Review

Not peer-reviewed version

ROS 2 Key Challenges and Advances: A Survey of ROS 2 Research, Libraries, and Applications

Abdulrahman S. Al-Batati * , Anis Koubaa , Mohamed Abdelkader

Posted Date: 15 October 2024

doi: 10.20944/preprints202410.1204.v1

Keywords: ROS; ROS 2; robotic operating system; modularity; real-time capabilities; security; multi-robot systems; literature review

Review

ROS 2 Key Challenges and Advances: A Survey of ROS 2 Research, Libraries, and Applications

Abdulrahman S. Al-Batati, Anis Koubaa and Mohamed Abdelkader

College of Computer and Information Sciences, Prince Sultan University

* Correspondence: aalbatati@psu.edu.sa

† Current address: 11586 Riyadh, Saudi Arabia

‡ This work was supported by the Robotics & Internet of Things Lab at Prince Sultan University

§ All authors are with the College of Computer and Information Sciences, Prince Sultan University, 11586 Riyadh, Saudi Arabia

Abstract: This study presents a comprehensive systematic review that addresses the critical transition from ROS 1 to ROS 2, spotlighting the significant enhancements and the pressing need for a detailed exploration of ROS 2 within the robotics community. Despite the extensive deployment and adaptations of ROS in varied robotics applications, literature lacks a cohesive synthesis that delineates the advancements, limitations, and broader impacts of ROS 2 compared to its predecessor, ROS 1. Our contribution bridges this gap by assembling the largest database of ROS-related research, encompassing 7,498 articles, with a focused analysis in this survey on 431 ROS2-specific publications. We categorize these into i.) articles that discuss and analyze core ROS 2 concepts, ii.) articles that propose frameworks or tools for ROS 2, and iii.) articles utilizing ROS 2. Furthermore, we summarize literature findings of ROS 2 challenges, advancements, and future direction in the fields of a.) security, b.) real-time, c.) middleware, d.) embedded and distributed systems, e.) communication reliability and QoS, and f.) multi-robot systems. The methodology involved meticulous data collection and categorization from multiple databases, facilitating an in-depth online accessible resource. Results underscore ROS2's enhancements in modularity, real-time capabilities, and security, extending its applicability across various robotic platforms and industries. However, challenges in scalability and reliability persist, signaling avenues for future enhancements. This review not only deepens the understanding of ROS2's contributions but also charts a path for ongoing improvements in robotic systems design. The original data presented in the study are openly available in https://www.ros.riotu-lab.org/

Keywords: ROS; ROS 2; robotic operating system; modularity; real-time capabilities; security; multi-robot systems; literature review

1. Introduction

Robot Operating System (ROS, aka ROS1) emerged in 2009 and has been evolving as the de facto standard ecosystem for developing robotics applications, including mobile robots, robotic arms [1], unmanned aerial systems and drones [2], self-driving cars [3], quadruped robots [4], space robots [5], and much more. ROS provides easy access to several open-source libraries, making developing robotic systems and applications faster. For example, open-source libraries like gmapping from OpenSLAM [4] and cartographer from Google [6] are integrated into ROS to provide a full-fledged navigation system for building maps and navigation. The OpenCV library [7] for image processing and the Point Cloud library [8] for point cloud processing are supported by ROS for advanced computer vision applications. ROS also provides high-level abstractions to low-level hardware drivers, simplifying the burden of low-level programming. ROS allowed robotics application designers and developers to focus on their scope of work rather than being distracted by marginal side details. These are just samples of ROS features that turned it into the most popular robotic framework.

However, at ROSCon 2014 conference, the Open Source Robotics Foundations (OSRF) and the ROS community started to discuss and identify some gaps in ROS, including (a) single point of failure

 doi:10.20944/preprints202410.1204.v1

with the ROS Master Node, which also limits its scalability (b) nonsupport of multi-robots systems as ROS was designed for a single robot usage, (c) and the lack of support for real-time guarantee or quality of service profiles, and thus its reliability was questionable.

The ROS community has formed several working groups to design and develop the next-generation version of ROS, which is currently called ROS 2. Alpha and Beta versions started showing up from August 2015 until September 2017, and the first official ROS 2 release, Ardent Apalone, was launched on December 8th, 2017. Despite the release of ROS 2, ROS was still being used more actively than ROS 2 until 2023, and the main reason is that ROS 2 has been under continuous development and has yet to reach the same level of maturity as ROS. In fact, the ROS 2 developers created the ROS Bridge package to communicate between ROS and ROS 2 when ROS is needed from missing functionalities in ROS 2.

However, things have changed since late 2021, when ROS 2 has accomplished most of the missing functionalities, including navigation, transformations, and others. It is expected that ROS 2 will completely dominate ROS after the EOL of the last ROS distribution (ROS Noetic). Figure 1 shows the road map to ROS 2 until the Jazzy distribution release.

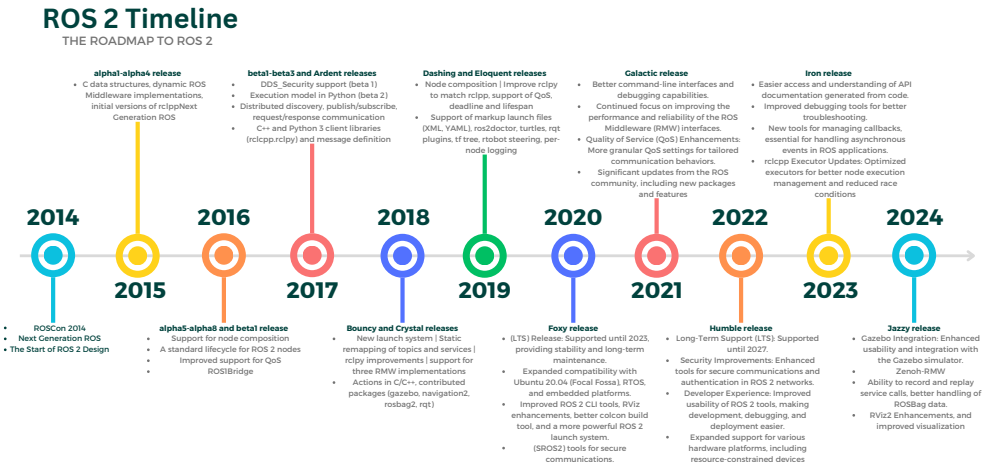

Figure 1. Timeline of ROS 2 until 2022.

1.1. Methodology

The methodology for obtaining and analyzing the literature on ROS 2 involved a comprehensive search across multiple databases including Google Scholar, IEEE, ACM, ScienceDirect, Springer, Elsevier, SAGE, and arXiv. From these sources, we collected a total of 7497 articles related to ROS 1 and ROS 2, dating from 2009 to the present. All these articles are recorded in our online database https://www.ros.riotu-lab.org/, where each entry includes information about which version of ROS it addresses or utilizes, the research field, the robotic platform used, the article DOI, and its GitHub repository if available. This database allows users to filter and sort the records to find specific information, such as all GitHub ROS 2 repositories for UAV swarms.

From the initial collection, we identified 431 articles specifically focused on ROS 2. These articles were further categorized into three groups as shown in Figure 2, and they are explained as follows:

1. **Articles that analyze ROS 2:** This group includes 98 articles that discuss various areas that are core to ROS 2 design, such as Security, Real-Time Capabilities, Communication, and Multi-robot Systems support. Each research area is explored in depth, providing critical insights into the literature, motivations for the researchers, their contributions, and current gaps in Section 4.
2. **Articles that propose tool-kits for ROS 2:** This group comprises 130 articles. These research works provide open-source software packages for ROS 2 users for different applications such as ROSGPT [9] for HRI, or CrazyChoir [10] for UAV in MRS settings. We cite some of these articles

 doi:10.20944/preprints202410.1204.v1

with their GitHub repositories in a table in Section 5. This highlights the significant frameworks and tools developed by the community to support ROS 2.

3. **Articles that utilize ROS 2:** This group includes 203 research works that use ROS 2 as a middleware to facilitate their development such as the use of ROS 2 in healthcare research [11]. We provide a taxonomy for these works. We mention some of these articles to showcase the diverse fields and applications that utilize ROS 2 in Section 6, demonstrating its broad applicability and impact.

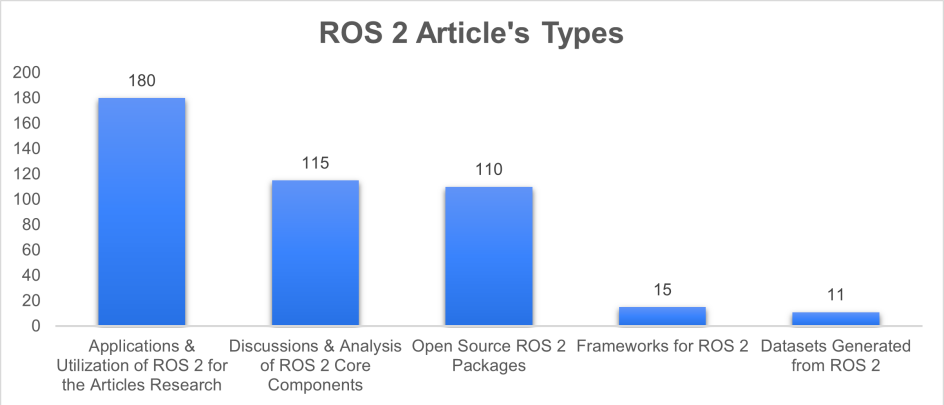

Figure 2. Taxonomy of ROS 2 articles types.

The detailed methodology for literature collection and categorization is illustrated in Figure 3, which outlines the process from initial database searches to the final categorization of articles. This approach ensures a comprehensive and systematic review of the ROS 2 literature, providing valuable insights and resources for researchers and developers in the field.

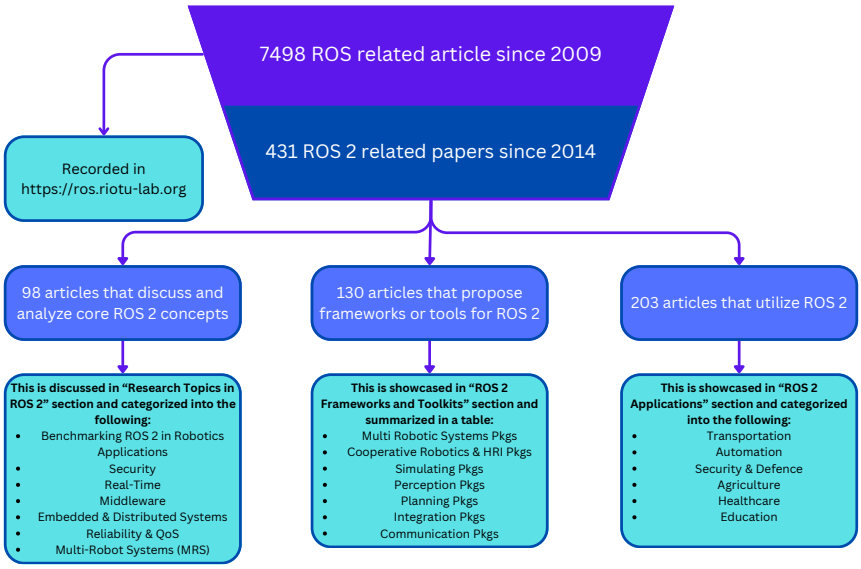

Figure 3. Literature collection methodology.

1.2. Related Surveys

The body of literature surrounding ROS 2 has expanded significantly, with various surveys addressing specific aspects of this robotic operating system. In this subsection, we highlight key

 doi:10.20944/preprints202410.1204.v1

surveys, their focal points, the gaps they reveal, and how our work contributes to the existing knowledge base. A summarized comparison is presented in Table 1.

Several noteworthy surveys have examined different facets of ROS 2. For instance, Audonnet et al. [12] conducted a systematic comparison of simulation software for robotic arm manipulation in ROS 2, while Zhang et al. [13] reviewed innovative architectures and technology readiness for distributed robotic systems within the edge-cloud continuum, focusing on ROS 2. Macenski et al. [14] offered a comprehensive survey of modern mobile robotics algorithms in ROS 2, with particular attention to navigation systems. Additionally, Choi et al. [15] explored priority-driven real-time scheduling in ROS 2, and Macenski et al. [16] examined the design, architecture, and practical applications of ROS 2 across various environments. Similarly, DiLuoffo et al. [17] addressed the critical need for secure robotic architectures, advocating for a holistic security approach in ROS 2. Finally, Alhanahnah [18] evaluated software quality in ROS through static analysis of ROS repositories.

While these surveys contribute valuable insights, each focuses on a narrow aspect of ROS 2. In contrast, our survey offers a comprehensive review, spanning a wide range of research areas, including security, multi-robot systems, modularity, and real-time performance. Furthermore, we introduce a detailed database of ROS and ROS 2 literature, categorized by research domain, targeted robotic platforms, industry focus, and article type. This extensive scope not only fills the gaps identified in previous surveys but also provides a holistic view of ROS 2's ecosystem and its evolving community contributions.

Table 1. Comparison of ROS 2 Surveys.

Survey	Focus Area	Gaps	Our Contribution
Audonnet et al. [12]	Simulation software for robotic arms	Limited to simulation software	Comprehensive coverage of ROS 2 applications
Zhang et al. [13]	Edge-cloud integration	Does not cover security, real-time performance	Wide range of research topics including security and modularity
Macenski et al. [14]	Mobile robotics navigation	Focuses only on navigation systems	Holistic view including multi-robot systems and real-time performance
Choi et al. [15]	Real-time scheduling	Limited to real-time performance	Inclusion of various research domains
Bonci et al. [19]	Industrial autonomy	Focused on industrial applications	Broader application fields
Macenski et al. [16]	ROS 2 architecture and uses	Lack of systematic research analysis	Systematic review and database of literature
DiLuoffo et al. [17]	Security in ROS 2	Focused solely on security	Inclusion of multiple research areas
Alhanahnah [18]	Software quality assessment	Centered on ROS 1	Extensive insights into ROS 2

1.3. Contributions

This survey provides a comprehensive analysis of the Robot Operating System 2 (ROS 2), highlighting its development, advancements, and applications across various research domains. Our contributions are summarized as follows:

1. **Extensive Literature Review:** Our survey systematically reviews the extensive body of literature on ROS 2, categorizing research topics into key areas such as security, multi-robot systems, modularity, and real-time performance. We provide detailed summaries and analyses of significant studies in each area.
2. **Community and Ecosystem Contributions:** We highlight the significant contributions of the ROS 2 community, including notable frameworks, tools, and open-source projects. This includes an overview of influential ROS 2 GitHub repositories and their impact on the robotics ecosystem.
3. **Development of a Literature Database:** As part of our survey, we introduce a comprehensive online database that catalogs ROS and ROS 2 literature. This database categorizes articles by research domain, targeted robotic platform, industry focus, and article type, making it a valuable resource for researchers and developers.

1.4. Paper Structure

The structure of this paper is as follows: **Section 2** provides an overview of ROS & ROS 2, detailing their development history, and core features. **Section 3** discusses ROS 2 architectural improvements over ROS 1, including a comparison between ROS 1 and ROS 2. **Section 4** explores various research topics within ROS 2, such as security, real-time capabilities, and other significant

 doi:10.20944/preprints202410.1204.v1

areas like middleware and communication. **Section 5** discusses the ROS 2 frameworks and toolkits, including major contributors and notable GitHub repositories. **Section 6** reviews the applications of ROS 2 across different industries and research domains, highlighting specific case studies of successful implementations. **Section 7** introduces the literature database created as part of this survey, providing an analysis of key insights and trends. **Section 8** concludes the paper by summarizing the key findings, discussing the impact of ROS 2, and emphasizing gaps, future directions, and the importance of continued research and community collaboration.

2. ROS & ROS 2 Overview

In this section, we will explore the distinctions between ROS and ROS 2. Originating in 2014, the Open Source Robotics Foundation (OSRF) and the ROS community identified several deficiencies in ROS, prompting the necessity for an evolved version of ROS. This was driven by the escalating demands of robotics applications, which necessitated enhancements in aspects such as real-time guarantees, quality of service, security, and scalability.

We will start by examining the core features of ROS, subsequently shift our focus to the primary design objectives of ROS 2, and conclude by juxtaposing the two versions. This comprehensive analysis aims to provide a detailed understanding of the evolution of ROS and the improvements brought about by the advent of ROS 2.

2.1. ROS

2.1.1. ROS Design Goals

ROS was fundamentally designed to serve as a unified platform for building single-robot applications. It offers a standard collection of tools, libraries, and conventions that aid in developing a broad spectrum of robotic applications, spanning from basic prototypes to intricate robotic systems. One of the salient design objectives of ROS was to promote reusability and modularity, enabling developers to readily share and consolidate software components, thereby facilitating the creation of sophisticated robotic applications. Notwithstanding the diverse use cases it covers, ROS's primary focus remains on single-robot systems. While there have been initiatives to extend ROS's capabilities towards multi-robot systems [20], these efforts were neither seamlessly integrated nor widely adopted.

The next section will provide an overview of ROS Architecture.

2.1.2. ROS Architecture

ROS architecture is depicted in Figure 4 and summarized as follows.

 doi:10.20944/preprints202410.1204.v1

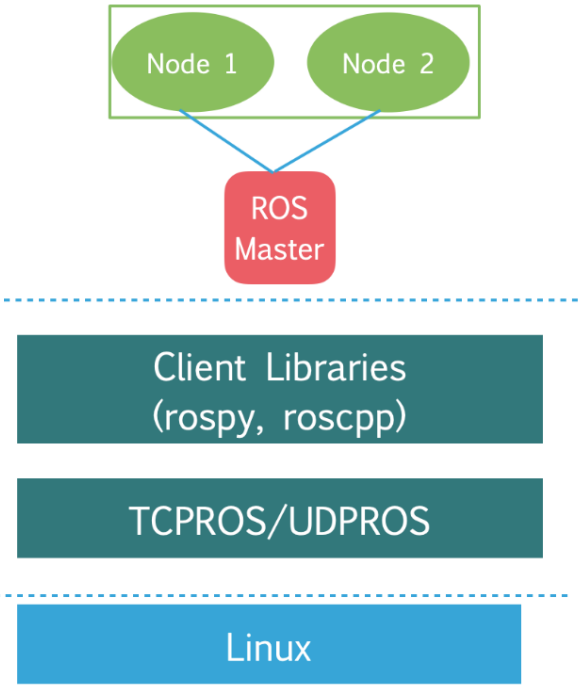

Figure 4. ROS 1 Architecture.

Natively Centralized

The Robot Operating System (ROS) primarily serves as a middleware framework, promoting communication between various processes, called *nodes*, within a computational machine. Conceptually, ROS can be envisaged as a *network of nodes (processes)*, exchanging messages to execute a specific task. This network pivots around a *central node*, termed the *ROS master node* (ROSCORE), which serves as a liaison between all nodes within the ROS network. The functionality of the ROS network is critically dependent on this master node. It is worth noting that the dependence on a single master node presents limitations in terms of scalability and introduces a single point of failure.

A ROS node can be a simple C++ program or a Python script, which reads data from other nodes, processes this data, and subsequently publishes it for other nodes. For instance, a camera node may acquire images via its drivers and publish this image data through a specified channel or topic (e.g., `/camera/rgb/`). Another node may receive this image data, process it using computer vision algorithms, and send the results to another node for appropriate action.

A Modular and Scalable Architecture

The structure of ROS, characterized by its distributed nodes, closely reflects the concept of a microservices architecture. In such an architecture, a monolithic application is partitioned into smaller, standalone services, each performing a specific function. This design is a strategic approach to managing complexity in large applications, facilitating modularity, and promoting scalability, as services can be developed, deployed, and scaled independently. Similarly, ROS strongly emphasizes modularity, allowing its applications to be executed and operated over multiple nodes. Each node in a ROS application performs a discrete task and can operate independently, analogous to a service in a microservices architecture. This arrangement allows for simplified debugging, testing, and understanding of individual components while permitting complex behaviors when the components interact. Thus, ROS's modular and distributed nature echoes the microservices architecture principles, enhancing maintainability and scalability in robotic applications.

 doi:10.20944/preprints202410.1204.v1

Communication Patterns

ROS employs several communication mechanisms that enable a seamless exchange of information between nodes.

- ***Publisher-Subscriber (Topics):*** Fundamental to ROS's communication infrastructure is the Publisher-Subscriber model, implemented through *Topics*. In this model, a node that generates data serves as the *Publisher*, broadcasting information over a channel known as a Topic. Other nodes that need this information can *subscribe* to the respective Topic, thereby consuming the published data. This decoupled communication model enables a many-to-many data flow, promoting the distribution of data to all interested nodes. Topics carry ROS messages, which encapsulate the information to be transmitted. The Publisher-Subscriber paradigm enhances flexibility, as nodes can dynamically publish or subscribe to Topics, and robustness, as the failure of one node does not directly impact others.
- ***Synchronous Client/Server: (Services)*** Services denote the synchronous client/server model within ROS. In this paradigm, a client node sends a request to a server node and waits for a response. The client is unable to perform other tasks until this response is received, indicating a *synchronous* operation. This model is applied when the server can complete the task swiftly and the client requires an immediate response. If the task duration is prolonged, the Services model becomes unsuitable, and the ROSLib concept is employed instead.
- ***Asynchronous Client/Server: (ActionLib)*** The ActionLib model signifies the *asynchronous client/server model* in ROS. This model contrasts with ROS Services in that the ActionLib client is not blocked while waiting for the server's response. The client can concurrently execute other tasks while the server processes its mission. In addition, the ActionLib server can send intermediate results to the client, updating it on the task progress. This model is particularly useful for tasks such as robot navigation, where the server processes a navigation request over an extended period, and the client monitors progress while performing other tasks.
- ***Messages and Services:*** In ROS, two distinct types of entities are used to facilitate communication: *Messages* for Topics and *Services* for synchronous client-server communication. Messages (`rosmsg`) are simple data structures comprising typed fields. They are used when nodes want to transmit data to one or multiple recipients. On the other hand, Services (`rossrv`) are defined by a pair of messages, one for the request and one for the response. They are used for synchronous RPC-like communication. This division, while serving its purpose, introduced complexity to the development process, as developers had to work with different types of entities depending on the communication model employed.

