2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS) | 978-1-6654-9172-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/DAPPS55202.2022.00014

Managing Collaborative Tasks within Heterogeneous Robotic Swarms using Swarm Contracts

Sanjaya Mallikarachchi
College of Computing and Digital Media
DePaul University
Chicago, IL, USA
Email: smallika@depaul.edu

Can Dai
College of Computing and Digital Media
DePaul University
Chicago, IL, USA
Email: cdai2@depaul.edu

Oshani Seneviratne
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY, USA
Email: senevo@rpi.edu

Isuru Godage
College of Computing and Digital Media
DePaul University
Chicago, IL, USA
Email: igodage@depaul.edu

***Abstract*—The growing number of applications in Cyber-Physical Systems (CPS) involving different types of robots while maintaining interoperability and trust is an ongoing challenge faced by traditional centralized systems. This paper presents what is, to the best of our knowledge, the first integration of the Robotic Operating System (ROS) with the Ethereum blockchain using physical robots. We implement a specialized smart contract framework called "Swarm Contracts" that rely on blockchain technology in real-world applications for robotic agents with human interaction to perform collaborative tasks while ensuring trust by motivating the agents with incentives using a token economy with a self-governing structure. The use of open-source technologies, including robot hardware platforms such as TurtleBot3, Universal Robot arm, and ROS, enables the ability to connect a wide range of robot types to the framework we propose. Going beyond simulations, we demonstrate the robustness of the proposed system in real-world conditions with actual hardware robots.**

***Index Terms*—Robotic Operating System, Cyber-Physical Systems, Simultaneous Localisation and Mapping, Swarm Robotics. Smart Contracts**

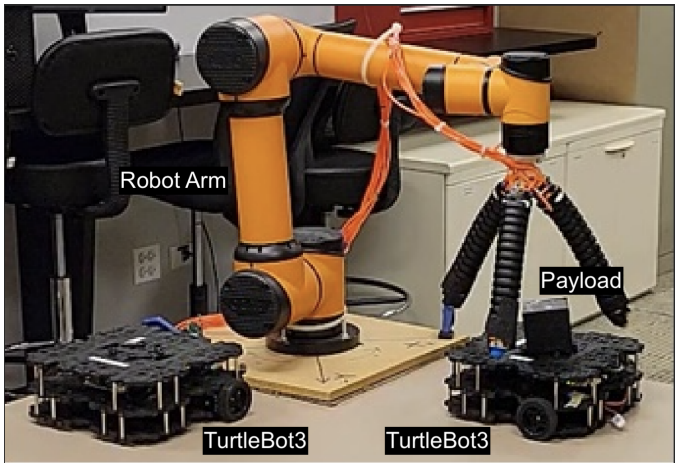

Fig. 1. A robotic arm and two mobile agents performing a collaborative task, moving a payload from one location to another

I. Introduction

In conventional homogeneous swarm robotic applications, it is possible to replace a robot if it malfunctions since the robots utilized are assumed to be interchangeable. However, with the rising complex real-world applications from search and rescue missions [1] to household robotic appliances [2] to industrial robotic applications [3], heterogeneous robots will need to collaborate with the other robots and human agents to accomplish specific tasks assigned to them. In order to achieve these tasks, a scalable, secure, and efficient way of standardized communication between the robots and the human agents have become increasingly essential. Fig. 1 shows an example of a collaborative task involving different types of robots that are moving a payload from one location to another but have mobility restrictions. A sophisticated communication, managing and governing system is required to perform collaborative tasks between heterogeneous robotics agents.

With the vision of designing more secure and trustworthy robotic systems, researchers have been testing out various distributed ledger (blockchain) technologies from private permission-based blockchains like Hyperledger Fabric Blockchain [4], [5] to smart contracts that run on Ethereum Virtual Machine [6]. In contrast to using a centralized system, the use of blockchain technologies in multi-robot systems has proven effective and reliable [7], [8] by nature of the blockchain, ensuring the security and the integrity of knowledge sharing based collaboration between robots.

A. Prior Work

With the rise of secure and practical blockchain technologies like Smart Contracts, researchers in various industries have been searching for ways to implement these technologies to enhance the efficiency and security of their systems. Smirnov et al. [9] propose an approach to utilizing smart contracts in agriculture, and Teslya et al. [10] introduce the use cases of

DOI 10.1109/DAPPS55202.2022.00014

smart contracts for dynamic planning and coalition of robots. Salimi et al. [11] present the concept of using smart contracts to manage industrial robotic fleets in warehouses. Due to its high performance and low requirements for end devices, many industrial applications, such as those mentioned above, utilize the Hyperledger Fabric Blockchain, a private permission-based blockchain.

In addition, Strobel et al. [12] present the concept of developing trust in the system by identifying and excluding byzantine swarm agents. This work has enabled swarm contracts to build trust over time to avoid adversarial agents and increase the efficiency of transactions by eliminating byzantine agents from participating in the system. Kapitonov et al. [13] propose a protocol of autonomous business activities for unmanned aerial vehicles using an integration of ROS and Ethereum blockchain. The authors' suggestion and implementation of this architecture opened ways for innovation in multiple industries like defense and agriculture. Further, Li et al. [14] introduce a blockchain-based collaborative edge knowledge inference framework for edge-assisted multi-robot systems to ensure the trust of knowledge sharing and conduct. Kaewkamnerdpong et al. [15] propose the concept of using swarm robotics intelligence to model nanorobot controls, which highlights another industry where swarm robotic collaboration can be effective. The idea of a framework that is possible to bring together robotic swarm applications and blockchain technology has been introduced by Ferrer [16].

