Robot Operating System (ROS)

MATTHEW LAWSON[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: laszewski@gmail.com*

paper2, April 30, 2017

The Open Source Robotics Foundation (OSRF) oversees the maintenance and development of the Robot Operating System (ROS). ROS provides an open-source, extensible framework upon which roboticists can build simple or highly complex operating programs for robots. Features to highlight include: a) ROS' well-developed, standardized intra-robot communication system; b) its sufficiently-large set of programming tools; c) its C++ and Python APIs; and, d) its extensive library of third-party packages to address a large proportion of roboticists software needs. The OSRF distributes ROS under the BSD-3 license.

Keywords: Cloud, I524, robot, ros, ROS

https://github.com/cloudmesh/sp17-i524/raw/master/paper2/S17-IO-3010/report.pdf

1. INTRODUCTION

The Open Source Robotics Foundation's middleware product *Robot Operating System*, or ROS, provides a framework for writing operating systems for robots. ROS offers "a collection of tools, libraries, and conventions [meant to] simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms" [2]. The Open Source Robotics Foundation, hereinafter OSRF or the Foundation, attempts to meet the aforementioned objective by implementing ROS as a modular system. That is, ROS offers a core set of features, such as inter-process communication, that work with or without pre-existing, self-contained components for other tasks.

2. ARCHITECTURE

The OSRF designed ROS as a distributed, modular system. The OSRF maintains a subset of essential features for ROS, i.e., the core functions upon which higher-level packages build, to provide an extensible platform for other roboticists. The Foundation also coordinates the maintenance and distribution of a vast array of ROS add-ons, referred to as modules. Figure 1 illustrates the ROS universe in three parts: a) the plumbing, ROS' communications infrastructure; b) the tools, such as ROS' visualization capabilities or its hardware drivers; and c) ROS' ecosystem, which represents ROS' core developers and maintainers, its contributors and its user base.

The modules or packages, which are analogous to packages in Linux repositories or libraries in other software distributions such as *R*, provide solutions for numerous robot-related challenges. General categories include a) drivers, such as sensor and actuator interfaces; b) platforms, for steering and image processing, etc.; c) algorithms, for task planning and obstacle avoidance; and, d) user interfaces, such as tele-operation and sensor data display. [3]

2.1. Communications Infrastructure

2.1.1. General

OSRF maintains three distinct communication methods for ROS: a) *message passing*; b) *services*; and, c) *actions*. Each method utilizes ROS' standard communication type, the *message* [4]. Messages, in turn, adhere to ROS' *interface description language*, or IDL. The IDL dictates that messages should be in the form of a data structure comprised of typed fields [5]. Finally, *.msg* files store the structure of messages published by various nodes so that ROS' internal systems can generate source code automatically.

2.1.2. Message Passing

ROS implements a publish-subscribe anonymous message passing system for inter-process communication, hereinafter pubsub, as its most-basic solution for roboticists. A pubsub system consists of two complementary pieces: a) a device, node or process, hereinafter node, publishing messages, i.e., information, to a *topic*; and b) another node *listening to* and ingesting the information from the associated topic. Designating topics to which a node should subscribe and topics to which a node should publish falls to the roboticist. ROS' *rosnode* command line tool conveniently "display[s] a list of active topics, the publishers and subscribers of a specific topic, the publishing rate of a topic, the bandwidth of a topic, and messages published to a topic" [6].

Pubsub's method of operation analogizes to terrestrial radio. In the analogy, the radio station represents the publishing node, the radio receiver maps to the subscribing node and the frequency on which one transmits and the other receives repre-

Fig. 1. A Conceptualization of What ROS, the *R*obot *O*perating *S*ystem, Offers to Roboticists [1]

sents the topic. Unlike terrestrial radio, though, ROS provides a lookup mechanism versus "flipping through the dial."

The OSRF touts the pubsub communications paradigm as the ideal method primarily due to its anonymity and its requirement to communicate using its message format. With respect to the first point, the nodes involved in bilateral or multilateral conversations need only know the topic on which to publish or subscribe in order to communicate. As a result, nodes can be replaced, substituted or upgraded without changing a single line of code or reconfiguring the software in any manner. The subscriber node can even be deleted entirely without affecting any aspect of the robot except those nodes that depend on the deleted node.

