Implementing and Simulating an ALLIANCE-based Multi-robot Task Allocation Architecture Using ROS

Wallace Pereira Neves dos Reis[1] and Guilherme Sousa Bastos[2]

[1] Federal Institute of Education, Science, and Technology of Rio de Janeiro - IFRJ
Campus Volta Redonda Volta Redonda - RJ, Brazil
wallace.reis@ifrj.edu.br
[2] Institute of System Engineering and Information Technology - IESTI
Federal University of Itajubá - UNIFEI
Itajubá - MG, Brazil
sousa@unifei.edu.br

Abstract. In this chapter, we discuss the implementation and simulation results of a ALLIANCE-based architecture on Robot Operating System (ROS). In this approach, the system parameters were set empirically and we do not discuss system performance metrics. The focus is implementing the task allocation algorithm. After briefly review MRTA problem, we compare known architectures in some key aspects. Although only simulations validate the ALLIANCE-based approach, system flexibility and adaptivity is notable despite its runs variations.

Keywords: Multi-robot system. Task Allocation. Robot Operating System.

1 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, not to mention others. On Multi-Robot Systems (MRS), a critical issue is the proper task allocation successfully achieving the final goal from cooperation among robots. 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 these systems, but are not limited to: decreased design complexity and cost while increased efficiency [2], dealing task complexity, increased system reliability [3], and, in the future, cooperation between robot and human, which is a long-term goal for Gerkey and Matarić [1].

When related to multi-task problems, single-robot systems demand multi-resources robots. But the system performance might decrease depending on mission challenges. Moreover, the loss of such a robot during a mission is critical. For instance, when applied to space exploration, MRS can easily overcome a

team member deficiency, but if there is only a monolithic robot in the mission, its failure means the loss of developing time, resources and a multi-million dollar budget [4]. For healthcare facilities applications, which is a dynamic environment and new emergency priority tasks might randomly appear, introducing robots means to reduce dependency on support staff, and 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 [5]. However, an efficient task allocation algorithm is necessary to handle with all heterogeneity of robots, tasks, and the stochastic environment to attain the benefits of using a MRS [5]. 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 proposes joining the MRS advantages with the ROS advantages on implementing a Multi-Robot Task Allocation (MRTA) architecture. Thus, it examines the MRTA problem and demonstrates a behavior-based MRTA architecture implementation, based on well-known ALLIANCE [6],[7], on a growing open-source framework for robotics development.

In this chapter, Section 2 presents some concepts and classifications comparing two MRTA systems, behavior-based and market-based approaches, emphasizing the architecture that underlies this work, ALLIANCE. Also, it shortly presents the Robot Operating System and its foundations. Section 3 defines the implemented architecture approach. Section 4 details the behavior sets implemented on robots discussing the undertaken simulations and the obtained results. Finally, section V presents the conclusions.

2 Related Work

2.1 Multi-robot Task Allocation Architectures

MRSs differ from other distributed intelligent agents (DIA) system because of their *embodiment* and implicit *real world* environment, requiring more complex models than software domain, like databases or 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], [6]. A single control agent observing the whole environment and accessing full system information essentially characterize centralized systems. This agent makes decisions to maximize the global utility and, by having whole environment information, it is capable of making optimal decisions, with some environment size constraints. However, the central station needs a robust and permanent communication with each single robot in the team. And it is also a weak link, a bottleneck, meaning it can never fail for the successful robots mission completion.

Not to mention the system complexity increases as the number of robots in the team grows, a decentralized system lacks such single control agent and can be divided into two categories: distributed and hierarchical [2]. In hierarchical architectures, usually in larger robot teams applications, a group defines a team

member as a local leader, centralizing on it the group decisions. Whereas, in a distributed architecture, all robots are equals in respect to control, even when they are heterogeneous. Another subcategory derives from distributed architectures: a fully distributed architecture, which is, no individual robot is responsible for others control [7]. The two last concepts differ, especially, in communication among robots and techniques used in task allocation. In a fully distributed architecture, robots communicate each other via broadcast messages, no two-ways communication is established. It means a robot does not depend on other robots acknowledgement to allocate a task although using the team members information. With regard to distributed architectures, robots need explicit or two-ways communication, since task allocation involves a negotiation [8]. As a result, a robot takes the control of task allocation process and becomes responsible for the messages concerning that task.

Decentralized architectures have several advantages in relation to centralized architectures. A decentralized system requires local communication among the robots and, if necessary, an intermittent communication with the central station. In addition to bringing fault tolerance, redundancy, reliability, and scalability to the system. In centralized schemes, reach the last two topics is a critical issue.

Another possible classification is regarding the type of cooperation among robots, as pointed out by Lynne Parker [6]. The emergent type of cooperation, likewise called swarm robotics, deals with a large number of homogeneous robots, each of which has deep capabilities constraints. 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 [6]. All robots in the Swarm-robotics system have the same control laws 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 task allocation [1].

On the other hand, intentional cooperation deals with a team with a limited number of individuals, typically heterogeneous robots, performing several distinct tasks [6]. Robots know about teammates, their actions, and states [8]. The team cooperates intentionally to achieve an exact level of efficiency and complete the mission, according to Gerkey and Matarić [1], often through task-related communication and negotiation. However, Gerkey and Matarić [1] yet highlights MRS to show coordinated behavior must not be intentional. Using both methods is possible to demonstrate same task execution. The debate about contributions of each approach remains open.

