ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/325136332

A Decentralised Approach to Task Allocation Using Blockchain

Chapter · May 2018

DOI: 10.1007/978-3-319-91899-0_5

CITATIONS
8

READS
2,128

4 authors, including:

Regio Michelin
UNSW Sydney
36 PUBLICATIONS **718** CITATIONS
SEE PROFILE

Avelino F. Zorzo
Pontifícia Universidade Católica do Rio Grande do Sul
138 PUBLICATIONS **1,212** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

CORRECT View project

Detecting Encrypted Attacks View project

All content following this page was uploaded by Regio Michelin on 08 March 2019.

The user has requested enhancement of the downloaded file.

A Decentralised Approach to Task Allocation Using Blockchain

Tulio L. Basegio*, Regio A. Michelin, Avelino F. Zorzo, and Rafael H. Bordini

Postgraduate Programme in Computer Science, School of Informatics (FACIN)
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, RS - Brazil
{tulio.basegio,regio.michelin}@acad.pucrs.br
{avelino.zorzo,rafael.bordini}@pucrs.br

Abstract. One of the challenges in developing multi-robot systems is the design of appropriate coordination strategies in such a way that robots perform their operations efficiently. In particular, efficient coordination requires judicious task allocation. Without appropriate task allocation, the use of multi-robot systems in complex scenarios becomes limited or even unfeasible. Real-world scenarios usually require the use of heterogeneous robots and task fulfillment with different structures, constraints, and degrees of complexity. In such scenarios, decentralised solutions seem to be appropriate for task allocation, since centralised solutions represent a single point of failure for the system. During the allocation process, in decentralised approaches, there are often communication requirements, as participants need to share information. Maintaining data integrity, resilience, and security in data access are some of the important features for this type of solution. In that direction, we propose an architecture for dynamic and decentralised allocation of tasks built on the idea of having communication and coordination in a multi-agent system through a private blockchain.

Keywords: task allocation, multi-robot systems, multi-agent systems, blockchain.

1 Introduction

One of the challenges in developing multi-robot systems today is the design of coordination strategies in such a way that robots perform their operations efficiently [21]. Without such strategies, the use of multi-robot systems in complex scenarios becomes limited or even unfeasible.

An important aspect considered in coordination problems is task allocation [11, 21]. There are several features that should be considered by a mechanism for allocating tasks to multiple robots in real-world scenarios such as considering

* Partly supported by Federal Institute of Rio Grande do Sul (IFRS) – Campus Feliz.

the heterogeneity of robots, the impact of individual variability to assign specific roles to individual robots, and the definition and allocation of different types of tasks. This is particularly true for disasters such as flooding [17].

During a rescue phase in a flooding disaster, teams are called into action to work in tasks such as locating and rescuing victims [13]. Such teams are normally organised by a hierarchy model [15], with individuals playing different roles during a mission. Task fulfillment during the rescue stage poses a number of risks to the teams. Using robots in a coordinated way to help the teams may minimise those risks.

Our work on task allocation has been inspired by the typical tasks in flooding disaster rescue. In particular, we needed an architecture for a dynamic and decentralised task allocation mechanism that takes into account different types of tasks for heterogeneous robot teams, where robots can play various different roles and carry out tasks according to the roles they can play. Although the actual flooding rescue tasks we are dealing with is not the focus of this paper, the architecture we presented here was inspired and is being developed for such a disaster response application, in particular in case of flooding disasters.

In order to manage the information exchanged during the allocation process, our architecture proposes the use of Blockchain Technology [14]. Blockchain is becoming increasingly popular as it provides data integrity, resilience, user confidence, fault-tolerant storage (decentralisation), security, and transparency, among other features. The use of blockchain as a technology to manage information is an innovative aspect of the proposed architecture and seems to be a promising way to deal with issues of consistency, integrity, security, and so on.

The main contribution of our work is therefore an architecture for dynamic and decentralised allocation of tasks built on the idea of having communication and coordination through a private blockchain. The architecture should support a dynamic and decentralised task allocation mechanism that considers different types of tasks to heterogeneous robot teams, where robots can play different roles and carry out tasks according to the roles they play. We use the term agent to refer to the main control software of an individual robot, so our multi-robot system is effectively treated as a multi-agent system.

The paper is organized as follows. Section 2 provides the background on blockchain and task allocation. Section 3 presents the proposed task allocation architecture. Section 4 discusses a particular case study. Section 5 describes related works. Finally, in Section 6 we conclude.

2 Background

2.1 Multi-Robot Task Allocation (MRTA)

Task allocation among multiple robots (and more generally among multiple agents) consists of identifying which robots should perform which tasks in order to achieve cooperatively as many global goals as possible and in the best possible way.

Market-based approaches have been largely studied for use in multi-robot task allocation. In these approaches, the robots are usually designed as self-interested [4] and have an individual utility function which quantifies how much a task contributes to the robots' objective when executed by it. The utility function can combine several factors (such as payoff to be received, the costs incurred, etc.) [6]. The global team utility can be quantified as a combination of the individual utilities. A common mechanism used in market-based approaches are called Auctions [4]. When allocating tasks through auctions, the robots provide bids, which are usually computed based in the utility values. The robot with the highest utility value for each task wins the task. In other words, the robots need to have information about the tasks and share information (bids) with each other.

