Proceedings of the 12th International Conference of Control Systems, and Robotics (CDSR 2025)
July 13, 2025 - July 15, 2025 | Imperial College London Conference Center, London, United Kingdom
Paper No. 121
DOI: 10.11159/cdsr25.121

Evaluating Robotic Operating Systems: A Survey on Enhancing Interoperability and Developer Support

Anshul Ranjan[1], Anoosh Damodar[1], Neha Chougule[1], Dhruva S Nayak[1], P.N Anantharaman[1], Dr. Shylaja S S[1]
[1]PES University
100 Feet Ring Road, Banashankari, Bengaluru, India
pes1202102171@pesu.pes.edu; anooshsd12379@gmail.com; neharajchougule@gmail.com;
dhruvanayak1905@gmail.com; ananth@jnresearch.com; shylaja.sharath@pes.edu

Abstract **-** This paper provides a comprehensive retrospection of the current state of robotics, focusing on enhancing interoperability and developer support. This paper examines ROS 1 and ROS 2, highlighting their evolution in design, architecture, and real-world use, while addressing new challenges developers face in many different robotics ecosystems. We examine recent developments in cloud-based frameworks such as RoboKube and FogROS2, which make cloud-robotics integration seamless. This paper also provides detailed information on communication protocols such as Cyclone DDS and Zenoh, emphasizing their role in improving time and multi-system performance. We analyse these systems and propose strategies to improve efficiency, performance, and ease of use, thereby laying the foundation for the next generation of robotics software frameworks. The goal of the survey is to provide researchers and clinicians with important insights into the future of robotics and to foster innovation and collaboration within the human community.

Keywords**:** Robotics, Robot Operating Systems, Cloud Robotics, Middleware, Multi-Agent Systems, Real-Time Systems, Communication Protocols, DDS, Zenoh

1. Introduction

The fast development of robots and the Internet of Things (IoT) has had a revolutionary effect on mankind, with an anticipated 75 billion linked devices by 2025. But this has its own set of difficulties as well. The intricacy of robotic systems presents formidable obstacles. It gets harder to create software for robots as they get more powerful and adaptable. Traditional software development and reuse approaches are insufficient for today's robotic systems, which require specific operating systems made for the Internet of Things and robotics [1]. There is no need to develop a universal software solution because every robot has a unique hardware setup.

In light of this, next-generation work that is appropriate for robots and the Internet of Things is required. The growing demand for safe, dependable, and expandable platforms that can manage the diverse range of sensors and actuators needed by contemporary systems must be satisfied by these systems. robotics day. Even with the success of systems like Robot Operating System 2 (ROS 2) [2], it is still very difficult to achieve smooth compatibility and interoperability across many platforms and versions. Despite the advancements in ROS 2, it still faces challenges in the integration of different hardware components and the complexity of robotics software development.

The aim of this study is to identify and evaluate a real open-source robotics environment by focusing on ROS 2 technology as the basis for a next-generation robotics framework. The aim is to provide a middleware solution that solves the communication and interoperability problems of multiple robotic systems while meeting the evolving needs of complex robotic applications. The middleware processes have facilitated the development of robotic software by providing flexible and adaptable processes, while also facilitating interaction with various physical objects. It also provides developers with simple design concepts and tools to create advanced robotic applications with unique functionality and performance. These middlewares offer new opportunities to create capable, high-performance robotic systems as a groundbreaking alternative to resource-constrained platforms.

2. Current ROS implementations and their features

2.1. Overview of the First ROS Release

ROS deviates from the concept of a traditional operating system by acting as a middleware communication layer [1]. Its deployment across heterogeneous computing clusters and comparison to existing robot software frameworks are noteworthy aspects. The modular architecture of ROS permits developers to construct flexible, utility-based software. Moreover, it promotes non-redundancy in design, allowing for further development by other parties. Consequently, ROS is recognized as a comprehensive application platform for diverse hardware configurations, research contexts, and execution requirements, thereby fostering advancements in robotic technology and innovation [1].

