International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 7, Issue 10, pp 371-375, October-2023
https://doi.org/10.47001/IRJIET/2023.710050

Comparative Analysis of Robotic Operating Systems

[1]Prof. S.B.Bele, [2]Prasad Katyarmal, [3]Swaraj Gangane, [4]Komal Thakare, [5]Tasmeeya Sheikh

[1]Professor, Department of MCA, Vidya Bharati Mahavidyalaya, Amravati, India
[2,3,4,5]Student, Department of MCA, Vidya Bharati Mahavidyalaya, Amravati, India

***Abstract* - The Robot Operating System (ROS) comprises a collection of software libraries and tools utilized for constructing robotic systems, distinguished by its distributed and modular design. Within the context of ROS, task planning pertains to the arrangement of actions into a structured sequence aimed at achieving predefined objectives, all while striving to minimize associated costs, whether in terms of time or energy consumption. Task planning assumes critical importance when guiding the actions of a robotic agent, particularly in scenarios where a causal sequence could potentially lead the agent into a deadlock situation. Furthermore, task planning finds utility in less restrictive environments, contributing to the delivery of more intelligent and adaptive robotic behavior. This paper introduces the ROSPLAN framework, an architectural solution designed to seamlessly integrate task planning into ROS-based systems. It presents a comprehensive overview of this framework.**

Keywords: Robotic Operating Systems (ROS), Simulation, Unity3D, SLAM, Real-time Robotics Operating Systems, Robotics Development Platforms.

1. Introduction

Developing software for robots is a complex task due to the varying hardware of different robots and the overwhelming amount of code required. Robotics software architectures must facilitate large-scale software integration efforts. The integration of task planning and robotics presents several challenges, including generating an initial state that aligns with current environmental conditions, transforming actions planned by task planners into concrete actions, and developing and executing plans with a strategic approach that accommodates action failures, plan failures resulting from uncertainty or dynamic environmental changes, and evolving mission requirements.

This paper presents a framework that establishes a connection between generic task planning and an execution interface seamlessly provided by the Robot Operating System (ROS). This approach bridges two established standards: PDDL2.1, the planning domain description language encompassing temporal and numeric aspects, and the Robot Operating System (ROS). The primary objective is to create a modular architecture that readily accommodates various temporal planners, ensuring the effectiveness of plan execution frameworks like T-REX.

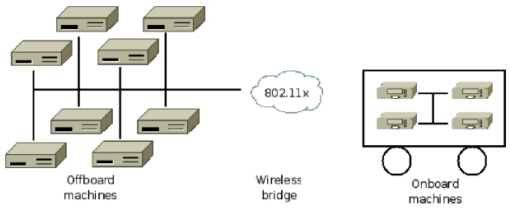


Figure 1: A typical ROS network configuration

ROS 1:

ROS 1 is a collection of libraries used to build various types of robots, including tools for monitoring processes, inspecting communications, and receiving time-series transformations. It has a thriving ecosystem of sensor, control, and algorithmic packages, enabling even small teams to build sophisticated robotics applications. However, ROS 1 struggles to deliver data consistently over lossy connections, has a single point of failure and lacks built-in security features. ROS 2 has been developed to address these issues but has faced numerous challenges due to architectural and engineering limitations. For example, the SROS project was introduced to enhance ROS 1's security but required consistent maintenance and further development to keep up with security trends.

Category	ROS 1	ROS 2
Network Transport	Bespoke protocol built on TCP/UDP	Existing standard (DDS), with abstraction supporting addition of others
Network Architecture	Central name server (`roscore`)	Peer-to-peer discovery
Platform Support	Linux	Linux, Windows, macOS
Client Libraries	Written independently in each language	Sharing a common underlying C library (`rcl`)
Node vs. Process	Single node per process	Multiple nodes per process
Threading Model	Callback queues and handlers	Swappable executor
Node State Management	None	Lifecycle nodes
Embedded Systems	Minimal experimental support (`rosserial`)	Commercially supported implementation (micro-ROS)
Parameter Access	Auxilliary protocol built on XMLRPC	Implemented using service calls
Parameter Types	Type inferred when assigned	Type declared and enforced

Figure 2: Summary of ROS 2 features compared to ROS 1

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 10, pp 371-375, October-2023

https://doi.org/10.47001/IRJIET/2023.710050

ROS 2:

ROS 2 is an open software platform that allows the development of robotic applications. Also known as Robotics Software Development Kit (SDK). ROS 2 is distributed under the Apache 2.0 License, which allows users to modify, implement, and redistribute the software without obligation to return it. ROS 2 is built on a government ecosystem that encourages partners to develop and publish their own software. Most plugin packages also use the Apache 2.0 license or a similar license. ROS 2 aims to increase mass adoption by making the code freely available, allowing users to use and distribute their applications without restrictions. 1. Middleware: This is also called ROS 2's pipeline. It manages communication between different components of the platform, from the network API to the language parser. 2. Algorithms: ROS 2 provides many algorithms frequently used in robotic applications. These include perception, SLAM, planning, etc. takes place. 3. Developer tools: ROS 2 has a command line and graphical tools that developers can use to configure, initialize, understand, visualize, debug, simulate, and log. There are also many tools for site management, design processes, and deployment.

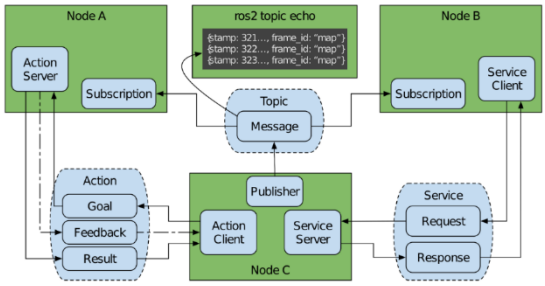


Figure 3: ROS 2 node interfaces: topics, services, and actions

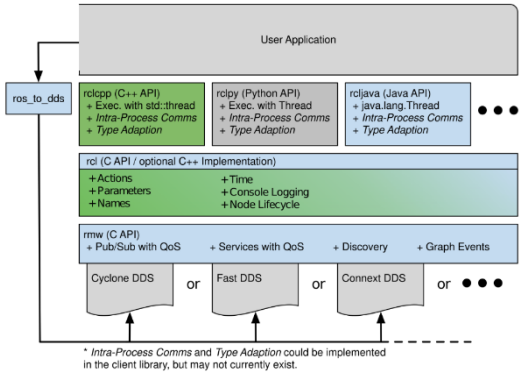


Figure 4: ROS 2 Client Library API Stack

2. Case studies

Five case studies were conducted to showcase the material acceleration provided by ROS 2. Each study presents a qualitative analysis of the impact of ROS 2 on the respective organization, based on interviews, customer experiences, and codebases analyzed during the study. The various use cases and scales demonstrate the significance of ROS 2 across the robotics sector.

A) Land: Ghost Robotics

Ghost Robotics, a Philadelphia-based company, specializes in quadruped robots for defense, enterprise, and research. These robots can navigate difficult environments like caves, mines, forests, and deserts. They have partnerships with the US military for base security and experimental applications. Ghost uses ROS 2 on its main computing platform, a Nvidia Jetson Xavier, for mission execution, high-level gait planning, terrain mapping, and localization. Their software architecture is heavily integrated with ROS 2, allowing for parallel development without disrupting other teams.

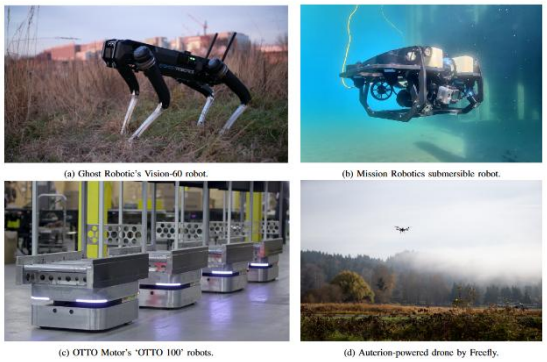


Figure 4: Case-study robot systems deployed on land, air, and sea

B) Sea: Mission Robotics

Mission Robotics, a San Francisco-based company, specializes in building marine robots for tasks such as structure inspection, environmental survey, salvage, and security. Their robots are designed to be flexible, allowing customers to customize their platform for their specific application. Mission's robots are equipped with sensors that gather data about the surface and underwater environment, allowing for more frequent, longer, and less risky underwater tasks. The mission uses ROS 2 as a common data bus to facilitate data streams and enable seamless integration of new hardware. The company's on-robot software is built on Cyclone and Connext DDS, which are more flexible and easier to customize. Mission Robotics uses ROS 2 as a common interface, allowing customers to create their own extensions and share a common infrastructure. For example, Mission collaborated with Aqua link to add depth sensing to an autonomous surface vessel using a Zed stereo camera, creating a starting point for developing new computer vision and autonomy capabilities for marine applications.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 10, pp 371-375, October-2023