Client Libraries and Communication Middleware

While the fundamental concepts of ROS are retained in ROS 2, substantial differences lie in the client libraries, the communication middleware, and their underlying implementations.

In ROS, two primary client libraries are used: rospy for Python and roscpp for C++. These libraries provide the means for creating ROS nodes and handling the communication between them. Rospy primarily supports Python 2 across all ROS versions, except for the final ROS version, ROS Noetic, which introduces Python 3 support.

Regarding communication middleware, ROS employs a custom-made middleware: ROSTCP and ROSUDP, based on the TCP and UDP protocols, respectively. ROSTCP ensures reliable, connection-oriented communication, while ROSUDP enables connectionless communication with less overhead but no delivery guarantees. However, these custom protocols present limitations, such as the lack of real-time capabilities and difficulty in configuring Quality of Service (QoS) settings.

In contrast, ROS 2 introduces new client libraries, namely rclcpp and rclpy, with significant improvements over their ROS counterparts. More notably, ROS 2 transitions from custom protocols to the standardized Data Distribution Service (DDS) middleware for its communication infrastructure.

 doi:10.20944/preprints202410.1204.v1

This shift addresses the limitations of ROS middleware and will be explored in detail in the following section.

2.1.3. Limitations of ROS and the Motivation for ROS 2

Despite the significant contributions of ROS to the robotics community, certain limitations have prompted the development of a more advanced version. The deliberation to address these limitations began at the ROSCon 2014 conference, where the Open Source Robotics Foundation (OSRF) and the ROS community collectively identified areas of improvement.

- ***Single-Robot Focus*:** The primary limitation of ROS is its exclusive design for single-robot applications. The absence of inherent support for multi-robot systems presented a challenge for applications involving swarms or collaborative robots. This is primarily attributed to ROS's design requirement of a central ROS Master node, creating an isolated network per robot. In multi-robot applications, this necessitates additional packages to enable inter-robot cooperation. ROS 2 has addressed this limitation by removing the requirement for a ROS Master node, which we will discuss later.
- ***Lack of Real-Time Guarantees and Quality of Service*:** Another significant drawback of ROS is the absence of real-time guarantees and Quality of Service (QoS) profiles for prioritizing critical messages. ROS's reliance on TCP and UDP protocols translates to a best-effort service for message delivery without any assurance of successful transmission. The absence of message prioritization implies that critical and regular messages are treated on par, making it unsuitable for industrial-grade applications with stringent real-time and safety requirements, such as autonomous vehicles or unmanned aerial systems.
- ***Reliability and Scalability Concerns*:** Lastly, the reliability and scalability of ROS are challenged by its dependency on the central ROS Master node. The entire network fails if the ROS Master node crashes, establishing a single point of failure. This design also constrains the scalability of ROS.

In the following section, we delve into how ROS 2 has been designed to address and overcome these limitations.

2.2. ROS 2

2.2.1. ROS 2 Design Goals

The design of ROS 2 was primarily driven by the need to overcome the limitations of ROS, as discussed in the previous section.

Focus on Swarm Robotics

In contrast to ROS, which was primarily designed for standalone robots, ROS 2 has been engineered emphasizing swarm robotic applications. The inherent complexity of swarm robotics necessitates a fully distributed middleware that enables ad-hoc communication between multiple devices. This distributed communication structure allows for the efficient coordination and collective behavior that characterize a robot swarm. This shift towards a more distributed architecture is a key factor differentiating ROS 2 from its predecessor.

Real-Time and QoS Guarantees

ROS's lack of support for real-time guarantees and Quality of Service (QoS) were significant limitations, especially for applications with stringent timing and reliability requirements. To address this, real-time support and QoS guarantees were positioned at the forefront of ROS 2's design considerations. The middleware used in ROS 2 is expected to provide sophisticated QoS policies, ensuring the prioritized and timely delivery of messages based on their importance and urgency. These

 doi:10.20944/preprints202410.1204.v1

features are crucial for applications that require high reliability and deterministic execution, such as autonomous vehicles and industrial automation systems.

Fast Prototyping and Cross-Platform Compatibility

Another important aspect of ROS 2's design is its emphasis on enabling a seamless transition from fast prototyping to production, coupled with cross-platform compatibility. This focus is designed to facilitate the rapid development and deployment of ROS 2 applications across a variety of platforms and into production environments. This compatibility and ease of deployment are expected to expedite the development cycle, a significant improvement over ROS, which presented challenges in these areas.

Middleware Selection

The above design goals significantly influenced the selection of the communication middleware for ROS 2. The need for a robust, efficient, and reliable communication infrastructure that could meet the advanced requirements of real-time processing, Quality of Service, and distributed system support guided the choice of middleware. After extensive deliberations in 2014, the Data Distribution Service (DDS) was chosen for its inherent ability to meet these requirements. DDS, a standard for high-performance, scalable, and interoperable publish-subscribe communication, was deemed perfectly suited for the task. The choice of DDS as the middleware plays a pivotal role in ensuring that ROS 2 successfully addresses the limitations of ROS and meets its design objectives.

2.2.2. ROS 2 Architecture

ROS 2's architecture, as presented in Figure 5, retains similarities with ROS in certain aspects but also presents a significant evolution to address ROS's limitations.

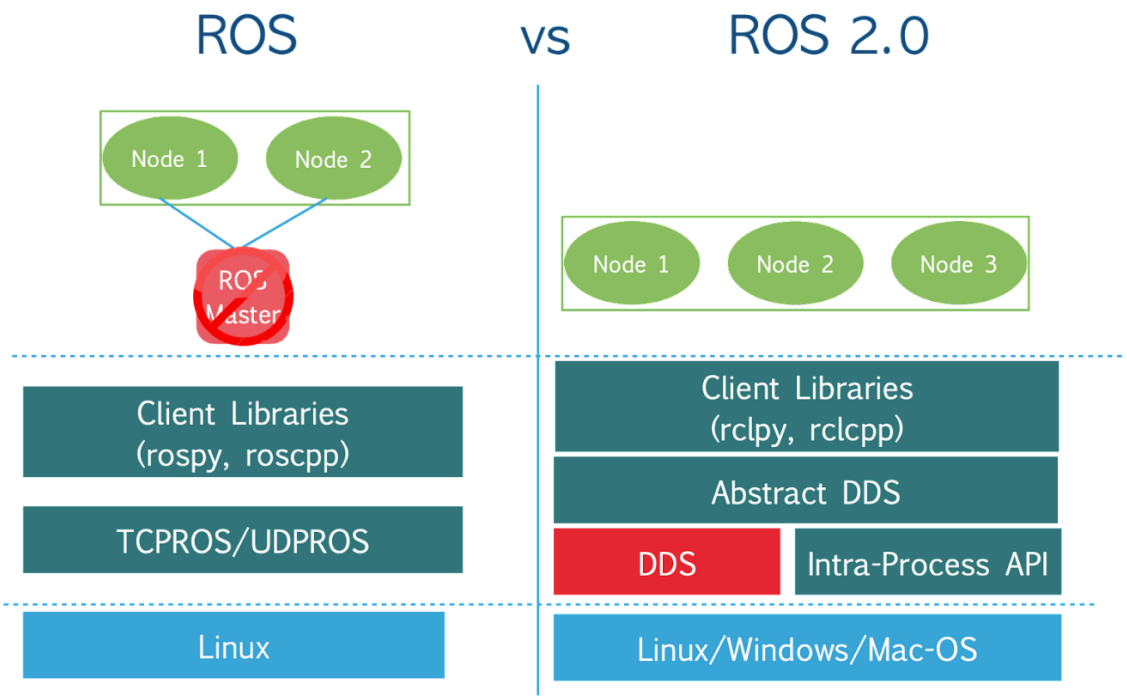

Figure 5. ROS vs. ROS 2 Architecture.

 doi:10.20944/preprints202410.1204.v1

Distributed by Design

In contrast to ROS's centralized approach, ROS 2 is designed to be inherently distributed. It eliminates the need for a central ROS master node, thereby addressing the single point of failure and scalability issues present in ROS. Instead, ROS 2 nodes communicate directly with each other, facilitated by the underlying DDS middleware. This decentralized design is more fitting for multi-robot systems and enables more efficient and reliable communication in large-scale, distributed robotic applications.

Maintained Modularity and Enhanced Scalability

Like ROS, ROS 2 upholds the principle of modularity, where each node operates independently and performs a specific task. However, ROS 2 extends this concept to incorporate the notion of 'Managed Nodes' that can be governed by a lifecycle manager, thereby providing more control over the system's state and behavior. Furthermore, the distributed nature of ROS 2 enhances scalability, enabling the system to handle a more considerable number of nodes efficiently.

Extended Communication Patterns

ROS 2 inherits the primary communication patterns of ROS, with some notable improvements and extensions.

- ***Publisher-Subscriber (Topics):*** The Publisher-Subscriber model, implemented via Topics, remains a vital part of ROS 2. However, ROS 2 introduces the concept of Quality of Service (QoS) policies for Topics, allowing more precise control over message delivery.
- ***Synchronous Client/Server (Services):*** The synchronous client/server model, facilitated by Services, is also retained in ROS 2. Yet, similar to Topics, Services in ROS 2 support QoS settings, enabling reliable and timely communication based on specific requirements.
- ***Asynchronous Client/Server (Actions):*** The asynchronous client/server model, formerly denoted by ActionLib in ROS, has been incorporated into the core of ROS 2 as Actions. Actions in ROS 2 provide a more streamlined interface and support QoS policies, offering improved reliability and flexibility.
- ***Interfaces:*** In ROS 2, the concept of *Interfaces* is introduced, which encapsulates the functionality of Topics, Services, and Action messages. Interfaces replace the separate message types used in ROS, namely `rosmsg` and `rossrv`. This consolidation is an attempt to simplify the message generation process and to facilitate compatibility between different communication models. Interfaces enhance the consistency and maintainability of the ROS 2 communication model, as developers only need to familiarize themselves with a single message type, regardless of the communication pattern they employ. This simplification also enhances code portability between different ROS 2 applications, leading to a more robust and flexible system design.

Enhanced Client Libraries and Advanced Communication Middleware

ROS 2 introduces updated client libraries, rclcpp for C++ and rclpy for Python, which are more robust, maintainable, and feature-rich compared to their ROS counterparts. The most significant shift in ROS 2 is adopting the *Data Distribution Service* (DDS) as its communication middleware. DDS is an industry-standard middleware designed for high-performance, real-time, scalable, and interoperable publish-subscribe communication. The use of DDS in ROS 2 addresses the limitations of the customs protocols used in ROS. It provides native support for real-time communication, configurable QoS policies, and robust security mechanisms. Furthermore, the standardized nature of DDS facilitates interoperability with other systems and technologies, broadening the scope and applicability of ROS 2.

In the next section, we will further examine the distinctions between ROS 1 and ROS 2 across various non-architectural aspects.

 doi:10.20944/preprints202410.1204.v1

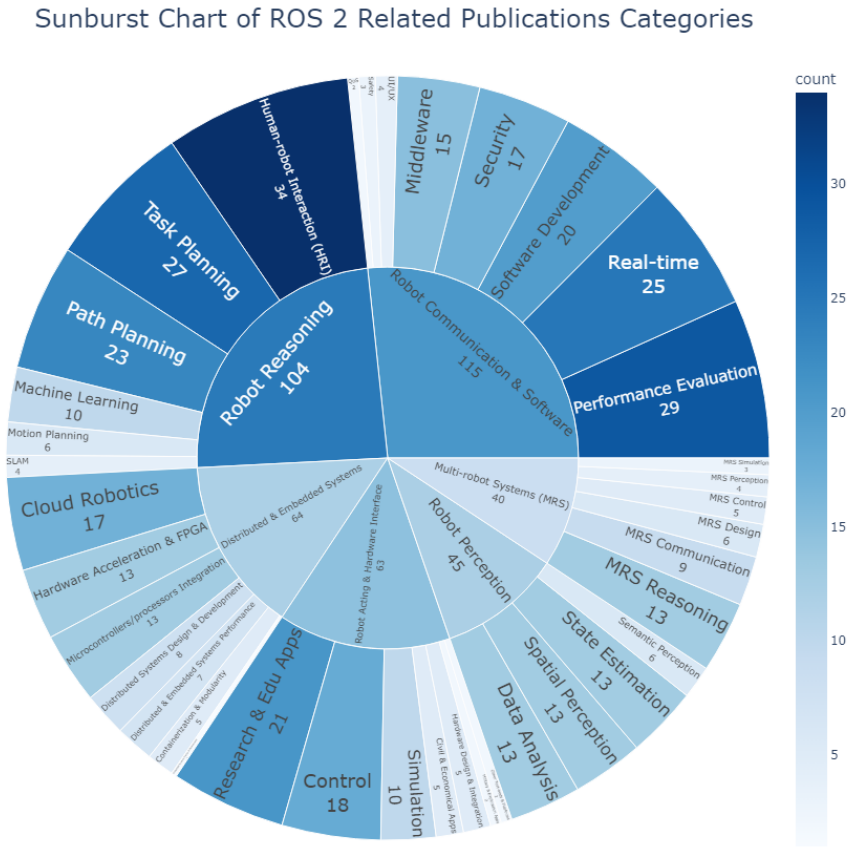

Figure 6. ROS 2 Puplications Catgories.

3. ROS vs. ROS 2

This section discusses the non-architectural differences between ROS and ROS 2 from different perspectives.

3.1. Programming Languages (C++ and Python Differences)

In ROS 2, both C++ and Python programming languages are embraced, but their utilization differs from ROS.

C++ in ROS 2: ROS 2 offers enhanced support for modern C++ features, including move semantics and lambda functions. This improved support leads to better performance and code readability in ROS 2 applications. Developers can leverage new language features and libraries that were previously unavailable in the older C++03 version. Notably, certain core libraries in ROS 2 have been upgraded to support C++14, expanding the range of modern C++ capabilities that can be utilized. These advancements empower developers to take full advantage of the latest features and functionalities offered by the C++ programming language.

Python in ROS 2: In contrast to ROS, which predominantly utilized Python 2, ROS 2 adopts Python 3 as its primary version. Consequently, ROS 2 packages are primarily written in Python 3. This transition allows developers to leverage recent Python enhancements, such as type hints and improved asyncio support. By aligning with the broader Python language community, ROS 2 enables seamless integration with other Python-based tools and libraries. However, it's important to note that compatibility updates are required for any existing Python 2 code intended for ROS 2 usage.

Language-Agnostic Communication Interfaces: One of the notable advancements in ROS 2 is the introduction of the Abstract Data Distribution Service (DDS) interface, a language-agnostic

 doi:10.20944/preprints202410.1204.v1

communication protocol. This protocol enables seamless interaction and communication between nodes written in different programming languages. With DDS, nodes can exchange messages and information efficiently, regardless of whether they are implemented in C++, Python, or any other supported language. This language-agnostic approach facilitates interoperability and collaboration among diverse robotic systems.

These disparities in language support between ROS and ROS 2 emphasize the significance of staying abreast of the latest language features and standards. By accommodating modern C++ and Python versions, ROS 2 empowers developers to fully capitalize on the latest advancements in these languages, enhancing the quality and efficiency of their robotic applications. Ultimately, the choice between C++ and Python hinges on the specific requirements of the target robotics system, and ROS 2's language flexibility caters to diverse needs.

3.2. Executors

In ROS 2, executors play a pivotal role in managing the execution of tasks within a node's lifecycle. They facilitate communication between nodes and ensure the execution of callbacks for incoming messages, services, and actions, significantly influencing the overall ROS 2 architecture.

ROS 2 encompasses three executor types each offering distinct characteristics and benefits:

- `SingleThreadedExecutor`: This executor executes all callbacks within a single thread, employing a round-robin scheduling approach to ensure fairness between tasks. However, computationally intensive tasks may experience slower execution due to the shared thread among all tasks.
- `MultiThreadedExecutor`: Designed to handle high workloads and computationally intensive tasks, this executor employs multiple threads to execute callbacks concurrently. While it provides improved performance, careful synchronization is necessary to avoid potential race conditions or deadlocks.
- `StaticSingleThreadedExecutor`: This executor is similar to `SingleThreadedExecutor` but is specifically designed for nodes with a fixed set of entities (such as sensors, actuators, or other components) known during the compilation process. It optimizes runtime costs by scanning the structure of the node, including subscriptions, timers, service servers, and action servers, only once during node addition. Unlike other executors, it does not regularly scan for changes. Therefore, the StaticSingleThreadedExecutor is most suitable for nodes that create all their subscriptions, timers, and other entities during initialization and do not dynamically add or remove them during runtime. By eliminating the need for continuous scanning, it improves performance and efficiency in such static systems.

In contrast, ROS employs a " spinner " mechanism to manage callback execution within a node's lifecycle. Two primary spinner types are available:

- **ros::spin():** This single-threaded spinner sequentially processes callbacks within a single thread until the node is shut down. It represents the simplest and most commonly used spinner in ROS.
- **ros::AsyncSpinner:** This multi-threaded spinner concurrently processes callbacks using multiple threads, suitable for scenarios involving computationally intensive tasks or varying callback execution times.

While spinners in ROS serve a similar purpose to executors in ROS 2, the latter offers advanced features and greater flexibility in task execution, including the StaticSingleThreadedExecutor for static systems. Moreover, introducing the Data Distribution Service (DDS) middleware in ROS 2 necessitates a refined and adaptable approach to task execution, effectively addressed by the concept of executors.

Using executors in ROS 2 provides efficient task management and optimal utilization.

 doi:10.20944/preprints202410.1204.v1

3.3. Transformations: tf vs tf2

The original library for managing transforms in the Robot Operating System (ROS), known as tf [21], utilizes a static transform tree to outline the relationships between coordinate frames. However, tf exhibits certain limitations, including susceptibility to errors during concurrent multi-thread access to the transform tree and inefficiency when handling large trees.

In contrast, tf2, an advanced version, employs a more efficient data structure for the transform tree representation, enhancing both speed and robustness. It is designed to be thread-safe, thereby eliminating the risk of errors during simultaneous access by multiple threads.

A novel feature introduced in tf2 is the Buffer, a sophisticated data structure that oversees the transform tree. It offers APIs for querying transforms between coordinate frames, thereby increasing the versatility and usability of the system.

A significant distinction between tf and tf2 lies in their support for coordinate representations. While tf is limited to the XYZ Euler angle representation for rotations, tf2 supports multiple representations. These include quaternions, rotation matrices, and angle-axis representations, thereby offering a more comprehensive and flexible system for users.

Moreover, tf2 introduces several new features, including support for non-rigid transforms and the capability to interpolate between transforms. It also enhances debugging and visualization support, thereby simplifying the process of diagnosing issues with the transform tree.

To summarize, due to its superior performance, thread safety, and additional features, tf2 is the recommended library for managing transforms in ROS 2. Its design and functionality improvements over tf make it a more robust and efficient tool for managing coordinate frame relationships.

3.4. ROS 2 Navigation Stack: Main Features and Comparison with ROS 1

The ROS 2 Navigation Stack, also known as Navigation2, is a significant upgrade from the ROS 1 navigation stack, with several key features that enhance its functionality and usability. A comprehensive description of the ROS 2 Navigation Stack can be found in the following reference: [22]. In what follows, we present a systematic and methodological comparison of the ROS 1 and ROS 2 navigation stacks:

- **Task Orchestration using Behavior Trees** introduces the use of a *behavior tree* for task orchestration, a feature absent in ROS 1. This tree orchestrates planning, control, and recovery tasks, with each node invoking a remote server to compute one of these tasks using various algorithm implementations.
- **Modularity and Configurability:** Navigation2 is designed to be highly modular and configurable, a marked improvement over ROS 1. It employs a behavior tree navigator and task-specific asynchronous servers, each of which is a ROS 2 node hosting algorithm plugin. These plugins are libraries dynamically loaded at runtime, allowing for unique navigation behaviors to be created by modifying a behavior tree.
- **Managed Nodes:** ROS 2 introduces the concept of Managed Nodes, servers whose life-cycle state can be controlled. Navigation2 exploits this feature to create deterministic behavior for each server in the system, a feature not present in ROS 1.
- **Feature Extensions:** Navigation2 supports commercial feature extensions, allowing users with complex missions to use Navigation2 as a subtree of their mission. This is a unique feature not found in ROS 1.
- **Multi-core Processor Utilization:** Unlike ROS 1, Navigation2 architecture leverages multi-core processors and the real-time, low-latency capabilities of ROS 2. This allows for more efficient processing and faster response times.
- **Algorithmic Refreshes:** Navigation2 focuses on modularity and smooth operation in dynamic environments. It includes the Spatio-Temporal Voxel Layer (STVL), layered costmaps, the Timed Elastic Band (TEB) controller, and a multi-sensor fusion framework for state estimation, Robot

 doi:10.20944/preprints202410.1204.v1

Localization. Each of these supports holonomic and non-holonomic robot types, a feature not as developed in ROS 1.

- **State Estimation:** Navigation2 follows ROS transformation tree standards for state estimation, making use of modern tools available from the community. This includes Robot Localization, a general sensor fusion solution using Extended or Unscented Kalman Filters. This is a more advanced approach compared to ROS 1.
- **Quality Assurance:** Navigation2 includes tools for testing and operations, such as the Lifecycle Manager, which coordinates the program lifecycle of the navigator and various servers. This manager steps each server through the managed node lifecycle: inactive, active, and finalized. This systematic approach to quality assurance is a significant upgrade from ROS 1.

