Multi-robot Task Allocation approach using ROS

Wallace Pereira Neves dos Reis
Federal Institute of Education, Science,
and Technology of Rio de Janeiro - IFRJ Campus Volta Redonda
Volta Redonda - RJ, Brazil
e-mail: wallace.reis@ifrj.edu.br

Guilherme Sousa Bastos
Institute of System Engineering and Information Technology
Federal University of Itajubá, UNIFEI
Itajubá - MG, Brazil
e-mail: sousa@unifei.br

***Abstract*—In this paper, we discuss the implementation and simulation results of a fault tolerant Multi-robot Task Allocation (MRTA) architecture on Robot Operating System (ROS). We have chosen ALLIANCE, a fault tolerant, fully distributed, cooperative, and behavior-based system, as this architecture. In this first approach, the system parameters were find and set empirically and we do not discuss systems metrics, focusing on implementing the task allocation algorithm. The primary contribution of this paper is to show the benefits that ROS can bring to MRTA problems dealing with communication between heterogeneous robots and data management.**

***Keywords*—Multi-robot system. Task Allocation. Robot Operating System.**

I. Introduction

Multi-robot systems have been extensively studied and applied in many different areas in recent decades. These applications involve areas such as healthcare, safety, cleanliness, rescue, transportation, among others. On multi-robot systems (MRS), a critical issue is the proper task allocation, so the final goal to be achieved successfully from cooperation among robots. Thus, this work addresses the Multi-Robot Task Allocation (MRTA) problem. Complexity of the studied systems is rising and has two major sources: (a) larger robot teams sizes and (b) greater heterogeneity of robots and tasks [1]. Nevertheless, several reasons explain interests in this type of system, involving design complexity and cost, efficiency [2] and, in the future, cooperation between robot and human, which is a long-term goal for [1].

For a single robot that has several different resources can be very complex complete a series of tasks, leading to reduced overall system performance. Moreover, the loss of such a robot during a mission is critical. For example, when applied to space exploration, MRS can easily overcome a problem like to lose one robot, but if there is only one monolithic robot in the mission, its lost means the loss of developing time, resources and a multi-million dollar budget [3]. For healthcare facilities applications, which is a dynamic environment and new emergency priority tasks may randomly appear, according to Das, McGinnity, Coleman and Behera, it is better to use a multi-robot system (MRS) consisting of heterogeneous robots than designing a single robot capable of doing all tasks [4]. However, an efficient task allocation algorithm is necessary to handle with all heterogeneity of robots and tasks and with the stochastic environment to reap the benefits of using a MRS [4]. In addition to previous statements, Cao *et al* stresses that may also be derived from experiments with MRSs insight into social sciences (organization theory and economics), life science and cognitive science (psychology, learning, artificial intelligence) [2]. This work addresses join the advantages of MRS systems with the ROS advantages on implementing a MRTA system.

The rest of the paper is organized as follows: Section II will compare two lines of research in MRTA systems, emphasizing the architecture that underlies the work, ALLIANCE, and shortly present the Robot Operating System; in section III we define the approach and assumptions of this work; section IV discuss the undertaken simulations and the obtained results; finally, section V presents the conclusions of this work and suggestions for future work.

II. Related Work

A. Multi-robot Task Allocation Architecture

MRSs are different from other distributed intelligent agents (DIA) system because of their "embodiment" and implicit "real" world environment, which are more complex to model than software environments, like databases, networks [2]. Among other classifications, we first highlight the decision making process [2], whether the system is centralized or decentralized, and type of cooperation, emergent or intentional [1], [5]. Centralized systems are characterized by a single control agent that observes the whole environment and make decisions to maximize the global utility. These systems are capable of making optimal decisions by having whole environment information, with some environment size constraints. However, the central station needs a robust and permanent communication with each single robot in the system.

Not to mention that the system complexity increases as the number of robots in the system grows. A decentralized system lacks such control agent and can be divides into two categories: distributed and hierarchical [2]. In a distributed architecture, all robots are equals in respect to control, even when they are heterogeneous. While in hierarchical systems, there is a local centralization with a robot defined as a leader. Decentralized systems have several advantages in relation to centralized systems. A decentralized system requires local communication between the robots and, if necessary, an intermittent communication with the central station [6], in addition to bringing to the system fault tolerance, redundancy, reliability and scalability [2]. The last couple of topics are hard to reach in centralized schemes.

