October 2016

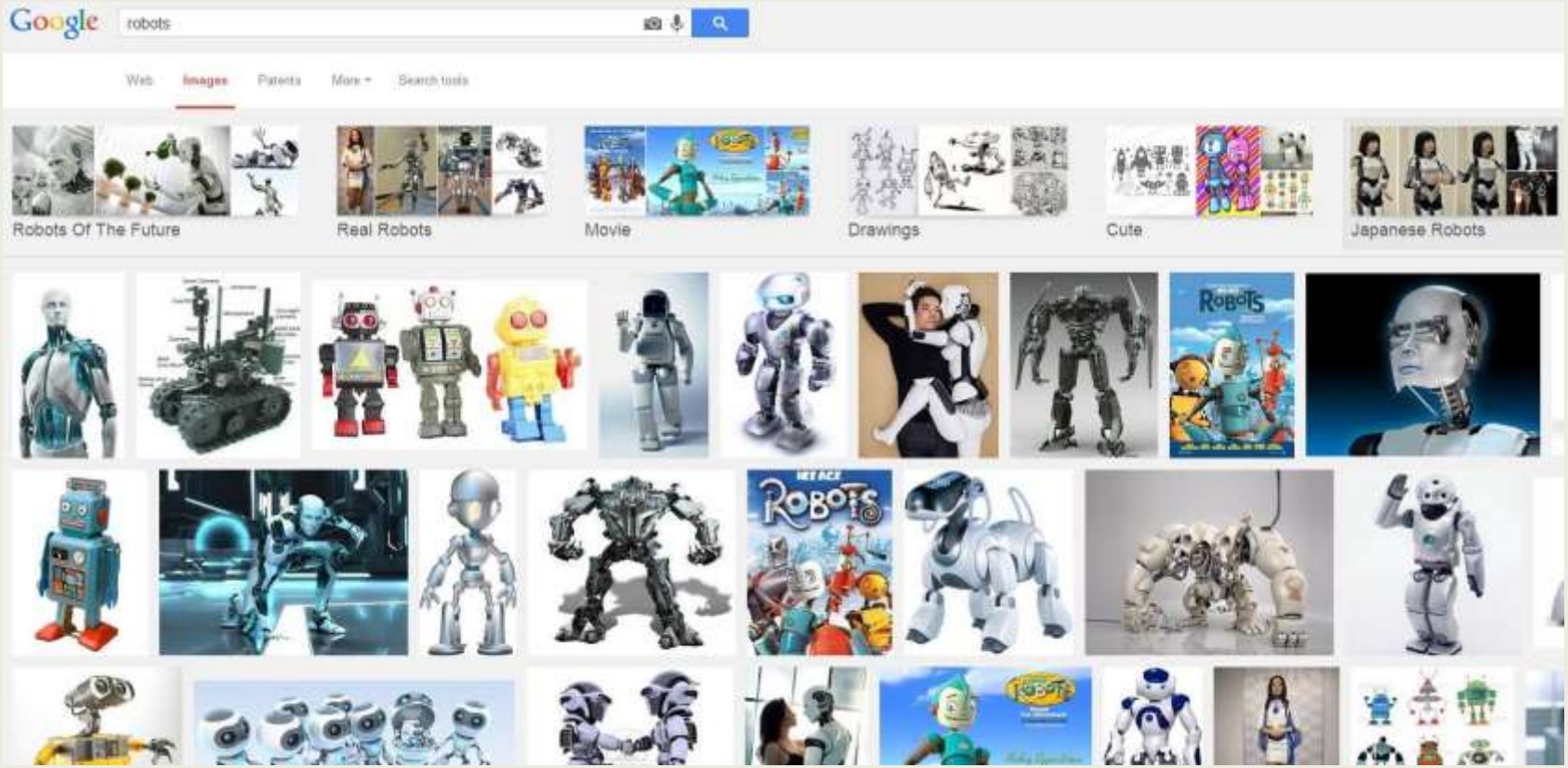
ROS - Lecture 1

ROS Introduction
Main concepts
Basic commands

BIRC
BIU Robotics Consortium

The Problem

- Lack of standards for robotics



What is ROS?