Navigation2 builds on the successful legacy of ROS Navigation but with substantial structural and algorithmic refreshes. It is more suitable for dynamic environments and a wider variety of modern sensors, making it a more advanced and versatile navigation stack than its predecessor, ROS 1.

3.5. ROS 2 Security

The key difference in security between ROS and ROS 2 is that ROS 2 has been designed with security in mind from the ground up, whereas security was not a primary consideration in the initial development of ROS.

In ROS, security mechanisms such as authentication and encryption were not implemented by default, leaving systems vulnerable to potential security threats. However, ROS users could implement security measures manually by using third-party libraries and plugins.

ROS 2, on the other hand, has a built-in security framework that provides authentication, encryption, and access control by default. This framework enables ROS 2 to support secure communication between nodes and across networks, even when communicating with nodes that may not support security natively.

SROS2 (Secure ROS2) [23] is a security extension for ROS 2 that provides additional security features beyond those provided by the ROS 2 security framework. SROS is designed to address some of the limitations of the ROS 2 security framework and to provide enhanced security for critical robotic applications. SROS provides a more flexible and configurable access control system than the default ROS 2 permission system, allowing administrators to define fine-grained access policies for topics, services, and nodes. This can help to ensure that sensitive data and resources are protected from unauthorized access.

Overall, by implementing SROS in ROS 2, users can benefit from additional security features that are not available in ROS. This can help to ensure the security and integrity of critical robotic applications while also providing a more flexible and configurable security solution that can be tailored to the specific needs of each application.

3.6. Comparison of Platform Support in ROS and ROS 2

ROS (Robot Operating System) and its successor, ROS 2, are widely used frameworks for developing robotic systems. This section explores the level of support and compatibility with different operating systems in both frameworks. Specifically, we will examine the support for Windows, macOS, and Linux platforms, highlighting the advancements made in ROS 2.

- **Windows Support:** When it comes to Windows support, ROS has experimental compatibility, while ROS 2 boasts more comprehensive and reliable support for Windows 10. The enhanced support in ROS 2 allows developers to leverage the framework more effectively on Windows machines, ensuring a stable and efficient environment for building robust robotic applications.
- **macOS Support:** Both ROS and ROS 2 offer official support for macOS. However, ROS 2 surpasses its predecessor in terms of compatibility with the latest macOS versions. By capitalizing on the latest macOS features, such as the Metal graphics API, ROS 2 maximizes performance

 doi:10.20944/preprints202410.1204.v1

in specific applications. This advanced compatibility empowers developers to fully exploit the potential of macOS when constructing sophisticated robotics systems.
- **Linux Support:** Both ROS and ROS 2 offer comprehensive support for Linux, with official support for various popular distributions. However, ROS 2 takes a more modular and flexible approach, making porting the framework to different Linux distributions and architectures easier. This flexibility is particularly advantageous in heterogeneous computing environments, allowing developers to deploy ROS 2 on various Linux systems.

The utilization of the Data Distribution Service (DDS), which inherently possesses cross-platform capabilities, significantly facilitated the seamless compatibility of ROS 2 with Windows, macOS, and Linux. In contrast, ROS 1 relied on the specific TCPROS and UDPROS protocols, which were more tightly coupled with the Linux environment, limiting its cross-platform compatibility.

While both ROS and ROS 2 offer support for multiple operating systems, ROS 2 surpasses its predecessor regarding platform compatibility. With its comprehensive support for Windows 10, better compatibility with the latest macOS versions, and flexibility in porting to different Linux distributions and architectures, ROS 2 provides a more robust and versatile framework for building robotic systems. The technical advancements in ROS 2, such as the use of DDS and modular architecture, further enhance its compatibility and usability across different operating systems. Developers can leverage these advancements to create efficient and reliable robotics applications on their preferred platforms.

4. Literature Analysis on Key ROS 2 Design Areas

This section provides a detailed analysis of the current research surrounding critical design areas in ROS 2, including real-time capabilities, middleware, QoS, security, and support for embedded and distributed systems, as well as multi-robot systems (MRS). Each of these areas plays a crucial role in shaping the effectiveness and performance of ROS 2 for modern robotics applications. By examining the key challenges and recent advances in each domain, we aim to highlight the gaps in the current literature and ongoing efforts to address them. Table 2 offers a taxonomy of these areas, outlining the core issues and the technological innovations that are advancing the state of the art.

Following this, we delve into each domain, providing insights into the motivations driving research, the contributions made, and the existing limitations that researchers and developers continue to tackle.

4.1. ROS 2 Benchmarking

The evolution from ROS 1 to ROS 2 has successfully addressed crucial shortcomings in real-time communication, modularity, and support for multi-robot systems (MRS), catering to the increasing complexity of robotic environments. This shift has catalyzed a wave of scholarly investigation, focusing on rigorously evaluating ROS 2's refined features across a spectrum of use cases. In this section, we explore pivotal research articles that critically assess the core functionalities of ROS 2, as highlighted in Figure 2. These studies offer insights into the system's enhanced performance, demonstrating its potential and identifying areas for further enhancement.

4.1.1. ROS 2 Performance and Software Quality Benchmarking

Several studies, such as [16,18,24,25], explore and benchmark the overall performance of ROS 2 across various parameters. Specifically, [25] examines node composition within ROS 2, contrasting it with traditional multi-process systems. This benchmarking focuses on memory footprint, CPU usage, and latency, revealing that node composition, especially with intra-process communication (IPC), dramatically reduces memory usage by 33% and CPU load by 28%. Notably, the application of IPC in node composition lowers CPU usage by nearly an order of magnitude for larger messages and achieves latency reductions as low as 40 μs, closely aligning with the performance of monolithic applications.

 doi:10.20944/preprints202410.1204.v1

Furthermore, [18] conducts an empirical study evaluating the software quality of ROS2 Java projects using the PMD static analysis tool. The study assesses multiple coding standards, including security, performance, and code style. It identifies 33,533 alerts, with 62% related to code style issues, 7% to performance, and a smaller percentage related to error-prone issues. Additionally, the study reveals specific security concerns, such as insecure default settings in ROS2's DDS implementation, where key security features like encryption and signing were disabled.

4.1.2. ROS 2 Tools and Navigation Benchmarking

Studies such as [12,14] benchmark various tools and developments for ROS 2. For instance, [12] systematically evaluates five simulation software tools—Ignition, Webots, Isaac Sim, PyBullet, and Coppeliasim—focusing on robotic manipulation tasks like pick-and-place and throwing. The results indicate that no single tool excels in all areas. Webots and Ignition demonstrated the most stability and accuracy in Task 1, with Webots achieving a task success rate of 88% (without GUI) and using 144% CPU and 1191 MB RAM (162% CPU and 1322 MB RAM in GUI mode). Ignition showed similar performance, with a task success rate of 91% and consuming 202% CPU and 686 MB RAM (205% CPU and 775 MB RAM in GUI mode). Isaac Sim, while offering advanced features like machine learning integration, exhibited a high memory usage of 10,070 MB RAM and achieved a task success rate of 89% but failed to maintain consistency across runs.

Meanwhile, PyBullet and Coppeliasim were more resource-efficient but showed lower task success rates, with PyBullet achieving only 18% task success in Task 1 and using 117% CPU and 663 MB RAM (140% CPU and 919 MB RAM in GUI mode), and Coppeliasim showing a 92% success rate but consuming 135% CPU and 860 MB RAM (100% CPU and 850 MB RAM in GUI mode). For Task 2, Webots demonstrated better throwing consistency, with a cube movement success rate of up to 50%, while Coppeliasim had a 50% success rate but a higher failure rate of 32%. Isaac Sim exhibited an 83% success rate for cube movement but suffered from inconsistent motion behavior, impacting overall performance.

Similarly, [14] focuses on the ROS 2 Nav2 project, benchmarking path and trajectory planners across various robot types, including quadrupeds and Ackermann-steering vehicles. The study presents a comprehensive analysis of planners such as NavFn, Lazy Theta*-P, Smac 2D-A*, Smac Hybrid-A*, and Smac State Lattice, with Smac Hybrid-A* achieving the best performance in plan time (38.77 ms) and Lazy Theta*-P providing the shortest path length (50.28 meters). In terms of trajectory planners, benchmarks show Model Predictive Path Integral (MPPI) operating at 125 Hz and Regulated Pure Pursuit achieving frequencies above 4,000 Hz. Path smoothers, such as the Simple Smoother and Constrained Smoother, significantly improved path quality, with the Constrained Smoother yielding the best smoothness at 87.85. Additionally, state estimators like fuse demonstrated improved accuracy (with an error of 2.81 meters over a 542-meter route) compared to robot localization, though at a higher CPU cost (5.19% vs. 1.38%). These findings highlight significant improvements in ROS 2 over ROS 1, including enhanced flexibility, performance, and better support for modern robotic platforms. ROS 2 is shown to be well-suited for product-grade applications, and ongoing developments continue to push its capabilities forward, making it ideal for complex and large-scale robotics projects.

 doi:10.20944/preprints202410.1204.v1

Table 2. Summary of Key Challenges and Advances in ROS 2.

Research Area	Key Challenges	Advances
Security	Inflexible access control, DDS vulnerabilities, trade-offs between performance and security	ABAC frameworks, blockchain integration, CAESAR encryption, forensic investigation tools, optimized security configurations
Real-Time Systems	Callback scheduling unpredictability, poor scalability, network delays	Priority-based schedulers, formal response-time analysis, dynamic GPU management, containerized architectures
Middleware	Inefficient intra-node communication, DDS interoperability issues, poor performance in wireless environments	Dynamic DDS binding, Zenoh middleware for wireless environments, optimizations for distributed systems
Embedded and Distributed Systems	Real-time performance in lossy networks, scalability, resource constraints	FPGA-based acceleration, Micro-ROS for constrained environments, edge computing for distributed systems
Quality of Service (QoS)	Message loss with RELIABLE setting, latency-security trade-offs, lack of proactive QoS verification	Caching mechanisms, QoS balancing, dynamic QoS management for wireless networks, DSL for design-time QoS specification
Multi-Robot Systems (MRS)	Synchronization issues in heterogeneous environments, communication inefficiency under high network loads, real-time performance in resource-constrained systems	Velocity-aware middleware, optimized communication architectures, cache-control algorithms, Zenoh middleware for mesh networks

4.2. ROS 2 Security

As robotic systems are deployed in critical sectors such as healthcare, defense, and autonomous transportation, ensuring robust security in ROS 2 becomes vital. The transition to ROS 2 has brought improvements in communication and modularity, but security remains a critical challenge, particularly in terms of access control, communication integrity, and forensic capabilities. This section synthesizes recent advancements, limitations, and ongoing challenges in securing ROS 2 systems.

Challenges in ROS 2 Security: Access control and communication security are two key areas where significant challenges persist. The traditional role-based access control (RBAC) mechanisms, such as those implemented in SROS2, have been criticized for their rigidity and lack of scalability in complex, multi-robot environments. The fixed nature of RBAC often leads to inefficiencies in managing permissions dynamically, particularly in distributed systems [26,27]. Additionally, vulnerabilities in the Data Distribution Service (DDS) communication layer, such as unauthorized access, credential masquerading, and denial-of-service (DoS) attacks, have exposed ROS 2 to threats in high-risk environments [28,29].

Advances and Solutions: Addressing these challenges, research has focused on developing more flexible access control systems. Attribute-based access control (ABAC) frameworks, combined with blockchain technology, have emerged as a solution to overcome the limitations of RBAC by allowing fine-grained control over user permissions in dynamic environments [26,27]. In terms of communication security, there has been a shift towards adopting stronger encryption methods. Studies have evaluated the effectiveness of CAESAR encryption algorithms like Ascon and Deoxys-II, which offer enhanced security without significantly compromising performance [30]. These encryption protocols have been particularly valuable in securing UAV swarms and multi-robot systems operating over wireless networks, where latency is a critical factor [31].

 doi:10.20944/preprints202410.1204.v1

Another area of progress is forensic investigation and anomaly detection. Tools such as ROS2Tester have been developed to provide runtime verification and vulnerability detection, enabling better monitoring of distributed systems [32]. However, research continues to highlight gaps in post-attack forensic capabilities, particularly in recovering tampered data in collaborative robotic environments [33]. These gaps point to a need for more sophisticated runtime verification techniques, such as the POLAR-Express framework for neural network-driven systems, which enhances anomaly detection and system introspection [34].

Open Challenges and Future Directions: A significant challenge remains the trade-off between security and system performance. Studies consistently show that enabling security features—such as encryption and authentication—introduces latency and reduces throughput, particularly in wireless and resource-constrained environments [35,36]. Moreover, ROS 2 lacks a holistic security framework that addresses vulnerabilities across all layers of the system, from hardware to application-level security. Research in this area is focusing on multi-layered security architectures that provide comprehensive protection without compromising performance [17].

While considerable advancements have been made in ROS 2 security, there is still a need for more adaptive and scalable solutions. Future research should focus on improving the balance between security and performance, particularly in real-time and resource-constrained environments. Additionally, enhancing forensic tools and anomaly detection mechanisms will be critical for ensuring the long-term security and reliability of ROS 2 systems. Table 3 summarizes the key challenges and advancements discussed.

Table 3. Summary of Security Research in ROS 2.

Security Area	Challenges	Advances
Access Control	Inflexible RBAC, lack of scalability	ABAC frameworks, blockchain-based solutions [26,27]
Communication Security	DDS vulnerabilities, weak encryption	CAESAR encryption algorithms, secure UAV communication [30,31]
Forensic Investigation	Lack of post-attack recovery tools	ROS2Tester for runtime verification, POLAR-Express for anomaly detection [32,34]
Performance vs Security	Increased latency with security features	Optimized cryptographic practices [35,36]
Anomaly Detection	Inadequate tools for real-time anomaly detection	Basic AD systems, improvements in introspection [37,38]
Holistic Security Framework	Incomplete security coverage across layers	Multi-layered security frameworks [17,39]

4.3. ROS 2 Real-Time Systems

As robotic systems evolve in complexity, real-time capabilities become crucial, especially for applications such as autonomous vehicles, multi-agent systems, and safety-critical robotic systems. Robot Operating System 2 (ROS 2) has gained attention for its potential to meet these demands through its support for Data Distribution Service (DDS) middleware and improvements over ROS 1 in scheduling and real-time operations. However, ROS 2 still faces challenges in ensuring predictable performance, reducing latency, and maintaining real-time constraints in diverse operating conditions. This section provides an aggregated discussion of the advancements, limitations, and open challenges in ROS 2 real-time systems based on recent literature.

Challenges in ROS 2 Real-Time Systems: The primary challenges stem from the unpredictability in callback scheduling, task prioritization, and worst-case response time (WCRT) guarantees. Many works identify that ROS 2's standard executors lack the necessary mechanisms for efficient real-time

 doi:10.20944/preprints202410.1204.v1

execution, particularly in distributed and multi-threaded environments. The default callback execution order and lack of priority mechanisms result in high execution-time variability, causing issues like missed deadlines and buffer overflows [40–42]. Moreover, ROS 2 suffers from performance degradation under stress and scalability issues when applied to complex, distributed setups [43,44].

Advances and Solutions: A key trend in addressing these challenges has been the development of customized schedulers and executors. Notable efforts include the introduction of priority-based scheduling frameworks such as PiCAS, which incorporates callback prioritization and resource allocation to reduce end-to-end latency [45,46]. Other solutions focus on formal real-time analysis frameworks to provide tighter bounds on WCRT for multi-threaded systems, helping designers guarantee that critical tasks meet their deadlines [47,48].

Another significant advance is the exploration of containerization and microservice architectures for improving real-time performance in ROS 2, particularly for SDVs and autonomous systems [49,50]. These studies have shown that containerized deployments can lead to better latency management and system resource utilization compared to bare-metal configurations. Furthermore, dynamic GPU management frameworks such as ROSGM offer improved processing efficiency, especially for tasks that require intensive computational power [51].

Open Challenges and Future Directions: Despite these advancements, several gaps remain. Current solutions often fail to provide scalable, flexible, and holistic real-time performance for highly distributed systems, where network delays and jitter introduce significant unpredictability [52,53]. Furthermore, while various priority-driven schedulers have been proposed, integrating these into existing ROS 2 frameworks without introducing additional overhead remains a challenge. There is also a growing need for tools that can provide fine-grained runtime monitoring and online latency management to ensure real-time constraints are continuously met in dynamic conditions [51,54].

The development of real-time systems in ROS 2 has made significant strides, but further advancements are needed to overcome scalability, flexibility, and overhead challenges. Future work should focus on enhancing distributed real-time performance, integrating more adaptive scheduling frameworks, and developing robust online monitoring tools. Table 4 summarizes the key trends, challenges, and solutions discussed.

Table 4. Summary of Real-Time Research in ROS 2.

Research Area	Challenges	Advances
Callback Scheduling	Unpredictability in execution order, lack of priority handling	PiCAS, priority-based schedulers [45,46]
Multi-Threaded Executors	Poor response time guarantees, high variability	Formal response-time analysis frameworks [44,47]
Distributed Real-Time Systems	Network delays, jitter, and scalability issues	Dynamic GPU management, ROSGM [51]
Containerization	Latency management, system resource utilization	Microservice architectures, containerized SDVs [49,50]

4.4. ROS 2 Middleware

The middleware layer in ROS 2 plays a crucial role in enabling efficient communication between distributed nodes in robotic systems. The Data Distribution Service (DDS) is the default middleware in ROS 2, providing real-time data-centric communication. However, challenges remain, especially in optimizing inter-node communication, handling varying Quality of Service (QoS) requirements, and improving reliability in complex and resource-constrained environments. This section discusses key advancements and challenges in middleware technologies within ROS 2, focusing on recent literature that explores optimizations, alternative middleware, and mechanisms to enhance communication performance.

Challenges in DDS Implementation: Despite the advantages of DDS in enabling real-time communication, it has limitations in scenarios involving intra-node communication, heterogeneous

 doi:10.20944/preprints202410.1204.v1

environments, and dynamic QoS requirements. Research has identified inefficiencies when ROS 2 relies on network-based communication mechanisms, even for local interactions, leading to unnecessary latency and overhead [55]. To address this, studies have proposed dynamic DDS binding mechanisms that allow switching between DDS implementations based on the communication characteristics of each node [56]. This approach optimizes resource usage and reduces latency by selecting the most appropriate communication method for each task.

Interoperability and Security Concerns: Another significant area of concern is the interoperability between different DDS implementations. While ROS 2 supports multiple DDS vendors (e.g., Fast DDS, Cyclone DDS, RTI Connext), studies highlight challenges when different implementations are used simultaneously, particularly with security features enabled [57]. The performance overhead introduced by security configurations varies significantly across DDS vendors, requiring careful consideration in critical systems. These studies suggest that ensuring seamless interoperability and minimizing security-related overheads remain open challenges.

Zenoh Middleware and Dynamic Switching: An important milestone in ROS 2's communication capabilities was the introduction of Zenoh in the Jazzy distribution. Zenoh offers a novel approach to ROS 2 communication, particularly in environments where traditional DDS may face challenges. The 'rmw_zenoh_cpp' middleware interface maps the ROS 2 RMW API onto Zenoh APIs, enabling ROS 2 to utilize Zenoh for data exchange. This integration is depicted in Figure 7, which illustrates the structure of the middleware and how Zenoh sessions interact with each other and with the Zenoh router. Notably, Zenoh sessions rely on peer-to-peer communication for data transfer while using a Zenoh router for discovery and gossip scouting. This architecture provides flexibility and performance advantages, particularly in distributed systems with varying network conditions.

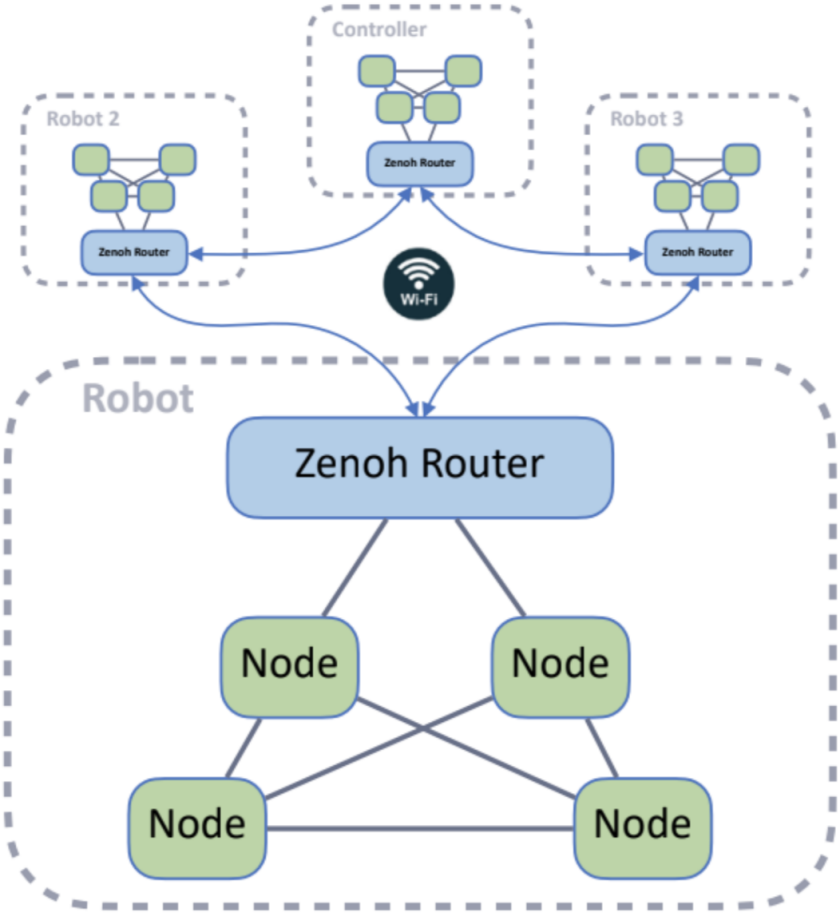

Figure 7. Default Configuration for Zenoh Sessions in ROS 2 Jazzy.

 doi:10.20944/preprints202410.1204.v1

The dynamic switching mechanism further enhances performance by allowing ROS 2 systems to switch between different middleware protocols based on network conditions and application demands. For instance, Zenoh excels in wireless environments, where it outperforms DDS in terms of latency and throughput, especially in edge-to-cloud communication [58].