Another possible classification is regarding to the type of cooperation among robots, as pointed out by Lynne Parker [5]. The emergent type of cooperation, also called swarm-robotics, deals with a large number of homogeneous robots, each of which has deeply constraints of capabilities on its own. This approach is well suited for applications which execution time is not a critical factor and that the tasks performed are repeated widely in relatively large areas, such as parking cleaning [5]. Swarm-robotics control laws are the same for all robots in the system and usually are based on biological cooperative communities. Despite showing cooperative behavior, system individuals respond solely to stimulus from other individuals or the environment, there is no explicit negotiation or allocation of tasks [1].

On the other hand, intentional cooperation deals with a team

978-1-4673-7129-2/15 $31.00 © 2015 IEEE
DOI 10.1109/LARS-SBR.2015.20

with a limited number of individuals, typically heterogeneous robots, performing several distinct tasks [5]. Robots cooperate with each other to intentionally achieve a certain level of efficiency and complete the mission, according to [1] often through task-related communication and negotiation. However, [1] highlights that MRS need not be intentional to show coordinated behavior. Using both methods is possible to demonstrate the execution of the same task. The debate about the contribution of each approach remains open.

In order to formalize the analysis of MRTA problems and treat it with a more theoretical view, Gerkey and Matarić [1] defined taxonomy for those problems. On their definition robots, tasks and assignment are three axes that describe and represent MRTA problems. Single-task robots are robots that execute a single task at a time (ST) and multi-task robots (MT) are those that can execute multiple tasks simultaneously. With respect to tasks, a single-robot task (SR) means that each task requests merely one robot, while multi-robot task (MR) means that some tasks can require multiple robots to be completed. Lastly, task assignment categories are instantaneous assignment (IA) and time-extended assignment (TA); IA types are bounded to instantaneous task assignments with respect to available information of the environment, the robots, and the tasks, without further planning, and TA types means that more information is available and tasks can be assigned over time. Concerning task assignment, [7] shows another perspective: in IA problems the number of robots are greater than the number of tasks and in TA problems, the opposite occurs, i.e, the number of tasks are greater than the number of robots. Based on the above definitions, the analysis is extended to the combination between the robot type (either ST or MT), the task type (either SR or MR), and the type of task assignment (either IA or TA).

The architecture employed in this work contemplates the simplest MRTA problem, ST-SR-IA. Robots perform only one task at a time and tasks require only one robot. In addition, tasks allocation is instantaneously depending on the robot motivation to perform one that is not running or is not in satisfactorily progress. Refer to [1] for a complete analysis of the MRTA systems taxonomy.

MRTA architectures of intentional cooperation can be crudely divided in behavior-based approaches and market-based ones. By behavior-based approaches we mean those where robots have internal motivation that lead them to take a task, as shown on [5] and [8] as $robot\ impatience$ and $robot\ acquiescence$, and on [9] as commitment and coordination. There is no negotiation[1] among robots, once one robot starts a task, it announces1 to others and its task will not be assign to other robot[2]. The internal motivation of a robot can be modified depending on how tasks are being executed, but not by the motivation of another robot. Moreover, a robot publishes that it is already executing a task that was idle, and not that it aims to execute this task, to start a negotiation with others on the team. Regarding market-based approaches we mean those where robots communicate each other (generally by broadcast communication) and negotiate which one in going to execute a specific task by evaluating each interested robots' bid, e.g., the architectures described in [10], [11]. After robots evaluate the proper metrics for announced task and calculate a fitness score, it publishes the bid in [10]. While in [11] robots broadcast its cost to executing a task.

[1]Generally, there is no direct communication, robots broadcast a message of its current task and that message could be lost. The architecture needs tools to deal with this issue.

[2]But if there is a fault, e.g., a robot suddenly turns off; the team must be able to reassign tasks to complete the mission.