Tasks: Different type of tasks can be used to address tasks in real-world scenarios, which cannot be adequately represented by only one type of task due to their complex structures and other domain-specific characteristics. In this paper we are considering the following type of tasks defined in [22]:

- Atomic task (AT): a task is atomic if it cannot be decomposed into substasks.
- Decomposable simple task (DS): a task that can be decomposed into a set of atomic subtasks or other decomposable simple tasks as long as the different decomposed parts have to be carried out by the same robot.
- Compound task (CT): task that can be decomposed into a set of atomic or compound subtasks. When each of the subtasks need to be allocated to a different robot we call it CN task (N subtasks that need exactly N robots). When there are no constraints, the subtasks can be allocated to one up to M robots, where M is the number of subtasks (CM tasks).

2.2 Blockchain

In 2008, Satoshi Nakamoto published a paper presenting an electronic peer-to-peer cash system, called Bitcoin [14]. His proposal is based on removing a third party, allowing two willing parties to transact directly. Bitcoin is the first truly decentralised global currency system, and it is based on hash algorithms and asymmetric cryptography. Since the network is decentralised, it relies on a network of volunteer nodes to collectively implement a replicated ledger. This public ledger tracks all transactions and the balance of all system accounts.

This public ledger, also known as blockchain, is applied in Bitcoin context to avoid a double spending problem, as well as giving publicity to all transactions. The transactions performed in the Bitcoin network are grouped in order to create a block with several transactions. Figure 1 shows a version of a block content. The block is divided into four main structures: **Size** - the size of the block, in bytes; **Header** - is composed of *version*, which is a software/protocol version, *merkle tree root* is a hash of the merkle tree root of block's transaction, *difficulty* is this block target that should be achieved throughout proof-of-work,

previous block hash a hash value from the previous block in the chain, *timestamp* block creation time and a *nonce* which is a counter used for the proof-of-work algorithm; **Transaction counter** identifies how many transactions are stored in the current block and **Content block** where all block transactions are stored.

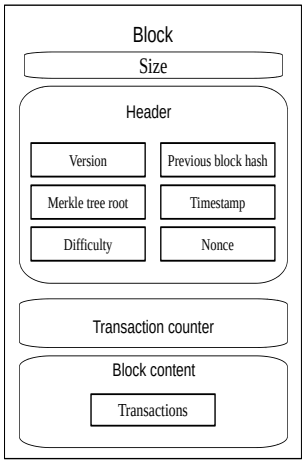

Fig. 1. Blockchain block structure

The previous block hash field, in the block header, is used to create a link between a new block and its predecessor, i.e., the hash of the previous block is stored in the previous block hash field of the new block, thus creating a chain. That is the reason for the public ledger to be called blockchain. The link between blocks is also the way how the historical information has its integrity ensured.

Since the blockchain is decentralised, each node in the Bitcoin network keeps a copy of the ledger where all transactions are stored, thus improving the blockchain resilience. Since each node has a copy of all transactions and every node is able to create a new transaction, a mechanism should be applied to avoid malicious nodes to be able to change information. In order to define which node is able to create a block and insert it into the blockchain, a consensus algorithm is applied. The Bitcoin blockchain uses a proof-of-work consensus algorithm [18].

Proof-of-work (PoW) is an algorithm that produces a block hash value which identifies the block. The operation to produce this piece of information, is very high cost in terms of CPU and power. In contrast to validate the produced information is a very cheap operation. In Bitcoin, the work consists of creating a block hash that is compliant with some rules, for example, the hash must have the N first digits consisting of zeroes. So any client that wants to insert a new block into the blockchain, must change the nonce field in the block header until its block hash value matches the defined zeroes. In order to solve this puzzle, brute-force is used, where the client must repeat the process until they find the solution, leading to a problem related to consuming CPU and power. This led some researchers to propose the use of different consensus algorithms like Proof-of-Stake [23].

The consensus algorithm is a real need in the public blockchain, in order to establish an organized way to the block insertion. This consensus criteria could be softer when running the blockchain in a controlled environment like in a private corporate network. Hardjono [9] in his research proposes a permissioned blockchain applied to the Internet of Things context, where only some nodes (defined by the sensor manufacturer) are able to write data to the blockchain. In contrast, every network node is able to read the information available in the blockchain.

Based on the characteristics of the blockchain we can highlight its attributes: *Data Integrity*; *Resilience*; *Decentralisation*; *Transparency*; *Immutability*.

3 The Task Allocation Architecture

In this section we describe our architecture for dynamic and decentralised allocation of tasks, which is built on the idea of having communication and coordination through a private blockchain. Considering a market-based task allocation approach where an organisation provides tasks to the agents, the organisation and the agents share information with each other in a dynamic process.