While DDS remains central to ROS 2 middleware, alternative solutions like Zenoh and dynamic DDS switching mechanisms address the limitations posed by static middleware configurations and challenging network environments. Future research should focus on refining these dynamic approaches, improving security interoperability, and expanding middleware support for diverse real-time applications. Table 5 summarizes the key trends, challenges, and solutions discussed.

Table 5. Summary of Middleware Research in ROS 2.

Research Area	Challenges	Advances
Intra-node Communication	Inefficiencies in local communication	IPC-based DDS implementations, dynamic DDS binding [55,56]
Interoperability	DDS vendor incompatibilities, security overhead	Evaluations of different DDS implementations [57]
Alternative Middleware	Limited performance in wireless/cloud environments	Zenoh middleware for ROS 2, dynamic middleware switching [58]
Real-time Performance	High latency in distributed systems	Middleware optimizations (e.g., Zenoh, PubSubBinder) [55,58]

4.5. ROS 2 Embedded Systems and Distributed Systems

The integration of ROS 2 into embedded and distributed systems has gained significant attention in recent years, primarily due to the growing demand for scalable, real-time, and efficient robotic systems. Embedded systems, such as those found in autonomous vehicles, space missions, and IoT devices, present unique challenges due to their constrained resources and real-time requirements. Distributed systems, on the other hand, necessitate efficient communication protocols, particularly when leveraging edge-cloud technologies or containerized environments. This section discusses key advances in these areas within the ROS 2 framework, highlighting the contributions and addressing critical gaps in the literature.

Trends in Embedded Systems: One prominent trend in embedded systems is the adaptation of ROS 2 for real-time, safety-critical applications such as automotive and aerospace systems [59,60]. Efforts like Apex.OS extend ROS 2's capabilities to meet the stringent requirements of automotive-grade environments by integrating best practices from traditional microcontroller-based systems like OSEK. Similarly, the development of Micro-ROS enables ROS 2 to be scaled down for resource-constrained platforms like CubeSats, demonstrating the framework's adaptability across various hardware configurations [60].

Moreover, hardware acceleration, particularly using FPGAs, has emerged as a key solution to overcome the limitations of traditional CPU-based implementations. ROS 2-centric architectures that facilitate the integration of FPGAs, such as fpgaDDS and ReconROS, address the inefficiencies in communication between hardware and software nodes [61,62]. These solutions significantly enhance real-time performance by leveraging parallel processing and reducing communication bottlenecks, enabling low-latency, high-speed operations in robotics applications.

Advances in Distributed Systems: The shift towards edge-cloud architectures in robotics is another significant trend, with ROS 2 playing a pivotal role in enabling distributed robotic systems [63]. Technologies like containerization and orchestration, using platforms such as Docker and Kubernetes, allow for scalable, platform-agnostic deployment of robotic applications. However, ROS 2's communication protocols face challenges in such environments, particularly concerning real-time performance and scalability. Recent research proposes optimization frameworks and

 doi:10.20944/preprints202410.1204.v1

orchestration-aware communication protocols to address these limitations [64]. Additionally, edge computing has proven essential in distributed autonomous driving systems, offloading computational tasks from vehicles to external edge units, thus enabling higher levels of autonomy without the need for expensive on-board resources [65].

Challenges and Future Directions: Despite these advances, several challenges remain. The scalability of ROS 2 in lossy, wireless networks, particularly in the context of unmanned assets, continues to be a limitation, with research showing increased latency and message loss when security features are enabled [66]. The integration of ROS 2 with IoT devices also faces resource constraints, although progress has been made with lightweight operating systems like RIOT [67]. Furthermore, while hardware acceleration shows promise, optimizing the dynamic mapping between hardware and software nodes, especially in event-driven environments, remains a challenge.

Significant strides have been made in adapting ROS 2 for embedded and distributed systems, but challenges in real-time performance, scalability, and resource optimization persist. Future work should focus on refining dynamic communication protocols, improving interoperability across hardware platforms, and further optimizing the integration of edge computing and orchestration in distributed robotic systems. Table 6 summarizes the key trends, challenges, and solutions discussed.

Table 6. Summary of Key Advances and Challenges in ROS 2 Embedded and Distributed Systems.

Research Area	Challenges	Advances
Real-Time Performance	Scalability in lossy networks, message latency	Micro-ROS for constrained environments, FPGA-based acceleration [60,61]
Distributed Systems	Interoperability, edge-cloud integration	Edge computing, orchestration-aware communication protocols [63,64]
Hardware Acceleration	Dynamic mapping of nodes, event-driven execution	fpgaDDS, ReconROS executor for FPGA acceleration [62]

4.6. ROS 2 Quality of Service (QoS)

As robotic systems increasingly operate in complex, dynamic environments, managing Quality of Service (QoS) in ROS 2 has emerged as a critical research focus. QoS in ROS 2 is essential for ensuring reliable communication between nodes, particularly in real-time applications like multi-robot coordination, remote surgery, and autonomous driving. While ROS 2 introduces flexible QoS profiles to manage data transmission, several challenges persist in optimizing performance, reliability, and security across varied network conditions. This section synthesizes recent research trends, addressing both the advancements and the ongoing limitations in ROS 2 QoS management.

Challenges in ROS 2 QoS: A primary challenge is the reliability of the communication framework, especially when employing the RELIABLE QoS setting. Despite its intention to guarantee message delivery, this setting often struggles in high-data-rate environments or under constrained buffer conditions, leading to message loss in multi-robot systems and aggregated processing architectures [68,69]. Additionally, balancing QoS parameters like DEADLINE and DEPTH can be difficult, particularly in real-time applications where both reliability and latency are critical.

Advances and Solutions: To address these issues, research has introduced caching mechanisms that store redundant data locally to reduce communication load and mitigate message loss [70]. These systems have shown promise in improving overall performance by optimizing data transmission efficiency, especially in environments with high communication demands. Moreover, careful balancing of QoS parameters has been shown to significantly enhance system reliability and timeliness, especially in multi-robot setups.

Another important focus has been the trade-off between QoS and security. ROS 2's use of the DDS standard introduces robust security features, but these features often introduce additional latency, especially in mission-critical environments like unmanned military vehicles [71]. Research in this

 doi:10.20944/preprints202410.1204.v1

area emphasizes the need to optimize security-aware QoS configurations, where encryption and authentication protocols are balanced with performance requirements to minimize communication delays without compromising security.

Design-Time QoS Specification: Another significant development is the move toward design-time QoS specification. Traditional ROS 2 systems primarily check QoS profiles at runtime, leaving room for unexpected performance issues. To mitigate this risk, researchers have proposed domain-specific languages (DSLs) to formalize QoS requirements before deployment. This proactive approach ensures that systems meet performance and reliability expectations in varying conditions, particularly in multi-agent systems with fluctuating network environments [72].

QoS in Wireless Networks: The challenges of managing QoS in wireless networks, such as those used in remote driving and teleoperated systems, have also been widely studied. Wireless networks introduce additional latency and data interference, necessitating more dynamic QoS management strategies. Solutions like WiROS dynamically adjust network parameters, such as Enhanced Distributed Channel Access (EDCA), to prioritize critical data flows in congested network environments, ensuring that latency-sensitive applications continue to perform effectively [73].

Open Challenges and Future Directions: Despite significant progress in QoS management, challenges remain in balancing latency, reliability, and security in diverse deployment scenarios. Research should continue to explore adaptive QoS strategies that incorporate real-time feedback from the network and optimize security mechanisms to minimize their impact on system performance. Future work should also focus on extending dynamic QoS management strategies to better support complex, distributed systems.Table 7 provides a summary of the key challenges and solutions discussed in ROS 2 QoS research.

Table 7. Summary of Key Advances and Challenges in ROS 2 QoS Optimization.

Research Area	Challenges	Advances
RELIABLE QoS Limitation	Message loss, buffer size constraints	Caching mechanisms, QoS balancing (DEADLINE and DEPTH) [68,70]
QoS and Security Trade-offs	Latency overhead from security features	Optimized security-aware QoS configurations [71]
Design-Time QoS Specification	Lack of proactive QoS verification	DSL for formal specification of QoS requirements [72]
Wireless Networks QoS	Latency and data interference	WiROS for dynamic QoS management over WiFi [73]

4.7. ROS 2 Multi-Robot Systems (MRS)

The development of multi-robot systems (MRS) within ROS 2 has led to significant advancements in communication flexibility and real-time coordination, largely enabled by the Data Distribution Service (DDS) protocol. Despite these improvements, challenges such as synchronization, communication efficiency, and real-time performance remain key concerns, particularly in heterogeneous environments where robots, such as unmanned aerial vehicles (UAVs) and ground vehicles (UGVs), interact under varying network conditions. This section provides a synthesized discussion of the progress, limitations, and future directions in MRS research within ROS 2.

Challenges in ROS 2 MRS: Synchronization between multiple robots in heterogeneous systems presents a primary challenge, especially when integrating different simulators or real-world robots in real-time, masterless environments. Traditional synchronization techniques from ROS 1 are insufficient, often resulting in high latency and packet loss. These issues are particularly pronounced in scenarios involving varying communication protocols and robot velocities, such as UAV-UGV collaboration [74]. Additionally, communication architecture inefficiencies, especially in centralized multi-agent systems, further complicate performance. The choice between one-to-one and many-to-one communication

 doi:10.20944/preprints202410.1204.v1

architectures can significantly impact key performance metrics, such as data age and data miss ratio [75].

Advances and Solutions: Research has proposed novel middleware solutions aimed at improving synchronization and reducing latency. For example, velocity-aware middleware has been introduced to optimize the coordination between heterogeneous robots, minimizing packet loss and improving real-time communication [74]. In centralized multi-agent systems, studies have highlighted the importance of selecting appropriate communication architectures to optimize system robustness and communication efficiency. Comparative studies of DDS vendors, such as CycloneDDS and FastDDS, provide valuable insights for optimizing real-time communication under heavy network loads [75].

Another key advancement is in managing real-time performance under constrained resources, particularly in low-cost robotic systems. Aggregated processing architectures and cache-control algorithms have shown to reduce latency and prevent data processing failures by offloading computational tasks to centralized environments. Experimental results demonstrate significant improvements in communication efficiency and system reliability, even in resource-constrained settings [76].

Middleware for Extreme Environments: The use of ROS 2 middleware in extreme environments, such as planetary exploration, has further expanded the field's understanding of middleware performance in dynamic mesh networks. Zenoh, a ROS 2 middleware implementation, has shown superior performance in delay reduction and CPU efficiency, making it a suitable candidate for space exploration missions where reliable and efficient communication is critical [77].

Open Challenges and Future Directions: Despite these advancements, several challenges remain in optimizing middleware, communication architectures, and real-time performance in MRS. The scalability of middleware solutions and the integration of adaptive QoS strategies are crucial areas that require further exploration. Additionally, enhancing the interoperability of heterogeneous robotic systems and improving real-time synchronization across diverse environments will be critical to advancing MRS research. Table 8 provides a summary of the key challenges and solutions discussed in ROS 2 MRS research.

Table 8. Summary of Key Challenges and Advances in ROS 2 MRS.

Research Area	Challenges	Advances
Synchronization in Heterogeneous Systems	High latency, packet loss	Velocity-aware middleware, real-time synchronization [74]
Communication Architecture Efficiency	Data age, data miss ratio, high network load	One-to-one communication, DDS vendor performance optimization [75]
Real-Time Performance in Low-Cost Systems	Latency, data processing failures	Aggregated processing, cache-control algorithms [76]
Middleware in Extreme Environments	Dynamic mesh network reliability	Zenoh for reduced delay, CPU efficiency [77]

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 October 2024 doi:10.20944/preprints202410.1204.v1

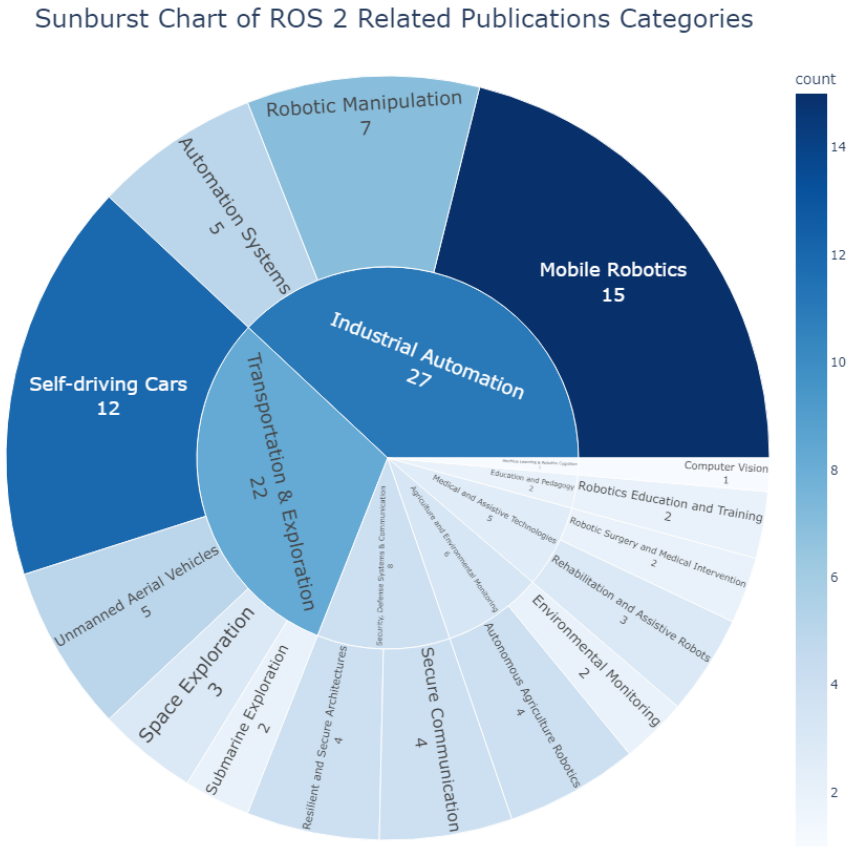

Figure 8. Taxonomy of different fields that utulizes ROS 2.

Name	Programming language	API support	Physics Engine	CAD files support	Multithreading enabled?	Family of robots	Supported actuators	ML Support	Headless Support	Open Source	Supported sensors
Gazebo	C++, Python	C++, Python	ODE Bullet Simbody DART	SDF/URDF OBJ STL COLLADA (.dae)	Yes	Supports a wide range of robots, including wheeled robots, legged robots, and drones.	Motors (position and velocity), servos, and force-torque actuators.	External	Full	Yes	Camera Sensor, Depth camera sensor, Distance and Proximity Sensors, Laser, Force Sensors
Webots	C, C++, Python, Java, MATLAB	C++, Python, ROS 2 API	ODE	WBT, VRML, X3D	Yes	Supports a wide range of robots, including humanoids, drones, wheeled robots, and custom designs.	Brake, Connector, Display, Emitter, Linear Motor, Muscle, Pen, Propeller, Rotational Motor, Speaker, Track (Conveyor Belt or tank robots)	External	Partial	Yes	Accelerometer, Camera, Compass, Distance Sensor, GPS, Gyro, Lidar, Position Sensor, Receiver, Touch Sensor
CoppeliaSim	Lua	C, C++, Python, Java, Urbi, Matlab, Octave	ODE, Bullet, Vortex, Newton	OBJ, STL, DXF, 3DS, Collada, URDF	No	Mobile, Humanoid, Industrial	Revolute (and some variants), Prismatic and Spherical Motors	External	Full	No	Vision Sensor Force Sensor Proximity Sensor Accelerometers Gyro Lasers
Unity	C#	C#	NVidia PhysX, Box2D	FBX, Collada, DXF, OBJ	Yes	Mobile, Humanoid, Industrial	Revolute and Prismatic Joints	External	Full	No	Cameras
Ignition	C++, Python	C++, Python	DART, Bullet, TPE, and ODE.	STL, COLLADA (.dae), and OBJ.	Yes	Supports a wide range of robots, including wheeled robots, humanoids, drones, and more.	Motors (velocity, position, and torque control), servos, and custom actuators.	External	Full	Yes	Cameras, LIDAR, IMUs, GPS, sonar, force-torque sensors, and many other sensors through the use of plugins and sensor models.
Isaac Sim	Python, C++	Python API, C++, and ROS 2 API integration	PhysX	URDF, OBJ, FBX, USD (Universal Scene Description)	Yes	Focuses on industrial robots, mobile robots, manipulators, drones, and custom robot designs.	Motors, servos, position, velocity, and force control actuators.	Integrated	Full	No	Cameras, LIDAR, IMUs, GPS, force-torque sensors, and many others through the extensive sensor model support in Isaac Sim.
PyBullet	Python, C++	Python API (primarily), C++	Bullet	URDF, SDF, STL and OBJ.	No	Supports a variety of robots, especially manipulators and mobile robots, often used for robotics research and reinforcement learning.	Motors (position, velocity, and torque control), force actuators, and custom actuators.	External	Full	Yes	Cameras, LIDAR, IMUs, force sensors, and custom sensor implementations through scripting.
Mujoco	C, Python (via bindings)	C API, Python API (via bindings like mujoco-py)	Mujoco	URDF and MJCF, OBJ and STL	No	Focuses on manipulators, humanoids, legged robots, and custom robots, widely used in research on robotics and control.	Motors (position, velocity, and torque control), muscle models, and force/torque actuators.	External	Full	Yes	IMUs, force-torque sensors, touch sensors, cameras, and various custom sensors through scripting and configuration files.

Figure 9. ROS 2 Simulators.

5. Literature Analysis on Key ROS 2 Frameworks and Toolkits

The adaptability and extensive ecosystem of ROS 2 have facilitated the development of numerous frameworks and toolkits, enhancing its utility across a variety of applications. Key frameworks such as Nav2, MoveIt, and Autoware exemplify ROS 2's capability to handle complex tasks in navigation, manipulation, and autonomous driving, respectively. These frameworks provide robust and flexible

 doi:10.20944/preprints202410.1204.v1

solutions for path planning, motion control, and vehicle autonomy, significantly advancing the state of robotic software development.

Nav2, for instance, is a comprehensive navigation framework that supports modular and scalable navigation solutions for mobile robots. It leverages ROS 2's advanced middleware to deliver reliable path planning and obstacle avoidance in dynamic environments. MoveIt extends ROS 2's capabilities to robotic manipulation, offering tools for motion planning, control, and 3D perception. Autoware, on the other hand, focuses on autonomous driving, providing a full-stack solution for self-driving vehicles, including perception, planning, and control. Table 10 shows different libraries and their application.

 doi:10.20944/preprints202410.1204.v1

Table 9. ROS 2 Open Source Packages.

Category	Citation
Multi Robotic Systems	Aerostack2 [78], CrazyChoir [10], KubeROS [79], ROS2SWARM [80], The Cambridge RoboMaster [81], Toychain [82], TestbedROS2Swarm [83], ChoiRbot [84], ROS2BDI [85],
Cooperative Robotics & HRI	ChoiRbot [84], ROSGPT [9], opendr [86,87], NAO [88], qml_ros2_plugin [89], PointIt [90], ros2-foxy-wearable-biosensors [91]
Simulators	MVSim [92], HuNavSim [93], LGSVL Simulator [94], LunarSim [95], MAES [96], UUV simulator [97]
Computer Vision	HawkDrive [98], ROSGPT_Vision [99], GLIM [100], UAV Volcanic Plume Sampling [101], YOLOX [102], direct_visual_lidar_calibration [103], Bridging 3D Slicer and ROS2 [104], Video Encoding and Decoding for High-Definition Datasets [105],
Reinforcement Learning	ros2-forest [106], gym-gazebo2 [107], drl_grasping [108], LPAC [109], opendr [86,87], ros2learn [110], An Educational Kit for Simulated Robot Learning in ROS 2 [111], [112]
Performance Evaluation	ChoiRbot [84], [113], FogROS2 [114], DriveEnv-NeRF [115], RobotPerf [116]
Real-Time	CARET [113,117], ros2_tracing [118]
Cyber Security	Bobble-Bot [119], Hyperledger Fabric Blockchain [120], rvd [121], KISS-ICP [122], RCTF [123], SROS2 [124],
Software Platforms	Aztarna [125], CFV2 [126], SkiROS2 [127], SMARTmBOT [128], Space ROS [129], ros2-3gppSA6-mapper [130]
State Estimation & Prediction	MixNet [131], FusionTracking [132], NanoMap [133], wayp [134], lidar_cluster_ros2 [135]
Planning	navigation2 [136], PlanSys2 [137], SAILOR [138], YASMIN [139]
Navigation	mola [140], depth_nav_tools [141], nav2_accountability_explainability [142], Navigation Approach based on Bird's-Eye View [143], DeRO [144], vox_nav [145,146], evo [147], FlexMap Fusion [148], Mobile MoCap [149], MOCAP4ROS2 [150], Multi-Robot-Graph-SLAM [151], pointcloudset [152], flexible_navigation [153], The Marathon 2 [22],
Embedded & Distributed Systems	embeddedRTPS [154], forest [155], ReconROS [156,157], FogROS [158], FogROS2 [159–161], ros2-message-flow-analysis [162], PAAM [163], RobotCore [164]
UAV	anafi_ros [165], CrazyChoir [10], UAV Volcanic Plume Sampling [101], HyperDog [166], Aerostack2 [78], MPSoC4Drones [167]
UUV	SUAVE [168], Angler [169], UUV simulator [97]
Self-driving Cars	Autoware_Perf [170], DriveEnv-NeRF [115], XTENTH-CAR [171]
Service Robots	MERLIN & MERLIN2 [172,173],
Product Integration	libiiwa [174], HRIM [175], kmriiwa [176], LBR-Stack [177], MeROS [178], OtterROS [179], RCLAda [180], RoboFuzz [181], Wrapyfi [112]

 doi:10.20944/preprints202410.1204.v1

Table 10. ROS 2 Meta-Libraries and Their Applications.

