ROS redesigned

Back in 2007 a robotics middleware system got introduced to the world in an attempt to set a baseline for Academia to harmonize and enable a more uniform way of developing their robotic systems. While this framework is named a robotic operating system "(ROS)" it is not an operating system in the traditional meaning of this concept, instead it operates as a middleware on top of a traditional operating system such as Linux.

A middleware framework provides services to an application, which aren't available within the underlying operating system. In this case ROS provides a modular processing network, that consists of a Graph of Nodes with each Node representing a single process running within the application. Due to ROS' modular nature allowing the same base core functionality and code to be applied within different kinds of end-goal applications, ROS quickly became a popular open-source framework.

Currently ROS is widely used for developing robotic applications, and quite commonly applied within robotic designs by developers and researchers in industry and academia alike [1]. It has even found its way into several art installations, most notably within an art installation by Diller Scofidio and Renfro, in collaboration with David Lang and Jody Elff referred to as "musings in a glass box". Where four robotic buckets powered by ROS, navigated throughout the room filled with actuated water dripping systems and custom computer-vision boxes for navigation and tracking [3].

As the understanding of robotics and their respective applications grows, so does an improved perception of all possible implementation horizons. With this there is the growing desire to enhance the current understanding of machine sensing, reasoning, safety, efficiency, and a more dynamic collaboration between humans and machines. The robotic systems adopting these concepts grow increasingly complex and become quite demanding especially within a pursuit in expanding their autonomous abilities in the forms of autonomous vehicles, humanoid robots and Industrial equipment. Due to these advancements the framework underlying ROS which (especially within the academic domain) makes assumptions based on an ideal world, as well as several other architectural shortcomings are holding back progress in industrial robotic development.

Need for Architectural Revolution

ROS in its current state has essentially reached a state of maturity within the limitations of the initial boundaries set within the original design of this framework. Given the intrusive nature of the changes that needed to be made in order to bring this framework to the next level it was decided to develop ROS2 as a parallel to the original ROS in order to keep both working properly for their respective use cases[6].

ROS was developed with the use case for the PR2 robot developed by Willows Garage in mind. Its primary goal was to provide a software toolset that would support novel research and development projects with the PR2 and simultaneously let ROS be useful for other robots, therefore abstraction played a heavy role in its design [2].

However, since it was still based on the PR2 use case the core characteristics of the framework respond to;

- single robots,
- on board computational resources,
- no real-time requirements or support of any kind,

- excellent network connectivity,
- a maximum flexibility of the systems and,
- most importantly the assumptions of an ideal world as is often the case within academic research.

While ROS remains a popular middleware system still widely available and applied within several robotic designs the limitations of the PR2 use case motivated the development of a new framework, ROS2.

ROS' current popularity remains due to the maturity of the ROS framework and its wide range of packages supporting several applications and relatively easier learning curve compared to ROS2. Thanks to this maturity ROS has been the de facto version, when there is a need for quick prototyping of a proof of concept. The widespread packages/modules and detailed documentation available for ROS provide essential domain and application specific support. When it's essential for a prototype to operate as close to real-time as possible, the original ROS framework is inadequate. With this in mind, ROS2 should be implemented from the ground up. Furthermore, due to ROS' centralized communication it's still quite relevant when there is no cooperation between several robots (and humans) required and no need for dynamic real-time response to changing environments.

Motivated by the limitations of the original ROS framework, ROS2 development started in 2015, with its original target release schedule to be in 2017. ROS2, is essentially a complete refactoring of the original framework through modernizations and improvements inspired by the needs of Industry. These needs are increasingly demanding due to the evolutionary development pace of the surrounding technologies (computing, sensors, software technologies, machine learning) implemented during the development of robotic applications. Within industrial robotic applications the real-time capabilities of the system are a critical aspect as production environments become more dynamic and flexible. Therefore the general design and architectural decisions made during the development of this new framework have been heavily influenced by this need for better real-time operation support.

Two key elements for achieving better real-time support are
- a decentralisation of nodes and,
- a flexible "Quality of Service" messaging system[2].

Currently ROS2 is still in its early (infant) stages. However the development and expansions of the ROS2 environment is backed by a very active community willing to bring ROS2 to a level where it can be used in real-world professional applications and it has great potential for the future[4].

The original ROS framework still has an active community development backing it towards the next and presumed last release (Noetic Ninjemys) expected to be released in May of 2020 with an estimated EOL (end of life) to be in May 2025 [6]. Simultaneously the development efforts behind ROS2 are growing quickly. The scientific community behind grows and industries are getting more and more interested in its potential within industrial applications [4], with the next ROS2 release (Foxy Fitzroy) scheduled for May of 2020.

Significant ROS2 use case considerations upon which new architectural design choices have been made are;
- no longer a centralized source of communication relying on a single master node instead the communication has been decentralized completely,
- the potential for swarm robotics, to be a platform for small embedded systems enabling them to become first-class participants,

- supporting real-time communication through inter-process and intermachine communications, allowing for non-ideal network connectivities,
- support evolvement towards real-world production environments implementing design patterns for building and,
- structuring systems such as life cycle management and static configuration for deployment [2].

If the ROS2 development is a success, the features above significantly improve the future potential of the ROS2 platform to become a more viable option for industrial applications. However, there are still some areas that require further development [7].

While it is possible to run ROS2 in a Windows, OS X and Linux environment, possibly due to the infant stage of this system there is a limited amount of public research available on the scheduling capabilities of ROS2. (Compared to extensively researched operating systems such as Linux. Since ROS2 operates as a middleware layer and is therefore not a proper operating system, there are effects on an application's runtime behavior which could be substantial and potentially rival the underlying operating system.

