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

Lukas Johannes Dust
Mälardalen University
Västerås, Sweden
lukas.dust@mdu.se

Emil Persson
Mälardalen University
Västerås, Sweden
emil.persson@mdu.se

Mikael Ekström[1]
Mälardalen University
Västerås, Sweden
mikael.ekström@mdu.se

Saad Mubeen[1]
Mälardalen University
Västerås, Sweden
saad.mubeen@mdu.se

Emmanuel Dean
Chalmers University of Technology
Göteborg, Sweden
deane@chalmers.se

***Abstract*—Multi-agent robot systems, specifically mobile robots in dynamic environments interacting with humans, e.g., assisting in production environments, have seen an increased interest over the past years. To better understand the ROS2 communication in a network with a high load of nodes, this paper investigates the communication handling of multiple robots to a single tracking node for centralized multi-agent robot systems using ROS2. Thereore, a quantitative analysis of two publisher-subscriber communication architectures and a comparative study between DDS vendors (CycloneDDS, FastDDS and GurumDDS) using ROS2 Galactic is performed. The architectures of consideration are a many-to-one approach, where multiple robots communicate to a central node over one topic, and the one-to-one communication approach, where multiple robots communicate over particular topics to a central node. Throughout this work, the increase in the number of robots at different publishing rates is simulated on a single computer for the different DDS vendors. A further simulation is done using a distributed setup with CycloneDDS. The simulations show that with an increase in the number of nodes, the average data age and the data miss ratio in the one-to-one approach were significantly lower than in the many-to-one approach. CycloneDDS was shown as the most robust regarding crashes and response time under system launch, while FastDDS showed better results regarding the data ageing.**

***Index Terms*—Multi-agent robot system, ROS2, Scalability, Autonomous Transport Robots**

I. Introduction

The number of robots acting in environments populated by humans has increased significantly in recent years and is expected to grow in the foreseeable future. This increase is enhanced by the transition from the overall goal of creating robots with highly specialised tasks operating in a controlled and defined environment to the goal of building robots able to execute a variety of tasks while operating in a dynamic environment or even collaborating with humans [1]. Original Equipment Manufacturers (OEMs) like Volvo GTO have followed the trend of developing and researching the creation of a multi-agent robot system to assist in transporting material in manufacturing plants using robot operating system 2 (ROS2) [2]. A centralized system has been created, where the robots are controlled and tracked using central computers, keeping each robot's computing power and intelligence limited to a minimum. ROS2 provides a framework for the distribution and communication between the different algorithms. However, the chosen centralized processing also relies on each robot's information, which contains the robot state data, e.g., battery state, odometry data, and load status. A sequence for robots periodically sending status information to a centralized track unit which forwards the collected information to further processing can be found in Figure 1. In data processing, the data age

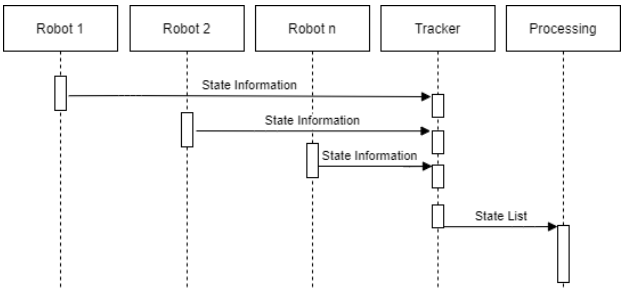


Fig. 1: Periodic communication of state information of robots towards a centralized tracking unit and further processing

[3] also plays a central role. For the communication interface between the robots and the tracking unit, different architectures can be built using ROS2. In order to allow dynamic scalability and investigate the capabilities of ROS2, this paper aims to study the two mentioned communication architectures and the use of different available DDS vendors.