In addition, ROS' pubsub requires well-defined interfaces between nodes in order to succeed. For instance, if a node publishes a message without a crucial piece information a subscribing node requires or in an unexpected format, the message would be useless. Alternatively, it would be pointless for an audio processing node to subscribe to a node publishing lidar data. Therefore, a message's structure must be well-defined and available for reference as needed in order to ensure compatibility between publisher and subscriber nodes. As a result, ROS has a modular communication system. That is, a subscriber node may use all or only parts of a publishing node's message. Further, the subscribing node can combine the data with information from another node before publishing the combined information to a different topic altogether for a third node's use. At the same time, a fourth and fifth node could subscribe to the original topic for each node's respective purpose.

Finally, ROS' pubsub can natively replay messages by saving them as files. Since a subscriber node processes messages received irrespective of the message's source, publishing a saved message from a subscriber node at a later time works just as well as an actual topic feed. One use of asynchronous messaging: postmortem analysis and debugging.

2.1.3. Services

ROS also provides a synchronous, real-time communication tool under the moniker *services* [7]. Services allow a subscribing node to request information from a publishing node instead of passively receiving whatever the publishing node broadcasts whenever it broadcasts it. A service consists of two messages, the request and the reply. It otherwise mirrors ROS' message passing function. Finally, users can establish a continuous connection between nodes at the expense of service provider flexibility.

2.1.4. Actions

ROS *actions* offer a more-advanced communication paradigm than either message passing or services [8]. Actions, which use the basic message structure from message passing, allow roboticists to create a request to accomplish some task, receive progress reports about the task completion process, receive task completion notifications and / or cancel the task request. For example, the roboticist may create a task, or equivalently, initiate an action, for the robot to conduct a laser scan of the area. The request would include the scan parameters, such as minimum scan angle, maximum scan angle and scan speed. During the process, the node conducting the scan will regularly report back its progress, perhaps as a value representing the percent of the scan completed, before returning the results of the scan, which should be a point cloud. Roboticists can program a ROS-driven robot to attempt to accomplish any task imaginable, subject to physical realities. Sample actions: a) move X meters left; b) detect the handle of a door; and / or c) locate an empty soda can on a table, pick it up to determine if it is empty enough to recycle, crush it if it is empty enough, identify the recyle bin and then place it in the recylce bin [9].

2.2. Tools

2.2.1. Message Standards

ROS' extensive use in the robotics realm has allowed it to create message standards for various robot components [4]. In this case, *message standards* refers to expectations regarding information and information types robot components will provide to subscribing nodes. For instance, standard message definitions exist "for geometric concepts like poses, transforms, and vectors; for sensors like cameras, IMUs and lasers; and for navigation data like odometry, paths, and maps; among many others." These standards facilitate interoperability amongst robot components as well as easing development efforts by roboticists.

2.2.2. Robot Geometry Library

Robots with independently movable components, such as appendages (with joints) or movable sensors, must be able to coordinate such movements in order to be usable. Maintaining an accurate record of where a movable component is in relation to the rest of the robot presents a significant challenge in robotics [4].

ROS addresses this issue with its *transform* library. The *tf* library tracks components of a robot using three-dimensional coordinate frames [10]. It records the relationship between co-