The real-time capabilities of this system are made an even more challenging task due to the tendency of ROS2 to multiplex independent message handlers within shared threads by using custom scheduling policies. This results in complex interdependencies on the timing of the applications running on top of the ROS2 middleware layer together with its underlying operating system. Realistically, automated end-to-end response-time analysis is therefore required to safely employ ROS2 in time-critical situations[5].

Furthermore, the implementation of the data distribution service “(DDS)” as defined by the Object Management Group “(OMG)”, is a significant change in the ROS2 framework compared to the old ROS design. This implementation of a decentralized communication network middleware based on the DDS standard, provides control over different “Quality of Service” options. Supporting the distributed discovery, serialization and transportation of data, which is implemented by a range of vendors. At the time of writing there are non-commercial DDS options available such as RTI’s non-commercial Connext version or ADLINK’s OpenSlice RTPS THere are also other implementations that do not fulfill the full DDS API, but provide enough functionally for ROS2, such as eProsima Fast RTPS [8].

Various implementations of a DDS middleware solution provide compatible enablers and possible improvements on the real-time capabilities and implementations within the ROS2 framework. The palette of use cases and ROS2 applications domains is so broad, that there are still often domain/purpose specific security and performance considerations that need to be made in order to balance the integration of ROS2 with the DDS system chosen. In order to support industrial integration and adoption of ROS2 there are world-wide ongoing research efforts made by academia and industry alike, expanding our knowledge in these specific areas.

Ongoing ROS 2 research efforts

Several academic institutes together with industrial partners who have shown interest in the potential of ROS2 and have started joining their efforts together in work groups in order to define a potential road forward.

Relatively recent research by Maruyama et. al. explores the performance of ROS2 as it was at the time of writing. They conducted a preliminary proof of concept for the DDS approach within ROS2 focused on evaluation of its performance within the system [7]. Stating that, while, ROS2, has the potential to run on embedded systems, and in addition, when utilizing the capabilities of DDS there is potential for real-time operations. Additionally, ROS2 is designed specifically for overcoming these real-time challenges, while inheriting (and simultaneously improving) the initial beneficial qualities of ROS. Moreover they state that as ROS2 is still in its early stages and under heavy development focus should be brought to maximizing the DDS potential by tuning and abstracting more QoS Policies for real-time processing and DDS configurations [7].

While it can be said that ROS2 definitely holds potentially significant capabilities for industrial real-world applications, research by Kim et. al. from Cornell University published in 2018 suggested that the DDS implementation is still lacking in certain security and safety aspects often required for real-world applications. It should not be assumed that the implementation of the DDS is accurate according to the given specifications since upon analysis of the default DDS middleware in ROS2 it was found that it did not conform to the security specifications by OMG [11].

Another study focused on the analysis and verification of DDS in ROS2 by Liu et. al. [10] from the university of Beijing and originally published in 2018 during the 16th ACM/IEEE International conference on Formal Methods and Models for System Design. This study concludes that the system within its current state is deadlock-free and meets the liveness requirements, and the high-priority nodes' messages can be preferentially transmitted and processed. However, while this research does provide a decent basis it itself also states that more pioneering work and in-depth research into the intricate inner workings of ROS2 is necessary to provide a better picture of its potential [10].

Furthermore, as is suggested by Daniel et. al. (2019) in their response-TIme analysis of ROS2 processing Chains under reservations-Based Scheduling, suggests there's a lack of knowledge on the timing effects of *mode changes* and the potential overhead of several DDS implementations. Therefore, while there is clearly an active research interest both within the academic and industrial domains, in this area of response-Time behaviour within ROS2 more research is required.

Here in Finland we have the Forum for Intelligent Machines (FIMA) in which Vaisto Solutions together with other academic and industry partners play a pivotal role. Vaisto Solutions is passionate about staying on the frontlines of new emerging technologies. Especially those that -from an Industrial perspective- appear to have a significant impact in prospective developments. As Vaisto Solutions is following ROS2 development closely, current versions are already applied within prototyping autonomous work cycle concepts. For the autonomous control systems Vaisto is developing real-time dataflows are a critical aspect of its successful implementations.

So follow this space as we'll be covering exciting ROS2 related topics such as; the real-time functionalities in ROS2, the DDS advantages and drawbacks based on its current ROS2 implementation and the LifeCycle management within ROS2 are under closer evaluation.

By :: Aimée de Koning, Embedded systems and ROS specialist at Vaisto Solutions

Sources ::

[1] Robots using ROS. URL:http://robots.ros.org
[2] B. Gerkey. Why ROS 2.0? URL:http://design.ros2.org/articles/why_ros2.html.

[3] Musings in a glass box https://www.galleriesnow.net/shows/musings-glass-box/
[4] ROS metrics http://download.ros.org/downloads/metrics/metrics-report-2019-07.pdf
[5] Response-Time Analysis of ROS 2 Processing Chains Under Reservation-Based Scheduling
https://drops.dagstuhl.de/opus/volltexte/2019/10743/
[6] ROS distrubutions http://wiki.ros.org/Distributions
[7] Exploring the performance of ROS2 https://www.researchgate.net/publication/309128426_Exploring_the_performance_of_ROS2
[8] ros2 and different dds/rtps vendors https://index.ros.org/doc/ros2/Concepts/DDS-and-ROS-middleware-implementations/
[9] FIMA and ROS workgroup https://www.fima.fi/events/ros-workgroup-kick-off/
[10] Y. Liu, Y. Guan, X. Li, R. Wang and J. Zhang, "Formal Analysis and Verification of DDS in ROS2," *2018 16th ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, Beijing, 2018, pp. 1-5.
Formal analysis and verification of DDS in ROS2 https://ieeexplore.ieee.org/abstract/document/8556970
[11] Security and Performance Considerations in ROS 2: A Balancing Act https://arxiv.org/abs/1809.09566