Grey et al. propose and implement a Swarm Contract framework [3], based on the blockchain technology that utilizes a token economy for robotic agents with human interaction to perform collaborative tasks while ensuring the trust by motivating the agents with incentives in completing a given task. Swarm contracts are custom smart contracts for robots to facilitate complete and uncompromising communication and collaboration to achieve tasks securely and in a decentralized manner among mutually distrusting and heterogeneous parties. Further, the Swarm Contracts incorporated a subcontracting framework [3] within the blockchain environment to allow the robotic agents to efficiently and cost-effectively perform complex jobs requiring multiple agents with various capabilities.

The purpose of using Swarm Contracts [17] is to maximize efficiency, minimize the possibility of exploitation, and guarantee the trustworthiness of all parties involved. In addition, the Swarm Contracts were created to solve the limitations of centralized robotic planning applications with a reward system and adjudication. However, the swarm contracts deployed were limited to moving payloads using one type of robot (mobile). In practice, different types of robots perform various and more complex tasks; therefore, a framework that can support various robot types is essential.

Moreover, unlike the simulation environments, implementing these systems in the real world is a challenging task in applications such as autonomous vehicles [18]. By performing real-world experiments, the whole system can be evaluated under situations where it is impossible to test in simulation environments, such as robot malfunctioning, running out of battery power, sensor feedback errors, communication losses and delays, and mechanical and electrical failures, etc.

B. Contributions

This research aims to show a smart contract-based robotic collaboration system's robustness even under challenging real-world situations and evaluate how well the system behaves while ensuring trust using blockchain technology. A decentralized system architecture such as the proposed Swarm Contract framework by Grey et al. [17], in which all the activities are assigned, checked, and reevaluated by the framework, establishes a dynamic execution environment that can maintain the trust between robotic agents while assessing and ensuring the completion of the tasks.

In this paper, we introduce three types of Swarm Contracts for three specific types of tasks:

- **Collaborative Handling Task:** where two mobile robots and a robotic manipulator perform a collaborative object handling task.
- **Navigation Task:** where two mobile robots navigate through an area with obstacles to generate a map.
- **Warehouse Handling Task:** where two mobile robots handle the payloads.

These tasks allow the utilization of different types of robots, such as robotic manipulators (arms) and mobile robots. Beyond the work of Grey et. al. [17], we use prototype robotic agents and models thereof with the Swarm Contract framework. This capability broadens the utility of the swarm contracts framework in applications such as disaster relief, search and rescue missions, and agriculture. Furthermore, compared to centralized trust-free systems that cannot guarantee designated task completion by workers, Swarm Contracts make decentralized applications a viable alternative to centralized command and control applications that are pervasive in multi-agent robotics applications today.

The core contribution of this research is integrating Ethereum Blockchain Smart Contracts technology with ROS [19]. This integration is an improvement over [3] – which was limited to simulations – to allow interfacing of various robot hardware, including industrial robot manipulators and mobile platforms beyond the simulations of the same robots in the Gazebo [20] simulation framework built into ROS. Also, we introduce a pipeline to integrate any robot into the proposed system framework and test the system's robustness in real life when there are instances where robots create troublesome situations such as WiFi connection loss/delay and battery outage.

The remainder of this paper is organized as follows: Sec. II: System Framework gives an overview of Smart Contract framework integration to ROS, robotic agents, and the trust system used in the framework. Sec. III: Evaluation Tasks presents the experiments conducted in detail. Sec. IV: Results illustrate the experimental results and interpretation. Finally, the conclusion and future work are described in Sec. V: Conclusion and Future Directions.