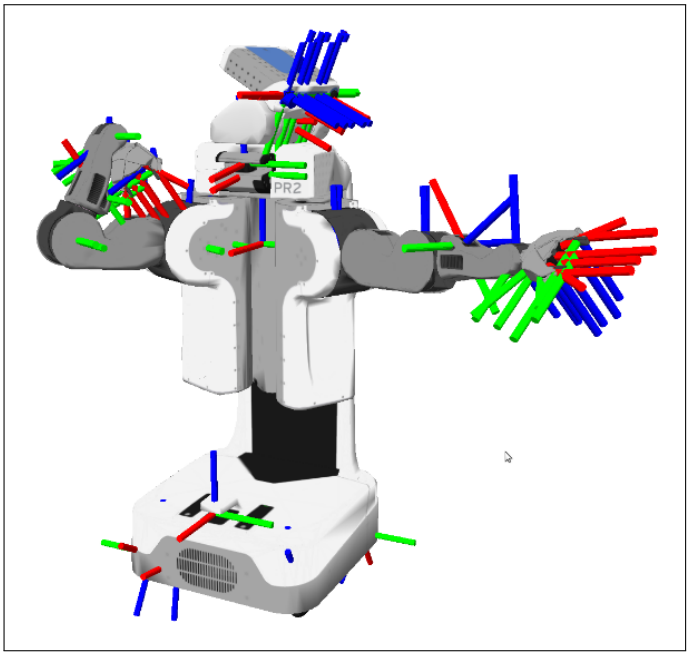

Fig. 2. A Simulated Robot with Many Coordinate Frames [10]

ordinate frame positional values at sequential points in time in a tree structure. tf's built-in functions allow the roboticist to transform a particular coordinate frame's values to same basis as a different coordinate frame's values. As a result, the user, or the user's program, can always calculate any coordinate frame's relative position to any or all of the other coordinate frame positions at any point in time. Although the first-generation library, *tf*, has been deprecated in favor of the second-generation one, *tf2*, the Foundation and ROS users still refer to the library as tf.

2.2.3. Robot Description Language

ROS describes robots in a machine-readable format using its *Unified Robot Description Format*, or URDF [4]. The file delineates the physical properties of the robot in XML format. URDF files enable use of the tf library, useful visualizations of the robot and the use of the robot in simulations.

2.2.4. Diagnostics

ROS' diagnostics meta-package, i.e., a package of related packages, "contains tools for collecting, publishing, analyzing and viewing diagnostics data [11]." ROS' diagnostics take advantage of the aforementioned message system to allow nodes to publish diagnostic information to the standard diagnostic topic. The nodes use the diagnostic_updater and self_test packages to publish diagnostic information, while users can access the information using the rqt_robot_monitor package. ROS does not require nodes to include certain information in their respective publications, but diagnostic publications generally provide some standard, basic information. That information may include serial numbers, software versions, unique incident IDs, etc.

2.2.5. Command Line Interfaces (CLI)

ROS provides at least 45 command line tools to the roboticist [12]. Therefore, ROS can be setup and run entirely from the command line. However, the GUI interfaces remain more popular among the user-base. Examples of ROS CLI tools include: a) rosmsg , which allows the user to examine messages, including the data structure of .msg files [5]; b) rosbag , a tool to perform various operations on .bag files, i.e., saved node publications; and, c) rosbash , which extends bash, a Linux shell program, with ROS-related commands.

2.2.6. Graphical User Interfaces (GUI)

OSRF includes two commonly-used GUIs, *rviz* and *rqt* [4], in the core ROS distribution. rviz creates 3D visualizations of the robot, as well as the sensors and sensor data specified by the user. This component renders the robot in 3D based on a user-supplied URDF document. If the end-user wants or needs a different GUI, s/he can use rqt, a Qt-based GUI development framework. It offers plug-ins for items such as: a) viewing layouts, like tabbed or split-screens; b) network graphing capabilities to visualize the robot's nodes; c) charting capabilities for numeric values; d) data logging displays; and, e) topic (communication) monitoring.

2.3. Ecosystem

ROS benefits from a wide-ranging network of interested parties, including core developers, package contributors, hobbyists, researchers and for-profit ventures. Although quantifiable use metrics for ROS remain scarce, ROS does have more than 3,000 software packages available from its community of users [13], ranging from proof-of-concept algorithms to industrial-quality software drivers. Corporate users include large organizations such as Bosch (Robert Bosch GmbH) and BMW AG, as well as smaller companies such as ClearPath Robotics, Inc. and Stanley Innovation. University users include the Georgia Institute of Technology and the University of Arizona, among others [14].

