Exportation to the Cloud of Distributed Robotic Tasks Implemented in ROS

João Rosa
Institute of Systems and Robotics
University of Coimbra
Coimbra, Portugal
jprosa93@gmail.com

Rui P. Rocha
Institute of Systems and Robotics
University of Coimbra
Coimbra, Portugal
rprocha@isr.uc.pt

ABSTRACT

Cloud computing is a paradigm shift in computation that has been gaining traction over the recent years, which is supported by the increasing ubiquity of a reliable wireless connection to the Internet. Cloud robotics, which aims at bringing this principle to the field of Mobile Robotics, allows robots accessing seemingly unlimited external computation, thus being able to free onboard computation power and perform more complex tasks or tasks that were not able to run otherwise.

This paper describes the migration of two multi-robot tasks previously implemented and tested in ROS by our research group – multi-robot SLAM and multi-robot patrolling – to a cloud robotics-based implementation using the Rapyuta framework [12], with the aim of studying the trade-off between robots' computation load decrease and bandwidth usage increase. With this purpose, both simulations and experiments with real robots were conducted.

CCS Concepts

•Computer systems organization → Cloud computing;

Keywords

Cloud Computing, Cloud Robotics, Wireless Technology, Computer Resources, Bandwidth

1. INTRODUCTION

Research and development over the years has sprouted a vast array of tasks able to be executed by robots. Some of these tasks though have to be carried out by a team of robots rather than a single one.

Although the advantages of having a team of robots are clear, its restraints are only evident when considering practical implementations, namely cost restraints. It becomes obvious that a team of robots has to be composed of multiple cheaper robots, meaning machines with less processing power and smaller batteries, but also easier to build, work with, repair and replace. This creates the need for interaction among robots in order to solve complex problems and run computationally heavy algorithms. In order to tackle the problem of communication between the elements of a multi-robot system, two main areas of study have been in research recently: *cloud computing* [8] and *robotic clusters* [9].

Cloud computing has been defined as being a model for providing remote access, on demand, to a pool of computing resources. It refers to both hardware and software systems delivered or made available primarily over the Internet, being servers, storage systems, applications or services examples of cloud computing [11]. Associated with this model is the term *cloud*, which refers to the data centre that provides such service.

This computing paradigm has been classified into three main service models, each having proven its utility which is evidenced by the comercial applications that have been created. In Infrastructure as a Service (IaaS), only minimal software (bare operating system) is provided to operate the hardware resources. The Amazon EC2 [2] is an example of such cloud service. In Platform as a Service (PaaS), the cloud provides an operating system, as well as a range of programming languages, libraries, tools and frameworks with which the consumer can develop its owns applications. As examples of PaaS we can point out the Google App Engine [1] or Heroku [3] where applications are developed and run entirely on the cloud. Finally, in Software as a Service (SaaS), the user can only access applications already implemented on the cloud, having no control over the infrastructure, servers, storage or individual application capabilities. This is the highest level of cloud structure being the most restrictive one but also the easier to use. The Google Docs is a fine example of a SaaS, where the client has access to applications like a text editor, which is usually installed and run locally on a machine, that are run instead on remote servers.

These traditional approaches to cloud computing do not cater specifically to robots nor present the right tools to develop robotic applications, thus limiting the applicability of these existing platforms to robotic scenarios [12]. However, there have been some attempts to bring this promising paradigm to the field of Robotics, also in three different delivery models.

Function as a Service (FaaS) or *Equipment as a Service* (EaaS) is a low-level model that uses the cloud to provide robot resources, such as sensors and cameras or robotic tasks

SAC 2017, April 03-07, 2017, Marrakech, Morocco

DOI: http://dx.doi.org/10.1145/3019612.3019703

solvers like SLAM frameworks, being [15] a good example. *Robot as a Service* (RaaS) is a model whereby the user has access to a robotic platform such as a teleoperated robot. As an example, authors present in [5] a system for an autonomous assistive robot that uses the cloud to improve the ability of the connected robots. *Robotic Service as a Service* (RSaaS) is a concept where the user is offered a robotic service without the user specifying the platform that executes such service.