A. Paper contributions

In the context of the communication handling of multiple robots to a single tracking node for centralized multi-agent robot systems, the one-to-one and multi-to-one communication architectures are presented and investigated in simulation. The performance in terms of data age and update misses of the two communication architectures are evaluated by simulation on a single computer as well as a distributed setup. Furthermore, the performance of the Data Distribution Service (DDS) vendors CycloneDDS, FastDDS and GurumDDS is evaluated on a single computer. The industrial use case, considered in this paper, is based on a legacy system built with a centralized architecture. In the simulation, we propose a dynamic connection handling approach utilizing ROS2 publisher-subscriber and service-client communication. The main contributions in the paper are summarized as follows:

- we evaluate, in simulation, the performance of a multi-to-one, and one-to-one communication architecture in terms of

[1]IEEE senior member

data age and update misses for an increase in the number of connected robots and different data publishing rates;
- we compare ROS2 Galactic using the DDS vendors CycloneDDS, FastDDS and GurumDDS in the context of data age, update misses and CPU utilization.

B. Organization of the paper

The remainder of the paper is organized as follows. Section II provides a review of the related works. Section III introduces the two investigated communication architectures. Section IV presents the system under investigation as well as the experiment results, evaluation and analysis. Finally, Section V concludes the paper with final remarks.

II. Related work

The usage of ROS2 in multi-agent robot systems is relatively new, and there have only been a few actual physical implementations of holistic systems. However, the use and performance of ROS2 in this context have seen a rising interest.

A. Multi-agent robot systems and ROS2 framework

First, there were first works conducted on the definition of multi-agent robot systems. Dudek et al. [4] defined primary taxonomy and classification of design choices and communication approaches for such systems. In the context of ROS2, there have been created frameworks and toolboxes with architectures for the development and deployment of multi-agent systems by Dehnavi et al. [5], Testa et al. [6] as well as McCord et al. [7]. Different system architectures for multi-agent systems using ROS are proposed by Vorapojpisut et al. [8] for the augmented environment as well as by Conte et al. [9] in the context of autonomous surface vehicles and Noh, and Park [10] in a manufacturing system. Barcis et al. [11] are delivering a proof of concept for robot collaboration and interaction in ROS2. Erös et al. [12] are investigating architectures for communication in ROS2 networks for control in automation systems. They are proposing different methods for many-to-one connection, but neither a performance evaluation nor the context of multi-agent robot systems is given. Our conducted research is based on a designed system with a given model, where a centralized system architecture is chosen. Furthermore, none of the found papers addresses the issue of dynamic connection handling in the context of ROS2 systems

B. ROS2 execution and real-time

Since the release of ROS2, there has been quite extensive interest and research towards the analysis of ROS2 execution and performance, which is significant for evaluating the two proposed communication architectures. Response time analysis for system architectures using a ROS network is proposed by Blaß, and Casini [13] as well as Tang et al. [14]. Li, Hasegawa, and Azumi [15] are proposing a performance analysis framework for ROS2 measuring callback execution time. Casini et al. [16] are describing the execution model for ROS2 nodes. By analyzing the ROS2 source code, the behaviour of the execution of nodes is defined and described. It is stated, that a node takes the latest available data from the DDS from all different subscribers at once and executes all relating callback before new data is taken. The execution of the callbacks is prioritized, while timer callbacks have the highest priority and service callbacks the lowest priority leading subscriber callbacks having the mean priority. Furthermore, a theoretical analysis of the task execution and processing of information for ROS2 processing chains has been proposed. Theoretical analysis for investigating and improving existing systems is desirable. However, no holistic method is proposed to analyze a distributed multi-agent robotic system. Towards execution of task under real-time constraints in a ROS network, research in exploring the capabilities of ROS2 has been conducted by Yang and Azumi [17] as well as Maruyama et al. [18]. Park, Delgado, and Choi [19] and Blass [20] are continuing those investigation in the real-time capabilities of ROS and performing studies in the context of multi-agent robot systems. Furthermore, Puck et al. [21] are proposing ROS2 real-time control architecture in time-synchronized networks and investigating real-time capabilities. Distribution and synchronization in computation for multi-agent robot systems towards real-time control of robots are addressed by the work of Puck and Keller [22]. Real-time communication inside ROS2 networks offers further potential in improving the end-to-end delay of the system. Gutiérrez [23] has researched communication for real-time robotics. These capabilities for real-time underline the goal of fast and reliable working systems. Our research aims not to implement real-time features. However, those approaches might be used for future work to consider improvements for the execution of the communication and the connection handling.

