# Preliminary Implementation of Measurement Method for ROS1 Callback Execution Time

MASATO FUKUI[†1] YOICHI ISHIWATA[†2]
TAKESHI OHKAWA[†3] MIDORI SUGAYA[†1]

***Abstract***: Currently, Robot Operating System (ROS) is widely used as a framework for distributed processing in autonomous mobile robot software development. ROS uses a process as a node and communicates between nodes by publishing/subscribing asynchronously. These mechanisms make it easier for users to use an asynchronous system with multiple processes. However, in ROS, it is difficult to measure the function called by the subscriber when it returns from waiting for an event (callback function). Therefore, in this study, we triggered messages on the ROS middleware to identify the part where the node function is executed and measured the execution time. This paper presents our results on running the application and measuring the node processing time.

***Keywords***: ROS, execution time, measurement

## 1. Introduction

In recent years, Robot Operating System (ROS) [1] has been widely used as a framework for distributed systems in robot software development. ROS is a software that includes useful tools and libraries to support the development of robot applications. Robot software development requires the development of various modules such as sensor, actuator, driver, and data processing logic. In ROS, which is a distributed system, the application is implemented separately in multiple nodes. This enhances code reusability and productivity. Data is exchanged between the divided nodes in the publish/subscribe pattern. This makes it possible to communicate without specifying other parties, but at the same time makes the structure of the application complicated. Consequently, it is difficult to correctly evaluate the performance of a ROS application in which multiple nodes operate asynchronously. In order to measure the performance of the application, it is necessary to investigate the execution time of each node and the communication time between them. However, ROS communicates via queue. In addition, it is difficult to grasp the callback execution time of node. For this reason, there is a problem that it is difficult to investigate the overall performance.

Therefore, this research proposes a mechanism to measure the execution time of node, especially at the time of callback of ROS. We also implemented the proposed mechanism in ROS and measured the execution time of the SLAM Gmapping algorithm, which is a typical ROS application consisting of multiple nodes.

The structure of this paper is as follows: Section 2 describes the problem of measurement with ROS. Section 3 explains the proposed method. Section 4 describes time measurement using the proposed method. A conclusion is given in Section 5.

## 2. ROS Issues

### 2.1 Structure of ROS

This section describes the basic structure of ROS. In ROS, node is used as the basic unit of software. This is the same as the process in the OS. Each node has a single function. In robot software development using ROS, a system is constructed by connecting multiple nodes. For ROS communication, the publish/subscribe pattern via a communication path called topic is used. Figure 1 illustrates data transmission and reception by the ROS nodes. In the figure, node and topic are represented by ellipse and rectangle, respectively. Node has a publisher that sends data and a subscriber that subscribes to it and receives and processes the data when there is an update. In our proposed model, we created two nodes, */talker_node* for publisher and */listener_node* for subscriber, and these nodes communicate via the topic */chatter*.

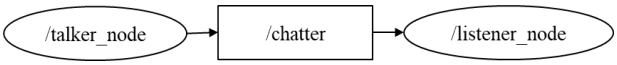


Figure 1 ROS communication model

The publisher, which is the starting point of the data flow, has a role in acquiring data from sensors and passing that data to other nodes. The subscriber receives and processes the data. Then, the processing result is sent to another node or used for robot operation. There are two types of ROS, ROS1 and ROS2. In this study, ROS1 was targeted.

### 2.2 Issues in measuring execution time in ROS applications

Node subscribes to the input topic and waits for an event to update the topic, and set a handler function (callback function) that triggers the reception of a message in node in advance. When a message is received, a callback function is called to process the data and output the result as a topic with a different name. We defined this processing time as the node execution time.

When measuring the node execution time, it is not possible to measure only the input/output topic time. Figure 2 shows the message flow of the ROS nodes. Similar to other message queuing systems, the ROS publish/subscribe pattern also puts received messages into the receive queue. Then, the message is

†1 Shibaura Institute of Technology
†2 Ales K.K.
†3 Tokai University

taken out from the queue and processed. The time that the message is waiting in the queue cannot be measured. In addition, the message may be discarded if the queue overflows. Therefore, it is not possible to measure the execution time of node just by monitoring topic. From these points, ROS has a problem that it is difficult to measure the execution time of an application in detail.