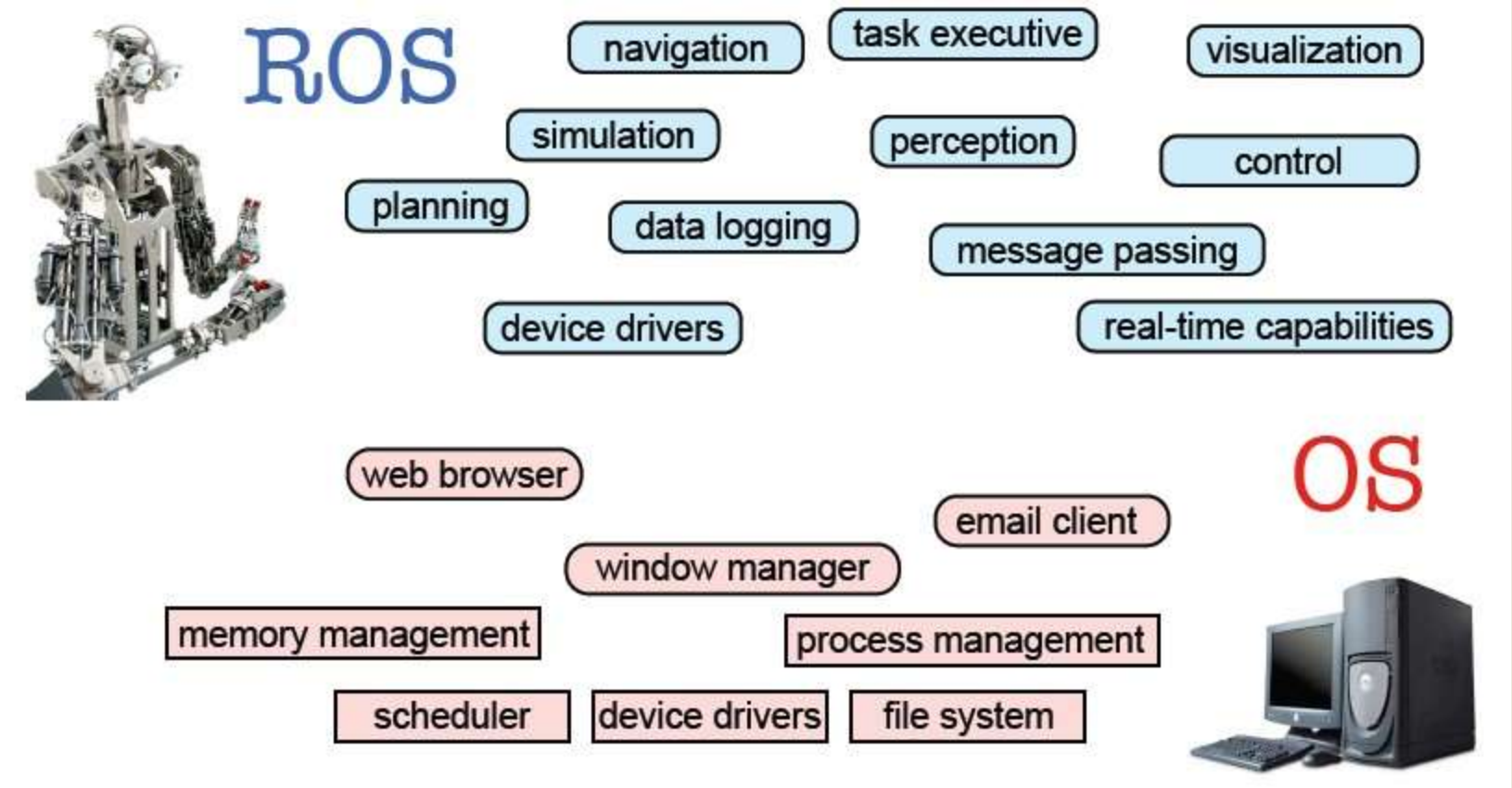
- ROS is an open-source **robot operating system**
- A set of software libraries and tools that help you build robot applications that work across a wide variety of robotic platforms
- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory and development continued at Willow Garage
- Since 2013 managed by OSRF (Open Source Robotics Foundation)

ROS Main Features

<u>ROS has two "sides"</u>

- The operating system side, which provides standard operating system services such as:
  - hardware abstraction
  - low-level device control
  - implementation of commonly used functionality
  - message-passing between processes
  - package management
- A suite of user contributed packages that implement common robot functionality such as SLAM, planning, perception, vision, manipulation, etc.

ROS Main Features



Taken from Sachin Chitta and Radu Rusu (Willow Garage)

ROS Philosophy

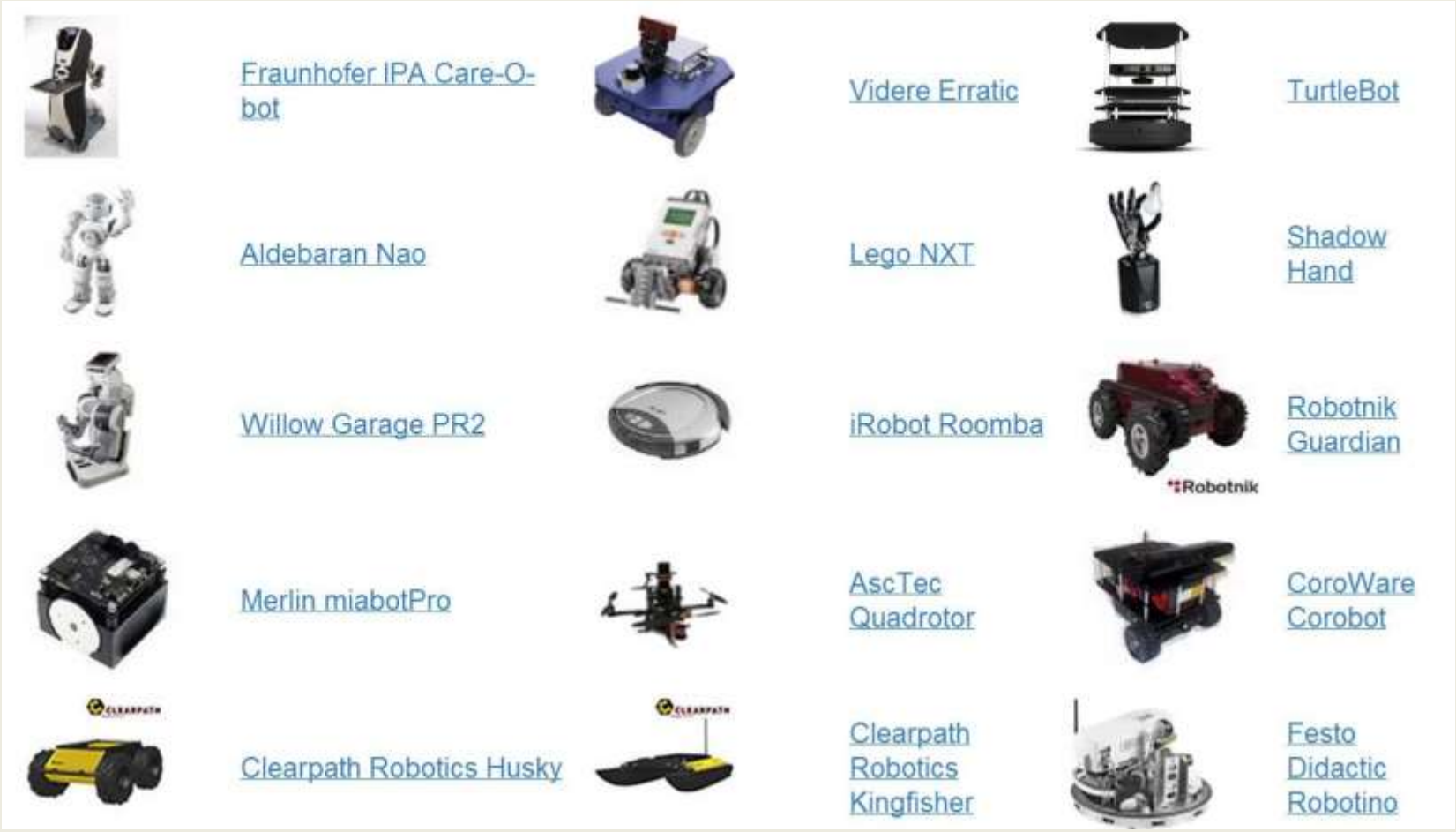
- **Peer to Peer**
  - ROS systems consist of numerous small computer programs which connect to each other and continuously exchange *messages*
- **Tools-based**
  - There are many small, generic programs that perform tasks such as visualization, logging, plotting data streams, etc.
- **Multi-Lingual**
  - 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 Wiki

