ADCAIJ: Advances in Distributed Computing and Articial Intelligence Journal
Regular Issue, Vol. 6 N. 1 (2017), 13-19
eISSN: 2255-2863
DOI: http://dx.doi.org/10.14201/ADCAIJ2017611319

Integrating Smart Resources in ROS-based systems to distribute services

Eduardo Munera[a], Jose-Luis Poza-Lujan[a], Juan-Luis Posadas-Yagüe[a], Jose-Enrique Simó-Ten[a], and Francisco Blanes[a]

[a]University Institute of Control Systems and Industrial Computing (ai2) Universitat Politécnica de Valéncia (UPV) Camino de vera, s/n. 46022 Valencia (Spain) emunera@ai2.upv.es, jopolu@ai2.upv.es, jposadas@ai2.upv.es, jsimo@ai2.upv.es, pblanes@ai2.upv.es

KEYWORD

Intelligent Sensors; Smart Resources.

ABSTRACT

Mobile robots need to manage a lot of sensors and actuators using micro-controllers. To do complexes tasks, a highly computation central unit is also needed. In many cases, a robot is a intelligent distributed system formed with a central unit, which manages and distributes several specific tasks to some micro-controller embedded systems on- board. Now these embedded systems are also evolving to more complex systems that are Distributed Systems; developed not only for executing simple tasks but offering some advanced algorithms Robot Operating System just as complex data processing, adaptive execution, or fault-tolerance and alarm rising mechanisms. To manage these types of embedded systems a paradigm, called Smart Resource has been developed. Smart Resources topology has been raised to manage resources which execution relies on a physical embedded hardware. These Smart Resources are defined as a list of distributed services that can configure its execution in order to accomplish a context and quality requirements. In order to provide a more general implementation Smart Resources are integrated into the Robot Operating System (ROS). Paper presents a solution based on the Turtlebot platform running ROS. The solution shows how robots can make use of all the functions and mechanisms provided by the ROS and the distribution, reliability and adaptability of the Smart Resources. In addition it is also addressed the flexibility and scalability of implementation by combining real and simulated devices into the same platform.

1. Introduction

Currently robots are increasing the complexity of their operation. For this reason there have been raised several proposals for easing their development: from system platforms, as robot specific operating systems, to programming languages or dedicated sensors and actuators. The Robot Operating System (ROS) (Quigley et al., 2009) is one of the most wide used software platform for robotics. In the market can be found many robots and devices that are characterized as «ROS-ready». There are a wide range of options from whole robots to only some parts such as sensors or actuators (e.g. robotic arms). ROS-ready devices allow robot to be accessed by using standard ROS communication protocol or including a certain set of libraries provided by developers.

Despite of this, there is a lack of standardization of the way these devices are managed. Furthermore, some extra capabilities like adaptability, fault-tolerance and quality restrictions are not mandatory, so they can be not implemented. Smart Resources, described in (Munera et al., 2015), are developed in order to provide distributed services which have to be executed within a certain requirements by according to the execution context. So, this implementation allows the user to know the performance of the service in any time. Context variation can be configured to trigger adaptation mechanisms to fit the new environment. Alarms will also be raised every time some requirements are not fulfilled, so user can decide if the Smart Resource is performing as expected and define the level of adaptation. Therefore, considering the convenience to make smart resource as a ROS-ready device, the main objectives to be addressed along this paper are summarized as follows:

- Review and test the advantages of implementing Smart Resources into a robotic system.
- Describe and detail the integration and usage of these Smart Resources to be characterized as a ROS ready device.
- Describe the standard interaction procedure for a ROS ready Smart Resource.
- Test and analyze this implementation on both real and simulate platform.

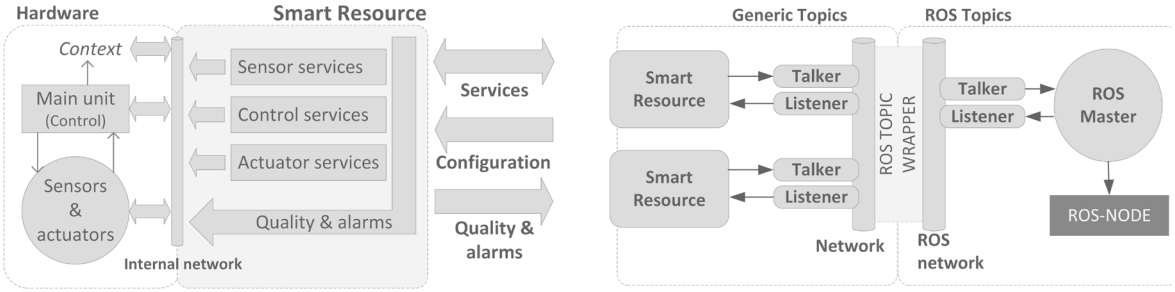
On the basis of the above, the article describes a method to integrate Smart Resources as smart sensors systems (Meijer et al., 2008). The article is organized in this order: First section presents a brief review of how sensors systems are included into ROS systems. Next section introduces smart resources used in robotic systems and their convenience for distributed systems. Thereupon, next section applies ROS integration described previously in a specific case study that describes the integration of a distributed smart resource based on a RGBD Camera in a mobile robot. Finally, some conclusions about the integration experiments are presented.

