



Comparison of Middlewares in Edge-to-Edge and Edge-to-Cloud Communication for Distributed ROS 2 Systems

Jiaqiang Zhang¹ · Xianjia Yu¹ · Sier Ha¹ · Jorge Peña Queralta¹ · Tomi Westerlund¹

Received: 28 September 2023 / Accepted: 18 October 2024
© The Author(s) 2024

Abstract

The increased data transmission and number of devices involved in communications among distributed systems make it challenging yet significantly necessary to have an efficient and reliable networking middleware. In robotics and autonomous systems, the wide application of ROS 2 brings the possibility of utilizing various networking middlewares together with DDS in ROS 2 for better communication among edge devices or between edge devices and the cloud. However, there is a lack of comprehensive communication performance comparison of integrating these networking middlewares with ROS 2. In this study, we provide a quantitative analysis for the communication performance of utilized networking middlewares including MQTT and Zenoh alongside DDS in ROS 2 among a multiple host system. For a complete and reliable comparison, we calculate the latency and throughput of these middlewares by sending distinct amounts and types of data through different network setups including Ethernet, Wi-Fi, and 4G. To further extend the evaluation to real-world application scenarios, we assess the drift error (the position changes) over time caused by these networking middlewares with the robot moving in an identical square-shaped path. Our results show that CycloneDDS performs better under Ethernet while Zenoh performs better under Wi-Fi and 4G. In the actual robot test, the robot moving trajectory drift error over time (96 s) via Zenoh is the smallest. It is worth noting we have a discussion of the CPU utilization of these networking middlewares and the performance impact caused by enabling the security feature in ROS 2 at the end of the paper.

Keywords Edge computing · ROS 2 · ROS middleware · DDS · MQTT · Zenoh

1 Introduction

Efficient and reliable communication protocols for distributed systems are in significant demand with the development of the Internet of Things (IoT) and edge computing. In addition to the enormous amount of data generated in the system, the heterogeneous devices and communication

protocols involved make data sharing and communication among edge devices and the cloud a challenging task. In the field of robotics and autonomous systems, the Robot Operating System (ROS) has become the *de-facto* standard. The updated version for distributed systems, ROS 2, has changed the middleware, enabling robots to more effectively utilize the computational resources of the cloud-edge continuum [1].

In order to adapt to the trend of decentralized and distributed systems in the field of robotics, ROS 2 uses Data Distribution Service (DDS) as its internal communication middleware, namely ROS Middleware (RMW) which provides an abstraction layer to different DDS implementations for communication with the ROS 2 Client Library. DDS [2] is a data-centric middleware solution that emphasizes efficient and reliable data transfer between devices. DDS provides a rich set of Quality of Service (QoS) policies that allow applications to specify the reliability, latency, and throughput requirements of their data. The number of studies and analyses comparing the performance of RMWs is limited [3, 4]. However, these studies are primarily conducted on a single

✉ Jiaqiang Zhang
jiaqiang.zhang@utu.fi

Xianjia Yu
xianjia.yu@utu.fi

Sier Ha
sierha@utu.fi

Jorge Peña Queralta
jopequ@utu.fi

Tomi Westerlund
tovewe@utu.fi

¹ Turku Intelligent Embedded and Robotic Systems (TIERS)
Lab, University of Turku, Turku, Finland

machine and lack a comprehensive system-level analysis. With the penetration of cloud computing and edge computing in the field of robotics, communication between multi-host systems, including within the edge and between the cloud and edges, has become increasingly important. In these cases, the network setups are mostly wireless connections, i.e. Wi-Fi or 4G, rather than Ethernet.

Although DDS addresses the need for real-time and reliable information dissemination, DDS was originally designed to be more suitable for ethernet environments, rather than wireless environments [5, 6]. In an open or wireless environment, DDS discovery protocols could cause the flooding effect [7]. In the IoT, there are other networking middleware used for remote devices or machine-to-machine communication, such as Message Queuing Telemetry Transport (MQTT) and Zenoh. MQTT [8] is a lightweight messaging protocol that uses a publish/subscribe model for data transfer. Zenoh [9] is an open-source protocol and suite of tools for data sharing and communication in distributed systems. Zenoh aims to provide a unified approach to data sharing and communication, regardless of the underlying hardware, network topology, or programming languages used.

Therefore, in Edge-to-Edge and Edge-to-Cloud communications based on ROS 2 among multiple machines, it still remains a question of how the DDS combined with other networking middlewares can perform rather than DDS only.

To address the above issues, in this study, we provide a comprehensive evaluation of the communication performance among multi-host ROS 2 systems, by solely relying on DDS and DDS integrated with MQTT or Zenoh. The quantitative analysis consists of the latency and throughput under various network setups including Ethernet, Wi-Fi, and 4G shown in Fig. 1. Notably, the ROS messages utilized in the above experiment are of distinct types and sizes assuring a sufficient assessment. Additionally, by offering the drift error caused by the aforementioned communication approaches in

the real-world moving experiment shown in Fig. 4, we further extend our evaluation process.

To the best of our knowledge, extant literature lacks a comprehensive examination of how networking middleware affects the performance in a multi-host ROS 2 system. Furthermore, the research landscape has not yet delved into the intricacies of assessing networking middleware performance within the context of tangible, real-world robotic systems. The present study, therefore, serves to bridge this evident gap in the scholarly discourse by conducting a meticulous comparative analysis of networking middleware functionalities utilizing ROS Messages and actualized experimentation on physical robotic platforms.

The rest of the paper is organized as follows. Section 2 provides an overview of the background and related work, discussing the key concepts of the middleware used in ROS systems and presenting the existing literature on the performance study of middleware used in robotics and IoT. Section 3 describes the experimental setup and methodology used to evaluate the performance of these networking middlewares. Section 4 presents the experiment results of DDS, MQTT, and Zenoh, including latency and throughput. We analyze the findings as well. Section 5 discusses CPU usage and the impact of the security feature on performance. Finally, Section 6 concludes the paper.

