

ROS - Basics of Nodes and Topics

ME 4140 - Introduction to Robotics - Fall 2019

I. Components

The major building blocks of a ROS system

1. Master Node

- *The ROS Master provides naming and registration services to the rest of the nodes in the ROS system.***
- master node runs first `$ roscore`
- core of the system or robot `$ ROS_MASTER_URI=http://12345`

2. Nodes

- *A node is a process that performs computation.***
 - each 'program' or 'element' of the robot is a node
- examples:
- sensor
 - navigation
 - hardware driver
 - keyboard or joystick
- start or run node individually after master
- ```
$ rosrn <packagename> <nodename> <options>
```
- all nodes are registered to the master and communicate in different ways
    - **topics** - publishing and subscribing
    - **parameter server** - static data
    - **services** - subroutine call

### 3. Packages

- *Software in ROS is organized in packages. A package might contain ROS nodes, a ROS-independent library, a dataset, configuration files, a third-party piece of software, or anything else that logically constitutes a useful module.\*\**
- a collection of related nodes, each node belongs to a package
- pre-built packages available with ros installation      `-desktop-full`
- pre-built packages available for installation

```
apt-get $ sudo apt-get install ros-<distribution>-<packagename>
rosdep $ rosdep install <packagename>
```

- update ubuntu before installing anything

```
$ sudo apt-get update
$ sudo apt-get check
```

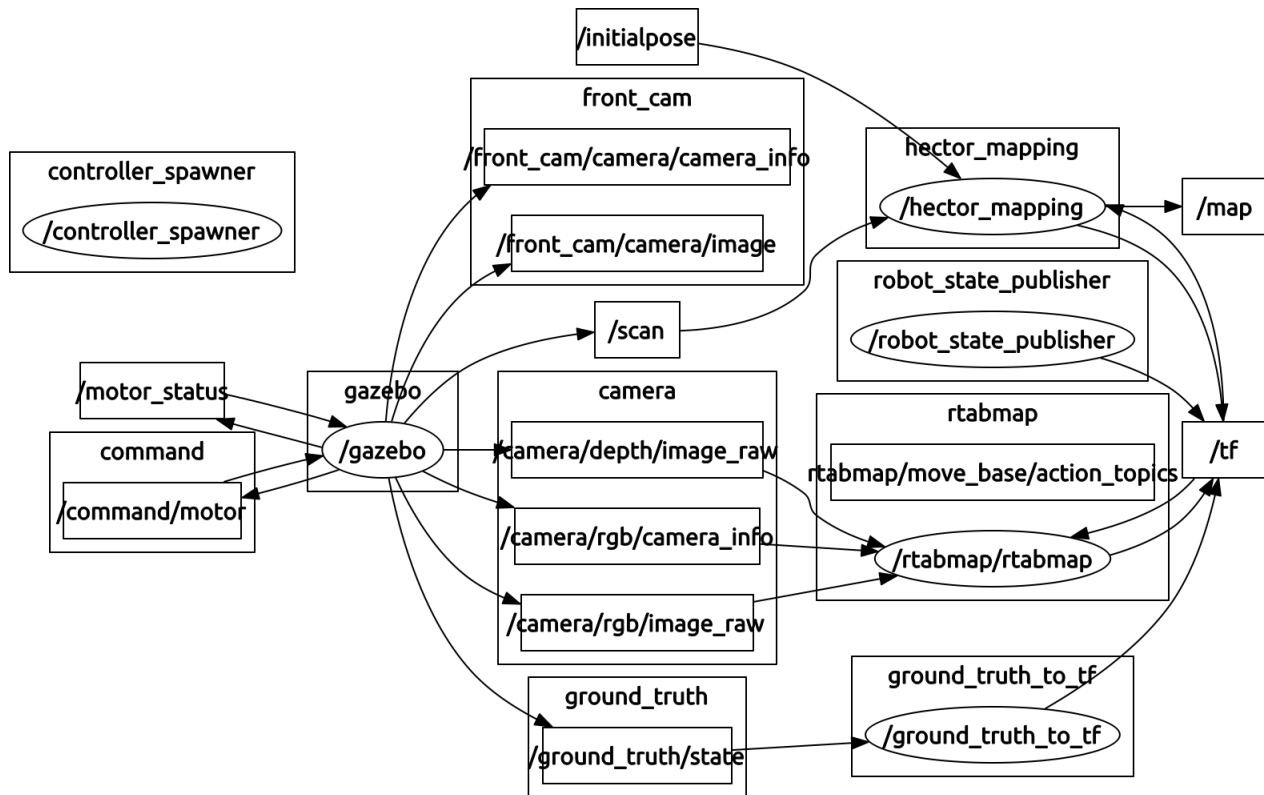
\*\* from (ros.org)