In our architecture, the sharing of information regarding the allocation process, either among agents, or among the organisation and the agents, is performed through the blockchain. The idea is to have blockchain acting as a decentralised database allowing the sharing of information. Due to its decentralised attribute, the blockchain is replicated to every participant, which ensures the architecture resilience.

In order to understand the proposal, first we present a general view of a basic task allocation process considering that the robots are part of an organisation. Next we describe the basic agent and organisation structures used in the architecture. Then we provide a detailed description about the interaction of the organisation and the agents with blockchain. Finally we describe our task allocation mechanism and how it uses blockchain.

3.1 Task Allocation Process – Overview

Figure 2 shows the parts considered in a task allocation process. Initially, we consider the existence of an organisation that is responsible for announcing the tasks that need to be carried out by the agents in a given mission. We use the term agent to refer to the main control software of an individual robot of any kind. The tasks provided by the organisation can be requested by the agents available for the mission. Finally, the environment is the place where agents carry out the tasks.

Regarding the process itself, it is initially considered that an organisation has a set of agents to carry out a mission and that these agents are waiting for the tasks they will be asked to carry out (the agents start executing having no assigned tasks). At a certain moment, the organisation announces a set of tasks.

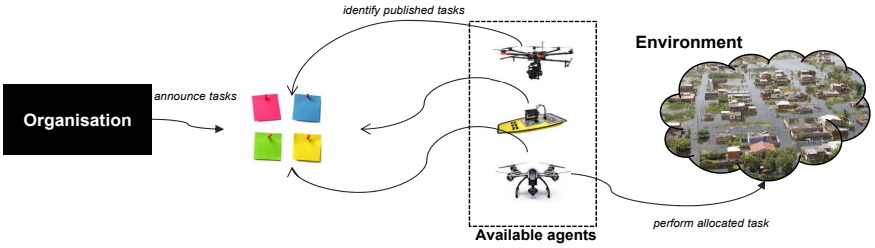

Fig. 2. Task allocation process – Overview.

When a new set of tasks is noticed, the agents begin the allocation process based on the architecture we introduce in this paper.

When the agents finish the task allocation process, those with allocated tasks start to carry them out. At the end of the allocation process, there might be agents without allocated tasks as well as tasks that could not be allocated to any suitable/available agent. Such results depend on the constraints indicated and the features of available agents.

3.2 Basic Agent and Organisation Architecture

Figure 3 shows the main aspects considered in the organisation and agent architecture. Initially, we consider agents based on the Belief-Desire-Intention (BDI) architecture [16]. The BDI architecture was used because it is widely used in several approaches. However, the proposed architecture could be easily integrated with other agent architectures, since its components preserve some independence from the other mechanisms of the agent itself.

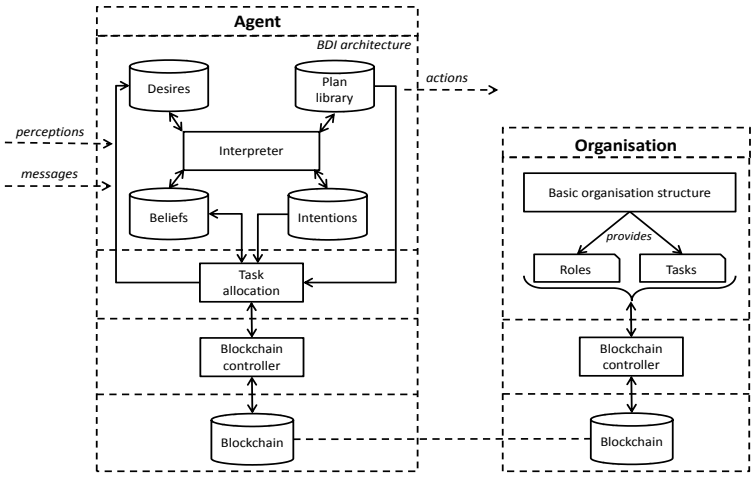

Fig. 3. Basic Organisation and Agent architectures.

Figure 3 shows a (BDI) agent with the addition of the components of our architecture. The task allocation mechanism interacts with the plan library, the belief base, and the intention base in order to have inputs for the allocation process and it also interacts with the desire base by adding new desires (goals) related to the tasks allocated to the agent. There is also an interaction between the task allocation mechanism and the blockchain controller component in order to get shared information and also to share information with other participants in the task allocation process. The blockchain controller is responsible for managing the interactions with the blockchain.

The organisation interacts with its own blockchain controller to add information related to the tasks that need to be carried out by the agents, as well as the available roles in the organisation (with capabilities required by each role).

Simply put, the organisation uses the blockchain to share information about the tasks that need to be carried out and the available roles in the organisation as well as the capabilities needed to play each role. The agents use the blockchain to share information such as their bids for the tasks during the allocation process. A detailed description of the interaction between the organisation and agents through the blockchain controller is described in the next section.

3.3 The Blockchain Controller

Here we introduce the actions which allow organisations and agents to communicate and coordinate through the blockchain. To make the description easier we will use the term entities to refer to organisation and agents. The blockchain controller provides a set of actions to allow the entities to manage and share information through the blockchain although not all are currently used in our task allocation process. The proposal is supposed to be generic enough to be useful for other solutions and not just for task allocation. Figure 4 shows these actions, which are shortly described below.

