Crash Course on ROS

11/10/2017

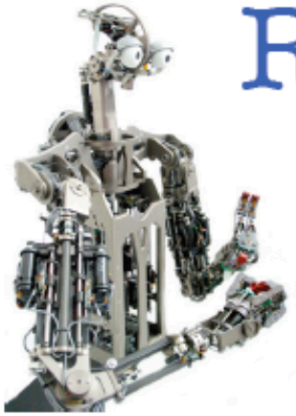
Agenda

- What is ROS?
- ROS Key Concepts
- Installation
- ROS Terminal Commands
  - turtlesim demo
  - Publish/Subscribe from Terminal
- First ROS Package
  - Publish/Subscribe from Python script
- UR5 with ROS

What is ROS?

- Robot Operating System
- A common language in terms of
  - communication
  - variable definitions
  - simulations
- 2007 - Stanford AI Lab and Willow Garage
- 2013 - Open Source Robotics Foundation

What is ROS?



ROS

navigation

task executive

visualization

simulation

perception

control

planning

data logging

message passing

device drivers

real-time capabilities

OS

web browser

email client

window manager

memory management

process management

scheduler

device drivers

file system

ROS Philosophy

- **Peer to Peer**
  - ROS systems consist of numerous small computer programs which connect to each other and continuously exchange *messages*
- **Tools-based**
  - There are many small, generic programs that perform tasks such as visualization, logging, plotting data streams, etc.
- **Multi-Lingual**
  - ROS software modules can be written in any language for which a *client library has been written.* Currently client libraries exist for C++, Python, LISP, Java, JavaScript, MATLAB, Ruby, and more.
- **Thin**
  - The ROS conventions encourage contributors to create stand-alone libraries and then *wrap those libraries so they send and receive messages to/from other ROS modules.*
- **Free and open source**

ROS Key Concepts

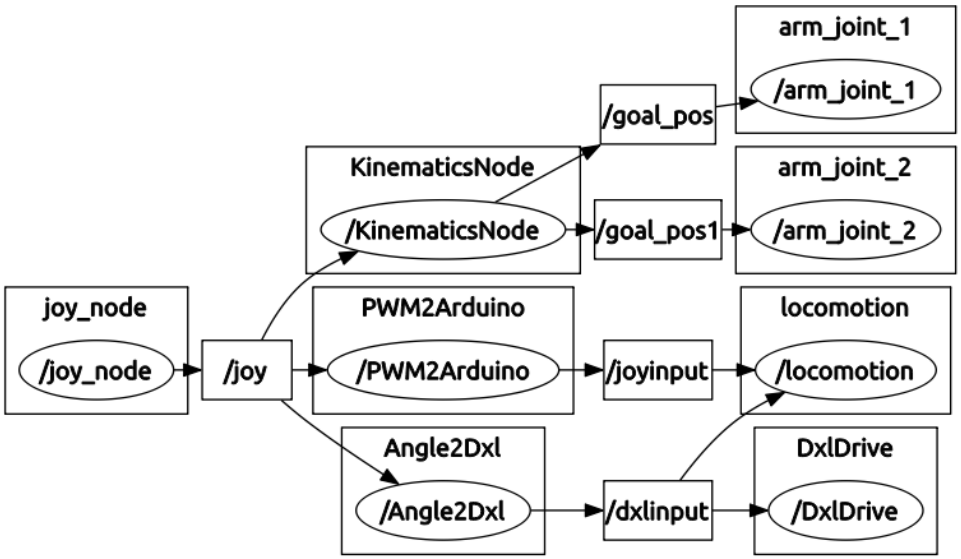
- Messages and Topics
- Nodes
- Services
- ROS Master
- Parameters
- Packages

ROS Messages and Topics

- Each data which is passed to ROS network is a **Message**.
- Messages have certain types to reach a common standard.
  - std_msgs/String or geometry_msgs/Point
- Each message is broadcasted under a **Topic**.

ROS Nodes

- Each executable program that connects to ROS network is a **Node**.
- Nodes can subscribe or publish to a **Topic**.
- Nodes can also provide or use **Services**.