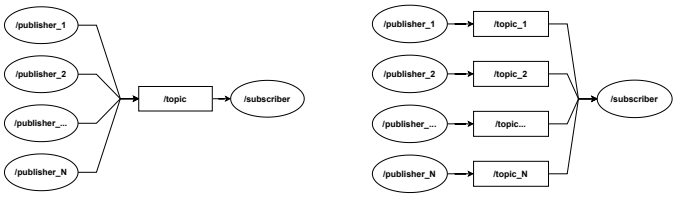
C. DDS and communication in ROS2

Another vital part of investigating ROS2 is the communication and the DDS of ROS2 systems. Kronauer et al. [24] are researching end to end latency for distributed ROS2 systems using standard QoS settings and different DDS. Guidelines are proposed for distributed ROS2 architectures to reduce the latency overhead. Maruyama, Kato, and Azumi [25] are exploring the potential and constraints of DDS and ROS2. Limitations regarded are the overhead of transforming data to DDS messages. Chen [26] is investigating network performance of ROS2 systems and investigates QoS and security constraints. Investigations are provided until a network of up to five nodes. Several theoretical approaches for analyzing the behaviour of execution and processing chains in ROS2 are provided in the conducted research. Those approaches are significantly restricted in their applicability, as simplifications and requirements are made for creating those theoretical analysis approaches. Our work aims to expand the evaluations on publisher-subscriber communication, and setting the context to multi-agent-robot systems. The related work can be used as a good framework, guiding to the research in this paper. To the best of our knowledge, an experimental evaluation regarding a high number of connected nodes under the aspect of data age and update misses has not been conducted and is therefore missing for an evaluation of ROS2 in the stated context.

III. Communication architectures

In ROS2 a subscriber-publisher communication between nodes (a process that performs computation) can be used, that is channeled through topics. One-to-one communication, where each robot publishes its data on a separate topic, and a many-to-one architecture, where all robots publish their data on the same topic, are the most common architectures and evaluated in this paper. A many-to-one, see Figure 2a, and a one-to-one topic configuration, see Figure 2b. The ROS2 nodes are

marked as ellipses in the figures and the specific topics as boxes. Throughout this paper, the many-to-one (M2O) and one-to-one architectures (O2O) are defined as follows. In M2O, several nodes communicate to a single node by utilizing only one topic, where all are publishing their messages. In contrast, if each publisher has its topic, it is considered O2O. In M2O, one subscriber reads from one topic, in which several nodes publish. In contrast, in O2O, the subscriber reads from several topics published by each individual node. Many different types of hybrid configurations could be used, for example, grouping publishers to specific topics, but this configuration is out of the scope of this paper. A significant concern with the M2O configuration is that the subscriber cannot discriminate the information between publishers. A case of high network utilization leads to message losses as other nodes overwrite messages before they are read. A one-to-one architecture may improve the separation between subscribers. In the case of overwriting of data, updated data of a robot is still available. However, this architecture could produce undesired side effects, such as architecture complexity and processing overhead.



(a) Communication architecture Many-To-One (M2O)

(b) Communication architecture One-To-One (O2O)

Fig. 2: Architectures under consideration.

IV. Experiments and evaluation

An abstracted version of a multi-agent robot system is simulated, focusing on the central problem under investigation to test the performance of the two communication architectures. The experimental settings and the results are presented in this section.