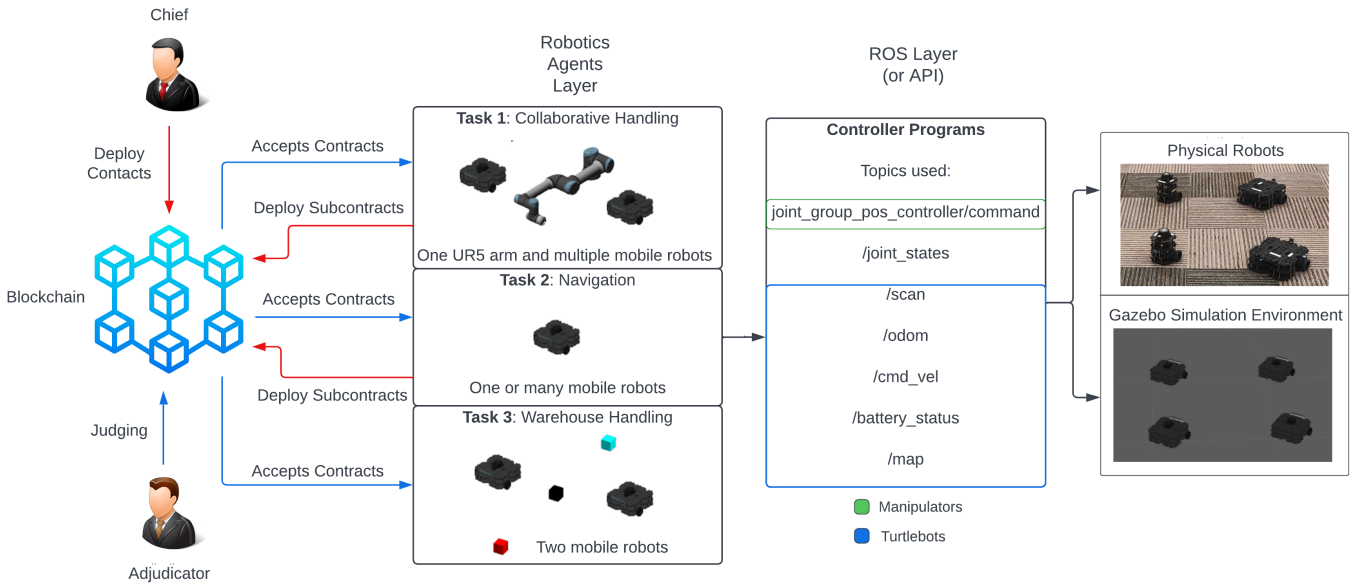

Fig. 2. System Overview: Robotic agents - integration of industrial robots; Control Programs: modules written to control the robots for accepted contract tasks

II. System Framework

This section discusses the robotic agents used in the research, integration between ROS and the Blockchain, and the swarm contracts and trust model. In addition, we discuss the implementation of the control program modules to drive the robots in the simulation environment as well as the actual robots in the field, as shown in the Fig. 2.

A. Robot Operating System (ROS)

Autonomous mobile robots perform complicated tasks that require successful navigation from any point A to point B in static and dynamic environments. ROS facilitates the communication between robotic agents in these navigational settings. ROS is widely used in the robotics community and has strong support from open-source robotics developers. Also, the ease of interfacing with various types of robots such as TurtleBot3 [21] Universal Robot [22] and the algorithm packages availability such as GMapping Simultaneous Localization and Mapping (SLAM) [23] for Light Detection, and Ranging (LiDAR) sensors motivated us to use ROS as the robot communication and control layer in this work. ROS also supports subscriber and publisher protocols similar to Message Queuing Telemetry Transport (MQTT) protocol [24]. ROS nodes are executable files within the ROS package that use the ROS client library to communicate between other nodes in the system. ROS nodes can publish and subscribe to topics to send and receive data. For instance, if a program node needs to receive odometer data, the program should subscribe to the "odom" topic. ROS master coordinates all the nodes within the system. Another significant advantage is the possibility of benefiting from the mature codebase intended for simulations with actual robots without making any changes. It provides a broader space to test the entire framework effectively and efficiently for simulation to physical-world experiments.

B. Robotic Agents

In this research, two types of robotic agents were used, including a robotic arm, i.e., the Universal Robot (UR) arm, and a mobile robot (TurtleBot3). Typical workshops and warehouses employ mobile robots and robotic manipulators for payload handling or other various complex tasks. Although engaged in collaborative tasks between robots, individual robots are independent agents and make independent decisions. Moreover, robots can perform independent tasks by accepting individual work contracts. However, these decisions are context-dependent and differ over time with the knowledge accumulated within the system that feeds the trust model explained in Sec. II-D.

C. ROS and Blockchain Integration

In this work, the ROS integration with the Swarm contract system framework gives us greater flexibility to integrate industrial robotic agents such as Universal Robot (UR) arms [22], and any other supported third-party robots. Furthermore, integrating open-source packages such as SLAM [23] and other algorithms makes the entire framework more practical in various robotic applications, from disaster-rescue missions to agriculture.

For the blockchain ecosystem to test the proposed framework, a locally hosted personal Ethereum blockchain, Ganache [25] is used. The communication between Ganache and the robot control programs was established via web3 (Ganache library for python) [26]. Typical contract deployment takes 1-3 seconds including the compilation of the contracts. In this work, contracts are not required to deploy at a faster rate.

But with a large number of agents, this will be a bottleneck since more contracts are completed simultaneously and new contracts need to be deployed more frequently. One can deploy the contracts for a collaborative robotic task to the blockchain, allowing robotic agents to decide whether they want to participate in collaborative tasks while accepting "collab-contracts." These "collab-contracts" make optimal and fair decisions in unpredictable (i.e., mechanical troubles, sensor malfunctions, etc.) real-world scenarios. One example is the action to be performed when one mobile robot is running out of battery. In such a case, the control program written for a particular mobile robot can detect the battery level since the power levels are continuously monitored by subscribing to the battery status topic. The control module then withdraws from the accepted contract and allows the contract to be accepted and carried by another worker. The workers, however, are penalized for withdrawing from the contract for any reason.

The earnings for mobile robots are calculated based on the amount of work performed (i.e., energy spent) in each task. For instance, robotic manipulators utilize the energy spent on movements via robot joint angular displacement, whereas mobile robots compute the energy spent via distance traveled. Upon successful task completion, robotic agents earn Ether tokens (denoted by Ξ).