The philosophical goals of ROS can be summarized as follows:

- Peer-to-peer communication model
- Tools-based development approach
- Support for multiple programming languages
- Lightweight architecture
- Free and open-source distribution

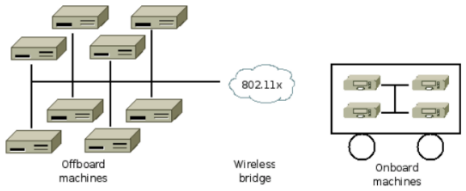


Fig. 1: A typical ROS network configuration.

2.2. ROS 2: Evolution in Design, Architecture, and Practical Applications

Robot Operating System 2, or ROS 2, is a cutting-edge open-source framework designed to offer a stable and flexible platform for creating robotic applications, meeting the diverse requirements of the robotics community [2]. In contrast to ROS, which was its predecessor, ROS 2 brings several improvements. These improvements include increased adaptability, instantaneous response times, and strengthened security features like transport-level encryption that protect the privacy and integrity of data sent between nodes. Whereas ROS 1 relied on the TCP protocol, ROS 2 chose a more dependable and scalable protocol, DDS (Data Distribution Service), as its main communication middleware. Even while ROS 2 outperforms ROS in the aforementioned areas, it still has certain drawbacks [2].

The paper presents five case studies across different domains (land, sea, air, space, large-scale), showing how ROS 2 has accelerated and helped facilitate the deployment of robots in real-world environments.

It provides security, real-time assistance, and multi-robot communication based on the Data Distribution Service (DDS) standard. Better security, reliability, and performance are only a few of the architectural changes and improvements of ROS 2 over ROS 1 that are covered in the article. This shows how important ROS2 is for robotics, and how it improves the deployment of real time robots in the real world for a variety of different reasons [1] [2].

2.3. Laying the Groundwork for Cloud-Native Advancements in Robotics

Robokube is a Kubernetes-based framework that aims to make deploying robotics applications via cloud technology straightforward [3]. It tackles issues related to porting traditional robotics apps into cloud-native environments, and untangles the mess of setting up network connectivity for systems spanning devices at the edge all the way to the cloud core. By walking users through containerization and how ROS nodes should be distributed alongside application deployment, RoboKube doesn't aim to do everything [3], it's just enough to kickstart your journey into cloud robotics with ROS. The framework is compatible with most Kubernetes distributions but highly recommends K3s due to its straightforwardness and ability to support edge computing scenarios. Take a tour on how we achieve this using real-

world examples such as cloudifying teleoperation testbed— RoboKube stands out as an enabler for marrying robotics with clouds thus steering the technological shift towards cloud-native developments in robotics [3].

2.4. Advancements in Cloud Robotics Frameworks: FogROS2

In this paper, the authors address the challenges faced by onboard computing resources in robots, which often struggle to keep pace with advancements in robotic algorithms and emerging computing hardware. Earlier work established FogROS (also known as FogROS1) [12], a framework that extends the Robot Operating System (ROS1) to provide rapid access to cloud resources, building based on cloud computing. FogROS1 did, however, have issues with automation, usability, and latency. In response, the current paper presents FogROS2, a revamped framework designed to mitigate latency, enhance usability, and automate further aspects of deploying robotic code in the cloud, while also broadening the range of potential robotic applications.

FogROS2 has been developed from the ground up to achieve full integration with ROS 2, capitalizing on advancements in networking, launch configurability, and command line interface functionality [12]. Additionally, integration points have been established with Foxglove to support remote monitoring capabilities from any location worldwide.

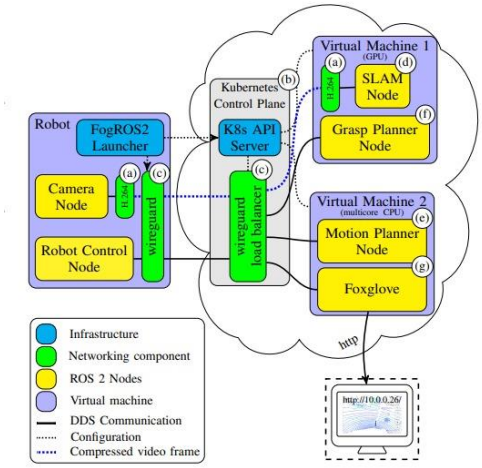


Fig. 2: Real-time robotics control with FogROS2 and DDS.