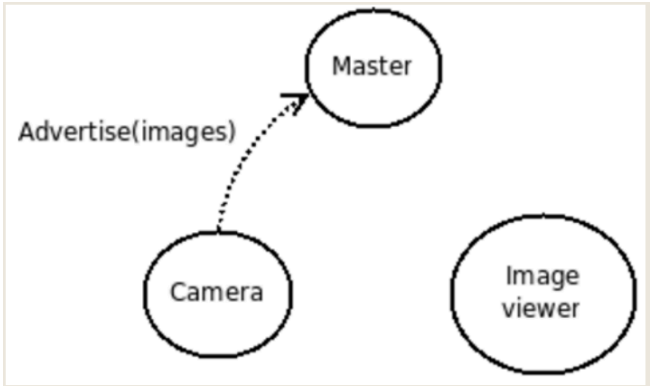
ROS Services

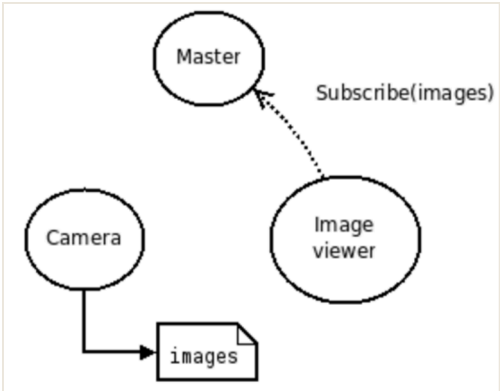
- Request and response structure.
- 1-to-1 communication
- Services don’t continuously publish messages like Nodes.

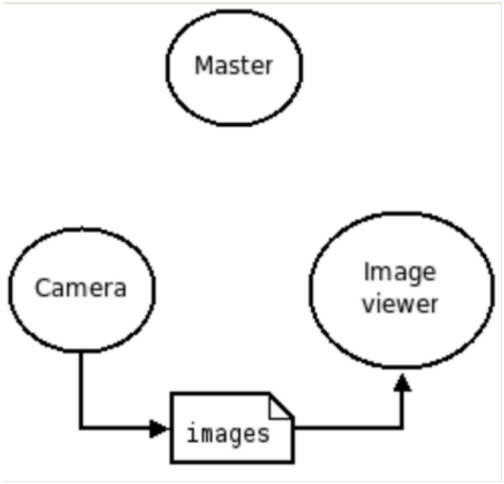
- You can compute remotely.
  - add2ints a b
- Taking a picture from rover’s control PC.

ROS Master

- The admin that makes sure everything is by the book!

