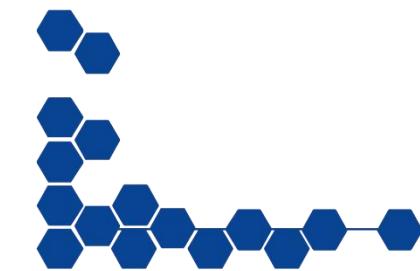




ROS-Industrial Basic Developer's Training Class

February 2017

Southwest Research Institute





Session 1: ROS Basics

February 2017

Southwest Research Institute



February 2017





Outline



- Intro to ROS
- Catkin (Create workspace)
- Installing packages (existing)
- Packages (create)
- Nodes
- Messages / Topics





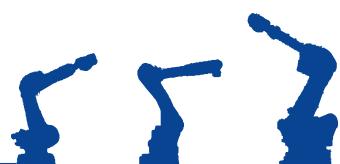
An Introduction to ROS



ROS

(Image taken from Willow Garage's "What is ROS?" presentation)

February 2017





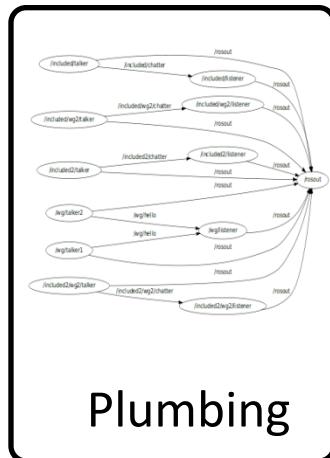
What is ROS?



ROS is...