2. Related Work

To distribute sensors and actuator messages between the components of a robot, it is necessary a communication system close to the robot system hardware and some processes to manage all the tasks involved on the robot operation. In the same way that a computer needs an operative system, a robot also need a operative system (Liu et al., 2000). From all robot operating systems, ROS (Quigley et al., 2009) has many advantages as a structured architecture based on nodes or a publish/subscribe method to connect nodes between them. ROS is an open source platform, consequently it is easy to modify. Last years, ROS has experimented an important growth (Cousins, 2011). To distribute sensors, actuators and control signals among all clients of the robot or the device (for example, navigation nodes, behavioural nodes, and others), it is necessary a set of communications functions, usually called robot middleware (Elkady and Sobh, 2012). ROS uses a publish/subscribe based on topics communication system (Cousins et al., 2010). If the complexity of the nodes increase, the robot nodes, sensors and actuators become a service provider (Remy and Blake, 2011). When the robot or device services are compatible with ROS, the device is defined as ROS-ready robot (or ROS-ready device). Have ROS-ready devices provides a set of advantages to the robot designed and robot user. These advantages include the possibility to make compatible different ROS nodes, the reuse of different developments made by different research groups and companies, and the ability to simulate different devices, robots and scenarios using the multi-robot simulator, Gazebo (Koenig and Howard, 2004). There are a great amount of develops ROS ready systems. In (Chen et al., 2013) is presented a robot that uses ROS to integrate all robot sensors. PhaROS (Estefó et al., 2014) is an architecture that adapts a robot programming language to be used in ROS to program dynamically a robot. ASTRO (Saraydaryan et al., 2014) provides a service architecture oriented to big and complex robot scenarios. ROSbridge (Crick et al., 2011) provides a bridge to view ROS based systems from non-ROS users, usually web services and Internet. These are just a few examples that ROS can be used in a wide range of robotics and automation system fields. In all the cases described above, ROS is accessed by means a method to translate ROS messages to devices functionalities. Paper proposes to offer devices as resources that interchange ROS messages by means similar topics, these kind of devices are known as ROS Ready devices and it is necessary that devices offer their functionality by means of services.

3. Smart Resource Integration

Smart Resources have been introduced as a suitable option for a wide range of applications. Therefore the main advantages of implementing Smart Resources into any robot architecture is addressed along this section. Furthermore, integration of smart resources into the ROS system, is also detailed as depicted in Fig 1. With this integration, Smart Resources are available in every ROS-based platform in order to ease the configuration and access to the distributed services provided by the Smart Resources.



(a) Smart Resource architecture and interfaces *(b) Connecting smart resources with a ROS network*

Figure 1: Elements of the Smart Resource integration

3.1 Smart Resource for robotics

In order to introduce Smart Resources for robotic is required to detail how the Smart Resources capabilities can suit the needs of a robot. First of all, the type of service that can be required by any kind of robot is similar to distributed control systems. Therefore, smart resources can provide sensor, actuator and control tasks that will be offered as distributed services. These services can be provided by different nodes, executed in different hardware platforms and supplied by different devices or even simulated devices.

Distributed services provided by Smart Resources are accessed through a network interface. This interface relies on a communication middleware whose main features are: network peer detection, active peer lists management, and a publish-subscriber communication (Eugster et al., 2003). As any publish-subscriber implementation, communication is driven by topics which can be checked or updated by the network peers. Therefore three main different kinds of topics according to the kind of information can be distinguished: configuration topics for parameterizing the desired tasks execution, the required service, and the quality of the service.