ROS 2 Meta-Library	Application
MoveIt 2	Motion planning for manipulators and humanoid robots
Navigation2 (Nav2)	Autonomous navigation for wheeled and legged robots
Perception (PCL, OpenCV)	3D perception, object recognition, and environment mapping
TF2 (Transform Library)	Coordinate transformations between multiple frames for robots
Gazebo	Physics-based simulation for robotic systems and environments
RTI Connext DDS	Middleware for real-time communication in distributed robotic systems
ros_control	Hardware abstraction, low-level control, and hardware interfaces for robots
BehaviorTree.CPP	Task and decision-making frameworks for autonomous robots
ROS2 Bag	Data recording and playback for debugging and analysis
Rviz	3D visualization tool for robot models and sensor data
Autoware.Auto	Autonomous driving stack for self-driving cars
CyberRT	Autonomous driving framework developed by Apollo for vehicle control and sensor fusion
Cartographer	Real-time 2D and 3D SLAM (Simultaneous Localization and Mapping)
Micro-ROS	ROS 2 for microcontrollers, used in embedded systems and resource-constrained robots
MAVROS	Communication interface between ROS 2 and MAVLink-based drones (e.g., multi-rotor UAVs)
Underwater ROS (UUV Simulator)	Simulation and control of underwater vehicles
Quadruped ROS (ROS 2 Quadruped)	Framework for controlling four-legged robotic platforms
SROS2 (Secure ROS 2)	Security features like authentication, encryption, and access control for ROS 2 systems
Fast-RTPS	Real-time communication middleware for distributed robotic systems (DDS implementation)
Lifecycle	Lifecycle management for ROS 2 nodes, handling transitions between states
SMACC	State machine framework for robot decision-making in ROS 2
ROS 2 Control	Advanced control framework for managing robot hardware and drivers
PlotJuggler	Real-time data visualization and plotting tool for ROS 2
Foxy Simulator (Foxglove)	Browser-based visualization tool for exploring robotic data
DDS-Security	Security plugins for encryption, authentication, and access control in DDS
Rosbridge	JSON API for ROS 2, enabling interaction with web interfaces and mobile devices
Orocos	Real-time control framework for complex robot control tasks, integrated with ROS 2
RViz2 Plugins	Custom plugins for extended visualization of data in RViz2

Simulation and visualization tools are integral to the development and testing of robotic systems. Gazebo and Webots are widely used simulators that integrate seamlessly with ROS 2, allowing for high-fidelity simulation of robots and environments. Gazebo supports complex physics simulations and is extensively used for testing algorithms in a controlled setting. Webots provides a user-friendly interface and supports a wide range of robotic platforms, making it ideal for both educational and research purposes. Figure 9 shows different simulators for ROS 2 along with different specifications.

Moreover, datasets generated using ROS 2 play a crucial role in training and validating machine learning models and other data-driven approaches in robotics. For example, Merzlyakov and

 doi:10.20944/preprints202410.1204.v1

Macenski [182] compared modern visual SLAM approaches, while Amano et al. [183] generated datasets for object recognition using FPGA nodes in ROS 2. Hallyburton et al. [184] created a multi-agent security testbed, and Rosende et al. [185] provided an urban traffic dataset for intelligent transportation systems. These datasets are crucial for benchmarking and improving the performance of various robotic applications.

Table 11. Datasets and Tools Generated by ROS 2.

Title	Authors	Description
An Urban Traffic Dataset Composed of Visible Images and Their Semantic Segmentation Generated by the CARLA Simulator	Rosende et al. [185]	Urban traffic dataset for training computer vision algorithms
Are you a robot? Detecting Autonomous Vehicles from Behavior Analysis	Maresca et al. [186]	Dataset and framework for detecting autonomous vehicles
Co-driver: VLM-based Autonomous Driving Assistant with Human-like Behavior and Understanding for Complex Road Scenes	[187]	VLM dataset for Understanding for Complex Road Scenes
HawkDrive: A Transformer-driven Visual Perception System for Autonomous Driving in Night Scene	Guo et al. [98]	Visual perception system for night-time autonomous driving
Learning to Grasp on the Moon from 3D Octree Observations with Deep Reinforcement Learning	[108]	simulation environment with procedurally-generated datasets is created to train agents under challenging conditions in unstructured scenes with uneven terrain and harsh illumination
Multimodal Mobile Robotic Dataset for a Typical Mediterranean Greenhouse: The GREENBOT Dataset	[188]	dataset designed explicitly for challenging agricultural environments
Race Against the Machine: A Fully-Annotated, Open-Design Dataset of Autonomous and Piloted High-Speed Flight	[189]	Open-Design Dataset of Autonomous and Piloted High-Speed Flight
ROSPaCe: Intrusion Detection Dataset for a ROS2-Based Cyber-Physical System and IoT Networks	Puccetti et al. [190]	Intrusion detection dataset for ROS 2-based systems
Safe Road-Crossing by Autonomous Wheelchairs: a Novel Dataset and its Experimental Evaluation	Grigioni et al. [191]	Dataset for safe road-crossing decisions by autonomous wheelchairs
Skeleton Tracking Based Complex Human Activity Recognition Using Kinect Camera	Anjum et al. [192]	Dataset for human activity recognition using Kinect camera
Swarm-SLAM: Sparse Decentralized Collaborative Simultaneous Localization and Mapping Framework for Multi-Robot Systems	Lajoie, Beltrame [193]	C-SLAM framework for multi-robot systems
Survey on Datasets for Perception in Unstructured Outdoor Environments	[194]	compare publicly available datasets available in unstructured outdoor environments

Simulation tools also play a critical role in the development and testing of ROS 2 applications. Gazebo, for example, is frequently used for simulating complex robotic scenarios with high-fidelity physics, making it invaluable for testing navigation, manipulation, and sensor integration. Webots, another popular simulator, provides a comprehensive environment for developing and testing robots, supporting a wide array of sensors and actuators. Additionally, the CARLA simulator has been used to generate urban traffic datasets, which are essential for training autonomous driving systems [185].

Frameworks and toolkits provided by the literature further extend the capabilities of ROS 2. For instance, FogROS2 by Ichnowski et al. [159] enables cloud and fog robotics, enhancing computational capabilities and resource utilization. MERLIN2 by González-Santmarta et al. [173] offers a hybrid cognitive architecture for symbolic planning and decision-making in service robots. Wrapyfi by Abawi et al. [112] provides a Python wrapper for integrating robots, sensors, and applications across multiple middleware platforms.

 doi:10.20944/preprints202410.1204.v1

Table 9 shows a list of open source packages for different applications and research fields.

6. Literature Analysis on ROS 2 Fields of Applications

The versatility of ROS 2 has enabled its deployment across a wide array of robotic platforms and applications, demonstrating its adaptability and robustness in diverse scenarios such as autonomous vehicles, unmanned aerial systems (UAVs), unmanned underwater vehicles (UUVs), healthcare, industrial automation, and human-robot interaction (HRI). This section highlights several applications from the literature, showcasing the breadth of ROS 2's capabilities. Figure 8 shows an approximate taxonomy of the research applications of ROS 2.

ROS 2 has shown significant advancements in the field of autonomous vehicles, facilitating seamless integration of sensors, decision-making algorithms, and vehicle control mechanisms. Studies such as those by Lu et al. [195] and Maresca et al. [186] have highlighted its critical role in intelligent and cooperative driving, as well as behavior analysis for differentiating autonomous vehicles from human-driven ones. Similarly, the field of UAVs has benefited from ROS 2's capabilities, with works by Park et al. [196] and Chen et al. [197] showcasing advancements in real-time path planning and multi-agent coordination.

In the realm of UUVs, Djizi et al. [198] demonstrated the potential of ROS 2 in hand gesture control for real-time operations, reflecting broader efforts to enhance intuitive control mechanisms in underwater environments. Additionally, the incorporation of digital twin technology for drone forensics by Almusayli et al. [199] represents a novel approach to enhance investigation accuracy and efficiency.

Healthcare robotics has also seen innovative applications of ROS 2, such as the knee rehabilitation system by Arcos et al. [11] and the autonomous floor-cleaning robot with UV sterilization by Mohan and Krishnan [200]. These studies underscore ROS 2's robustness in ensuring precise control and navigation in medical environments. In industrial automation, Pandey et al. [201] developed an autonomous robot assistant, while Kołcon et al. [202] showcased the use of micro-ROS in smart warehouses, highlighting the system's effectiveness in navigating complex environments and managing tasks efficiently.

Furthermore, ROS 2's application in enhancing human-robot interaction is exemplified by Sobrín-Hidalgo et al. [203], who utilized large language models to generate explanations for autonomous robot actions, improving transparency and trust in robotic systems. The educational sector has also benefited, with Lages [204] demonstrating the use of ROS 2 as a teaching platform for robotics control and dynamics, providing students with hands-on experience in sophisticated tools.

In military UAV deployments, works such as those by Lu et al. [195], Park et al. [196], and Chen et al. [197] have utilized ROS 2 in both simulation and experimental settings, showcasing its versatility and effectiveness. Similarly, in logistics and manufacturing, ROS 2's ability to integrate various sensors and algorithms has been highlighted in studies such as those by Pandey et al. [201] and Kołcon et al. [202].

7. Literature Database and Analysis

As part of our comprehensive survey on ROS 2, we have created an extensive online database to catalog and analyze the wealth of literature available on ROS 1 and ROS 2. This database serves as a valuable resource for researchers and practitioners in the field of robotics, offering a centralized repository of information on various research domains, targeted robotic platforms, industries of focus, and types of articles.

The database comprises a total of 7498 articles published since 2009, meticulously categorized to facilitate easy access and navigation. Each entry in the database includes detailed metadata such as the version of ROS addressed or utilized, the specific field of application, the robotic platform used, the article DOI, and links to any associated GitHub repositories. This level of detail ensures that users can quickly locate relevant research and resources tailored to their specific needs.

 doi:10.20944/preprints202410.1204.v1

To enhance usability, the database features advanced filtering and sorting capabilities. Users can filter articles based on specific criteria such as research domain, robotic platform, industry of focus, and article type. For example, a user interested in finding all GitHub repositories related to ROS 2 for UAV swarms can easily do so with just a few clicks. This functionality significantly reduces the time and effort required to find pertinent information. For example, Figure 6 shows a sunburst chart of ROS 2 publications taxonomy.

Our literature database is not only a repository but also an analytical tool. By aggregating and categorizing the research, we have been able to identify key trends, gaps, and emerging areas in ROS 2 research. This analysis is invaluable for guiding future research directions and fostering collaborations within the robotics community. As an example, Figure 10 shows the trend lines of ROS 1 and ROS 2 number of publications. We can notice the gap between the two and also the rise of ROS 2 publication, adding the fact that ROS 1 support will end after 2025, we suspect a decline in the trend line.

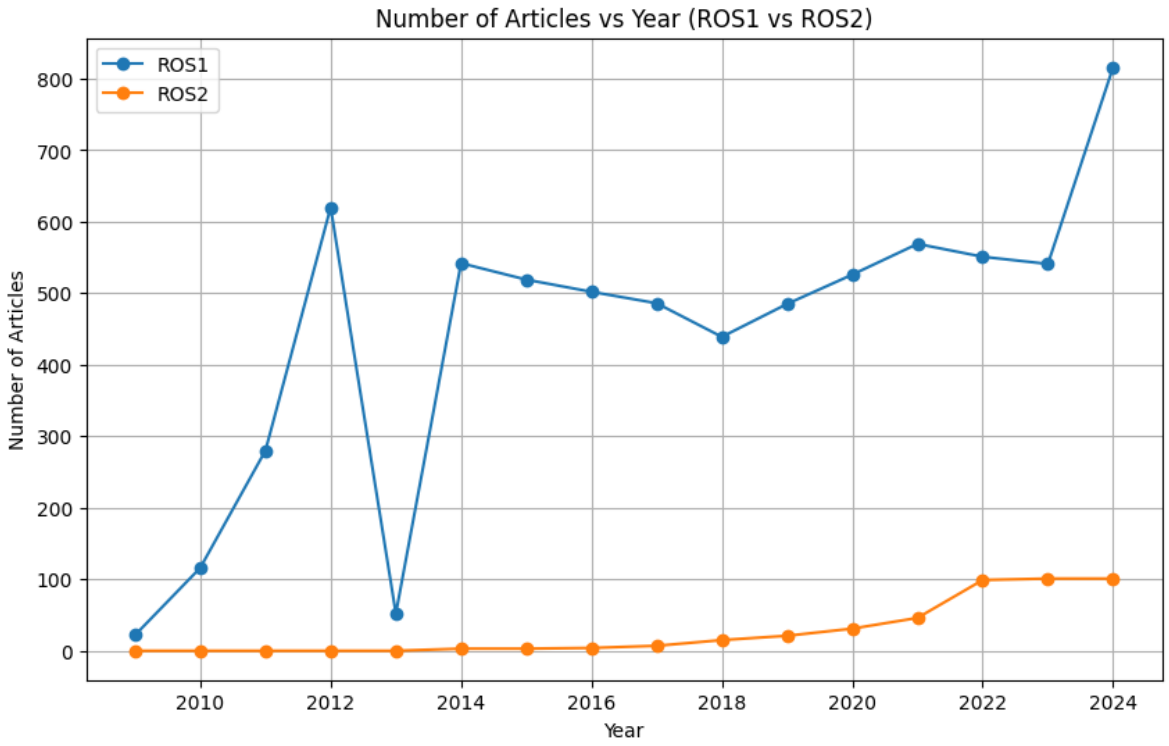

Figure 10. Number of Articles vs Year (ROS1 vs ROS2).

In Figure 11 we can notice that ROS 2 has already exceeded ROS 1 publications, that could indicate ROS 2 support for hardware acceleration applications and research.

 doi:10.20944/preprints202410.1204.v1

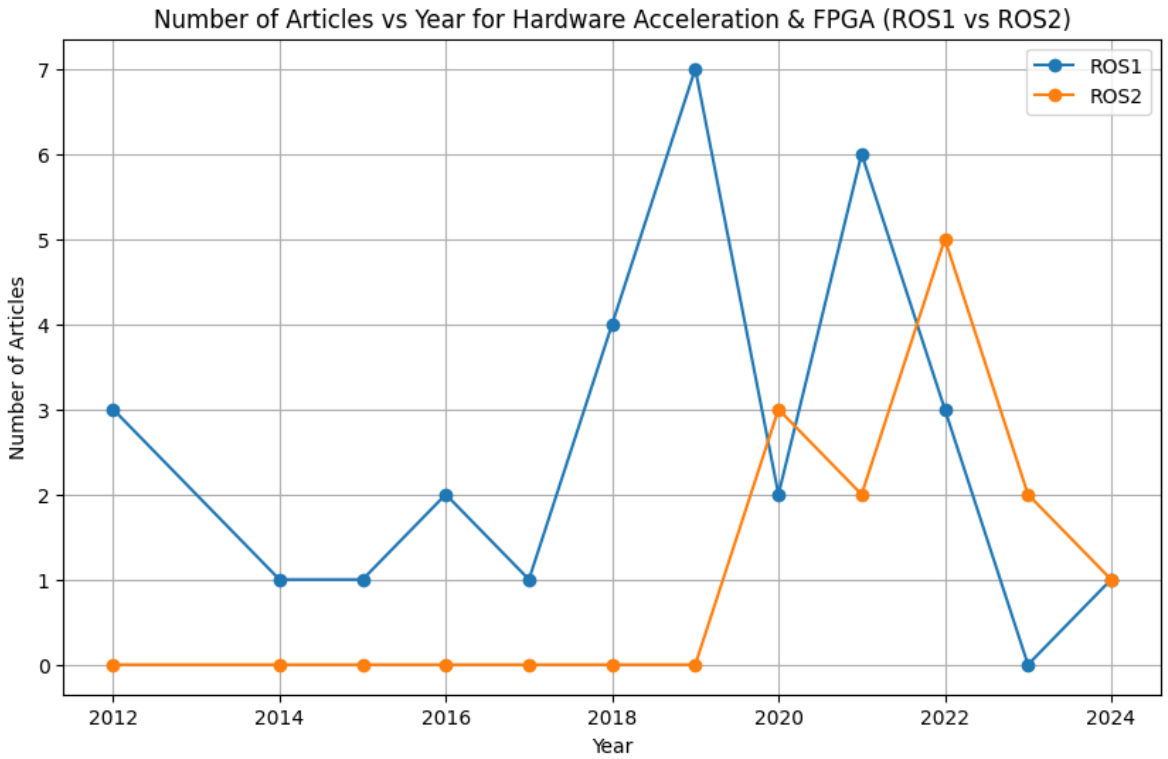

Figure 11. Number of Articles vs Year for FPGA (ROS1 vs ROS2).

In addition, Figure 12 shows that middleware research on ROS 2 is also comparative to the mature ROS 1. Which suggest huge research contributions to the DDS gaps existing today and improving upon it.

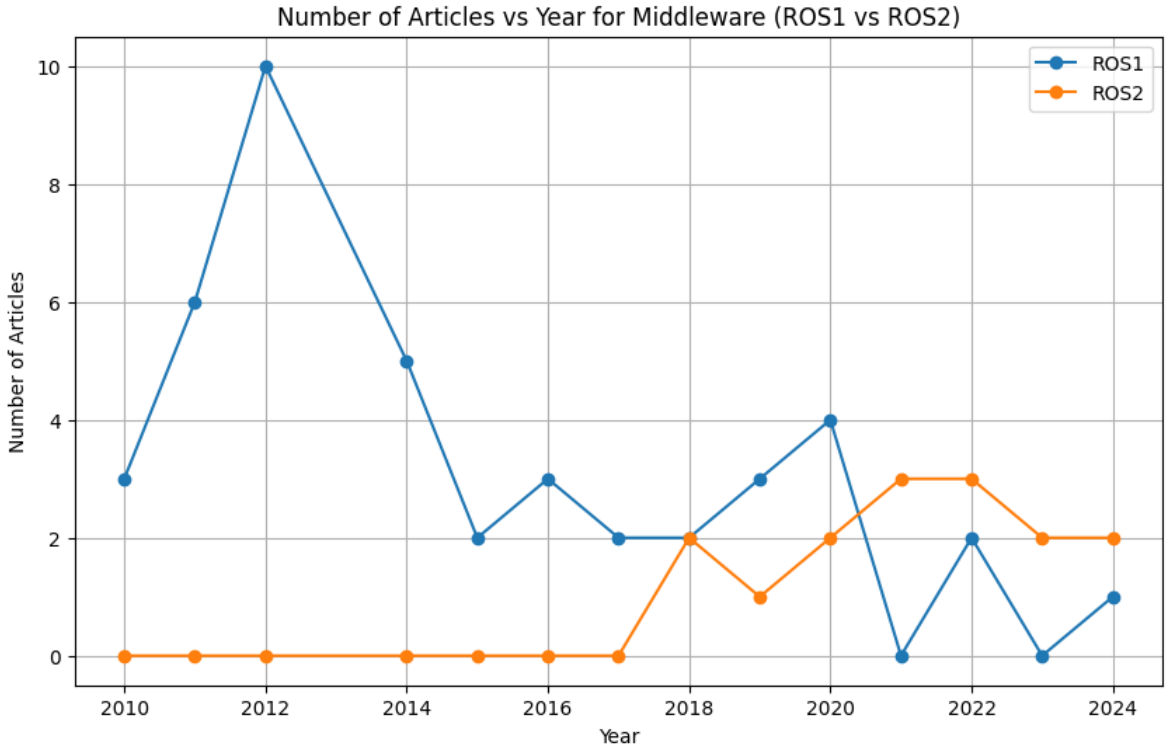

Figure 12. Number of Articles vs Year for Middleware (ROS1 vs ROS2).

 doi:10.20944/preprints202410.1204.v1

As a final example, Figure 13 shows clear sign of the security community favoritism towards ROS 2 research. Indicating ROS 2 bright future in the field of cyber security.

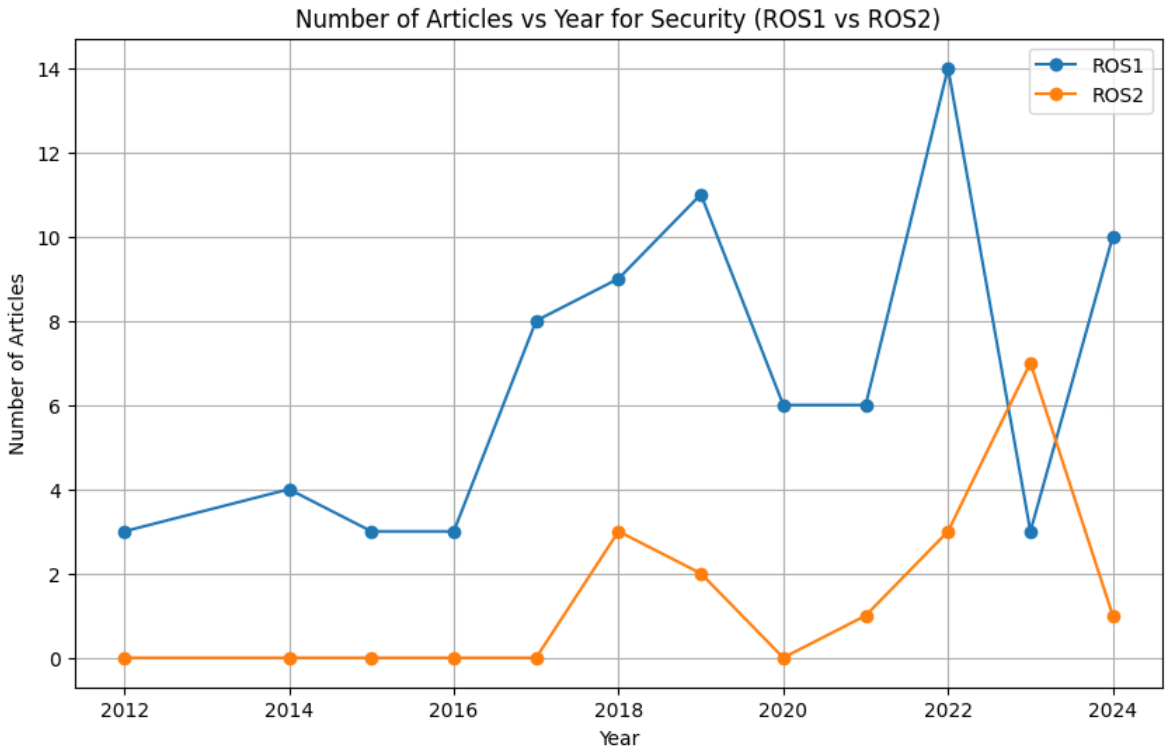

Figure 13. Number of Articles vs Year for Security (ROS1 vs ROS2).