## II. The Graph of the System

ROS works on a system of interconnected nodes. It is very useful to visualize this in a graph.

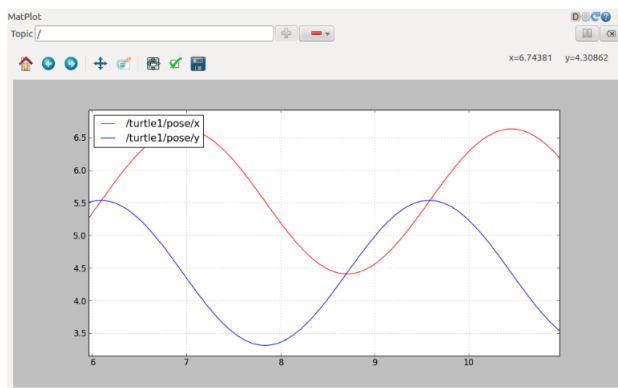
1. [RQT Graph](#) A very useful tool. A node `rqt_graph` in a package `rqt_graph`.

```
$ rosrun rqt_graph rqt_graph
```



2. [RQT Plot](#) A very useful tool. A node `rqt_plot` in a package `rqt_plot`.

```
$ rosrun rqt_plot rqt_plot
```



### III. Topics, Publishers, and Subscribers

The nodes in a ROS system communicate.

#### 1. Topics

- data available to nodes in the system
- each topic has a name
- data is stored and transferred in standard ros data types
- generally data is streaming, but does not have to be

#### 2. Publishers

- data produced by a node can be shared with the system by publishing a topic
- a node which outputs topic data is a publisher
- a node may publish multiple topics

#### 3. Subscribers

- a registered node can access the data in a topic by subscribing to a topic
- a node which gets topic data as input is a subscriber
- a node may subscribe to multiple topics

#### 4. rostopic

- a very useful tool, a built in package
- used differently than other packages, does not require rosrn
- a set of different tools

**list**      \$ rostopic list

**echo**      \$ rostopic echo /topicname

**type**      \$ rostopic pub /topicname

#### 5. data types - topics are published in standard types called messages

- std\_msgs/int32
- std\_msgs/float32
- geometry\_msgs/Point
- geometry\_msgs/Pose
- nav\_msgs/Odometry
- nav\_msgs/Path

#### 6. let show an example now!

#### IV. A Simple Robot Simulator :

1. Update and Install [turtlesim](#).

```
sudo apt-get update
```

```
sudo apt-get install ros-melodic-turtlesim
```

2. Start the ROS core.

```
roscore
```

3. Start the turtlesim node in a new terminal window.

```
roslaunch turtlesim turtlesim_node
```

4. Now lets install and run a controller node.

```
sudo apt-get install ros-melodic-teleop-twist-keyboard
```

```
roslaunch teleop_twist_keyboard teleop_twist_keyboard.py
```

5. There is a problem, we need to make sure the nodes are talking. Add the following *option* to the end of the previous command and rerun the node.

```
cmd_vel:=turtle1/cmd_vel
```