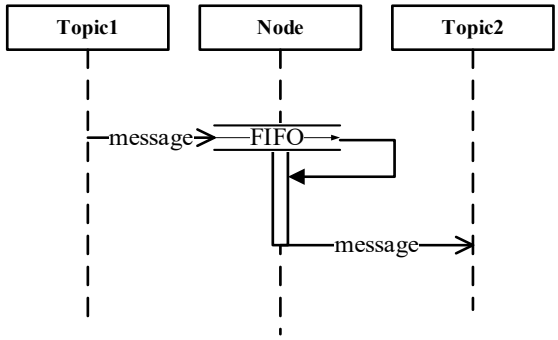


Figure 2 Dataflow in ROS publish/subscribe pattern. Node temporarily adds the message received from Topic1 to the message queue. When a message is queued, node's handler function (callback function) retrieves the data and executes the process. The processing result is sent to another node using Topic2.

### 2.3 Measurement tools in the ROS

A rosbag [2] is a toolset for recording and playing back ROS topic. It can record the time the message is sent and the content of the message. The rosbag subscribes the topic to be recorded, and records the current time and data contents each time the topic is changed. Since this tool targets topic, it is not possible to measure the execution time of node.

## 3. "hCT": high precision Callback Tracer

The ROS measurement tool cannot measure callback functions. Therefore, in this resarch, we designed and implemented a tool for tracking callback functions in ROS node.

It is necessary to track the callback function inside ROS in order to identify which topic-related function was called along with the callback function. Inside ROS, two types of keys are used to identify node and callback functions. A hash value called md5sum is used in the API that creates a node and sets a callback function. On the other hand, the key called *removal_id* is used in *CallbackQueue :: CallOneCB ()*, which is the API that actually calls the callback function. Therefore, hCT binds and saves these two keys to identify the callback function. To measure how long it took to process the node, we used the *clock_gettime()* function before and after calling the callback function and calculated the difference.

## 4. Measurement

We measured the execution time of the ROS node using the proposed measurement tool. We measured the execution time of */turtlebot3_slam_gmapping* node of the SLAM gmapping package, which is one of the ROS applications. SLAM gmapping is an application for map generation and robot self-position estimation using robot sensor data.

For the experimental environment, we used Gazebo, a ROS physics simulator, and the TurtleBot3 burger robot. SLAM gmapping was executed using the sensor data from the TurtleBot3 burger installed in the simulation space. At this time, the time required for processing was measured using hCT.

Table 1 shows the experiment environment. The experiment was performed using Ubuntu 18.04.5 on KVM. After starting the Gazebo simulator, the target node was executed for 10 minutes. We measured the execution time of the callback function for each event.

Table 1 Experiment environment

| | |
|---|---|
| **CPU** | Intel Xeon Gold 6230 (20Core, 2.1GHz, 27.5MB cache) x 2 |
| **RAM** | 32GiB |
| **OS** | Ubuntu18.04.5 |
| **ROS distro** | melodic |

Figure 3 shows the measurement result. The horizontal axis shows the number of simulations SLAM was performed. The vertical axis shows the execution time at that time in seconds. Except for the first four runs, which have significantly shorter execution times, the average execution time was 3.81 seconds, the minimum value was 2.71 seconds, the maximum value was 4.86 seconds, and the variance was 0.25 seconds.

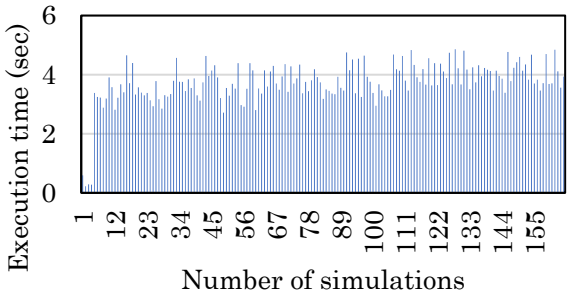


Figure 3 Measurement result

## 5. Conclusion

In this paper, we proposed the execution time measurement method of the ROS Node callback function. Based on this, we implemented a prototype of the measurement tool. We confirmed that the execution time of SLAM gmapping node can be measured using our designed tool.

## References

[1]M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, et al., "Ros: an open-source robot operating system", ICRA workshop on open source software, vol. 3, no. 3.2, pp. 5, 2009.

[2]"rosbag - ROS Wiki". https://wiki.ros.org/rosbag, (accessed 2021-10-18).

**Acknowledgments** This research was supported by Japan Science and Technology Agency (JST), CREST, JPMJCR19K1. We would like to express our gratitude to all of them.