8. Conclusions

This survey provides a comprehensive examination of the current state and potential of ROS 2, emphasizing its evolution from ROS 1 and highlighting the significant advancements that make ROS 2 a powerful framework for modern robotic applications. Our analysis demonstrates that ROS 2 introduces key architectural and functional improvements, including better modularity, scalability, and real-time performance—features that are crucial for developing reliable and complex robotic systems.

Through a comparative analysis of ROS 1 and ROS 2, we have shown that ROS 2 effectively addresses the limitations of its predecessor. Specifically, ROS 2 overcomes challenges such as dependency on a central master node, the lack of real-time guarantees, and limited support for multi-robot systems. The transition to a more decentralized, scalable, and real-time framework positions ROS 2 as a leading solution for a wide range of applications, including industrial automation, healthcare, and autonomous vehicles.

8.1. Key Research Findings

Our research identifies several key areas of active investigation within the ROS 2 ecosystem, reflecting the ongoing efforts to enhance its performance and reliability. These include:

- **Security Improvements**: ROS 2 incorporates built-in security mechanisms such as encryption, authentication, and access control. While these mechanisms significantly enhance security for critical robotic applications, balancing performance and security remains a challenge, particularly in resource-constrained environments.
- **Real-Time and Multi-Robot Systems (MRS)**: With its real-time capabilities, supported by the Data Distribution Service (DDS) middleware, ROS 2 greatly improves communication reliability and timing for multi-robot systems. While custom schedulers and priority-driven executors have

 doi:10.20944/preprints202410.1204.v1

been proposed to further optimize real-time performance, challenges such as latency in complex environments persist.

- **Middleware Advancements**: Dynamic DDS implementations and middleware options such as Zenoh offer optimized communication across various network conditions, especially in edge-cloud and distributed systems. These enhancements strengthen ROS 2's scalability and real-time coordination, particularly in multi-robot scenarios.
- **Modularity and Hardware Acceleration**: ROS 2's modularity and compatibility with hardware acceleration platforms, such as FPGAs, enable efficient deployment in resource-intensive applications like autonomous vehicles and robotic arms. This feature offers a pathway to better performance in AI-driven and industrial robotics.
- **Quality of Service (QoS)**: ROS 2's flexible QoS settings allow for fine-tuned control over communication reliability and latency. These settings are critical in real-time applications such as autonomous driving and healthcare robotics. However, finding the right balance between QoS parameters and security requirements remains an ongoing area of research.
- **Containerization and Cloud Integration**: ROS 2's adaptability to containerized environments and its integration with orchestration tools such as Kubernetes provide robust support for scalable deployment in distributed robotic systems, particularly in edge-cloud architectures.

The key challenges and advances in each of these areas are summarized in Table 2.

8.2. Answers to Research Questions

- **What are the key challenges in transitioning from ROS 1 to ROS 2?**
  ROS 1's limitations, such as reliance on a central master node, lack of real-time guarantees, and challenges in supporting multi-robot systems, have been effectively addressed in ROS 2 through decentralized architecture, real-time DDS middleware, and enhanced scalability.
- **How does ROS 2 improve scalability and reliability in multi-robot systems compared to ROS 1?**
  By removing the single point of failure (the master node) and introducing dynamic middleware switching and advanced synchronization protocols, ROS 2 significantly improves scalability and reliability in multi-robot systems.
- **What are the advancements in real-time performance with ROS 2, and what limitations remain?**
  ROS 2's real-time performance has been improved through the introduction of priority-based schedulers and dynamic QoS management. However, limitations such as callback unpredictability, network delays, and latency in distributed systems still require further refinement.
- **How effective are ROS 2's built-in security features in mitigating vulnerabilities?**
  ROS 2 introduces native security features like SROS2, enhancing security in robotic applications. However, performance overhead from encryption and authentication, especially in resource-limited systems, remains a challenge.
- **What are the performance differences between different DDS implementations in ROS 2?**
  Performance varies among DDS vendors, such as FastDDS and CycloneDDS, particularly when security features are enabled. Middleware choices should therefore be made based on specific network and performance needs.
- **How does ROS 2 support containerization and orchestration in distributed robotic applications?**
  ROS 2 supports distributed systems through containerization tools like FogROS2, which enable cloud and edge integration, facilitating scalable robotic applications across diverse network environments.
- **How has ROS 2 enabled advancements in robot navigation, motion planning, and control frameworks?**

 doi:10.20944/preprints202410.1204.v1

ROS 2's modularity and real-time capabilities have been leveraged in frameworks such as Nav2, MoveIt, and Autoware to enhance path planning, motion control, and autonomous navigation, establishing ROS 2 as an essential platform for advanced robotics solutions.

- **What are the benefits and challenges of using ROS 2 in embedded systems and resource-constrained environments?**
  Micro-ROS extends ROS 2 to resource-constrained devices, supporting applications in embedded systems like IoT and CubeSats. However, challenges remain in optimizing real-time performance in these resource-limited settings.
- **How do ROS 2's QoS settings balance between performance, security, and communication reliability?**
  ROS 2's QoS settings enable fine-tuned control over data transmission, improving both performance and reliability. However, optimizing the balance between security features and communication efficiency remains a challenge.
- **Is ROS 2 mature right now?**
  ROS 2 has reached a high level of maturity, especially with the completion of core functionalities such as navigation, transformations, and multi-robot support. Its widespread adoption in industries like healthcare, autonomous driving, and industrial robotics underscores its readiness for complex real-world applications. However, challenges such as maintaining real-time performance under high network loads and refining security without compromising performance indicate that ongoing improvements are still necessary to meet the demands of large-scale, mission-critical deployments. Tools like FogROS2 and Micro-ROS further reflect the ecosystem's maturity in handling distributed and embedded systems.

8.3. Summary and Future Directions

This survey comprehensively explores the current state of ROS 2, illustrating its evolution from ROS 1 and emphasizing significant advancements that reinforce its status as a leading framework for modern robotics. Key architectural improvements—such as enhanced modularity, scalability, and real-time performance—have been instrumental in establishing ROS 2 as a robust and reliable platform for developing sophisticated robotic systems.

Several key research areas within ROS 2, including security, multi-robot systems, hardware acceleration, and real-time capabilities, are actively being investigated. These efforts reflect the community's drive to continually improve ROS 2's performance and reliability, with diverse applications ranging from industrial automation to healthcare and autonomous vehicles.

Furthermore, this survey provides a full taxonomy of open-source ROS 2 packages (Table 9), summarizing critical frameworks for multi-robot systems (MRS), human-robot interaction (HRI), reinforcement learning (RL), security, and other domains. This taxonomy facilitates easy access to essential tools for researchers across various fields.

The applications section highlights ROS 2's versatility, showcasing its broad use across different robotic platforms and fields of application, as summarized in Figure 8. Additionally, the ecosystem's richness is evident in the discussion of meta frameworks such as Nav2, MoveIt, and Autoware, along with simulation and visualization tools like Gazebo and Webots, all of which support comprehensive development and testing environments.

Finally, our literature database, accessible at https://www.ros.riotu-lab.org/, offers an invaluable resource for the robotics community. Supported by advanced filtering and sorting capabilities, the database helps researchers navigate the vast array of research articles and resources. Figures 10, 11, 13, and 12 provide further analysis of ROS 2's growth and adoption across multiple sectors.

In conclusion, ROS 2 represents a significant leap forward in the evolution of robotic operating systems. It delivers a scalable, flexible, and robust framework that meets the demands of modern robotics applications. Ongoing community collaboration and continuous development within the ROS 2 ecosystem are vital to overcoming existing challenges and unlocking new possibilities for

robotics. We hope this survey serves as a valuable reference for researchers and practitioners, fostering innovation and progress in the field.

Funding: This research received no external funding.

Data Availability Statement: The original data presented in the study are openly available in https://www.ros.riotu-lab.org/.

Acknowledgments: The authors would like to thank Prince Sultan University for their support.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A. ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software, 2009.
2. Shim, D.; Kim, S.H.; Cho, K. Open-source robotic platforms in ROS for advanced robotics research. *International Journal of Control, Automation and Systems* **2016**, *14*, 1205–1215.
3. Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; others. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* **2016**.
4. Xie, L.; Hu, J.; Li, J.; Zhang, H.; Wang, Y. A novel mechanical design of a quadruped robot with force-position hybrid control. *International Journal of Advanced Robotic Systems* **2018**, *15*, 1729881418810173.
5. Badger, J.; Gooding, D.; Ensley, K.; Hambuchen, K.; Thackston, A., ROS in Space: A Case Study on Robonaut 2. In *Robot Operating System (ROS): The Complete Reference (Volume 1)*; Koubaa, A., Ed.; Springer International Publishing: Cham, 2016; pp. 343–373. doi:10.1007/978-3-319-26054-9_13.
6. Gao, H.; Wu, Z.; Liu, J. Cartographer: a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations. *IEEE Robotics and Automation Letters* **2018**, *3*, 750–757.
7. Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* **2000**.
8. Rusu, R.B.; Cousins, S. 3D is here: Point Cloud Library (PCL). IEEE International Conference on Robotics and Automation (ICRA); , 2011.
9. Koubaa, A. ROSGPT: Next-Generation Human-Robot Interaction with ChatGPT and ROS, 2023. doi:10.20944/preprints202304.0827.v1.
10. Pichierri, L.; Testa, A.; Notarstefano, G. CrazyChoir: Flying Swarms of Crazyflie Quadrotors in ROS 2. *ArXiv* **2023**, *abs/2302.00716*.
11. Arcos, L.; Vicente, K.; Cruz, P.; Abad, J.; Zambrano, I. A ROS2 based Trajectory Tracking Controller of a 3UPS-1RPU Parallel Robot for Knee Rehabilitation. 2022 IEEE Sixth Ecuador Technical Chapters Meeting (ETCM), 2022, pp. 1–6. doi:10.1109/ETCM56276.2022.9935755.
12. Audonnet, F.P.; Hamilton, A.; Aragon-Camarasa, G. A Systematic Comparison of Simulation Software for Robotic Arm Manipulation using ROS2, 2022. doi:10.48550/ARXIV.2204.06433.
13. Zhang, J.; Keramat, F.; Yu, X.; Hernández, D.M.; Queralta, J.P.; Westerlund, T. Distributed Robotic Systems in the Edge-Cloud Continuum with ROS 2: a Review on Novel Architectures and Technology Readiness. 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC), 2022, pp. 1–8. doi:10.1109/FMEC57183.2022.10062523.
14. Macenski, S.; Moore, T.; Lu, D.V.; Merzlyakov, A.; Ferguson, M. From the desks of ROS maintainers: A survey of modern and capable mobile robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems* **2023**, *168*, 104493. doi:https://doi.org/10.1016/j.robot.2023.104493.
15. Choi, H.S.; Enright, D.; Sobhani, H.; Xiang, Y.; Kim, H. Priority-Driven Real-Time Scheduling in ROS 2: Potential and Challenges. 1st International Workshop on Real-time And intelliGent Edge computing (RAGE), 2022.
16. Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; Woodall, W. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics* **2022**, *7*, eabm6074, [https://www.science.org/doi/pdf/10.1126/scirobotics.abm6074]. doi:10.1126/scirobotics.abm6074.
17. DiLuoffo, V.; Michalson, W.R.; Sunar, B. Robot Operating System 2: The need for a holistic security approach to robotic architectures. *International Journal of Advanced Robotic Systems* **2018**, *15*, 1729881418770011, [https://doi.org/10.1177/1729881418770011]. doi:10.1177/1729881418770011.

18. Alhanahnah, M. Software Quality Assessment for Robot Operating System, 2020, [arXiv:cs.SE/2012.07196].
19. Bonci, A.; Gaudeni, F.; Giannini, M.C.; Longhi, S. Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey. *Applied Sciences* **2023**, *13*. doi:10.3390/app132312796.
20. Koubâa, A.; Sriti, M.; Bennaceur, H.; Ammar, A.; Javed, Y.; Alajlan, M.; Al-Elaiwi, N.; Tounsi, M.; Shakshuki, E.M. COROS: A Multi-Agent Software Architecture for Cooperative and Autonomous Service Robots. In *Cooperative Robots and Sensor Networks 2015*; Koubâa, A.; de Dios, J.R.M., Eds.; Springer, 2015; Vol. 604, *Studies in Computational Intelligence*, pp. 3–30. doi:10.1007/978-3-319-18299-5_1.
21. Foote, T. tf: The transform library. Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on, 2013, Open-Source Software workshop, pp. 1–6. doi:10.1109/TePRA.2013.6556373.
22. Macenski, S.; Martín, F.; White, R.; Clavero, J.G. The Marathon 2: A Navigation System. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 2718–2725. doi:10.1109/IROS45743.2020.9341207.
23. Vilches, V.M.; White, R.; Caiazza, G.; Arguedas, M. SROS2: Usable Cyber Security Tools for ROS 2, 2022. doi:10.48550/ARXIV.2208.02615.
24. Henle, J.; Stoffel, M.; Schindewolf, M.; Nägele, A.T.; Sax, E. Architecture platforms for future vehicles: a comparison of ROS2 and Adaptive AUTOSAR. 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), 2022, pp. 3095–3102. doi:10.1109/ITSC55140.2022.9921894.
25. Macenski, S.; Soragna, A.; Carroll, M.; Ge, Z. Impact of ROS 2 Node Composition in Robotic Systems, 2023, [arXiv:cs.RO/2305.09933].
26. Salimi, S.; Keramat, F.; Westerlund, T.; Queralta, J.P. A Customizable Conflict Resolution and Attribute-Based Access Control Framework for Multi-Robot Systems, 2023, [arXiv:cs.RO/2308.16482].
27. Fu, L.; Salimi, S.; Queralta, J.P.; Westerlund, T. Event-driven Fabric Blockchain - ROS 2 Interface: Towards Secure and Auditable Teleoperation of Mobile Robots, 2023, [arXiv:cs.RO/2304.00781].
28. Patel, Y.; Rughani, P.H.; Desai, D. Analyzing Security Vulnerability and Forensic Investigation of ROS2: A Case Study. Proceedings of the 8th International Conference on Robotics and Artificial Intelligence; Association for Computing Machinery: New York, NY, USA, 2023; ICRAI '22, p. 6–12. doi:10.1145/3573910.3573912.
29. DiLuoffo, V.; Michalson, W.R.; Sunar, B. Credential Masquerading and OpenSSL Spy: Exploring ROS 2 using DDS security. *ArXiv* **2019**, *abs/1904.09179*.
30. Takemoto, S.; Nishida, K.; Nozaki, Y.; Yoshikawa, M.; Honda, S.; Kurachi, R. Performance Evaluation of CAESAR Authenticated Encryption on SROS2. Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference; Association for Computing Machinery: New York, NY, USA, 2020; AICCC 2019, p. 168–172. doi:10.1145/3375959.3375976.
31. Rotta, R.; Mykytyn, P. Secure Multi-hop Telemetry Broadcasts for UAV Swarm Communication, 2024, [arXiv:cs.CR/2401.11915].
32. Yang, S.; Guo, J.; Rui, X. Formal Analysis and Detection for ROS2 Communication Security Vulnerability. *Electronics* **2024**, *13*. doi:10.3390/electronics13091762.
33. Patel, Y.; Rughani, P.H.; Desai, D. Network Forensic Investigation of Collaborative Robots: A Case Study. 2022 7th International Conference on Mechanical Engineering and Robotics Research (ICMERR), 2022, pp. 51–54. doi:10.1109/ICMERR56497.2022.10097787.
34. Yang, F.; Zhan, S.S.; Wang, Y.; Huang, C.; Zhu, Q. Case Study: Runtime Safety Verification of Neural Network Controlled System, 2024, [arXiv:cs.RO/2408.08592].
35. Kim, J.; Smereka, J.M.; Cheung, C.; Nepal, S.; Grobler, M. Security and Performance Considerations in ROS 2: A Balancing Act. *ArXiv* **2018**, *abs/1809.09566*.
36. Sandoval, S.; Thulasiraman, P. Cyber Security Assessment of the Robot Operating System 2 for Aerial Networks. 2019 IEEE International Systems Conference (SysCon), 2019, pp. 1–8. doi:10.1109/SYSCON.2019.8836824.
37. Gupta, R.; Kurtz, Z.T.; Scherer, S.; Smereka, J.M. Open Problems in Robotic Anomaly Detection, 2018, [arXiv:cs.RO/1809.03565].
38. Goerke, N.; Timmermann, D.; Baumgart, I. Who Controls Your Robot? An Evaluation of ROS Security Mechanisms. 2021 7th International Conference on Automation, Robotics and Applications (ICARA), 2021, pp. 60–66. doi:10.1109/ICARA51699.2021.9376468.

39. Tanadechopon, T.; Kasemsontitum, B. Proposed Technique for Data Security with the AES Algorithm in Robot Operating System (ROS). 2023 27th International Computer Science and Engineering Conference (ICSEC), 2023, pp. 153–156. doi:10.1109/ICSEC59635.2023.10329645.
40. Casini, D.; Blaß, T.; Lütkebohle, I.; Brandenburg, B.B. Response-Time Analysis of ROS 2 Processing Chains Under Reservation-Based Scheduling. 31st Euromicro Conference on Real-Time Systems (ECRTS 2019); Quinton, S., Ed.; Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik: Dagstuhl, Germany, 2019; Vol. 133, *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 6:1–6:23. doi:10.4230/LIPIcs.ECRTS.2019.6.
41. Park, S.; Choi, J.; Hwang, S.; Lee, C.G. ROS2 Extension of Functionally and Temporally Correct Real-Time Simulation of Cyber Systems for Automotive Systems. 2021 International Symposium on Electrical, Electronics and Information Engineering; Association for Computing Machinery: New York, NY, USA, 2021; ISEEIE 2021, p. 185–189. doi:10.1145/3459104.3459137.
42. Teper, H.; Betz, T.; Von Der Brüggen, G.; Chen, K.H.; Betz, J.; Chen, J.J. Timing-Aware ROS 2 Architecture and System Optimization. 2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2023, pp. 206–215. doi:10.1109/RTCSA58653.2023.00032.
43. Park, J.; Delgado, R.; Choi, B.W. Real-Time Characteristics of ROS 2.0 in Multiagent Robot Systems: An Empirical Study. *IEEE Access* **2020**, *8*, 154637–154651. doi:10.1109/ACCESS.2020.3018122.
44. Tang, Y.; Feng, Z.; Guan, N.; Jiang, X.; Lv, M.; Deng, Q.; Yi, W. Response Time Analysis and Priority Assignment of Processing Chains on ROS2 Executors. 2020 IEEE Real-Time Systems Symposium (RTSS), 2020, pp. 231–243. doi:10.1109/RTSS49844.2020.00030.
45. Choi, H.; Xiang, Y.; Kim, H. PiCAS: New Design of Priority-Driven Chain-Aware Scheduling for ROS2. 2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS), 2021, pp. 251–263. doi:10.1109/RTAS52030.2021.00028.
46. Sobhani, H.; Choi, H.; Kim, H. Timing Analysis and Priority-driven Enhancements of ROS 2 Multi-threaded Executors. 2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS), 2023, pp. 106–118. doi:10.1109/RTAS58335.2023.00016.
47. Tang, Y.; Guan, N.; Jiang, X.; Luo, X.; Yi, W. Real-Time Performance Analysis of Processing Systems on ROS 2 Executors. 2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS), 2023, pp. 80–92. doi:10.1109/RTAS58335.2023.00014.
48. Patel, D.; Maiti, C.; Muthuswamy, S. Real-Time Performance Monitoring of a CNC Milling Machine using ROS 2 and AWS IoT Towards Industry 4.0. IEEE EUROCON 2023 - 20th International Conference on Smart Technologies, 2023, pp. 776–781. doi:10.1109/EUROCON56442.2023.10199020.
49. Betz, T.; Wen, L.; Pan, F.; Kaljavesi, G.; Zuepke, A.; Bastoni, A.; Caccamo, M.; Knoll, A.; Betz, J. A Containerized Microservice Architecture for a ROS 2 Autonomous Driving Software: An End-to-End Latency Evaluation, 2024, [arXiv:cs.RO/2404.12683].
50. Blaß, T.; Casini, D.; Bozhko, S.; Brandenburg, B.B. A ROS 2 Response-Time Analysis Exploiting Starvation Freedom and Execution-Time Variance. 2021 IEEE Real-Time Systems Symposium (RTSS), 2021, pp. 41–53. doi:10.1109/RTSS52674.2021.00016.
51. Li, R.; Hu, T.; Jiang, X.; Li, L.; Xing, W.; Deng, Q.; Guan, N. ROSGM: A Real-Time GPU Management Framework with Plug-In Policies for ROS 2. 2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS), 2023, pp. 93–105. doi:10.1109/RTAS58335.2023.00015.
52. Kronauer, T.; Pohlmann, J.; Matthé, M.; Smejkal, T.; Fettweis, G. Latency Analysis of ROS2 Multi-Node Systems. 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2021, pp. 1–7. doi:10.1109/MFI52462.2021.9591166.
53. Randolph, C. Improving the Predictability of Event Chains in ROS 2. Master's thesis, Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2021.
54. Parmar, N.; Ranga, V.; Simhachalam Naidu, B. editor=Ranganathan, G.; Chen, J.; Rocha, Á. Syntactic Interoperability in Real-Time Systems, ROS 2, and Adaptive AUTOSAR Using Data Distribution Services: An Approach. Inventive Communication and Computational Technologies; Springer Singapore: Singapore, 2020; pp. 257–274.
55. Katsuya, M.; Ren, M.; Sho'ji, S. A Study of Optimizing Inter-node Communication with an IPC-based DDS Implementation in ROS 2. Asia Pacific Conference on Robot IoT System Development and Platform 2019 (APRIS2019), 2020.