A. Simulated system

A ROS2 node system is created based on a legacy system. An overview of the ROS2 system is given in Figure 3, where the ellipsis represents the participating nodes, and the boxes represent the topics and service connections between the nodes. The system consists of robot nodes, one central tracking node, one visualisation node, and one for conducting measurements. In the legacy system, this tracker node is responsible for receiving the state information sent periodically from the robots and creating a list containing the status of all the robots. To allow dynamic connection handling a process that allows new robots to signal the required participation or disconnection in the network needs to be defined. ROS2 service-client communication as synchronous and directed communication for the request is a good base for implementing such a procedure. A service can be created inside the tracker, which can be called by the robots individually, sending their base information with the request. As the communication is synchronous and directed, addressing the acknowledgement to the right robot is guaranteed. That ensures that a robot firstly participates when the tracker has processed the request. The implemented approach follows three main steps:

1. The robot is launched and requires to connect to the network. The client inside the robot is calling the connection service in the tracker.
2. The service inside the tracker creates the required publishers and executes the required id handling. After the creation, an acknowledgement is sent back to the robots.
3. The robot receives the acknowledgement and creates the publisher and timer for publishing the state information periodically.

In the system, the selection between the two communication architectures can be made with parameters at system startup. The architectures are marked in the figure with dotted boxes representing the state topic in the O2O and the solid box for the O2M architecture. The execution of the nodes in the network is as follows. The robot nodes are executed periodically, creating and sending their state information received and collected by the tracker node. The tracker node is then periodically publishing a state list, which is used for visualization in RVIZ and received in a node, which in the legacy system would do further processing, for example, fleet control using the robots' state. In our case, the node is used to take measurements. Throughout the execution of the nodes, each state information will get four different timestamps, marked T1-T4 in Figure 3. T1 is taken when the state information is created for each robot. T2 is taken once the state information is received from the tracker. T3 is taken when the state list is created and transmitted, while T4 is taken when the state list is received in the Measurement Receiver node. A sequence diagram for periodic information update of connected robots can be foun in Figure 1.

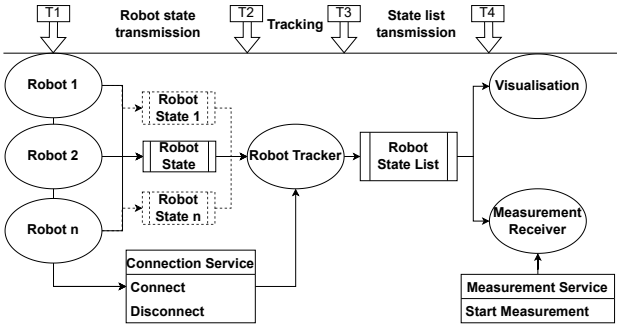


Fig. 3: Designed experiment system. The ROS2 nodes are marked as ellipses and the specific topics and services as boxes. The dashed and solid boxes for the robot state topic are representing M2O and O2O communication. T1-T4 are presenting timestamps taken in the system.

B. Experiment design

The created system is used to perform experiments to test and evaluate a ROS2 system using the connection handling and the two different architectures and DDS vendors. The performance of the two communication architectures is evaluated using the ROS2 distribution Galactic and the DDS vendors CycloneDDS, FastDDS and GurumDDS. The publishing periods for the robot and tracker nodes are set to be 1 ms and 2 ms, which are desired values in a legacy system. At the same time, the system consists of 10, 30, 50, 70 and 90 robots. The

system will be launched on a single computer with the two architectures for each possible parameter combination using the PC1 from Table I. Each measurement is configured to take 500 samples of the state list, storing all the timestamps of 500 state list updates at the end of the processing chain. Furthermore, the CPU utilization is measured as well. The timestamps are used to calculate the average data age and the amount of update misses of data, meaning that a robot was not updating its state data from one period to another.
In the second step of the evaluation, the default configuration of ROS2 Galactic and CycloneDDS is used in a distributed setup. The computers PC2 and PC3 are used to execute the robots. The tracker is executed on PC1, while the measurement node is executed on PC4. The clocks of the computers are synchronized using chrony with the tracker computer configured as the NTP server and the other three computers as the NTP clients. The measurements will be taken for 1 ms update rate and 10 to 150 robots in steps of 20 robots. The evaluation and data processing results are presented in the following subsection.

