Samir Vedantham
Whit Smit, Mick West
Underwater Navigation and Communications
A Comprehensive Review of Robot Middleware Software in Use Today

1. Introduction

Robot Operating System, or ROS, is a software abstraction layer that sits above a robot's OS and below the high-level software application layer. Software developers face unique challenges in testing software/hardware compatibility, dealing with hardware, and maintaining software quality when dealing with robots [1]. ROS is an open-source attempt to mitigate these difficulties and to create more cohesiveness in research within the robotics community [2]. This paper outlines the underlying technology and applications of ROS.

2. Overview of ROS

2.1 Communications and Messaging

Because robots often carry embedded computers, each responsible for a section of the robot's overall performance, effective communication between these systems requires a middleware capable of distributed computing. ROS, at its core, provides messaging capability in a distributed environment. This includes anonymous message passing, message playback, and remote procedure calls (RPC) between computers [3]. These three features would allow an embedded system to capture data from a sensor, send that data to another computer, and call a procedure from a third computer to complete a certain task. Without ROS, software developers would have to program specific hardware-level procedure calls to manage the movement of data [1]. ROS allows a software developer to use high-level languages to accomplish these routine tasks. ROS also supports serialization/deserialization of messages to increase security within the communication system [3].

2.2 Inner-Workings of ROS

ROS middleware has two major functions and three major components. It can be customized to each robotics project as it contains a suite of tools and packages that enhance it. The first major function is to take in data from sensors, devices, and other hardware on the robot. This data does not have to be of the same type, size, or format. The data collection component of ROS handles this data and outputs it to various services within ROS that require it [4]. These services, the second component of ROS, may include device handling, security, context awareness, etc. [4]. The second function of ROS is to interface with the application-level software. This is completed by the final component - a data provisioning layer

which aggregates the information generated by the services and packages them for in higher layers of the program [4].

2.3 Real-Time Processing

ROS provides capability for real-time processing through the RTROS extension. RTROS sits on top of the ROS middleware and below the application-level programs. This extension functions by adding a software layer on top of each component of ROS [5]. For example, the main ROS node has a corresponding RTROS node and the main operating system of the robot is given real-time capability through an OS extension called Xenomai [5]. Real-time processing of messages and communication is component-based and RESTful, allowing for software abstraction and prudent design. RTROS also supports RPC functionality similar to ROS without the Real-Time extension [5]. It is also important to note that RTROS is not as secure as pure ROS [1].

3. Commercial Applications

ROS is maintained by an open-source community which is rapidly increasing in size. Although this middleware is used mainly by research labs, industrial robotics companies are adopting ROS and ditching their custom middleware software packages at a fast rate [6]. This means that the number of contributors to ROS is growing and that the reach and efficiency of the software is increasing [6]. However, using open-source software does present unique challenges. Code may not follow uniform documentation or quality standards, and unique use-cases may not be implemented yet.

4. Technology

ROS is compatible and supported with Ubuntu and Debian distributions of Linux as well as Windows. All compatible hardware is listed on the ROS support website. ROS programs are commonly written in C++, but development languages such as Python and Java support most client libraries [2]. It is important to note that ROS is more than a library and cannot be solely define as an Integrated Development Environment (IDE). This middleware also includes a client server, a command-line interface, and a build system [2].

5. Conclusion

The middleware software layer in any robotics project is an essential element in the design of a robotic system. It is an intermediary between the operating system, the hardware, and high-level software. It simplifies the job of the software developer by allowing development time to be spent in design rather than verification and debugging. ROS is a widely used middleware that allows for remote communication between embedded devices in a robot without the hassle of dealing with complex OS procedure calls.

References

[1] A. Elkady and T. Sobh, "Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography," *Journal of Robotics*, vol. 2012, p. 15, Jan. 2012.

[2] J. O'Kane, "A Gentle Introduction to ROS," University of South Carolina, Columbia, SC, Apr. 20, 2016, p. 1-9, [Online]. Available: https://cse.sc.edu/~jokane/agitr/agitr-letter.pdf. Accessed: Oct. 24, 2016.

[3] "WHY ROS?," in *ROS.org*. [Online]. Available: http://www.ros.org/core-components/. Accessed: Oct. 24, 2016.

[4] X. Li, J.-F. Martínez, J. Rodríguez-Molina, and N. Martínez, "A survey on Intermediation architectures for underwater robotics," *Sensors*, vol. 16, no. 2, p. 190, Feb. 2016.

[5] J. Carstensen and A. Rauschenberger, "Real-Time Extension to the Robot Operating System", Seoul, South Korea, 2016.

[6] S. Bouchard, "Robot operating system making its way into industrial robotics," in *IEEE Spectrum*, IEEE Spectrum: Technology, Engineering, and Science News, 2011. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/robotics-software/robot-operating-system-making-its-way-into-industrial-robotics. Accessed: Oct. 24, 2016.