ROS Bootcamp - Day 1

Introduction To ROS1

JC Cruz and Alec Graves

UTSA RAS

May 15, 2023

Scheduling

- 4 lectures over 4 days
  - Dates: May 15 - May 18
  - Time: 6:30PM - 7:45PM
  - Location: EB 2.04.23
  - Modality: In-person (Possible live streaming)
- 30 min lecture with slides
- 45 min guided activity

Day-by-Day Breakdown

- Day 1:
  - ROS architecture & philosophy
  - ROS master, nodes, and topics
  - Console commands
  - Catkin workspace and build system
  - Launch-files
  - Gazebo simulator
  - **Activity:** ROS Basics, Build a package, launch files
- Day 2:
  - ROS package structure
  - Integration and programming
  - ROS subscribers and publishers
  - ROS parameter server
  - RViz visualization
  - **Activity:** IMU Visualization with ESP32

Day-by-Day Breakdown (cont.)

- Day 3:
  - ROS services
  - ROS actions (actionlib)
  - ROS time
  - ROS bags
  - **Activity:** Custom Package, Autonomous Robot Sim
- Day 4:
  - TF Transformation System
  - rqt User Interface
  - Robot models (URDF)
  - Simulation descriptions (SDF)
  - ROS Control
  - Intro to ROS2
  - **Activity:** Services, Bag file, RQT Plot

Today's Topics

- ROS architecture & philosophy
- ROS master, nodes, and topics
- Console commands
- Catkin workspace and build system
- Launch-files
- Gazebo simulator
- **Activity:** ROS Basics, Build a package, launch files

ROS History

- Developed in 2007 at Stanford AI Lab.
- Managed by OSRF since 2013.
- Widely used by robots, universities, companies.
- De facto standard for robot programming.
- More details: ros.org.

ROS Philosophy

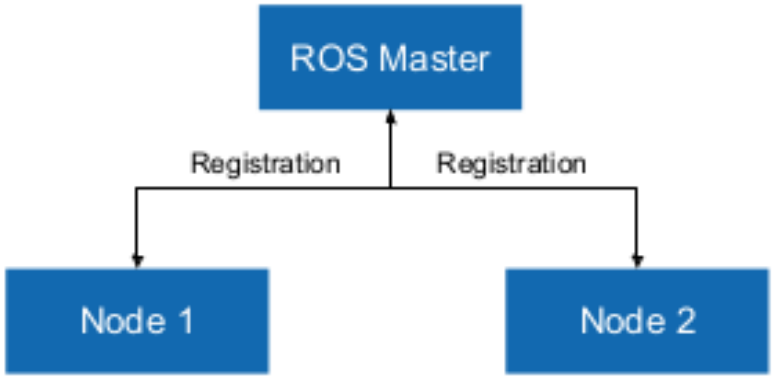
- **Peer-to-peer:** Programs communicate via defined APIs.
- **Distributed:** Run programs on multiple computers.
- **Multi-lingual:** Supports C++, Python, MATLAB, Java, etc.
- **Light-weight:** Stand-alone libraries with a thin ROS layer.
- **Free and open-source:** Most software is open-source and free.

ROS Architecture

- **Nodes:** Individual programs running in the system
- **Messages:** Data structures for communication
- **Topics:** Communication channels for messages
- **Services:** Synchronous RPC-style communication
- **Actions:** Asynchronous RPC-style communication
- **Parameter Server:** Centralized storage for configuration data
- **ROS Master**: Name and registration service for nodes, topics, and services

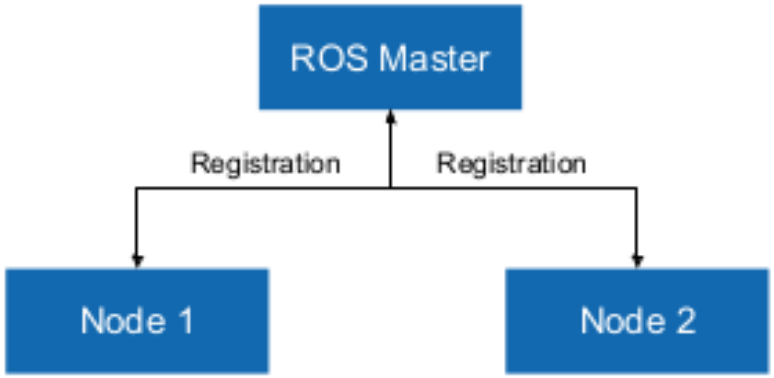
ROS Master

- Manages the registration of nodes, topics, and services
- Enables nodes to find and communicate with each other
- Start the ROS master with:
  `roscore`