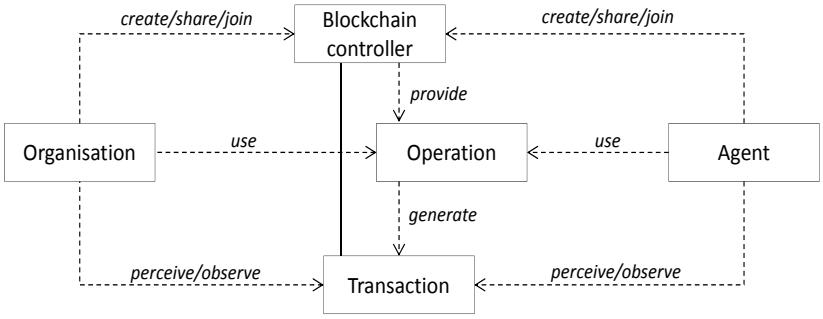

Fig. 4. Conceptual model for the interaction through blockchain.

– **Create/share/join blockchain:** the following actions can be performed by the entities to manage the blockchain.

  - *createBlockchain*: instantiates a new blockchain for the entity which executed this operation;
  - *shareBlockchain*: allows to share a blockchain with other entities;
  - *joinBlockchain*: allows an entity to join and focus on a blockchain in order to obtain and share information through that blockchain;
  - *stopPerceivingBlockchain*: allows an entity to stop receiving new transactions added to a blockchain. The entity is still able to access the blockchain data and adding new transactions to it.
  - *deleteBlockchain*: removes the blockchain from the entity that executed it.
  - *duplicateBlockchain*: create a private copy of the blockchain for an entity's own use.
- **Use operation**: An operation represents a set of instructions to allow entities to access transaction data. The following operation can be performed by the entities to add new transactions to the blokchain.
  - *insertTransaction*: allows to insert a new transaction into a blockchain.
- **Perceive/observe transaction**: the transaction represents information shared by some entity.
  - perceiving transaction: every time a new transaction is added to the blockchain the entities will be able to perceive it (entities that are sharing that blockchain). The *stopPerceivingBlockchain* action is used to stop perceiving.

3.4 The Task Allocation Mechanism

In this section, we describe our task allocation mechanism and how it interacts with the blockchain controller. Each agent in the organisation executes the task allocation mechanism, shown in Figure 5, characterising a decentralised solution.

Simply put, each agent initially perceives through the blockchain controller the tasks that need to be carried out. Based on the perceived tasks the agent identifies, through task allocation mechanism, the tasks it can carry out based on the roles it can play in the organisation, which are also perceived through the blockchain. The agent then identifies the tasks it will try to allocate to itself and calculates its bids for those tasks. The agent then communicate its bids putting that information in the blockchain through the blockchain controller (i.e., executes the *insertTransaction* operation). The bids added to the blockchain will be perceived by all other agents who will then check if some of the bids improve on its own bid for a task it allocated to itself. If that is the case, the agent withdraws that task from the list of its pre-allocated tasks and then checks which task it will bid for next, to replace the task it relinquished. These steps are repeated until all agents agree on the allocation, that is, until the tasks allocated to all agents do not undergo any further modifications.

Figure 6 presents a model with the main concepts of the proposed task allocation mechanism. As shown in the figure, we assume that an agent can have different capabilities that can be related to its type of locomotion (e.g., the possibility of sailing or flying) or even to the resources available to the agent (i.e.,

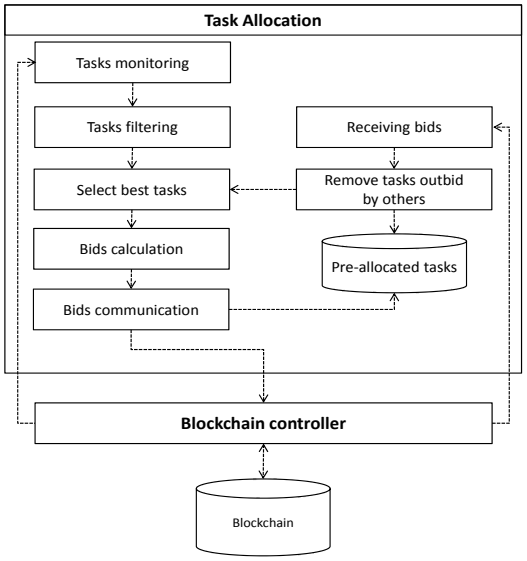

Fig. 5. The Task Allocation Model.

the robot's payload such as cameras, sensors, etc.). An agent may play one or more roles. The roles are defined by the organisation the agents belong to and each role is related to a set of capabilities that an agent needs to have in order to play that role. The organisation is also responsible for defining the tasks that are required in a given mission.