- http://wiki.ros.org/
- Installation: http://wiki.ros.org/ROS/Installation
- Tutorials: http://wiki.ros.org/ROS/Tutorials
- ROS Tutorial Videos
  - http://www.youtube.com/playlist?list=PLDC89965A56E6A8D6
- ROS Cheat Sheet
  - http://www.tedusar.eu/files/summerschool2013/ROScheatsheet.pdf

Robots using ROS

http://wiki.ros.org/Robots



ROS Core Concepts

- Nodes
- Messages and Topics
- Services
- ROS Master
- Parameters
- Stacks and packages

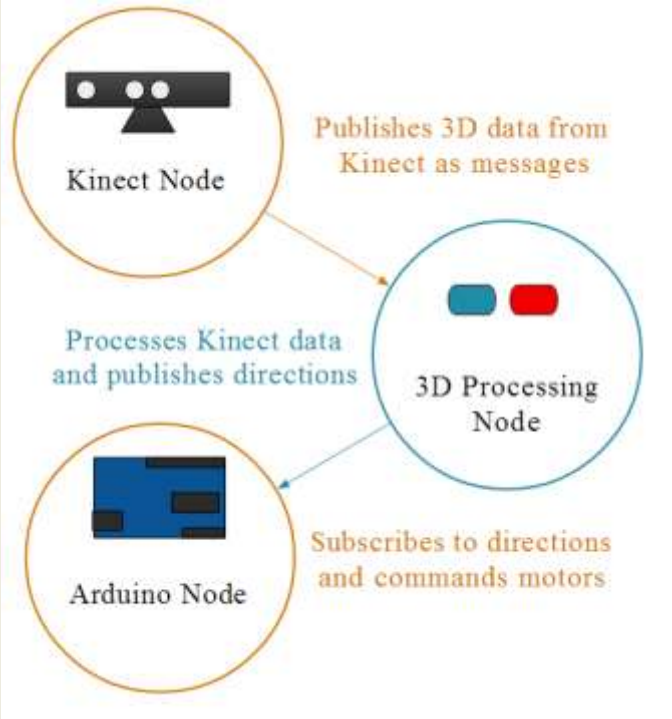
ROS Nodes

- Single-purposed executable programs
  - e.g. sensor driver(s), actuator driver(s), mapper, planner, UI, etc.
- Individually compiled, executed, and managed
- Nodes are written using a ROS **client library**
  - roscpp – C++ client library
  - rospy – python client library
- Nodes can publish or subscribe to a Topic
- Nodes can also provide or use a Service