Gerkey and Matarić [12] suggest that one of the primary challenges facing MRS is how efficiently define utility of a given course of action. Especially that most if not all coordination approaches rely on some form of utility, also with different designations like fitness, cost, capability, and reward [12]. In addition, according to [7] the greater majority of MRTA architectures use non-variable utility in task allocation problem, but in real world tasks have priorities and time lifespan. Therefore, to better solutions, modeling of the problem should contemplate variable utility. ALLIANCE addresses this issue by using a type of variable utility, $robot\ impatience$ and $robot\ acquiescence$, which are variables value over time [5], [7], [8], as we will explain next.

Lynne E. Parker discusses in [5] and [8] a behavior-based, fault tolerant MRTA architecture for cooperative mobile robot control called ALLIANCE. Robots perform missions composed by loosely coupled subtasks, but with dependence upon the execution priority. The model uses an intentional cooperation between the robots. The behaviors are activated or inhibited depending on the motivation of each robot, which is influenced by the degree of $impatience$ and $acquiescence$ for each behavior. Furthermore, it takes into account information from the sensors and information from the other robots through the broadcast communication. Behaviors are equivalent to tasks, so when a behavior is selected, a task is allocated. The architecture will be better explained in the next section where we will make assumptions for our approach, which is a simpler implementation based on ALLIANCE.

B. Robot Operating System - ROS

Robot Operating System (ROS) is an open-source framework for robotic development that aim to meet a specific set of challenges when developing large-scale service robots [13]. According to the authors, the philosophies that guide the framework are unique and derived from specific previous designs to manage particular robotic systems.

Nodes, messages, topics, and services are the fundamental concepts of the implementation [13]. Nodes communicate each other through messages. Nodes publish messages in topics and other nodes should subscribe a topic to receive a message. Besides that, there are also services, analogous to web services, with a message of request and a message with response.

Arising from framework design philosophies, its main advantages are the reuse of code, the multi-lingual nature, and its adaptability since it is free and open-source. Kerr and Nickels describe in [14] a survey to determine the robotics framework that best fits an undergraduate lab. To this end, five criteria are established: easy to use, capable, adaptable, easy to install and maintain, and developmental stage. ROS scored good ratings on the criteria of adaptability and development stage. Nevertheless, it scored one of the lowest ratings on the easy to use criteria. This work shows that, despite all the advantages, ROS is not a user's environment and requires programing experience to developing. Even the authors of [13] acknowledge that the ROS is not the best framework for all robots. In addition, extend beyond, stating that they believe that this framework does not exist [13]. However, it should be held that the platform is in great development in recent years, adding increasingly features and robots.

III. Our ALLIANCE-based Approach

Lynne E. Parker sets the premises for the ALLIANCE in [5]. Out of these, the premises of this work are:

- Robots of the team can detect the effect of their own actions, with probability equal to 1, since it is a simulation.
- Team members do not lie and they are not intentional enemies.
- The overall environmental knowledge is not available on a centralized storage. Thus, for a mission in whom the mapping is critical, there should be a mapping behavior. This is not the case with this work.
- The means of communication is not guaranteed to be available and messages may be lost.
- The robots have perfect sensors and actuators, for being a simulation.
- If a robot fails, it does not necessarily communicate to others.

Based on formal of architecture model, we define a robot set $R = [robot_0, robot_1, robot_2]$ and a set of tasks $T = [task_1, task_2, task_3]$. In addition, each robot has a behavior set which is in charge of allocating tasks through behaviors activation. $Robot_1$ has the set of behavior $A_i = [a_{i0}, a_{i1}, a_{i2}, a_{i3}]$.

Our approach uses the same architecture of the input variables [5], with variations in their statement. Our approach uses the same input variables of architecture of [5]. These variables are combined to calculate the motivation level for each behavior. There is a certain motivation level that activates behaviors, called threshold of activation. When the motivation value of a certain behavior reaches threshold, the behavior is activated, i.e., a task is allocated. The main inputs are:

1) $sensory\_feedback$: This entry defines when a behavior is applicable (1) or not (0) from the information from the robot's sensors. Sensors can be both physical and virtual. In our implementation, the feedback values are constant and will change in the simulations to observe the behavior of the robots.