TABLE I: Hardware setup for experiments part 1

	PC1	PC2, PC3, PC4
OS	Ubuntu 20.04.4 LTS x86_64	Ubuntu 20.04.4 LTS x86_64
Kernel	5.13.0-40-generic	Linux 5.17.5-76051705-generic
CPU	Intel Xeon E5-1660 v2 (12) @ 4.000GHz	Intel(R) Core(TM) i5-9500 CPU @ 3.00GHz
GPU	NVIDIA Quadro K2000	UHD Graphics 630
RAM	32GiB	16GiB

C. Results

The results from the timing measurements regarding the average data age can be found in Table II for 1 ms and in Table III for 2 ms. The Figure 4, Figure 5 and Figure 6 are showing the composition of the data age for the different architectures using the different DDS vendors for the 1 ms update period. While performing the measurements for the FastDDS, it was seen that the system was crashing randomly with the exit code -6 (client will not receive response) after a number higher than 50 robots in the system. Connections and measurements were possible after several systems relaunches, but connecting 90 robots to the system was impossible during the experiment. For GurumDDS, at the setup with 90 robots at a 1 ms update period, the connection service was not responding for the last robots, and the measurements could not be taken.

TABLE II: Average data age with 1ms update period

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

For all measured configurations, it can be seen that the average data age increases considerably at a specific number of robots, but much more for M2O than for O2O, except for GurumDDS with 50 and 70 robots for 2 ms date period. Furthermore, it can be seen that in O2O, the most significant part of the data ageing occurs between timestamp T2 and T3. In comparison,

TABLE III: Average data age with 2ms update period

DDS	CycloneDDS		FastDDS		GurumDDS	
Robots	M2O	O2O	M2O	O2O	M2O	O2O
10	1.334	1.590	1.259	1.122	1.181	1.837
30	2.502	1.788	1.340	1.354	1.671	1.707
50	241.665	2.310	1.710	1.566	405.730	467.093
70	295.255	13.753	4.697	2.865	83.183	1852.359
90	228.581	22.758	-	-	705.112	469.506

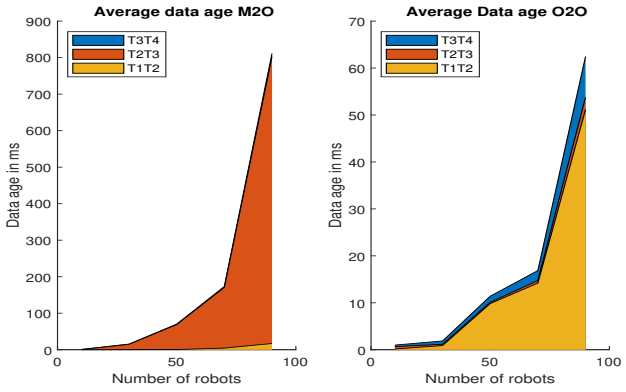


Fig. 4: Average data age in M2O and O2O using CycloneDDS for 1 ms update period with different magnitudes. The colours are representing the different parts of the network.

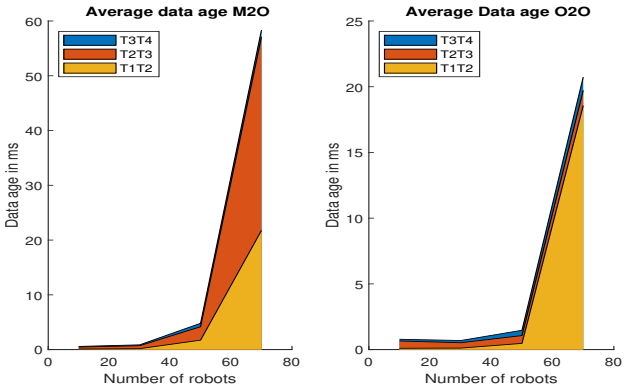


Fig. 5: Average data age in M2O and O2O using FastDDS for 1 ms update period with different magnitudes. The colours are representing the different parts of the network.

