Service Discovery-based Hybrid Network Robotic Middleware for Efficient Communication

Shiyao Sang[1] and Yinggang Ling[2]

Abstract— **With the ongoing advancements in artificial intelligence and robotics technology, distributed robotic operating systems have become crucial in fields such as intelligent robotics, autonomous vehicles, and smart manufacturing. These systems rely on efficient robotic middleware for reliable inter-component communication. However, existing solutions still face challenges in managing diverse communication demands, optimizing cross-domain communication efficiency, and ensuring the determinacy of communication scheduling.**

This research introduces an innovative robotic middleware framework designed to overcome these challenges through a hybrid network communication strategy. By flexibly switching the scope of service discovery, the framework supports diverse communication methods ranging from in-process shared pointers to cross-device various Ethernet protocols. A core data stream forwarding mechanism significantly reduces data redundancy and non-deterministic delays in cross-domain communication, thus enhancing communication efficiency and stability. This study not only extends the communication capabilities of robotic operating systems but also offers a new solution path for complex communications in distributed systems.

I. INTRODUCTION

As artificial intelligence and robotics technology[1], [2] rapidly advance, distributed robotic operating systems are increasingly applied in fields like intelligent robotics[3], autonomous driving[4], [5], and smart industry[6]. Compared to traditional dedicated embedded systems, robotic operating systems offer more extensive software and hardware support and facilitate the migration of software programs and algorithms to diverse hardware platforms through the abstraction of hardware interfaces and the unification of software interfaces. This high degree of flexibility and scalability has led to the widespread adoption of robotic operating systems in various sectors.

Robotic middleware, serving as the core communication hub within robotic operating systems, plays a crucial role[7]. It not only provides high-level communication protocols, shielding the complexity of underlying implementations but also automates communication configurations. By intelligently selecting the most suitable communication methods based on the physical relationships between nodes, such as in-process, inter-process, or cross-device communications, it significantly simplifies user configurations.

*This work was supported by Chery and Zdrive.ai

[1]Shiyao Sang conducted the core research at the Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Jiangsu, China, and is currently with Zdrive.ai, a subsidiary of Chery Automobile Co., Ltd., Anhui, China. `www.fengmao@outlook.com`

[2]Yungang Ling is with the Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Jiangsu, China. `958393566@qq.com`

Despite the advantages of existing robotic middleware systems, there are still limitations in the automatic selection of communication methods. The current service discovery-based automatic selection mechanism is confined to a few communication modes and does not adequately meet the diverse requirements of cross-domain communication in terms of transmission delays, throughput, and resource consumption. Moreover, message redundancy often occurs in cross-domain communication, where identical messages create duplicate data streams when transmitted between different devices, particularly increasing network load during the transmission of large messages[8]. Additionally, uncertainties in the transmission process pose challenges to the scheduling of cross-domain communication, affecting time consistency and processing stability[9].

Addressing these issues, this paper proposes an innovative design for a robotic middleware—RIMAOS2C, which possesses hybrid network communication capabilities. By dynamically adjusting the range of service discovery, it supports a variety of communication methods from shared pointers within processes to various Ethernet protocols across devices. The introduced data stream forwarding mechanism effectively reduces data redundancy in cross-domain communication and minimizes non-deterministic delays, thereby significantly enhancing communication efficiency and scheduling certainty.

The main contributions of this paper can be summarized as follows: 1) The design and implementation of the RIMAOS2C, a service discovery-based hybrid network robotic middleware, which has been validated and tested in L4 autonomous driving scenarios; 2) The implementation of a service discovery mechanism that automatically switches among multiple communication methods, effectively resolving the limitations of existing robotic middleware in communication method selection; 3) The optimization of data flow, which addresses the issue of message redundancy in cross-domain communication, significantly improving communication efficiency; 4) The reduction of non-deterministic delays in cross-domain communication scheduling, enhancing the stability and consistency of data processing.

II. RELATED WORK

As robotic systems are increasingly applied in complex environments, particularly in those supporting highly distributed and dynamically changing conditions, the requirements for their communication mechanisms are also rising. Robotic middleware, serving as the critical bridge for seamless interaction between system components[10], plays an essential role. Middleware frameworks such as Data Distribution Service (DDS)[11], Robot Operating System (ROS)[12], and their advanced versions, ROS 2[13] and Cyber RT[14], provide standardized communication interfaces and protocols that significantly simplify development processes, driving technological advancements in fields like industrial automation, autonomous vehicles, and medical robotics. However, despite their success in streamlining development and fostering innovation, these middleware solutions still face challenges in ensuring efficient and reliable cross-domain communication. Addressing these challenges is crucial for enabling more complex and dynamic robotic applications in the future.