2.5. Quantitative analysis of communication handling for centralized multi-agent robot systems using ROS2.

The application of multi-agent robot systems, specifically mobile robots operating in dynamic human-interactive settings such as production environments, has experienced considerable growth in recent years [5]. To gain insights into ROS2 communication performance under conditions of high network load, this paper investigates the data handling capabilities of a single tracking node receiving information from multiple robots within a centralized multi-agent system architecture. A quantitative performance analysis is presented comparing two publisher-subscriber communication architectures, employing the ROS2 Galactic framework with CycloneDDS, FastDDS, and GurumDDS as the underlying DDS vendors [5]. The examined architectures comprise a many-to-one configuration, where all robots communicate over a shared topic, and a one-to-one configuration, where each robot communicates via a dedicated topic. The study employed single-computer simulations, increasing the number of simulated robots and their data publishing rate, to evaluate the performance of each DDS implementation. A further simulation using a distributed architecture with CycloneDDS was also conducted [5].

The simulation results indicate that, with an increasing node count, the one-to-one communication approach exhibited significantly reduced average data age and data miss ratios compared to the many-to-one approach. In the context of system startup, CycloneDDS displayed the greatest robustness regarding crashes and response time, while FastDDS demonstrated superior performance in mitigating data ageing. On a single-machine implementation, FastDDS excelled in minimizing update misses and data ageing; however, it was prone to random crashes during system launch and execution, making it the least stable. Consequently, CycloneDDS is favored as a more reliable choice given its absence of crash behavior and its improved metrics of data age, update miss ratio, and CPU utilization compared to GurumDDS [5].

Table 1: Average data age with a 1ms update interval.

Robots	CycloneDDS		FastDDS		GurumDDS	
	M2O	O2O	M2O	O2O	M2O	O2O
10	0.930	0.990	0.580	0.790	0.900	0.910
30	15.520	1.870	0.870	0.690	2.800	2.080
50	70.120	11.400	4.810	1.470	24.470	13.740
70	173.140	16.860	58.370	20.730	355.120	98.540
90	811.260	62.460	-	-	-	-

Table 2: The average data age is calculated using a 1ms update interval.

Robots	CycloneDDS		FastDDS		GurumDDS	
	M2O	O2O	M2O	O2O	M2O	O2O
10	1.334	1.590	1.259	1.122	1.181	1.837
30	2.502	1.788	1.340	1.345	1.671	1.707
50	241.665	2.310	1.710	1.566	405.730	467.093
70	295.255	13.753	4.697	2.865	83.183	1852.359
90	228.581	22.758	-	-	705.112	469.506

2.6. Systematic Gap Analysis of Robot Operating System in Real-time Systems

Today, most of the high-tech industrial robots and autonomous vehicles operate within these immediate constraints. The Robot Operating System (ROS) is designed as a framework that includes open libraries and tools that facilitate the development of robotic applications and devices [4]. A new version of ROS called ROS 2 was developed due to some shortcomings of the old version of ROS in the current meeting. ROS 2 includes features such as Data Distribution Service (DDS) for instant messaging. However, ROS 2 does not have an immediate impact yet and is therefore still in development.

This paper conducts a literature review focusing on previous studies on the real-world performance of Robot Operating System 2 (ROS 2). It addresses the transition to real-time constraints for machine learning and autonomous vehicles and the evolution of ROS as a framework to support these applications. Although ROS 2 introduces improvements such as the Data Distribution Service (DDS) for real-time communication, it still faces challenges in meeting real-time and demand-side requirements, which require continuous improvement [4].