As stated before, Smart Resources offer the capability of adapting to the system which is configured to work within some quality bounds. Every time a Service is requested, it must be configured in order to fulfil the need of the client. Most of these requirements are set in terms of temporal and spatial requirements, information reliability and operation performance. A certain configuration with certain quality restrictions are defined as a System Profile. System Profiles are pre-programmed in the Smart Resource. A quality requirement is translated into the equivalent (more similar) System Profile. Because of some of these qualities cannot be provided, Smart Resource must try to adapt its execution to suit the requested configuration. Alarms notifies the client every time a quality is not satisfied, being managed as an adaptation event. If minimum requirements can not be satisfied, Smart Resource will execute the most similar performance according to the existing restrictions by offering a best-effort profile. In that case system alarms make the client aware of a non-adaptable event that should be managed as an undesirable or unexpected situation. These non-adaptable events must not be common and are interpreted as a bad design of the Smart Resource or a non-realistic quality requirements set by the client.

Although the service quality adaptation mechanism offers an optimum management of Smart Resource capabilities, the context configuration turns out to be a critical step for increasing the system efficiency. Robots can be designed to perform in a wide range of environments and contexts. More specifically, the mobile robots design usually faces different contexts and dynamic environments and situations. According to this the performance of the required services should be adapted to the context in which the robot is developing its tasks. For

this reason Smart Resources can be configured to manage different kind of information according to the needs of the robot, and also modify the quality requirements by changing the System profile.

3.2 ROS integration

The integration between Smart Resources and ROS systems aims to adapt the service management and supply to be easily accessed through a ROS API turning out as a ROS-Ready device. This is achieved in the communication layer. ROS communications are implemented as a Publish/Subscriber topology, the same used by the communication middleware on the Smart Resources. Therefore an integration layer adapts the service topic information to be managed as ROS topics. Service information is offered into a data structure understandable for ROS nodes. All the alarm and events mechanisms are also be adapted to manage this data structures. As a result the information off the services can be addressed as any other information provided by a regular ROS node.

4. Experiments: Distributed Smart Resource for Turtlebot robot

4.1 Turtlebot robot as a smart resources

In order to check the integration of the Smart Resources, it is presented a use case in which a ROS-based robot will switch from a centralized design to a decoupled paradigm by implementing the Smart Resource devices. The chosen platform in this use case is the Turtlebot II which can be considered as one of the most spreader ROS-ready robotic platforms. The typical hardware and software configuration is depicted in Fig. 2.

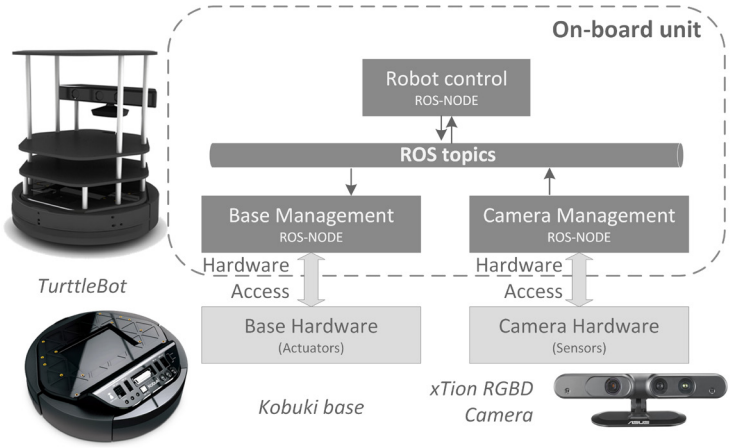


Figure 2: Turtlebot base system used in the experiments

The Turtlebot robot is originally endowed with an on-board RGBD camera (Asus Xtion or Microsoft Kinect) to sense the environment and a Kobuki mobile base for being able to perform a displacement around it. Both devices must be connected to on-board processing unit in order to allow the Turtlebot to manage them. To consider the ROS-based Turtlebot robot as a smart resources base and sensor must be considered different devices. Additionally, it is necessary to include the list of provided services, configuration values, and quality measures of both Smart Resources (Fig. 3).

A sensor or actuator management task involves a certain computational cost. Although some of this tasks reflect small costs, some others can be set as one of the most resource consuming tasks. One clear example is the data acquisition, management and interpretation of the RGB-D sensor data. As far as more sensors or actuators are added more computation power will be assigned to deal with them. Consequently the computational power available for other control tasks is reduced.