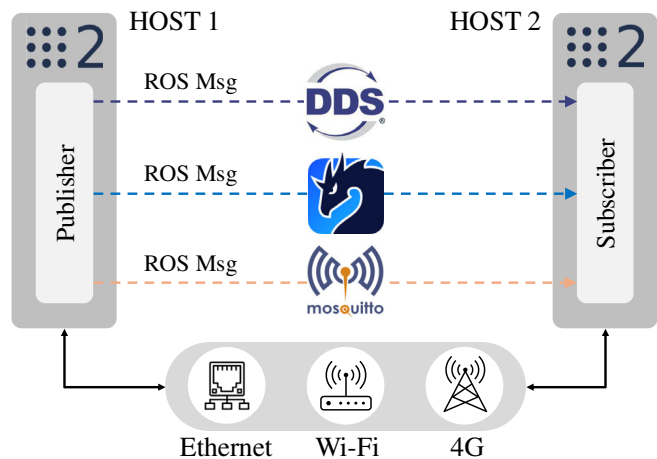
2 Background and Related Work

Through this section, we give a brief introduction to ROS 2 and the most used communication protocols, as well as the existing study on the performance of middleware.

2.1 ROS 2 and SROS 2

Robot Operating System (ROS) has emerged as a popular middleware for developing robotic systems due to its modular

Fig. 1 Proposed evaluation scheme to test DDS, Zenoh, and mosquitto in different network setups



architecture, open-source nature, and large community support. ROS provides a framework for communication between various modules of a robotic system using message passing and service calls. ROS 2, the latest version of ROS, was released in 2015, with an emphasis on real-time systems, large-scale deployments, and distributed systems. ROS 2 has improved modularity and supports different middleware for communication.

SROS 2, short for Secure ROS 2, is a security framework designed to enhance the data protection and communication security of robotic systems built on ROS 2. SROS 2 provides a set of tools, and libraries for providing security, authentication, and access control for ROS 2 entities [10]. In [11], the authors investigate the trade-off between security and performance in ROS 2 middleware. It provides guidelines for selecting appropriate security mechanisms and highlights the importance of a “defense-in-depth” approach. The authors conduct experiments to measure the impact of different security mechanisms on communication latency and throughput.

2.2 Middleware

2.2.1 RMW

ROS Middleware (RMW) is the abstraction layer responsible for facilitating communication between different components of a robotic system [12]. It provides a set of APIs and protocols for sending and receiving data between publishers and subscribers. The use of RMW allows for decoupling of the communication protocol from the application code, enabling interoperability between different components and platforms.

2.2.2 DDS

DDS is a publish-subscribe middleware that provides a data-centric communication model for distributed systems. It is designed for real-time systems and supports QoS settings to optimize communication performance. As the default RMW of ROS 2, DDS provides a decentralized architecture shown in Fig. 2, where nodes can communicate directly with each other without relying on a centralized broker. This architecture ensures reliability, fault tolerance, and scalability.

2.2.3 MQTT

MQTT is a lightweight publish-subscribe protocol designed for constrained devices and low-bandwidth network environments and has almost become a *de-facto* standard in the field of IoT [13, 14]. It supports QoS levels to ensure message delivery and allows for a wide range of applications, including sensors and IoT devices.

2.2.4 Zenoh

Zenoh is a new type of middleware that provides a unified data space for distributed systems. It is designed to provide seamless communication between edge devices, cloud systems, and hybrid environments. Zenoh provides a data-centric communication model and supports different data formats, including structured and unstructured data. It also provides a decentralized architecture and supports fault tolerance and scalability.

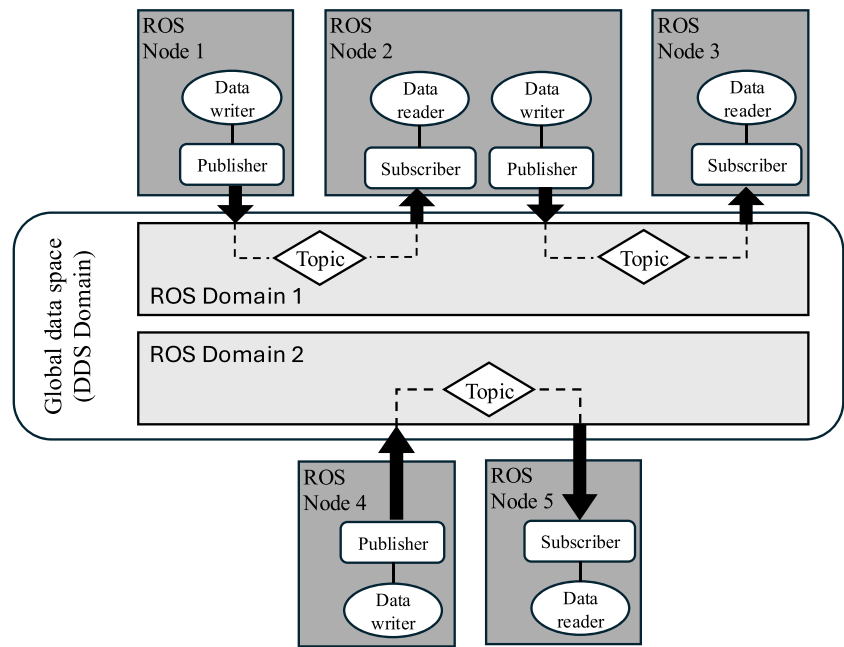
DDS, MQTT, and Zenoh are three different middleware options with distinct features. DDS is a data-centric middleware that offers a decentralized architecture, enabling direct peer-to-peer communication and extensive QoS settings for optimizing reliability, latency, and throughput. MQTT, on the other hand, is a lightweight publish-subscribe protocol designed for resource-constrained devices and networks, with a client-server architecture and low network overhead. It is suitable for IoT applications. Zenoh provides a unified data space, allowing seamless communication across edge devices, cloud systems, and hybrid environments.

2.3 Performance Tools

In this part, we present the tools or packages that are developed to assess the performance of the ROS 2 system.

- The `performance_test` [15] tool tests latency and other performance metrics of various middleware implementations that support a pub/sub pattern. It is used to simulate the non-functional performance of the application.
- The `ros2_tracing` [16] is a tool that provides tracing capabilities within ROS 2. It enables the collection of detailed runtime data, including message passing, data flow, and timing characteristics.
- The `ros-network-analysis` [17] is a ROS package that provides tools to analyze the wireless network such as the signal quality, latency, throughput, link utilization, connection rates, error metrics, etc., between two ROS nodes, computers, or machines.
- CARET (Chain-Aware ROS Evaluation Tool) [18] provides a customizable and analyzable framework for capturing and analyzing real-time execution traces in ROS-based systems. It allows developers to gain a deep understanding of the timing behavior and performance characteristics of their real-time applications, facilitating the development and optimization of reliable and efficient robotic systems.
- The `ros2_latency_evaluation` [19] provides tools and scripts for measuring and analyzing latency in ROS 2 communication between multiple nodes. It includes code that sets up the necessary infrastructure to perform latency measurements, collects latency data during the

Fig. 2 DDS architecture in the ROS 2 system



execution of ROS 2 systems, and provides analysis scripts to process and visualize the collected data.