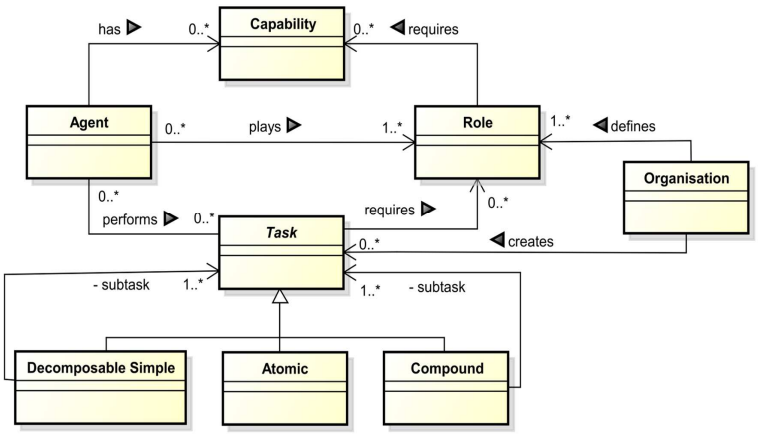

Fig. 6. Conceptual model for the main aspects in our task allocation process.

As described in Section 2, a task can be atomic, simple decomposable or compound. Compound tasks have a list of subtasks that can be, in turn, atomic or compound (this way it is possible to create a complex hierarchy of tasks,

which increases the applicability of this approach to different scenarios). Similar structures are also possible for decomposable simple tasks. Atomic tasks will not be directly considered in the proposed mechanism (they will be indirectly regarded as subtasks of both compound tasks and simple decomposable tasks). The tasks require one or more agents able to play particular roles to carry them out. That is, agents may not be able to carry out certain tasks if they cannot play the required role. We consider that each agent has a maximum number of tasks that can be allocated to itself. This constraint may be related, for example, to the amount of energy (fuel) available to the robot. This may vary among robots as well as it may vary while the tasks are being carried out.

A version of the task allocation mechanism without the use of blockchain technology has been implemented and runs in BDI agents developed in JaCaMo framework. The algorithms that constitute the mechanism are detailed in [1], where initial results, obtained from Monte-Carlo simulations, demonstrate that the proposed mechanism seems to scale well, as well as provides near-optimal allocations.

Figure 7 shows the average results of simulations using our framework, varying the number of agents (from 5 to 35) when allocating 24 tasks. Figure 7(a) shows that the performance of the proposed solution improves and is closer to the optimal solution (i.e., 100%) as we increase the number of agents. Figure 7(b) shows a small standard deviation in all simulations (comparing with the optimal solutions). The average execution time of these simulations was 4 seconds for each simulation with 5 agents up to 15 seconds with 35 agents. In [1], tests with up to 60 tasks were also performed with similar results.

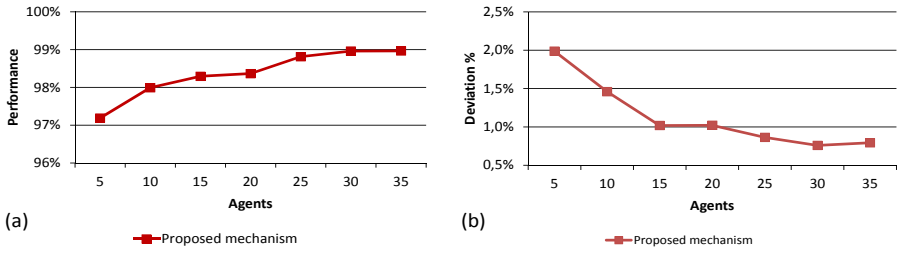

Fig. 7. Simulation results by varying the number of agents.

4 Case Study: Allocating Tasks in a Flooding Scenario

This section describes the use of our architecture through a case study on a flooding scenario. We chose this scenario because it represents a real multi-robot application scenario with several constraints that need to be considered by the software architecture, such as the heterogeneity of the robots, the impact of individual variability to assign specific roles and the accomplishment of different types of tasks. According to Murphy, there are several tasks that can be performed or assisted by robots during flood disasters [12]. One of the key tasks

to be accomplished is to obtain situational awareness of the affected region [13]. This task involves mapping the affected areas, where the robots are allocated to obtain images of a region. In order to accomplish this task, in this case study, the robot needs to have flight capability and a camera to obtain the images. Another important task in flood disasters is the collection of water samples for analysis, such as verifying the level of water contamination [17]. To perform this task, the robot must have navigation capability and be able to collect water samples. In this case study we will focus on these two specific tasks.

Consider an organisation that needs to work on a flooding disaster by performing tasks such as mapping areas and collecting water samples for analyses. The organisation has three robots available to help in those tasks: one USV (Unmanned Surface Vehicle) and two UAVs (Unmanned Aerial Vehicle), which we will call respectively USV1, UAV1, and UAV2. USV1 has sailing capability and resources to collect water samples while UAV1 and UAV2 have flying capabilities and cameras to take images. The predicates below represent the information each robot has about itself.

USV1: $capabilities([sail, waterCollector])$
UAV1: $capabilities([fly, camera])$
UAV2: $capabilities([fly, camera])$

In the organisation, there are two possible roles to be played: mapper and collector. In order to play the mapper role, a robot must have the capability to fly and must have a camera to take pictures. For the collector role, robots must have the capability to navigate and resource to collect water samples. The predicates below represent this information.