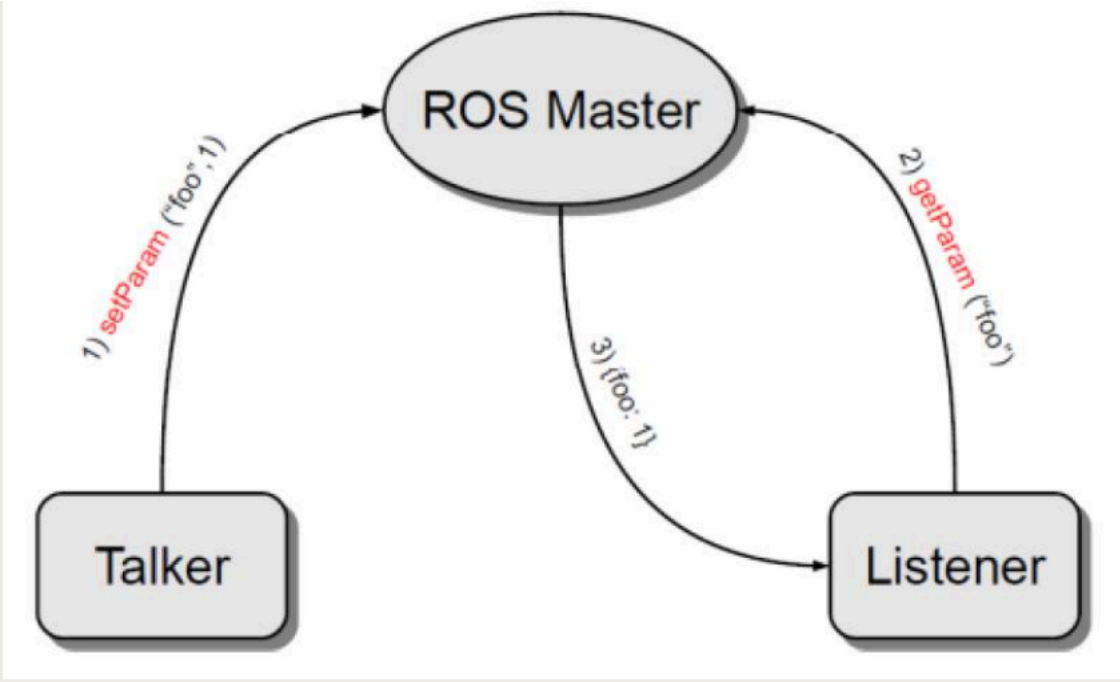






ROS Parameters

- A value dictionary stored on **ROS Master**.
- Not for dynamic operations, can be used for configuration of newly started **Node**.



ROS Packages

- Software in ROS is organized in packages.
- Multiple **Nodes** can be present in a **ROS Package**.
- Packages can be developed by user or can be downloaded online.

Installation

- ROS has LTS concept as Ubuntu.
  - Current LTS version is Kinetic Kame.
  - Kinetic Kame works on Ubuntu 15.10 and 16.04.

- http://wiki.ros.org/kinetic/Installation/Ubuntu

ROS Terminal Commands

- roscore
- rosrun
- roslaunch
- rosnode
- rostopic

roscore

- **roscore** starts the **ROS Master**.
- It should be the first thing you do while bringing up the system.

  – roscore

rosrun

- **rosrun** starts a node in a package.
  - rosrun <package> <executable>
  - rosrun turtlesim turtlesim_node

roslaunch

- Group of **rosrun**s that bring up systems with one command.

rosnode

Displays debugging information about ROS nodes, including publications, subscriptions and connections

Command	
$rosnode list	List active nodes
$rosnode ping	Test connectivity to node
$rosnode info	Print information about a node
$rosnode kill	Kill a running node
$rosnode machine	List nodes running on a particular machine

rostopic

Gives information about a topic and allows to publish messages on a topic

Command	
$rostopic list	List active topics
$rosnode echo /topic	Prints messages of the topic to the screen
$rostopic info /topic	Print information about a topic
$rostopic type /topic	Prints the type of messages the topic publishes
$rostopic pub /topic type args	Publishes data to a topic

rostopic

- rostopic list

- rostopic type /turtle1/cmd_vel
  - geometry_msgs/Twist

- rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist '{linear: {x: 0.2, y: 0, z: 0}, angular: {x: 0, y: 0, z: 0}}'

First ROS Package

- You should have built a workspace.

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make

$ source devel/setup.bash
```

catkin_create_pkg <package_name> <dependency1> <dependency2>

catkin_create_pkg ozu_biomech_chat std_msgs rospy roscpp

cd ~/catkin_ws/
catkin_make -> this builds the packages

Publish/Subscribe from Python script

- We need to form some scripts under
  - ~/catkin_ws/src/package_name/src
  - http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29

- If you develop in C++, you need to compile with **catkin_make**.

- rosrun ozu_biomech_chat talker.py
- rosrun ozu_biomech_chat listener.py

UR5 with ROS

- sudo apt-get install ros-kinetic-universal-robot

- https://github.com/ThomasTimm/ur_modern_driver

roslaunch ur_modern_driver urXX_bringup.launch robot_ip:=ROBOT_IP_ADDRESS

rosrun ur_modern_driver test.py

ur_modern_driver package will not compile as it is. You need to update ur_hardware_interface.cpp using the link below.

https://github.com/iron-ox/ur_modern_driver/commit/883070d0b6c0c32b78bb1ca7155b8f3a1ead416c