Introduction to ROS with Python

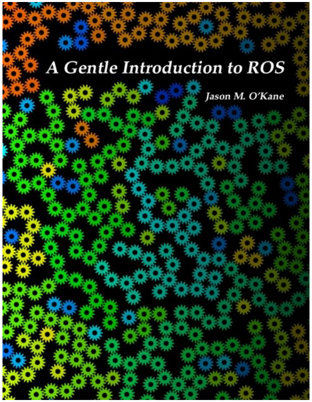
Evan Krell

Texas A&M University - Corpus Christi

November 2018

Outline

Sources



Jason M. O'Kane
`cse.sc.edu/~jokane/agitr`

Structure Python-based ROS Package
Simon Birrel
`artificialhumancompanions.com`

Introduction

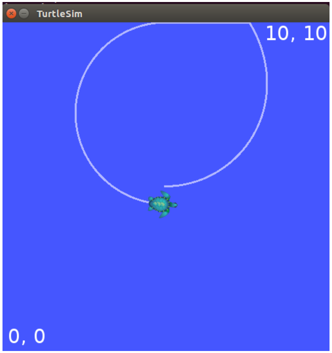
Package for this Tutorial

The content presented is demonstrated in an ROS package written in Python.

Location

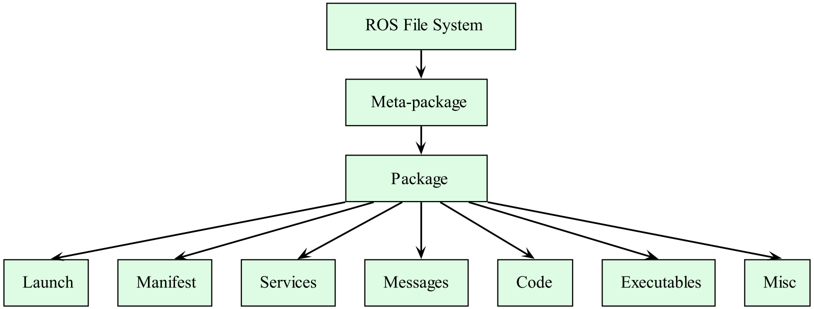
`https://github.com/ekrell/ros_python_workshop`

Turtlesim Environment



Getting Started

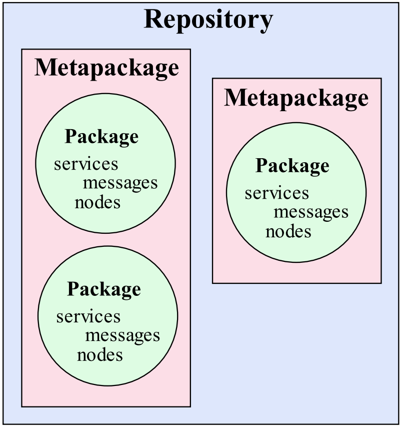
Packages



Getting Started

Packages

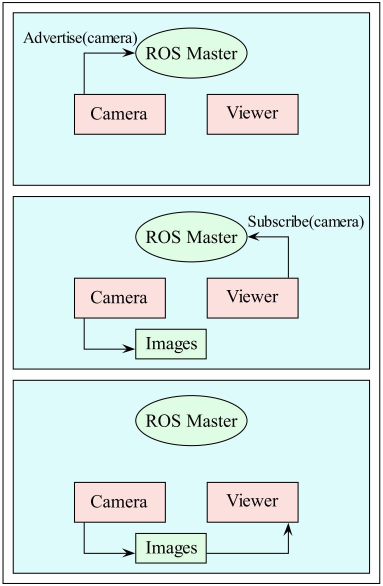
- Collection of files that fulfill single purpose (code, executables, etc)
- Simply a directory with **manifest** file called `package.xml`
- **Manifest** file has package definition, with name, version, dependencies
- Facilitates organization, sharing



