A Study of Optimizing Inter-node Communication with an IPC-based DDS Implementation in ROS 2

Katsuya Matsubara[1,a)] Ren Morita[1] Sho'ji Suzuki[1]

Abstract: The Robot Operating System 2.0, known as ROS 2, has been being developed from scratch to support new applications. One of the most significant changes in ROS 2 is to follow the DDS standard specification as an interface of inter-module communication. ROS 2 allows specifying only one DDS implementation to be used for the whole system through application code or an environment variable at runtime. Nevertheless, each communication may have different characteristics and each DDS implementation may be optimized for different communication types. So we have proposed a mechanism to allow multiple DDS implementations existing in a system and to dynamically bind one into each communication. Under assuming the dynamic DDS binding, this paper describes a new DDS implementation with 'OpenBinder', which is a low-latency IPC framework and available in recent Linux kernel, especially for the inter-node communication within the same machine.

Keywords: Robot Operating System, Inter-process Communication, Publish-subscribe Messaging, OpenBinder

1. Introduction

The Robot Operating System (ROS), that is an open-source software framework[1] organized by the Open Source Robotics Foundation[2], is the de-facto standard platform for modern robots. ROS 2 has been presently being re-designed and implemented from scratch in order to address growing requirements like multiple robots cooperation, smaller embedded system, real-time, and unstable networks. One of the most significant changes in ROS 2 is to adopt the Data Distribution Service (DDS) specification, that has been standardized by OMG, as a framework for inter-node communication. It allows selectively using one for each system from multiple DDS implementations provided by different vendors and communities.

Unfortunately, the inter-node communication in ROS 2 still has issues in practice. First, each message transfer in a robot system may have different characteristics; periodic sensor output, arbitrary device control, and contiguous video streaming, for example. Nevertheless, the ROS 2 current implementation requires to specify only one DDS implementation to be used for the whole system. Regarding this issue, we have proposed a mechanism of dynamic binding for DDS implementations[4], which allows selectively using a proper one according to the characteristics of each message transfer rather than each system.

Besides, it could be an issue that all of the current existing DDS implementations for ROS 2, including FastRTPS, Connext, and OpenSplice, rely on the UDP multicast regardless of remote or local communication. Using the UDP multicast for local communication may affect performance and power consumption because of useless network transfers. In fact, a more significant part of the inter-node communication in some ROS-based systems such as a drone and an autonomous robot could be local[5]. Especially for the communication among nodes which placed on the same machine, we have been implementing a new DDS implementation which can suppress the useless network transfers and improves communication latency and throughput. The DDS implementation can be co-existed with other DDS implementations in a system and can be utilized only for specified local communications by our dynamic DDS binding as described above.

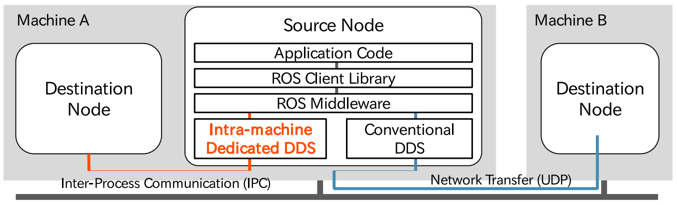


Fig. 1 Bind different DDS implementations for local and remote nodes

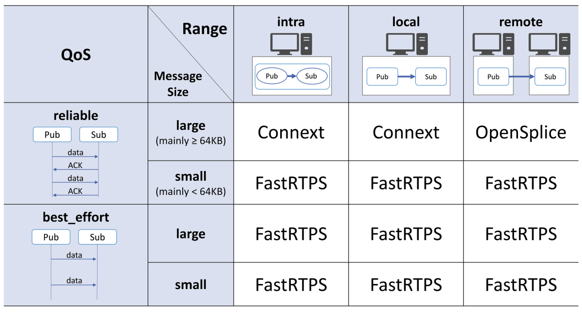
This paper explains our DDS binding mechanism briefly, discusses the comparison of Linux IPC frameworks, and then shows results of the preliminary evaluation with a prototype implementation consequently.

2. Dynamic DDS Binding for ROS 2

We have implemented functionality to dynamically switch DDS implementations with following the combination of the parameters that show characteristics of inter-node communication. Each specified topic has attached some parameters to associate the features of DDS implementations with the attributes of inter-node communication. We have defined three parameters; data size, communication range, and QoS. The data size can be specified with 'large' or 'small.' The communication range can be assumed through 'intra,' 'local,' or 'remote.' The QoS parameter can take a value 'reliable' or 'best effort.'