3. API

ROS supports robust application program interfaces, APIs, through libraries for C++ and Python. It provides more-limited, and experimental, support for nodejs, Haskell and Mono / .NET programming languages, among others. The latter library opens up use with C# and Iron Python [15].

4. LICENSING

The OSRF distributes the core of ROS under the standard, three-clause BSD license, hereinafter BSD-3 license. The BSD-3 license belongs to a broader class of copyright licenses referred to as *permissive licenses* because it imposes zero restrictions on the software's redistribution as long as the redistribution maintains the license's copyright notices and warranty disclaimers [16].

Other names for BSD-3 include: a) BSD-new; b) New BSD; c) revised BSD; d) The BSD License, the official name used by the Open Source Initiative; and, e) Modified BSD License, used by the Free Software Foundation.

Although the OSRF distributes the main ROS elements under the BSD-3 license, it does not require package contributors or end-users to adopt the same license. As a result, full-fledged ROS programs may include other types of *Free and Open-Source Software* [17], or FOSS, licenses. In addition, programs may depend on proprietary or unpublished drivers unavailable to the broader community.

5. USE CASES

ROS' end-markets, its use cases, include manipulator robots, i.e., robotic arms with grasping units; mobile robots, such as autonomous, mobile platforms; autonomous cars; social robots; humanoid robots, unmanned / autonomous vehicles; and an assortment of other robots [14].

