robotics

MDPI

Project Report

Performance Improvement of Multi-Robot Data Transmission in Aggregated Robot Processing Architecture with Caches and QoS Balancing Optimization

Abdul Jalil, Jun Kobayashi and Takeshi Saitoh *

Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology, Iizuka 820-8502, Japan; malla.abdul-jalil545@mail.kyutech.jp (A.J.); jkoba@ics.kyutech.ac.jp (J.K.)
* Correspondence: saitoh@ai.kyutech.ac.jp

Abstract: Robot Operating System 2 (ROS 2) is a robotic software that uses a set of Quality of Service (QoS) policies to manage the quality of robot data transmissions in a network, such as the RELIABLE and KEEP_LAST options. In ROS 2 node communication, the RELIABLE connection guarantees that all message data can be properly sent from the publisher to the subscriber. However, strict reliability is not guaranteed if the RELIABLE connection uses the KEEP_LAST option to transmit the robot data in the publish–subscribe communication. This study aims to analyze the efficiency of local cache, cache control, and QoS balancing optimization to improve ROS 2 node communication when using the RELIABLE and KEEP_LAST options to transmit multi-robot data in Aggregated Robot Processing (ARP) architecture. Our idea in local cache and cache control is to streamline the sensor data output before processing it when the sensor device produces the data with the same value in a row. Furthermore, QoS balancing optimization aims to balance the DEPTH and DEADLINE QoS configuration to determine the rates and buffer size in ROS 2 node communication. This study shows that combining local cache and QoS balancing optimization improves multi-robot data transmission and cooperation in ARP architecture.

Keywords: multi-robot; aggregated robot processing; caches; QoS; optimization; ROS 2

check for updates

Citation: Jalil, A.; Kobayashi, J.; Saitoh, T. Performance Improvement of Multi-Robot Data Transmission in Aggregated Robot Processing Architecture with Caches and QoS Balancing Optimization. *Robotics* **2023**, *12*, 87. https://doi.org/10.3390/robotics12030087

Academic Editor: Kagan Eugene

Received: 29 April 2023
Revised: 27 May 2023
Accepted: 12 June 2023
Published: 15 June 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Multi-Robot Systems (MRS) consist of several robots cooperating to handle complex tasks together. Each robot can exchange information data based on the node communication system in the network using Robot Operating System 2 (ROS 2) and uses a set of Quality of Service (QoS) policies to manage the quality of robot data transmissions, such as using a RELIABLE connection and KEEP_LAST option to transmit robot data and configure DEADLINE and DEPTH to determine the rate and buffer, respectively [1]. In ROS 2 node communication, strict reliability is not guaranteed if the RELIABLE connection uses the KEEP_LAST options to store the data sample in the publish–subscribe communication. It happens if the DEADLINE rates for transmitting the message data are not balanced with the buffer size configured in the DEPTH. If DEADLINE configures the rates with high frequency and DEPTH configures the buffer with a small size, some packets will be lost in ROS 2 node communication [2–5]. Otherwise, if the DEADLINE configures the rates with low frequency, the rates for data transmission from the publisher and subscriber will be low, affecting the real-time message data transmission between the publisher and subscriber.

Studies on improving robot data transmission using ROS 2 are interesting topics that some researchers have developed. Fernandez carried out the study on improving ROS 2 performance with different QoS and cyber security settings [2]. That study showed that a difference in QoS profiles and security settings could affect the latency and throughput of data transmission. Choi [6] designed the priority-driven chain-aware scheduling (PiCAS) implementation for ROS 2 callbacks, nodes, and executors. The researchers used PiCAS to

improve end-to-end message data transmission latency through a default ROS 2 scheduler. Wang [7] has developed an improvement in robot data transmission in ROS 2 using a partial serialization algorithm. In that study, the researchers used a partial serialization algorithm to analyze the efficiency of inter-process communication (IPC), called Toward Zero Copy (TZC). The use of a priority synthesis algorithm to improve the predictability of event chains in ROS 2 has been analyzed by Randolph [8]. In that study, a priority synthesis algorithm was used to improve the predictability of ROS 2 applications in response time, jitter, and missed deadlines. Furthermore, Jiang et al. [9] have implemented the Adaptive Two-Layer Serialization Algorithm (ATSA) to optimize message passing in ROS 2.

The contribution of this study is to analyze the performance of local cache [4], cache control [10], and QoS balancing optimization [5] to improve the quality of multi-robot communication in Aggregated Robot Processing (ARP) architecture when ROS 2 for robot data transmission uses the RELIABLE and KEEP_LAST options to transmit the robot data. ARP is an architecture in robotic systems that centralizes multi-robot data processes and communicates the MRS on a computer, called the Computer Environment Dedicated to Data Processing (CEDDP). In the ARP architecture, the robot computer and CEDDP can exchange message data through wireless networks.

2. Materials and Methods

2.1. Aggregated Robot Processing

The flow of robot data processes generally consists of three components: sensing, planning, and actuation [11]. These components can be connected as node communication systems in the network using ROS 2. ROS 2 is a robotic software built on top of the Data Distribution Service (DDS) [1]. In the ARP architecture, the sensing and actuation components run in the robot computer, and the planning component runs in CEDDP. Figure 1 shows the MRS data transmission in the ARP architecture based on the ROS 2 node communication mechanism.