- The `Autoware_Perf` [20] can measure the callback latency, node latency, and communication latency in ROS 2 applications. In addition, `Autoware_Perf` calculates the end-to-end latency by a convolutional integral as an estimated value.

2.4 Related Work on the Performance of Middleware

Several studies have investigated the performance and characteristics of different middleware solutions in the context of robotic systems and communication frameworks. These studies shed light on various aspects such as latency, throughput, queuing systems, dataflow architectures, and message analysis in distributed environments.

Dhas Y J. et al. [21] provided a qualitative and conceptual overview of IoT protocols and service-oriented middleware, offering insights into their roles, capabilities, and challenges in the Internet of Things domain. Examining a decentralized serverless architecture, [22] addresses the challenges of dataflow in the Cloud-to-Edge continuum. The research presents an architectural solution that can provide insights into optimizing communication and data processing in the context of distributed systems. In [23], the authors focus on message flow analysis in distributed ROS 2 systems, addressing complex causal relationships. The paper's insights into message propagation and interactions contribute to understanding the behavior of ROS Middleware in complex scenarios. L. Puck et al. investigated the real-time performance of ROS 2-based robotic control within a time-synchronized distributed network [24]. In [25], the

authors use a standardized metric and reproducible experimental environment to provide a fair comparison among five message queuing systems, including Kafka, RabbitMQ, RocketMQ, ActiveMQ, and Pulsar. The latency characteristics of multi-node systems within ROS 2 are examined in [19]. It highlights the criticality of low latency in robotics and presents an in-depth analysis of factors affecting latency in ROS 2. The work sheds light on the significance of optimizing communication mechanisms for real-time applications. A comprehensive performance study on Zenoh, MQTT, Kafka, and DDS is provided in [7]. By evaluating throughput and latency, the study contributes to understanding the trade-offs and capabilities of different middleware implementations, thus aiding in informed middleware selection.

3 Experiment Methodology

In this section, we outline the experimental methodology employed to evaluate and compare the performance of DDS, MQTT, and Zenoh in Edge-to-Edge and Edge-to-Cloud communication scenarios. We describe the experimental setup, test scenarios, the equipment utilized for testing, and performance metrics used for comparison.

3.1 Experimental Setup

Our experiments were conducted in a networked environment consisting of a multi-host ROS 2 system. As shown in Fig. 1, the setup includes a combination of devices and communication channels, including Ethernet, Wi-Fi, and 4G connectivity. Each networking middleware (DDS, MQTT,

and Zenoh) was configured and integrated into the ROS 2 framework, enabling communication between the robotic systems. Notably, the RMW is DDS in all experiments. In the experiments of Zenoh and MQTT, we enabled “Localhost Only” and set bridge between DDS and Zenoh or MQTT, so that DDS will publish or subscribe ROS Messages only to or from the local host. In other words, for Zenoh or MQTT, the communication within one host used DDS, and communication between two hosts used Zenoh or MQTT.

Regarding the ROS 2 DDS implementation, in this paper we use CycloneDDS due to its higher compatibility with Zenoh. It is out of the scope of this work to study the performance with different DDS implementations, since there are a number of studies and technical reports dedicated to this matter already in the literature [3, 4, 19]. Owing to the fact that our work focuses on inter-host communication, we configure all hosts to use CycloneDDS as RMW for consistency across experiments.

3.1.1 Hardware

The experiments were conducted in multi-machine scenarios. All devices involved in the experiments run Ubuntu 20.04. The device information is shown in Table 1.

In order to evaluate the performance of different networking middleware implementations on a real-robot platform, we set the networking middlewares between a TurtleBot 4 robot and a laptop. The Turtlebot4 robot is shown in Fig. 3.

As shown Fig. 4, the TurtleBot 4 and the laptop both connect to the same Wi-Fi local network. The laptop sends velocity commands to the TurtleBot 4 to execute a square-shaped trajectory. The TurtleBot 4 will run four identical square circles in 96 seconds. The OptiTrack motion capture (MOCAP) system records the actual trajectory of the TurtleBot 4 during the execution of the square shape.

3.1.2 Network

In order to assess the performance of the networking middleware in different network conditions, we set three different network environments, as shown in Fig. 5. The first setup utilized an Ethernet local network, representing a high-bandwidth and low-latency environment for Edge-to-Edge communication. The LAN Switch used in Ethernet Local

Table 1 Hardware setup

Devices	System	PROCESSOR	Memory
Host 1	Ubuntu 20.04	AMD® Ryzen 7 5800h	16GB
Host 2	Ubuntu 20.04	Intel® Core i3-1215U	16GB
Router/broker	Ubuntu 20.04	Intel® Core i7-9700K	64GB