$role(mapper, [fly, camera])$
$role(collector, [sail, waterCollector])$

Considering the flooding scenario, the organisation has defined the following tasks to be performed. The predicates below are composed of (and in this particular order): the task identifier, the task name, the region where the task is to be performed, and the role a robot needs to perform that task.

$task(t1, collectWater, regionA, collector)$
$task(t2, takeImage, regionA, mapper)$
$task(t3, takeImage, regionB, mapper)$

Considering the above, we describe now how the task allocation process works using blockchain. First, the organisation creates a new blockchain using the *createBlockchain* action. This action returns an identifier for the created blockchain. To facilitate the explanation, consider that the identifier returned by the action is $bcTaskAlloc1$.

The organisation then can share the blockchain with the available robots using the action *shareBlockchain*, using as parameter the name of the blockchain being shared and a list of the robots with which it should be shared. The following action shares the blockchain $bcTaskAlloc1$ with the robots in the organisation.

$$shareBlockchain(bcTaskAlloc1, [USV1, UAV1, UAV2]);$$

After that, each robot will have a copy of the shared blockchain $bcTaskAlloc1$. The organisation then uses the $insertTransaction$ operation to share role and task information with the robots through the blockchain. The following operations are used to share information about the mapper and collector roles.

$$insertTransaction(role(mapper, [fly, camera]));$$
$$insertTransaction(role(collector, [sail, waterCollector]));$$

Block 1 in Figure 8 represents the role information added to the blockchain by the organisation. The blockchain technology is responsible for synchronising the role information with the copy of the blockchain available in the robots. Once the robots' blockchains are updated with the new transactions, the blockchain controller in each robot will generate a percept to the robot about the new information. Each robot is now able to identify which roles it can play in the organisation. USV1 identifies it can play role collector while UAV1 and UAV2 identify they can only play the mapper role.

The following operations are used to share information about the tasks.

$$insertTransaction(task(t1, collectWater, regionA, collector));$$
$$insertTransaction(task(t2, takeImage, regionA, mapper));$$
$$insertTransaction(task(t3, takeImage, regionB, mapper));$$

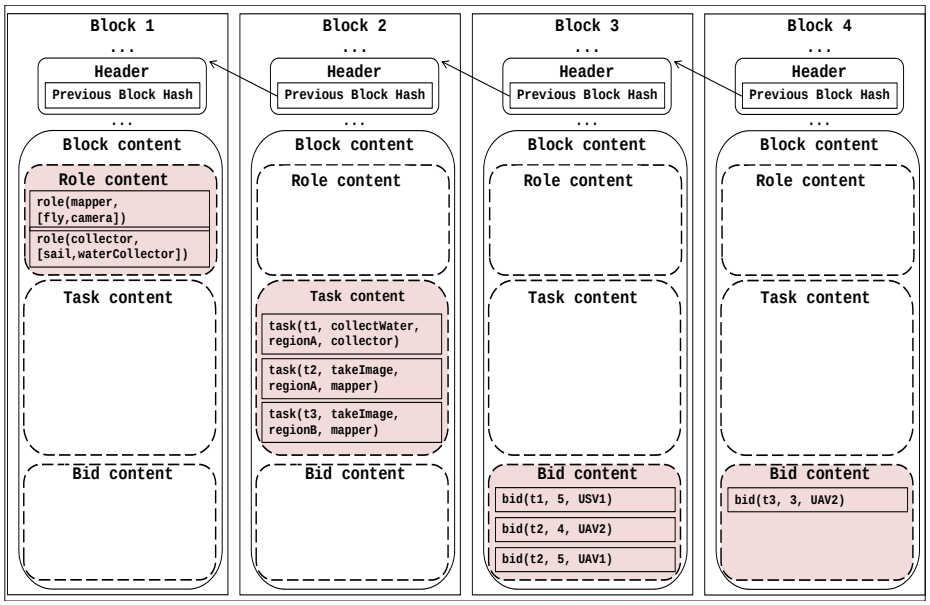

Fig. 8. Example of our blockchain content.

Block 2 in Figure 8 represents the task information added to the blockchain by the organisation. Again, the blockchain is responsible for synchronising the

information with the blockchain in the robots, and the blockchain controller in each robot will generate percepts for the robot when the new information is added to the blockchain. Each robot is now able to identify the tasks available in the organisation and the ones it can bid for based on the roles it can play. USV1 realises it can bid only for task t1, while UAV1 and UAV2 realise they can bid for tasks t2 and t3. With information about the roles and the tasks, the robots are able to start the allocation process.

We also assume that all robots start bidding at the same time, each one inserting a new transaction within their copy of the blockchain. The following operations represent the bids from each robot. In order to calculate a bid, each robot uses inputs from the plan library as well as the belief and intention bases.