The framework with new contracts proposed in this work can be implemented in the real world without a hassle by introducing task-specific smart contracts such as Collaborative Task Swarm Contracts using ROS and the Ethereum blockchain integration. The physical experiments using the proposed framework are further explained in Sec. III.

D. Swarm Contracts and Trust Model

Swarm contracts [17], tailored Smart Contracts [6] for swarm robots. They were introduced to expedite secure, stable communication and collaboration between robots to achieve tasks assigned using Smart Contracts technology. The swarm contracts are designed to ensure the trustworthiness of all parties involved in the system. Grey et al. [17] introduced the concept of having a "board" of mutually trustworthy parties that include the *Chief*, which deploys contracts, and *Adjudicators*, that are assigned to judge the completion of a given task with public accountability. Fig. 2 depicts the various agent types including both robotics and the no-robotics agents. All the robotic agents and the field are shown in Fig. 3. In the warehouse area (where the payload boxes are located) mobile robots who handles boxes are defined as *Cowboy* (TB1) and *Driver* (TB2) while in the collaborative area the three robots (two mobile robots and one robots arm) who do collaborative handling are defined as *Handler1* (TB3), *Handler2* (TB4) and *UR* (Manipulator1). Another two mobile robots, *Navigator1* (TB5) and *Navigator2* (TB6) work in the mapping area.

In a real-world scenario, the employees/workers go through selection, interview, and offer stages before starting work for a company. However, in a trust-free decentralized system, the number of new employees, in this case, the robotic agents that must guarantee to perform the work assigned, is constantly changing and in large numbers. To ensure the guarantee to perform work in the system, Grey et al. [17] proposed a "collateral" method. The collateral is collected from the robotic agent, which accepts the swarm contract with the promise to complete the job. In addition, the robotic agents are rewarded after being judged by the Adjudicator on the successful completion of the given task. The reward mechanism, along with the collateral method, promotes all participating agents to ensure trust and motivation in the system.

The architecture of the smart contracts and the agent definitions are the same as in Grey et al. [3] which the *Chief*s deploy the contracts to the blockchain with *Adjudicators*, and robotic agents can accept the contracts. Once the agents state that the task is completed, *Adjudicators* will judge the task. By comparing with the ground truth, the completion of tasks by the workers and the judgments of the *Adjudicators* are evaluated. Over time, the adversarial agents will be neglected from the system (less likely to get a chance to accept a contract) based on the trust model. This behavior with comparisons in agents' earnings can be observed in Sec. IV below. In comparison to the real world, we can consider these adversarial workers as fake nodes who pretend to be robotic workers who try to accept the contracts and gain profits without performing the accepted contract tasks.

III. Evaluation Tasks

As mentioned in the Sec. I-B, under the introduction, real-world tasks have been performed to test the system under troublesome real-world situations such as WiFi connection loss/delay and battery outage. Therefore we performed around 10% of tests with actual robots and the rest in the simulation environment. Limited actual tests are to prevent the hardware from being overused. For the simulation environment, Gazebo simulation software was used to utilize the Open Dynamics Engine (ODE) [27] with the support of ROS. The experiments are performed for three robotics tasks shown in the TABLE I.

For the tests performed in this paper, the following assumptions were made:

- No sensor errors from odometer and LiDAR: Valid in Gazebo simulation environment. The LiDAR and odometer sensors [28] used in the Turtlebot3 have an accuracy error of 3%. It is acceptable for completing the local navigation tasks in hardware evaluations related to **Task 2**.
- No WiFi communication disruptions or delays: Valid in Gazebo simulation framework. Since the real-world experiments conducted are conducted in a closed space research lab with a stable WiFi connection, the assumption remains valid in testing in the real world.
- There is no wheel slipping: Valid in Gazebo simulation environment. In addition, it is valid for the tests with the actual robots since the ground surfaces the robots were driven on had sufficient friction with the robot wheels.
- LiDAR sensor has infinite range: Valid in the Gazebo simulation framework. The LiDAR sensor [28] used in