A key feature of modern robotic middleware is its ability to automatically discover communication endpoints and select appropriate communication methods based on system configurations and network topology. Traditional approaches, such as the Master node architecture in ROS 1, rely on centralized service discovery, which introduces potential single points of failure and limits system scalability. In contrast, newer frameworks like ROS 2 and Cyber RT employ decentralized mechanisms, such as UDP multicast for service discovery[15], [16], thereby enhancing system robustness and scalability. DDS-based robotic middleware can automatically select suitable communication methods based on the physical relationships between communication nodes and dynamically adjust QoS settings to optimize communication efficiency for different message characteristics and network conditions[17], [8].

However, several challenges remain for robotic middleware. First, due to the physical relationship limitations, systems like ROS 2 and Cyber RT support only three types of communication: intra-process, inter-process on the same device, and inter-process across different devices. As a result, current mainstream robotic middleware supports only these three communication modes for adaptive selection, lacking the ability to dynamically accommodate additional communication methods.

Cross-domain communication also presents issues, particularly in ensuring the robustness and efficiency of robotic systems. One significant challenge is data redundancy, where identical messages are transmitted multiple times between nodes, increasing network traffic and resource utilization[18]. This redundancy is especially pronounced in scenarios involving large data loads, such as sensor data streams or high-resolution images.

Another challenge in cross-domain communication is the uncertainty associated with communication scheduling. Fluctuations in network latency and message transmission times can lead to inconsistencies in callback execution, affecting the temporal consistency and reliability of data processing pipelines. Addressing these uncertainties requires complex synchronization mechanisms and real-time scheduling algorithms to ensure deterministic behavior and temporal accuracy in distributed robotic systems.

ROS 1 implements a topic-based publish-subscribe model using TCPROS and UDPROS libraries, supporting communication via shared pointers and TCP. However, due to the unreliability of UDP communication, it is typically used only in specific scenarios. During development, ROS 2 considered using ZeroMQ[19] as an alternative middleware component[20], and some community members experimented with implementing ROS based on ZeroMQ[21]. However, this approach was ultimately abandoned due to ZeroMQ's lack of support for inter-process zero-copy communication and its inadequate service discovery performance. After extensive testing of various DDS middleware options[22], [23], such as Fast DDS[24] and RTI DDS[25], ROS 2 eventually implemented an abstraction layer that, while sacrificing some performance, ensures compatibility with all DDS implementations[26]. ROS 2, built on DDS, supports shared pointers, shared memory, and network communication.

In contrast, the Cyber RT framework used in the Apollo project developed its own shared memory system due to early FastDDS implementations not supporting shared memory communication. Cyber RT implements shared pointers, shared memory, and DDS-based network communication. Additionally, due to the unicast implementation of DDS communication within the middleware framework, data redundancy among multiple subscribers to the same topic remains an issue in cross-domain communication, as seen in the ROS 1 era. Moreover, uncertainties in multi-path data communication callback scheduling persist due to the instability of Ethernet transmission.

To address these challenges, recent research has begun exploring the use of advanced network protocols such as QUIC[27] and SCTP[28] to improve network transmission latency and congestion control. The QUIC protocol combines the low latency of UDP with the reliability of TCP, enhancing transmission performance. SCTP supports multi-stream transmission and advanced congestion control algorithms, significantly reducing network latency and the impact of scheduling fluctuations. Additionally, Software-Defined Networking (SDN)[29] and Network Function Virtualization (NFV)[30] offer unprecedented flexibility in managing network resources for robotic middleware, allowing dynamic resource allocation based on real-time needs. Simultaneously, the development of edge computing and fog computing[31] enables data processing closer to the data source, thereby reducing overall data transmission volumes, alleviating the burden on middleware, and enhancing the real-time data processing capabilities of distributed robotic systems.