In order to formalize MRTA problems analysis and treat it with a more theoretical view, Gerkey and Matarić defined a formal taxonomy [1]. On their definition robots, tasks, and assignments are the three axes that describe and represent MRTA problems. Single-task robots can 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. Whereas multi-robot task (MR) means a task can require multiple robots to be completed. Lastly, task assignment categories

are the instantaneous assignment and the time-extended assignment. Instantaneous assignment (IA) allocation is bounded to instantaneous task assignments with respect to environment available data, the robots, and the tasks, with no further planning. By contrast, the time-extended assignment (TA) allocation means that more information is available, and tasks can be assigned over time. Concerning task assignment, Bastos [9] shows another perspective: in IA problems, the number of robots is greater than the number of tasks, and, in TA problems, the opposite occurs, i.e., the number of tasks is 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).

Even though Gerkey and Matarić formal taxonomy [1] describes a wide range of problems, there are task constraints, excluding a collection of MRTA problems. As their work assumes independent utilities, also are the tasks. For this reason, Korsah, Dias, and Stentz [10] expanded the standard taxonomy for more complex cases, calling it *iTax*, handling with interrelated utilities and constraints issues. The key concept of *iTax* is task decomposition, creating four dependencies classes, from elementary tasks to complex tasks, which involve many other tasks. And [10] adopts all previous concepts of robots, tasks, and assignment of [1]. As the implemented architecture, based on ALLIANCE and as well it, assumes independent tasks, from now on, only [1] taxonomy is considered, for simplicity.

To finish MRTA architectures classification in this chapter, intentional cooperation can be crudely divided into behavior-based and market-based approaches. There are other classifications available, the intent is not to cover it all. In behavior-based approaches, the task assignment commonly occurs without explicit robot team discussing [8]. As an example of fully distributed cooperation, robots use the external knowledge (such as feedback sensors, team members states and actions, team members' capabilities, mission status) and internal parameters to determine which robot should perform which task. Unlike behavior-based, market-based approaches involve an explicit communication among robots, since it needs to negotiate the required tasks [8]. Based on Economy market theory, robots typically try to optimize the overall utility based on individuals utilities that robots bid to perform a task. Commonly, task allocation occurs in a greedy fashion. The task negotiation needs a mediator. So a robot takes its responsibility, until the final task assignment. In market-based distributed architectures, any robot can play the mediator role. For this reason, robots could be responsible for different task assignments since all team members are equals.

As behavior-based architecture examples, there are ALLIANCE [6], [7], [8] and *Emergency Handling Task* [11], among others, in which robots have internal motivation that lead them to take a task. On ALLIANCE, *robot impatience* and *robot acquiescence* guide the motivation level, while on *Emergency Handling Task commitment* and *coordination* have this effect. No explicit negotiation occurs among robots, once a robot starts a task, the team must inhibit that task,

based on previous broadcast messages sent. Then, that task will not be assign to other robot.[3] The internal motivation of a robot can be modified depending on how tasks are being executed, but not by another robot's behavior set motivation. Moreover, a robot broadcast a message with an already assigned task that was previously idle.

There are several market-based architectures in the literature, as MURDOCH [12], M+ [13], and *Dynamic role assignment* [14], besides many others based on the same premises. In these architectures, after the newly available task announcement, robots evaluate the proper metrics and calculate a bid for task execution. This bid is the task *fitness score* in [12], the task *cost* in [13], and the task *utility* in [14]. The bidders broadcast their messages and the mediator robot, called *auctioneer* in [12] or *attach leader* in [14], evaluates the bids to assign the task. In the specific case of MURDOCH, the auctioneer monitors task progress and if it is satisfactory, it renews the contract with the winner bidder, until the task accomplishment. If it is not, the task is back to actions. Despite broadcast communication in common to above architectures, MURDOCH presents an advantageous anonymous communication, often called *subject-based addressing*, based on resource centric, *publish/subscribe* model [12], [8].

Gerkey and Matarić [15] suggest that one of the primary challenges facing MRS is how efficiently define the utility of a given action in course. Especially that most if not all coordination approaches rely on some form of utility, also with different designations as seen in previous paragraphs. In addition, according to Bastos [9] the greater majority of MRTA architectures use the 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 inherently valued over time [6], [7], [9], as better defined in Section 3. In contrast, MURDOCH *fitness score* bid depends on *metrics* forms, like simple or weighted sums of variables (such as feedback sensors and robot states), but not always taking into account the time.

With a view to formal taxonomy classification, ALLIANCE attempts to ST-SR-IA and ST-SR-TA problems [1], [8]. As defined [1], in the simpler case of instantaneous assignment, the architecture iterates task assignment in a greedy fashion until completing the mission. In the time-extended assignment, the problem is a variant MRTA problem called *ALLIANCE efficiency problem* (AEP) [1], which consists of minimizing the time taken by a robot to accomplish its allocated tasks but given each robot a subset of tasks making up the mission, not a single task [16]. Still, MURDOCH is a variant of ST-SR-IA problem, defined as on-line assignment since the architecture solve MRTA problems in which tasks are randomly injected into the system, so when a new task is introduced, the fittest robot will execute it [1]. Finally, M+ also addresses ST-SR-IA and ST-SR-TA problems. In this case, as robots know a previous task execution ordering,

[3] 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.