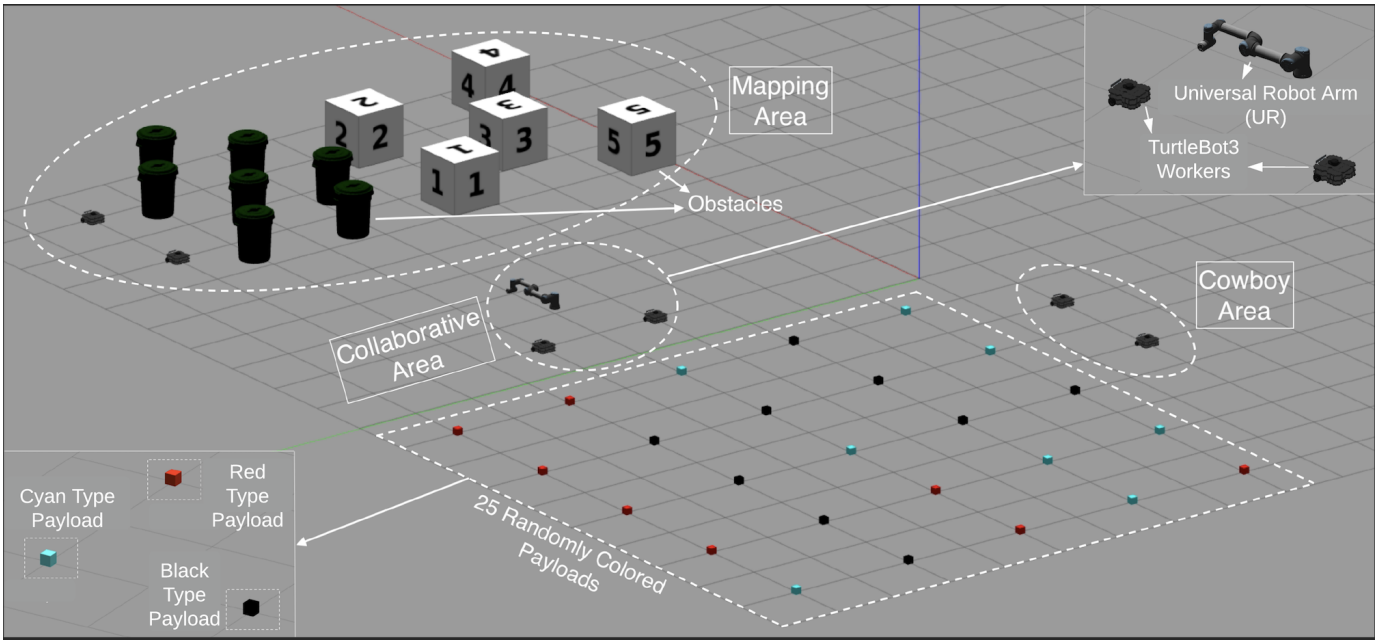

Fig. 3. Simulation Environment a) Navigation area where mobile robots avoid the obstacles and generate the territory map. b) Collaborative handling area: one UR arm and two TurtleBots c) Warehouse handling: two TurtleBots *Cowboy* and the *Driver*

the Turtlebot3 has a maximum range of 8 m, which is more than the dimensions of the hardware testing area (5 m x 5 m). Thus, the sensor can detect any object or wall without any difficulties, and the assumption remains valid in testing in hardware.

These assumptions were considered otherwise it brings complex scenarios that cannot be handled within the proposed system. None of these robots are robust for dynamic environmental conditions, only suited for controlled environments which align with the capabilities of the robots.

TABLE I
TYPES OF TASKS

Task	Description	Robot arms	Mobile Robots
1	Collaborative handling	1	2
2	Navigation	0	1 or n
3	Warehouse Handling	0	2

A. Task Descriptions

Task 1 is a Collaborative handling task in which three robotic workers (one robot arm and two mobile robots) are needed. The robots will start working after the swarm contract is accepted by all three members. In the contract, the tasks for each robotic agent are specified. For example, the picking position and the placing position for the robot arm task are stored in the swarm contract. The destinations of the mobile robots are also specified in the Collaborative handling task contract.

Task 2 is the navigation task, which is performed by a single or multiple mobile robot. In the swarm contract, the area to navigate is specified by two points. One or many mobile robots can participate in the navigation task. However, only the robot that accepts the contract can deploy subcontracts by dividing the area to be covered. We use only two robots for this task in the simulations to keep the real-time-factor parameter close to 1.0 in the Gazebo simulation environment. This is a limitation within the simulation environment when it uses a higher processing power of the ROS master node to perform a task. When the number of robots increases, the real-time-factor parameter tends to go below 1.0, making the system simulation slower. If one wants to run more robots to perform these navigation tasks, it is recommended to use multiple simulation nodes which runs on multiple computers.

Task 3 is the Warehouse handling task. In this task, two mobile robots *Cowboy* and *Driver* are available. *Cowboy* collects the specified boxes from the *Cowboy* area and delivers them to an intermediate staging location specified in the contract. *Driver* moves the objects from the staging area to delivery locations specified in the smart contracts. Table II shows the agent composition of each trial. Adversarial workers, who accept the contracts and do nothing. There are three types of *Adjudicators* Fair, Owner-biased, and Worker-biased within the society. Fair *Adjudicators* judge the completion of tasks without any bias while Owner-biased and Worker-biased *Adjudicators* judge in favor of the Owner and Worker respectively. Trial 1 and Trial 4 have the same agent composition for both workers and the adjudicators. In between the two trials, the Trial 2 and the Trial 3 composition was arranged to feed the trust model with randomness. We cannot expect the same composition throughout every trial in the real world. After performing all of these trials, we can compare Trial 1, which is before accumulating the knowledge for the trust model, and Trial 4,

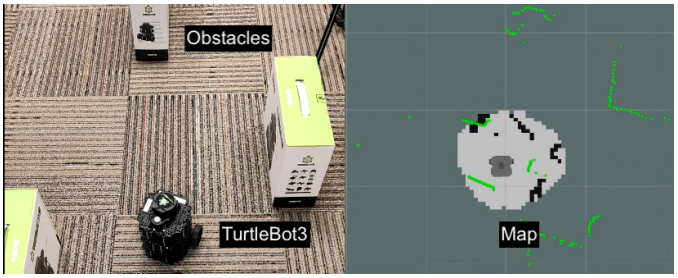

Fig. 4. Robotic agent performing a navigation task using SLAM. Left image showing robotic agent navigating through environment with obstacles. Right image showing the mapped area by robotic agent's LiDAR sensor.

which is after accumulating the knowledge for the trust model.