USV1 operation: $insertTransaction(bid(t1, 5, USV1))$;
UAV1 operation: $insertTransaction(bid(t2, 5, UAV1))$;
UAV2 operation: $insertTransaction(bid(t2, 4, UAV2))$;

Each operation will add the transaction to the blockchain in the respective robot. The blockchain in each robot is responsible for synchronising the information about the bids with the blockchain in the other robots and also in the organisation blockchain. Block 3 in Figure 8 represents the bidding information after the synchronisation. For each new transaction updated in the robots blockchain, the blockchain controller will generate a percept to the robot (information about the current bids). Each robot is now able to identify if it has lost to another robot some of the tasks for which it placed bid (i.e., when another robot has provided a higher bid). In our scenario, UAV1 and UAV2 provided bids for the same task t2. Since UAV1 bid for task t2 is higher than UAV2 bid for the same task, the UAV2 will provide a bid to another task as specified in the following operation.

UAV2 operation: $insertTransaction(bid(t3, 3, UAV2))$;

The operation will add the transaction to the blockchain in the respective robot which again will be synchronised with the other robots and the organisation, generating percepts to the robots. Block 4 in Figure 8 represents the bidding information after the synchronisation.

Assume that robots agreed on the allocated tasks and the allocation finished after this last bid. Thus, USV1 won the bid for task t1, UAV1 won for task t2, and UAV3 won for task t3. The information about the allocated tasks will be added to the belief base, as well as new desires (goals) will be added to the desire base of each robot, so that they can start the execution of the allocated tasks with the support of their architectural components.

5 Related work

There are several works on task allocation, some of them aim at allocating an initial set of tasks to a set of robots as in [3, 11, 19], while others focus on the allocation of tasks that arise during the execution of other tasks as in [20].

Regarding the tasks, most of the solutions available in the literature, such as the ones presented in Gernert [7] and Settimi [19], focus only on atomic tasks, unlike our proposal, which comprises other types of tasks as well. Das [3] and Luo [11] are examples of work that deal with subtasks in some way.

In Gernert [7], the authors propose a decentralised mechanism for task allocation along with an architecture that focuses on exploring disaster scenarios. However, in the solution, as in many others, the robots can carry out any task, i.e. heterogeneity and capabilities are not considered. There are also works like Settimi [19] and Das [3], where heterogeneous robots and their capabilities are considered in the task allocation.

In Gunn [8] is described a framework for allocating new tasks discovered by robots in the missions. It proposes the use of heterogeneous robots organised in teams. The robot with the best computational resources is responsible for the allocation process. Thus, it could be said of that there is still a single point of failure within each team, so it is not exactly a decentralised solution like ours.

To the best of our knowledge, there are no studies using blockchain in the task allocation process either for multi-robot or multi-agent systems. In this way, we introduce as related work the use of blockchain in other applications.

Blockchain technology was first applied to the Bitcoin currency in 2008, but since then it was used in several other different applications. Ferrer's research [5] describes possibilities in the use of blockchain to improve three aspects related to a swarm robotic system: security through blockchain digital signature, public and private keys; distributed decision making where the blockchain can be applied to handle collective map building, obstacle avoidance and reach agreements; swarm control behavior differentiation considering linking several blockchains in a hierarchical manner, which would allow robotic swarm agents to act differently according to the blockchain being used.

In a different context, Lee [10] uses blockchain to control the manufacturer firmware version installed in its devices. The idea is to create a blockchain where all devices are connected through a peer-to-peer network. Through this blockchain each device can check its firmware version and, once identified the need for update, it requests to the node that has the most recent version.

Bogner's [2] research uses the blockchain applied in a Ethereum cryptocurrency. In his work the blockchain is used to handle device renting in Internet of Things context. For example if a user wants to rent a bike in a station, he just performs an operation transferring the rent value, and when the transfer is persisted in the blockchain, the station releases the bike.

In our research we identified that blockchain could be applied to solve problems in areas that are note directly related to currency. That motivates the current work in order to bring blockchain technology to handle the task allocation problem in the multi-agent context.

6 Conclusions

In this paper, we have presented an architecture for dynamic and decentralised allocation of tasks built on the idea of having communication and coordination in a multi-agent system through blockchain. The architecture was inspired by and is being developed for such an application in a multi-institution project funded by the government to address disaster response, in particular in case of flooding. Considering that real-world scenarios like flooding disasters typically require the use of heterogeneous robots and task fulfillment with different complexities and structures, our architecture takes into account the allocation of different types of tasks for heterogeneous robot teams, where robots can play different roles and carry out tasks according to their capabilities.

Our architecture takes advantage of the blockchain technology which is a promising way to deal with issues such as consistency, data integrity, resilience, security, decentralisation, transparency, and immutability. For example, using blockchain in our architecture allows all the participants to share the same knowledge about the task allocation process. Since all information about the allocation is stored in the blockchain, new robots can be added to the process at any time. The organisation (or some agent) can share the blockchain with the new robots, which will have access to the data previously stored in the blockchain. That allows the robots to synchronise their knowledge about the allocation process and so to participate in it. Security is also an important aspect for task allocation in flooding scenarios since the robots can be target of threats and attacks and that may impact in the search and rescue of victims. Blockchain uses an encryption scheme based on asymmetric cryptography which ensures the security to the information stored.