Despite progress in optimizing network protocols, current approaches are often tailored to specific application scenarios and fail to comprehensively address the complexity and dynamism of cross-domain communication in robotic middleware. Existing solutions frequently lack systematic integration, making them ineffective in dealing with data redundancy and scheduling uncertainties under diverse communication requirements. Therefore, this study proposes a comprehensive framework designed to provide a straightforward and effective solution to address the diversity, efficiency, and scheduling challenges in cross-domain commu-

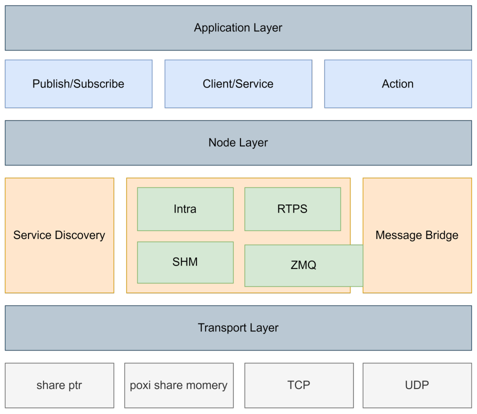

Fig. 1. RIMAOS System Architecture Diagram

nication. This framework not only meets current application needs but also provides a solid foundation for future robotic systems operating in increasingly complex and dynamic environments.

III. System Design

The proposed robotic middleware in this study inherits and extends the core concepts of the ROS series of robotic middleware, aiming to provide an efficient communication solution for distributed robotic systems based on a node network. This middleware framework is divided into three layers: the application layer, the node layer, and the transport layer.

- Application Layer: This layer provides end users with a series of communication protocol interfaces, including but not limited to publish/subscribe mode, request/response mode, and action mode, to support different types of data interaction and control logic.
- Node Layer: In this layer, nodes recognize each other through a service discovery mechanism and automatically select the most suitable communication method based on the network topology and communication needs. This layer implements core functions of the system, including service discovery, node management, and message routing.
- Transport Layer: Responsible for the specific communication implementation, supporting various communication methods such as intra-process shared pointers, shared memory based on POSIX interfaces, and network communication based on UDP and TCP. The flexibility and diversity of this layer ensure that the robotic middleware can adapt to various communication needs and network environments.

As shown in the Figure 2, the robotic middleware framework includes a basic middleware core and a message bridge to achieve flexible and efficient data exchange. The core components support shared pointers, shared memory, and UDP-based DDS communication, as well as TCP-based ZeroMQ[19] communication as a plugin. These components can intelligently select the most appropriate communication

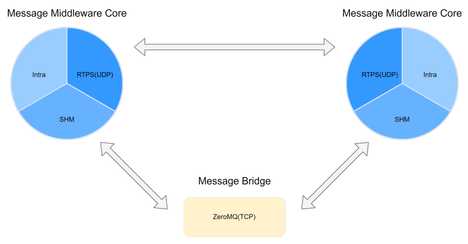

Fig. 2. RIMAOS Communication Design Diagram

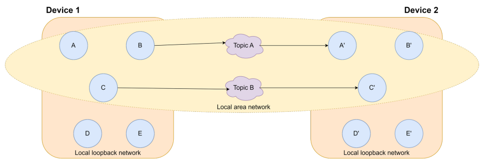

Fig. 3. Multi-level Service Discovery Diagram

method based on service discovery results and the physical location relationship between nodes.

A. Service Discovery

Service discovery is the cornerstone of robotic middleware, allowing nodes in the system to dynamically discover and identify each other. This middleware adopts a UDP multicast mechanism similar to ROS 2 and Cyber RT to achieve automatic discovery and state synchronization between nodes. This mechanism not only improves the system's dynamism and robustness but also ensures real-time and data consistency through continuous state broadcasting and updates.

For cross-domain communication, this framework achieves intelligent classification and identification of publishers and subscribers by configuring different multicast networks, thereby switching to appropriate Ethernet communication protocols (such as DDS[11] or ZeroMQ[19]) when necessary to optimize communication efficiency and stability.