ST-SR-IA problems iterates negotiation until there is no task to be negotiated [1]. But, as robots can negotiate a task by anticipation, while executing a task, it also addresses ST-SR-TA problems [13]. In summary, ALLIANCE is, basically, a decentralized, fully distributed, behavior-based architecture addressing ST-SR-IA and ST-SR-TA problems with no explicit task communication among robot team members. Section 3 details it better.

2.2 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 [17]. According to the authors, the philosophies that guide the framework are unique and derived from specific previous designs to manage particular robotic systems. Among framework premises, one can highlight the multi-lingual and tools-based characteristics. The first allows the code reuse and coding in developer preference languages. The last allows the developer to decide what tools use for a specific application.

Nodes, messages, topics, and services are the fundamental concepts of the implementation [17]. Nodes are processes that perform computation. A ROS-system is commonly conprised of many nodes. And nodes communicate each other through messages, which are stricly typed data structure. Nodes publish messages in topics and other nodes should subscribe a topic to receive a message. ROS-topics follows the *subject-based addressing* concept and *publish/subscribe* model. 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 [18] 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 environment and requires programing experience to developing. Even the authors of [17] 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 [17]. However, it should be held that the platform is in great development in recent years, adding increasingly features and robots.

3 Our ALLIANCE-based Approach

Lynne E. Parker sets the premises for the ALLIANCE in [6]. Out of these, the ALLIANCE-based approach premises are:

- Robots of the team can detect the effect of their own actions, with probability greater than 0.[4]
- Team members do not lie and they are not intentional enemies.
- The overall environmental knowledge is not available on a centralized storage.[5]
- The means of communication is not guaranteed to be available and messages may be lost.
- The robots do no possess perfect sensors and actuators.[6]
- If a robot fails, it does not necessarily communicate to others.

Based on architecture formal model, one defines a robot set $R = [robot_0, robot_1, robot_2, ..., robot_n]$ and a set of tasks $T = [task_1, task_2, task_3, ..., task_m]$. In addition, each robot has a behavior set which is in charge of allocating tasks through behavior sets activation. For instance, robot r_i has the *behavior sets* or *high-level task-achieving functions* $A_i = [a_{i1}, a_{i2}, ...]$.

The ALLIANCE-based approach uses the same input variables of original architecture in [6]. These variables are combined to calculate the motivation level for each behavior. The specific motivation level that activates behavior sets is the threshold of activation parameter, θ. Since behaviors impatience and acquiescence vary throughout the behavior sets, a single threshold of activation is sufficient. 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 physical or *virtual* sensors at time t. Each behavior set has proper *sensory_feedback* function to determine when it is applicable. Some function inputs, are messages received from ROS topics, others are outputs from behavior sets.

2) *comm_received*: this entry is responsible for the inter-robot communication and (1) determines its value. The robot messaging rate is called ρ_i and express the rate robot r_i publishes its current activity on the specific ROS topic: $/TaskAllocation$. There is also a time parameter which increases fault tolerance, τ_i, that determines how long robot r_i allows to pass without receiving message from another robot r_k before deciding that r_k has ceased its activity. So, inter-robot communication function is

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

[4] In this chapter, since all experiments are simulated, this probability is equal to 1. Even if sensors and robot models includes deviation errors, they never fail completely. But, the architecture covers the real robot case.

[5] For the simulated experiments, robots previously know map limits and each of which initial position.

[6] Sensors models in the simulator include a standard deviation error.

3) *activity_suppression*: When a behavior set is activated, it simultaneously begins to inhibit the other robots behavior sets. On ALLIANCE-based approach, the behavior set $IDLE$ does not inhibit any other, wherein the robot waits to activate a behavior set and execute a task. This entry receives value 1 for behavior set not inhibited and 0 for inhibited behavior sets.

4) *impatience*: robot impatience basically involves two parameters, $\delta(t)$, and ϕ. The $\delta(t)$ parameter is the level of impatience which robot r_i can range if another robot r_k executes a task. There are two distinguished levels of impatience, $\delta_{fast}(t)$ and $\delta_{slow}(t)$. In the last case, the robot is more patient for a running task. The ϕ parameter is a value of time which determines how long robot's r_k communication influences the motivation of robot r_i. This brings, even more, fault tolerance. After ϕ time units, r_k's communication messages does not affect r_k's motivation. 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 r_i maintain the behavior before allowing robot r_k to activate it. The λ parameter indicates how long r_i keep an active behavior before attempting another behavior, i.e., another task.

Finally, the motivational controller calculates each behavior set motivation by (2):

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

In the original task selection algorithm, a robot first separates tasks 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 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. ALLIANCE-based implementation in ROS, the priority of a given task was made from different parameters, *impatience* and *acquiescence* values for each task and each robot. Thus, the highest priority task has a higher level of *impatience*, causing the motivation for this task reaching the threshold of activation before others. Still on the original work, robots have previous experience with its team, learning the

ability of each other in carrying out a task. The author of [6] and [7] shows the L-ALLIANCE learning architecture [16] that is a learning tool in MRTA problems. Proving it simplicity, the paremeters of ALLIANCE-based architecture, such as impatience and acquiescence levels, waiting time, and communication time were empirically set.