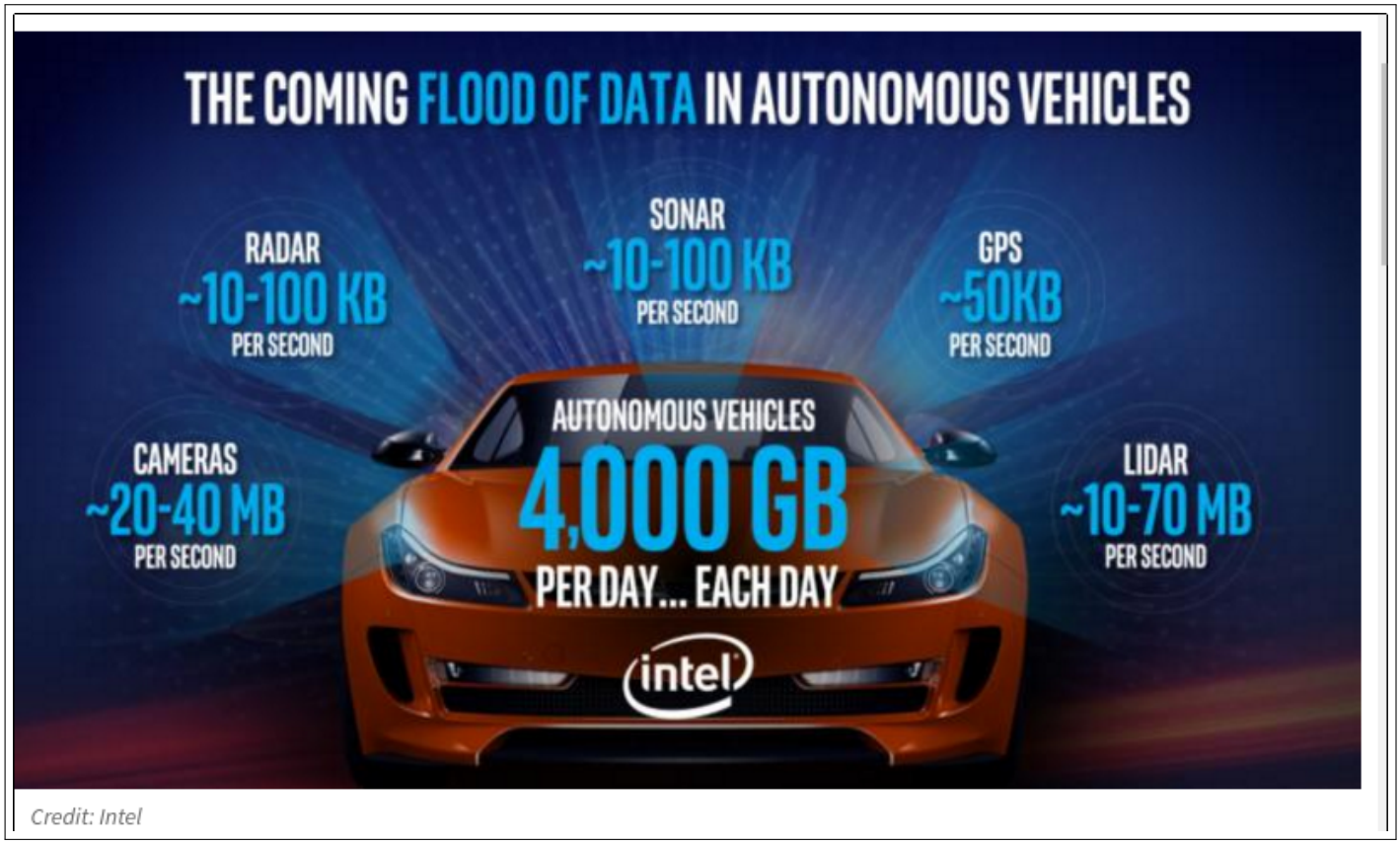

Fig. 3. Estimated Amount of Data Produced by an Autonomous Automobile [18]

5.1. Automated Data Collection Platform

Fetch Robotics, Inc. offers its *Automated Data Collection Platform* robot to the market so corporations can "[g]ather environmental data more frequently and more accurately for [its] Internet of Things...and Big Data Applications [19]." Fetch's system automatically collects data, such as RFID tracking (inventory management) or in-store shelf surveys. The latter service began in January 2017 when Fetch partnered with Trax Image Recognition, which makes image recognition software [20].

5.2. Autonomous Robots of Various Natures

The applications of autonomous robots abound, especially for monotonous or dangerous work. For instance, ClearPath Robotics, a company that sells autonomous robots using ROS, currently highlights the following use cases: a) "geotechnical mapping of rock masses, a crucial step in predicting potentially lethal rock falls and rock bursts in and around mines [21];" minesweeping [22]; and vibration control of bridges [23]. In order to accomplish any of the aforementioned tasks, or any of the other tasks, the robots need sensors, such as lidar, high-definition cameras and / or sonar. One estimate places lidar and camera data production at 10MB to 70MB per second and 20MB to 40MB per second [18]. Given the multitude of addressable end markets, as well as the probability of multiple robots with multiple sensors associated with one effort in any particular end market, the data produced by a ROS robot vaults it into *big data* territory. A single ROS instance may never create 4,000GB of data per day like an autonomous consumer automobile, but a small swarm of moderately complex robots run by a three-person team will likely prompt the team to avail itself of big data techniques.

6. CONCLUSION

ROS offers a number of attractive features to its users, including a well-developed and standardized intra-robot communication system, modular design, vetted third-party additions and legitimacy via real-world applications. Although its status as open source software precludes direct support from its parent organization, OSRF, for-profit organizations and the software's active community of users provide reassurances to any roboticist worried about encountering a seemingly-insurmountable problem.

7. ACKNOWLEDGMENT

I would like to thank my employer, Indiana Farm Bureau, for its support of my continuing education.

REFERENCES

[1] H. Boyer, "Open Source Robotics Foundation And The Robotics Fast Track," web page, nov 2015, accessed 19-mar-2017. [Online]. Available: https://www.osrfoundation.org/wordpress2/wp-content/uploads/2015/11/rft-boyer.pdf

[2] Open Source Robotics Foundation, "About ROS," Web page, mar 2017, accessed 16-mar-2017. [Online]. Available: http://www.ros.org/about-ros/

[3] National Instruments, "A Layered Approach to Designing Robot Software," Web page, mar 2017, accessed 18-mar-2017. [Online]. Available: http://www.ni.com/white-paper/13929/en/