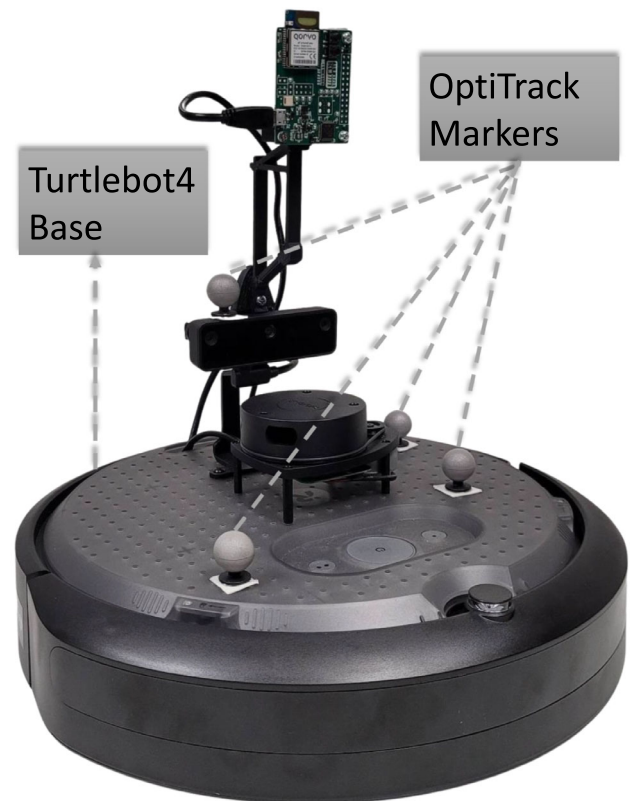


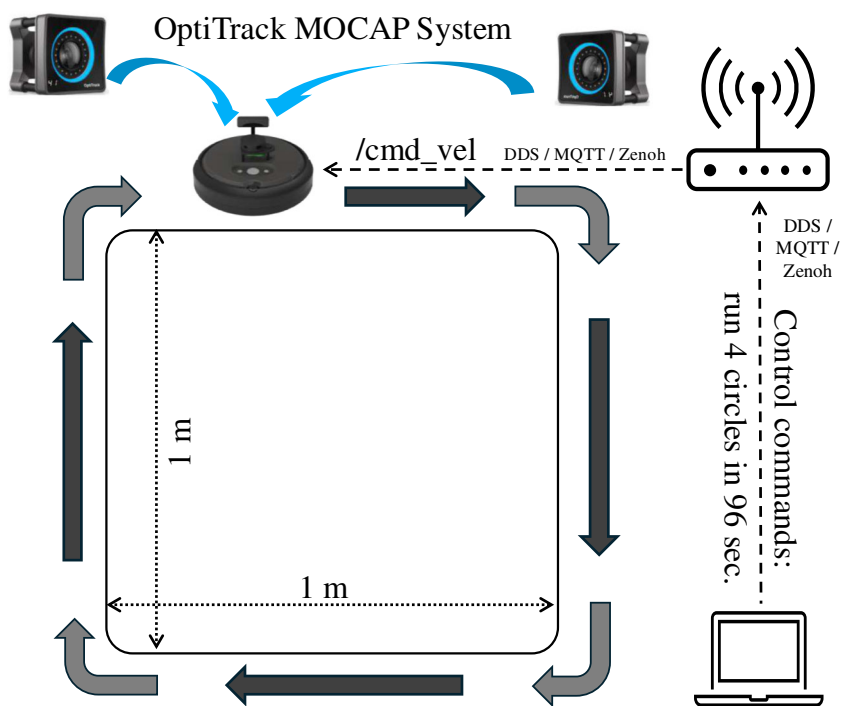
Fig. 3 The Turtlebot 4 robot used in the experiment with OptiTrack markers on top

Network is Unifi Flex Mini. The second setup involved a local network using Wi-Fi connectivity, simulating Edge-to-Edge communication within a confined area. The Wi-Fi Router is Huawei B593. Lastly, we leveraged the Zerotier virtualized network to 4G setup, which emulates Edge-to-Cloud communication scenarios. In this scenario, for HOST 1 & 2, we used HUAWEI 4G Dongle E3372 as 4G module, and for broker, we used Netgear 4G LTE Mobile Router as 4G module.

3.1.3 ROS Message

To comprehensively evaluate the performance of the networking middleware, we employed various types and sizes of ROS Messages in our experiments, including *Array1k*, *Array4k*, *Array16k*, *Array64k*, *Array256k*, *Array1m*, *Array2m*, *PointCloud512k*, *PointCloud1m*, and *PointCloud2m*, where *k* and *m* are *Kilobyte* and *Megabyte*. The ROS Messages were published and subscribed with a frequency of 10 Hz. By using different sizes and types of ROS Messages, we aimed to assess the networking middleware’s ability to handle messages of varying complexities and data sizes, simulating real-world scenarios encountered in robotic systems.

Fig. 4 Actual robot setup. The TurtleBot 4 receives the “/cmd_vel” velocity command from the HOST to run a square trajectory. The linear velocity is set to 0.33 m/s forwarding, and the turning velocity is set to 30 degree/s. The OptiTrack MOCAP System is only used to record the trajectories, which are subsequently employed in calculating the drift errors



By designing our experimental setup to encompass different hardware configurations, network conditions, and message types, we ensured a comprehensive evaluation of the performance of DDS, MQTT, and Zenoh in Edge-to-Edge and Edge-to-Cloud communication scenarios. The chosen setups allowed us to capture the nuances and implications of each networking middleware’s performance under various conditions, facilitating a robust comparison and analysis of their capabilities.

3.2 Performance Metrics and Tool

We focus on two key performance metrics including latency and throughput, following the methodology outlined by the DDS foundation, particularly for publish/subscribe protocols

[26]. The performance tool selected for this research is performance_test [15]. The primary objective of this work is to compare the performance of different networking middlewares in Edge-to-Edge and Edge-to-Cloud environments. The performance_test tool is specifically designed for benchmarking and performance evaluation, making it well-suited for this purpose. It provides standardized methods for generating synthetic message traffic, measuring latency and throughput, and facilitating direct performance comparisons between different middlewares. In addition, the performance_test supports SROS 2 with a simple setup.

The use of the performance_test, enabled accurate measurement of latency and throughput. To ensure reliable data collection, we configured the QoS reliability setting to be reliable, guaranteeing message delivery without loss. The

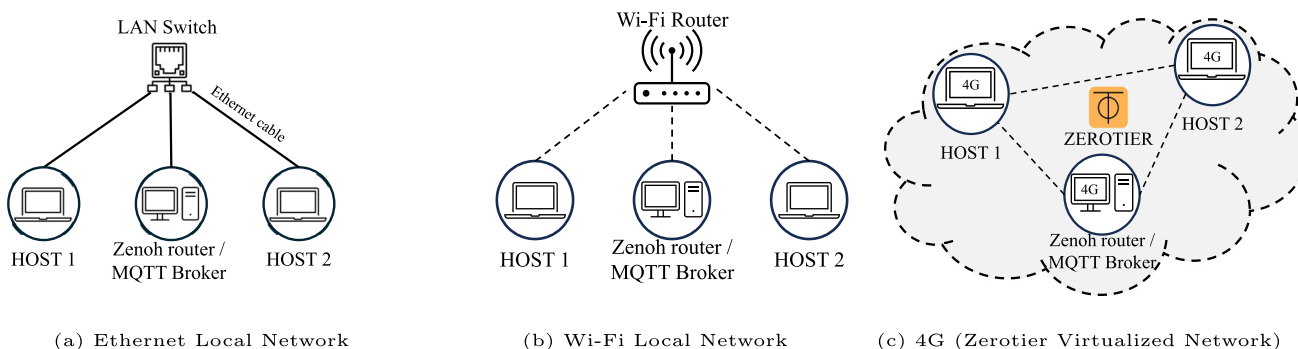


Fig. 5 Network setups. The router or broker is for Zenoh and MQTT; DDS does not need an individual broker

reliable QoS settings ensured that our data collection was robust, minimizing the impact of data loss or distortion.