One of the prime benefits of cloud computing is the ability to offload a computationally heavy task to the cloud, while taking benefit of its parallel processing capabilities to have a relatively low execution time when compared to a machine with good processing power running the same task locally. Hence, the following issues arise: How much data should be offloaded to the cloud? How should the cloud spread the task across its resources? Energy consumption and deadline requirements are deciding factors when offloading a task to the cloud [7].

Time required in communication between robot and the cloud must also be taken into consideration, especially when considering mobile robots whose only reasonable source of connection is through some wireless network. Wireless communication systems are prone to delivery failures, either by environmental reasons or channel clutter. These particularities combined with package overhead can make a cloud-based solution to a multi-robot system less desirable, especially when considering real-time scenarios.

Finally we can identify security challenges. For obvious reasons, a remote data centre can be attacked, more so if it interfaces with its users wirelessly. Consequently, strong barriers against outside malicious threats must be taken into consideration, with virtualization of resources being the most used form of security. Cloud stored data must also be protected, so confidentiality protection mechanisms are also needed to ensure data integrity and privacy.

Overall it is difficult to predict the benefits of a Cloud-based system. Even in tasks that require a large amount of data, having more information being fed remotely does not result necessarily into better performance [16]. This coupled with other initiatives that look towards the future, such as the *Internet of Things* [4], serves as additional motivation for this research.

Section 2 presents how the robotic tasks implemented previously in ROS [14] were adapted in this work to a cloud environment. Section 3 presents the experiments and its results. Finally, Section 4 discusses the conclusions of this work.

2. EXPORTATION OF TWO MULTI-ROBOT TASKS TO THE CLOUD

In this section, we present the two multi-robot tasks that were exported to a cloud environment. This cloud environment was created using the Rapyuta framework [12] and below we present the systems adapted to the communication mechanisms provided by this framework.

2.1 Multi-robot SLAM

The first use case is based on a ROS stack previously developed by our research group for multi-robot SLAM [10], which enables any working single-robot SLAM technique to be performed by a team of robots, provided that such technique conforms to ROS standards [14] and outputs occupancy grids. The ROS stack for multi-robot SLAM has five running nodes [10], as seen on Fig. 1 (note that the node *slam_gmapping* is not a part of the stack).

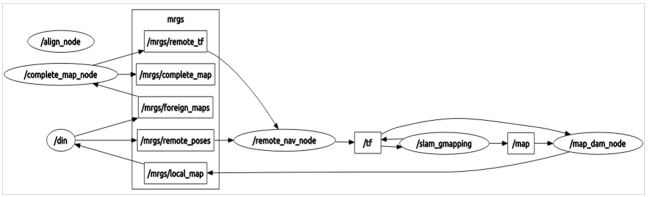

Figure 1: Interaction between the nodes and topics of the multi-robot SLAM stack [10].

The *map_dam_node* is a simple auxiliary node that subscribes to the map topic, which is fed by a single-robot SLAM technique such as *gmapping* [6] and crops it. The *data_interface_node* is responsible for broadcasting the robot's local map to the remaining robotic agents, while receiving their local maps and publishing them in a map vector.

The *complete_map_node* upon receiving an updated map vector calls a service provided by the *align_node* building a global map in a hierarchical fashion. As the diagram in Fig. 2 shows, maps are coupled and merged. The resulting map from the merging process then moves up in the pyramid, being itself coupled with another map resulting from the same merging process. In the event of an odd number of maps, the last one automatically moves up. This process repeats itself until only one global map is found. Whenever one of the local (bottom) maps is changed only the subsequent maps that depend on it are remerged, thus avoiding costly and unnecessary merging operations.