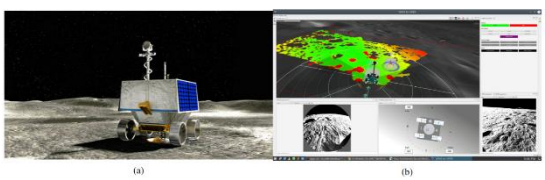
https://doi.org/10.47001/IRJIET/2023.710050

C) Large Scale: OTTO Motors

OTTO Motors, a Canadian company, offers autonomous robots for material handling services in warehouses and factories, replacing manually controlled equipment at scale. With thousands of robots deployed worldwide, OTTO operates fleets of over 100 robots in a single facility. Customers like Toyota and General Electric have adopted OTTO's technology. Initially developed on ROS 1, OTTO found it could not test more than 25 robots on the same shared network due to a custom multi-master system. To address this, they decided to use DDS, a flexible and efficient data exchange protocol widely used in robotics. As an early adopter of ROS 2, OTTO could scale up to 100+ robots in customer facilities, thanks to ROS 2's fine-grained network topology management and better support for bandwidth management through QoS on shared network links.

D) Air: Auterion Systems

Auterion, a Swiss aerial drone startup, aims to develop commercial autopilots based on the open-source PX4 Autopilot project. The company supports various types of airframes and aims to extend drone operations into unstructured spaces with hazards while enhancing autonomy. Auterion uses ROS 2 to integrate higher-level functionality into its drone systems, focusing on logging and introspection capabilities. ROS 2's logging capabilities collect runtime events, metadata, and raw data streams from all layers of the system, making it crucial for effective development, debugging, and validation processes. Auterion also uses rviz2, a 3-dimensional renderer, to visualize drones and sensor data in an interactive environment. These capabilities enable Auterion to focus on core flight control capabilities and customer requirements, allowing them to focus on building foundational tooling instead of building foundational tools. The company's focus on open standards and open standards has led to improved efficiency in its development process.



(a) VIPER on Lunar Surface (rendering), (b) Command and Operations Software

E) Space: NASA VIPER

NASA's Volatiles Investigating Polar Exploration Rover (VIPER) mission is set to launch in November 2023 to explore the southern polar region of the Moon. The rover will spend 100 days searching for water ice and other resources using various instruments. It will communicate with Earth using the X-band link to the Deep Space Network and use Earth-based computer resources to map terrain and compute stereo solutions. Earth-based operation tools, compute modules and high-fidelity simulations are based on ROS 2 and Gazebo.

The VIPER team is focused on producing highly reliable software and is extensively utilizing Gazebo for high-fidelity testing of all their components and systems. The project developed new plugins to model mission-specifics, such as camera lens flare, lunar lighting conditions, gravity, and terrain on the lunar surface. The VIPER team used Gazebo to test and validate almost all of their rover's software prior to launch.

VIPER reused 284,500 significant lines of code (SLOC) without modification from Gazebo, modifying <1% to pass validation. This code reuse accelerated development, allowing them to produce a simulation in just 266 work months focused on VIPER-specific elements. A combination of Gazebo and ROS 2 is used to train the rover's operators.

3. Research Methodology

"Comparative Analysis of Robotic Operating Systems" would require a structured methodology to systematically compare different robotic operating systems (ROS) and provide meaningful insights. To conduct a comparison of robotic operating systems, it is important to first define the research problem. Which is followed by an explanation of the significance and relevance of the study. The research then states its objectives and hypotheses. A brief overview of robotic operating systems is provided, including their history and evolution. Some existing literature on ROS is reviewed to identify gaps and areas of comparison. The key features, characteristics, and components of ROS that are relevant for comparison have been discussed. A set of ROS packages is compared and identified, based on relevance, popularity, and diversity. Data sources and collection methods are described, and information about each ROS package is gathered, including documentation, user feedback, technical specifications, and community support. Criteria and parameters for comparison are defined, such as hardware compatibility, real-time capabilities, middleware architecture, community support and ecosystem, security features, scalability, and performance metrics. Each ROS package is evaluated based on the defined criteria, using quantitative and qualitative analysis methods, such as surveys, interviews, or experimental data, depending on the criteria. The results are presented in tables, charts, and visual aids, with explanations and interpretations for each comparison point. Later the strengths and weaknesses of each ROS package are highlighted, and the results' implications are discussed. The