TABLE II
TRIAL COMPOSITION

Trial	Adversarial Workers	Fair *Adjudicators*	Owner-biased *Adjudicators*	Worker-biased *Adjudicators*
1	6	4	1	1
2	5	3	2	2
3	5	3	0	0
4	6	4	1	1

There are seven fair workers in all trials. The number of robotic workers appearing in the simulation environment or real-world tests is the same throughout trials. It is possible to add more fair workers, but we had to limit it because of the real-time-factor parameter in the Gazebo simulation, as mentioned previously. All the adversarial workers are virtual.

B. Simulated Experiments

We used the Gazebo simulation environment to perform longer-range and considerably time-consuming tasks such as **Task 1 and 3**. Fig. 3 shows the various simulation environment setup for the tasks discussed, navigation area, collaborative handling area, and warehouse handling area. Therein, to accomplish the objectives of **Task 1**, we used the UR robot arm as the manipulator and TurtleBot3 to move objects, respectively. In addition, we utilized SLAM [23] with the help of the in-built LiDAR sensors to navigate the robots in the evaluation space (see Fig. 4). The SLAM algorithm uses the Gmapping algorithm [23]. We rely on the odometer sensor data for the local navigation of mobile robots to the specified locations. Once a contract is accepted, the navigator knows the two points(starting and ending point). So it can navigate through the mapping area and collect the corresponding map data. Here the map data is stored in an array that is directly read by subscribing to the "/map" topic of each navigator robot node. Once the task is completed, they send the map data array to the contract and notifies that they have completed the task.

In addition to the evaluations in simulation environments, we implemented a selected set of tasks in corresponding real robotic hardware, i.e., Turtlebot3, and UR arm, to demonstrate the feasibility of the proposed swarm contracts framework under practical constraints such as running out of battery situations.

C. Real World Experiments

We interface and control the hardware TurtleBot3 mobile robots and the UR manipulator robot via the ROS communication layer. Here, we leverage the seamless interoperability of the ROS-powered Gazebo simulation environment, which facilitated the hardware testing on **Task 2** through a mere change of the agent IP address. This feature is also desirable for integrating other types of hardware and software robotic agents. Fig. 4 displays a mobile robotic agent performing a navigation task using SLAM. On the left, the robotic agent navigates through the obstacles and maps the position of the environmental obstacles (boxes) using LiDAR sensors. We see the resulting SLAM image created using the sensor data on the right. The black lines show the obstacles LiDAR sensors sense. In addition, it is essential to note that the robotic agent is only able to map the environment at a certain distance between 160 - 8,000 mm, specified by the ROBOTIS e-Manuel [28] for TurtleBot agents. The grey area shown on the right is the area that LiDAR sensors can sense. As the agent navigates through the environment, the grey area and the obstacle positions will update and display the complete representation of the environment.

IV. RESULTS

The results generated from both simulation environment and real-world experiments were taken into account to evaluate the entire system's behavior. The average earnings and the transactions that happened for the experiments Trial 1, Trial 2, Trial 3, and Trial 4 are reported in Table III. Fig. 5 shows how the earnings changed throughout the trials for workers.

Throughout the trial iterations, it can be observed that the earnings of all adversarial workers get reduced over time (see Fig. 5). Adversarial agents either posted losses or failed to profit like their fair peers. The earnings of fair workers sharply increased in all tests and across all trials. These data prove that the system is both self-governing and self-improving through the power of incentives. While the system does suffer initial "growing pains" while establishing trust, it eventually becomes resilient to attacks, even though the contract issuers and acceptors do not recognize or track each other's identities, making them effectively anonymous. The first trial starts with six to seven adversarial to fair-worker ratio. It shows that adversarial workers earned more in this case since the system has not achieved self-governance yet. This result is undesirable, and we performed Trial 2 and Trial 3 with different combinations to accumulate knowledge within the system expecting the system's self-governance.

It is significant to highlight that the fair worker data shown in the second trial (see Table III) exhibits a reduction in the overall earnings of all fair workers (see Fig. 5). This result is due to the increase in the number of owner-biased and worker-biased *Adjudicators* and the reduction in fair *Adjudicators*. There is a 50% increase of biased *Adjudicators* compared to Trial 1 (see Table III). This change results in fewer completed contracts compared to the ground truth. For instance, in the first trial (see Table III) *Cowboy* and *Driver* workers earned

TABLE III
TRIAL RESULTS

Task	Worker Type	Trial 1		Trial 2		Trial 3		Trial 4	
		Avg. profit	Trans.	Avg. profit	Trans.	Avg. profit	Trans.	Avg. profit	Trans.
	Adversarial	27.1	73	5.25	39	0.58	11	0	0
Warehouse handling	TB1 (*Cowboy*)	12.5	115	9.27	66	16.44	138	24	122
	TB2 (*Driver*)	14	134	8.2	94	19.83	156	21.08	117
Navigation	TB5 (Navigator 1)	5.02	47	4.2	41	8.1	80	1.8	84
	TB6 (Navigator 2)	4	39	3.58	33	7.8	71	1.64	91
Collaborative handling	TB3 (Handler 1)	7.66	87	5.26	47	12.07	119	16.5	112
	TB34 (Handler 2)	7.66	79	4.94	53	14.61	99	17.9	105
	UR (Manipulator 1)	15.2	102	7.73	52	21.39	144	26	127