Each topic can select a proper DDS implementation from a matrix of the combination of the three parameters and DDS im-

1 Future University Hakodate, 116-2 Kamedanakano, Hakodate, Hokkaido 041–8655, Japan
a) matsu@fun.ac.jp



QoS	Range / Message Size	intra	local	remote
reliable	large (mainly ≥ 64KB)	Connext	Connext	OpenSplice
	small (mainly < 64KB)	FastRTPS	FastRTPS	FastRTPS
best_effort	large	FastRTPS	FastRTPS	FastRTPS
	small	FastRTPS	FastRTPS	FastRTPS

Pub
Sub
data
ACK

Fig. 2 An example of the DDS binding matrix

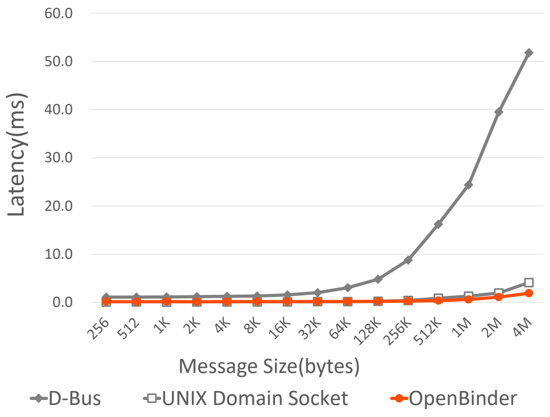


Fig. 3 Comparison of latency on each Linux IPC framework

plementations based on the attached parameters. Fig. 2 shows an example of the matrix. This matrix example declares should be used for reliable and large data transfer among nodes in the same machine, OpenSplice for reliable large data transfer to remote nodes, FastRTPS for the other typed transfers.

3. Comparison of Linux IPC Frameworks

For implementing the intra-machine dedicated DDS, we evaluated UNIX domain sockets, OpenBinder, and D-Bus, which can be used on Linux, the target OS platform for ROS. The latency comparison result, shown in Fig. 3, indicates OpenBinder could be suitable for implementing low-latency inter-node communication.

4. Prototyping and Preliminary Evaluation

We have been implementing an intra-machine dedicated DDS implementation with OpenBinder. The most important technical issue to be solved in the implementation is that the communication model is different between OpenBinder and the DDS specification. OpenBinder adopts an RPC-like synchronous and round-trip messaging model. On the other hand, the DDS specification requires the Pub-sub messaging model which features asynchronous and one-way communication. We implemented a prototype ’PubSubBinder’ as a wrapper which realizes the pub-sub communication model on OpenBinder.

Fig. 5 shows the latency measurement results when sending messages to two processes at the same time. In the OpenBinder’s communication model, messages cannot be sent to multiple processes simultaneously, so the second (B) latency includes the transmission of (A) and the latency of its acknowledgment (See

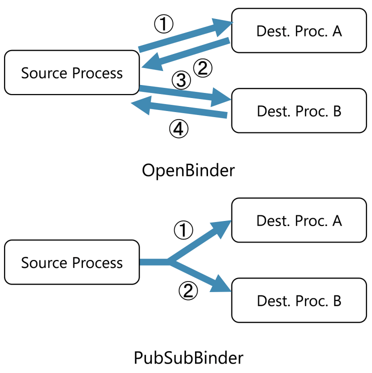


Fig. 4 Transmission flow for two destination processes

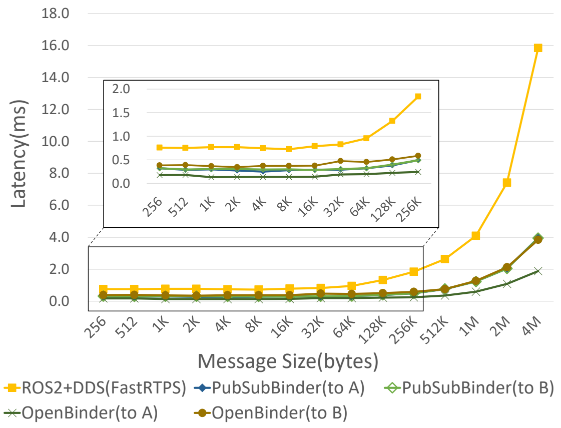


Fig. 5 Latency in sending to two processes simultaneously

Fig. 4). Since the PubSubBinder can send messages to two processes simultaneously, the latency for (A) and the latency for (B) were almost the same.

5. Conclusion

To optimize the communication among ROS 2 nodes where in the same machine, we have proposed the dynamic DDS binding and an intra-machine dedicated DDS implementation. OpenBinder could be suitable for low-latency communication; nevertheless, its communication model is different from the DDS one. So we have implemented a wrapper to realize the publish-subscribe messaging on the OpenBinder’s RPC communication model. The result of preliminary evaluation with the prototyping shows that the wrapper can be implemented with reasonable overhead compared to the conventional DDSimplementation.

Further evaluations with practical ROS systems would be required to validate our DDS binding mechanism and the intra-machine dedicated DDS implementation.

References

[1] ROS.org, `http://www.ros.org/`(accessed December 17 2018).
[2] M. Quigley et al., ”ROS: an open-source Robot Operating System,” in ICRA workshop on open source software, vol. 3, p. 5, 2009.
[3] Data Distribution Services, `https://www.dds-foundation.org/`(accessed September 21 2019).
[4] R. Morita and K. Matsubara, ”Dynamic Binding a Proper DDS Implementation for Optimizing Inter-Node Communication in ROS2,” in 24th IEEE Int. Conf. on Embedded and Real-Time Comp. Sys. and App., pp. 246-247, 2018.
[5] J. Scherer, S. Yahyanejad, et al., ”An Autonomous Multi-UAV System for Search and Rescue,” ACM DroNet’15, pp. 3338, A2015.