B. Message Bridge

To bridge the gap between different communication protocols, this study introduces the concept of a message bridge. A message bridge is a special node that can subscribe to

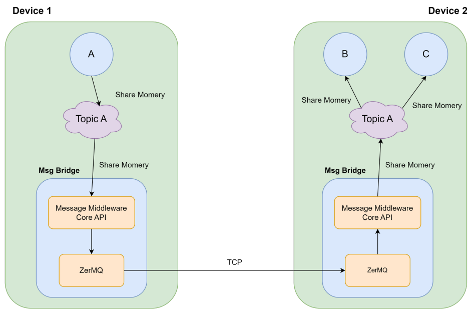

Fig. 4. Implementation Details of Message Bridge

other nodes' messages on the local host and transmit them across the network to another host with minimal delay and maximum efficiency using shared memory and ZeroMQ's advanced zero-copy capabilities. This design not only enhances communication flexibility and scalability but also significantly reduces network load and resource consumption by minimizing redundant message transmission. Moreover, the introduction of the message bridge optimizes latency fluctuations in cross-domain communication, thereby improving the determinism and stability of message processing.

IV. Implementation and Evaluation

RIMAOS2C is the second-generation C-type implementation of the RIMAOS system. This system is based on the Cyber RT middleware from the Apollo project, with improvements aimed at maximizing the use of existing middleware resources while minimizing the impact on current autonomous driving software and algorithms, thereby reducing development costs and time. Although RIMAOS2C inherits Cyber RT's middleware, Cyber RT merely serves as a carrier for the RIMAOS design philosophy. RIMAOS2C introduces new functionalities that further develop and optimize the existing middleware architecture.

RIMAOS2C has been tested on x86 architecture desktop machines equipped with Intel i7 12700T processors and 32GB of memory, as well as on Jetson Orin devices with 8 Cortex-A78 processors and 32GB of memory, demonstrating good compatibility across different hardware architectures. Moreover, RIMAOS2C has been successfully deployed and operated on L3 and L4 autonomous vehicles based on Jetson Orin chips, achieving outstanding operational results.

To comprehensively evaluate the new features of RIMAOS2C, we designed a series of experiments to assess its performance in automatic communication mode switching, improving cross-domain communication efficiency, and optimizing the consistency of cross-domain message callbacks.

A. Multi-Communication Mode Evaluation

TABLE I
LATENCY COMPARISON FOR DIFFERENT COMMUNICATION METHODS

Size (μs)	**1 KB**	**10 KB**	**100 KB**	**1 MB**
SHM	181	210	375	991
ZMQ	2182	2427	3223	12783
Fast DDS	1340	1310	1948	11023

Size (μs)	**2 MB**	**6 MB**	**10 MB**
SHM	1779	4399	N/A
ZeroMQ	23341	66475	106357
Fast DDS	21635	65299	109073

The evaluation of latency across multiple communication modes is designed to verify RIMAOS2C's support for adaptive switching between multiple communication modes and to conduct basic performance evaluations. RIMAOS2C middleware can automatically switch between four communication modes, while in contrast, Cyber RT and ROS 2 only support three. Additionally, RIMAOS2C's support is evident in that whether using ZeroMQ or Fast DDS on the same or different devices, the same topic name can be used without requiring a change to avoid cross-domain subscription issues during distributed deployment. Thanks to the optimization of RIMAOS2C's service discovery network, the selection of communication modes across different devices is more flexible, without needing to change topic names due to cross-domain subscription restrictions.

This experiment was conducted on a Jetson Orin autonomous driving domain controller, and the communication latency was measured by averaging 200 communication trials. The experiment assumed that a subscriber node subscribes to a publisher node, with the subscriber automatically selecting the appropriate communication mode based on different physical relationships and establishing topic-based communication with the publisher node. The middleware natively supports shared pointers, shared memory, and Fast DDS communication, while ZeroMQ communication is indirectly accessed through shared memory via Message Bridge. By segmenting the service discovery network, automatic switching between different communication networks is achieved, supporting multiple communication modes.

The experimental results show that RIMAOS2C can support various communication modes, including shared memory and multiple Ethernet protocols, and that performance indicators across these modes meet the expectations for robotic and autonomous driving systems, demonstrating the middleware's flexibility and adaptability. It should be noted that since the basic capability of shared pointers is achieved at the microsecond level, it is not separately listed here. Additionally, we do not intend to compare the transmission performance of ZeroMQ and Fast DDS in this paper due to the impact of Message Bridge's internal implementation, the version of ZeroMQ and Fast DDS, and the support of kernel protocols, making the evaluation of performance superiority potentially inaccurate. We originally planned to optimize the transmission of large messages using different communication modes, but due to underlying implementation issues with different protocols and libraries, further empirical research is needed. However, this is not the core goal of this multi-communication mode experiment but rather a direction for future research.