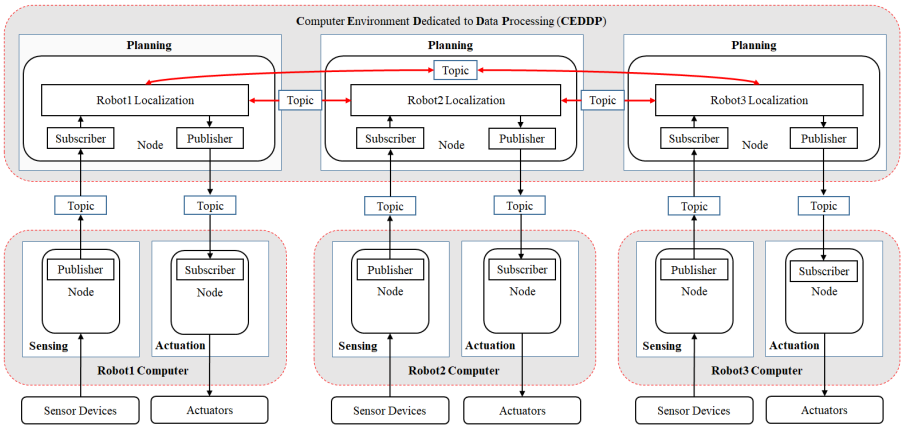

Figure 1. Aggregated robot processing architecture.

Based on ROS 2 node communication, the function of a node in the sensing component is to manage the sensor hardware, read the sensor output, and send the sensor data to a node in the planning component through a topic. After that, the node in the planning component will process the sensor data to determine the robot's action and localization and then send the result to a node in the actuation component to control the robot actuators. The topic is a ROS bus to transmit message data from the publisher to the subscriber.

2.2. Local Cache

The local cache function in this study is to streamline the transmission of sensor data from the sensing to the planning components when the sensor device produces output data with the same value in a row. Figure 2 shows the publish–subscriber mechanism in ROS 2, local cache work, and the local cache flowchart.

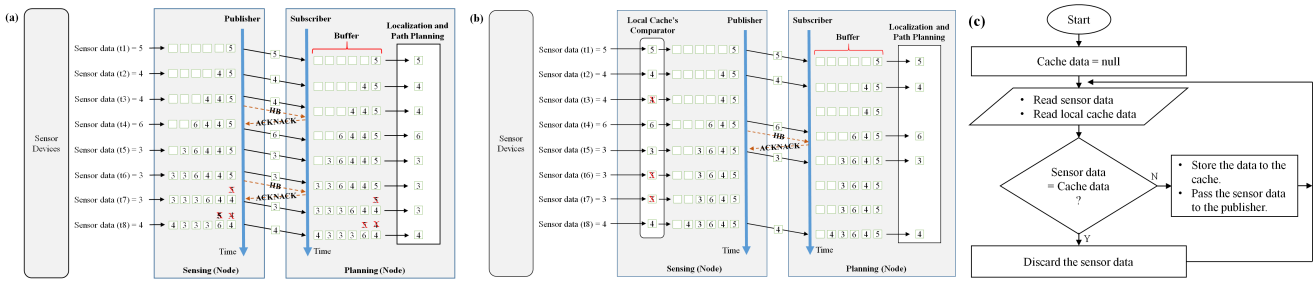

Figure 2. (**a**) Publish-subscribe mechanism in ROS 2; (**b**) local cache works in a sensing component to streamline the sensor data before publishing it to the subscriber; and (**c**) local cache flowchart.

Figure 2a shows the publish–subscribe mechanism in ROS 2 and illustrates how the buffer stores and discards data samples. In ROS 2 node communication, the RELIABLE connection uses the Heartbeat (HB) and Acknowledgment (ACKNACK) mechanism to transmit the data sample between the publisher and the subscriber [2]. However, if the publisher's buffer cannot accommodate storing the lost data sample in HB-ACKNACK transactions due to the buffer size, strict reliability is not guaranteed in a RELIBALE connection. Furthermore, Figure 2b shows the local cache mechanism to streamline sensor data before the publisher sends them to the subscriber in a sensing component. It can be seen that the local cache holds the sensor data transmission if the sensor device produces the data output with the same value in a row. Based on a RELIABLE connection mechanism, we propose the local cache method to improve MRS data transmission by reducing the HB and ACKNACK rates and helping the buffer not to store the data sample with the same value in a row in the buffer's queue. Furthermore, Figure 2c shows the local cache flowchart to streamline sensor data in a sensing component.

2.3. Cache Control

Figure 3a shows the cache control mechanism to streamline sensor data transmission in CEDDP before processing it in the planning component. In the ARP architecture, each robot in MRS cooperates based on communication of the planning component in CEDDP. Our idea in cache control is to improve the MRS communication performance in CEDDP by reducing the planning component work to receive the sensor data transmission sent from the sensing component. Furthermore, Figure 3b shows the cache control flow chart to streamline sensor data sent from the sensing component. To run the cache control in CEDDP, we used a node in CEDDP to subscribe to the sensor data sent from the sensing component and then compared it with the cache data values. If the sensor data are identical to the cache data, then the cache control will discard the sensor data. If they differ, the cache control will store the sensor data in the cache and read the next sensor data. This cache control runs repeatedly until the node is shut down.

