Fachhochschule Dortmund

University of Applied Sciences and Arts

(Master Embedded Systems for Mechatronics)

The Robot Operating System 2 (ROS 2) Evaluation

Research’s Thesis

Author: LE, Quang Hien

Matriculation Number: 7206974

Supervisor: Prof. Dr. Christof Röhrig

Co-Supervisor: Merlin Stampa

Date: 06/2021

# Abstract

Since ROS 1 (Robot Operating System) has been initially created in 2007 by Willow Garage, it has become one of the most common open-source robotics communities. Along with many advantages, it has also some disadvantages like not providing real-time control and robot security, ROS 2 has been created to leverage what is great about ROS 1 and providing what is not. The development of UAV (Unmanned Aerial Vehicle) is complicated and always requires real-time operation and non-ideal network handling when needed. While ROS (Robot Operating System) 1 cannot support these features, ROS 2 has been developed to fill the void. Here the QoS (Quality of Service) will be evaluated in a lossy wireless network. An experiment is set up using a micro-RTPS bridge, Fast-RTPS(DDS) to establish a bridge between the PX4-Autopilot (open-source autopilot system for autonomous aircraft) and ROS2 which sends setpoints to the PX4 and also receives HD images and sensor data from PX4. The throughputs, delays, message ages, message periods will be observed and assessed. The result shows that with properly adjusting QoS settings, the setpoints messages to the PX4 which is the most important are all transmitted without a very single message loss.

# Declaration

The Master’s thesis needs to contain a declaration regarding scientific integrity. It is as follows:

German version (legally relevant, see RahmenPO §31 (2)):

Ich versichere, dass ich meine Arbeit – bei einer Gruppenarbeit meinen entsprechend

gekennzeichneten Anteil der Arbeit – selbstständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

*Datum, Unterschrift*

English version:

I hereby confirm that I have written the Master Thesis at hand independently – in case of group work: my respectively designated part of the work -, that I have not used any sources or materials other than those stated and that I have highlighted any citations properly.

*Date, signature*

Table of Contents

[Abstract ii](#_Toc74232300)

[Declaration iii](#_Toc74232301)

[List of Figures vi](#_Toc74232302)

[List of Tables vii](#_Toc74232303)

[1. Introduction 1](#_Toc74232304)

[1.1. Motivation 1](#_Toc74232305)

[1.2. Problem statement 2](#_Toc74232306)

[1.3. Structure 2](#_Toc74232307)

[2. Background 2](#_Toc74232308)

[2.1. ROS 2 basics 3](#_Toc74232309)

[2.1.1. Nodes 3](#_Toc74232310)

[2.1.2. Topics 3](#_Toc74232311)

[2.1.3. Services 3](#_Toc74232312)

[2.1.4. Parameters 3](#_Toc74232313)

[2.1.5. Actions 3](#_Toc74232314)

[2.1.6. Workingspace 3](#_Toc74232315)

[2.1.7. Packages 3](#_Toc74232316)

[2.1.8. Launch files 3](#_Toc74232317)

[2.2. RTPS/DDS Interface: PX4-Fast RTPS(DDS) Bridge 3](#_Toc74232318)

[2.2.1. Architectural overview 3](#_Toc74232319)

[2.2.2. Code generation 3](#_Toc74232320)

[2.2.3. uORB messages support 3](#_Toc74232321)

[2.2.4. Client (PX4/PX4-Autopilot) 3](#_Toc74232322)

[2.2.5. Agent in an Offboard Fast DDS interface (ROS-independent) 3](#_Toc74232323)

[2.3. Quality of Service 3](#_Toc74232324)

[2.3.1. Overview 3](#_Toc74232325)

[2.3.2. QoS policies 3](#_Toc74232326)

[2.3.3. Comparison to ROS 1 3](#_Toc74232327)

[2.3.4. QoS compatibilities 3](#_Toc74232328)

[2.3.5. QoS events 3](#_Toc74232329)

[2.3. Evaluation tools 3](#_Toc74232330)

[2.3.1. Linux network traffic control utility 3](#_Toc74232331)

[2.3.2. ROS 2 Topic Statistics 3](#_Toc74232332)

[3. Experiments Implementation 3](#_Toc74232333)

[3.1. Architecture of the experiments 3](#_Toc74232334)

[3.2. Experiment with the default policy of QoS settings 3](#_Toc74232335)

[3.2.1. Introduction 3](#_Toc74232336)

[3.2.2. Implementation 3](#_Toc74232337)

[3.3. Experiment with the best-effort policy of QoS settings 4](#_Toc74232338)

[3.3.1. Introduction 4](#_Toc74232339)

[3.3.2. Implementation 4](#_Toc74232340)

[4. Results Discussion 4](#_Toc74232341)

[4.1. Experiment with the default policy of QoS settings 4](#_Toc74232342)

[4.2. Experiment with the best-effort policy of QoS settings 4](#_Toc74232343)

[4.3. Comparison 4](#_Toc74232344)

[5. Conclusion 4](#_Toc74232345)

[Bibliography 5](#_Toc74232346)

# List of Figures

[Figure 1: Example for a figure 1](#_Toc45008050)

# List of Tables

[Table 1: Example for a table 1](#_Toc45010229)

# 1. Introduction

## 1.1. Motivation

To create a development environment for PR2 robot, Willow Garage has started developing ROS. It is aimed to provide the software tools that users can take to undergo research and production development of PR2. However, there is not only PR2 in the world, so the team of ROS had decided to support other robots as well and then made a great effort to create software interfaces that allow software as much as possible to be used elsewhere.