2) $comm\_received$: this entry is responsible for the inter-robot communication and (1) determines its value. The each robot messaging rate is called ρ_i. There is also a time parameter which increases fault tolerance, τ_i, that determines how long $robot_i$ allows to pass without receiving message from another robot before deciding that this robot has ceased its activity. With the approach using ROS, the messaging rate of other robots is determined from the task of publishing intervals allocated for each robot on the $/TaskAllocation$ topic.

$$comm\_received(i,k,j,t_1,t_2) = \begin{cases} 1 & \text{if } robot_i \text{ has received message from } robot_k \\ & \text{concerning task } j \text{ in the time span } (t_1,t_2), \\ & \text{where } t_1 < t_2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

3) $activity\_suppression$: When a behavior is activated, simultaneously begins to inhibit the other and at that time the robot actually selected a task. In the implementation, the behavior $IDLE$ does not inhibit any other, wherein the robot waits to choose a task. This entry receives value 1 for behaviors not inhibited and 0 for inhibited behavior.

4) $impatience$: robot impatience basically involves two parameters, $\delta(t)$, and ϕ. The $\delta(t)$ parameter is the level of impatience which can range if another robot executes a certain task. There are two distinguished levels, $\delta_{fast}(t)$ and $\delta_{slow}(t)$. In the last case, the robot is more patient for a running task. The ϕ parameter is a time value which determines how long $robot_k$'s communication influences the motivation of $robot_i$. This brings even more fault tolerance. The parameter ϕ of impatience, unlike the original architecture, does not vary with time, although the final motivation varies.

5) $impatience\_reset$: this input resets the motivation of a certain behavior when a robot sends for the first time a broadcast message to the selected task. The limited influence aims to not let a slow robot delay system performance as a whole.

6) $acquiescence$: acquiescence indicates the ability of the robot to realize that a certain behavior, shall read task, should be abandoned because it is not being carried out satisfactorily. There are two parameters related to this entry: ψ and λ. The ψ parameter indicates the time at which $robot_i$ maintain the behavior before allowing $robot_k$ to activate it. The λ parameter indicates how long $robot_i$ keep an active behavior before attempting another behavior, i.e., another task.

Finally, motivation is calculated by (2):

$$\begin{aligned} m_{ij}(0) &= 0 \\ m_{ij}(t) &= [m_{ij}(t-1) + impatience_{ij}(t)] \\ &\quad \times (sensory\_feedback_{ij}) \\ &\quad \times (activity\_suppression_{ij}) \\ &\quad \times (impatience\_reset_{ij}) \\ &\quad \times (acquiescence_{ij}) \end{aligned} \quad (2)$$

In the original task selection algorithm, the robot first separates them into two categories: (a) those that he knows he can perform better than the other robots and that no other is carrying out, and (b) such other tasks, as it can perform. After splitting into two categories, firstly, robot selects at first task's category until there is no more idle tasks and then searches tasks in the second category. It runs until the $sensory\_feedback$ warn there is no more tasks. Implementation in ROS, the priority of a given task was made from different $impatience$ values for each task. Thus, the highest priority task has a higher level of $impatience$, causing the motivation for this task reaches the threshold before the other motivation. Still on the original work, the robots have previous experience with its team, learning the ability of each in carrying out a task. The author of [5] and [8] shows the L-ALLIANCE learning architecture [15] that is a learning tool in MRTA problems.

The research verification used the MobileSim simulator for robots that use an open-source ARIA library [16]. This simulator was chosen to contain the real robots to be used in the next steps of the research. And for using this simulator along with ROS, it is necessary to use an interface between ROS and ARIA, which is RosAria package [17].

The robots used in the simulation were three Amigobots and a Pioneer 3DX by Adept Mobile Robotics. These robots are available in the robotics laboratory of the Federal University of Itajubá. Amigobot, Fig. 1, is a small differential-drive robot assembled with an array of eight sonars, wheel encoders, buzzer, bumpers and other sensors.

Fig. 1. Amigobot: differential drive-robot mostly used in this work.

Additionally, those robots carry wireless point access for control and communication. In the case of Amigobots, an off-board machine processes all data. To address data exchanging and robot control, we used RosAria to read only the sensors that we need and send to robots velocity commands, besides using the same wireless mean to send and receive messages of task allocation.