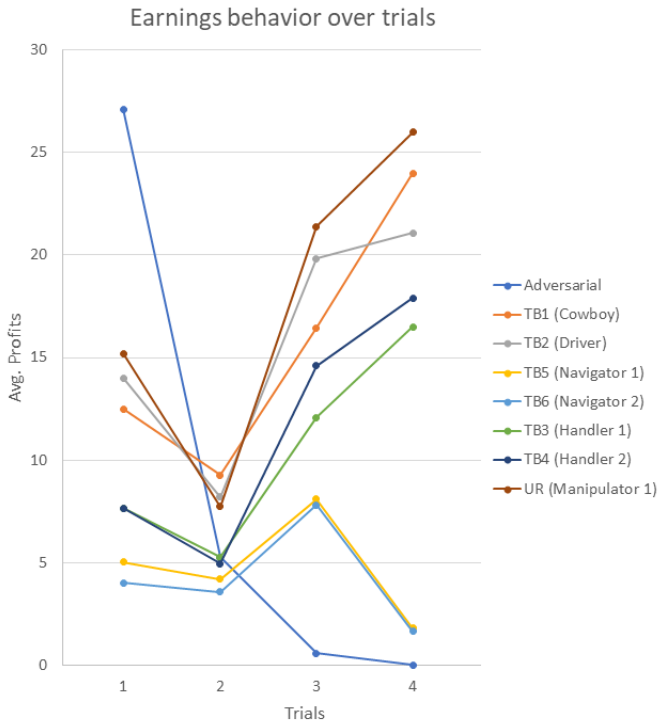

Fig. 5. Earnings of the workers over trials.

12.5Ξ and 14Ξ; however, the earnings in Trial 2 (see Table III) got reduced to 9.27Ξ and 8.2Ξ respectively. It gives us an idea of how the system behaves with the non-availability of few fair workers. In the third trial, the adversarial worker and the fair adjudicator amounts kept the same and observed higher earnings for all workers. That shows more completions of the contracts since more correctly judged contracts are compared to the ground truth. In addition, the system is gaining an equilibrium and increasingly neglecting all untrusted parties.

Trial 3 (see Table III) shows an increase in all earnings for fair workers and a significant decrease for adversarial workers. We can assume a scenario where adversarial judges have been engaged with another system where they can no longer judge the completion of task contracts. This behavior is credited to more completion of contracts due to reduced unbiased *Adjudicators*, and now the system is learned from the previous iterations to neglect adversarial attacks. The earnings exhibit this behavior clearly in Trial 3,Fig. 5.

Trial 4 earnings, shown in Table III, consists of the physical world robot testing. Trial 4 also consists of the same trial composition as Trial 1, returning to the same society again to compare the society without the accumulated knowledge (Trial 1) and society with the accumulated knowledge (Trial 4). It is essential to highlight that the adversarial workers could not earn any token even with the same agent composition. The self-governance trust feature neglected all untrustworthy adversarial workers. After three trials of trust training, the system behaves like an ideal system without adversarial workers where they ended up earning 0Ξ during Trial 4. While this shows the system's robustness and capability in self-handling adversarial agents organically, it is likely a theoretical result as no changes in the economy participants. Nevertheless, in a dynamic ecosystem where the market participants change, there can be exploitation with newer adversarial agents. To handle this exploitation issue, additional trust metrics can be used, such as rating agents' behavior with a proper mechanism.

The navigation task is performed multiple times to experience out-of-battery situations. In most cases, the mobile robots were granted to function until their battery degraded and charged again by a small amount before putting into the experiment. We iterated the same process without charging the mobile robots full charge to experience more frequent out-of-battery situations. As a result, the workers in the navigation task withdrew from their accepted smart contracts more frequently, resulting in lower earnings for them, as shown in Fig. 5. Most of the contracts deployed for navigation tasks were left incomplete. The exact withdrawal mechanism can be used for sensor malfunctioning and communication loss scenarios as well.

V. Conclusions and Future Directions

This work introduced the integration of ROS and Ethereum Blockchain Smart Contracts technology and highlighted the system's robustness in real-world experiments. In addition, the pipeline to integrate any robot into the proposed system framework introduced makes it a reliable option to be used in warehouses or work environments in which heterogeneous robotic swarm technologies are utilized. Tests and results

proved that the proposed system framework can be used for real-world robotic operations, even for complex collaboration tasks requiring cooperation between multiple robotic agents. The scalability of the system can be increased further using multiple ROS master nodes using more remote computers. But still, the system has a bottleneck due to a slower contract deployment rate and less number of transactions per second (TPS) on the Ethereum blockchain. The Ethereum 2.0 upgrade [29] is promising to provide 100,000 TPS which will eventually be capable to build more scalable applications on it. The objective of achieving the system equilibrium for the entire system with the real-world experiments was successful with the obtained results. The system was able to create trust within the system framework that learned from the behavior of the robotic agents.