In conclusion, while simulation tools are valuable for evaluating system performance, real-time execution observations remain paramount due to the dynamic nature of distributed embedded systems [4]. Scheduling methods play a critical role in real-time systems, but the trade-off between schedulability and computation overhead necessitates further research to strike a balance.

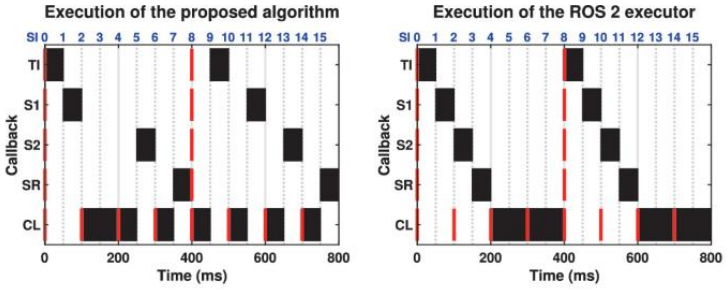
2.7. Priority Scheduling for Periodic Systems Using ROS 2

This paper introduces a new dynamic priority scheduling algorithm for ROS 2 systems. The algorithm calculates callback deadlines based on channel buffer sizes and update rates [10]. We demonstrate its effectiveness with an example, showing a reduction in required buffer size compared to the standard single-threaded ROS 2 executor.

The algorithm assigns deadlines to callbacks based on predicted buffer overflow times. It follows an earliest-deadline-first scheduling approach. Initially, deadlines are set to infinite, but as new data arrives, the algorithm calculates the minimum time difference between arrivals to predict buffer overflow times and set deadlines accordingly [10]. It then updates the set of ready callbacks by scanning input buffers, adds newly arrived data to the ready set, and predicts

the time for the next trigger instance arrival to minimize middleware interactions. It selects the callback with the earliest deadline for execution, prioritizing those with shared deadlines based on buffer utilization and registration order [10].

The suggested scheduling technique maintains a buffer usage of no more than 40%, while the ROS2 system can reach a maximum of 60%. Also, fewer contacts with the RMW are required than in the ROS2 executor, enhancing efficiency.



(a) Execution example of the proposed algorithm (b) Execution example of the ROS 2 executor

Fig. 3: Comparison of the current and proposed algorithms.

3. Analysing Key Communication Protocols

3.1. Performance Insights on Zenoh, MQTT, Kafka, and DDS

With the growing demand for faster response times in applications like the Industrial Internet of Things (IoT), there's a noticeable shift from traditional cloud computing toward the edge of the network. This change is driving the popularity of the publish/subscribe communication model, which has become a go-to method for sharing information. Two of the most widely used protocols in this space are MQTT and Kafka. Additionally, the Data Distribution Service (DDS) has emerged as a key player, serving as the communication protocol for ROS 2 (Robot Operating System 2) [2]. This enables efficient interactions in robotics applications. Known for its speed, reliability, and decentralized nature, DDS is also widely applied in critical industries like military, aerospace, and transportation [5].

In their study, the authors dive into a performance comparison of four prominent communication protocols: Zenoh, MQTT, Kafka, and DDS. They focus specifically on two important performance metrics: throughput and latency. Zenoh, which is a newer protocol based on a data-centric, pub/sub model, stands out for its capabilities in cloud-to-edge and edge computing. This makes it especially relevant for IIoT, robotics, and autonomous systems [6].

The paper not only introduces each protocol but also details the experimental setup and the testing scenarios used in their evaluation. The experiments were set up in two different ways: one on a single machine and the other with multiple machines connected via Ethernet [6]. Researchers measured the average latency and throughput in both scenarios to see how well each protocol performed. For MQTT and Kafka, messages had to go through a broker, while Zenoh used a router to ensure fair comparisons, with both the publisher and subscriber running in client mode. To keep the process smooth, the team created scripts that automated the measurement of throughput and latency, ensuring a consistent and reliable testing environment. The findings reveal that Zenoh outperforms the other protocols in terms of performance. Ultimately, this study aims to provide valuable insights to developers, helping them choose the most appropriate communication protocols for their specific application requirements [6].