There is an important comparison to make about the communication between robots in the original work [5], [8] and implementation on ROS. In the original work, communication between robots takes place through an integrated radio-based positioning system. This system consists of a transceiver coupled to each robot and two sonar base stations for use in triangulating the robot positions, reporting its position to itself and the other team members. For applications restricted to small environments without physical interference with communication, such as walls, this system works well. Howbeit, the larger the area to be covered by the robots, more critical it is communication, and more complex will be using an external system. Thusly, ROS brings benefits to architecture, since communication among robots takes place through the TCP/IP protocol, i.e., over wireless means. Robots can cooperate just kilometers away over the internet. Indoors like office corridors, the already common use of routers solves the problem. Furthermore, for more complex applications such as outdoors, the location held by only one type of sensor can contain very large errors. While on location, ROS has packages and various implementations of Simultaneous Localization and Mapping (SLAM), as [18] and [19], which can improve the tracking system of the robot team.

Therefore Fig. 2 shows the system communication scheme. Robots publishes their current tasks with an "id" at the $/TaskAllocation$ topic. Similarly, all robots subscribe this topic and from received messages, update their motivation levels. The following topics $[...]/cmd\_vel$, $[...]/sonar$, and $[...]/odom$ are the interface link between ROS nodes and ARIA nodes, made by RosAria. The ROS topics aforementioned are responsible for the robot speed, distance measurement of sonar and odometry, respectively. And each robot subscribes on its own topic of speed and also publishes messages in it, thus controlling the speed of the robot. In ROS topics of measuring distance and odometry, robot is only subscriber to read the sensor values for control of the behavior.

For complex behaviors, RosAria also supports other topics that have not been treated in this study because they are not used for the sensors. In addition, ROS allows other devices are also used together with the robots, such as on-board cameras and lasers scans.

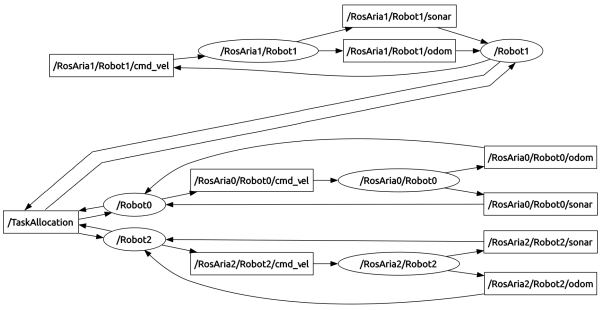

Fig. 2. Graphical representation of communication among nodes and topics of the implemented ALLIANCE architecture on ROS.

IV. Simulations Results

To check the implemented architecture, four experiments were conducted. Importantly, the focus was to verify whether the allocation of tasks occurs as defined. Accordingly, we have established four behaviors of set $A = [IDLE, WANDER, BOUNDARY\ OVERWATCH, REPORT]$. The $IDLE$ behavior, as the name suggests, keeps the robot on hold, awaiting activation of a behavior. The $WANDER$ behavior is basically the implementation of an obstacle avoidance algorithm. This behavior can be applied to more complex situations when it requires environmental mapping, for example. The $BOUNDARY\ OVERWATCH$ behavior is implementing a Follow Wall algorithm, and the robot guarda certain perimeter. Finally, the $REPORT$ behavior is performed by a robot in order to inform on the progress of the mission. In such a case, the robot that enables this behavior becomes publisher of a certain ROS topic, suitable for communication with requestor of the mission, for example.

The first simulation performed involves three robots with the same parameters except for the initial frequency of publications on the topics that are publishers. Thus, the motivation of all progresses in the same way, however, only one robot can select certain behavior. Fig. 3 shows four frames of the mission evolution.