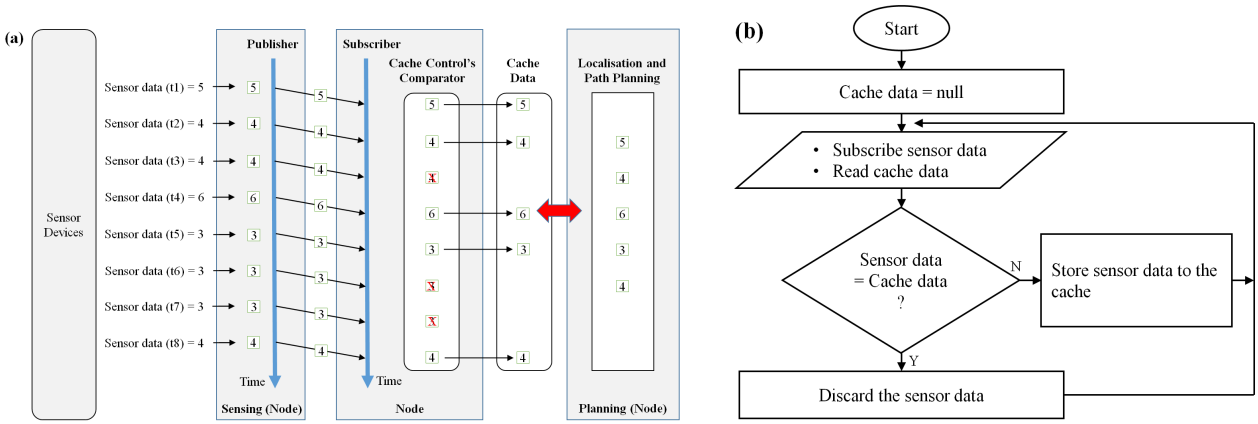

Figure 3. (**a**) Cache control works in the CEDDP.; and (**b**) cache control flowchart.

2.4. QoS Balancing Optimization

Figure 4a,b shows MRS communication in the ARP architecture and illustrate DEPTH and DEADLINE functions, respectively. DEPTH is a QoS policy in ROS 2 to configure the buffer size when the KEEP_LAST option is chosen to store the data sample. Furthermore, DEADLINE configures the rate.

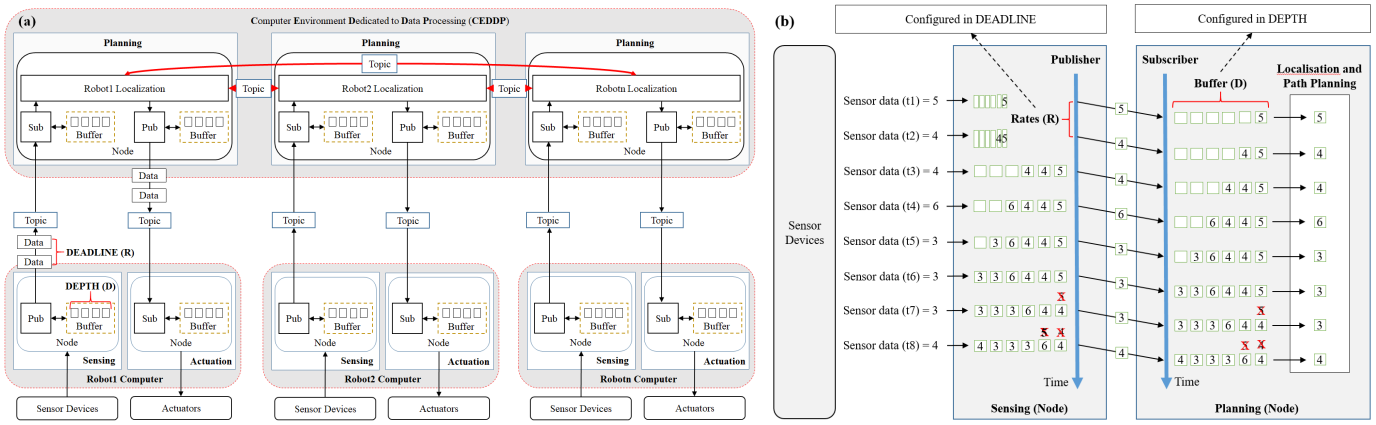

Figure 4. (**a**) QoS balancing optimization for MRS communication in ARP architecture; (**b**) DEPTH and DEADLINE QoS configuration illustration.