[4] Open Source Robotics Foundation, "Why ROS?: Features: Core components," Web page, mar 2017, accessed 17-mar-2017. [Online]. Available: http://www.ros.org/core-components/
[5] Open Source Robotics Foundation, "ROS graph concepts: Messages," Web page, aug 2016, accessed 18-mar-2017. [Online]. Available: http://wiki.ros.org/Messages
[6] Open Source Robotics Foundation, "rostopic: Package Summary," Web page, jun 2016, accessed 09-apr-2017. [Online]. Available: http://wiki.ros.org/rostopic
[7] Open Source Robotics Foundation, "Services," web page, feb 2012, accessed 18-mar-2017. [Online]. Available: http://wiki.ros.org/Services
[8] Open Source Robotics Foundation, "actionlib: Package summary," Web page, feb 2017, accessed 18-mar-2017. [Online]. Available: http://wiki.ros.org/actionlib
[9] MathWorks, Inc. The, "Control PR2 Arm Movements Using ROS Actions and Inverse Kinematics," Web page, apr 2017, accessed 09-apr-2017. [Online]. Available: https://goo.gl/3tZqow
[10] Open Source Robotics Foundation, "tf2: Package summary," Web page, mar 2016, accessed 18-mar-2017. [Online]. Available: http://wiki.ros.org/tf2
[11] Open Source Robotics Foundation, "diagnostics: Package Summary," Web page, aug 2015, accessed 19-mar-2017. [Online]. Available: http://wiki.ros.org/diagnostics
[12] Open Source Robotics Foundation, "ROS command-line Tools," Web page, aug 2015, accessed 19-mar-2017. [Online]. Available: http://wiki.ros.org/ROS/CommandLineTools
[13] Open Source Robotics Foundation, "Why ROS?: Is ROS for me?" Web page, mar 2017, accessed 16-mar-2017. [Online]. Available: www.ros.org/is-ros-for-me/
[14] Open Source Robotics Foundation, "Robots Using ROS," Web page, mar 2017, accessed 19-mar-2017. [Online]. Available: http://wiki.ros.org/Robots
[15] Open Source Research Foundation, "APIs," Web page, mar 2016, accessed 19-mar-2017. [Online]. Available: http://wiki.ros.org/APIs
[16] Wikipedia, "BSD licenses — Wikpedia, the free encyclopedia," Web page, mar 2017, accessed 16-mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/BSD_licenses
[17] Wikipedia, "Free and open-source software – Wikipedia, the free encyclopedia," Web page, mar 2017, accessed 18-mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/Free_and_open-source_software
[18] P. Nelson, "Just one autonomous car will use 4,000 GB of data/day," Web page, dec 2016, accessed 09-apr-2017. [Online]. Available: https://goo.gl/HBL9GM
[19] Fetch Robotics, Inc., "Automated Data Collection Platform," Web page, mar 2017, accessed 19-mar-2017. [Online]. Available: http://fetchrobotics.com/automated-data-collection-platform/
[20] Retail Touch Points, "Trax and Fetch Robotics Unite for Better Retail," Web page, jan 2017, accessed 19-jan-2017. [Online]. Available: http://www.retailtouchpoints.com/features/news-briefs/trax-and-fetch-robotics-unite-for-better-retail-insights
[21] Clearpath Robotics Inc., "Husky performs slam on stereonets to help predict rock falls and rock bursts," Web page, apr 2017, accessed 09-apr-2017. [Online]. Available: https://www.clearpathrobotics.com/husky-queens-mining-slam-stereonets/
[22] Clearpath Robotics Inc., "Bomb-detecting husky to remove killer landmines," Web page, apr 2017, accessed 09-apr-2017. [Online]. Available: https://www.clearpathrobotics.com/coimbra-autonomous-demining-husky/
[23] Clearpath Robotics Inc., "Deployable, autonomous vibration control of bridges using husky ugv," Web page, apr 2017, accessed 09-apr-2017. [Online]. Available: https://www.clearpathrobotics.com/autonomous-vibration-control-bridges-using-husky-ugv/

AUTHOR BIOGRAPHIES

Matthew Lawson received his BSBA, Finance in 1999 from the University of Tennessee, Knoxville. His research interests include data analysis, visualization and behavioral finance.

A. WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Matthew Lawson. Matthew researched and wrote all of the material for this paper.