LEDGER
ledgerjournal.org

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

PROCEEDINGS ARTICLE

Grex: A Decentralized Hive Mind

Alex Khawalid,*† Dan Acristinii,‡ Hans van Toor,§ Eduardo Castelló Ferrer¶

Abstract. Swarm Robotics (SR) faces a series of challenges impeding widespread adoption for real-world applications. Distributed Ledger Technology (DLT) has shown it can solve a number of these challenges. An experiment was conducted to showcase the resolution of these challenges. A search and rescue mission was simulated using drones coupled with single board computers and several simulated agents. Inter-agent communications were facilitated through DLT in a completely decentralized network. A frontend interface was built to demonstrate the ease with which information can be extracted from the system. This paper shows the feasibility of the application of DLT to SR-related challenges in a practical experiment. For future work, it is proposed to focus on more complex tasks through federated learning or inter-swarm communications, possibly through Cosmos.

1. Introduction

The application of Swarm Robotics (SR) has the potential to improve processes across different industries. Recent research has showcased the application of SR to aircraft engine maintenance,[1] agriculture,[2] and space exploration.[3] Barca *et al.*, "Swarm Robotics Reviewed" (2013),[4] reviewed a series of challenges that impede widespread adoption of SR systems for real-world applications. In this paper, six challenges are identified that impede real-world applications of SR (see Table 1 below).

The inspiration for this paper came from "The Blockchain: A New Framework for Robotic Swarm Systems," written by Eduardo Castelló Ferrer.[5] This paper aims to demonstrate the feasibility of implementing Distributed Ledger Technology (DLT) in an SR system and addresses the challenges that DLT can ease.

This paper will discuss relevant research about the use of DLT with SR. Following this, the experiment conducted by Kryha at the 2018 Blockchaingers Hackathon will be discussed.[6] The results from the experiment will be presented and discussed. Finally, conclusions will be presented and future research will be suggested.

* bc1qf2l4274wv8pl422nyahsu6tdtfuverr6wsgn6h
† A. Khawalid (alex@kryha.io) is a blockchain engineer at Kryha, in Amsterdam, Netherlands: `https://kryha.io`.
‡ D. Acristinii (dan@kryha.io) is head of research and development at Kryha.
§ H. van Toor (hans@tesnetwork.io) is the founder of TES Network, in Amsterdam, Netherlands: `https://tesnetwork.io`.
¶ E. Castelló Ferrer (ecstll@media.mit.edu) is post-doctoral researcher at MIT Media Lab, MIT, Cambridge, USA.

Table 1. A series of challenges that impede widespread adoption of SR for real-world applications.

#	Challenge	Description	Source
1	Hybrid system	Building a hybrid system, which combines the benefits from fully decentralized and fully centralized systems.	Barca (2013) p. 353[4]
2	Supervising SR systems	Supervising a fully decentralized SR system on an abstract level, first-time-right.	Barca (2013) p. 353[4]
3	Scalability and robustness	The system must be able to accommodate a large number of agents without being prone to errors.	Barca (2013) p. 353[4]
4	Accountability, transparency, and trust	As SR systems are autonomous, implementing accountability and transparency is complicated, making it more difficult to place trust in them.	Calvaresi *et al.* (2018) p. 8,[7] Castelló Ferrer (2016) p. 4[5]
5	Implementation of SR and DLT	The majority of published papers propose conceptual solutions and describe implementation as a challenge	Calvaresi *et al.* (2018) p. 10[7]
6	Sustainable SR system	The system should stay operational even if energy sources are depleted before the given task is completed.	Barca (2013) p. 353[4]

2. Relevant Research

This section discusses the identified relevant research. Firstly, Distributed Ledger Technology is introduced. Secondly, the use of DLT with SR is discussed, explaining the relevant technologies. Thirdly, the challenges faced by SR are discussed. Finally, research combining DLT and SR is explored. These challenges are addressed through the combination of DLT and SR. Related work done in the field of SR as well as relevant research are highlighted to showcase the significance of this research.

2.1. Distributed Ledger Technology—In 2008, Satoshi Nakamoto published a paper containing the outline for the concept of Bitcoin. One of the more significant ideas outlined in this paper is at present referred to as a blockchain. A blockchain is an immutable public ledger, which is shared among agents running nodes. Each of these nodes keeps a full copy of the ledger, which contains all of the transactions made on the network.[8] Then a consensus algorithm is used so the agents can agree on what the correct state of the public ledger is. DLT is a superset that subsumes blockchains, the latter referring only to distributed, append-only, decentralized, and replicated databases. In contrast to blockchains, DLT also encompasses private networks or networks not necessarily following a block structure to record transactions.

2.2. Distributed Ledger Technology in Swarm Robotics—In recent years, multiple papers have been written on the topic: Distributed Ledger Technology (DLT) in Swarm Robotics (SR). Two of these will be highlighted in the following section.