56. Morita, R.; Matsubara, K. Dynamic Binding a Proper DDS Implementation for Optimizing Inter-Node Communication in ROS2. 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2018, pp. 246–247. doi:10.1109/RTCSA.2018.00043.
57. Aartsen, M.; Banga, K.; Talko, K.; Touw, D.; Wisman, B.; Meïnsma, D.; Björkqvist, M. Analyzing Interoperability and Security Overhead of ROS2 DDS Middleware. 2022 30th Mediterranean Conference on Control and Automation (MED), 2022, pp. 976–981. doi:10.1109/MED54222.2022.9837282.
58. Zhang, J.; Yu, X.; Ha, S.; Queralta, J.P.; Westerlund, T. Comparison of DDS, MQTT, and Zenoh in Edge-to-Edge and Edge-to-Cloud Communication for Distributed ROS 2 Systems, 2023, [arXiv:cs.RO/2309.07496].
59. Pöhnl, M.; Tamisier, A.; Blass, T. A Middleware Journey from Microcontrollers to Microprocessors. 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022, pp. 282–286. doi:10.23919/DATE54114.2022.9774609.
60. Mamani-Saico, A.; Yanyachi, P.R. Implementation and Performance Study of the Micro-ROS/ROS2 Framework to Algorithm Design for Attitude Determination and Control System. *IEEE Access* **2023**, *11*, 128451–128460. doi:10.1109/ACCESS.2023.3330441.
61. Lienen, C.; Middeke, S.H.; Platzner, M. fpgaDDS: An Intra-FPGA Data Distribution Service for ROS 2 Robotics Applications, 2023, [arXiv:cs.RO/2303.00532].
62. Lienen, C.; Platzner, M. ReconROS Executor: Event-Driven Programming of FPGA-accelerated ROS 2 Applications. *ArXiv* **2022**, *abs/2201.07454*.
63. Zhang, J.; Keramat, F.; Yu, X.; Hernández, D.M.; Queralta, J.P.; Westerlund, T. Distributed Robotic Systems in the Edge-Cloud Continuum with ROS 2: a Review on Novel Architectures and Technology Readiness. 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC), 2022, pp. 1–8. doi:10.1109/FMEC57183.2022.10062523.
64. De Marchi, M.; Bombieri, N. Orchestration-Aware Optimization of ROS2 Communication Protocols. 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2024, pp. 1–6. doi:10.23919/DATE58400.2024.10546777.
65. Yu, Y.; Lee, S. Measurements of the Benefits of Edge Computing on Autonomous Driving. 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), 2022, pp. 2155–2159. doi:10.1109/ICTC55196.2022.9952693.
66. Thulasiraman, P.; Chen, Z.; Allen, B.; Bingham, B. Evaluation of the Robot Operating System 2 in Lossy Unmanned Networks. 2020 IEEE International Systems Conference (SysCon), 2020, pp. 1–8. doi:10.1109/SysCon47679.2020.9275849.
67. Dauphin, L.; Baccelli, E.; Adjih, C. RIOT-ROS2: Low-Cost Robots in IoT Controlled via Information-Centric Networking. 2018 IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), 2018, pp. 1–6. doi:10.23919/PEMWN.2018.8548798.
68. Jalil, A.; Kobayashi, J. Efficacy of Local Cache for Performance Improvement of Reliable Data Transmission in Aggregated Robot Processing Architecture. 2022 22nd International Conference on Control, Automation and Systems (ICCAS), 2022, pp. 1339–1344. doi:10.23919/ICCAS55662.2022.10003765.
69. Jalil, A.; Kobayashi, J.; Saitoh, T. Performance Improvement of Multi-Robot Data Transmission in Aggregated Robot Processing Architecture with Caches and QoS Balancing Optimization. *Robotics* **2023**, *12*. doi:10.3390/robotics12030087.
70. Jalil, A.; Kobayashi, J.; Saitoh, T. QoS Balancing Optimization in Aggregated Robot Processing Architecture: Rate and Buffer. *Journal of Advances in Artificial Life Robotics* **2023**, *3*, 209–213. doi:10.57417/jaalr.3.4_209.
71. Fernandez, J.; Allen, B.; Thulasiraman, P.; Bingham, B. Performance Study of the Robot Operating System 2 with QoS and Cyber Security Settings. 2020 IEEE International Systems Conference (SysCon), 2020, pp. 1–6. doi:10.1109/SysCon47679.2020.9275872.
72. Parra, S.; Schneider, S.; Hochgeschwender, N. Specifying QoS Requirements and Capabilities for Component-Based Robot Software. 2021 IEEE/ACM 3rd International Workshop on Robotics Software Engineering (RoSE), 2021, pp. 29–36. doi:10.1109/RoSE52553.2021.00012.
73. Jaiswal, B.; Tyagi, H.; Gopalan, A.; Sevani, V. WiROS: A QoS Software Solution for ros2 in a WiFi Network. 2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS), 2023, pp. 216–218. doi:10.1109/COMSNETS56262.2023.10041412.
74. Dey, E.; Walczak, M.; Anwar, M.S.; Roy, N. A Reliable and Low Latency Synchronizing Middleware for Co-simulation of a Heterogeneous Multi-Robot Systems, 2022. doi:10.48550/ARXIV.2211.05359.

75. Dust, L.J.; Persson, E.; Ekström, M.; Mubeen, S.; Dean, E. Quantitative analysis of communication handling for centralized multi-agent robot systems using ROS2. 2022 IEEE 20th International Conference on Industrial Informatics (INDIN), 2022, pp. 624–629. doi:10.1109/INDIN51773.2022.9976160.
76. Jalil, A.; Kobayashi, J. Experimental Analyses of an Efficient Aggregated Robot Processing with Cache-Control for Multi-Robot System. 2020 20th International Conference on Control, Automation and Systems (ICCAS), 2020, pp. 1105–1109. doi:10.23919/ICCAS50221.2020.9268225.
77. Chovet, L.P.; Garcia, G.M.; Bera, A.; Richard, A.; Yoshida, K.; Olivares-Mendez, M.A. Performance Comparison of ROS2 Middlewares for Multi-robot Mesh Networks in Planetary Exploration, 2024, [arXiv:cs.RO/2407.03091].
78. Fernandez-Cortizas, M.; Molina, M.; Arias-Perez, P.; Perez-Segui, R.; Perez-Saura, D.; Campoy, P. Aerostack2: A Software Framework for Developing Multi-robot Aerial Systems, 2023, [arXiv:cs.RO/2303.18237].
79. Zhang, Y.; Wurll, C.; Hein, B. KubeROS: A Unified Platform for Automated and Scalable Deployment of ROS2-based Multi-Robot Applications. 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 9097–9103. doi:10.1109/ICRA48891.2023.10160632.
80. Kaiser, T.K.; Begemann, M.J.; Plattenteich, T.; Schilling, L.; Schildbach, G.; Hamann, H. ROS2SWARM - A ROS 2 Package for Swarm Robot Behaviors. 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 6875–6881. doi:10.1109/ICRA46639.2022.9812417.
81. Blumenkamp, J.; Shankar, A.; Bettini, M.; Bird, J.; Prorok, A. The Cambridge RoboMaster: An Agile Multi-Robot Research Platform, 2024, [arXiv:cs.RO/2405.02198].
82. Pacheco, A.; Denis, U.; Zakir, R.; Strobel, V.; Reina, A.; Dorigo, M. Toychain: A Simple Blockchain for Research in Swarm Robotics, 2024, [arXiv:cs.RO/2407.06630].
83. Castillo-Sánchez, J.B.; González-Parada, E.; Cano-García, J.M. A novel testbed for evaluating ROS 2 robot swarm wireless communications. *2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON)* **2024**, pp. 68–73.
84. Testa, A.; Camisa, A.; Notarstefano, G. ChoiRbot: A ROS 2 Toolbox for Cooperative Robotics. *IEEE Robotics and Automation Letters* **2021**, *6*, 2714–2720. doi:10.1109/LRA.2021.3061366.
85. Alzetta, F.; Giorgini, P. Towards a Real-Time BDI Model for ROS 2. Workshop From Objects to Agents, 2019.
86. Passalis, N.; Pedrazzi, S.; Babuska, R.; Burgard, W.; Dias, D.; Ferro, F.; Gabbouj, M.; Green, O.; Iosifidis, A.; Kayacan, E.; Kober, J.; Michel, O.; Nikolaidis, N.; Nousi, P.; Pieters, R.; Tzelepi, M.; Valada, A.; Tefas, A. OpenDR: An Open Toolkit for Enabling High Performance, Low Footprint Deep Learning for Robotics. Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (to appear), 2022.
87. Angleraud, A.; Ekrekli, A.; Samarawickrama, K.; Sharma, G.; Pieters, R. Sensor-based human–robot collaboration for industrial tasks. *Robotics and Computer-Integrated Manufacturing* **2024**, *86*, 102663. doi:https://doi.org/10.1016/j.rcim.2023.102663.
88. Bono, A.; Brameld, K.; D'Alfonso, L.; Fedele, G. Open Access NAO (OAN): a ROS2-based software framework for HRI applications with the NAO robot, 2024, [arXiv:cs.RO/2403.13960].
89. Fabian, S.; Stryk, O.v. Open-Source Tools for Efficient ROS and ROS2-based 2D Human-Robot Interface Development. 2021 European Conference on Mobile Robots (ECMR), 2021, pp. 1–6. doi:10.1109/ECMR50962.2021.9568801.
90. Abbate, G.; Giusti, A.; Paolillo, A.; Gromov, B.; Gambardella, L.; Rizzoli, A.E.; Guzzi, J. PointIt: A ROS Toolkit for Interacting with Co-Located Robots Using Pointing Gestures. Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction. IEEE Press, 2022, HRI '22, p. 608–612.
91. Jo, W.; Wilson, R.; Kim, J.; McGuire, S.; Min, B. Toward a Wearable Biosensor Ecosystem on ROS 2 for Real-time Human-Robot Interaction Systems. *CoRR* **2021**, *abs/2110.03840*, [2110.03840].
92. Blanco-Claraco, J.L.; Tymchenko, B.; Mañas-Alvarez, F.J.; Cañadas-Aránega, F.; Ángel López-Gázquez.; Moreno, J.C. MultiVehicle Simulator (MVSim): lightweight dynamics simulator for multiagents and mobile robotics research, 2023, [arXiv:cs.RO/2302.11033].
93. PÃl'rez-Higueras, N.; Otero, R.; Caballero, F.; Merino, L. HuNavSim: A ROS 2 Human Navigation Simulator for Benchmarking 1416 Human-Aware Robot Navigation, 2023. [arXiv:cs.RO/2305.01303].
94. Rong, G.; Shin, B.H.; Tabatabaee, H.; Lu, Q.; Lemke, S.; Možeiko, M.; Boise, E.; Uhm, G.; Gerow, M.; Mehta, S.; Agafonov, E.; Kim, T.H.; Sterner, E.; Ushiroda, K.; Reyes, M.; Zelenkovsky, D.; Kim, S. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving, 2020, [arXiv:cs.RO/2005.03778].

95. Pieczyński, D.; Ptak, B.; Kraft, M.; Drapikowski, P. LunarSim: Lunar Rover Simulator Focused on High Visual Fidelity and ROS 2 Integration for Advanced Computer Vision Algorithm Development. *Applied Sciences* **2023**, *13*. doi:10.3390/app132212401.
96. Andreasen, M.; Holler, P.; Jensen, M.; Albano, M. MAES: a ROS 2-compatible simulation tool for exploration and coverage algorithms. *Artificial Life and Robotics* **2023**.
97. Manhães, M.M.M.; Scherer, S.A.; Voss, M.; Douat, L.R.; Rauschenbach, T. UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation. OCEANS 2016 MTS/IEEE Monterey. IEEE, 2016. doi:10.1109/oceans.2016.7761080.
98. Guo, Z.; Perminov, S.; Konenkov, M.; Tsetserukou, D. HawkDrive: A Transformer-driven Visual Perception System for Autonomous Driving in Night Scene, 2024, [arXiv:cs.CV/2404.04653].
99. Benjdira, B.; Koubaa, A.; Ali, A.M. ROSGPT_Vision: Commanding Robots Using Only Language Models' Prompts, 2023, [arXiv:cs.RO/2308.11236].
100. Koide, K.; Yokozuka, M.; Oishi, S.; Banno, A. GLIM: 3D range-inertial localization and mapping with GPU-accelerated scan matching factors. *Robotics and Autonomous Systems* **2024**, *179*, 104750. doi:https://doi.org/10.1016/j.robot.2024.104750.
101. Rolland, E.G.A.; Grøntved, K.A.R.; Christensen, A.L.; Watson, M.; Richardson, T. Autonomous UAV Volcanic Plume Sampling Based on Machine Vision and Path Planning. 2024 International Conference on Unmanned Aircraft Systems (ICUAS), 2024, pp. 1064–1071. doi:10.1109/ICUAS60882.2024.10556912.
102. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO Series in 2021. *arXiv preprint arXiv:2107.08430* **2021**.
103. Koide, K.; Oishi, S.; Yokozuka, M.; Banno, A. General, Single-shot, Target-less, and Automatic LiDAR-Camera Extrinsic Calibration Toolbox. 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 11301–11307. doi:10.1109/ICRA48891.2023.10160691.
104. Connolly, L.; Deguet, A.; Leonard, S.; Tokuda, J.; Ungi, T.; Krieger, A.; Kazanzides, P.; Mousavi, P.; Fichtinger, G.; Taylor, R.H. Bridging 3D Slicer and ROS2 for Image-Guided Robotic Interventions. *Sensors* **2022**, *22*. doi:10.3390/s22145336.
105. Li, J.; Xu, B.; Schwertfeger, S. High-Quality, ROS Compatible Video Encoding and Decoding for High-Definition Datasets, 2024, [arXiv:cs.RO/2408.00538].
106. Leal, D.P.; Sugaya, M.; Amano, H.; Ohkawa, T. FPGA Acceleration of ROS2-Based Reinforcement Learning Agents. 2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW), 2020, pp. 106–112. doi:10.1109/CANDARW51189.2020.00031.
107. Lopez, N.G.; Nuin, Y.L.E.; Moral, E.B.; Juan, L.U.S.; Rueda, A.S.; Vilches, V.M.; Kojcev, R. gym-gazebo2, a toolkit for reinforcement learning using ROS 2 and Gazebo. *ArXiv* **2019**, *abs/1903.06278*.
108. Orsula, A.; Bøgh, S.; Olivares-Mendez, M.; Martinez, C. Learning to Grasp on the Moon from 3D Octree Observations with Deep Reinforcement Learning. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 4112–4119. doi:10.1109/IROS47612.2022.9981661.
109. Agarwal, S.; Muthukrishnan, R.; Gosrich, W.; Kumar, V.; Ribeiro, A. LPAC: Learnable Perception-Action-Communication Loops with Applications to Coverage Control, 2024, [arXiv:cs.RO/2401.04855].
110. Nuin, Y.L.E.; Lopez, N.G.; Moral, E.B.; Juan, L.U.S.; Rueda, A.S.; Vilches, V.M.; Kojcev, R. ROS2Learn: a reinforcement learning framework for ROS 2, 2019. doi:10.48550/ARXIV.1903.06282.
111. Almeida, F.; Leão, G.; Sousa, Armando editor=Marques, L.; Santos, C.; Lima, J.L.; Tardioli, D.; Ferre, M. An Educational Kit for Simulated Robot Learning in ROS 2. Robot 2023: Sixth Iberian Robotics Conference; Springer Nature Switzerland: Cham, 2024; pp. 513–525.
112. Abawi, F.; Allgeuer, P.; Fu, D.; Wermter, S. Wrapyfi: A Python Wrapper for Integrating Robots, Sensors, and Applications across Multiple Middleware. Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI '24). ACM, 2024. doi:10.1145/3610977.3637471.
113. Peng, B.; Hasegawa, A.; Azumi, T. Scheduling Performance Evaluation Framework for ROS 2 Applications. 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), 2022, pp. 2031–2038.

 doi:10.20944/preprints202410.1204.v1

114. Ichnowski, J.; Chen, K.; Dharmarajan, K.; Adebola, S.; Danielczuk, M.; Mayoral-Vilches, V.; Jha, N.; Zhan, H.; LLontop, E.; Xu, D.; Kubiatowicz, J.; Stoica, I.; Gonzalez, J.; Goldberg, K. FogROS2: An Adaptive Platform for Cloud and Fog Robotics Using ROS 2, 2022. doi:10.48550/ARXIV.2205.09778.
115. Shen, M.Y.; Hsu, C.C.; Hou, H.Y.; Huang, Y.C.; Sun, W.F.; Chang, C.C.; Liu, Y.L.; Lee, C.Y. DriveEnv-NeRF: Exploration of A NeRF-Based Autonomous Driving Environment for Real-World Performance Validation, 2024, [arXiv:cs.RO/2403.15791].
116. Mayoral-Vilches, V.; Jabbour, J.; Hsiao, Y.S.; Wan, Z.; Crespo-Álvarez, M.; Stewart, M.; Reina-Muñoz, J.M.; Nagras, P.; Vikhe, G.; Bakhshalipour, M.; Pinzger, M.; Rass, S.; Panigrahi, S.; Corradi, G.; Roy, N.; Gibbons, P.B.; Neuman, S.M.; Plancher, B.; Reddi, V.J. RobotPerf: An Open-Source, Vendor-Agnostic, Benchmarking Suite for Evaluating Robotics Computing System Performance, 2024, [arXiv:cs.RO/2309.09212].
117. Kuboichi, T.; Hasegawa, A.; Peng, B.; Miura, K.; Funaoka, K.; Kato, S.; Azumi, T. CARET: Chain-Aware ROS 2 Evaluation Tool. 2022 IEEE 20th International Conference on Embedded and Ubiquitous Computing (EUC), 2022, pp. 1–8. doi:10.1109/EUC57774.2022.00010.
118. Bédard, C.; Lütkebohle, I.; Dagenais, M. ros2_tracing: Multipurpose Low-Overhead Framework for Real-Time Tracing of ROS 2. *IEEE Robotics and Automation Letters* **2022**, *7*, 6511–6518. doi:10.1109/LRA.2022.3174346.
119. Moore, M.; Sooknanan, J.; Holley, J. Bobble-Bot: An educational platform for real-time control with ROS. 2019 IEEE International Symposium on Measurement and Control in Robotics (ISMCR), 2019, pp. B3–2–1–B3–2–7. doi:10.1109/ISMCR47492.2019.8955713.
120. Salimi, S.; Queralta, J.P.; Westerlund, T. Hyperledger Fabric Blockchain and ROS 2 Integration for Autonomous Mobile Robots. 2023 IEEE/SICE International Symposium on System Integration (SII), 2023, pp. 1–8. doi:10.1109/SII55687.2023.10039326.
121. Vilches, V.M.; Juan, L.U.S.; Dieber, B.; Carbajo, U.A.; Gil-Uriarte, E. Introducing the robot vulnerability database (rvd). *arXiv preprint arXiv:1912.11299* **2019**.
122. Vizzo, I.; Guadagnino, T.; Mersch, B.; Wiesmann, L.; Behley, J.; Stachniss, C. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)* **2023**, *8*, 1029–1036. doi:10.1109/LRA.2023.3236571.
123. Mendia, G.O.; Juan, L.U.S.; Bascaran, X.P.; Calvo, A.B.; Cordero, A.H.; Ugarte, I.Z.; Rosas, A.M.; Vilches, D.M.; Carbajo, U.A.; Kirschgens, L.A.; Vilches, V.M.; Gil-Uriarte, E. Robotics CTF (RCTF), a playground for robot hacking, 2021, [arXiv:cs.CY/1810.02690].
124. Vilches, V.M.; White, R.; Caiazza, G.; Arguedas, M. SROS2: Usable Cyber Security Tools for ROS 2, 2022. doi:10.48550/ARXIV.2208.02615.
125. Vilches, V.M.; Mendia, G.O.; Baskaran, X.P.; Cordero, A.H.; Juan, L.U.S.; Gil-Uriarte, E.; de Urabain, O.O.S.; Kirschgens, L.A. Aztarna, a footprinting tool for robots. *arXiv e-prints* **2018**, p. arXiv:1812.09490, [arXiv:cs.CR/1812.09490].
126. Morales, C.; Fuenzalida, C.; Sierra, G. CFV2: an Open-Source Robot Controller Board for Education and Research. 2023 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 2023, pp. 1–6. doi:10.1109/CHILECON60335.2023.10418618.
127. Mayr, M.; Rovida, F.; Krueger, V. SkiROS2: A Skill-Based Robot Control Platform for ROS. 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023, pp. 6273–6280. doi:10.1109/IROS55552.2023.10342216.
128. Jo, W.; Kim, J.; Wang, R.; Pan, J.; Senthilkumaran, R.K.; Min, B.C. SMARTmBOT: A ROS2-based Low-cost and Open-source Mobile Robot Platform, 2022. doi:10.48550/ARXIV.2203.08903.
129. Probe, A.B.; Deans, M.C. Space ROS-ACO: Space Robot Operating System. Game Changing Development Anual Review, 2022.
130. Szabó, G. Toward the Automatic Network Resource Management of Robot Operating System in Programmable Mobile Networks. *IEEE Access* **2023**, *11*, 65934–65955. doi:10.1109/ACCESS.2023.3289922.
131. Karle, P.; Török, F.; Geisslinger, M.; Lienkamp, M. MixNet: Physics Constrained Deep Neural Motion Prediction for Autonomous Racing. *IEEE Access* **2023**, *11*, 85914–85926. doi:10.1109/ACCESS.2023.3303841.
132. Karle, P.; Fent, F.; Huch, S.; Sauerbeck, F.; Lienkamp, M. Multi-Modal Sensor Fusion and Object Tracking for Autonomous Racing. *IEEE Transactions on Intelligent Vehicles* **2023**, pp. 1–13. doi:10.1109/TIV.2023.3271624.
133. Walker, V.; Vanegas, F.; Gonzalez, F. NanoMap: A GPU-Accelerated OpenVDB-Based Mapping and Simulation Package for Robotic Agents. *Remote Sensing* **2022**, *14*. doi:10.3390/rs14215463.