Based on the illustration shown in Figure 4a,b, to find the optimal value of DEADLINE and DEPTH, in the first step, we need to identify the total topic used to transmit the data sample from each agent.

$$T = \sum_{i=1}^{n} t_i \tag{1}$$

where T is the total topic of each agent, t is the topic used to send sensor data and receive the localization result, and n is the number of topics (1, 2, 3, . . . , n). In this study, t is a parameter that identifies the number of topics of each agent. The topic is a critical part of ROS 2 to transmit the data sample in the form of one-to-one, one-to-many, many-to-one, and many-to-many. After obtaining the total topic from each agent, calculate the average topic used to transmit the data sample in MRS.

$$T_{avg} = (\sum_{a=1}^{A} T_a)/A \tag{2}$$

where T_{avg} is the average of topics in MRS, and A is the number of agents (1, 2, 3, . . . , A). In Equation (2), T is a parameter used to identify the total topic implemented to transmit the data sample from each agent, and A is a parameter used to know the number of agents in MRS. The values of T and A can be changed based on the total topic used to transmit the data from each agent and the number of agents, respectively. In this optimization, our idea to find the optimal value of DEADLINE R is to divide the maximum data transmission rate $Rmax$ by the average topic used to transmit the data sample in MRS. $Rmax$ is the maximum rate when only one topic is used to transmit the data sample in MRS. Furthermore, we create our idea to determine the DEADLINE R with the equation:

$$R = \frac{Rmax}{T_{avg}} \tag{3}$$

In Equation (3), we divide the maximum data transmission rate $Rmax$ by the average topic T_{avg} to balance the rates with all topics in MRS. $Rmax$ and T_{avg} are the parameters in Equation (3) to put the maximum rate value to transmit the data sample and the average topic used in MRS, respectively. $Rmax$ value can be changed based on the necessary rates used to transmit the data in MRS, affected by the real-time data transmission between

the publisher and the subscriber. Then, the T_{avg} value changes based on the input of the parameters T and A in Equation (2). Based on the idea of Equation (3), we create the first constraint in this optimization to find the optimal value of DEADLINE R with

$$\frac{Rmax}{T_{avg}} - R \geq Rmin \tag{4}$$

In the first constraint, the optimal value of DEADLINE R should be greater than or equal to the minimum data transmission rate $Rmin$, which means that the transmission between the publisher and the subscriber is satisfied when the optimal value of DEADLINE is greater than or equal to the minimum data transmission rate. Next, find the optimal value of DEPTH D to determine the buffer size. Our idea here to find the optimal value of the DEPTH is to balance it with a DEADLINE tune. If the DEADLINE tune is large and close to the maximum rate, the DEPTH tune will also be large and close to the maximum buffer size $Dmax$. Otherwise, if the DEADLINE tune is low, the DEPTH tune will also be low and close to the minimum buffer size $Dmin$. $Dmax$ is the maximum buffer size to store the data sample in the publish–subscribe communication. Based on this idea, we create the second constraint to find the optimal value of DEPTH D with:

$$Dmax \times \frac{R}{Rmax} - D \geq Dmin \tag{5}$$

Next, for the following constraints, we bound the DEADLINE variable R to not be large and less than the maximum and minimum rates $Rmin \leq R \leq Rmax$ and bound the DEPTH variable D to not be large and less than the maximum and minimum buffer size $Dmin \leq D \leq Dmax$. For the objective function, we use the multi-objective optimization [12] to determine the maximum configuration of DEADLINE R and DEPTH D. Finally, we create the optimization of this study with the following:

$$\begin{aligned} \max \quad & R, D \\ \text{s.t.} \quad & \frac{Rmax}{T_{avg}} - R \geq Rmin \\ & Dmax \times \frac{R}{Rmax} - D \geq Dmin \\ & Rmin \leq R \leq Rmax \\ & Dmin \leq D \leq Dmax \end{aligned} \tag{6}$$

In this study, we maximize the DEADLINE R to make data transfer rates in MRS high and close to real-time data transfer. Then, maximize the buffer size in DEPTH D to adjust the change in the data transfer rate between the publisher and the subscriber. Furthermore, to implement the optimization, we used CVXPY to solve the problem, the open-source Python-embedded modeling language to solve the problem in convex optimization [13].

3. Results

3.1. Experimental Result in Actual Machine

This experiment analyzes the performance of MRS data transmission when the robot computer/machine sends various sensor data with local cache, cache control, and QoS balancing optimization in the ARP architecture. We did this experiment in a static environment with objects moving around the sensor devices to obtain various data values or conditions sent from the sensor devices to the robot machine. To perform the experiment, we used three Raspberry Pi 4 as robot machines in MRS with a Quad-Core Cortex A72 (ARM v8) processor @ 1.5 GHz and 8 GB of memory, respectively. Then, a laptop computer with an Intel Core i5 processor @ 2.60 GHz × 4 and 12 GB memory as a CEDDP. The OS installed on the MRS machine and CEDDP is a Linux Ubuntu 20.04 LTS, ROS 2 Foxy Fitzroy for robotic software, and Fast-RTPS DDS for ROS 2 node communication in the ARP network.

Figure 5 shows the experimental setup and the sensor devices connected to the machines. In this experiment, we used several sensor devices connected to each Raspberry Pi as an actual machine in MRS. Table 1 shows the sensor device and types, message data type, and data size, then Table 2 shows the QoS configurations in our experiment. Figure 6 illustrates this actual machine experiment. To analyze the performance of MRS data transmission, we measure the latency and calculate the total packet loss of the"Hello" message data transmitted in MRS communication. Figures 7 and 8 show the latency and packet loss analysis results, respectively. Based on the results of latency and packet loss, it can be seen that the combination of local cache and QoS balancing optimization effectively improves latency and reduces packet loss in MRS data transmission, compared to the situation without the implementation of optimization and combined with cache control.