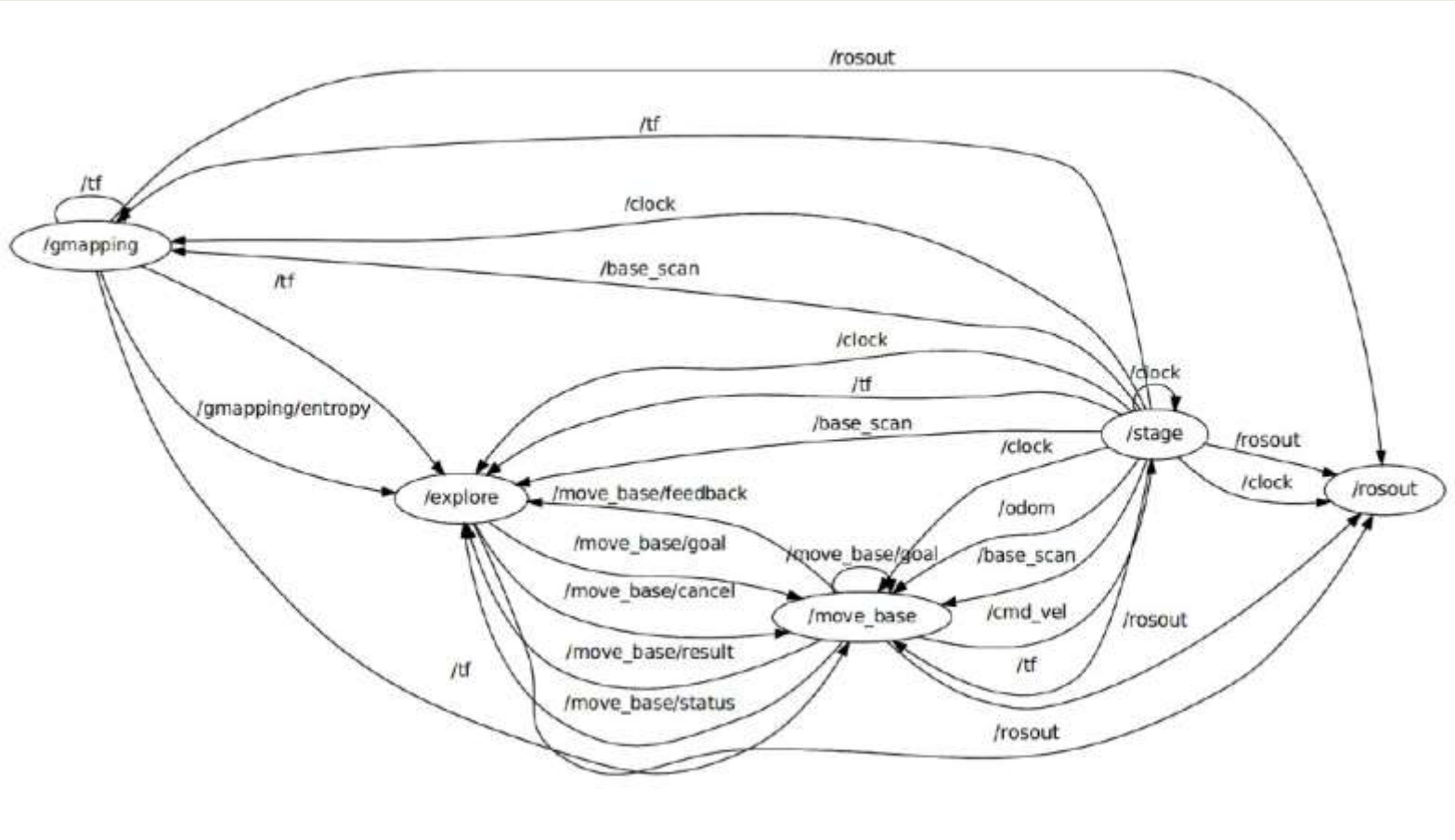
ROS Topics

- A topic is a name for a stream of messages with a defined type
  - e.g., data from a laser range-finder might be sent on a topic called scan, with a message type of LaserScan
- Nodes communicate with each other by publishing messages to topics
- Publish/Subscribe model: 1-to-N broadcasting

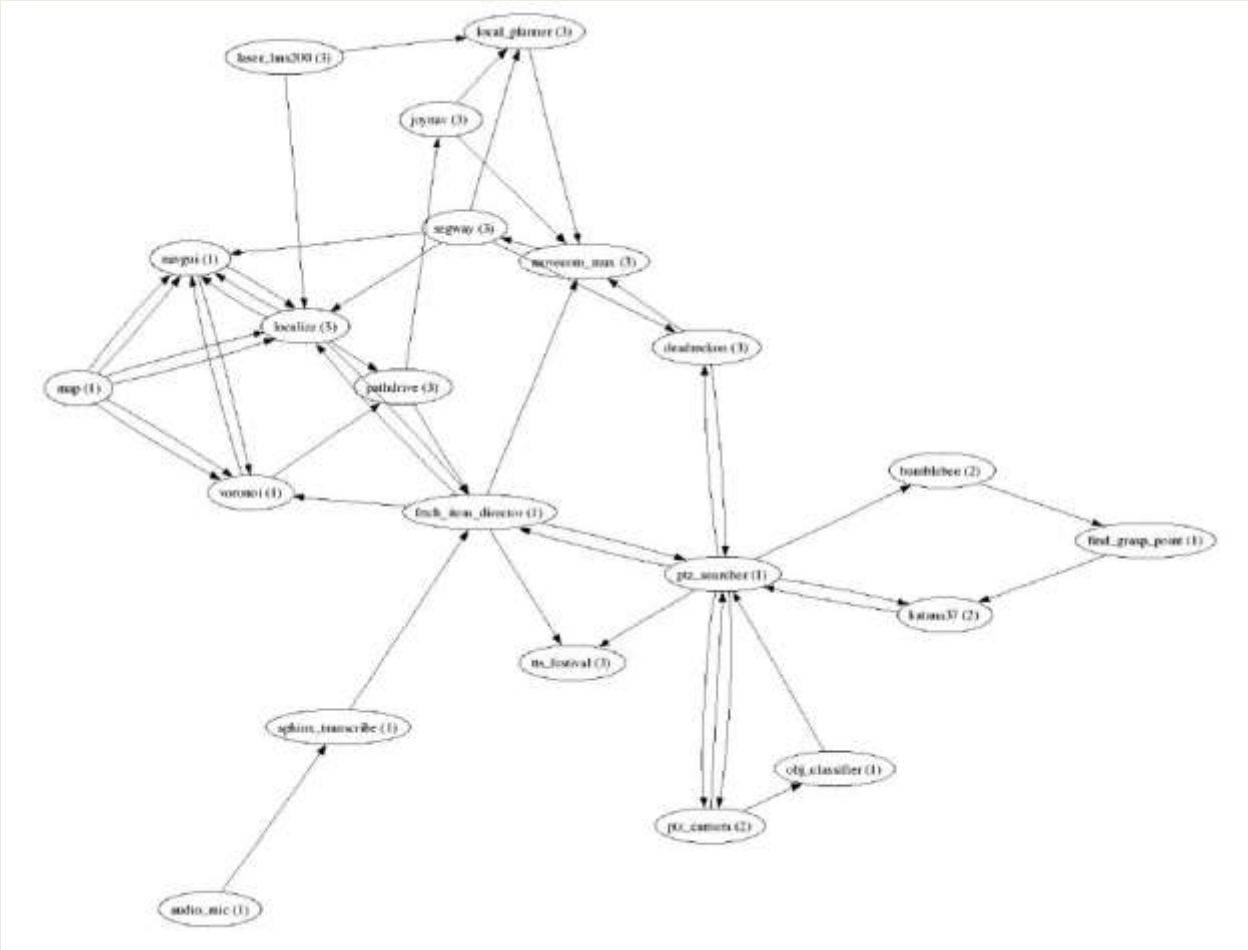
ROS Topics


Kinect Node
Publishes 3D data from Kinect as messages
Processes Kinect data and publishes directions
3D Processing Node
Arduino Node
Subscribes to directions and commands motors

The ROS Graph



Fetch an Item Graph



Taken from Programming Robots with ROS (Quigley et al.)