The architecture verification used the MobileSim simulator for robots that use an open-source ARIA library [19]. This simulator was chosen to contain the real robots to be used in the next steps of the research. In addition, using real robots or this simulator along with ROS, it is necessary to use an interface between ROS and ARIA, the RosAria package [20]. This package provides an interface between ROS and Adept MobileRobots and ActivMedia robot bases.The robots used in the simulation were Amigobots 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 by Adept MobileRobots used in this work.

Additionally, Amigobots carry wireless access point for control and communication. In their case, an off-board machine processes all data. To address data exchanging and robot control, the motivational controller subscribe only the needed sensors RosAria topics and publishes robots velocity commands in the specific topic, besides using the same wireless mean to publish/subscribe messages of task allocation.

There is an important comparison to make about the communication between robots in the original work [6], [7] 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 [21] and [22], which can improve the tracking system of the robot team.

All in all, Fig. 2 shows the system communication scheme. Robots publishes their current tasks with an identification, called *robot_id*, at the */TaskAllocation* topic. Similarly, all robots are subscribers of this topic updating their motivation levels according to *comm_received* input. The following topics *[...]/cmd_vel*, *[...]/sonar*, and *[...]/odom* are the interface between ROS nodes and ARIA nodes, made by RosAria package. The ROS topics aforementioned are responsible for the robot speed commands, range measurement and odometry measurement, respectively. Each robot node, responsible for ALLIANCE-based architecture processing, processes the data of subscribed topics and publishes control messages according the active behavior set.

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.

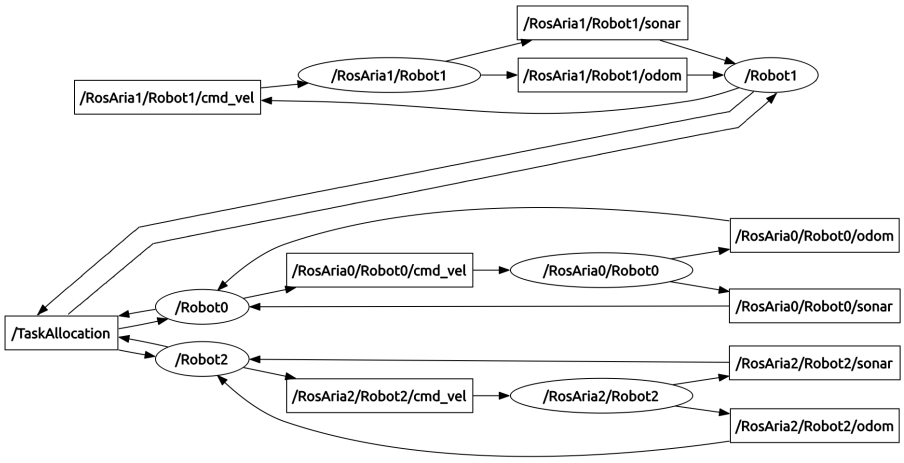

Fig. 2. Graphical representation of communication among nodes (ellipses) and topics (retangles) of the implemented ALLIANCE-based architecture on ROS for a three robots team. */Robot0*, */Robot1*, and */Robot2* are the ALLIANCE-based nodes.

4 Behavior sets details and Simulations Results

We have simulated ALLIANCE-based approach numerous times under different conditions. In this section, we describe the developed behaviors sets programmed on robots nodes and detail three experiments and their results. We present the results in simulation snapshots and robots motivation values during each simulation run. In addition to these preliminary definitions, the experiments map has always the same length and the boxes do not change places. Tasks are geographically distributed around the map, so to speak. It will be defined next.

4.1 The behavior sets

In ALLIANCE-based architecture, the tasks are previously programmed on robots as behavior sets. By activating a behavior set, a robot executes a task. The following behavior sets were implemented:

1) *wander-right-side* and *wander-left-side*: two low-level behaviors composed these behavior sets: *wander* and *side-keep*. The first is responsible for making the robot wander the environment by an obstacle avoidance algorithm, exploring the environment. The second keeps the robot in the map correct side, either left or right, from the initial position and odometry information. The behavior set *sensory_feedback* input variables are *[...]/odom* and *covered-total-distance*. Likewise, the behavior set output variables are *[..]/cmd_vel*, *covered-distance*, and *covered-total-distance*. The *covered-distance* variable outputs the distance robot has already covered with the behavior set activated. When it reaches a predefined distance value, behavior set outputs the *covered-total-distance* variable and the motivational controller takes the task as accomplished.

2) *boundary patrol*: *find-wall* and *follow-wall* low-level behaviors form this behavior set. The low-level behaviors are as simple as their names. But, once robot safely detects the wall, its map position is saved. The meaning of it is to robot surely patrol the perimeter, at least, a single time. The *sensory_feedback* input variables are *[...]/odom* and *perimeter-round*, and output variables are *[...]/cmd_vel* and *perimeter-round*. With boundary patrol behavior set activated, when the robot gets back to the same point it found the wall, the behavior set outputs perimeter-round variable and conclude the task.