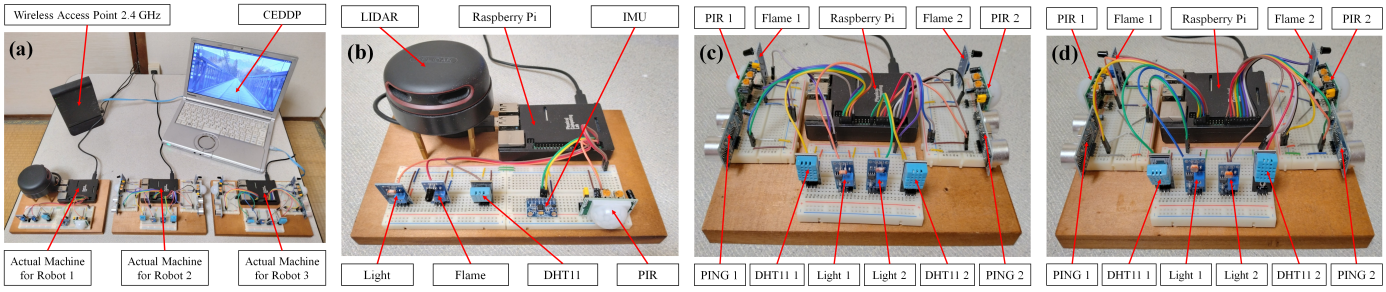

Figure 5. (**a**) Experimental setup on the actual machine; (**b**) sensor devices connected to the actual machine 1; (**c**) sensor devices connected to the actual machine 2; and (**d**) sensor devices connected to the actual machine 3.

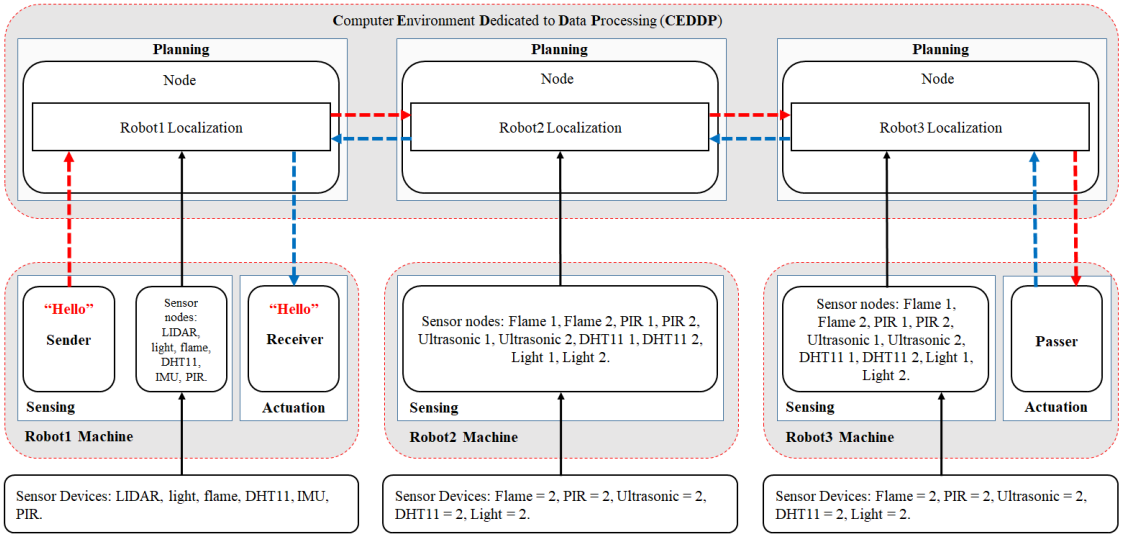

Figure 6. Experimental illustration in actual machine.

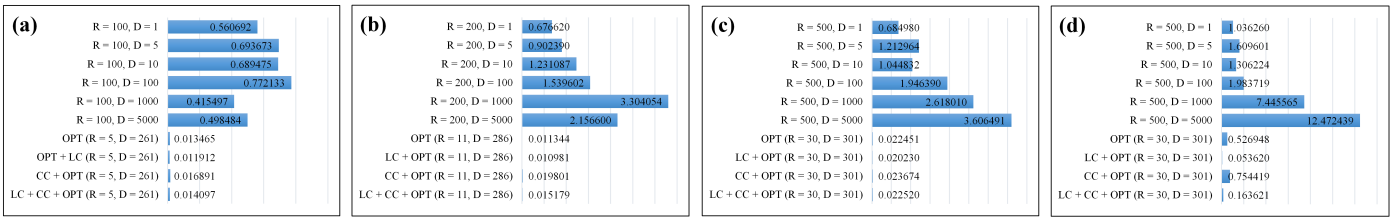

Figure 7. (**a**) Latency when the maximum data transmission rate $Rmax$ = 100 Hz; (**b**) latency when $Rmax$ = 200 Hz; (**c**) latency when $Rmax$ = 500 Hz; and (**d**) latency when $Rmax$ = 1000 Hz.