ROS Messages

- Strictly-typed data structures for inter-node communication
- For example, geometry_msgs/Twist is used to express velocity commands:

```
Vector3 linear
Vector3 angular
```

  – Vector3 is another message type composed of:

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

ROS Services

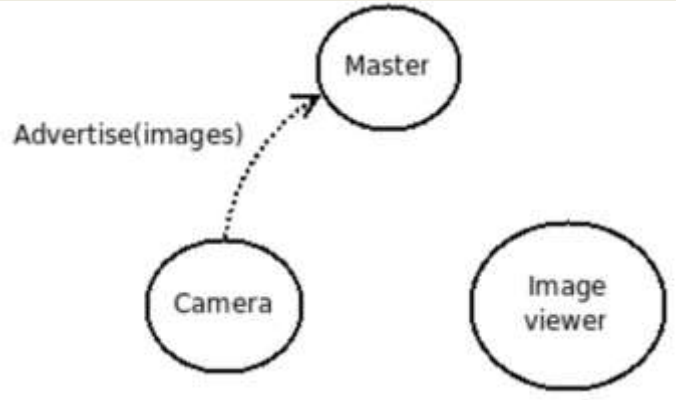
- Synchronous inter-node transactions / RPC
- Service/Client model: 1-to-1 request-response
- Service roles:
  - carry out remote computation
  - trigger functionality / behavior
- Example:
  - map_server/static_map – retrieves the current grid map used by the robot for navigation

ROS Master

- Provides connection information to nodes so that they can transmit messages to each other
  - Every node connects to a master at startup to register details of the message streams they publish, and the streams to which that they to subscribe
  - When a new node appears, the master provides it with the information that it needs to form a direct peer-to-peer connection with other nodes publishing and subscribing to the same message topics

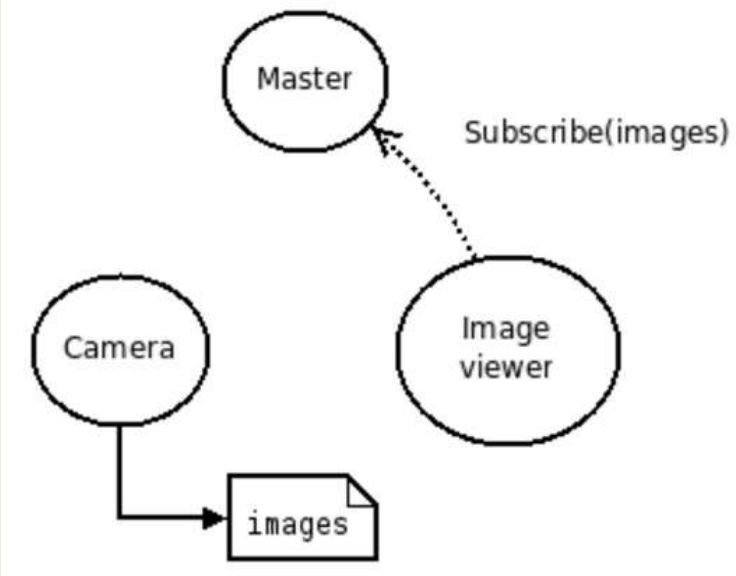
ROS Master

- Let’s say we have two nodes: a Camera node and an Image_viewer node
- Typically the camera node would start first notifying the master that it wants to publish images on the topic "images":



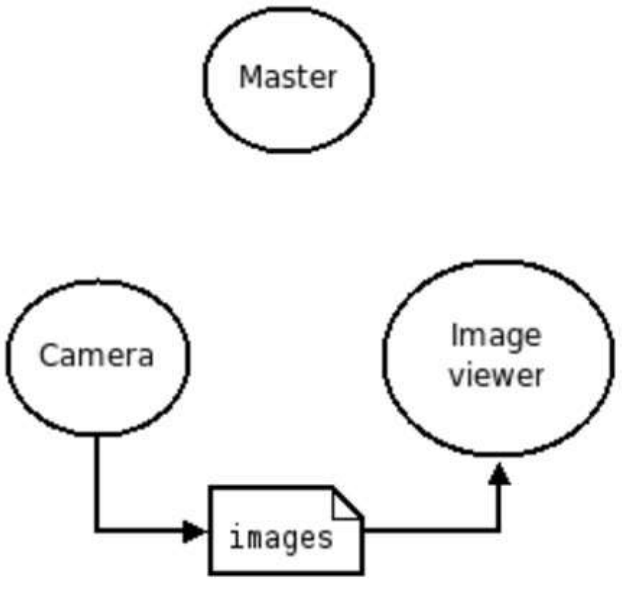
ROS Master

- Now, Image_viewer wants to subscribe to the topic "images" to see if there's maybe some images there:



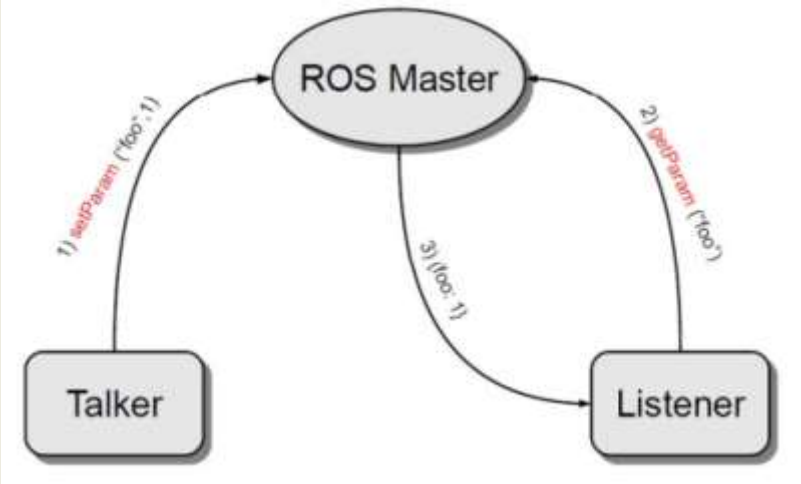
ROS Master

- Now that the topic "images" has both a publisher and a subscriber, the master node notifies Camera and Image_viewer about each others existence, so that they can start transferring images to one another:



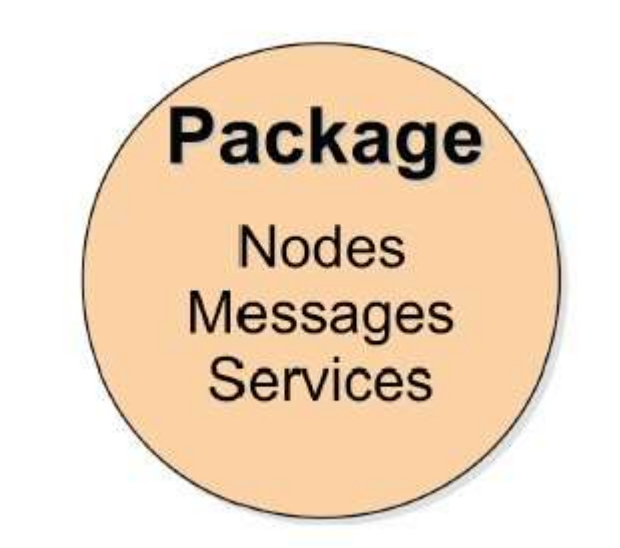
Parameter Server

- A shared, multi-variate dictionary that is accessible via network APIs
- Best used for static, non-binary data such as configuration parameters
- Runs inside the ROS master

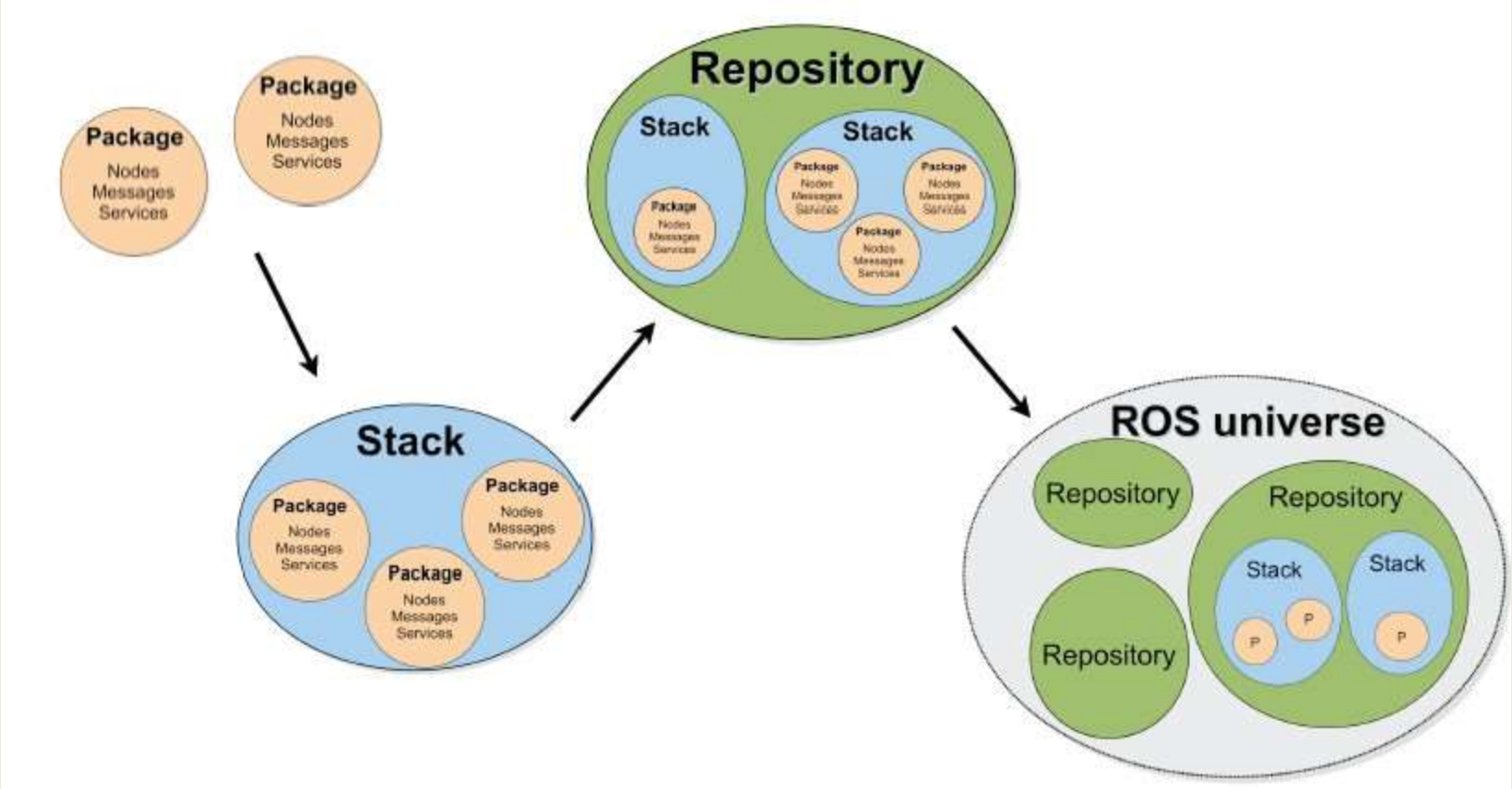


ROS Packages

- Software in ROS is organized in *packages*.
- A package contains one or more nodes and provides a ROS interface
- Most of ROS packages are hosted in GitHub