3) *report*: encloses *go-to-report-area* and *publish-report* low-level behaviors. The reporting area is a predefined map location robots must achieve to report the accomplishment of the mission. This low-level behavior is a simple go-to-goal algorithm. As report behavior set is appropriate exclusively after the completion of other tasks, *sensory_feedback* input variables are *covered-total-distance*, for both sides of the map, *perimeter-round*, and *[...]/odom*. And the output variables are *[...]/cmd_vel* and *report-published*. When the behavior set outputs the last one, the mission is successfully over.

4.2 Simulations Results

The conducted experiments aim to demonstrate some of the architecture key features supported by a patrolling mission. Therefore, the mission requires robots

to patrol a room dividing the tasks into patroling the room border, patrolling the left and the right sides of the room, and report tasks completion. So that, the set of tasks for the specified mission request a single task of each behavior set. The first experiment serves as a default for a normal system run, without any robot failure or communication problems. The robots activities evolution while executing the tasks is shown in Fig. 3, and robots motivation over time in Fig. 4.

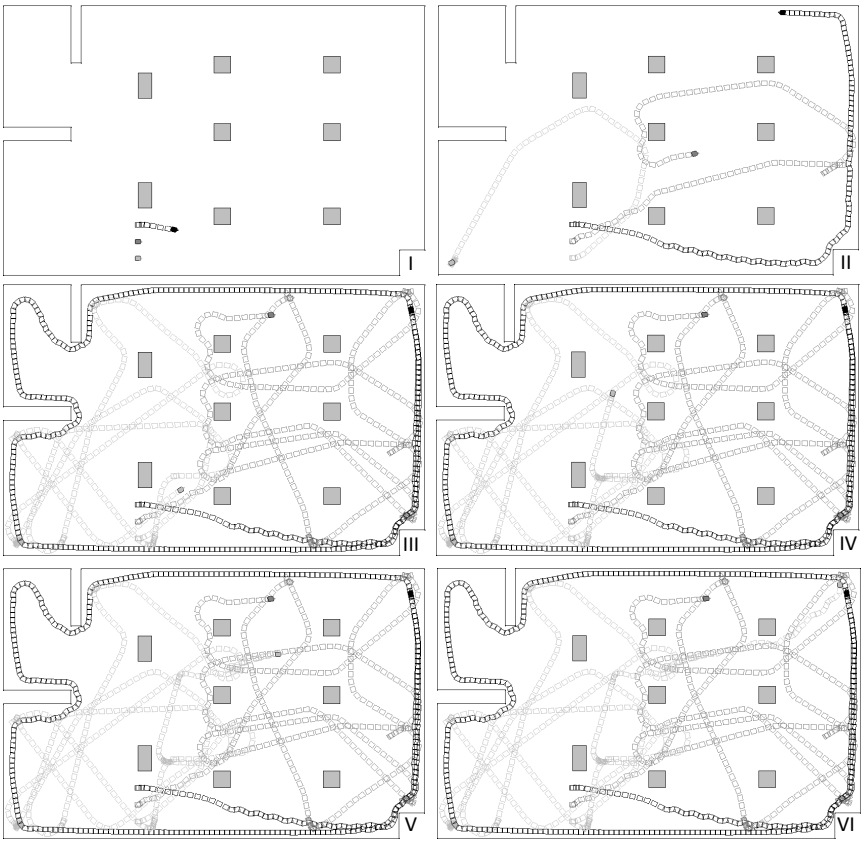

Fig. 3. Experiment 1: task allocation without robot failure or task reassignment. Robots are enumerated r_0 (black trails), r_1 (gray trails), and r_2 (light gray trails), from top first position to bottom. In frame I, r_0 has activated boundary patrol behavior set and executes its task. In the second frame, we observe that all robots have activated behavior sets. In frame III, r_0 and r_1 have already finished their tasks, however, *report* behavior set is still inhibited by *sensory_feedback*. Additionally, in the same frame, r_2 still executes first allocated task. In frame IV, r_2 complete its task and, for now on, *report* behavior set motivation starts to grow over time. In frame V, r_2 activates the last behavior set available, *report*, and accomplishes the mission, in frame VI.

At the simulation beginning, the motivation of robots grows according to the fast level of *impatience*, since all tasks are idle. As shown in behavior sets, *report* manifests only after receiving the applicable *sensory_feedback* from the other behavior sets of tasks completion. After the behavior set activation with regard each robot, it inhibits other behaviors, throughout the time tasks are performed satisfactorily. With the end of the three main tasks, one of the robots must report the mission ending, heading to the location already specified in low-level behaviors.