4 Experiment Results

In this section, we present the results of our experiments comparing the performance of DDS, MQTT, and Zenoh in different network conditions and on an actual robot platform. In the Section of 4.1 Latency and 4.2. Throughput, we present five cases, including CycloneDDS, Zenoh-TCP, Zenoh-broker,

MQTT-no-broker, MQTT-broker. CycloneDDS is the default RMW implementation in ROS 2. In this configuration, the middleware is CycloneDDS alone. In the Zenoh-TCP configuration, TCP is used as the transport protocol for peer-to-peer communication. MQTT is a lightweight messaging protocol using a publish/subscribe model. In the MQTT-no-broker configuration, one host is set as the role of broker, and the communication occurs directly between two hosts without a centralized broker. In the Zenoh-broker or MQTT-broker configuration, a broker-based architecture is set to manage communication between hosts.

Table 2 Mean latency with different network setup

ROS Message	CycloneDDS	Zenoh-TCP	Zenoh-broker	MQTT-nobroker	MQTT-broker
(a) Mean latency (ms) with Ethernet					
Array1k	1.29	1.98	1.78	89.74	91.01
Array4k	1.30	1.97	1.91	86.59	3.87
Array16k	1.55	2.36	2.24	86.98	3.58
Array64k	3.35	3.35	3.27	3.51	14.77
Array256k	5.37	7.42	7.42	10.75	48.50
Array1m	19.28	32.88	42.69	136.60	5545.04
Array2m	37.97	70.97	70.61	312.89	7468.62
PointCloud512k	11.16	20.51	18.47	18.45	87.07
PointCloud1m	21.95	37.42	41.24	94.92	1020.70
PointCloud2m	39.94	76.78	77.12	540.25	5625.40
(b) Mean latency with Wi-Fi					
Array1k	51.40	14.82	12.58	92.93	173.62
Array4k	39.84	32.34	54.11	101.19	73.64
Array16k	56.73	48.30	139.63	101.77	125.69
Array64k	606.50	452.51	326.96	278.81	4382.25
Array256k	1574.14	3021.77	2323.91	13538.75	19788.57
Array1m	—	5960.16	7599.04	20950.00	—
Array2m	—	9386.79	11337.42	—	—
PointCloud512k	—	5164.90	5407.28	11307.95	—
PointCloud1m	3748.00	5326.69	5657.96	24115.00	—
PointCloud2m	—	9283.26	8805.96	—	—
(c) Mean latency with 4G					
Array1k	58.80	72.11	134.02	279.17	679.91
Array4k	103.75	102.86	98.05	228.84	537.87
Array16k	116.95	150.81	145.42	220.82	530.20
Array64k	163.29	3349.90	434.19	541.76	2222.11
Array256k	—	7458.61	7556.25	10736.73	19987.27
Array1m	—	10086.88	11811.52	—	—
Array2m	—	15365.29	10870.59	—	—
PointCloud512k	—	7357.50	6591.85	—	—
PointCloud1m	—	13254.00	10333.13	—	—
PointCloud2m	—	14309.50	11900.27	—	—

4.1 Latency

Table 2 presents the mean latency results obtained from the experiments conducted in different network setups, including Ethernet, Wi-Fi, and 4G. In each network setup, we measured the mean latency for different middleware implementations and various ROS message types. The results provide insights into the latency performance of each middleware under different network conditions and message sizes.

It is evident that the mean latency increases as the message size grows in all network scenarios. The results also demonstrate the impact of different network setups on the latency performance of networking middleware. Each middleware demonstrates superior performance under Ethernet conditions, which offer the highest bandwidth, compared to Wi-Fi and 4G environments. Notably, CycloneDDS performs best in Ethernet setup, while Zenoh performs best in Wi-Fi and 4G setups. This is caused by the DDS discovery mechanism. DDS leverages UDP multicast features to broadcast messages in the transport layer. It enables fast peer discovery and QoS-based low-latency data transmission in wired local networks where bandwidth and packet loss rate are promised. On the other hand, in wireless networks such as Wi-Fi or 4G, UDP multicast could cause the flooding effect, resulting in poorer performance [6, 7, 27]. These findings highlight the importance of considering network conditions and message characteristics when selecting an appropriate middleware for Edge-to-Edge and Edge-to-Cloud communication in ROS-based robotic systems.

4.2 Throughput

Figure 6 presents the throughput results. The results showed that the bandwidth of the network has the largest impact on throughput. Furthermore, as illustrated in Fig. 6(a), CycloneDDS exhibited comparable performance to Zenoh under Ethernet conditions; however, Zenoh outperformed CycloneDDS in Wi-Fi and 4G scenarios.

4.3 Performance on Actual Robot Platform

As mentioned in 3.1.1, we conducted an experiment on a real-robot platform, TurtleBot 4. The setup is shown in Fig. 4. The TurtleBot 4 and the laptop, connected to the same Wi-Fi network, enable the laptop to send velocity commands for the TurtleBot 4 to execute a square-shaped trajectory. At the same time, the OptiTrack motion capture system records the actual trajectory. Figure 7 shows the moving trajectory of the TurtleBot 4 over 96 s. Figure 8 shows the drift error between the last and the first circle.

