International Journal of Advance and Applied Research
www.ijaar.co.in
ISSN – 2347-7075 Impact Factor – 7.328
Peer Reviewed Bi-Monthly
Vol.9 No.6 July – Aug 2022

AN ANALYSIS OF ROBOTIC OPERATING SYSTEMS (ROS 2) IN REAL-TIME SYSTEMS

S. Parthiban
Student (UG), Robotics and Automation, Karunya Institute of Technology and Sciences (KITS), Coimbatore, Tamil Nadu Pin: 6411001
***Corresponding Author*- S. Parthiban**
E mail: parthibans.work@gmail.com
DOI- 10.5281/zenodo.7071055

Abstract

In the Contemporary world, the most advanced Industrial Robots and autonomous vehicles perform under rehearsal-time real systems. Robot Operating System (ROS) has developed an opensource lib library it contains a tool that aids the building of robot applications. The Primary version of ROS had not met real-time constraints as expected a newer version of ROS is developed, called ROS 2. But still ROS 2 has shortcomings in the real-time analysis as it is still in the development stage. This paper aims to analyze the Real-time performance of ROS 2 and also aims to identify the problems and solutions regarding the real-time constraints of the Robot Operating System (ROS 2). On the whole, it demonstrates the research on the real-time performance of ROS 2 for the improvement of robotic applications.

Rationale of the Study:

In recent years, industrial robots, autonomous vehicles, process systems, path control, air traffic control systems, and the applications of ROS became diverse, the necessity of the high-tech device is expanding every day and real-time analysis are common characteristics such as reliability and time predictability. The application of ROS for example is that the mobile robot should detect the obstacle, make a decision to move, and also it should react in a certain amount of time and should provide the required results for the user. These are very common in the robotic field and it usually requires high real-time analytical capabilities.

Robot Operating System (ROS) was developed as a framework composes of open-source software libraries and tools that help roboticists to construct robots for various applications. Since the original version of ROS does not support priority and synchronization for tasks for the real-time systems and there is no efficient approach for the real-time approach, so the efficiency for the real-time applications is low for the robot. These shortcomings made the efficiency of the ROS very low in the real-time field so it was upgraded as ROS 2 which has been maintained by Willo George and The Open-Source Robotics Foundation (OSRF) also they have included the use of DSS communication service in ROS 2 for the real-time systems.

Objective of the Study:

The core objective of this paper is to analyze the Robot Operating System (ROS 2) in the real-time field by identifying the issues and their solutions for its betterment when compared to the Primary Version of ROS 1.

Hypothesis:

The main feature which can distinguish the ROS 2 from its primary version is its real-time performance when compared to the ROS and for considering the real-time performance these characteristics have critical

support performance according to Buttazzo.

Predictability:

It is to satisfy the operation of real-time systems at the desired level and the desired system should predict the results for any of the decisions made.

Efficiency:

Since most real-time systems are embedded systems limited in weight, energy consumption, and computational power the management of these resources depicts the efficiency of a real-time system.

Robustness:

The system should not continue to work in high workload conditions hence it should be designed to be able to manage the huge computational loads and also adaptation to different conditions are essential.

Maintainability:

The Architecture of the system should be designed while considering the modularity features to make the system more approachable and easier to maintain for the user

These characteristics play an important role in the real-time applications of the ROS and ROS 2 and ROS 2 has a better efficiency when compared to the ROS because of the following:

ROS 2 Architecture:

When compared to the primary version of the ROS, ROS 2 is connected to multiple layers and these layers support many libraries which are supported by the programming languages such as C++ and python. The ROS Client library (RCL) helps to provide consistency with the help of the APIs. The architecture is shown below:

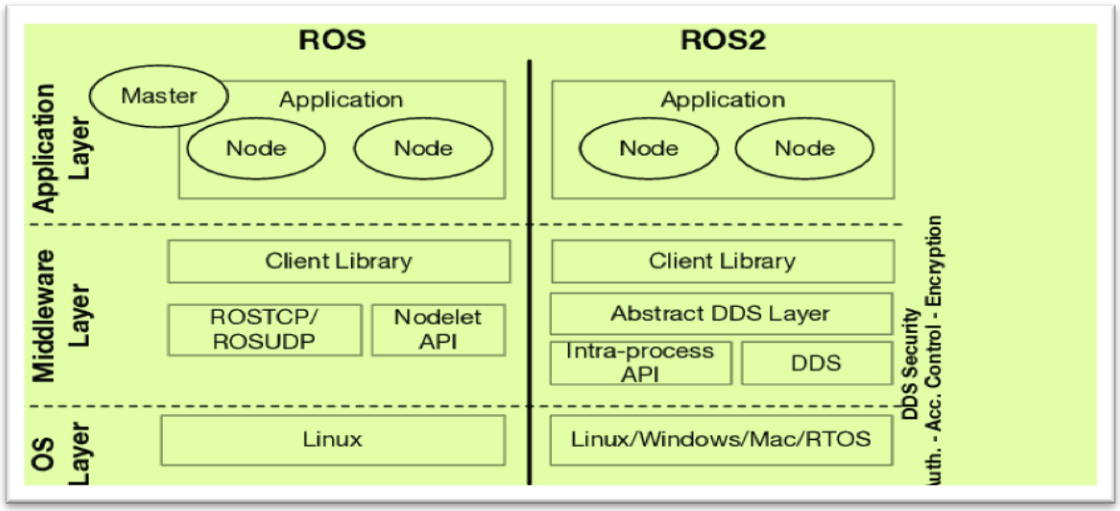


Figure:1- Architecture of ROS1 and ROS2

Multiple nodes in the same executable:

ROS1 a node is too tight to an executable and in ROS1 these are rectified by the addition of the Nodelets but in ROS 2 core it is called the "Components" so here the user can handle as many as nodes from the same executable using components and a component is simply a modified node class.

Services:

In ROS1 the services are synchronous such that when the service client calls a request to the server it is stuck until the server responds but in ROS2 they are asynchronous which makes them more reliable than ROS1.

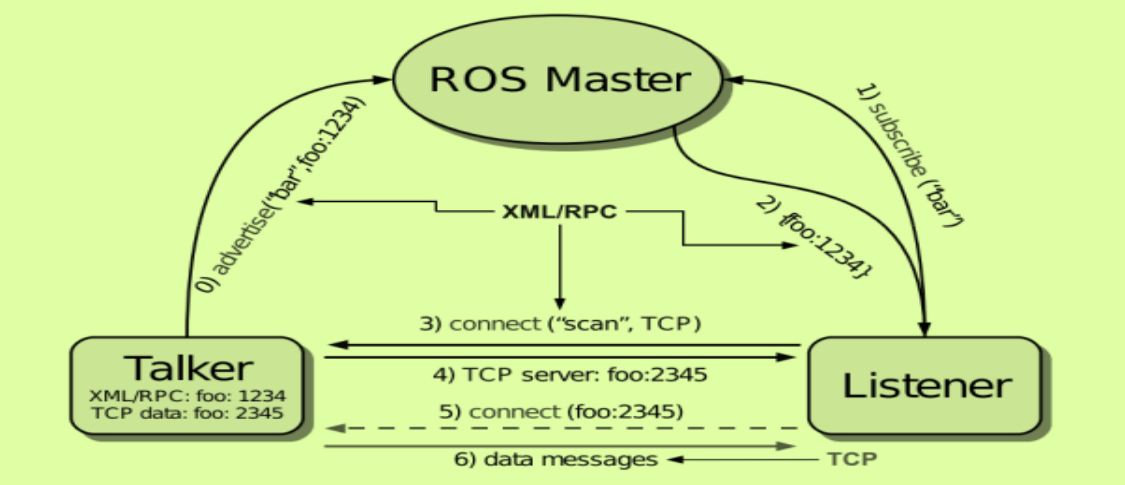


Figure:2- ROS2 Message Communication

CPP and Python Packages:

In ROS1 the packages are created then we need to add any CPP/python file that we want but here in ROS2 we have to specify ament_cmake or ament_python while creating a specific CPP or python package.

Workspace:

Sourcing the environment or workspace is still the same between ROS1 and ROS2 but here in ROS2 it brings a new concept of overlays hence it allows the user to have multiple workspaces on top of the each other. This will come in handy when the user has to create or develop an application and have a certain number of packages but have to create an overlay for just one package and it allows the quick Iteration of the process.

OS Support:

ROS1 mainly supported the Ubuntu platform and it made it difficult for many users as they have to learn a new OS from their well-known OS, but with a new Architecture the ROS2 will be supported even in Ubuntu, macOS, and Windows 10+ which makes it more accessible and embeddable in many robotic applications.

Table:1- ROS1 and ROS2 Timeline

Sl.No.	Years	ROS Versions
	2019-2021	ROS1 Kinetic
	2019-2023	ROS1 Melodic
	2020-2025	ROS1 Noetic
	2020-2021	ROS2 Dashing
	2021-2023	ROS2 Foxy
	2022-2026	ROS2 H

Source: ROS Survey by OSRF

The Above table provides the data of which versions of the ROS have been used by the users in the following years and here we can see that there is a decrement in ROS1 usage and more people are preferring fOS2. One of the main reasons which assist the above statement is that ROS1 Noetic will cease to function in the year 2025 as mentioned by Willo George and ROS Foundation that all the operations can't be further provided or done in the ROS1 Noetic version.So many of the user's eyes are focused on the ROS2 Foxy since it has more predictability, and reliability when compared to the Noetic version but still, there are some shortcomings too for the ROS2 Foxy.