The use of blockchain as a technology to manage task allocation information is an innovative aspect of our architecture and seems to be useful also in other problems in multi-agent systems.

References

1. Basegio, T.L., Bordini, R.H.: An Algorithm for Allocating Structured Tasks in Multi-Robot Scenarios. In: Proceedings of the 11th International KES Conference, KES-AMSTA 2017.Vilamoura, Algarve, Portugal. To appear.
2. Bogner, A., Chanson, M., Meeuw, A.: A Decentralised Sharing App Running a Smart Contract on the Ethereum Blockchain. In: Proceedings of the 6th International Conference on the Internet of Things. pp 177–178 (2016)
3. Das, G.P., McGinnity, T.M., Coleman, S.A.: Simultaneous allocations of multiple tightly-coupled multi-robot tasks to coalitions of heterogeneous robots. In: Robotics and Biomimetics (ROBIO), 2014 IEEE Int. Conf. on. pp. 1198–1204 (2014)
4. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. Proceedings of the IEEE 94(7), 1257–1270 (July 2006)
5. Ferrer, E.: The blockchain: a new framework for robotic swarm systems. {Online}, Available in `https://arxiv.org/pdf/1608.00695.pdf` (2016)
6. Gerkey, B., Mataric, M.: A formal analysis and taxonomy of task allocation in multi-robot systems. International Journal of Robotics Research 23(9), 939–954 (2004)

7. Gernert, B., Schildt, S., Wolf, L., Zeise, B., Fritsche, P., Wagner, B., Fiosins, M., Manesh, R.S., Mller, J.P.: An interdisciplinary approach to autonomous team-based exploration in disaster scenarios. In: 2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014). pp. 1–8 (Oct 2014)
8. Gunn, T., Anderson, J.: Effective task allocation for evolving multi-robot teams in dangerous environments. In: Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM Int. Joint Conf. on. vol. 2, pp. 231–238 (2013)
9. Hardjono, T., Smith, Ned.: Cloud-Based Commissioning of Constrained Devices Using Permissioned Blockchains. In: Proceedings of the 2Nd ACM International Workshop on IoT Privacy, Trust, and Security. pp. 29–36 (2016)
10. Lee, B., Lee, J.: Blockchain-based secure firmware update for embedded devices in an Internet of Things environment. In: The Journal of Supercomputing. pp 1–16 (2016)
11. Luo, L., Chakraborty, N., Sycara, K.: Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks. IEEE Transactions on Robotics 31(1), 19–30 (2015)
12. Murphy, R.R.: Disaster Robotics. The MIT Press (2014)
13. Murphy, R.R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., Erkmen, A.M.: Search and Rescue Robotics, pp. 1151–1173. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
14. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. {Online}, Available in `https://bitcoin.org/bitcoin.pdf` (2008)
15. Ramchurn, S.D., Huynh, T.D., Ikuno, Y., Flann, J., Wu, F., Moreau, L., Jennings, N.R., Fischer, J.E., Jiang, W., Rodden, T., Simpson, E., Reece, S., Roberts, S.J.: Hac-er: A disaster response system based on human-agent collectives. In: Int. Conf. on Aut. Agents and Multiagent Systems. pp. 533–541. AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2015)
16. Rao, A.S., Georgeff, M.P.: Bdi agents: From theory to practice. In: Proceedings of the Int. Conference on Multi-Agent Systems (ICMAS-95). pp. 312–319 (1995)
17. Scerri, P.; Kannan, B.; Velagapudi, P.; Macarthur, K.; Stone, P.; Taylor, M.; Dolan, J.; Farinelli, A.; Chapman, A.; Dias, B.; and Kantor, G. 2012. Flood Disaster Mitigation: A Real-World Challenge Problem for Multi-agent Unmanned Surface Vehicles. Springer Berlin Heidelberg. 252–269.
18. Sleiman, M. D., Lauf, A. P., Yampolskiy, R.: Bitcoin Message: Data Insertion on a Proof-of-Work Cryptocurrency System. In: 2015 International Conference on Cyberworlds (CW). pp. 332–336 (2015)
19. Settimi, A., Pallottino, L.: A subgradient based algorithm for distributed task assignment for heterogeneous mobile robots. In: 52nd IEEE Conference on Decision and Control. pp. 3665–3670 (2013)
20. Urakawa, K., Sugawara, T.: Task allocation method combining reorganization of agent networks and resource estimation in unknown environments. In: Innovative Computing Technology (INTECH), 2013 Third Int. Conf. on. pp. 383–388 (2013)
21. Yan, Z., Jouandeau, N., Cherif, A.A.: A survey and analysis of multi-robot coordination. International Journal of Advanced Robotic Systems 10 (2013)
22. Zlot, R.M.: An Auction-Based Approach to Complex Task Allocation for Multirobot Teams. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave (2006)
23. Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A., Kishigami, J.: Blockchain contract: Securing a blockchain applied to smart contracts. In: 2016 IEEE Int. Conference on Consumer Electronics (ICCE). pp. 467–468 (2016)

View publication stats