134. Horváth, E.; Pozna, C.; Kőrös, P.; Hajdu, C.; Ballagi, A. Theoretical background and application of multiple goal pursuit trajectory follower. *Hungarian Journal of Industry and Chemistry* **2020**, *48*, 11–17. doi:10.33927/hjic-2020-03.
135. Unger, M.; Horváth, E.; Pup, D.; Pozna, C.R. Towards Robust LIDAR Lane Clustering for Autonomous Vehicle Perception in ROS 2. 2024 IEEE International Conference on Mobility, Operations, Services and Technologies (MOST), 2024, pp. 229–234. doi:10.1109/MOST60774.2024.00031.
136. Macenski, S.; Booker, M.; Wallace, J. Open-Source, Cost-Aware Kinematically Feasible Planning for Mobile and Surface Robotics, 2024, [arXiv:cs.RO/2401.13078].
137. Martín, F.; Clavero, J.G.; Matellán, V.; Rodríguez, F.J. PlanSys2: A Planning System Framework for ROS2. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 9742–9749. doi:10.1109/IROS51168.2021.9636544.
138. GonzÃąlez-Santamarta, M.Ã.; RodrÃ guez-Lera, F.J.; Olivera, V.M. SAILOR: Perceptual Anchoring For Robotic Cognitive 1526 Architectures, 2023. [arXiv:cs.RO/2303.08204].
139. González-Santamarta, M.Á.; Rodríguez-Lera, F.J.; Matellán-Olivera, V.; Fernández-Llamas, C. YASMIN: Yet Another State MachINe. ROBOT2022: Fifth Iberian Robotics Conference; Tardioli, D.; Matellán, V.; Heredia, G.; Silva, M.F.; Marques, L., Eds.; Springer International Publishing: Cham, 2023; pp. 528–539.
140. Blanco-Claraco, J.L. A flexible framework for accurate LiDAR odometry, map manipulation, and localization, 2024, [arXiv:cs.RO/2407.20465].
141. Drwięga, M.; Jakubiak, J. A set of depth sensor processing ROS tools for wheeled mobile robot navigation. *Journal of Automation, Mobile Robotics & Intelligent Systems (JAMRIS)* **2017**. doi:10.14313/JAMRIS_2-2017/16.
142. Fernández-Becerra, L.; González-Santamarta, M.A.; Sobrín-Hidalgo, D.; Guerrero-Higueras, Á.M.; Lera, F.J.R.; Olivera, Vicente Matellán editor=García Bringas, P.; Pérez García, H.; Martínez de Pisón, F.J.; Martínez Álvarez, F.; Troncoso Lora, A.; Herrero, Á.; Calvo Rolle, J.L.; Quintián, H.; Corchado, E. Accountability and Explainability in Robotics: A Proof of Concept for ROS 2- And Nav2-Based Mobile Robots. International Joint Conference 16th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2023) 14th International Conference on EUropean Transnational Education (ICEUTE 2023); Springer Nature Switzerland: Cham, 2023; pp. 3–13.
143. Galvis, J.; Pediaditis, D.; Almazrouei, K.S.; Aspragathos, N. An Autonomous Navigation Approach based on Bird's-Eye View Semantic Maps. 2023 27th International Conference on Methods and Models in Automation and Robotics (MMAR). IEEE, 2023, pp. 81–86.
144. Do, H.V.; Kim, Y.H.; Lee, J.H.; Lee, M.H.; Song, J.W. DeRO: Dead Reckoning Based on Radar Odometry With Accelerometers Aided for Robot Localization. 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)., to be published, 2024.
145. Atas, F.; Cielniak, G.; Grimstad, L. Elevation State-Space: Surfel-Based Navigation in Uneven Environments for Mobile Robots. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 5715–5721. doi:10.1109/IROS47612.2022.9981647.
146. Atas, F.; Grimstad, L.; Cielniak, G. Evaluation of Sampling-Based Optimizing Planners for Outdoor Robot Navigation. *CoRR* **2021**, *abs/2103.13666*, [2103.13666].
147. Grupp, M. evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo, 2017.
148. Leitenstern, M.; Sauerbeck, F.; Kulmer, D.; Betz, J. FlexMap Fusion: Georeferencing and Automated Conflation of HD Maps with OpenStreetMap, 2024, [arXiv:cs.RO/2404.10879].
149. Lvov, G.; Zolotas, M.; Hanson, N.; Allison, A.; Hubbard, X.; Carvajal, M.; Padir, T. Mobile MoCap: Retroreflector Localization On-The-Go, 2023, [arXiv:cs.RO/2303.13681].
150. Martin, F.; Guerrero, J.M.; Garcia, A.A.; Rodriguez, F.; Matellan, V. MOCAP4ROS2: An Open Source Framework for Motion Capture Systems in Robotics. Proceedings of the 18th International Symposium on Open Collaboration; Association for Computing Machinery: New York, NY, USA, 2022; OpenSym '22. doi:10.1145/3555051.3555076.
151. Serov, A.; Clemens, J.; Schill, K. Multi-Robot Graph SLAM Using LIDAR. 2024 10th International Conference on Automation, Robotics and Applications (ICARA), 2024, pp. 339–346. doi:10.1109/ICARA60736.2024.10553070.

152. Goelles, T.; Schlager, B.; Muckenhuber, S.; Haas, S.; Hammer, T. 'pointcloudset': Efficient Analysis of Large Datasets of Point Clouds Recorded Over Time. *Journal of Open Source Software* **2021**, *6*, 3471. doi:10.21105/joss.03471.
153. Zutell, J.M.; Conner, D.C.; Schillinger, P. ROS 2-Based Flexible Behavior Engine for Flexible Navigation. SoutheastCon 2022, 2022, pp. 674–681. doi:10.1109/SoutheastCon48659.2022.9764047.
154. Kampmann, A.; Wüstenberg, A.; Alrifaee, B.; Kowalewski, S. A Portable Implementation of the Real-Time Publish-Subscribe Protocol for Microcontrollers in Distributed Robotic Applications. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019, pp. 443–448. doi:10.1109/ITSC.2019.8916835.
155. Leal, D.P.; Sugaya, M.; Amano, H.; Ohkawa, T. Automated Integration of High-Level Synthesis FPGA Modules with ROS2 Systems. 2020 International Conference on Field-Programmable Technology (ICFPT), 2020, pp. 292–293. doi:10.1109/ICFPT51103.2020.00052.
156. Lienen, C.; Platzner, M. Design of Distributed Reconfigurable Robotics Systems with ReconROS. *ACM Trans. Reconfigurable Technol. Syst.* **2022**, *15*. doi:10.1145/3494571.
157. Lienen, C.; Platzner, M.; Rinner, B. ReconROS: Flexible Hardware Acceleration for ROS2 Applications. 2020 International Conference on Field-Programmable Technology (ICFPT), 2020, pp. 268–276. doi:10.1109/ICFPT51103.2020.00046.
158. Chen, K.; Yuan, J.; Jha, N.; Ichnowski, J.; Kubiatowicz, J.; Goldberg, K. FogROS G: Enabling Secure, Connected and Mobile Fog Robotics with Global Addressability, 2022, [arXiv:cs.RO/2210.11691].
159. Ichnowski, J.; Chen, K.; Dharmarajan, K.; Adebola, S.; Danielczuk, M.; Mayoral-Vilches, V.; Jha, N.; Zhan, H.; Llontop, E.; Xu, D.; Buscaron, C.; Kubiatowicz, J.; Stoica, I.; Gonzalez, J.; Goldberg, K. FogROS2: An Adaptive Platform for Cloud and Fog Robotics Using ROS 2. 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 5493–5500. doi:10.1109/ICRA48891.2023.10161307.
160. Chen, K.; Hoque, R.; Dharmarajan, K.; LLontop, E.; Adebola, S.; Ichnowski, J.; Kubiatowicz, J.; Goldberg, K. FogROS2-SGC: A ROS2 Cloud Robotics Platform for Secure Global Connectivity, 2023, [arXiv:cs.RO/2306.17157].
161. Ichnowski, J.; Chen, K.; Dharmarajan, K.; Adebola, S.; Danielczuk, M.; Mayoral-Vilches, V.; Jha, N.; Zhan, H.; LLontop, E.; Xu, D.; Kubiatowicz, J.; Stoica, I.; Gonzalez, J.; Goldberg, K. FogROS2: An Adaptive Platform for Cloud and Fog Robotics Using ROS 2, 2022. doi:10.48550/ARXIV.2205.09778.
162. Bédard, C.; Lajoie, P.Y.; Beltrame, G.; Dagenais, M. Message flow analysis with complex causal links for distributed ROS 2 systems. *Robotics and Autonomous Systems* **2023**, *161*, 104361. doi:10.1016/j.robot.2022.104361.
163. Enright, D.; Xiang, Y.; Choi, H.; Kim, H. PAAM: A Framework for Coordinated and Priority-Driven Accelerator Management in ROS 2, 2024, [arXiv:cs.RO/2404.06452].
164. Mayoral-Vilches, V.; Neuman, S.M.; Plancher, B.; Reddi, V.J. RobotCore: An Open Architecture for Hardware Acceleration in ROS 2. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 9692–9699. doi:10.1109/IROS47612.2022.9982082.
165. Sarabakha, A. anafi_ros: from Off-the-Shelf Drones to Research Platforms, 2023, [arXiv:cs.RO/2303.01813].
166. Mudalige, N.D.W.; Zhura, I.; Babataev, I.; Nazarova, E.; Fedoseev, A.; Tsetserukou, D. HyperDog: An Open-Source Quadruped Robot Platform Based on ROS2 and micro-ROS, 2022. doi:10.48550/ARXIV.2209.09171.
167. Nyboe, F.F.; Malle, N.H.; Ebeid, E. MPSoC4Drones: An Open Framework for ROS2, PX4, and FPGA Integration. 2022 International Conference on Unmanned Aircraft Systems (ICUAS), 2022, pp. 1246–1255. doi:10.1109/ICUAS54217.2022.9836055.
168. Silva, G.R.; Päßler, J.; Zwanepol, J.; Alberts, E.; Tarifa, S.L.T.; Gerostathopoulos, I.; Johnsen, E.B.; Corbato, C.H. SUAVE: An Exemplar for Self-Adaptive Underwater Vehicles. 2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2023, pp. 181–187. doi:10.1109/SEAMS59076.2023.00031.
169. Palmer, E.; Holm, C.; Hollinger, G. Angler: An Autonomy Framework for Intervention Tasks with Lightweight Underwater Vehicle Manipulator Systems. IEEE International Conference on Robotics and Automation, 2024.
170. Li, Z.; Hasegawa, A.; Azumi, T. Autoware_Perf: A tracing and performance analysis framework for ROS 2 applications. *Journal of Systems Architecture* **2022**, *123*, 102341. doi:https://doi.org/10.1016/j.sysarc.2021.102341.

171. Sivashangaran, S.; Eskandarian, A. XTENTH-CAR: A Proportionally Scaled Experimental Vehicle Platform for Connected Autonomy and All-Terrain Research, 2022, [arXiv:cs.RO/2212.01691].
172. González-Santamarta, M.Á.; Rodríguez-Lera, F.J.; Álvarez-Aparicio, C.; Guerrero-Higueras, Á.M.; Fernández-Llamas, C. MERLIN a cognitive architecture for service robots. *Applied Sciences* **2020**, *10*, 5989.
173. González-Santmarta, M.A.; Rodríguez-Lera, F.J.; Fernández-Llamas, C.; Matellán-Olivera, V. MERLIN2: MachinEd Ros 2 pLanINg. *Software Impacts* **2023**, *15*, 100477. doi:https://doi.org/10.1016/j.simpa.2023.100477.
174. Serrano-Muñoz, A.; Elguea-Aguinaco, Í.; Chrysostomou, D.; Bøgh, S.; Arana-Arexolaleiba, N. A Scalable and Unified Multi-Control Framework for KUKA LBR iiwa Collaborative Robots. 2023 IEEE/SICE International Symposium on System Integration (SII). IEEE, 2023, pp. 1–5.
175. Zamalloa, I.; Muguruza, I.; Hernández, A.; Kojcev, R.; Mayoral, V. An information model for modular robots: the Hardware Robot Information Model (HRIM). *arXiv preprint arXiv:1802.01459* **2018**.
176. Heggem, C.; Wahl, N.M.; Tingelstad, L. Configuration and Control of KMR iiwa Mobile Robots using ROS2. 2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS), 2020, pp. 1–6. doi:10.1109/SIMS49386.2020.9121554.
177. Huber, M.; Mower, C.E.; Ourselin, S.; Vercauteren, T.; Bergeles, C. LBR-Stack: ROS 2 and Python Integration of KUKA FRI for Med and IIWA Robots, 2023, [arXiv:cs.RO/2311.12709].
178. Winiarski, T. MeROS: SysML-based Metamodel for ROS-based Systems, 2023, [arXiv:cs.RO/2303.08254].
179. Sears, T.M.C.; Cooper, M.R.; Button, S.R.; Marshall, J.A. OtterROS: Picking and Programming an Uncrewed Surface Vessel for Experimental Field Robotics Research with ROS 2, 2024, [arXiv:cs.RO/2404.05627].
180. Mosteo, A.R. RCLAda, or Bringing Ada to the Robot Operating System. *Ada Lett.* **2020**, *39*, 35–40. doi:10.1145/3394514.3394518.
181. Kim, S.; Kim, T. RoboFuzz: fuzzing robotic systems over robot operating system (ROS) for finding correctness bugs. Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering; Association for Computing Machinery: New York, NY, USA, 2022; ESEC/FSE 2022, p. 447–458. doi:10.1145/3540250.3549164.
182. Merzlyakov, A.; Macenski, S. A Comparison of Modern General-Purpose Visual SLAM Approaches. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 9190–9197. doi:10.1109/IROS51168.2021.9636615.
183. Amano, H.; Mori, H.; Mizutani, A.; Ono, T.; Yoshimoto, Y.; Ohkawa, T.; Tamukoh, H. A dataset generation for object recognition and a tool for generating ROS2 FPGA node. 2021 International Conference on Field-Programmable Technology (ICFPT), 2021, pp. 1–4. doi:10.1109/ICFPT52863.2021.9609880.
184. Hallyburton, R.S.; Hunt, D.; Luo, S.; Pajic, M. A Multi-Agent Security Testbed for the Analysis of Attacks and Defenses in Collaborative Sensor Fusion, 2024, [arXiv:cs.RO/2401.09387].
185. Rosende, S.B.; Gavilán, D.S.J.; Fernández-Andrés, J.; Sánchez-Soriano, J. An Urban Traffic Dataset Composed of Visible Images and Their Semantic Segmentation Generated by the CARLA Simulator. *Data* **2024**, *9*. doi:10.3390/data9010004.
186. Maresca, F.; Grazioli, F.; Albanese, A.; Sciancalepore, V.; Negri, G.; Costa-Perez, X. Are you a robot? Detecting Autonomous Vehicles from Behavior Analysis, 2024, [arXiv:cs.RO/2403.09571].
187. Guo, Z.; Lykov, A.; Yagudin, Z.; Konenkov, M.; Tsetserukou, D. Co-driver: VLM-based Autonomous Driving Assistant with Human-like Behavior and Understanding for Complex Road Scenes, 2024, [arXiv:cs.RO/2405.05885].
188. Cañadas-Aránega, F.; Blanco-Claraco, J.L.; Moreno, J.C.; Rodriguez-Diaz, F. Multimodal Mobile Robotic Dataset for a Typical Mediterranean Greenhouse: The GREENBOT Dataset. *Sensors* **2024**, *24*. doi:10.3390/s24061874.
189. Bosello, M.; Aguiari, D.; Keuter, Y.; Pallotta, E.; Kiade, S.; Caminati, G.; Pinzarrone, F.; Halepota, J.; Panerati, J.; Pau, G. Race Against the Machine: A Fully-Annotated, Open-Design Dataset of Autonomous and Piloted High-Speed Flight. *IEEE Robotics and Automation Letters* **2024**, *9*, 3799–3806. doi:10.1109/LRA.2024.3371288.
190. Puccetti, T.; Nardi, S.; Cinquilli, C.; Zoppi, T.; Ceccarelli, A. ROSPaCe: Intrusion Detection Dataset for a ROS2-Based Cyber-Physical System and IoT Networks. *Scientific Data* **2024**, *11*, 481. doi:10.1038/s41597-024-03311-2.
191. Grigioni, C.; Corradini, F.; Antonucci, A.; Guzzi, J.; Flammini, F. Safe Road-Crossing by Autonomous Wheelchairs: a Novel Dataset and its Experimental Evaluation, 2024, [arXiv:cs.RO/2403.08984].

192. Anjum, M.L.; Ahmad, O.; Rosa, S.; Yin, J.; Bona, B. Skeleton tracking based complex human activity recognition using kinect camera. Social Robotics: 6th International Conference, ICSR 2014, Sydney, NSW, Australia, October 27-29, 2014. Proceedings 6. Springer, 2014, pp. 23–33.
193. Lajoie, P.Y.; Beltrame, G. Swarm-SLAM : Sparse Decentralized Collaborative Simultaneous Localization and Mapping Framework for Multi-Robot Systems, 2023, [arXiv:cs.RO/2301.06230].
194. Mortimer, P.; Maehlisch, M. Survey on Datasets for Perception in Unstructured Outdoor Environments, 2024, [arXiv:cs.CV/2404.18750].
195. Lu, J.; Hossain, S.; Lam, W.; Sheng, W.; Bai, H. A Research Testbed for Intelligent and Cooperative Driving in Mixed Traffic. *IEEE Transactions on Intelligent Transportation Systems* **2024**, pp. 1–13. doi:10.1109/TITS.2024.3375297.
196. Park, J.; Shim, J.; Lee, G.; Choi, S. Deep Reinforcement Learning based Real-Time Path Planning and Flight Validation of small UAS Application. AIAA SCITECH 2024 Forum, 2024, p. 1711.
197. Chen, L.; Wang, Y.; Miao, Z.; Feng, M.; Zhou, Z.; Wang, H.; Wang, D. Reciprocal Velocity Obstacle Spatial-Temporal Network for Distributed Multirobot Navigation. *IEEE Transactions on Industrial Electronics* **2024**, pp. 1–11. doi:10.1109/TIE.2024.3379630.
198. Djizi, H.; Lakehal, A.; Zahzouh, Z. A quadrotor controlled in real-time using hand gestures and ROS2 multi-node communication within GAZEBO 3D environment. *International Journal of Automation and Control* **2024**, *18*, 214–231, [https://www.inderscienceonline.com/doi/pdf/10.1504/IJAAC.2024.137069]. doi:10.1504/IJAAC.2024.137069.
199. Almusayli, A.; Zia, T.; Qazi, E.u.H. Drone Forensics: An Innovative Approach to the Forensic Investigation of Drone Accidents Based on Digital Twin Technology. *Technologies* **2024**, *12*. doi:10.3390/technologies12010011.
200. Mohan, A.; Krishnan, A.R. Design and Simulation of an Autonomous Floor Cleaning Robot with Optional UV Sterilization. 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), 2022, pp. 1–6. doi:10.1109/MysuruCon55714.2022.9972558.
201. Pandey, K.K.; Shah, T.; Yadrave, S.; Shinde, S.; Pagare, V. Design and Development of an Autonomous Robot Assistant. Intelligent Manufacturing Systems in Industry 4.0; Springer Nature Singapore: Singapore, 2023.
202. Kołcon, T.; Malki, A.; Maciaś, M. Smart Warehouse as an Example of Micro-ROS Application. Automation 2022: New Solutions and Technologies for Automation, Robotics and Measurement Techniques; Szewczyk, R.; Zieliński, C.; Kaliczyńska, M., Eds. Springer International Publishing, 2022.
203. SobrÃ n-Hidalgo, D.; GonzÃąlez-Santamarta, M.A.; ÃĄngel M. Guerrero-Higueras.; RodrÃ guez-Lera, F.J.; MatellÃąn-Olivera, V. Explaining Autonomy: Enhancing Human-Robot Interaction through Explanation Generation with Large Language Models, 1683 2024. [arXiv:cs.RO/2402.04206].
204. Lages, W.F. Remote Teaching of Dynamics and Control of Robots Using ROS 2. *IFAC-PapersOnLine* **2022**, *55*, 279–284. 13th IFAC Symposium on Advances in Control Education ACE 2022.