Firstly, Strobel *et al.* "Managing Byzantine Robots via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario" (2018), describes an environment simulating a swarm using ARGoS, where each agent is running a full Geth (Go Ethereum implementation) node on a PoW Ethereum network.[9] Agents could communicate within a distance of 50cm. The objective for the agents was to estimate the ratio of black to white tiles on a randomly generated grid of tiles. Agents selected moves using the random walk algorithm. A smart

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

contract implemented a reputation system based on standard deviation from the mean, where a higher standard deviation resulted in a lower reputation. The influence of Byzantine agents within the system was tested. The results showed higher stability in mean average error, as well as a lower mean average error for a low number of Byzantine agents (p. 9).[9]

Secondly, Cameron *et al.*, "Research and Implementation of Multiple Blockchain Byzantine Secure Consensus Protocols for Robot Swarms" (2018) conducts an experiment with DLT and SR by simulating "*the harvesting game*." The harvesting game starts with a grid which has two possible states: dirt or crops. Agents had to decide when to harvest or sow the squares in the grid. The experiment used a heterogeneous SR system, running on two separate private DLT systems. There were two types of agents: harvestBots, and surveyBots. HarvestBots called a smart contract used by surveyBots to learn the state of the grid. The consensus on the state of the grid among harvestBots was however decoupled from the DLT consensus. A voting mechanism was implemented in a smart contract on the harvestBot DLT; this mechanism was used to reach consensus among harvestBots.[10]

2.3. Challenges—Table 1 shows the challenges that were identified which impede widespread adoption of SR in real-world applications. The following section describes the challenges and their relation to DLT. The first challenge suggests that fully centralized systems could negatively impact the robustness of SR systems as the loss of "*leaders*" could shut the entire system down. Fully decentralized systems suffer from an inability to synthesize local sensory data into global knowledge which can be used by members of the swarm to make decisions. DLT resolves both of these issues by making use of certain consensus algorithms such as Proof of Work or Tendermint.[11,12] In a swarm system in which all agents run full DLT nodes and are miners, all agents are leaders. No single leader is needed to label a source of data as valid or invalid; thus the loss of leaders does not necessarily shut down the system.

The second challenge addresses the challenge of supervising an SR system. Part of the second challenge addresses data extraction. In most decentralized settings, it is difficult to extract data generated by an SR system and process it into information that can be used to supervise the system.[5,9] DLT replicates the ledger across all agents and propagates updates to agents within communication range.[8] DLT allows for the ability to generate a complete overview of the system since all agents have a copy of the entire ledger.[8]

The third challenge suggests scalability and robustness of the system as issues. Distributed ledgers allow all of the submitted data to be extracted, as long as a single agent can be accessed, making DLT somewhat robust. This extraction of data is possible through the property of DLT, as mentioned above, that propagates the entire ledger to all agents. Moreover, distributed ledgers are decentralized to a certain degree. Consensus about the state of the shared ledger is reached in a distributed manner,[8] thus ensuring that the failure of a limited amount of individual agents has no bearing on the ability to propagate data, given that the remaining agents still have a means of communicating with each other.

The fourth challenge refers to accountability, transparency, and trust of SR systems. DLT adds to the security, transparency, and accountability of any SR system. Distributed ledgers, such as Ethereum and BigchainDB are append-only, thus guaranteeing a moderately transparent system by ensuring the history of events cannot be retroactively changed.[12,13] Moreover, if all inter-agent communications go through a DLT, they will be processed as transactions submitted by a specific public key. Thus, allowing each transaction to be traced back to a single user.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

Additionally, consensus algorithms, such as Tendermint, are Practical Byzantine Fault Tolerant (PBFT),[12] thus guaranteeing that even if up to a third of all agents become malicious, the network will still function correctly. This guarantee provides a form of security, thus allowing for more trust in the system.

The fifth challenge acknowledges that the majority of research on the combination of SR and DLT is at the conceptual level. This challenge can be addressed by implementation of SR and DLT in practical experiments with physically embodied agents. Hardware that can be implemented with limited resources is suitable for SR, as swarms usually operate with a large number of agents.

Finally, the sixth challenge refers to the time available to complete tasks for agents. Often, agents have a limited power supply (*e.g.*, a battery) while operational. This challenge will be more difficult to overcome with the use of DLT. DLT requires a large amount of communication between agents and therefore a significant amount of energy is used. Furthermore, depending on the consensus algorithm used, for instance, Proof of Work (PoW), DLT requires heavy computation which also increases energy consumption.[11] Currently, alternative consensus algorithms, such as Proof of Stake and Proof of Kernel Work, are available that are more energy efficient.[14,15] The energy efficiency of DLT systems is actively being researched by organizations such as Intel, IBM, and MIT.[16]

The use of DLT eases the difficulty of overcoming the challenges mentioned in Table 1. DLT can potentially have a positive influence on the following challenges:

- Challenge 1: balance centralized and decentralized control
- Challenge 2: allow for easier extraction of information
- Challenge 3: scale to a large number of transactions
- Challenge 3: handle massive failure among agents
- Challenge 4: increase accountability of agents
- Challenge 4: become more trustworthy through its append-only property
- Challenge 5: be implemented on relatively simple hardware