In order to save the on-board processing unit to be used only for robot behaviour and control tasks the sensor and actuator management are distributed. For this reason in these uses case the RGBD camera and the mobile base are decoupled by being implemented as a ROS-ready Smart Resource just as described along this paper. As a result the actuation and sensor services, in addition to the usual Smart Resource interface, are accessible

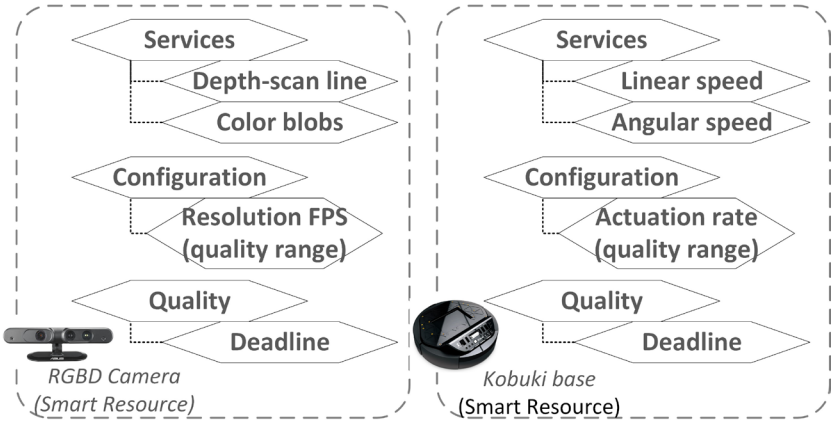


Figure 3: Interfaces provided as a smart resources

through ROS topics. In this use case it is tested the versatility of this implementation by replacing the physical camera and mobile base for simulated ones. Therefore it is described how the robot behaviour node implemented in the on-board processing unit can perform in the same way since the Smart Resources can provide the same type of service. The physical decoupled integration, as showed on Fig 4, makes use of an Asus Xtion RGBD and a Kobouki base connected to their respective BeagleBone Black board as a processing unit characterized as a ROS-ready Smart Resource.

The simulated version summarized on Fig?? make use of a virtual model of the Kobuki and the Asus Xtion camera disposed on the Gazebo (Koenig and Howard, 2004) simulator. The processing units can be also simulated as different virtual nodes that provide the Smart Resource interface.

In Fig 6 is displayed the graphical result of the colour service provided by the Xtion Smart Resource in the physical and the simulated implementation. As can be checked, the on-board processing unit node is performing in the same way in both versions as far as the provided services are performing in the way that is expected by offering the required read/write ROS topics.

5. Conclusions

Since the implementation of ROS into robotic researches and development is increasing everyday the definition of a ROS ready Smart Resource offers a high level of versatility. Adding these resources into the ROS topology has been proved to provide a standardized way to implement isolated procedures, which can be defined as abstract services that manage high-level information. Smart Resources also offer a high configuration capabilities and performance monitoring. As a result we obtain a decoupled system in which every resource provides a service adapted to perform in the most optimum way and can be accessed by means of ROS topics, being easily integrated in every ROS based platform and highly reusable. As a future work, it will be studied how to develop a ROS node to provide a high-level sensor fusion that relies on the information provided by the Smart Resources. For this goal it will make use of the alarms and reconfiguration mechanisms to increase the fusion reliabilit without increasing the computational load of the central unit.