in M2O, the most significant part of the ageing occurs between timestamps T1 and T2 and T3 and T4, indicating the time it takes to transfer the state and state list messages. The ageing between T2 and T3 in O2O is related to the data staying longer inside the tracker. That can occur when updates are missed, and data is used several times for creating the state list. That occurs significantly more for M2O and explains the difference in the magnitude of the data age (see Figure 7). From the analysis, it can be seen that for up to 70 robots, FastDDS is showing the minor data ageing, followed by CycloneDDS. In contrast, the ageing with GurumDDS is significantly higher, especially for 70 robots which shows a peak at 2 ms. The update miss ratio is shown for the 1 ms update rate in Figure 7. This value describes the percentage of missed robot updates. For O2O, the update misses are around the same percentage for the different numbers of connected robots. For M20, for

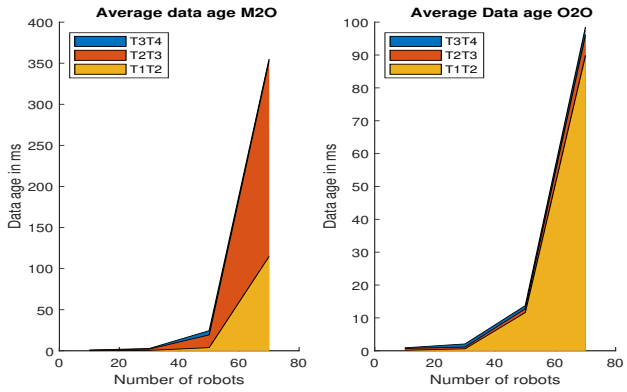


Fig. 6: Average data age in M2O and O2O using GurumDDS for 1 ms update period with different magnitudes. The colours are representing the different parts of the network.

a higher number of robots, the ratio is increasing in all DDS vendors up to 98 %. The processor utilization was measured

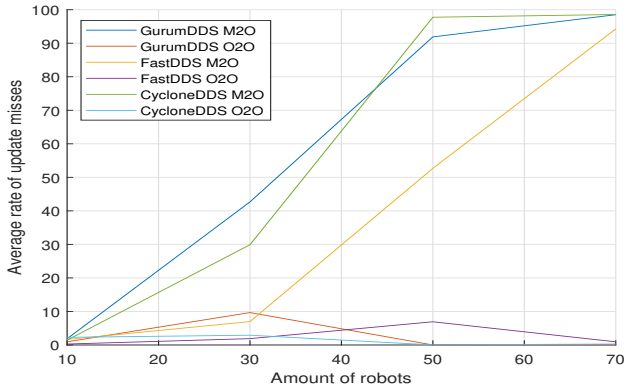


Fig. 7: Update miss ration for the different architectures and DDS vendors for 1 ms update period.

during the measurements and presented in Table IV. During the measurements, there was no difference in the chosen communication architecture recognized. It can be seen that the increase of the nodes causes a significantly higher utilization of the processor than CycloneDDS and FastDDS. The results for

TABLE IV: CPU utilization, no differences for the architectures were seen.

DDS	CycloneDDS		FastDDS		GurumDDS	
Robots	1 ms	2 ms	1 ms	2 ms	1 ms	2 ms
10	30	15	25	20	25	20
30	40	20	45	40	65	35
50	65	30	65	50	90	65
70	85	45	95	60	100	90
90	100	70	-	-	-	100

the distributed setup using four computers and ROS2 Galactic with CycloneDDS can be seen in Figure 8 and Table V. It can be seen that, like in the first part, the average data age is significantly higher for M2O than for O2O. Furthermore, the most significant impact on the data ageing is for M2O between T2T3 and for O2O between T1T2 and T3T4. The update miss ratio shows an increase close to 100 % for M20, while O2O is showing zero to 3 % update misses.

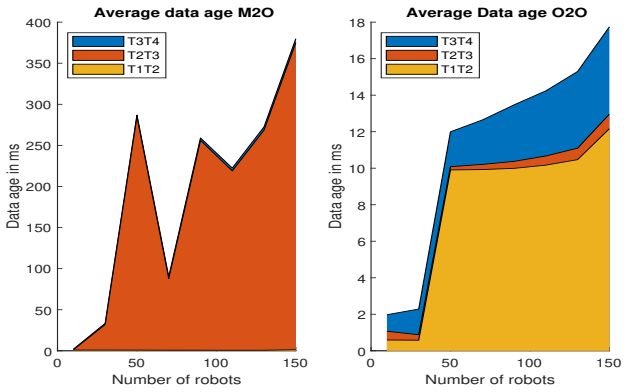


Fig. 8: Average data age in M2O and O2O using distributed setup and CycloneDDS for 1 ms update period with different magnitudes. The colours are representing the different parts of the network.