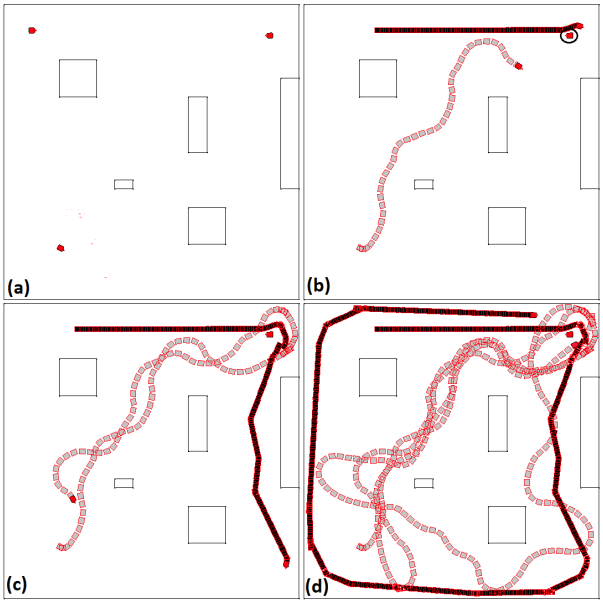

Fig. 3. The ongoings of robots to the first simulation. In (a) no task has been allocated yet. In (b) all robots already have allocated tasks, in particular a robot with a $REPORT$ task, it needs no drive. (c) and (d) only show the evolution of the execution of the tasks.

In second simulation, we changed $sensory\_feedback$ configuration parameters of the robots, thus the $robot_0$ and $robot_1$ can not accomplish $BOUNDARY\ OVERWATCH$ task, see Fig. 4 and Fig. 5. Hence, in case of $robot_2$ failure, the task would not be fulfilled. Is therefore important to task redundancy between robots even though the focus be to embracing heterogeneous robots in the system.

In the third simulation, Fig. 6, one of the robots fail. As its task was the one with the higher priority, after seconds, one of two robots remaining proceeds to perform the task.

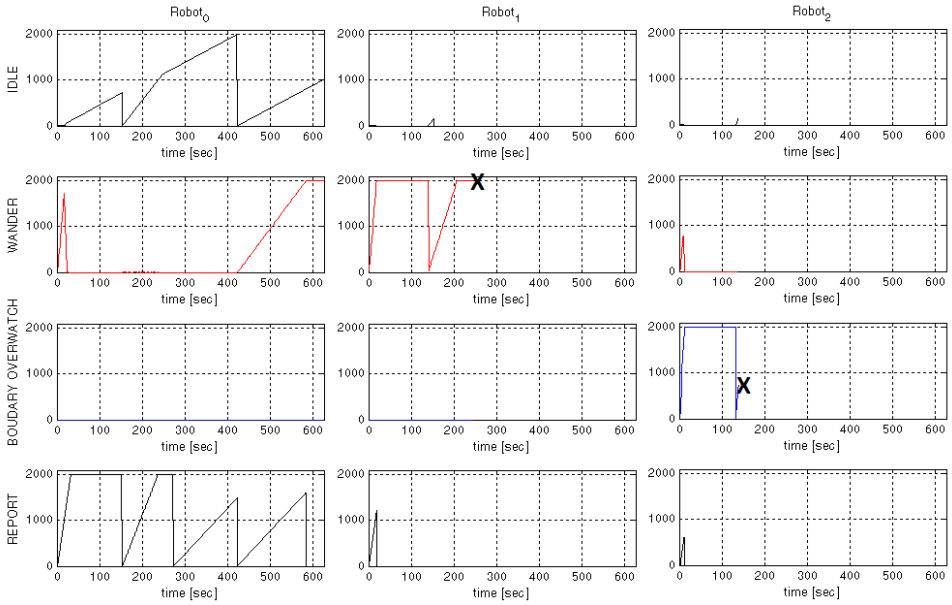

Fig. 4. Evolution of motivation over time for the second experiment. As $sensory\_feedback$ for $BOUNDARY\ OVERWATCH$ of $robot_0$ and $robot_1$ are zero, the motivation for this behavior never grows. The threshold value for all behaviors is 2000. Each behavior evolves in a certain level, because of diferent levels of impatience. The crosses in both graphs show the moment that $robot_1$ and $robot_2$ fail.