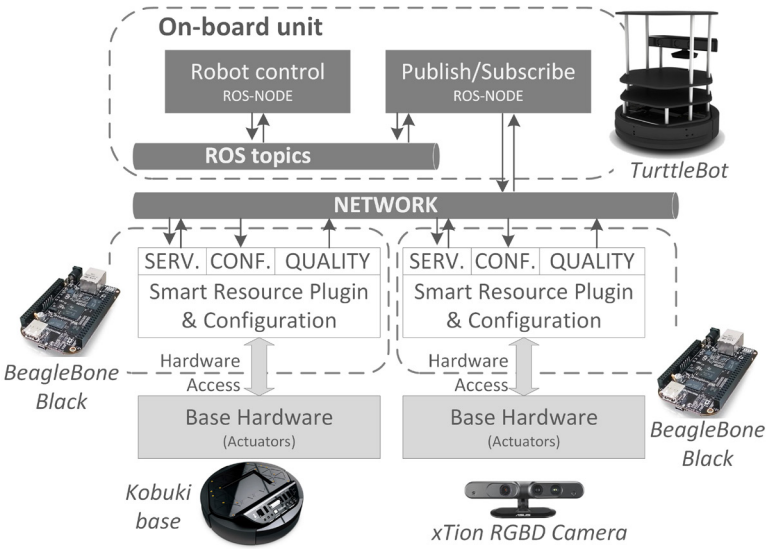


Figure 4: Turtlebot HW and SW decoupled setup using sensorial Smart resource

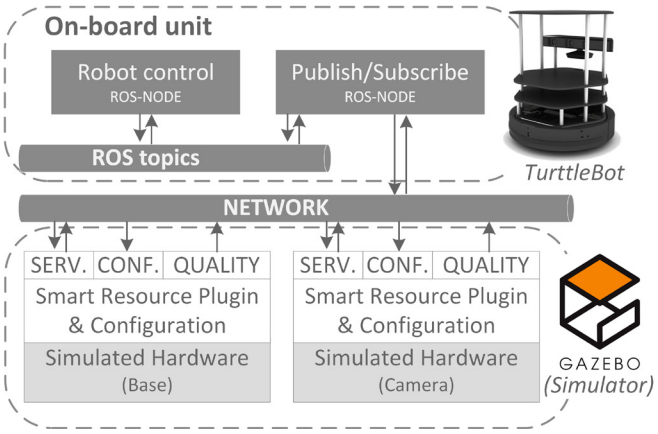
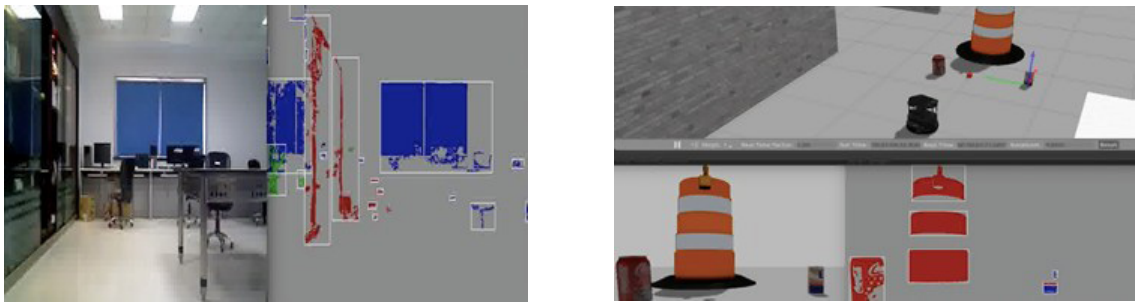


Figure 5: Simulated version used in the experiments



(a) Blob color service real. *(b) Simulated.*

Figure 6: Xtion Smart Resource implementation

6. Acknowledgements

Work supported by the Spanish Science and Innovation Ministry MICINN: CICYT project M2C2: «Codiseño de sistemas de control con criticidad mixta basado en misiones» TIN2014-56158-C4-4-P and PAID (Polytechnic University of Valencia): UPV-PAID-FPI-2013.

7. References

Chen, H., Cheng, H., Zhang, B., Wang, J., Fuhlbrigge, T., and Liu, J., 2013. Semiautonomous industrial mobile manipulation for industrial applications. In *Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2013 IEEE 3rd Annual International Conference on*, pages 361-366. IEEE.

Cousins, S., 2011. Exponential growth of ros [ros topics]. *IEEE Robotics & Automation Magazine*, 1 (18):19-20. Cousins, S., Gerkey, B., Conley, K., and Garage, W., 2010. Sharing software with ros [ros topics]. *Robotics & Automation Magazine, IEEE*, 17 (2): 12-14.

Crick, C., Jay, G., Osentoski, S., Pitzer, B., and Jenkins, O. C., 2011. Rosbridge: Ros for non-ros users. In *Proceedings of the 15th International Symposium on Robotics Research*.

Elkady, A. and Sobh, T., 2012. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, 2012.

Estefó, P., Campusano, M., Fabresse, L., Fabry, J., Laval, J., and Bouraqad, N., 2014. Towards Live Programming in ROS with PhaROS and LRP. *arXiv preprint arXiv:1412.4629*.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M., 2003. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35 (2):114-131.

Koenig, N. and Howard, A., 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149-2154. IEEE.

Liu, F., Narayanan, A., and Bai, Q., 2000. Real-time systems.

Meijer, G. C., Meijer, C. M., and Meijer, C. M., 2008. *Smart sensor systems*. Wiley Online Library.

Munera, E., Poza-Lujan, J.-L., Posadas-Yagüe, J.-L., Simó-Ten, J.-E., and Noguera, J. F. B., 2015. Dynamic Reconfiguration of a RGBD Sensor Based on QoS and QoC Requirements in Distributed Systems. *Sensors*, 15 (8):18080-18101.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y., 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5.

Remy, S. L. and Blake, M. B., 2011. Distributed service-oriented robotics. *Internet Computing, IEEE*, 15 (2):70-74.

Saraydaryan, J., Jumel, F., and Guenard, A., 2014. ASTRO: Architecture of services toward robotic objects. *International Journal of Computer Science Issues (IJCSI)*, 11 (4):1.