TABLE V: Measurement results for distributed setup with 1 ms update period using CycloneDDS

	Data age in ms		Update miss ratio		CPU Utilization			
Robots	M2O	O2O	M2O	O2O	Tr	R1	R2	Meas
10	2.021	1.976	0.140	3.146	10	5	5	5
30	33.426	2.284	41.757	2.558	10	30	30	5
50	286.904	11.995	75.856	0.000	20	30	30	20
70	90.199	12.641	98.571	0.000	20	40	40	20
90	259.049	13.470	98.889	0.000	20	60	60	20
110	222.391	14.237	99.091	0.000	20	70	70	20
130	272.596	15.296	99.231	0.000	20	85	85	20
150	379.953	17.743	99.333	0.003	20	100	100	20

D. Discussion

Even though the first part of the evaluation is conducted on only a single computer, whereas a legacy system would have the different parts of the network operating on different computers, some results can be taken from the experiments. FastDDS is showing better results regarding the data age and update misses but is showing crash behaviour throughout the launch. The lowest data ageing might be explained by the zero-copy functionality of FastDDS, which could not be used in a distributed setup. CycloneDDS is more stable throughout the launch, and the data ageing is not as significant as for GurumDDS, which even more causes the highest processor utilization. Therefore, CycloneDDS might be the better choice for a robot system until the crashing problem in FastDDS is found and solved. Then another evaluation would be needed in a more realistic distributed setup for both. For the higher amount of robots, the O2O architecture is significantly better than the M2O architecture regarding the data age. Furthermore, O2O stays robust against update misses, as only the data age increases, but not as much as in M2O. This behaviour matches the expectations that in M2O, the data of robots might get overwritten by other nodes without the tracker noticing, which causes the data to age inside the tracker node. In O2O, the sending and reception of the data are the most significant problems at high utilization. Furthermore, the seen behaviour is validated in a distributed setup closer to reality.

V. Conclusion

A sample system of a centralized multi-agent robot system was designed and simulated successfully using a dynamic

connection handling approach. The system was used to evaluate the use of CycloneDDS, FastDDS and GurumDDS in the distribution of ROS2 Galactic. FastDDS showed the best results regarding update misses and data-ageing on a single computer. However, it was the most unstable regarding the system launch and execution, where random crash behaviour was seen. Therefore, CycloneDDS is a preferred choice in the proposed setup. No crash behaviour was seen and showed better results regarding data age, update miss ratio and CPU utilization than GurumDDS. The simulations and measurements regarding the communication architectures have shown that the data age is generally increasing over the number of connected robots for both architectures. The chosen communication approach furthermore influences the data age. In a many-to-one approach, where multiple nodes communicate through the same topic, the data ageing is more significant than in the one-to-one approach. The update miss ratio increased significantly for the many-to-one communication approach over the number of connected robots.

ACKNOWLEDGMENT

The work in this paper is supported by the Swedish Knowledge Foundation (KKS) via the project DPAC & HERO. We thank all our industrial partners, especially Volvo GTO.

REFERENCES

[1] M. Ford, "Rise of the robots : Technology and the threat of a jobless future," 2015.

[2] A. Khalif and L. Jutvik, "Path following for atrs using embedded nonlinear model predictive control," 2021.

[3] S. Mubeen, T. Nolte, M. Sjödin, J. Lundbäck, and K.-L. Lundbäck, "Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints," *Softw. Syst. Model.*, vol. 18, no. 1, 2019.

[4] G. Dudek, M. R. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, no. 4, pp. 375–397, 1996.

[5] S. Dehnavi, M. Koedam, A. Nelson, D. Goswami, and K. Goossens, "Compros: A composable ros2 based architecture for real-time embedded robotic development," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6449–6455.