However, the relevant research suggests that DLT will not have a positive influence on the challenge:

- Challenge 6: energy consumption within these systems

3. Methods

The experiment conducted in this paper builds upon previous work, such as of Strobel *et al*. (2018) by employing physical agents,[9] and Cameron *et al*. (2018) by running full nodes and using a different consensus algorithm, BigchainDB.[10] In both Strobel *et al.* (2018) and Cameron *et al*. (2018), the Ethereum platform was used as the DLT platform.[9,10] In both of these papers, the focus was mostly on consensus with Byzantine agents involved. However, the focus of the present experiment was to showcase the added value of the combination of SR and DLT, through the implementation of such a system. The primary focus of the experiment—as described in

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

this paper—was on the fifth challenge, implementation of SR and DLT, in order to demonstrate implementation of DLT to an SR system. The secondary focus was on the second challenge (supervision), the third challenge (robustness), and the fourth challenge (transparency, trust, accountability).

Section 3.1 describes the search and rescue mission that was simulated. Section 3.2 describes the software architecture including BigchainDB and the frontend. Section 3.3 describes the hardware setup.

3.1. Mission—The experiment described in this paper simulated a search and rescue mission. The objective of the experiment was for a swarm of agents to collaborate on a rescue operation by exploring and mapping an area (search), identifying points of interest, and swarming to the location of these points (rescue).

The search and rescue mission consisted of two tasks; Search and Rescue (See Figure 1). In the search phase, the search and rescue drones randomly picked new locations to move to (Task 1: Search). Once a drone found a point of interest (*e.g.*, a distress signal), it registered the event in the world state.

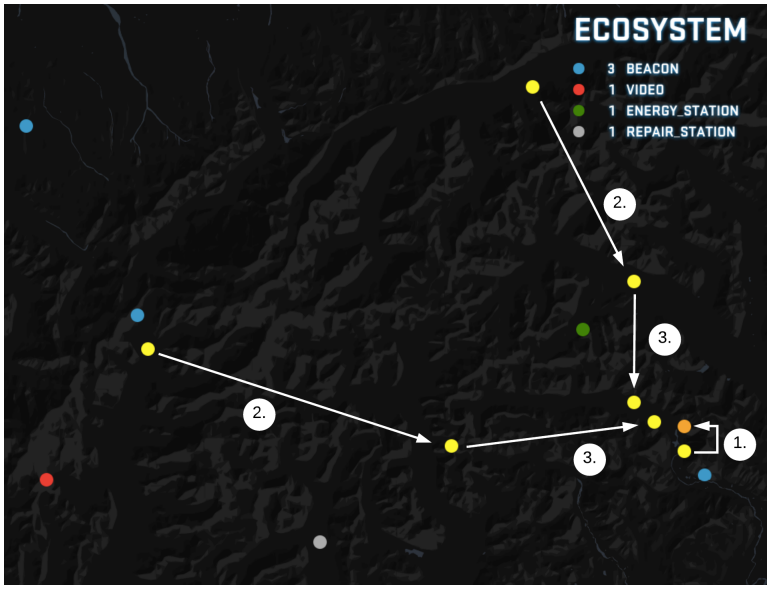


Fig. 1. Search and Rescue Mission. Step 1: One drone detects an object. Step 2: The other drones move towards the tagged location. Step 3: All the drones converge upon the location, concluding the mission.

The rest of the drones entered a rescue state and flocked to the location of this drone to theoretically perform a rescue operation (Task 2: Rescue).

The information propagated through DLT pertains to the world state observed by the agents, such as obstacles and points of interest. Additionally, information such as the path taken by an agent, its current activity, and its battery level were also stored in the ledger.

3.2. Software—BigchainDB was chosen as the DLT platform, primarily due to the fact that BigchainDB has a higher transaction throughput (tx/s), an order of magnitude higher than Ethereum. In testing it has reached at least 298 tx/s and can theoretically reach up to a 1000 tx/s,[17] as opposed to the 15 tx/s that the standard Ethereum implementation has reached.[17,18] Therefore, when compared to Ethereum, BigchainDB increases scalability, one of the challenges facing SR, see Table 1, Challenge 3. Additionally, BigchainDB takes an asset-based approach

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

to DLT, which allows for a shared asset-based representation of all agents and their knowledge. This representation allows any parties connected to the network, to easily access and process these representations into an interface.

BigchainDB allows all transactions to be performed upon an asset. Each transaction is either a create or transfer type of transaction. Moreover, it uses the Tendermint consensus algorithm as opposed to the Proof of Work algorithm used by platforms like Ethereum. This consensus is beneficial in this setting because Tendermint is computationally less taxing. Thus it will consume less energy. Proof of Work was engineered to do more computations,[11] consuming more energy in the process. Tendermint focuses on reaching consensus in the presence of malicious agents.