Zenoh demonstrated the closest adherence to the square trajectory, exhibiting minimal drift and accurately reproducing the desired path. This result highlights Zenoh’s

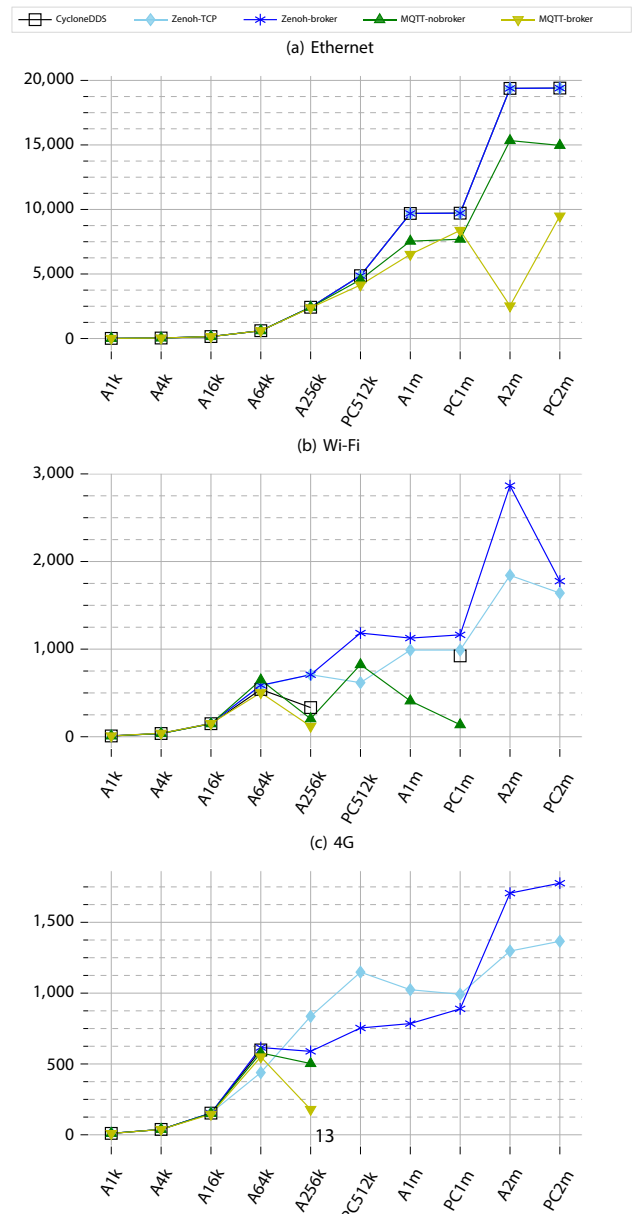


Fig. 6 Throughput (Kilobytes/sec) with different network setup. In Subfig (a), since the data difference is very small, the lines of CycloneDDS, Zenoh-TCP, and Zenoh-broker overlap

robustness and capability to maintain precise position and movement control on a real robot platform. On the other hand, the trajectories of CycloneDDS and MQTT showed varying degrees of drift over time.

5 Discussion

Throughout the paper, we have provided a comprehensive comparison of DDS, MQTT, and Zenoh by giving the experiment results and their analysis. The evaluation of latency

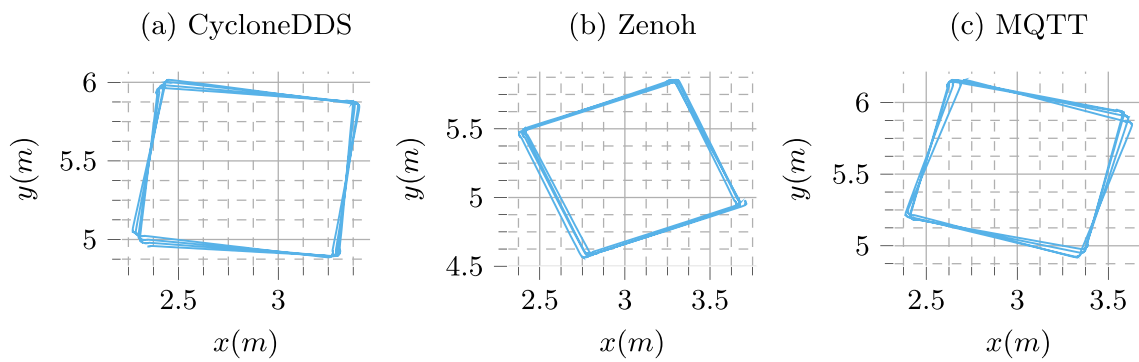


Fig. 7 Robot moving trajectory over 96 s of different middleware. The trajectories are for illustration and are not comparable

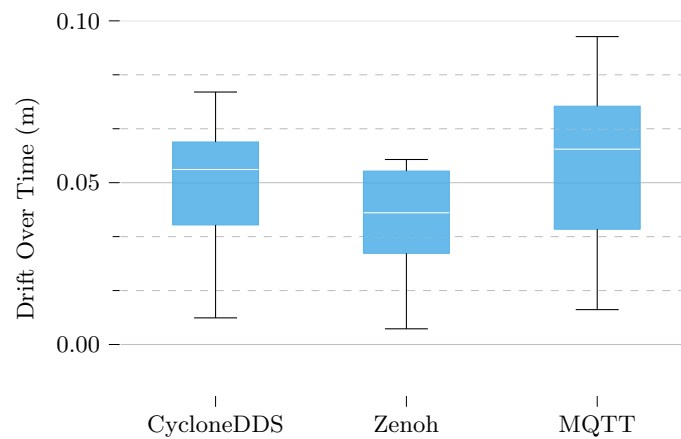


Fig. 8 Robot drift over 96 seconds for various middlewares, analyzed using data from 2200 frames per middleware