[6] A. Testa, A. Camisa, and G. Notarstefano, "Choirbot: A ros 2 toolbox for cooperative robotics," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, 2021.

[7] C. McCord, J. P. Queralta, T. N. Gia, and T. Westerlund, "Distributed progressive formation control for multi-agent systems: 2d and 3d deployment of uavs in ros/gazebo with rotors," in *European Conference on Mobile Robots (ECMR)*, 2019, pp. 1–6.

[8] S. Vorapojpisut, M. Lhongpol, R. Amornlikitsin, and T. Phuapaiboon, "A robot augmented environment based on ros multi-agent structure," in *4th International Conference on Control, Robotics and Cybernetics*, 2019.

[9] G. Conte, D. Scaradozzi, L. Sorbi, L. Panebianco, and D. Mannocchi, "Ros multi-agent structure for autonomous surface vehicles," in *OCEANS Genova*, 2015.

[10] S. Noh and J. Park, "System design for automation in multi-agent-based manufacturing systems," in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, 2020, pp. 986–990.

[11] A. Barciś, M. Barciś, and C. Bettstetter, "Robots that sync and swarm: A proof of concept in ros 2," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2019, pp. 98–104.

[12] E. Erős, M. Dahl, K. Bengtsson, A. Hanna, and P. Falkman, "A ros2 based communication architecture for control in collaborative and intelligent automation systems," *Procedia Manufacturing*, vol. 38, 2019.

[13] T. Blaß, D. Casini, S. Bozhko, and B. B. Brandenburg, "A ros 2 response-time analysis exploiting starvation freedom and execution-time variance," in *IEEE Real-Time Systems Symposium (RTSS)*, 2021, pp. 41–53.

[14] Y. Tang, Z. Feng, N. Guan, *et al.*, "Response time analysis and priority assignment of processing chains on ros2 executors," in *2020 IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 231–243.

[15] Z. Li, A. Hasegawa, T. Azumi, "Autoware-perf: A tracing and performance analysis framework for ros 2 applications," *Journal of Systems Arch.*, vol. 123, 2022.

[16] D. Casini, T. Blaß, I. Lütkebohle, and B. Brandenburg, "Response-time analysis of ros 2 processing chains under reservation-based scheduling," in *31st Euromicro Conference on Real-Time Systems*, 2019, pp. 1–23.

[17] Y. Yang and T. Azumi, "Exploring real-time executor on ros 2," in *IEEE International Conference on Embedded Software and Systems (ICESS)*, 2020, pp. 1–8.

[18] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ros2," in *2016 International Conference on Embedded Software (EMSOFT)*, 2016, pp. 1–10.

[19] J. Park, R. Delgado, and B. W. Choi, "Real-time characteristics of ros 2.0 in multiagent robot systems: An empirical study," *IEEE Access*, vol. 8, 2020.

[20] T. Blass, "Real-time execution management in the ros 2 framework," *Ph.D. dissertation*, 2022.

[21] L. Puck, P. Keller, T. Schnell, *et al.*, "Performance evaluation of real-time ros2 robotic control in a time-synchronized distributed network," in *17th International Conference on Automation Science and Engg.*, 2021.

[22] L. Puck, P. Keller, T. Schnell, *et al.*, "Distributed and synchronized setup towards real-time robotic control using ros2 on linux," Oct. 2020.

[23] C. S. V. Gutiéerrez, "Towards a distributed and real-time framework for robots: Evaluation of ros 2.0 communications for real-time robotic applications," *Tech. Rep*, 2018.

[24] T. Kronauer, J. Pohlmann, M. Matthé, T. Smejkal, and G. Fettweis, "Latency analysis of ros2 multi-node systems," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2021.

[25] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ros2," in *Proceedings of the 13th International Conference on Embedded Software*, 2016.

[26] Z. Chen, "Performance analysis of ros 2 networks using variable quality of service and security constraints for autonomous systems," Naval Postgraduate School Monterey United States, Tech. Rep., 2019.