Furthermore, it is also PBFT,[12] this means that even if up to one-third of the agents fail or act maliciously, the network will continue processing transactions. Finally, there are tools available to describe and treat assets as an Object-Relational Mapping (ORM). Jernej Pregelj, creator of the ORM module of BigchainDB, describes ORM as "an abstraction layer where database items are represented as programming language objects with variables as data, relations between items and functions that represent operations on that specific item in the database."[19] This approach allows the agents, as well as shared information of their surroundings to be described using the ORM, rather than interfacing with transactions directly.

The results of the experiment will be captured through a script using a WebSocket listener on a node connected to the network. The script logged certain aspects of the transactions every time a move transaction was propagated through the network. The data captured by the script is as follows: number of agents active, total transactions, transactions per agent, agents that have completed the objective, time elapsed, objective location, and the time at which a transaction is received.

Object Relational Map The Object Relational Map in this experiment consists of two elements; World Map and Agent Map.

World Map This mapping was used to synthesize knowledge of the world through a persistently updated shared state of an agent's surrounding physical environment and the agent itself.

```
this.bdbOrm.define("worldModel", {
    type: String,
    size: Object,
    drones: Object,
    grid: Object,
    name: String
});
```

Listing 1. World mapping

Within this mapping, the complete state of a single mission was stored: the size of the area to map, which agents were interacting on the grid (this is an array of stored assetIDs linking to the agent map of the respective agents) and the map as mapped by the agents issued on the mission, see Listing 1.

Agent Map The agent mapping was used to represent the active agents in the BigchainDB network.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

```
this.bdbOrm.define("droneModel", {
    id: String,
    location: Object,
    action: String,
    object_detected: Boolean,
    battery: Number,
    cost: Number,
    type: String
});
```

Listing 2. Drone mapping

As shown in Listing 2, the agent mapping contains seven elements which are related to its status within the physical world. Using these elements, the agent's location was known throughout the network, and it could act based on the observations made by all agents.

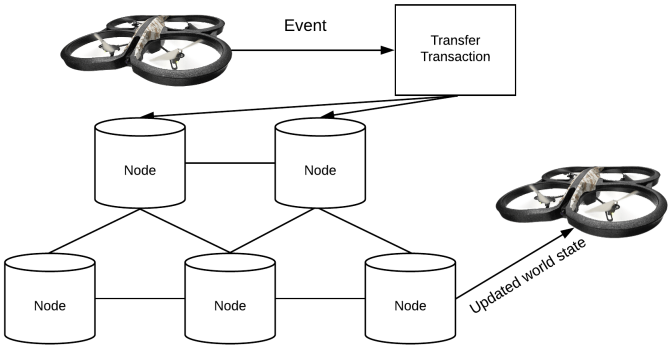


Fig. 2. Flow diagram

When an agent performed an action within the world, *e.g.*, moving left, right, forward, or backward, the network was updated on its complete state. The agent updated the network with its current location, battery life, and the actions it performed (see Figure 2).

A frontend was built using the NodeJS implementations of the BigchainDB driver. This frontend was built to provide insight into the workings of the application and showcase its functionality. The frontend also demonstrates the possibility to supervise a swarm of agents and instantly synthesize and extract information from the system, which resolves the issues in Table 1, Challenges 2 and 4.

3.3. Hardware—The SR system used in this experiment is heterogeneous, composed of the following types of agents:

- Air drones – Parrot AR Drone 2.0;
- Ground drones – Parrot Jumping Night;
- Simulated drones – which ran a physical full DLT node, but were not coupled with moving hardware.

All agents ran full DLT nodes and propagated information through DLT over a decentralized network, but only a number of them were coupled with physical drones. A complete overview of the hardware is detailed below.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

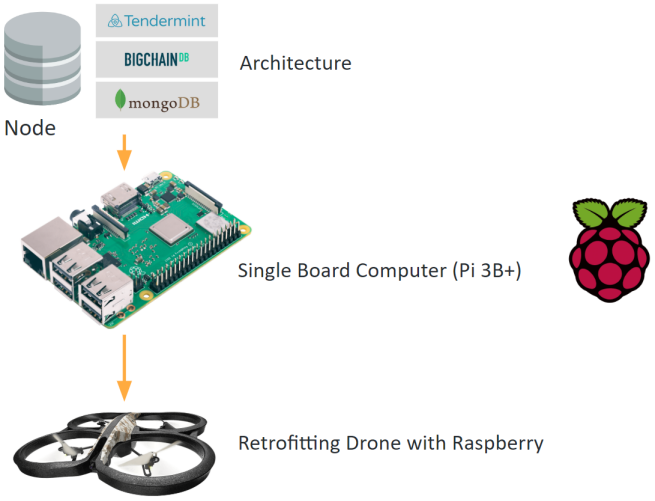


Fig. 3. Hardware used for the creation of physical agents

The architecture was designed and implemented for drones coupled with single board computers (Parrot AR Drones 2.0, retrofitted with Raspberry Pi 3B), as shown in Figure 3. These single board computers were chosen due to their availability, relatively low cost, and great simplicity. Due to flight restrictions and a lack of a space large enough to fly, the solution was deployed on nine single board computers to simulate an active multi-node decentralized network and a ground drone (Parrot Jumping Night Drone) was used as a physical agent.