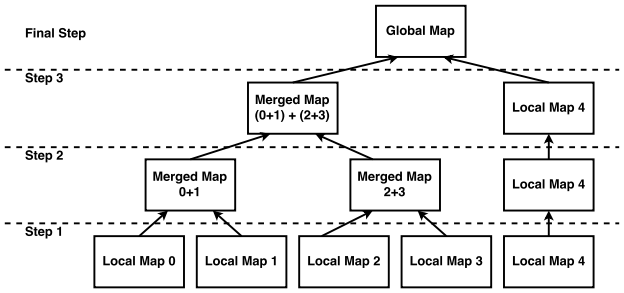

Figure 2: Tree-like process of map merging [10].

The *align_node* provides a service that receives two occupancy grids as input, computes a transformation between them using computer vision, and outputs a new occupancy grid representing the result of the merging process. Finally, the *remote_nav_node* is another auxiliary node, whose function is to propagate the transform information from the *data_interface_node* and the *complete_map_node* to the *tf* ROS topic.

These five nodes are modular enough to enable three different configurations [10]. If there is only one instance of the nodes that generate the global map, the system assumes a centralized configuration. On the other hand, if there is an instance of these nodes for every robot then we are upon a distributed configuration. It is also possible to set up a mixed configuration. All communication passes through

the *data_interface_node*, thus the topics this node subscribes and publishes are handled by the appropriate Rapyuta interfaces. By exporting everything to the cloud, including the SLAM node (*gmapping* in our case), we get a multi-robot configuration of the system as shown in Fig. 3.

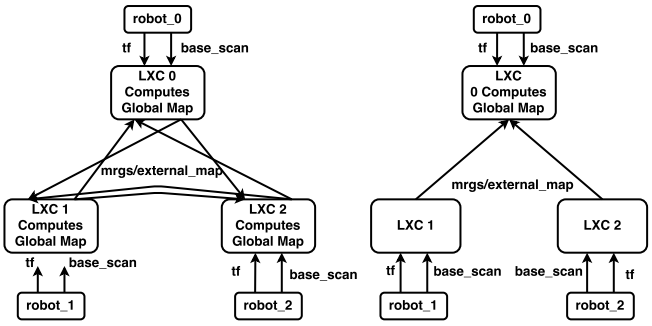

Figure 3: Comparison between distributed (left) and centralized (right) modes of operation in the cloud environment. The boxes labeled with LXC represent Linux containers running on cloud machines.

2.2 Multi-Robot patrolling

The second use case we use in this work is based on a ROS package previously developed by our research group that implements multi-robot patrolling algorithms [13]. The *patrol_isr_demo* package takes advantage of the ROS navigation stack in order to provide robots with a way of moving through the environment. It implements the Travelling Salesman Problem (TSP) and the Concurrent Bayesian Learning Strategy (CBLS) methods of patrolling. The TSP technique implemented was selected in this work as it is well known but also offers more predictable agents' trajectories. The package also implements a *monitor* node, which only aids in the synchronization of the robotic team's agents and logs all the data, thus the node can be shut down after the patrolling starts.

The patrolling task is achieved by the cooperation between the *TSP* and the previously mentioned navigation stack nodes (Fig. 4). The *TSP* node computes a trajectory based on the algorithm with the same name and informs the remaining agents of its intentions. Secondly, it provides a goal (or target pose) to the navigation stack, which represents the next vertex to be visited. The navigation stack then takes care of moving through the environment as well as avoiding obstacles.

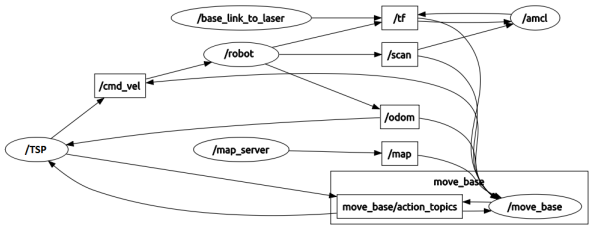

Figure 4: Interaction between the nodes and topics of the multi-robot patrolling package [13].

