science, vol. 9882, pp. 65–76. Springer (2016). https://doi.org/https://doi.org/10.1007/978-3-319-44427-7_6

[21] Valentini, G., Ferrante, E., Dorigo, M.: The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives. Frontiers in Robotics and AI **4**, 9 (2017). https://doi.org/10.3389/frobt.2017.00009, http://journal.frontiersin.org/article/10.3389/frobt.2017.00009

[22] Welford, B.P.: Note on a method for calculating corrected sums of squares and products. Technometrics **4**(3), 419–420 (1962), http://www.jstor.org/stable/1266577

[23] Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper. (2014), http://gavwood.com/paper.pdf

[24] Zikratov, I., Maslennikov, O., Lebedev, I., Ometov, A., Andreev, S.: Dynamic trust management framework for robotic multi-agent systems. In: Galinina, O., Balandin, S., Koucheryavy, Y. (eds.) Proc. of 12th Int. Conf. on Next Generation Teletraffic and Wired/Wireless Advanced Networking, NEW2AN, and the 5th Conf. on Internet of Things and Smart Spaces, ruSMART 2016, pp. 339–348. Springer (2016)