Robotic Operating System

Version 1.0

Ruotolo Vincenzo

January 16, 2022

Contents

List of Code Listings

Part I
Introduction

1 Definition

ROS is an open-source, meta-operating system for robotic systems, providing:

- hardware abstraction
- low-level device control
- implementation of commonly used functionality
- message-passing between processes
- package management
- tools and libraries for obtaining, building, writing, and running code across multiple computers

The purpose of ROS is:

- Support code reusability in robotics research and development
- Thin: code written for ROS can be used with other robot software frameworks
- ROS-agnostic libraries: the preferred development model is to write ROS-agnostic libraries with clean functional interfaces
- Language independence: implementation in different programming language (most used C++ and Phyton)
- Easy testing: built-in unit/integration test framework called rostest
- Scaling: ROS is appropriate for large runtime systems and for large development processes.

2 Description

ROS is a distributed framework of processes called nodes which enable executables to be individually designed and loosely coupled at runtime. Communication between nodes is implemented by means of topics, services, actions and parameters, for each of which is defined a data structure. Messages can be recorded through bag functionality which allows to reproduce offline them. The launch functionality allows to run multiple nodes simultaneously and settings their parameters. This represents the ***computational graph level***.

Nodes and in general anything that is useful to organize together, are organized into packages, which are the smallest unit that is possible to build in ROS, fulfilling a defined function and defined by their own configurational files. This is the base of the ***filesystem level***. Packages can be grouped into stacks, which can be built as packages in a nested framework. Furthermore, within packages can be defined message structures.

ROS features are accessible via code through ***client libraries***, allowing to implement nodes in different coding languages. Actually, the most supported are C++ (roscpp) and Python (rospy), alongside LISP (roslisp) and Matlab/Simulink.

The ***community level*** is the essence of ROS. Packages and stacks can be shared across the ROS community through a federated system of online repositories.

Furthermore, a ***higher-level architecture*** is necessary to build larger systems on top of ROS.