ISSN (online): 2581-3048
Volume 7, Issue 10, pp 371-375, October-2023
https://doi.org/10.47001/IRJIET/2023.710050

findings are compared with the literature and existing knowledge, and potential future research directions in the field of ROS are suggested. Finally, the key findings are summarized, and the significance of the study is done.

4. Future Scope

The future scope of research papers in comparative analysis of Robotics operating systems (ROS) is vast and exciting. ROS is a middleware platform that provides a set of tools and libraries for developing robotic applications. It has become the de facto standard for ROS development and is used by researchers and developers all over the world.

One area of future research is in the development of new ROS-based applications for a variety of industries, such as healthcare, manufacturing, and logistics. For example, ROS could be used to develop new surgical robots, autonomous assembly line robots, and self-driving delivery trucks.

Another area of future research is developing new ROS-based tools and libraries to make it easier and more efficient to develop robotic applications. For example, researchers could develop new ROS-based tools for planning, navigation, and control.

Finally, researchers could also focus on improving the performance and reliability of ROS. For example, they could develop new ROS-based tools for real-time task scheduling and fault tolerance.

5. Conclusion

ROS 2 has been completely redesigned to meet the challenges of modern robotics. It was created with a thoughtful set of principles and modern robotics requirements in mind, allowing for extensive customization. Built on top of DDS, ROS 2 is a reliable and high-quality robotics framework that can support a wide range of applications. It continues to accelerate the deployment of robots and drive the next wave of the robotics revolution.

Through a series of case studies, we have shown that ROS 2 is significantly accelerating companies and institutions toward useful deployment in various environments and scales. ROS 2 is an enabler, equalizer, and accelerator. Standardization around ROS 2 in multiple industries is creating opportunities for new collaborations, faster development, and propelling newly developed technologies forward. This trend is expected to continue to manifest in the coming years as ROS 2 reaches peak maturity.

REFERENCES

[1] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leib, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In IEEE International Conference on Robotics and Automation Workshop on Open Source Software, 2009.

[2] Sachin Chitta, Eitan Marder-Eppstein, WimMeeussen, Vijay Pradeep, Adolfo Rodriguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Ludtke, and Enrique Fernandez Perdomo. ROS control: A generic and simple control framework for ROS. Journal of Open-Source Software, 2(20):456, 2017.

[3] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The Office Marathon: Robust navigation in an indoor office environment. In IEEE International Conference on Robotics and Automation, pages 300–307, 2010.

[4] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study. Journal of Software Engineering for Robotics, 5(1):3–16, 2014.

[5] Brian Cairl (Fetch Robotics Inc.). Deterministic, asynchronous message-driven task execution with ROS. In ROSCon Madrid 2018. Open Robotics, September 2018.

[6] Steve Macenski, Francisco Martin, Ruffin White, and Jonatan Gines Clavero. The Marathon 2: A Navigation System. In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020.

[7] G. Pardo-Castellote. OMG Data-Distribution Service: architectural overview. In the International Conference on Distributed Computing Systems Workshops, pages 200–206, 2003.

[8] William Woodall. ROS on DDS. https://design.ros2.org/articles/ros_on_dds.html, accessed February 11, 2022.

[9] Benjamin Kuipers, Edward A. Feigenbaum, Peter E. Hart, and Nils J. Nilsson. Shakey: From Conception to History. AI Magazine, pages 88–103, 2017.

[10] Richard E Fikes and Nils J Nilsson. Strips: A new approach to applying theorem proving to problem-solving. Artificial intelligence, 2(3-4):189–208, 1971.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 10, pp 371-375, October-2023

https://doi.org/10.47001/IRJIET/2023.710050

Citation of this Article:

Prof. S.B.Bele, Prasad Katyarmal, Swaraj Gangane, Komal Thakare, Tasmeeya Sheikh, "Comparative Analysis of Robotic Operating Systems" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 7, Issue 10, pp 371-375, October 2023. Article DOI https://doi.org/10.47001/IRJIET/2023.710050

****