3.2. Performance Benefits of Zenoh's Design

Eclipse Zenoh unifies calculations, data in motion, data in use, and data at rest to provide a complete solution. It achieves a degree of time and space efficiency that is superior to mainstream stacks by deftly fusing geo-distributed storage, queries, and computations with conventional publish/subscribe techniques.

One of Zenoh's key strengths is its fully decentralized architecture, which enhances fault tolerance at the communication layer. Zenoh removes the possibility of a single point of failure (SPOF), in contrast to the majority of current pub/sub systems that depend on centralized implementations, such MQTT and Kafka, where brokers are necessary for connecting peers. Accordingly, Zenoh guarantees strong peer-to-peer connectivity without the risk of broker failure, whereas centralized systems may have serious problems. Centralized systems sometimes need extra hardware or software resources, including clustering many machines to increase broker availability, to reduce SPOF concerns [6].

Peer-to-peer communication is made feasible via Zenoh's full distributed architecture, which permits peers to connect directly for data exchange whenever practical. This feature aids in avoiding communication snags that are frequently connected to centralized brokers. The Mesh and Clique peer-to-peer configurations are among the models depicted in the graphic below.

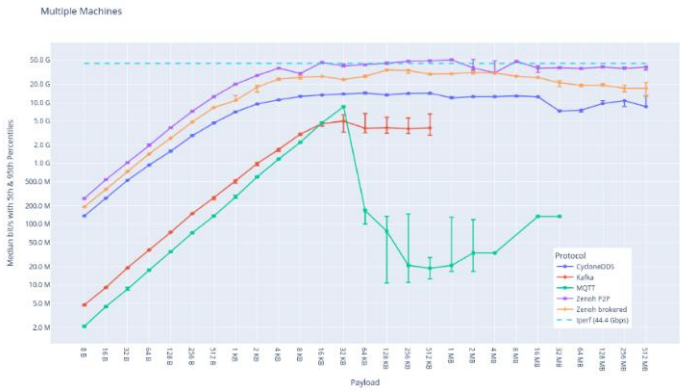


Fig. 4: Throughput data in msg/s for the single-machine scenario.

3.3. Impact of ROS 2 Node Composition in Robotic Systems

Node composition allows developers to combine multiple nodes into a single process, either manually or dynamically, while keeping their development modular and distributed [7]. This composition model significantly enhances the performance of resource-constrained robotic systems by optimizing memory usage, reducing latency, and enabling large-scale processing tasks, such as those seen in sensor data pipelines. By benchmarking the Component node against traditional multi-process architectures, the research demonstrates notable improvements in both computation and memory utilization, especially in applications that involve numerous nodes or require processing large amounts of sensor data, like images and point clouds [7].

The paper benchmarks different methods of node composition, including manual and dynamic, comparing them with standard multi-process systems. Through these experiments, the paper evaluates memory usage, CPU load, latency, and data throughput (goodput) on embedded systems such as the Raspberry Pi 4 [7]. The findings show that node composition drastically reduces the memory and CPU overhead by minimizing communication delays between nodes. Additionally, the use of intra-process communication (IPC) and zero-copy techniques further boosts performance, particularly for large message sizes. These improvements make ROS 2 composition a crucial tool for real-time robotic applications. The paper also explores the role of executors in task scheduling and system performance, outlining the different executor models available in ROS 2 that enable developers to fine-tune system execution [7].

3.4. ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation

The ALLIANCE architecture is designed for fault-tolerant and adaptive control of multi-robot teams, enabling them to perform cooperative tasks in dynamic and unpredictable environments [11]. This architecture is fully distributed, meaning each robot in the team makes decisions independently without relying on centralized control. The core concept

of ALLIANCE revolves around allowing heterogeneous mobile robots to autonomously select their tasks and actions based on the overall mission objectives, the status of other robots, and the current environmental conditions [11].