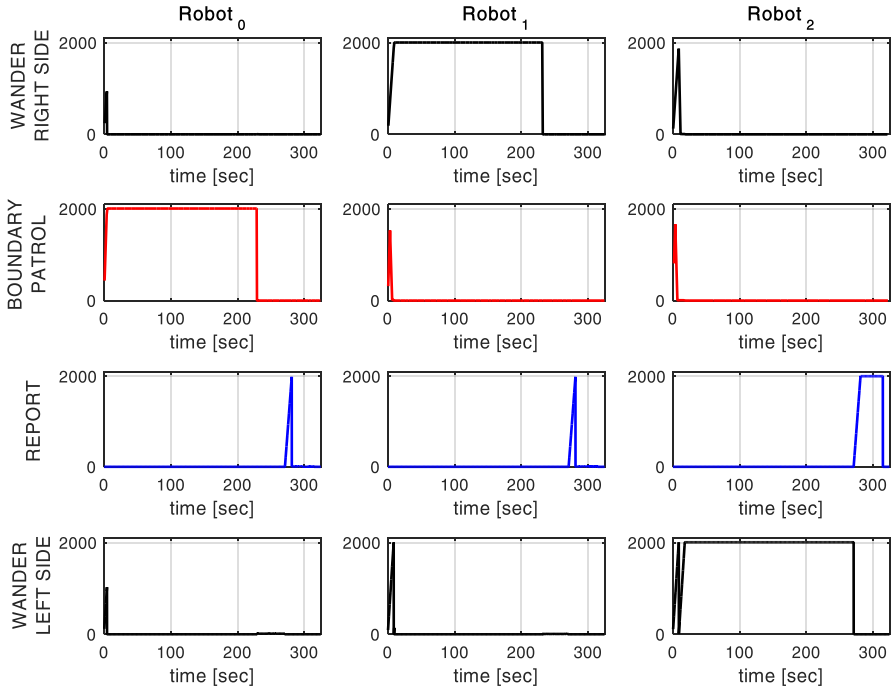

Fig. 4. Evolution of motivation values over time for the first experiment. As *sensory_feedback* for *report* behavior set, of all robots, inhibits itself, its motivation only grows when other tasks are complete. The threshold value for all behavior sets is 2000. Each behavior evolves to a particular level, because of different levels of *impatience*.

The second experiment examines architecture robustness of the point of view of failure, namely the lack of progress and noisy communication, as Fig. 5 and Fig. 6 show. Here, we consider a lack of progress a non-critical robot failure, i.e., failures during the task execution, such as delays, that do not represent a total collapse of a robot.

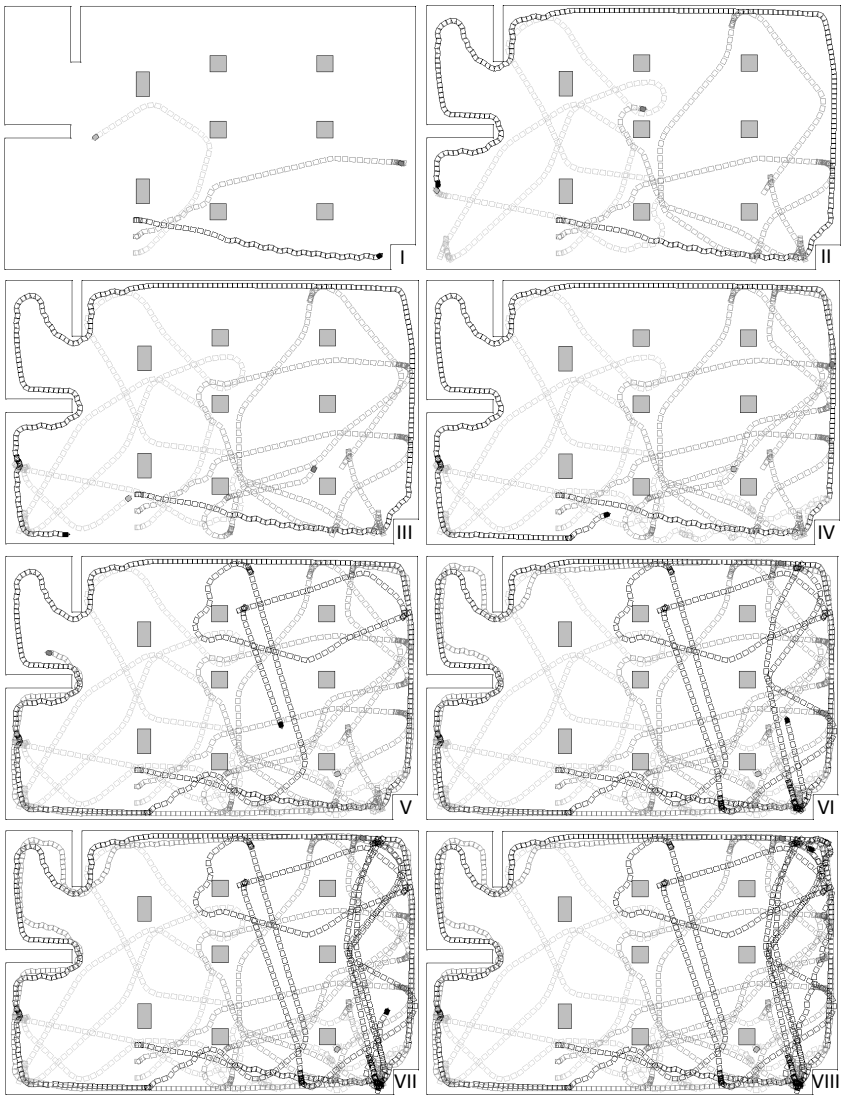

Fig. 5. Experiment 2: lack of progress and noisy communication. Robots are enumerated as the previous simulation. In frame I and II, robots activate behavior sets and start tasks execution. Still in the second frame, r_0 gets stuck in r_2, what harmed their task execution time. In frame III, robots still executing the same tasks. But in frame IV, r_0 has activated *wander-right-side* and r_1 has activated *boundary-patrol* behavior sets. Frame V shows tasks evolution, after the task reassignment. In frame VI, robots have finished their primary tasks, but due to noisy communication, *report* behavior set could not be activated. Finally, r_0 activates *report* behavior set in frame VII, and in frame VIII, it finishes the mission.