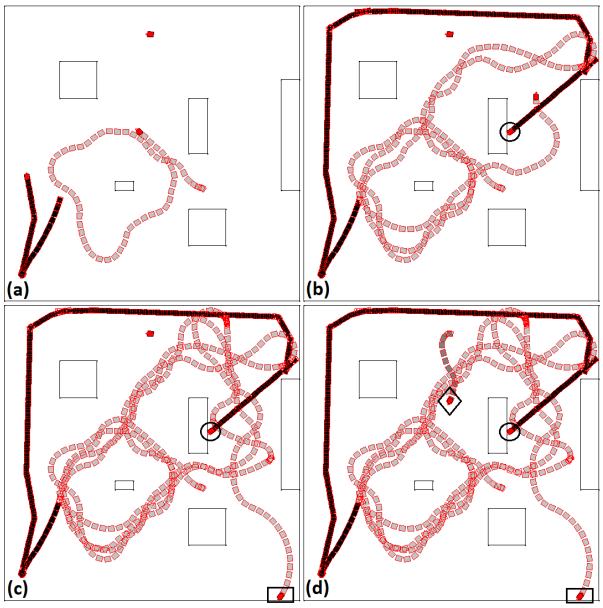

Fig. 5. Robots progress for the second simulation. In (a) robots set their behavior, i.e., allocated tasks, in particular a robot with a $REPORT$ task, it needs no drive. In (b) the robot that execute $BOUNDARY\ OVERWATCH$, marked with a circle and with the black path, fails. In (c) the robot performing $WANDER$, marked with a rectangle and with ligth gray path, fails. In (d), as the $sensory\_feedback$ of the last running robot, marked with a diamond and with dark gray path, indicates that only $WANDER$ is a applicable behavior, it selects this task.

Lastly, in the fourth simulation, Fig. 7, we intended to verify the system behavior with more robots than tasks. In such way, we placed four robots in order to represent the number of robots physically available in the robotics lab of the institution. Such experience demonstrates the ease of increasing the number of robots in the system, because the same code is used for the new member. This is also due to the fact the new robot inserted in the system part of RosAria package. Notwithstanding, as robots communicate through the same topics with the same kinds of messages, adaptations for robots with different language wrappers are small.

V. Conclusions and Future Work

This paper describes the implementation and simulation of a behavior- based, decentralized, distributed and fault-tolerant multi-robot task allocation architecture proposed in [5] in ROS. One of the advantages of the approach is the ease of implementation of communication between heterogeneous robots that already have developed interface packages. As ROS works with simple messages submission, another important advantage is the possibility of increasing the number of robots with little change in the main code. Despite having handicaps as the initial difficulty programming for beginners, the volume of packages and applications development in ROS demonstrates how the framework has gained ground among developers. For the proposed experiments, the system worked properly, particularly regarding the dynamically allocation of tasks.

Regarding future work, the implementation of L-ALLIANCE will bring a refinement in the robot configuration parameters, by learning capabilities of the robot itself and the other team members. Moreover, the study of motivation calculation function and a possible modification of the parameters must be studied, mainly on the utility point of view.

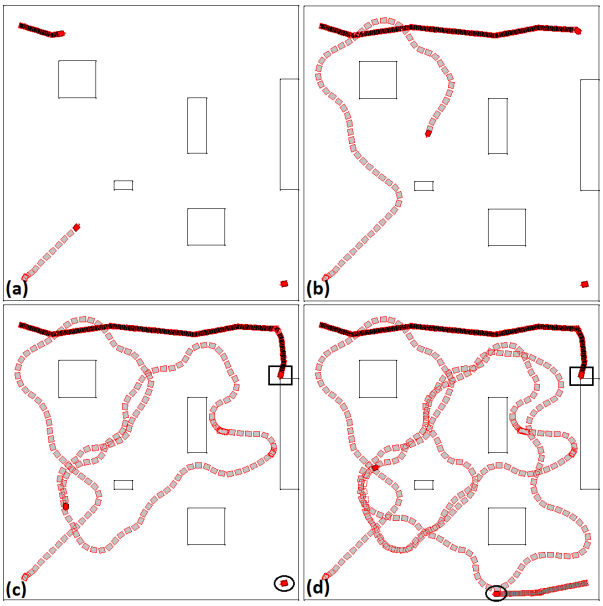

Fig. 6. Progression of robots to the third simulation. In (a) robots have just allocated tasks. In (b) the robots perform their tasks normally. In (c) the robot that performs the task $BOUNDARY\ OVERWATCH$ (the black path), marked with the square, fails and this change will affect the motivation of the robot that performs the $REPORT$ task. In (d) the robot, marked with a circle and path colored in dark gray, selects another task of higher priority, in the case, $BOUNDARY\ OVERWATCH$.