In ALLIANCE, each robot is equipped with multiple behavior sets, where each set corresponds to a high-level task the robot can perform. The architecture uses behavior-based control, where tasks are represented by behavior sets that include specific actions, such as avoiding obstacles, picking up objects, or navigating to a location. The activation of these behaviors is regulated by a set of motivational behaviors, which determine when a robot should start or stop a task [11].These motivational behaviours are determined by internal factors, notably impatience, which is exhibited by a robot taking over a task when it perceives the original robot is not completing it efficiently, and acquiescence, which is a robot's tendency to abandon a task if it is not performing optimally or if a robot is better suited. These motivations are essential for allowing robots to respond adaptively to failures, inefficiencies, or dynamic alterations in the environment. For example, if a robot becomes incapacitated or experiences sensor malfuntion, another robot can takeover its task without the necessity of centralized coordination.

Within this architecture, decision-making is driven by robots actively and continuously evaluating the environmental conditions and their teammates' behaviors, while simultaneously broadcasting their individual actions to the entire team. Instead of having direct one-on-one discussions, the robots interact via broadcast messages, simplifying the communication structure and allowing for higher scalability. This architecture allows robots to flexibly reallocate labor, recover from individual robot failures, and assure mission accomplishment without human intervention. The inherent fault tolerance of ALLIANCE is a key component that enables the team to carry on efficiently even in the case of a robot failure by reassigning its responsibilities to other participating robots [11].

4. Conclusion

We provide an in-depth analysis of the Robotics Operating System (ROS 2), point out its advantages and disadvantages, and sketch out potential directions for future research. Key architectural issues that impede the integration of various robotic applications are highlighted by our thorough investigation. Our research has focused on cross-platform operation issues, collaboration, and real-time operations, all of which enable us to come up with practical answers.

Through a deep understanding of the ROS 2 architecture and communications, we are developing strategies to improve reliability and enhance developer support. Focusing on performance and efficiency, we have identified a clear path to improve and expand the capital structure of ROS 2 by moving to upper-mid-range products such as DDS and FogROS2.

Our primary findings highlight the need of addressing limits in real-time communication, optimizing resource utilization in multi-robot systems, and improving interaction with cloud-based robotics. Additionally, we highlight upcoming ROS 2 advancements that will facilitate hardware abstraction, guarantee improved support for a variety of devices, and enhance security protocols.

Essentially, our work lays a strong basis for further development of ROS 2 and its use in the broader robotics domain. By applying our results to enhance robotic frameworks, researchers and developers might push the boundaries of what the next generation of robotic systems can do. The solutions we propose provide the foundation for future robotics advancements by providing a way to enhance developer support, scalability, and interoperability.

Acknowledgements

This research has received valuable support from the Department of Computer Science and Engineering at PES University, located in Bangalore, India Pin: 560085. Their guidance and resources have been instrumental in the successful completion of this work.

References

[1] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, and A. Y. Ng, “ROS: an open-source Robot Operating System,” ResearchGate, vol. 3, 2009.

[2] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, “Robot Operating System 2: Design, architecture, and uses in the wild,” Science Robotics, vol. 7, no. 66, May 2022.

[3] Y. Liu, Aitor Hernandez Herranz, and R. C. Sundin, “RoboKube: Establishing a New Foundation for the Cloud Native Evolution in Robotics,” Feb. 2024.

[4] S. Mobaiyen, “Systematic Gap Analysis of Robot Operating System (ROS 2) in Real-time Systems”, Dissertation, 2022.

[5] Lukas Johannes Dust, Emil Persson, Mikael Ekstrom, Saad Mubeen, and Emmanuel Dean, “Quantitative analysis of communication handling for centralized multi-agent robot systems using ROS2”, 2022 IEEE 20th International Conference on Industrial Informatics (INDIN), Perth, Australia.

[6] Wen-Yew Liang, Yuyuan Yuan and Hsiang-Jui Lin, “A Performance Study on the Throughput and Latency of Zenoh, MQTT, Kafka, and DDS”, arXiv:2303.09419 [cs.DC], Mar. 2023.