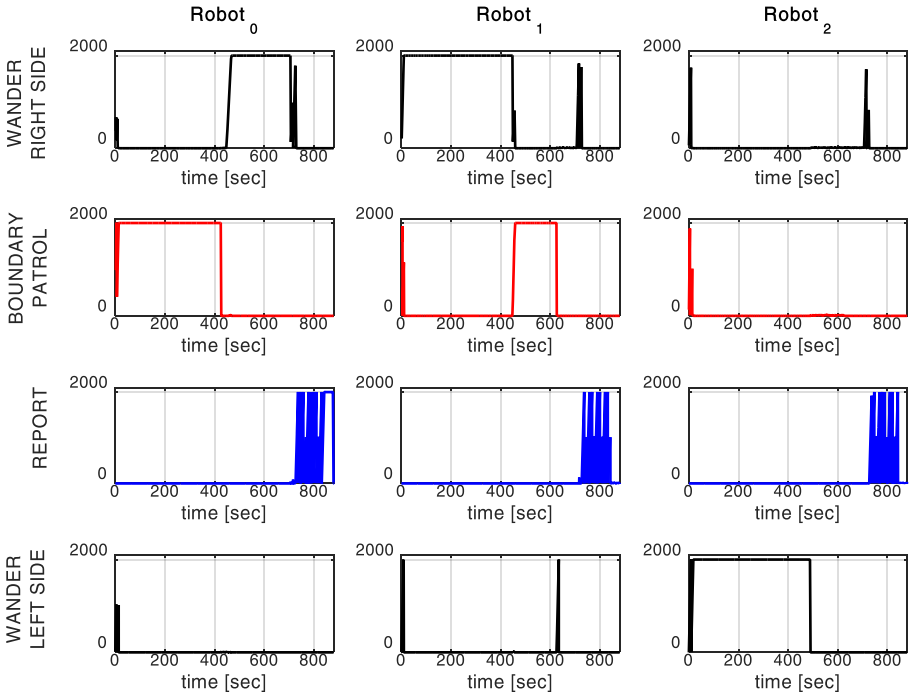

Fig. 6. Evolution of motivation over time for the second experiment. The threshold value for all behavior sets is 2000. The delay in task execution by robots "stuck" causes exchange in tasks, arround 400 sec. Because of the noisy comunication, robots motivation evolves but the behavior set REPORT could not be activated. The noisy communication causes one robot inhibits another, but ate the same time. After a while, $robot_0$ activates REPORT and then finishes the mission.

During the simulation, r_0 and r_2 could not avoid a collision and end up stuck for a while. This lack of progress causes a delay in completing the task, which is observed by all team members. Thus, the *acquiescence level* of r_0 regarding *boundary-patrol* grew and the robot left the task, turning off the respective behavior set. Thus, the r_1 activated the set of behaviors for *boundary-patrol*, since their degree of *impatience* reached the threshold of activation value. Along these lines, the mission carried out remained in execution even with a tasks reassingment. Continuing with the experiment, the effects of noisy communication appear in the report activation since this behavior set requires the outputs of the other sets. Thus, lost messages inhibit the activation of behavior sets. After a period of conflict, a robot actives behavior set and ends the mission.

With a new team member the third experiment aims to verify architecture robustness as regards to robot total failure, however with redundant resources. Noisy communication still affects robots. With the resource redundancy, we ob-

serve the task allocation of the failed robot is virtually instantaneous, not causing harm to the mission, as we can see by robots motivation in Fig. 7. During the simulation, Fig. 8, robot r_1 fails and immediately stops communicating with other team members. We recognize that task reassignment in such case occurs in a very quickly manner than when failed robot can not perceive its failure for a while. Thus, acquiescence parameter so important, to prevent a mission to stops because of a team member.