ROS Package System



Taken from Sachin Chitta and Radu Rusu (Willow Garage)

ROS Distribution Releases

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Kinetic Kame (**Recommended**)	May 23rd, 2016			May, 2021
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015

ROS Supported Platforms

- ROS is currently supported only on Ubuntu
  - other variants such as Windows and Mac OS X are considered experimental (will be supported on ROS 2.0)
- ROS distribution supported is limited to <=3 latest Ubuntu versions
- ROS Jade supports the following Ubuntu versions:
  - Vivid (15.04)
  - Utopic (14.04)
  - Trusty (14.04 LTS)
- ROS Indigo supports the following Ubuntu versions:
  - Trusty (14.04 LTS)
  - Saucy (13.10)

ROS Installation

- If you already have Ubuntu installed, follow the instructions at:
  - http://wiki.ros.org/indigo/Installation/Ubuntu
  - You can also download a VM with ROS Indigo Pre-installed from here:
  - http://nootrix.com/downloads/#RosVM
- Two VMs are available: one with Ubuntu 32Bits and the other with Ubuntu 64Bits (.ova files)
- You can import this file into VirtualBox or VMWare

ROS Environment

- ROS relies on the notion of combining spaces using the shell environment
  - This makes developing against different versions of ROS or against different sets of packages easier
- After you install ROS you will have setup.*sh files in '/opt/ros/<distro>/', and you could source them like so:

```
$ source /opt/ros/indigo/setup.bash
```

- You will need to run this command on every new shell you open to have access to the ros commands, unless you add this line to your bash startup file (~/.bashrc)
  - If you used the pre-installed VM it’s already done for you

ROS Basic Commands

- roscore
- rosrun
- rosnode
- rostopic

roscore

- roscore is the first thing you should run when using ROS

```
$ roscore
```

- roscore will start up:
  - a ROS Master
  - a ROS Parameter Server
  - a rosout logging node

roscore

```
roscore http://c3po:11311/
viki@c3po:~$ roscore
... logging to /home/viki/.ros/log/c54cfa00-5cfb-11e4-8e38-000c293f9c00/roslaunc
h-c3po-3511.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://c3po:55749/
ros_comm version 1.11.8


SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.8

NODES

auto-starting new master
process[master]: started with pid [3523]
ROS_MASTER_URI=http://c3po:11311/

setting /run_id to c54cfa00-5cfb-11e4-8e38-000c293f9c00
process[rosout-1]: started with pid [3536]
started core service [/rosout]
```

rosrun

- rosrun allows you to run a node
- Usage:

```
$ rosrun <package> <executable>
```

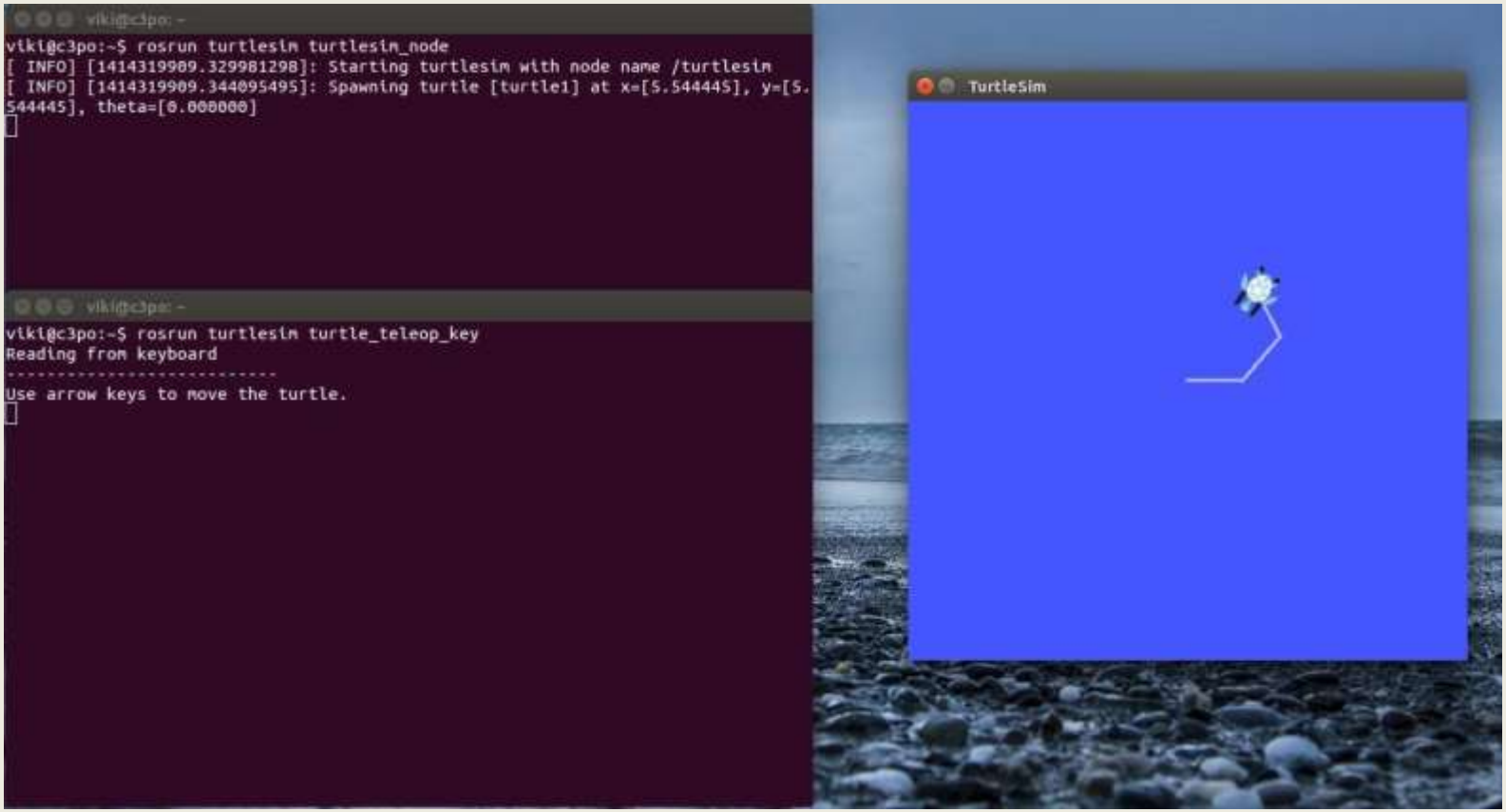
- Example:

```
$ rosrun turtlesim turtlesim_node
```