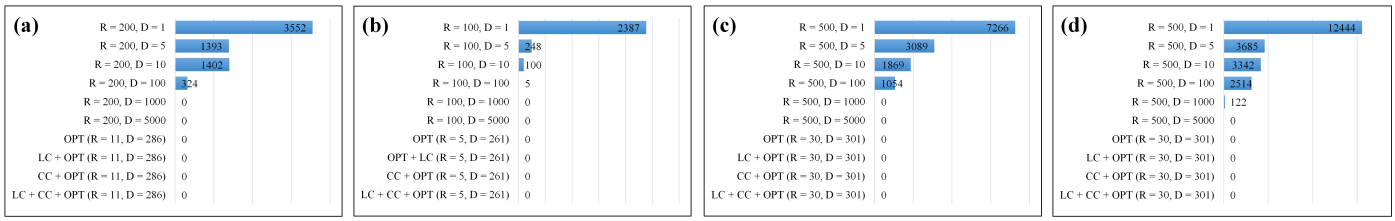

Figure 8. (**a**) Packet loss when the maximum data transmission rate *Rmax* = 100 Hz; (**b**) packet loss when *Rmax* = 200 Hz; (**c**) packet loss when *Rmax* = 500 Hz; and (**d**) packet loss when *Rmax* = 1000 Hz.

Table 1. Sensor device and types, message data type, and data size.

Sensor Devices	Sensor Types	Message Types	Data Size (Bytes)
LIDAR	SLAMTEC, A2M8	Float	24
Flame	AYNEF, flame module	Boolean	28
DHT11	HiLetgo, DHT11	Float	24
IMU	KKHMF, MPU-6050	Float	24
Ultrasonic	ELEGOO, HC-SR04	Float	24
PIR	VKLSVAN, HC-SR501	String	59
Light	VKLSVAN, photosensitive	String	54

Table 2. Configuration of QoS policies in the experiment.

QoS Policies	Options
RELIABILITY	RELIABLE
HISTORY	KEEP_LAST
DEPTH	1, 5, 10, 100, 1000, 5000, Opt (D)
DEADLINE	100 Hz, 200 Hz, 500 Hz, 1000 Hz, Opt (R)
DURABILITY	VOLATILE
LIVELINESS	AUTOMATIC

3.2. Experimental Result in Simulation

This experiment was carried out to analyze the performance of MRS cooperation in simulation. To implement the experiment, we used three Raspberry Pi 4 as robot computers to perform the robot simulation, CEDDP to process MRS data and communicate MRS, and access point 2.4 GHz to transmit robot data between the Raspberry Pi and CEDDP. The OS installed on the Raspberry Pi and CEDDP is Linux Ubuntu 20.04 LTS, ROS 2 Foxy Fitzroy for robotic software, and the Gazebo application for robot simulation. Based on the experimental setup shown in Figure 9a, three Raspberry Pis were used to run the robot simulation using the Gazebo application and CEDDP to process the LIDAR data for robot localization and communicate MRS. Furthermore, referring to the MRS design shown in Figure 9b, we used three mobile robots with the same design and specifications, and each robot was equipped with a LIDAR sensor to perceive the environment.

To analyze the performance of MRS cooperation, we have designed the MRS task based on the robot path shown in Figure 10a. The task of MRS in this simulation is to move the robots in parallel with constant velocity in different areas based on the movement of robot1 as the leader in MRS. Figure 10b shows the experimental illustration in the MRS simulation. Each robot in the simulation sends eight LIDAR sensor data to CEDDP for robot localization through eight nodes, respectively. LIDAR data sent from each robot to CEDDP for robot localization are 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°. The data type used to transmit the LIDAR data from the sensing to the planning components was float data type with a capacity of 24 bytes, respectively. For the QoS configuration, we analyze the performance of MRS cooperation when the QoS policy for ROS 2 node communication was configured based on the QoS configurations shown in Table 2. Furthermore, Figure 9 shows the experimental setup in simulation and robot design, Figure 10 presents the MRS

tasks and the illustration of MRS communication in the experiment, Figure 11 shows the results, then Figures 12 and 13 shows the success and failure of MRS cooperation in the simulation, respectively.