B. Cross-Domain Communication Efficiency Assessment

TABLE II
CROSS-DOMAIN COMMUNICATION EFFICIENCY TEST RESULTS

Size (μs)	**1KB**	**10KB**	**100KB**	**1MB**	**2MB**	**6MB**
Sub1	2338	2521	3635	13431	24939	68721
Sub2	2331	2521	3637	13418	24938	68721
Sub3	1154	1185	2996	21036	39982	114807
Sub4	1157	1265	2533	20970	39603	114116

The assessment of cross-domain communication efficiency aims to evaluate whether RIMAOS2C's cross-domain message forwarding design can significantly improve the transmission efficiency of cross-domain communication. In scenarios such as robotics, autonomous driving, and intelligent manufacturing, multiple subscriber nodes often need to subscribe across domains to messages from a publisher node located on another device. This situation can lead to a large number of identical data streams in the network channel, increasing network communication load and device computational resource usage, thereby affecting the communication latency of the middleware. This paper validates these effects through experimentation.

This experiment was conducted on a Jetson Orin autonomous driving domain controller. Communication latency was measured by averaging the results of 200 communication trials. The experiment assumed a situation where two subscriber nodes subscribe across domains to the same publisher node. On device A, the publishers Pub1 and Pub2 publish different topics with the same data stream, while on device B, the subscribers Sub1, Sub2, Sub3, and Sub4 subscribe to the same data stream across these topics. Sub1 and Sub2 on device A subscribe to the Pub1 topic forwarded by Message Bridge, which is transmitted via ZeroMQ and redistributed using shared memory. Sub3 and Sub4 subscribe to Pub2's data stream via Fast DDS unicast communication. The Table II presents the results of the communication efficiency tests.

The experimental results show that RIMAOS2C middleware effectively reduces data stream redundancy in cross-domain communication, significantly improving communication efficiency. This effect is more pronounced in scenarios where multiple subscribers subscribe to the same publisher. Compared with existing middleware solutions, RIMAOS2C reduces communication latency by approximately 30% to 50% when handling messages larger than 100KB. Furthermore, this design effectively reduces the network communication load, making it particularly important for applications requiring the cross-domain transmission of large messages.

C. Cross-Domain Message Scheduling Stability Analysis

TABLE III
AVERAGE TIMESTAMP DIFFERENCE IN CROSS-DOMAIN COMMUNICATION

Average Difference (μs)	**1KB**	**10KB**	**100KB**
Bridge	54.09	55	53.47
Unicast	92.48	116.02	467.89

Average Difference (μs)	**1MB**	**2MB**	**6MB**
Bridge	70.95	68.28	86.48
Unicast	255.69	746.67	773.81

The stability analysis of cross-domain message scheduling aims to evaluate whether RIMAOS2C's cross-domain message forwarding design can effectively reduce scheduling uncertainty in cross-domain communication. Due to the instability of Ethernet transmission and the scheduling uncertainty of the robotic operating system, the time at which cross-domain messages arrive at another device may fluctuate. This fluctuation can cause variations in the activation of callbacks receiving the same topic, thereby affecting the convergence of time differences across multiple information sources within the callback chain of the robotic operating system.

This experiment was conducted on a Jetson Orin autonomous driving domain controller, and communication latency was measured by averaging the results of 200 communication trials. The experiment assumed a situation where two subscriber nodes subscribe across domains to the same publisher node. On device A, the publishers Pub1 and Pub2 publish different topics with the same data stream. On device B, subscribers Sub1, Sub2, Sub3, and Sub4 subscribe to the same data stream across these topics. Sub1 and Sub2 on device A subscribe to the Pub1 topic forwarded by Message Bridge, which is transmitted via ZeroMQ and redistributed using shared memory. Sub3 and Sub4 subscribe to Pub2's data stream via Fast DDS unicast communication. Bridge refers to the average absolute difference in message timestamps between Sub1 and Sub2, while Unicast refers to the average absolute difference in message timestamps between Sub3 and Sub4.

As shown in the Table III, RIMAOS2C middleware, through cross-domain message forwarding, effectively reduces scheduling uncertainty in cross-domain communication, significantly improving communication stability. Compared with Cyber RT's native Fast DDS unicast communication, RIMAOS2C can reduce the time difference in callback triggering by 50% to 1000%. This is crucial for enhancing system reliability and predictability. Since ROS 2 is also based on DDS, similar callback inconsistencies may occur in ROS 2.