In order for the *TSP* node to compute a valid route and to send valid goals, two files containing detailed map information are needed. Topological information from a map is used as a way of determining the regions of interest (or vertexes) in the multi-robot patrolling problem [13]. With this purpose, a simple text file containing all the necessary information to make a web of vertexes is used to represent a map. This information includes coordinates, neighbours, direction of each neighbour and the cost to move to each one. Since the optimal TSP route (in case one exists) does not change over time, another file containing this route is fed to the node so that it does not need to be computed every time. Having established the general TSP route, the node has to determine afterwards where to start (since the route is cyclical). To do so, it consults a coded list of initial positions that, based on the robot's ID, provides a correct starting position. However, this assumes the robot with a given ID always starts on the same position.

Upon starting execution, communication among agents is required to compute a degree of proximity and to adjust trajectories accordingly, so as to avoid potential collisions. This inter-robot communication is once again handled by the Rapyuta interfaces. Finally, each goal provided to the navigation stack nodes are based on the coordinates retrieved from the file containing the map's topological information.

Throughout the whole execution of the navigation stack nodes, a considerable amount of computer resources are consumed, even if the robot is held to a stop, since the nodes are continuously scanning for obstacles and computing/sending velocity commands.

Exporting every node to the cloud we get a system as the one shown in Fig. 5.

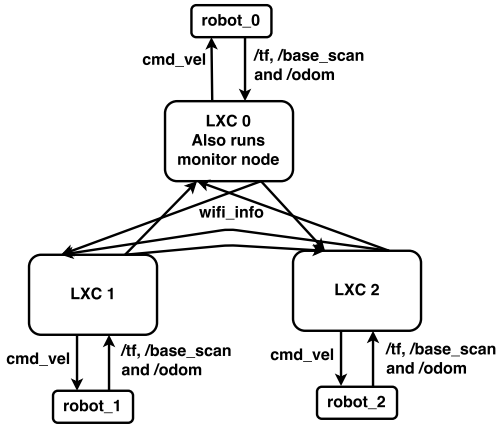

Figure 5: The multi-robot patrolling package and the navigation stack nodes running on Rapyuta's Cloud environment. The boxes labeled with LXC represent Linux containers running on Cloud machines.

3. RESULTS AND DISCUSSION

As discussed previously, the main characteristic that a cloud-based solution provides to a robotic system is the scalability it provides in terms of computer resources. Not only these resources can be made seemingly infinite to the robots, but also forming a large robotic team is made much cheaper. As such, we believe experiments involving such

a team should be conducted as a proof of concept of this highly desirable characteristic. Thus, we carried out experiments with twenty robots in a simulation environment using the *stage* simulator available in ROS [14]. We believe simulated robots are only justifiable if the order of magnitude of the size of the team of simulated robots is greater than the one achievable with the available real robots. On the other hand, we only had available three desktop machines with Intel® Core™2 Quad Processor Q6600, Intel® Core™2 Quad Processor Q9300 and Intel® Core™2 Quad Processor Q9400, which limited the cloud's resources and the size of the robotic team to the specified twenty.

Apart from providing with a proof of concept, it was also our objective to evaluate the tradeoff between CPU time and bandwidth that comes with a cloud-based approach, from the point of view of a single robot. With this in mind, experiments were conducted in order to register the CPU time spent by the considered robotic tasks and the required bandwidth in both the traditional and the cloud-adapted systems.

The use of simulation time and clock mismatches in the cloud and local machines can cause timing mismatch errors. In order to solve this, we implemented a ROS node that runs on every container and subscribes to every time sensitive topic and republishes them with the machine's local current time. Due to the topic naming that Rapyuta's interfaces provide, these topics do not overlap. Still on this topic, it is important to note that these timing considerations can have a negative impact on the robotic task or even make it unreliable to run in the cloud. This is specially true for tasks with strict hard real-time constraints.

3.1 Simulation Experiments

As stated above, two types of simulation experiments were conducted. On this first experiment, as a proof of concept, we launched twenty robots on the *stage* simulator. On the side of the cloud, there were twenty containers (one for each robot) running the core nodes for both tasks. For the multi-robot patrol task, one of these containers ran the monitor node, while on the multi-robot SLAM case there was an additional container running the merging and global map building nodes.