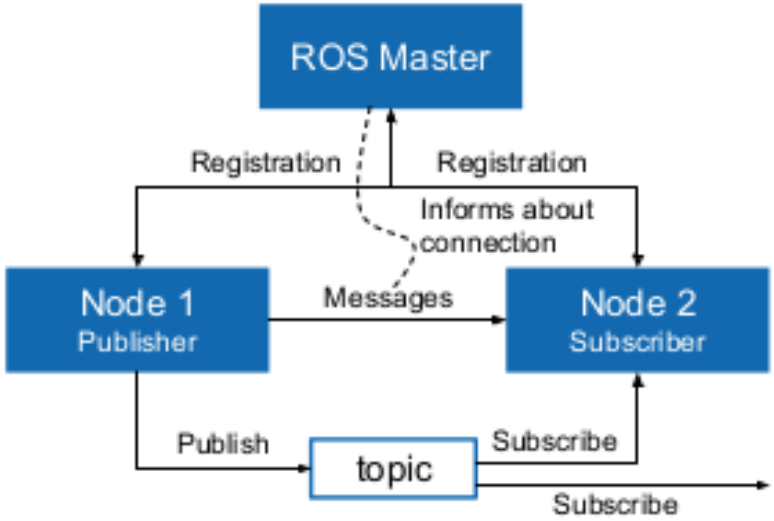
ROS Nodes

- Individual programs that perform a specific task
- Can communicate with other nodes through topics, services, and actions
- Run a node with: `rosrun package_name node_name`



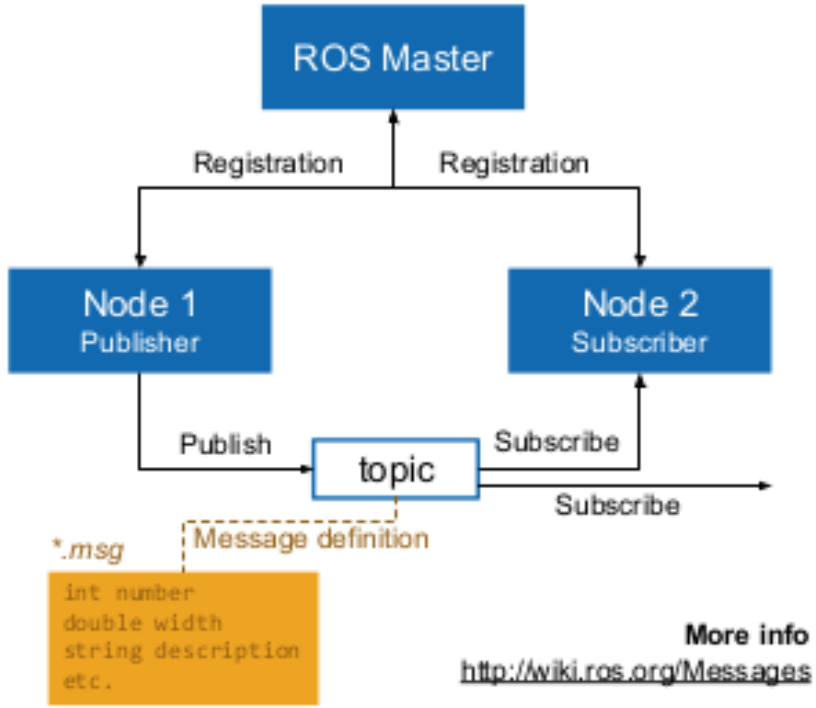
ROS Topics

- Nodes communicate using message-based publish-subscribe pattern
- Nodes publish messages to topics, and nodes subscribe to topics to receive messages
- List all active topics with: `rostopic list`



ROS Messages

- Each message has a type that determines the structure of the data
- Message types can be simple like integers or strings, or complex types defined by other messages
- See the type of a topic with: `rostopic type /topic_name`



ROS Message Example

geometry_msgs/Point.msg

```
float64 x
float64 y
float64 z
```

sensor_msgs/Image.msg

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

geometry_msgs/PoseStamped.msg

```
std_msgs/Header header
 uint32 seq
 time stamp
 string frame_id
geometry_msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
```

Figure: Pose Stamped Example

Console Commands

- `roscore`: Start ROS master
- `rosrun`: Run a ROS node
- `rostopic`: List and inspect topics
- `rosservice`: Call and inspect services
- `roslaunch`: Start multiple nodes and set parameters
- `roscd`: Change directory to a ROS package
- `rospack`: List and find ROS packages

Catkin Workspace

The catkin workspace contains the following spaces:

Work here

src

The *source space* contains the source code. This is where you can clone, create, and edit source code for the packages you want to build.