In addition, adding more decision-making ability to the system will lead to more successful governance by the agents within their agent society. One powerful feature that will be possible to equip is reinforcement learning on agents to make more successful decisions on more complex adversarial attacks. The adversarial behavior is only included when agents compete to accept and judge the contracts. In the physical world where the robotic tasks are performed, adversarial behaviors can be expected during the execution of specific tasks. For example, a robot can always do the opposite of what it is asked to do. In such scenarios, reinforcement learning will be utilized to block destructive behaviors with the swarm contracts being used to reward good behavior and penalize bad behavior. This reinforcement learning feature is part of ongoing work and will be published in the future.

ACKNOWLEDGEMENT

This work is supported in part by the National Science Foundation (NSF) Grants IIS–1718755, IIS–2008797, CMMI–2048142, and CMMI–2132994.

REFERENCES

[1] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *Ieee Access*, vol. 8, pp. 191 617–191 643, 2020.

[2] P. Fiorini and E. Prassler, "Cleaning and household robots: A technology survey," *Autonomous robots*, vol. 9, no. 3, pp. 227–235, 2000.

[3] J. Grey, I. Godage, and O. Seneviratne, "Blockchain-Based Mechanism for Robotic Cooperation Through Incentives: Prototype Application in Warehouse Automation," in *Proceedings of the 2021 IEEE Blockchain Conference*. IEEE, 2021.

[4] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," *CoRR*, vol. abs/1801.10228, 2018. [Online]. Available: http://arxiv.org/abs/1801.10228

[5] S. Dalla Palma, R. Pareschi, and F. Zappone, "What is your distributed (hyper) ledger?" in *2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 2021, pp. 27–33.

[6] V. Buterin, "Ethereum white paper: A next generation smart contract & decentralized application platform," 2013. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[7] J. Li, J. Wu, J. Li, A. K. Bashir, M. J. Piran, and A. Anjum, "Blockchain-based trust edge knowledge inference of multi-robot systems for collaborative tasks," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 94–100, 2021.

[8] T. T. Nguyen, A. Hatua, and A. H. Sung, "Blockchain approach to solve collective decision making problems for swarm robotics," in *International Congress on Blockchain and Applications*. Springer, 2019, pp. 118–125.

[9] A. Smirnov and N. Teslya, "Robot coalition coordination in precision agriculture by smart contracts in blockchain," in *Agriculture Digitalization and Organic Production*, A. Ronzhin, K. Berns, and A. Kostyaev, Eds. Springer Singapore, 2022, pp. 271–283.

[10] N. Teslya and S. Potryasaev, "Execution plan control in dynamic coalition of robots with smart contracts and blockchain," *Information*, vol. 11, p. 28, 01 2020.

[11] S. Salimi, J. P. Queralta, and T. Westerlund, "Towards managing industrial robot fleets with hyperledger fabric blockchain and ros 2," 2022. [Online]. Available: https://arxiv.org/abs/2203.03426

[12] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 541–549.

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

[14] J. Li, J. Wu, J. Li, A. K. Bashir, M. J. Piran, and A. Anjum, "Blockchain-based trust edge knowledge inference of multi-robot systems for collaborative tasks," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 94–100, 2021.

[15] B. Kaewkamnerdpong and P. J. Bentley, "Modelling nanorobot control using swarm intelligence: A pilot study," in *Innovations in Swarm Intelligence*, ser. Studies in Computational Intelligence, C. P. Lim, L. C. Jain, and S. Dehuri, Eds. Springer, 2009, vol. 248, pp. 175–214. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04225-6_10

[16] E. C. Ferrer, "The blockchain: a new framework for robotic swarm systems," in *Proceedings of the future technologies conference*. Springer, 2018, pp. 1037–1058.

[17] J. Grey, I. Godage, and O. Seneviratne, "Swarm Contracts: Smart Contracts in Robotic Swarms with Varying Agent Behavior," in *Proceedings of the 2020 IEEE Blockchain Conference*. IEEE, 2020.

[18] S. Jain, N. J. Ahuja, P. Srikanth, K. V. Bhadane, B. Nagaiah, A. Kumar, and C. Konstantinou, "Blockchain and autonomous vehicles: Recent advances and future directions," *IEEE Access*, 2021.

[19] ROS Developers. Robotic Operating System. [Online]. Available: https://www.ros.org

[20] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[21] turtlebot3 Developers. turtlebot3. [Online]. Available: https://emanual.robotis.com/

[22] Universal Robots GmbH. universal-robots. [Online]. Available: https://www.universal-robots.com/

[23] X. Zhang, J. Lai, D. Xu, H. Li, and M. Fu, "2d lidar-based slam and path planning for indoor rescue using mobile robots," *Journal of Advanced Transportation*, vol. 2020, 2020.

[24] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s — a publish/subscribe protocol for wireless sensor networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, 2008, pp. 791–798.

[25] ConsenSys Software Inc. 2021. Ganache – One Click Blockchain. [Online]. Available: https://www.trufflesuite.com/ganache

[26] P. Merriam and J. Carver, "Web3. py," 2018.

[27] Russell L. Smith. Open Dynamics Engine. [Online]. Available: https://www.ode.org/

[28] (2022) Robotis e-manuel @ONLINE. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_02/

[29] S. E. Foundation. Ethereum 2.0 specifications. [Online]. Available: https://github.com/ethereum/eth2.0-specs