[7] Steve Macenski, Alberto Soragna, Michael Carroll and Zhenpeng Ge, “Impact of ROS 2 Node Composition in Robotic Systems”, arXiv:2305.09933 [cs.RO], May. 2023.

[8] Yu-Ping Wang, Yuejiang Dong and Gang Tan, “ROS-SF: A Transparent and Efficient ROS Middleware using Serialization-Free Message”, Middleware '22: Proceedings of the 23rd ACM/IFIP International Middleware Conference, Nov. 2022.

[9] Harun Teper, Tobias Betz, Georg Von Der Brüggen, Kuan-Hsun Chen, Johannes Betz and Jian-Jia Chen, “Timing-Aware ROS 2 Architecture and System Optimization”, 2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Niigata, Japan.

[10] Lukas Dust and Saad Mubeen, “Dynamic Priority Scheduling for Periodic Systems Using ROS 2”, Engineering of Computer-Based Systems. ECBS 2023. Lecture Notes in Computer Science, vol 14390. Springer, Cham.

[11] Minglong Li, Zhongxuan Cai, Xiaodong Yi, Zhiyuan Wang, Yanzhen Wang, Yongjun Zhang and Xuejun Yang “ALLIANCE-ROS: A Software Architecture on ROS for Fault-Tolerant Cooperative Multi-robot Systems”, PRICAI 2016: Trends in Artificial Intelligence. PRICAI 2016. Lecture Notes in Computer Science, vol 9810. Springer, Cham.

[12] J. Ichnowski, K. Chen, K. Dharmarajan, S. Adebola, M. Danielczuk, V. Mayoral-Vilches, N. Jha, H. Zhan, Edith LLontop, D. Xu, C. Buscaron, J. Kubiatowicz, I. Stoica, J. Gonzalez and Ken Goldberg, “FogROS2: An Adaptive Platform for Cloud and Fog Robotics Using ROS 2”, arXiv:2205.09778 [cs.RO], Apr. 2023.

[13] Grosjean, Leefke, J. Sachs, J. Ansari, Norbert Reider, Aitor Hernandez Herranz, and Christer Holmberg. 2025. "A Framework for Communication Compute Control Co-Design in Cyber Physical Systems" Electronics 14, no. 5: 864.

[14] P. M. Gupta, E. Pairet, T. Nascimento, and M. Saska, “Landing a uav in harsh winds and turbulent open waters,” IEEE Robotics and Automation Letters, vol. 8, no. 2, pp. 744–751, 2023.

[15] H. Guo, F. Wu, Y. Qin, R. Li, K. Li, and K. Li, “Recent trends in task and motion planning for robotics: A Survey,” ACM Computing Surveys, Feb. 2023.

[16] A. Mitrevski, P. G. Plöger, and G. Lakemeyer, “A hybrid skill parameterisation model combining symbolic and subsymbolic elements for introspective robots,” Robotics and Autonomous Systems, vol. 161, p. 104350, Dec. 2023.

[17] Professor Lucas Hernandez, Dr. Anna Kim. “Swarm Robotics: Collective Behavior and Scalability Challenge.” Robotics: Science and Policy, 2024.

[18] Z. Yanfei, Y. Tan, Y. Chen, L. Chen, and Kwang Y. Lee, "UAV Path Planning Based on Random Obstacle Training and Linear Soft Update of DRL in Dense Urban Environment" Energies 17, no. 11: 2762, Jun. 2024.

[19] T. Schreiter, A. Rudenko, M. Magnusson, and A. J. Lilienthal, “Human gaze and head rotation during navigation, exploration and object manipulation in shared environments with robots,” in 2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN), 2024, pp. 1258–1265.

[20] T. Schreiter, J. V. Ruppel, R. Hazra, A. Rudenko, M. Magnus-son, and A. J. Lilienthal, “Evaluating efficiency and engagement in scripted and llm-enhanced human-robot interactions,” arXiv:2501.12128, 2025.