Table 3 CPU usage (%) with different network setup

ROS Message	CycloneDDS	Zenoh-TCP	Zenoh-broker	MQTT-nobroker	MQTT-broker
Array1k	(0.04, 0.04, 0.04)	(0.05, 0.05, 0.06)	(0.06, 0.05, 0.06)	(0.05, 0.05, 0.05)	(0.05, 0.05, 0.05)
Array4k	(0.04, 0.03, 0.04)	(0.05, 0.05, 0.06)	(0.06, 0.05, 0.06)	(0.05, 0.05, 0.05)	(0.05, 0.05, 0.05)
Array16k	(0.05, 0.04, 0.05)	(0.06, 0.06, 0.06)	(0.06, 0.06, 0.06)	(0.05, 0.05, 0.05)	(0.04, 0.05, 0.06)
Array64k	(0.07, 0.06, 0.06)	(0.06, 0.05, 0.05)	(0.07, 0.05, 0.06)	(0.05, 0.05, 0.06)	(0.05, 0.03, 0.06)
Array256k	(0.19, 0.09, 0.06)	(0.09, 0.05, 0.04)	(0.09, 0.05, 0.05)	(0.08, 0.05, 0.05)	(0.07, 0.04, 0.05)
Array1m	(0.51, 0.08, 0.11)	(0.16, 0.04, 0.03)	(0.18, 0.03, 0.04)	(0.18, 0.13, 0.13)	(0.17, 0.12, 0.14)
Array2m	(0.77, 0.08, 0.19)	(0.30, 0.04, 0.03)	(0.32, 0.04, 0.03)	(0.30, 0.17, 0.19)	(0.20, 0.18, 0.19)
PointCloud512k	(0.30, 0.07, 0.07)	(0.12, 0.06, 0.03)	(0.12, 0.04, 0.04)	(0.12, 0.08, 0.07)	(0.12, 0.09, 0.09)
PointCloud1m	(0.57, 0.09, 0.12)	(0.18, 0.04, 0.03)	(0.19, 0.04, 0.04)	(0.17, 0.12, 0.14)	(0.21, 0.15, 0.14)
PointCloud2m	(0.84, 0.11, 0.19)	(0.32, 0.03, 0.03)	(0.30, 0.03, 0.04)	(0.33, 0.18, 0.20)	(0.22, 0.22, 0.20)

The data is in the format of (Ethernet_CPU, Wi-Fi_CPU, 4G_CPU)

Table 4 Security difference

Security	Mean latency	Throughput	CPU usage
With security	77.35 ms	9701 bytes/sec	0.06 %
Without security	72.11 ms	9804 bytes/sec	0.06 %

and throughput within various network setups, including Ethernet, Wi-Fi, and 4G, yielded significant insights into the performance of different middlewares. Besides these insights, the ensuing subsections delve into two other dimensions worth noting—namely, “CPU Usage” and “Security”—to provide a multi-angle understanding of the middleware.

5.1 CPU usage

The CPU usage of Host 1 as shown in Table 3, acquired through the “performance_test” tool, provides valuable information about the computational efficiency of each middleware.

5.2 Security

Enabling ROS 2 security affects the performance metrics such as latency and throughput of the ROS 2 communication channel [11]. The use of Array1k messages to test latency, throughput, and CPU usage underlines the effects of enabling security features. As shown in Table 4, the results reveal that activating security measures causes a noticeable increase in mean latency, indicating the additional processing overhead introduced by security mechanisms. Furthermore, there is a decrease in throughput, which aligns with expectations due to the encryption and decryption processes associated with securing message communication. And CPU usage remains relatively unaffected by the security measures.

6 Conclusion

In this paper, we comprehensively evaluate three networking middlewares used in ROS 2 system. We compared the latency and throughput of CycloneDDS, Zenoh, and MQTT with ROS Messages under different network setups, including Ethernet, Wi-Fi, and 4G. Moreover, we tested the performance of these middlewares on a real robot platform, TurtleBot 4. We set a host and the TurtleBot 4 under the local Wi-Fi network, and use the host to control the TurtleBot 4 to run a square-shape trajectory for 96 s through different middleware. The trajectories were recorded by the Optitrack MOCAP system. Our experimental results reveal that under Ethernet, CycloneDDS has minimal latency and throughput, due to its UDP multicast mechanism. Zenoh exhibits superior

performance under Wi-Fi and 4G conditions. On the real robot platform, Zenoh achieves the smallest trajectory drift error. Our findings that Zenoh performs best are aligned with the survey results from ROS 2 community that Zenoh was the alternative most suggested by users, and current efforts by the ROS 2 TSC Meeting to include Zenoh in the next ROS 2 release.

Acknowledgements This research work is supported by the R3Swarms project funded by the Secure Systems Research Center (SSRC), Technology Innovation Institute (TII).

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Jiaqiang Zhang, Xianjia Yu and Sier Ha. The first draft of the manuscript was written by Jiaqiang Zhang and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Open Access funding provided by University of Turku (including Turku University Central Hospital). The authors have no relevant financial or non-financial interests to disclose.

Declarations

Ethical Approval No Ethics approval is needed.

Consent to Participate No Consent to participate is needed.

Consent for Publication No Consent for publication is needed.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Zhang, J., Keramat, F., Yu, X., Hernández, D.M., Queralt, J.P., Westerlund, T.: Distributed robotic systems in the edge-cloud continuum with ros 2: a review on novel architectures and technology readiness. In: 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEEC), pp. 1–8 (2022). IEEE
2. Pardo-Castellote, G.: Omg data-distribution service: Architectural overview. In: 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings., pp. 200–206 (2003). IEEE
3. osrf: ROS 2 Default RMW TSC reports. <https://osrf.github.io/TSC-RMW-Reports/>. [Online]