Getting Started

ROS Master

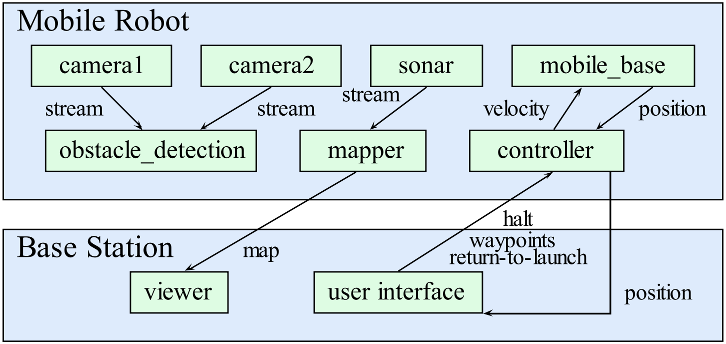
- Maintains directory of nodes, messages, services, parameters, etc
- Enables communication among nodes
- **Parameter server**: directory of parameters and values



Getting Started

Nodes

- Single executable using ROS
- Communicate over **topics** (publish, subscribe)



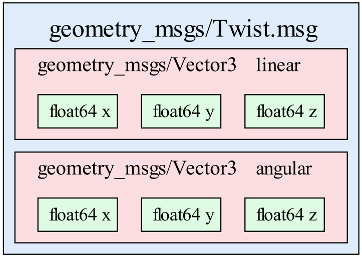
Getting Started

Topics

- Named buses for node communication
- Each has a specific message **type**
- Types are integers, floats, strings, and composite structures

Messages

- Units of communication
- Each message is of a specific **type**



Writing ROS Programs

Install ROS

Installation guide: `wiki.ros.org/ROS/Installation`

Setup Catkin

catkin: build system for ROS `/wiki.ros.org/catkin`

ROS for Python

rospy: ROS Python library `wiki.ros.org/rospy/Tutorials`

Writing ROS Programs

Install ROS Package

```
cd ~/catkin_ws/src
git clone https://github.com/ekrell/ros_python_workshop.git
cd ~/catkin_ws
catkin_make
```

Execute ROS package

```
roscore
rosrun PACKAGE_NAME SCRIPT.py
```

Log Messages

ROS Logging

View log in console: `rostopic echo /rosout`

View log in GUI: `rqt_console`

Log Message Severity

Debug: `rospy.logdebug(msg, *args)`	Lowest severity
Warn: `rospy.logwarn(msg, *args)`	
Info: `rospy.loginfo(msg, *args)`	...
Error: `rospy.logerr(msg, *args)`	
Fatal: `rospy.logfatal(msg, *args)`	Highest severity

Log Messages

Python Example

```
rospy.loginfo_throttle(10, status2str(pose, params["goal"]))
```

Result

```
rostopic echo /rosout
```

```
    level: 2
    name: "/purepursuit"
    msg: "Position:␣(x:5.5,␣y:5.5,␣theta:0.0),␣Goal:␣(x:9,␣y:9)"
```

Graph Resources

Naming Scheme

- ROS organizes nodes, topics, services, parameters in graph
- Thus, elements are called graph resources
- Flexible naming scheme for referencing these resources
- Facilitates modularity and existence of duplicate executions of same node
- But can be difficult to find where resources come from at first

$$\underset{\text{current namespace}}{\text{/turtle1}} \quad + \quad \underset{\text{relative name}}{\text{cmd\_vel}} \quad \text{=>} \quad \underset{\text{global name}}{\text{/turtle1/cmd\_vel}}$$

Launch Files

Launch Multiple Nodes

- Launch files setup and run multiple nodes
- Relieves burden of opening multiple terminals, executing each node in order, remembering all parameters, etc
- **Modular:** launch files can call launch files
- `roslaunch ros_python_workshop ros_python_workshop`
- **Ctrl-C** will (ideally) gracefully shut down each node

Example

`ros_python_workshop/launch/ros_python_workshop.launch`

`rosnode list`

```
    /purepursuit
    /rosout
    /turtlesim_node
```

Parameters

Parameter Server

- Handled within ROS Master
- Dictionary shared by nodes
- Setting & getting inside and outside node
- Just **strings**, not ROS message types
- **Caution:** Node must manually check for param changes

Console Basics

```
rosparam list

rosparam get goal

rosparam get roslaunch

rosparam set goal "[9, 9]"

rosparam dump testdump.txt roslaunch

rosparam load testdump.txt roslaunch
```

Services

Recording & Replaying Messages

Image Sources

Slide 5	Mastering ROS for Robotics Programming
Slide 7	`wiki.ros.org/Master`
Slide 8	ASV C-Worker USV