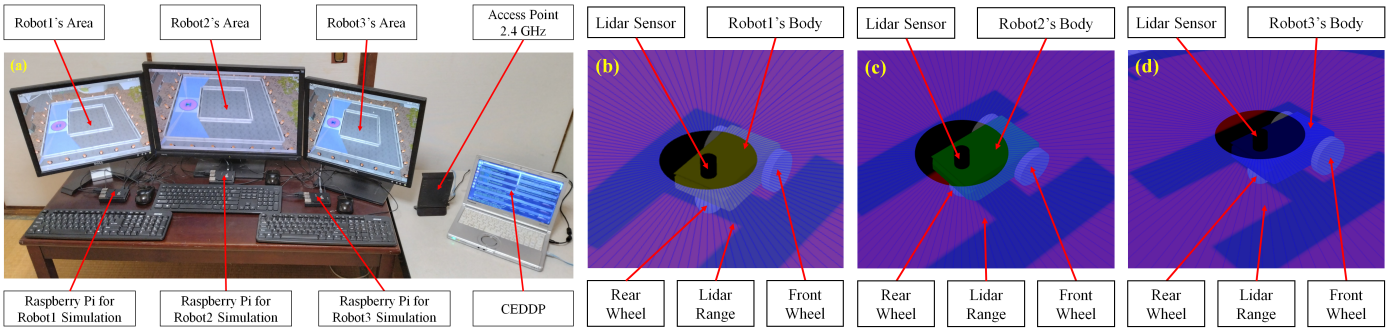

Figure 9. (**a**) Experimental setup; (**b**) robot1 design; (**c**) robot2 design; and (**d**) robot3 design.

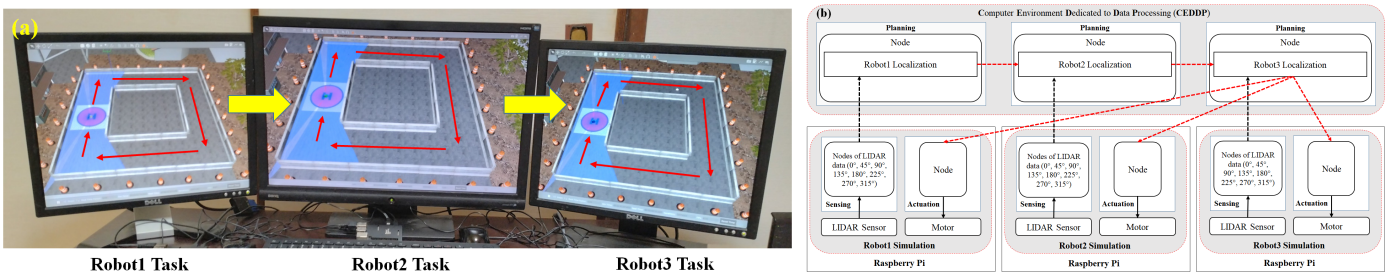

Figure 10. (**a**) Multi-robot tasks in the experiment; and (**b**) experimental illustration in MRS simulation.

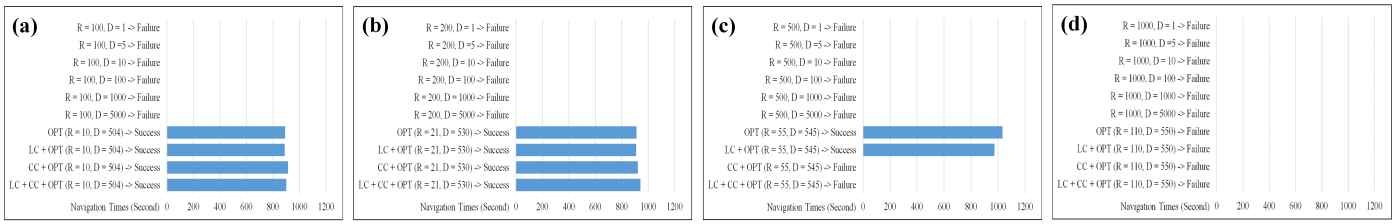

Figure 11. (**a**) Simulation result when *Rmax* = 100 Hz; (**b**) simulation result when *Rmax* = 200 Hz; (**c**) simulation result when *Rmax* = 500 Hz; and (**d**) simulation result when *Rmax* = 1000 Hz.

Figure 12. (**a**) MRS ready to navigate the hallway; (**b**) MRS turned right in the hallway; (**c**) MRS successfully navigated the hallway; and (**d**) MRS successfully returned home.

Figure 13. (**a**) MRS ready to navigate the hallway; (**b**) MRS turned right in the hallway; (**c**) MRS failed to navigate the hallway; and (**d**) MRS failed to complete the task.

4. Discussion and Conclusions

We have analyzed the performance of MRS communication in the ARP architecture when the robot data are transmitted without caches and optimization, with local cache, with cache control, with optimization, and combined between local cache, cache control, and optimization. Based on the results of the actual machine experiment shown in Figures 7 and 8, it can be seen that the combination of local cache and QoS balancing optimization effectively improves latency and reduces packet loss in MRS data transmission. Furthermore, Figure 8 confirms that unbalanced rates and buffer size in RELIABLE connections can affect packet loss in data communication between components. According to the simulation results shown in Figure 11, it can be seen that data transmission rates have a significant impact on the ability of MRS to complete tasks. This happens because the robot computer resource used in our experiment to run the simulation has limited capabilities, affecting real-time MRS cooperation in the ARP architecture. However, the simulation result shown in Figures 12 and 13 confirms that the combination of local cache and QoS balancing optimization can be relied on to improve MRS cooperation in the ARP architecture. In our simulation experiment, real-time multi-robot cooperation highly depends on the robot computer's resource capacity and the wireless network's reliability to exchange data between the robot computer and CEDDP.