A clusterboard composed of seven SOPINE A64 and two ROCK64 single board computers were used (see Figure 4). Additional simulations to test the system on more capable hardware were run on a Microsoft Azure server; specifications are as follows: Standard F72s_v2 (72 vCPUs, 144 GB memory) for the BighainDB node and Standard F16s_v2 (16 vCPUs, 32 GB memory) for the agent application.

Two experimental configurations were used to produce results for this experiment. One configuration consisting of nine single board computers was used to model "search and rescue" situations (the real configuration), while the other configuration consisting of a Microsoft Azure server (the simulated configuration) was used to test the limits of the system.

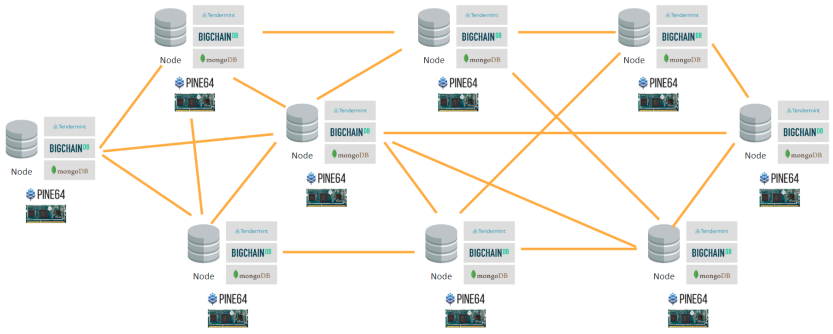


Fig. 4. Network structure to enable inter-agent communications

While the system was designed to accommodate a heterogeneous swarm, the DLT layer is consistent across all agent types. Agents used their sensors to acquire data and subsequently propagated data to BigchainDB. Each agent's BigchainDB instance was then synchronized over

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

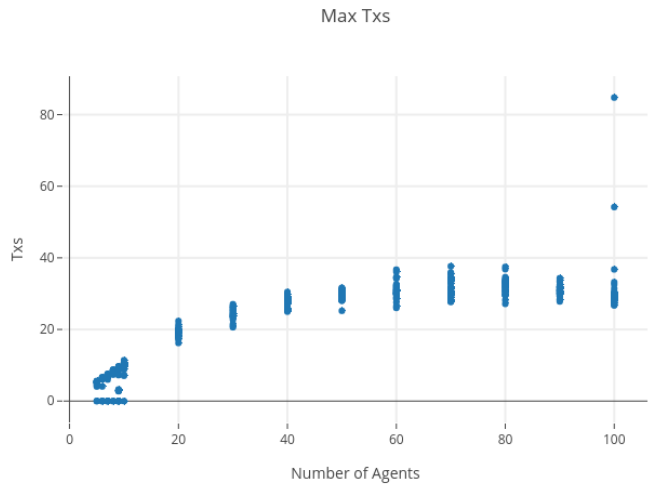


Fig. 5. This figure shows the maximum rate at which transactions occurred during the experiment. These results were obtained from the simulated configuration.

the network. For an illustrated version see Figure 4.

4. Results

The experiment aimed to investigate the implementation of DLT in SR. The environment was implemented using BigchainDB and NodeJS, as described in the experiment section. The code was then deployed and executed on the single board computers. The ground drone was used as a search and rescue drone. Upon activation, the agents entered the search state. After successfully detecting a point of interest, they entered the rescue state. The simulated drones then flocked to the location of the objective. The source code can be found on GitHub.

The experiment ran on two different configurations, one resembling the configuration at the hackathon (real configuration), and the other on an Azure server (simulated configuration), with a total of 404 simulations. The results can be found in Figure 5 and Figure 6. All of the data produced by these tests can be found on plot.ly.[20]

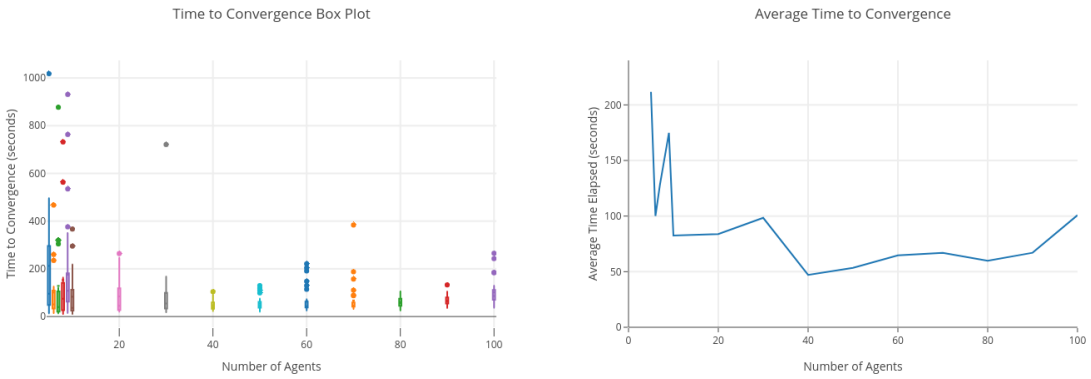
The results shown in Figure 5 show the maximum rate at which transactions occurred during the experiment. An increase in maximum tx/s occurs as the number of agents increases. In the figure, outliers in tx/s are visible, the extreme outliers only occur in tests with 100 agents.