Based on these experimental results, RIMAOS2C middleware not only excels in supporting multiple communication modes but also demonstrates significant advantages in the efficiency and stability of cross-domain communication. These features are not achievable by Cyber RT or ROS 2. RIMAOS2C further inherits and expands upon the advantages of existing robotic operating system middleware, bringing new perspectives and innovative ideas to the field of robotics research and application. Whether in a laboratory environment or in practical autonomous vehicle applications, this middleware can effectively meet the demands of distributed communication. Currently, RIMAOS2C has been successfully applied to several L3 and L4 autonomous driving projects, such as Robotaxi and autonomous cleaning vehicles, and has undergone hundreds of real-world road tests, demonstrating its reliability and performance in real scenarios.

V. DISCUSSION

Although RIMAOS2C made relatively small improvements to the existing robotic middleware architectures (such as Cyber RT and ROS 2), these improvements achieved significant results in distributed communication network topology, distributed communication efficiency, and mitigating the scheduling uncertainty caused by unstable Ethernet communication. These achievements provide forward-looking insights for the design of complex communication architectures in the future.

First, RIMAOS2C's network isolation, built through different service discovery mechanisms, enables complex node networks to adaptively switch between different communication modes, thereby facilitating adaptive decision-making in distributed communication. This capability is critical for handling diverse and dynamic communication needs.

Second, the experiments explored using different transport layer protocols to construct a complex transmission network. Although the optimal protocol combination or hybrid approach has yet to be determined due to factors such as the version of the foundational communication component library, the middleware software architecture, and Linux kernel support, it is foreseeable that fully leveraging different transport layer protocols to enhance communication efficiency will become a key feature of high-performance robotic middleware in the future.

Finally, the experimental results further underscore the importance of constructing local communication domains and shared message pools within robotic middleware. By designing an efficient message forwarding mechanism, not only is communication efficiency improved, and communication load reduced, but the stability and consistency of the scheduling system are also significantly enhanced. These findings provide valuable references and practical guidance for the design of middleware in robotic systems.

VI. Conclusion

In this study, we designed and implemented a novel robotic middleware system that adopts a service discovery-based hybrid network communication mechanism, effectively addressing several key challenges in current technology. By flexibly adjusting the broadcast range of the service discovery network, this middleware effectively solves the compatibility issues of various Ethernet communication modes and significantly reduces data redundancy and uncertainty in cross-domain communication through an efficient message forwarding mechanism.

Experimental results show that compared with native Cyber RT, this middleware supports more types of communication in its adaptive selection process, and demonstrates significant performance improvements in handling large-scale message communication, managing cross-domain message loads, and ensuring the determinism of cross-domain message scheduling. These improvements not only enhance the overall efficiency of the middleware but also provide strong support for efficient and reliable communication in complex robotic systems.

Although current test results indicate that the middleware performs excellently, we recognize that there is still room for further optimization. Future work will focus on exploring more advanced distributed communication architectures to address issues of symmetry and generality between distributed nodes, and on studying the application of hybrid transmission protocols to fully exploit the advantages of different protocols under the constrained bandwidth and computational resources of embedded devices. We believe that as research continues to advance, robotic middleware with complex network architectures and hybrid communication protocols will play an increasingly important role in the development of intelligent robotic systems.

Acknowledgment

During the writing and research process of this paper, I received a great deal of help and support from many people. Here, I would like to express my heartfelt gratitude to them. The completion of this work would not have been possible without their encouragement and assistance.

First, I would like to thank my mentors. They not only taught me a lot about computer science and robotics but also imparted the art of engineering and how to conduct meaningful research. These teachings are invaluable and irreplaceable.

Secondly, I want to thank my parents and friends. Their encouragement and recognition have been one of the driving forces behind my continued research in a niche field. In the face of challenges and difficulties, the support and recognition from loved ones have brought me great joy and motivation.

Finally, I want to thank the robotics and internet communities. During my research, I encountered many difficulties, but the great open-source community and selfless programmers provided me with substantial help. I will continue to uphold this spirit and assist others in the community.

Once again, I sincerely thank everyone who has provided help and support during this research process!