Each simulated robot fed the necessary data to the *rce-ros* node, which propagates them to a container in the cloud under a different topic name due to the aforementioned timing constraints. These topics were then republished under their normal name and with a new timestamp. Figs. 6 and 7 represent successful experiments of these setups.

A second set of experiments centered around the point of view of a robot, rather than the results of the system as a whole, was conducted in order to evaluate and better comprehend the tradeoff between CPU time and bandwidth imposed by this cloud solution. In this case, only one robot was launched at a time, executing its task under similar conditions while the CPU time was measured.

As Tabs. 1 and 2 show, the difference in terms of computational power required to run the system on both approaches is stark. Not only do you save up a lot of CPU time on the cloud approach, but you also gain a more predictable CPU consumption independently from the environment.

Although the differences in CPU time are clear and significant, the bandwidth aspect is trickier to evaluate. The cloud approach requires a steady amount of bandwidth as it is moving local topics that are published at a predefined rate to the cloud (10 Hz in this case). On the other hand, the traditional systems do not possess this characteristic. Moreover, in the case of multi-robot SLAM, the required amount of bandwidth depends a lot on the environment. As the

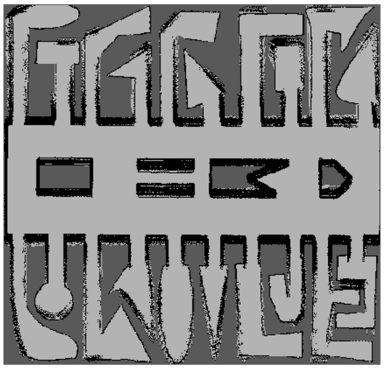

Figure 6: Global map generated through multi-robot SLAM and the merging process of 20 different local maps. Each of the simulated robots discovered the central area and one of the 20 outside branches.

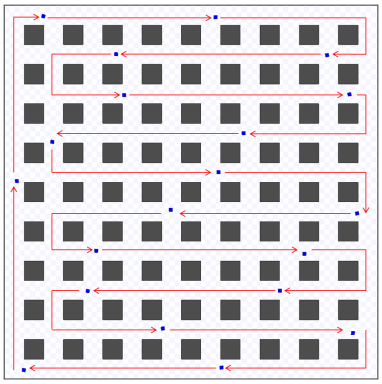

Figure 7: Successful experiment with 20 simulated robots patrolling a TSP generated trajectory. The considered vertexes are placed in a grid-like fashion, where each square is surrounded by up to 4 vertexes to a total of 100.

Table 1: Comparison between the CPU time used by the multi-robot SLAM stack on the traditional and Rapyuta-adapted systems from the point of view of a robot.

System	Mean(%)	Max(%)	Min(%)
Rapyuta-adapted	2.4	2.5	2.3
Traditional	45.0	46.1	43.7

Table 2: Comparison between the CPU time used by the multi-robot patrolling package on the traditional and Rapyuta-adapted systems from the point of view of a robot.

System	CPU time (%)
Rapyuta-adapted	3.8
Traditional	39.1

maps are compressed, a larger amount of features in the environment will lead to a worse compression rate, while a blank featureless one will lead to a higher one. The size of the environment will also take a considerable impact on the final size of the maps to be sent to the network. Maps are also not sent at a steady rate but only when a significant change to the local map is detected, thus the speed of the robot is another variable that impacts the traditional system's bandwidth requirements. On the multi-robot patrol system, robots only exchange vectors containing four elements of type *int8* (total of four bytes) among each other. This means that there is little reason to compare bandwidth requirements between the two systems as the Rapyuta-adapted one will always require much more than the traditional one. Thus, this is a rather extreme case of the tradeoff present when moving a system to the cloud.