Figure 6 displays the time to convergence for the system. Results in Figure 6 (a) do not show a clear correlation between the number of agents and time to convergence. However, it is clear that the results in the lower regions have a high variance, number of outliers produced decreases with the number of agents up to a threshold of around 20 agents. When this threshold is crossed, the number of agents has little influence on the number of outliers produced by the tests.

The results in Figure 6 (b) show the average time to convergence decreases as the number of agents increases, until the number of 40 agents is reached. After reaching 40 agents, the time to convergence increases as the number of agents increase.

Figure 7 shows a comparison between the time the agents took to converge within the simulated configuration and the real configuration. The simulated configuration has less variance and lower convergence times.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176



(a) Boxplot of time to convergence

(b) Average time to convergence

Fig. 6. These figures show the time the agents took to converge to the objective from the moment they were initialized. These results were obtained from the simulated configuration as described in the hardware section on page 48.

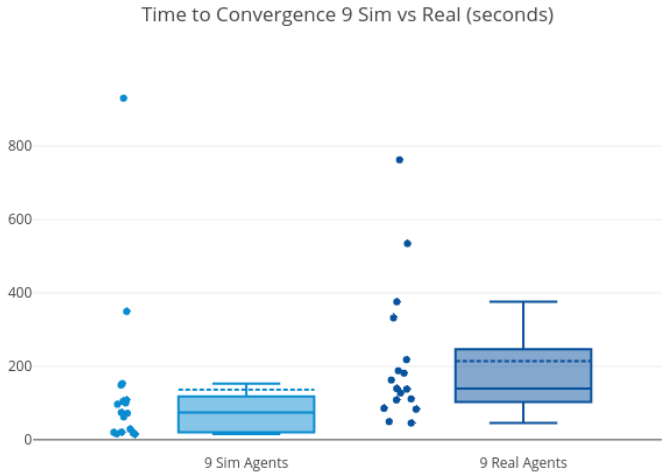


Fig. 7. This figure shows a comparison between the time the agents took to converge on the simulated configuration and the real configuration.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

5. Discussion

Table 2. A series of challenges that impede the widespread adoption of SR for real-world applications, and their resolution through DLT within this experiment.

#	Challenge	Resolution	Source
1	Hybrid system	The SR system was implemented without assigning a leader among the agents. The tasks were performed autonomously, and consensus was reached without issues.	Barca (2013) p. 353[4]
2	Self-organizing SR system	The resolution was demonstrated by showcasing real-time information propagated by all active agents in a frontend interface. The interface displayed a map showcasing all agents, their position relative to each other and their current status (battery level, current task, point of interest found).	Barca (2013) p. 353[4]
3	Scalability and robustness	The experiment was conducted with multiple active agents, sending transactions at a certain interval. The experiment has shown the network to operate at around 20 tx/s (mean equal to 21.5886txs), with peak performance above 50 txs. Scalability tests of BigchainDB indicate that it should be possible to process at least 298 transactions per second.[17] This discrepancy can be attributed to the use of the ORM driver.	Barca (2013) p. 353[4]
4	Accountability, transparency, and trust	Each of the actions taken by the agents in the experiment could be traced back to a transaction on BigchainDB. All actions taken by agents were announced on BigchainDB.	Calvaresi *et al.* (2018) p. 8,[7] Castelló Ferrer (2016) p. 4[5]
5	Implementation of SR and DLT	This paper implemented an SR system using DLT on physically embodied fully autonomous agents communicating through a fully decentralized network.	Calvaresi *et al.* (2018) p. 10[7]
6	Sustainable SR system.	NA	Barca (2013) p. 353[4]

Table 2 shows the challenges mentioned in the relevant research section and adds the suggested resolutions suggested from the results of this experiment. The successful deployment demonstrates that it is possible to run full BigchainDB nodes on relatively simple hardware.

The experiment was initially executed during the Blockchaingers hackathon, which lasted two days; demonstrating that implementation of DLT in SR is possible within a short timeframe with limited resources. The relative ease with which DLT capabilities can be added to devices could be employed for other use cases, most notably in the Internet of Things domain. This environment was used to simulate the search and rescue mission.