Don't touch

build

The *build space* is where CMake is invoked to build the packages in the source space. Cache information and other intermediate files are kept here.

Don't touch

devel

The *development (devel) space* is where built targets are placed (prior to being installed).

You can name workspace whatever you want. Will commonly see `catkin_ws`.

Build System

- Catkin is the default build system for ROS
- Builds packages and generates executables
- Build from top of your workspace with: `catkin_make`
- Source the workspace setup with: `source devel/setup.bash`

ROS Launch

- Tool for launching multiple nodes and setting parameters
- Written in XML as *.launch files
- Start a launch file with: `roslaunch package_name file_name.launch`
- More info: `http://wiki.ros.org/roslaunch`

Figure: Launch File Example

```
<?xml version="1.0"?>
<launch>
<arg name="use_sim_time" default="true"/>
<arg name="world" default="
     gazebo_ros_range"/>
<arg name="debug" default="false"/>
<arg name="physics" default="ode"/>
<group if="$(arg use_sim_time)">
    <param name="/use_sim_time" value="
     true" />
</group>
<include file="$(find gazebo_ros)
    /launch/empty_world.launch">
    <arg name="world_name" value="$(find
     gazebo_plugins)/
    test/test_worlds/$(arg world).world"/
     >
    <arg name="debug" value="$(arg debug)
     "/>
    <arg name="physics" value="$(arg
     physics)"/>
</include>
</launch>
```

ROS Launch Arguments

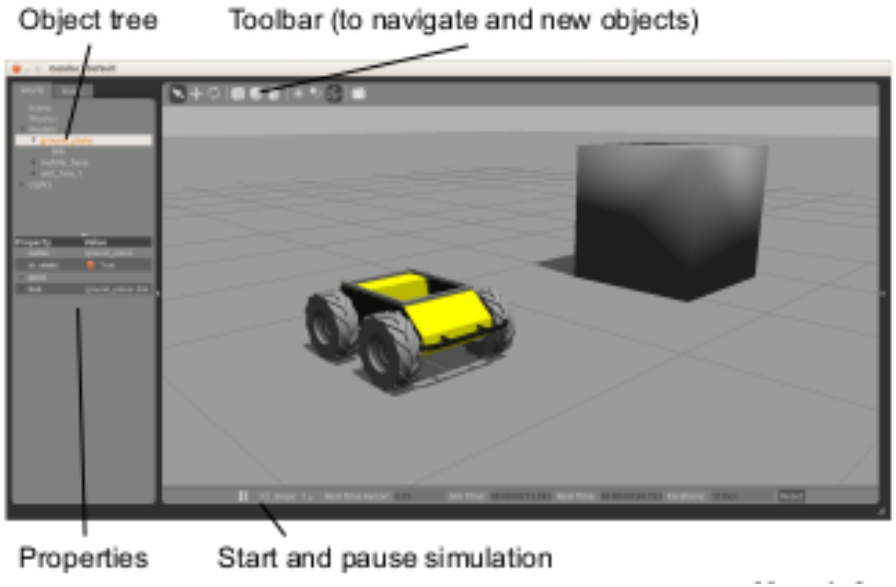
- Create reusable launch files with `<arg>` tag
- Use arguments in launch file with `$(arg arg_name)`
- Set arguments when launching with:
  `roslaunch launch_file.launch arg_name:=value`
- More info: `http://wiki.ros.org/roslaunch/XML/arg`

Including Other Launch Files

- Organize large projects by including other launch files with `<include>` tag
- Find system path to other packages with `$(find package_name)`
- Pass arguments to the included file with `<arg name="arg_name" value="value"/>`
- More info: `http://wiki.ros.org/roslaunch/XML/include`

Gazebo Simulator

- Simulates 3D rigid-body dynamics, sensors, and user interaction
- Provides a ROS interface and is extensible with plugins
- Run Gazebo with: `rosrun gazebo_ros gazebo`
- More info: `http://gazebosim.org/`



Activity

Find Today's activity sheet at: \docs\lec1

`https://github.com/UTSARobotics/ros1_bootcamp`