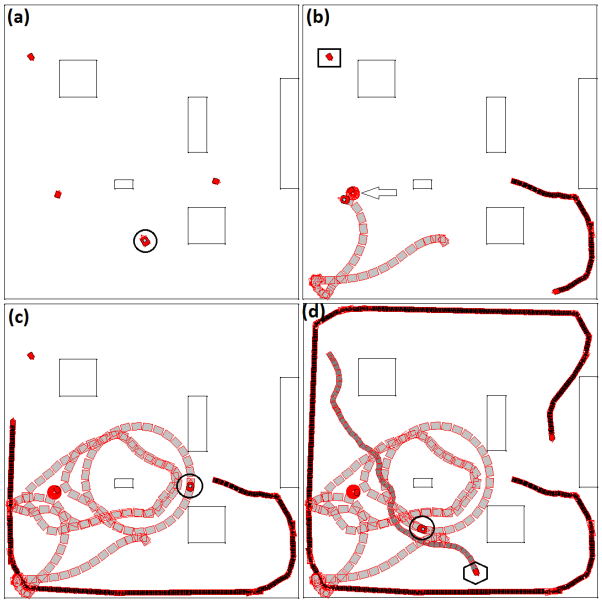

Fig. 7. Simulation with a new team member. In (a) the new member is highlighted with a circle, realize that is another robot, reinforcing the heterogeneous robot control feature of architecture. In (b) robots select their tasks. To differentiate the $IDLE$ state to $REPORT$ task, robots were programmed to be walking around when the last of the two is selected, as highlighted by the arrow. The robot highlighted with a square is in the $IDLE$ state. In (c) the new team member fails, ligth gray path. It fails to perform its task and start going in circles. In the table (d), the robot had not selected any task starts executing the task of the damaged robot, dark gray path.

REFERENCES

[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[2] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng, "Cooperative mobile robotics: Antecedents and directions," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1995, pp. 226–234.

[3] L. Yliniemi, A. K. Agogino, and K. Tumer, "Multirobot coordination for space exploration." American Association for Artificial Intelligence, 2014.

[4] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, "A distributed task allocation algorithm for a multi-robot system in healthcare facilities," *Journal of Intelligent & Robotic Systems*, pp. 1–26, 2014.

[5] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 2, pp. 220–240, 1998.

[6] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, "Multi-robot cooperation in the martha project," *Robotics & Automation Magazine, IEEE*, vol. 5, no. 1, pp. 36–47, 1998.

[7] G. S. Bastos, C. H. C. Ribeiro, and L. E. de Souza, "Variable utility in multi-robot task allocation systems," in *Robotic Symposium, 2008. LARS'08. IEEE Latin American*. IEEE, 2008, pp. 179–183.

[8] L. E. Parker, "On the design of behavior-based multi-robot teams," *Advanced Robotics*, vol. 10, no. 6, pp. 547–578, 1995.

[9] E. H. Østergård, M. J. Matarić, and G. S. Sukhatme, "Distributed multi-robot task allocation for emergency handling," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2001, pp. 821–826.

[10] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multirobot coordination," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 758–768, 2002.

[11] S. C. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1234–1239.

[12] B. Gerkey and M. J. Matarić, "Are (explicit) multi-robot coordination and multi-agent coordination really so different," in *Proceedings of the AAAI spring symposium on bridging the multi-agent and multi-robotic research gap*, 2004, pp. 1–3.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[14] J. Kerr and K. Nickels, "Robot operating systems: Bridging the gap between human and robot," in *System Theory (SSST), 2012 44th Southeastern Symposium on*. IEEE, 2012, pp. 99–104.

[15] L. E. Parker, "L-ALLIANCE: Task-oriented multi-robot learning in behavior-based systems," *Advanced Robotics*, vol. 11, no. 4, pp. 305–322, 1996.

[16] Adept Mobile Robots, "Mobilesim simulator," accessed: 25 jun. 2015. [Online]. Available: http://robots.mobilerobots.com/wiki/MobileSim

[17] ROS Wiki, "Rosaria package summary," accessed: 15 jul. 2015. [Online]. Available: http://wiki.ros.org/ROSARIA

[18] J. Machado Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in robot operating system," in *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 1–6.

[19] S. Zaman, W. Slany, and G. Steinbauer, "Ros-based mapping, localization and autonomous navigation using a pioneer 3-dx robot and their relevant issues," in *Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International*. IEEE, 2011, pp. 1–5.