The use of BigchainDB increased scalability compared to Ethereum, at around 20 tx/s. This peak throughput is not as high as the expected throughput of BigchainDB. This discrepancy is partially due to using the ORM driver for BigchainDB, which caused the system to experience a significantly lower transaction throughput than the existing benchmarks. The ORM increases the transaction payload size; it also performs transfer transactions on assets with a long chain, which requires it to retrieve the entire asset and its history for each transaction. In the experiment a create transaction is performed for each agent and objective creation. In the following transactions a transfer is performed from the old world state to the new one. Increasing the length of the asset chains decreases the transaction speed of the application as time increases, as each subsequent

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

transfer requires more data than the previous one. These findings have been confirmed by Pregelj.[21]

When looking at the average time to convergence in Figure 6 (b), a high variance is visible for tests with 0 to 20 agents. This threshold is most likely the result of the task that is being simulated in the experiment, the grid upon which it is performed does not change size. After a certain threshold, the effect of adding more agents is limited.

The results in Figure 6 (b) show the optimal number of agents to complete the task is 40. When there are more than 40 agents, the system is most likely limited by the speed at which the furthest agents move to the objective. A number of agents lower than 40 will likely take more time to find the objective.

There are dissimilarities illustrated in Figure 7. However, these differences are to be expected as the simulated configuration is running on a machine capable of handling a higher workload. Furthermore, the BigchainDB nodes and agent scripts run on separate instances in the simulated configuration, as opposed to running on the same instances in the real configuration.

The frontend allowed for observers to have an overview of the SR system, which enabled abstract supervision. In the interface, the current actions of agents were displayed, this led to increased accountability and transparency. A screenshot of the interface can be seen in Figure 1.

6. Conclusions and Future Work

SR faces many challenges that impede widespread adoption of SR for real-world applications. Within this paper, motivations for combining SR and DLT have been highlighted. A number of challenges were identified from the literature mentioned in the references. From research, the challenges, and current resolutions were discussed (see Table 1).

The main challenge the experiment aimed to investigate was the implementation of DLT for SR. The individual challenges and their resolutions can be found in Table 2. The contents of the table describe the resolution of the challenges mentioned in Table 1.

The experiment was conducted, simulating a search and rescue mission with a heterogeneous swarm. The experiment was successful (*i.e.*, the agents updated their state to other agents and executed their tasks), thus showcasing the feasibility of implementation of DLT for SR.

The experiment conducted in this paper demonstrated that the drawbacks of a fully centralized and decentralized system could be mitigated through DLT. Without the vulnerability of a single agent controlling all the agents, consensus was reached. Through the use of a frontend interface, the synthesized knowledge could be extracted by the user and supervising the system on an abstract level, first-time-right is improved. The use of BigchainDB eased the third challenge, as it has increased scalability. The fourth challenge was resolved by logging all actions taken by agents in the DLT. A solution for the sixth challenge could be a self-sustaining ecosystem, however, this possible solution has not been researched due to time constraints; during research, no limitations for this possible solution were encountered. Subsequent research could focus on physical implementation and development of a self-sustaining ecosystem.

Future research could also focus on applying this combination to more difficult tasks. The method used by agents to determine their next action did not allow for very complicated tasks. Presenting the system with complex tasks in the presence of Byzantine agents would allow for more meaningful results to be produced. Additionally, for real-world applications the

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

environment perpetually changes. Following this, it is suggested to explore the implementation of artificial intelligence methods in future research. A distributed approach to machine learning like Federated Learning would enable this, while keeping communications sparse.[22] The continuously updated model is essential for agents operating in a perpetually changing environment. Castelló Ferrer *et al*. "RoboChain: A Secure Data-Sharing Framework for Human-Robot Interaction" (2018) describes a Blockchain robotics platform implementing a Federated Learning solution, RoboChain.[23] Instead of sharing data across multiple parties, hubs aggregate data from their local sources, and algorithms are exchanged. These algorithms are then applied to the aggregated data, and the results are returned. These features are enabled through the use of MIT's OPen ALgorithm (OPAL) platform.[23] A solution built for the system described in this paper could be implemented as described in Castelló Ferrer *et al*. (2018).[23]

While the environment constructed for this experiment did not make use of multiple ledgers, it is possible to connect multiple Tendermint based DLT platforms through Cosmos.[24] Cosmos can connect these platforms because Tendermint offers transaction finality, meaning that when a transaction has been propagated, it is not possible to change it retroactively.[12] This inter-platform connection might be desirable if the scalability impedes proper functionality within the network, or it could be applied when agents need to communicate across different swarms.

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 751615 and No. 701236.

Notes and References

[1] No Author. "Rolls-Royce and SWARM Robots." (2018) (accessed 9 March 2019) https://wyss.harvard.edu/media-post/rolls-royce-and-swarm-robots/.

[2] Bennett, C. "Swarm Bots Could Shift Ag History." (2018) (accessed 9 March 2019) https://www.agweb.com/article/swarm-bots-could-shift-ag-history-naa-chris-bennett/.