4. eProxima: eProxima Fast DDS performance. <https://www.eprosima.com/index.php/resources-all/performance/eprosima-fast-dds-performance>. [Online]
5. Kang, Z., An, K., Gokhale, A., Pazandak, P.: A comprehensive performance evaluation of different kubernetes cni plugins for edge-based and containerized publish/subscribe applications. In: 2021 IEEE International Conference on Cloud Engineering (IC2E), pp. 31–42 (2021). IEEE
6. Peeroo, K., Popov, P., Stankovic, V.: Exploring the effects of multicast communication on dds performance. [arXiv:2209.09001](https://arxiv.org/abs/2209.09001). (2022)
7. Liang, W.-Y., Yuan, Y., Lin, H.-J.: A performance study on the throughput and latency of Zenoh, Kafka, and DDS, MQTT (2023)
8. OASIS Message Queuing Telemetry Transport (MQTT) TC. <https://www.oasis-open.org/committees/mqtt/>. [Online]
9. Eclipse: Zenoh. <https://zenoh.io/>. [Online]
10. Mayoral-Vilches, V., White, R., Caiazza, G., Arguedas, M.: Sros2: usable cyber security tools for ros 2. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 11253–11259 (2022). IEEE
11. Kim, J., Smereka, J.M., Cheung, C., Nepal, S., Grobler, M.: Security and performance considerations in ROS 2: a balancing act (2018)
12. Maruyama, Y., Kato, S., Azumi, T.: Exploring the performance of ros2. In: Proceedings of the 13th International Conference on Embedded Software, pp. 1–10 (2016)
13. Yoshino, D., Watanobe, Y., Naruse, K.: A highly reliable communication system for internet of robotic things and implementation in rt-middleware with amqp communication interfaces. *IEEE Access*. **9**, 167229–167241 (2021)
14. Light, R.A.: Mosquitto: server and client implementation of the mqtt protocol. *J Open Source Softw* **2**(13), 265 (2017)
15. Apex.AI: performance_test. https://gitlab.com/ApexAI/performance_test
16. Bédard, C., Lütkebohle, I., Dagenais, M.: ros2_tracing: Multipurpose low-overhead framework for real-time tracing of ros 2. *IEEE Robot and Auto Lett* **7**(3), 6511–6518 (2022). <https://doi.org/10.1109/LRA.2022.3174346>
17. Pandey, P., Parasuraman, R.: Empirical analysis of bi-directional wi-fi network performance on mobile robots in indoor environments. In: 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), pp. 1–7 (2022). <https://doi.org/10.1109/VTC2022-Spring54318.2022.9860438>
18. Kuboichi, T., Hasegawa, A., Peng, B., Miura, K., Funaoka, K., Kato, S., Azumi, T.: CARET: Chain-aware ROS 2 evaluation tool. In: Proceedings of IEEE International Conference on Embedded and Ubiquitous Computing (EUC) (2022)
19. Kronauer, T., Pohlmann, J., Matthe, M., Smejkal, T., Fettweis, G.: Latency analysis of ROS2 multi-node systems (2021)
20. Li, Z., Hasegawa, A., Azumi, T.: Autoware perf: a tracing and performance analysis framework for ros 2 applications. *Journal of Systems Architecture*. **123**, 102341 (2022). <https://doi.org/10.1016/j.sysarc.2021.102341>
21. Dhas, Y.J., Jeyanthi, P.: A review on internet of things protocol and service oriented middleware. In: 2019 International Conference on Communication and Signal Processing (ICCSP), pp. 0104–0108 (2019). IEEE
22. Escobar, J.J.L., Gil-Castañeira, F., Redondo, R.P.D.: Decentralized serverless iot dataflow architecture for the cloud-to-edge continuum. In: 2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pp. 42–49 (2023). IEEE
23. Bédard, C., Lajoie, P.-Y., Beltrame, G., Dagenais, M.: Message flow analysis with complex causal links for distributed ros 2 systems. *Robot and Auto Syst* **161**, 104361 (2023) <https://doi.org/10.1016/j.robot.2022.104361>
24. Puck, L., Keller, P., Schnell, T., Plasberg, C., Tanev, A., Heppner, G., Roennau, A., Dillmann, R.: Performance evaluation of real-time ros2 robotic control in a time-synchronized distributed network. In: 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), pp. 1670–1676 (2021). IEEE
25. Fu, G., Zhang, Y., Yu, G.: A fair comparison of message queuing systems. *IEEE Access*. **9**, 421–432 (2020)
26. Xiong, M., Parsons, J., Edmondson, J., Nguyen, H., Schmidt, D.C.: Evaluating the performance of publish/subscribe platforms for information management in distributed real-time and embedded systems. omgwiki.org/dds. (2010)
27. Kydos: Minimizing discovery overhead in ROS2. <https://zenoh.io/blog/2021-03-23-discovery/#leveraging-resource-generalisation>. [Online]

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Jiaqiang Zhang is currently a Doctoral Researcher at the Turku Intelligent and Embedded Robotic System (TIERS) Lab, Department of Computing, University of Turku, Finland. He started pursuing his PhD in April 2022. Before that, he was a software developer in Shanghai, China. He got his Master of Science in 2020, from the Shanghai Institute of Technical Physics of the Chinese Academy of Sciences, Shanghai, China. Jiaqiang's research topics include robotics, edge computing, sensor fusion, and computer vision.

Xianjia Yu is currently a senior researcher affiliated with the Turku Intelligent and Embedded Robotic System (TIERS) Lab, Department of Computing at the University of Turku, Turku, Finland. Xianjia Yu has been working with robotics and AI since early 2018. He started his doctoral research in robotics in Jan 2021. Before that, he was a senior robotic algorithm engineer in Shanghai, China. Xianjia got his doctoral degree in robotics and autonomous systems at the University of Turku. He attained his Master of Science (Engineering) in Electronics Engineering from Fudan University, Shanghai China. Additionally, he obtained his Master of Science (Technology) degree in Information and Communication Technology from the University of Turku. Xianjia's research direction includes the fields of robotics, multi-modal sensing, machine learning, computer vision, sensor fusion, and federated learning.

Sier Ha is currently a doctoral researcher affiliated with the Turku Intelligent and Embedded Robotic System (TIERS) Lab, Department of Computing at the University of Turku, Turku, Finland. He has been working in the fields of robotics and AI since 2022 and began their doctoral research in 2023. He holds a Master of Science (Engineering) degree in Electronic Information from Fudan University in Shanghai, China, and a Master of Science (Technology) in Robotics and Autonomous Systems from the University of Turku. His research focuses on machine learning, computer vision, and sensor fusion.

Jorge Peña Queralta received B.Sc. degrees in mathematics and physics engineering from UPC BarcelonaTech, Spain, in 2016, a M.Sc. (Tech.) degree in ICT from the University of Turku, a M. Eng. degree in Electronics and Communication Engineering from Fudan University, China, in 2018, and a Ph.D. (Tech.) degree from the University of Turku in 2022. He was a postdoctoral researcher at the Swiss Federal School of Technology, ETH Zürich, Switzerland, from 2023 to 2024. His research interests include multi-robot systems, collaborative autonomy, distributed perception, and robot learning.

Tomi Westerlund is a professor in robotics and autonomous systems. He is the research group leader of the Turku Intelligent Embedded and Robotics Systems (TIERS) lab at the University of Turku (UTU), Finland. TIERS's main research interests are in multi-robot systems, autonomous robots and edge computing. The main application areas for TIERS include collaborative and heterogeneous multi-robot systems, localization and mapping of urban and unstructured environments, hardware accelerators for AI (edge AI) and distributed intelligence. In essence, to better understand how robots operate and can perform a variety of functions in different fields like environmental monitoring, delivery, agriculture, and search and rescue missions. Prof. Westerlund is a senior member of IEEE.