Having said that, we registered the bandwidth used for the Rapyuta-adapted system and the amount of data propagated through the network by the topics of the traditional multi-robot SLAM system. The latter gives an idea to which degree the map size changes even on our simple and small-sized map (25x25 meters). This does not result in a direct comparison between both, but rather in an upper-bound estimate on how many robots could be viably deployed under current Wi-Fi standards.

Although the amount of data sent to the cloud is steady (see Tab. 3), there is still some variation, due to the fact that TCP requires acknowledge packages to be sent, as well as error checking that might lead to some being resent. Nevertheless, the bandwidth required by each robot is somewhat predictable. Additionally, laser scan length and sample rate are variables that we control and that have great impact on bandwidth, which gives the system an appealing flexibility.

Table 3: Bandwidth required by one robot running the Rapyuta-adapted systems. Values in KB/s

System	Mean	Max	Min
Rapyuta-adapted SLAM	31.0	35.5	25.9
Rapyuta-adapted patrol	47.1	53.5	39.4

A modern and off-the-shelf 802.11n compatible router can provide a wireless speed of 65 Mb/s (or 8125 MB/s), which can serve, in theory, over 150 robots, if we take a conservative mindset. While theoretical values might be far from practical ones, the resulting number is high enough to comfortably say that a dedicated network could serve any realistic team of robots. Furthermore, with the right setup, the 802.11n protocol is reported to reach speeds of 600 Mb/s and the newest 802.11ac over 1Gb/s. Again, despite being only theoretical values, they give a sense on how advanced wireless networks are nowadays, and that they could easily provide a good enough connection to a large robotic team.

In the case of the traditional multi-robot SLAM system, instead of simply measuring the consumed bandwidth, we registered the size of the topics used to propagate local maps through the robotic team (Tab. 4). Due to the unsteady way maps are generated, *i.e.* after a map is sent there is a considerable period of time where there is no bandwidth usage, reading the bandwidth would result in improper values. Thus we present the size of the topic at the end of the experiments.

The first point to note is that the average rate is negligible due to the fact that local maps are only sent once significant changes have occurred. During the experiments each robot only sent around 22 local maps, over the course of 2 to 3 minutes. However, while in the Rapyuta-adapted system the volume of information is steady and predictable, in the traditional system, network load comes in the form of bursts, which is not an appealing feature. Secondly, Tab. 4 shows clearly the high variability on map size growing from a minimum of 5.6 KB to a maximum of 38.0 KB. This tells us that we can never predict the total bandwidth necessary of the whole system, as the size and rate of messages depend on uncontrollable variables, and transmission over the network occurs in a burst fashion.

Table 4: Size of the topic exchanged among robots running the traditional multi-robot SLAM system.

	Mean (KB)	Min (KB)	Max (KB)	Average rate (KB/s)
rostopic	27.7	5.6	38.0	5.2

3.2 Experiments with real robots

In order to further prove the concept, experiments with real robots were also conducted. These experiments had the objective of proving the applicability of a cloud-based solution on real world scenarios. Although simulation provides a close approximation to reality under ideal conditions, real experiments are subject to erroneous sensor readings, robot slips during movement, and a not ideal wireless Internet connection. All of these impact the final outcome, and it is important for the system to be robust enough to handle such imperfections.

Fig. 8 represents a successful multi-robot SLAM experiment where all processing was done on the cloud, which was made of a single desktop for this case. Similarly to the map used in simulation (Fig. 6), there was a common corridor open to all the robots and three side branches to be explored by only one. Despite the few holes on the environment picked up by the laser-range finders and a few erroneous readings, the global map built by the system is usable and a close approximation to reality, thus providing a result equivalent to the one obtained by a traditional multi-robot system not resorting to the cloud.

Figure 8: Experiment with real robots running the Rapyuta-adapted multi-robot SLAM system.