[3] Kang, C.-K. "Marsbee - Swarm of Flapping Wing Flyers for Enhanced Mars Exploration." (2018) (accessed 9 March 2019) https://www.nasa.gov/directorates/spacetech/niac/2018_Phase_I_Phase_II/Marsbee_Swarm_of_Flapping_Wing_Flyers_for_Enhanced_Mars_Exploration/.

[4] Barca, J. C., Sekercioglu, Y. A. "Swarm Robotics Reviewed." *Robotica* **31.3** 345–359 (2013) https://doi.org/10.1017/S026357471200032X.

[5] Castelló Ferrer, E. "The Blockchain: A New Framework For Robotic Swarm Systems." *arXiv* (2016) (accessed 9 March 2019) http://arxiv.org/abs/1608.00695.

[6] Disse, T. "Envisioning (and Building!) a New Way to Program Robots Using Blockchain - Meet Grex.ai." (2018) (accessed 9 March 2019) https://medium.com/kryha/envisioning-and-building-a-new-way-to-program-robots-using-blockchain-meet-grex-ai-c9d94a8609ed.

[7] Calvaresi, D., Dubovitskaya, A., Calbimonte, J. P., Taveter, K., Schumacher, M. "Multi-Agent Systems and Blockchain: Results from a Systematic Literature Review." In Y. Demazeau, B. An, J. Bajo, A. Fernández-Caballero (Eds.), *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection* Springer 110–126 (2018) https://doi.org/10.1007/978-3-319-94580-4_9.

[8] Brakeville, S., Perepa, B. "Blockchain Basics: Introduction to Distributed Ledgers." (2018) (accessed 9 March 2019) https://www.ibm.com/developerworks/cloud/library/cl-blockchain-basics-intro-bluemix-trs/index.html.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176

[9] Strobel, V., Castelló Ferrer, E., Dorigo, M. "Managing Byzantine Robots via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario." In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems 541–549 (2018) http://dl.acm.org/citation.cfm?id=3237383.3237464.

[10] Cameron, A., Payne, M., Prela, B. "Research and Implementation of Multiple Blockchain Byzantine Secure Consensus Protocols for Robot Swarms." (2018) Unpublished research (accessed 9 March 2019) https://courses.csail.mit.edu/6.857/2018/project/Cameron-Payne-Prela-ByzRobSwarm.pdf.

[11] Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System." (2008) (accessed 9 March 2019) https://bitcoin.org/bitcoin.pdf.

[12] Buchman, E. "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains." PhD Thesis, University of Guelph (2016) http://hdl.handle.net/10214/9769.

[13] Wood, G. "Ethereum: A Secure Decentralised Generalised Transaction Ledger." (2017) Ethereum project yellow paper, revision 1e18248, (accessed 9 March 2019) http://ljk.imag.fr/membres/Jean-Guillaume.Dumas/Enseignements/ProjetsCrypto/Ethereum/ethereum-yellowpaper.pdf.

[14] Ethereum "Proof of Stake FAQs." (2018) (accessed 9 March 2019) https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs.

[15] Lundbæk, L.-N., Beutel, D. J., Huth, M., Jackson, S., Kirk, L., Steiner, R. "Proof of Kernel Work: A Democratic Low-Energy Consensus for Distributed Access-Control Protocols." *Royal Society Open Science* **5.8** 180–422 (2018) https://doi.org/10.1098/rsos.180422.

[16] Zhao, H. "Bitcoin and Blockchain Consume an Exorbitant Amount of Energy. These Engineers Are Trying to Change That." *CNBC* (2018) (accessed 9 March 2019) https://www.cnbc.com/2018/02/23/bitcoin-blockchain-consumes-a-lot-of-energy-engineers-changing-that.html.

[17] McConaghy, T. "And We're Off to the Races!" (2018) (accessed 9 March 2019) https://blog.bigchaindb.com/and-were-off-to-the-races-1aff2b66567c.

[18] Etherchain "Etherchain - The Ethereum Blockchain Explorer." (2018) (accessed 4 October 2018) https://www.etherchain.org/.

[19] Pregelj, J. "A CRAB-based ORM for BigchainDB." (2017) (accessed 9 March 2019) https://blog.bigchaindb.com/crab-create-retrieve-append-burn-b9f6d111f460.

[20] The GREX plot.ly can be found at https://plot.ly/~grexai.

[21] Pregelj, J. *BigchainDB* (2018) Private Communication.

[22] Konecný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., Bacon, D. "Federated Learning: Strategies for Improving Communication Efficiency." *arXiv* (2016) (accessed 9 March 2019) http://arxiv.org/abs/1610.05492.

[23] Castelló Ferrer, E., Rudovic, O., Hardjono, T., Pentland, A. "RoboChain: A Secure Data-Sharing Framework for Human-Robot Interaction." *arXiv* (2018) (accessed 9 March 2019) http://arxiv.org/abs/1802.04480.

[24] Kwon, J., Buchman, E. "Cosmos: A Network of Distributed Ledgers." *Cosmos* (2016) Whitepaper (accessed 9 March 2019) https://cosmos.network/whitepaper.

SSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.176