Demo - Turtlesim

- In separate terminal windows run:
  - roscore
  - rosrun turtlesim turtlesim_node
  - rosrun turtlesim turtle_teleop_key

Demo - Turtlesim



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

rosnode info

```
viki@c3po: ~
viki@c3po:~$ rosnode info turtlesim
--------------------------------------------------------------------------------
Node [/turtlesim]
Publications:
 * /turtle1/color_sensor [turtlesim/Color]
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Services:
 * /turtle1/teleport_absolute
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level
 * /reset
 * /spawn
 * /clear
 * /turtle1/set_pen
 * /turtle1/teleport_relative
 * /kill


contacting node http://c3po:54205/ ...
Pid: 3825
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /turtle1/cmd_vel
    * to: /teleop_turtle (http://c3po:47526/)
    * direction: inbound
    * transport: TCPROS

viki@c3po:~$
```

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 list

- Displays the list of current topics:

```
roiyeho@ubuntu: ~
roiyeho@ubuntu:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
roiyeho@ubuntu:~$
```

Publish to ROS Topic

- Use the **rostopic pub** command to publish messages to a topic
- For example, to make the turtle move forward at a 0.2m/s speed, you can publish a cmd_vel message to the topic /turtle1/cmd_vel:

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

  – To specify only the linear x velocity:

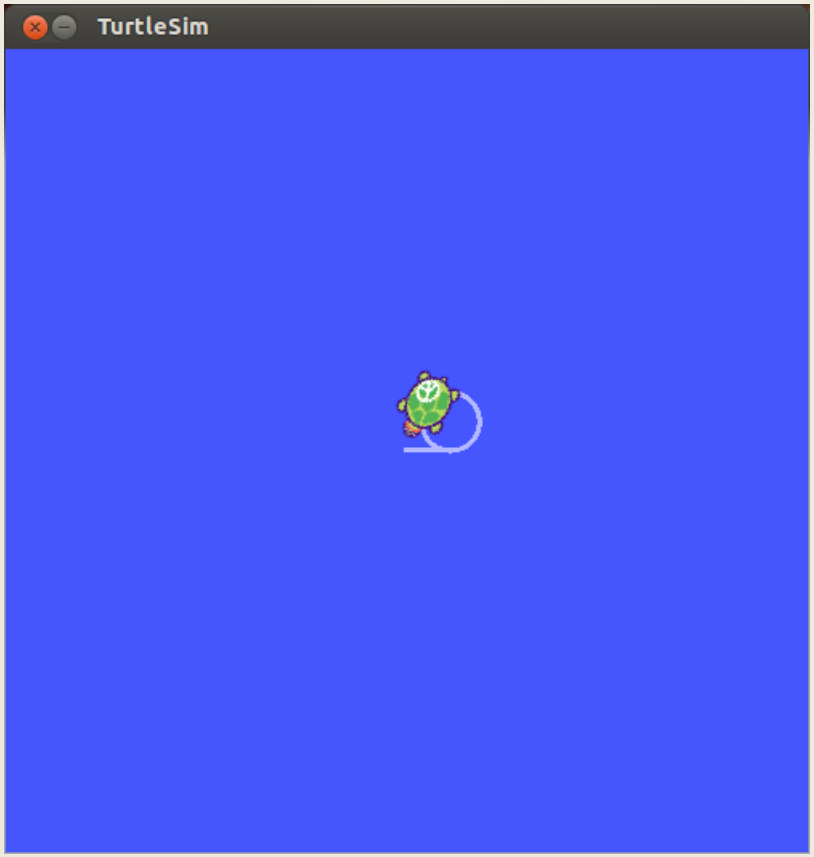
```
$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist '{linear: {x: 0.2}}'
```

Publish to ROS Topic

- Some of the messages like cmd_vel have a predefined timeout
- If you want to publish a message continuously use the argument **-r** with the loop rate in Hz
- For example, to make the turtle turn in circles continuously, type:

```
$ rostopic pub /turtle1/cmd_vel -r 10 geometry_msgs/Twist ‘{angular: {z: 0.5}}'
```

Publish to ROS Topic



Ex. 1

- Run the turtlesim node
- Send a command to turtlesim to move backwards continuously at 5Hz rate