References

[1] N. J. Nilsson, *Principles of Artificial Intelligence*. Springer Science & Business Media, 1982.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] E. Tsardoulias and P. Mitkas, "Robotic frameworks, architectures and middleware comparison," Nov. 2017.

[4] M. Reke, D. Peter, J. Schulte-Tigges, S. Schiffer, A. Ferrein, T. Walter, and D. Matheis, "A self-driving car architecture in ROS2," in *2020 International SAUPEC/RobMech/PRASA Conference*. IEEE, 2020, pp. 1–6.

[5] L. Belluardo, A. Stevanato, D. Casini, G. Cicero, A. Biondi, and G. Buttazzo, "A Multi-Domain Software Architecture for Safe and Secure Autonomous Driving," in *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. Houston, TX, USA: IEEE, Aug. 2021, pp. 73–82.

[6] M. Soori, R. Dastres, B. Arezoo, and F. Karimi Ghaleh Jough, "Intelligent robotic systems in Industry 4.0: A review," *Journal of Advanced Manufacturing Science and Technology*, pp. 2 024 007–0, 2024.

[7] W. D. Smart, "Is a common middleware for robotics possible?" in *Proceedings of the IROS 2007 Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware*. Citeseer, 2007.

[8] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ROS2," in *Proceedings of the 13th International Conference on Embedded Software*. Pittsburgh Pennsylvania: ACM, Oct. 2016, pp. 1–10.

[9] Y. Saito, T. Azumi, S. Kato, and N. Nishio, "Priority and Synchronization Support for ROS," in *2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*. Nagoya, Japan: IEEE, Oct. 2016, pp. 77–82.

[10] A. Elkady and T. Sobh, "Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography," *Journal of Robotics*, vol. 2012, no. 1, p. 959013, 2012.

[11] R.-T. Innovations, "OMG Data-Distribution Service: Architectural Overview."

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3. Kobe, Japan, 2009, p. 5.

[13] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, May 2022.

[14] "Introduction — Cyber RT Documents documentation," https://cyber-rt.readthedocs.io/en/latest/index.html.

[15] "ROS on DDS," https://design.ros2.org/articles/ros_on_dds.html.

[16] "Topological Discovery and Communication Negotiation," https://design.ros2.org/articles/discovery_and_negotiation.html.

[17] J. M. Cruz, A. Romero-Garcés, J. P. B. Rubio, R. M. Robles, and A. B. Rubio, "A DDS-based middleware for quality-of-service and high-performance networked robotics," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 16, pp. 1940–1952, 2012.

[18] T. Kronauer, J. Pohlmann, M. Matthe, T. Smejkal, and G. Fettweis, "Latency Analysis of ROS2 Multi-Node Systems," Jun. 2021.

[19] "ZeroMQ," https://zeromq.org/.

[20] "ZeroMQ and Friends," https://design.ros2.org/articles/ros_with_zeromq.html.

[21] R. Joshi, "Ravijo/ros_zeromq_tutorial," Jul. 2024.

[22] "ROS 2 Default RMW TSC reports," https://osrf.github.io/TSC-RMW-Reports/.

[23] "New Fast DDS Performance Testing - Next Generation ROS," https://discourse.ros.org/t/new-fast-dds-performance-testing/29539, Jan. 2023.

[24] "DDS API — Fast DDS 2.14.3 documentation," https://fast-dds.docs.eprosima.com/en/stable/.

[25] "Infrastructure Software for Smart-World and Autonomous Systems | RTI," https://www.rti.com/industries.

[26] "ROS 2 middleware interface," https://design.ros2.org/articles/ros_middleware_interface.html.

[27] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. Los Angeles CA USA: ACM, Aug. 2017, pp. 183–196.

[28] J. Eklund, K.-J. Grinnemo, and A. Brunstrom, "Using multiple paths in SCTP to reduce latency for signaling traffic," *Computer Communications*, vol. 129, pp. 184–196, Sep. 2018.

[29] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[30] W. Zhang, G. Liu, A. Mohammadkhan, J. Hwang, K. K. Ramakrishnan, and T. Wood, "SDNFV: Flexible and Dynamic Software Defined Control of an Application- and Flow-Aware Data Plane," in *Proceedings of the 17th International Middleware Conference*. Trento Italy: ACM, Nov. 2016, pp. 1–12.

[31] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. Helsinki Finland: ACM, Aug. 2012, pp. 13–16.