CSCE 452/752 Fall 2024

4. Introduction to ROS

TEXAS A&M UNIVERSITY

AARRG

Robot Operating System

For the programming assignments in this class, we will use a software platform called “Robot Operating System,” or ROS.

The official description is:

> *The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications.*

Goal for today: Understand why ROS exists and understand the basic concepts that ROS uses.

Why ROS exists

Distributed computation

Software for robots is often **distributed**.

- Sending commands from a human's desktop computer to a robot.
- Multiple robots cooperating with each other.
- Multiple computers performing separate but related tasks in a single robot.
- Software decomposed into many smaller, stand-alone parts.

ROS makes this easier by providing **message passing** mechanisms for moving data between processes.

Code reuse

Software development is painful if standard algorithms must be re-implemented for each new robot or operating environment.

ROS makes this easier by providing **usable implementations** of many robotics algorithms, including both reliable, commonly-used techniques, and leading-edge research results.

Equally important, because these algorithms interact with each other using the message passing system described above, it is easy to **combine many algorithms** in a single application.

Simulation

Robot programming is often much faster when the part of the initial testing can be done using a **simulator**, rather than a real robot.

ROS makes this easier by providing a consistent way for simulators and robots to expose the **same APIs**.

The code never “knows” whether it is talking to a real robot or to a simulator.

Basic ROS Concepts

Some nouns

- A **package** is a collection of related executables and associated files.
- A **node** is an executing program that uses ROS to communicate with other nodes.
- A **topic** is a channel for one-way, many-to-many communication between nodes, from **publishers** to **subscribers**, based on sending **messages**.
- The **ROS graph** has edges for each publish-subscribe relationship.
- A **service** is a channel for two-way, one-to-one communication between a **client** and a **server**, based on a **request** and a **response**.
- An **action** is a method for asking a node to complete a time-extended task, based on a **request** (or **goal**) which leads to a **result**, along with **feedback** along the way.
- An **interface** is a data type used for a topic, service, or action.

But wait, there's more!

- A **launch file** is a Python program or XML file used to launch and configure several nodes at once.
- A **parameter** is a bit of configuration data associated with a node.
- A **bag** is a file that store messages to be replayed later.

Getting more information

ROS versions

Major versions:

- ROS1: 2007–2025
- **ROS2**: 2015–present

Distributions:

- Ardent Apalone
- Bouncy Bolson
- Crystal Clemmys
- Dashing Diademata
- Eloquent Elusor
- Galactic Geochelone
- Foxy Fitzroy
- **Humble Hawksbill**
- Iron Irwini

Documentation

https://docs.ros.org/

Documentation

Some relevant pages for Project 1:

- Tutorials
- Installation
- Configuring environment
- Introducing Turtlesim
- Understanding ROS2 Nodes
- Understanding ROS2 Topics
- Understanding ROS2 Services
- Writing a Simple Publisher and Subscriber
- Creating a Workspace
- Creating a Package
- Using `colcon` to build packages