The first idea for the following experiment was to patrol the O-shape corridor (a loop) of the floor where our laboratory is located. This was not possible because the Wi-Fi access points (AP) are located inside the laboratories, which causes the Internet connection to degrade quickly. Thus we restricted the patrolling area to the vertexes with good In-

ternet connection, which were patrolled as shown in Fig. 9. This solidifies the idea that a solid Internet access coverage is required for any Cloud-based system. Only one robot was used for these experiments for two main reasons: firstly, there was not enough space with proper Internet access to justify adding a second robot; secondly, the behaviour of the acting robot would not change by adding a team mate, unless their trajectories intersected each other.

Figure 9: Experiment of a real robot running the Rapyuta-adapted robot patrolling system.

Despite the robot patrolling just a portion of the whole building, it did so repeatedly proving the correct behavior of the cloud-based system. This tells us that if a proper Internet access coverage was available, the robot would have been able to patrol the entire floor.

4. CONCLUSION

A cloud-based solution always presented itself as a tradeoff between computer resources, namely CPU time and storage, and network bandwidth. We have successfully proven that tradeoff favourably, as high bandwidth wireless Internet connections are commonly available nowadays with off-the-shelf components. However, it should be noted that the tasks run on the cloud can not degrade with time delays, *i.e.*, there should be no hard-real time deadlines.

The cloud-based solution was also shown as a scalable one as long as the computer cluster that forms the Cloud is powerful enough and the tasks running in it enable such scaling. During the adaptation process of existing robotic tasks to the cloud, not many changes to the original systems were needed, thus this cloud-based ecosystem can coexist with traditional systems. This becomes a key aspect when developing new robotic tasks, as debugging a program running on a foreign machine is much more troublesome. We also believe the *multimaster* nature of the presented cloud environment is a big plus, as it allows for robots to be more independent, multi-robot system to be more modular, and if the robot control is done locally a robot is not lost in the case of network failures.

As for future work, it would be interesting to complement this research with an energy analysis. While processing power can consume a lot of power so can wireless communications, even while inactive.

Acknowledgments

This work was supported by ISR—University of Coimbra (project PEst-C/EEI/UI0048/2011) funded by "Fundação para a Ciência e a Tecnologia" (FCT).

5. REFERENCES

[1] App Engine - Platform as a Service. https://cloud.google.com/appengine/. Accessed: 07-Sep-2016.

[2] EC2 - Amazon Web Services. http://aws.amazon.com/ec2/. Accessed: 07-Sep-2016.

[3] Heroku: Cloud Application Platform. https://www.heroku.com/. Accessed: 07-Sep-2016.

[4] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[5] M. Bonaccorsi, L. Fiorini, F. Cavallo, et al. A Cloud Robotics Solution to Improve Social Assistive Robots for Active and Healthy Aging. *International Journal of Social Robotics*, pages 1–16, 2016.

[6] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[7] G. Hu, W. P. Tay, and Y. Wen. Cloud robotics: architecture, challenges and applications. *IEEE Network*, 26(3):21–28, 2012.

[8] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on Automation Science and Engineering*, 12(2):398–409, 2015.

[9] A. Marjovi, S. Choobdar, and L. Marques. Robotic clusters: Multi-robot systems as computer clusters: A topological map merging demonstration. *Robotics and Autonomous Systems*, 60(9):1191–1204, 2012.

[10] G. S. Martins. A cooperative SLAM framework with efficient information sharing over mobile ad hoc networks. Master's thesis, Univ. of Coimbra, 2014.

[11] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 2011.

[12] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel. Rapyuta: A cloud robotics platform. *IEEE Transactions on Automation Science and Engineering*, 12(2):481–493, 2015.

[13] D. B. S. Portugal. *Effective cooperation and scalability in multi-robot teams for automatic patrolling of infrastructures*. PhD thesis, Univ. of Coimbra, 2013.

[14] 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*, 2009.

[15] L. Turnbull and B. Samanta. Cloud robotics: Formation control of a multi robot system utilizing cloud infrastructure. In *Proceedings of IEEE Southeastcon*, pages 1–4, 2013.

[16] M. Waibel, M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, et al. A World Wide Web for Robots. *IEEE Robotics & Automation Magazine*, 18(2):69–82, 2011.