Problems of ROS2:

ROS2 is welcomed by everyone but still, the major problem with ROS2 is that the stability of the platform as

ROS1 is more stable than that many of the users are still using ROS1 more than ROS2 The other major one is that many of the users are still in the Learning Phase of ROS2 as it has many changes when compared to its parent's version The Sources for learning or the libraries for ROS2 are still lacking so it makes the learning process difficult and slower, also many found that it is difficult to switch from ROS1 to ROS2

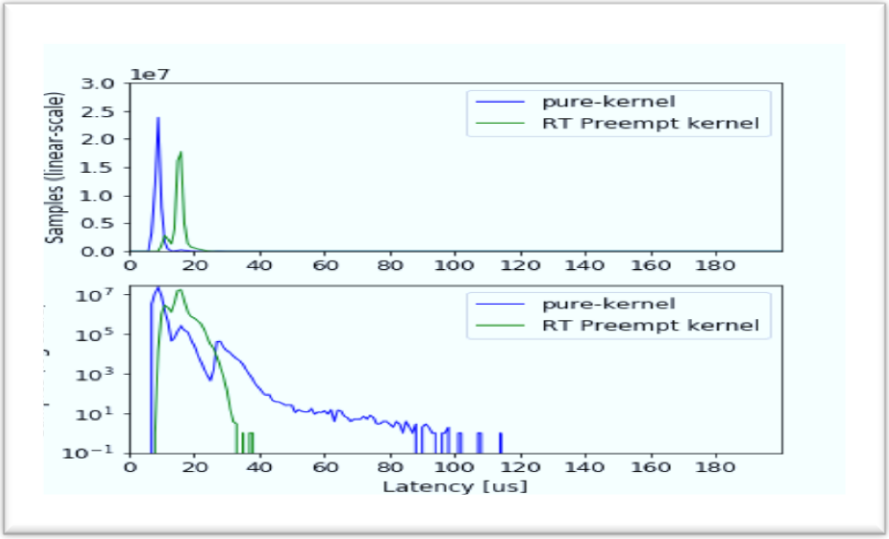
Solution:

The OSRF provided new nodes which allow the user to easily understand the ROS2 features and also, they are providing the ros1_bridge package which allows the user to use the ROS1 and ROS2 together Even though it is possible but they are not directly compatible and some adaptations we required between them and ros1_bridge provide or acts as a bridge between the ROS1 and ROS2

Real-Time Analysis features:

Finally, the real-time analysis of latency of ROS2 when compared with the ROS1 is depicted below in the graph:

Graph:1- ROS1 and ROS2 latency



The above graph shows the ROS1 and ROS2 latency features as we can see that ROS1 has more latency when compared to that ROS2 since its architecture has no support over multiple layers and nodes at a time.

Research Methodology:

This study is analytical and the data sources have been collected from various sources like papers, Journals, Databases, and National and International Publications. Furthermore, the analysis is conducted accurately with a well-defined methodology as this research summarised the problems and provided analytical solutions to the problems identified.

Conclusion:

This paper presents a study of analysing the real-time performance of ROS2. This work identifies the key features and applications of ROS2 in the field of robotics which helps to hinder the gap between robotics and humans. It also identifies the problems regarding ROS2 and also provides an analytical solution to the problem from the aspect. It also compares the Robot Operating System 2 with its predecessor ROS1 and also with its other versions and provides a review about which is the best for the users for the upcoming periods.

In Summary, ROS2 is an open-source robot operating system that helps in various applications such as path planning, autonomous navigation, and self-mapping process for mobile robots. Further directions of this work would include the fabrication, large data, a detailed review, and ever more analysis of the ROS2. Through this paper, I

intend to inspire to aspiring robotic engineers, roboticists who are interested in the field of ROS2 operating systems, and researchers who review this field for more improvisations in the field of robotics and take it to broader aspects in the future directions.

References:

1. Buttazzo, G.C., Hard real-time computing systems: predictable scheduling algorithms and applications. Vol. 24. 2011: Springer Science & Business Media.
2. Mubeen, S., E. Lisova and A. Vulgarakis Feljan, Timing predictability and security in safety-critical industrial cyber-physical systems: A position paper. Applied Sciences, 2020. 10(9): p. 3125.
3. Kay, J. and A.R. Tsouroukdissian, Real-time control in ROS and ROS 2.0. ROSCon15, 2015.
4. Maruyama, Y., S. Kato and T. Azumi. Exploring the performance of ROS2. in Proceedings of the 13th International Conference on Embedded Software. 2016.
5. DiLuoffo, V., W.R. Michalson and B. Sunar, Robot Operating System 2: The need for a holistic security approach to robotic architectures. International Journal of Advanced Robotic Systems, 2018. 15(3): p. 1729881418770011
6. Barut, S., M. Boneberger, P. Mohammadi and J.J. Steil. Benchmarking Real-Time Capabilities of ROS 2 and OROCOS for Robotics Applications. in 2021 IEEE International Conference on Robotics and Automation (ICRA). 2021. IEEE.
7. Blaß, T., D. Casini, S. Bozhko and B.B. Brandenburg. A ROS 2 Response-Time Analysis Exploiting Starvation Freedom and Execution-Time Variance. in 2021 IEEE Real-Time Systems Symposium (RTSS). 2021. IEEE
8. Ye, Y., P. Li, Z. Li, F. Xie, X.-J. Liu and J. Liu. Real-Time Design Based on PREEMPT_RT and Timing Analysis of Collaborative Robot Control System. in International Conference on Intelligent Robotics and Applications. 2021. Springer