ROS has been successfully fulfilled requirements to provide the software tools for PR2 in the use case of a single robot, with no real-time requirements, excellent network connectivity, and applications in mostly academia [1]. Nonetheless, when adapting to a variety of robots like wheeled robots, legged humanoids, industrial arms, outdoor ground vehicles, aerial vehicles, ROS has shown some disadvantages with other new use cases.

Furthermore, a huge number of robots used ROS in the industry alongside the academic research which is the initial focus. ROS-based products like commercial cleaning robots, manufacturing robots, agricultural robots are coming to the market [1]. ROS has also been used by government agencies.

With the new use cases of the broader ROS community including teams of multiple robots, small embedded platforms, real-time systems, non-ideal networks, production environments, Prescribed patterns for building and structuring systems [1], there should be some changes to adapt with these use cases.

At the core of ROS is an anonymous publish-subscribe middleware system that is built almost entirely from scratch [1]. Since 2007, there has been development, adoption of several new technologies that are relevant to ROS such as Zeroconf, Protocol Buffers, ZeroMQ (and the other MQs), Redis, WebSockets, and DDS (Data Distribution Service) [1]. Now, ROS should be rebuilt like a middleware system using existing open-source libraries so that it can be less code, taking a lot of features from open source libraries.

It has been several years, API has been changed a lot. Although the current APIs are still stable, there are not the best for ROS. So the new ROS, aka ROS 2, will have new APIs which are sophisticated of the community.

From the first alpha release of ROS 2 was in 2015 to the latest version of ROS 2 Foxy Fitzroy, ROS 2 has gained popularity and ROS developers has started to discover ROS 2. As a result, ROS 2 evaluation should be carried out to obtain a full understanding of all the great features of ROS 2 that ROS 1 does not have.

## 1.2. Problem statement

Originally, ROS 1 does not support real-time which is the key feature of many robotics systems.

## 1.3. Structure

Chapter 2, Background, describes features of ROS 1 and ROS 2, comparison between them along with evaluation tools for the experiments. Chapter 3, Experiment Implementation, explains how ROS 2 had been evaluated. Chapter 4, Results Discussion, reviews the outcome of the experiments. Chapter 5, Conclusion, gives the summary of the results and the answers to the problem statement.

# 2. Background

Robot Operating System (ROS) is an open-source robotics middleware suite. Although ROS is not an operating system but a collection of software frameworks for robot software development, it provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [2]. ROS is not a real-time Operating System, but it is possible to integrate ROS with real-time code [3].

The primary goal of ROS is to support code reuse in robotics research and development. ROS is a distributed framework of processes (aka Nodes) that enables executables to be individually designed and loosely coupled at runtime. These processes can be grouped into Packages and Stacks, which can be easily shared and distributed [3].

## 2.1. ROS 2 basics

### 2.1.1. Nodes

### 2.1.2. Topics

### 2.1.3. Services

### 2.1.4. Parameters

### 2.1.5. Actions

### 2.1.6. Workingspace

### 2.1.7. Packages

### 2.1.8. Launch files

## 2.2. RTPS/DDS Interface: PX4-Fast RTPS(DDS) Bridge

### 2.2.1. Architectural overview

### 2.2.2. Code generation

### 2.2.3. uORB messages support

### 2.2.4. Client (PX4/PX4-Autopilot)

### 2.2.5. Agent in an Offboard Fast DDS interface (ROS-independent)

## 2.3. Quality of Service

### 2.3.1. Overview

### 2.3.2. QoS policies

### 2.3.3. Comparison to ROS 1

### 2.3.4. QoS compatibilities

### 2.3.5. QoS events

## 2.3. Evaluation tools

### 2.3.1. Linux network traffic control utility

### 2.3.2. ROS 2 Topic Statistics

# 3. Experiments Implementation

## 3.1. Architecture of the experiments

## 3.2. Experiment with the default policy of QoS settings

### 3.2.1. Introduction

### 3.2.2. Implementation

## 3.3. Experiment with the best-effort policy of QoS settings

### 3.3.1. Introduction

### 3.3.2. Implementation

# 4. Results Discussion

## 4.1. Experiment with the default policy of QoS settings

## 4.2. Experiment with the best-effort policy of QoS settings

## 4.3. Comparison

# 5. Conclusion

# Bibliography

|  |  |
| --- | --- |
| [1] | B. Gerkey, "Why ROS 2?," Open Source Robotics Foundation, Inc., n.d.. [Online]. Available: https://design.ros2.org/articles/why\_ros2.html. [Accessed 23 March 2021]. |
| [2] | "Robot Operating System," Wikipedia, n.a.. [Online]. Available: https://en.wikipedia.org/wiki/Robot\_Operating\_System. [Accessed 9 April 2021]. |
| [3] | "ROS Introduction," Open Robotics, n.a.. [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Accessed 9 April 2021]. |