Based on the analysis results shown in the actual machine and simulation experiment, it can be concluded that the combination of local cache and QoS balancing optimization effectively improves latency, reduces packet loss in a RELIABLE connection, and improves MRS cooperation in ARP architecture. In this study, the local cache effectively enhances ROS 2 performance because it can reduce the HB and ACKNACK rates when the node in the sensing component transmits the sensor data to the planning component in CEDDP, compared to the cache control that streamlines the sensor data in CEDDP and the QoS balancing optimization that transferred the sensor data with common rules. However, our proposed study is effective depending on the task, computer performance, wireless communication speed, number of sensors, and sensor types used in our experiment. In the future, we will analyze the efficiency of the local cache, cache control, and QoS balancing optimization when implemented in low-level hardware controllers, such as microcontrollers, and analyze it when implemented in real multi-robot systems with different computing performances, various data types and sizes, network types, and Internet of Things technology.

Author Contributions: Conceptualization, A.J. and J.K.; methodology, A.J. and J.K.; software, A.J.; validation, T.S.; formal analysis, A.J.; investigation, A.J.; resources, J.K. and T.S.; data curation, A.J.; writing—original draft preparation, A.J.; writing—review and editing, A.J. and T.S.; visualization, A.J. and T.S.; supervision, J.K. and T.S.; project administration, A.J.; funding acquisition, T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

Nomenclature

ROS 2	Robot Operating System 2
QoS	Quality of Service
ARP	Aggregated Robot Processing
RELIABLE	RELIABILITY option to guarantee of sending data sample without fault
MRS	Multi-Robot Systems
KEEP_LAST	The buffer size to store the data samples configured in DEPTH
DEPTH	QoS policy to determine the buffer size in KEEP_LAST option
DEADLINE	The rates of data transmission between publisher and subscriber
DDS	Data Distribution Service
CEDDP	Computer Environment Dedicated to Data Processing

ACKNACK	Acknowledgment
HB	Heartbeat
CVXPY	Python-embedded modeling language for convex optimization problems

References

1. Maruyama, Y.; Kato, S.; Azumi, T. Exploring the Performance of ROS2. In Proceedings of the 2016 International Conference on Embedded Software (EMSOFT), Pittsburgh, PA, USA, 2–7 October 2016. [CrossRef]
2. Fernandez, J.; Allen, B.; Thulasiraman, P.; Bingham, B. Performance Study of the Robot Operating System 2 with QoS and Cyber Security Settings. In Proceedings of the 2020 IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 24 August–20 September 2020. [CrossRef]
3. Chen, Z. Performance Analysis of ROS 2 Networks Using Variable Quality of Service and Security Constraints for Autonomous Systems. Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, September 2019.
4. Jalil, A.; Kobayashi, J. Efficacy of Local Cache for Performance Improvement of Reliable Data Transmission in Aggregated Robot Processing Architecture. In Proceedings of the 2022 22nd International Conference on Control, Automation and Systems (ICCAS), Jeju, Republic of Korea, 27 November–1 December 2022. [CrossRef]
5. Jalil, A.; Kobayashi, J.; Saitoh, T. Optimization Algorithm for Balancing QoS Configuration in Aggregated Robot Processing Architecture. In Proceedings of the 2023 International Conference on Artificial Life and Robotics (ICAROB2023), Oita, Japan, 9–12 February 2023.
6. Choi, H.; Xiang, Y.; Kim, H. PiCAS: New Design of Priority-Driven Chain-Aware Scheduling for ROS2. In Proceedings of the 2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS), Nashville, TN, USA, 18–21 May 2021. [CrossRef]
7. Wang, Y.P.; Tan, W.; Hu, X.Q.; Manocha, D.; Hu, S.M. TZC: Efficient Inter-Process Communication for Robotics Middleware with Partial Serialization. *arXiv* **2020**, arXiv:1810.00556. [CrossRef]
8. Randolph, C. Improving the Predictability of Event Chains in ROS 2. Master's Thesis, Delft University of Technology, Delft, The Netherlands, 15 March 2021.
9. Jiang, Z.; Gong, Y.; Zhai, J.; Wang, Y.P.; Liu, W.; Wu, H.; Jin, J. Message Passing Optimization in Robot Operating System. *Int. J. Parallel Program.* **2020**, *48*, 119–136. [CrossRef]
10. Jalil, A.; Kobayashi, J. Experimental Analyses of an Efficient Aggregated Robot Processing with Cache-Control for Multi-Robot System. In Proceedings of the 2020 20th International Conference on Control, Automation and Systems (ICCAS), Busan, Republic of Korea, 13–16 October 2020. [CrossRef]
11. Staschulat, J.; Lütkebohle, I.; Lange, R. The rclc Executor: Domain-specific deterministic scheduling mechanisms for ROS applications on microcontrollers: Work-in-progress. In Proceedings of the 2020 International Conference on Embedded Software (EMSOFT), Shanghai, China, 20–25 September 2020. [CrossRef]
12. Gunantara, N. A review of multi-objective optimization: Methods and its applications. *Cogent Eng.* **2018**, *5*, 1502242. [CrossRef]
13. Diamond, S.; Boyd, S. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *J. Mach. Learn. Res.* **2016**, *17*, 2909–2913.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.