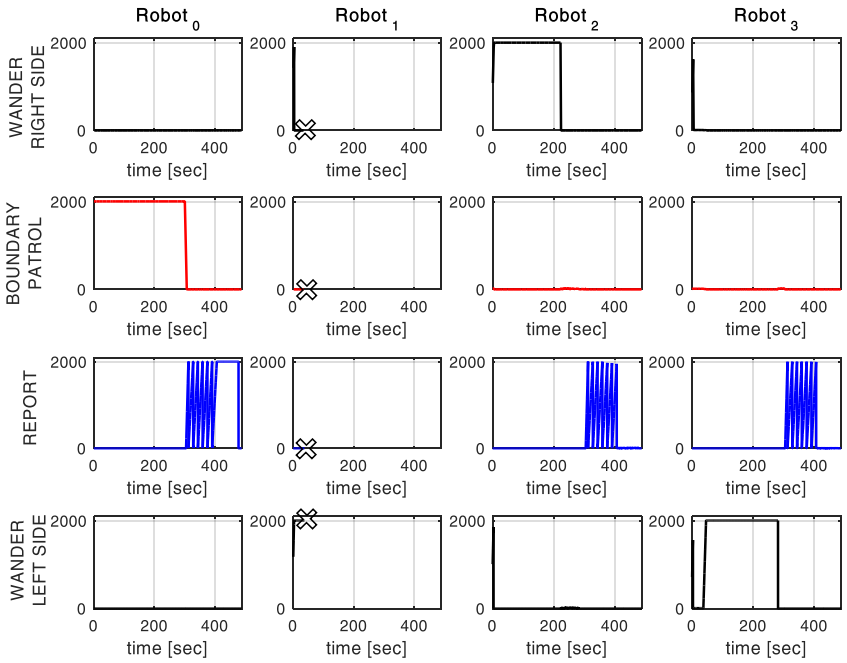

Fig. 7. Evolution of motivation over time for the third experiment. The threshold value for all behavior sets is 2000. The crosses in graphs show the moment that $robot_1$ fails.

5 Conclusions

This paper describes the implementation and simulation of a fully distributed, behavior- based, decentralized, and fault-tolerant multi-robot task allocation based on architecture proposed in [6] using ROS as development framework.

Fig. 8. Experiment 3: Introduction of a new team member and occurrence of a robot total failure. With a new member, robots are enumerated r_0 (black trails), r_1 (gray trails), r_2 (light gray trails), and r_3 (gray trails), from top first position to bottom. In frame I, r_0, r_1, and r_2 execute the available tasks. Frame II shows r_2 failure. The new team member, r_3 activates the behaviors set in frame III. In frame IV, both wanderside tasks are complete. So, r_0 finishes its task in frame V. Also r_0 activates REPORT and finishes the mission, as showed in frame VI.

One of the advantages of the approach is the ease of implementation of communication between 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 and reallocation of tasks, adaptability, and fault tolerance. Although only simulations validate the ALLIANCE-based approach, unexpected failure can occur, such as robots crashing. Even so, it is possible to perceive system flexibility and adaptivity despite its runs variations.

References

1. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research 23(9) (2004) 939–954
2. Cao, Y.U., Fukunaga, A.S., Kahng, A.B., Meng, F.: 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. Volume 1., IEEE (1995) 226–234
3. Badreldin, M., Hussein, A., Khamis, A.: A comparative study between optimization and market-based approaches to multi-robot task allocation. Advances in Artificial Intelligence 2013 (2013) 12
4. Yliniemi, L., Agogino, A.K., Tumer, K.: Multirobot coordination for space exploration, American Association for Artificial Intelligence (2014)
5. Das, G.P., McGinnity, T.M., Coleman, S.A., Behera, L.: A distributed task allocation algorithm for a multi-robot system in healthcare facilities. Journal of Intelligent & Robotic Systems (2014) 1–26
6. Parker, L.E.: ALLIANCE: An architecture for fault tolerant multirobot cooperation. Robotics and Automation, IEEE Transactions on 14(2) (1998) 220–240
7. Parker, L.E.: On the design of behavior-based multi-robot teams. Advanced Robotics 10(6) (1995) 547–578
8. Parker, L.E.: Multiple mobile robot systems. In: Springer Handbook of Robotics. Springer (2008) 921–941
9. Bastos, G.S., Ribeiro, C.H.C., de Souza, L.E.: Variable utility in multi-robot task allocation systems. In: Robotic Symposium, 2008. LARS'08. IEEE Latin American, IEEE (2008) 179–183
10. Korsah, G.A., Stentz, A., Dias, M.B.: A comprehensive taxonomy for multi-robot task allocation. The International Journal of Robotics Research 32(12) (2013) 1495–1512
11. Østergård, E.H., Matarić, M.J., Sukhatme, G.S.: Distributed multi-robot task allocation for emergency handling. In: Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on. Volume 2., IEEE (2001) 821–826
12. Gerkey, B.P., Matarić, M.J.: Sold!: Auction methods for multirobot coordination. Robotics and Automation, IEEE Transactions on 18(5) (2002) 758–768
13. Botelho, S.C., Alami, R.: M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Volume 2., IEEE (1999) 1234–1239
14. Chaimowicz, L., Campos, M.F., Kumar, V.: Dynamic role assignment for cooperative robots. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. Volume 1., IEEE (2002) 293–298

15. Gerkey, B., Matarić, M.J.: 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) 1–3
16. Parker, L.E.: L-ALLIANCE: Task-oriented multi-robot learning in behavior-based systems. Advanced Robotics 11(4) (1996) 305–322
17. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA workshop on open source software. Volume 3.
18. Kerr, J., Nickels, K.: Robot operating systems: Bridging the gap between human and robot. In: System Theory (SSST), 2012 44th Southeastern Symposium on, IEEE (2012) 99–104
19. Adept Mobile Robots: Mobilesim simulator Accessed: 25 jun. 2015.
20. ROS Wiki: Rosaria package summary Accessed: 15 jul. 2015.
21. Machado Santos, J., Portugal, D., Rocha, R.P.: 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) 1–6
22. Zaman, S., Slany, W., Steinbauer, G.: 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) 1–5