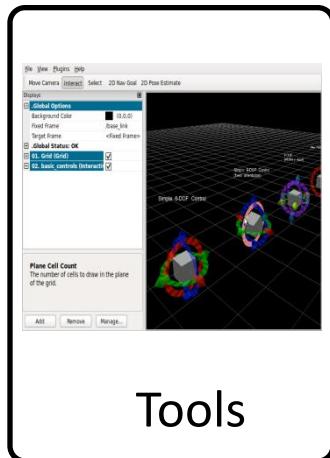


=



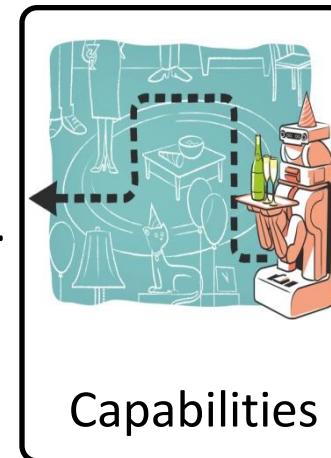
Plumbing

+



Tools

+



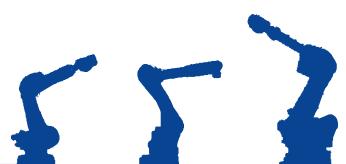
Capabilities

+



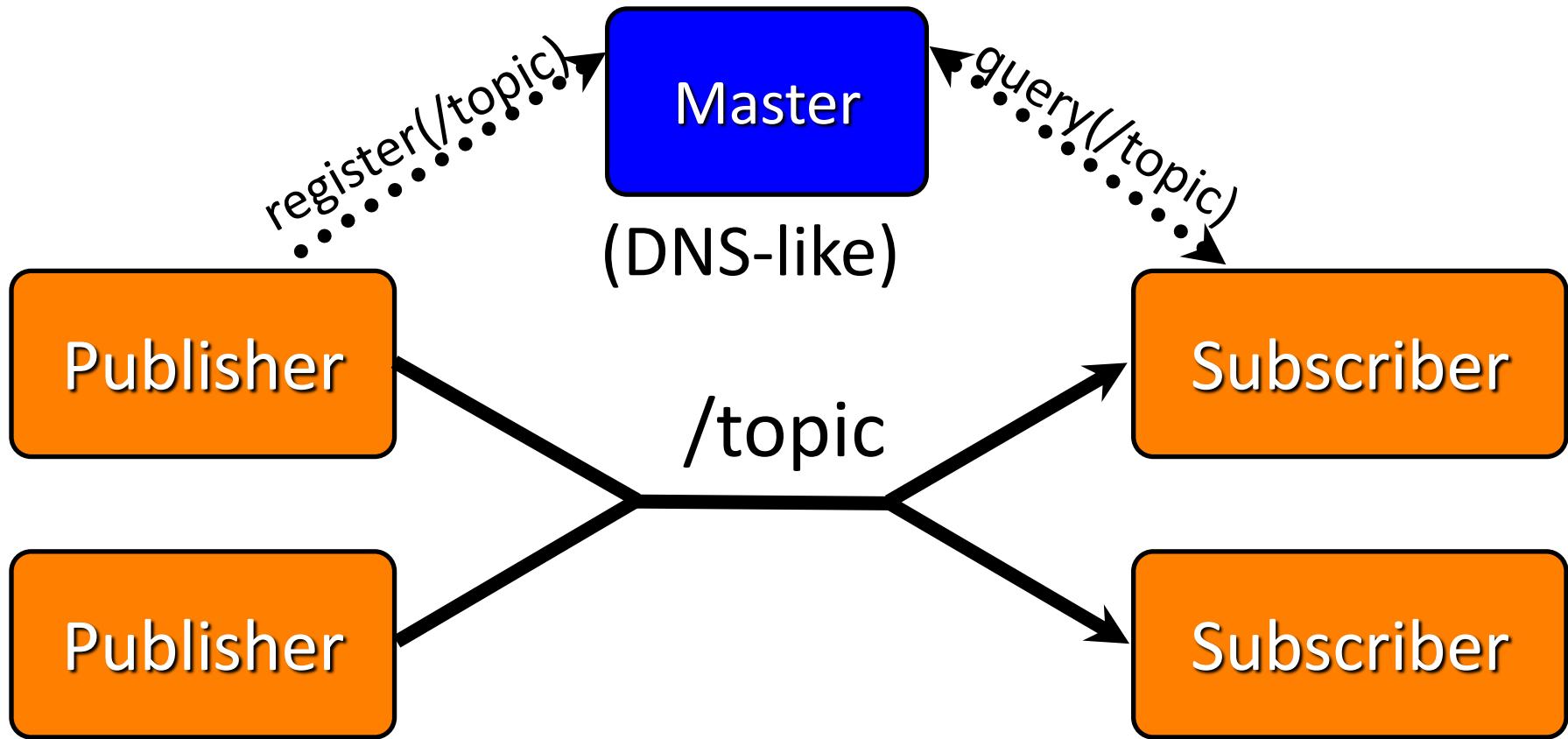
Ecosystem

(Adapted from Willow Garage's "What is ROS?" Presentation)





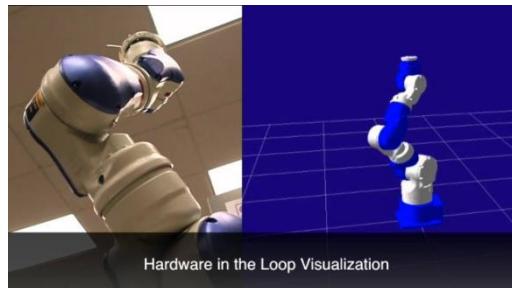
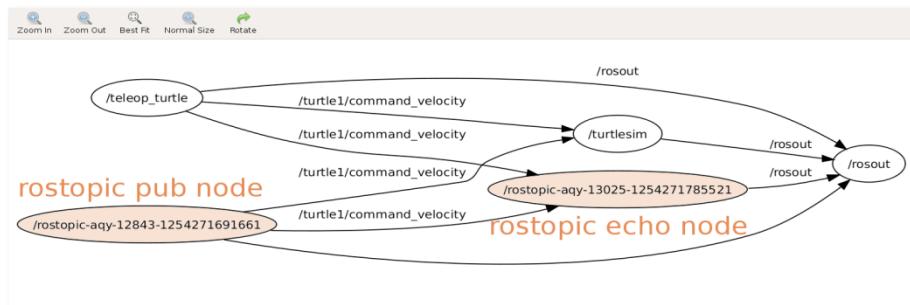
ROS is... plumbing



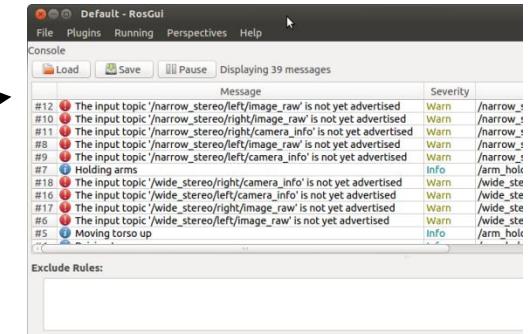
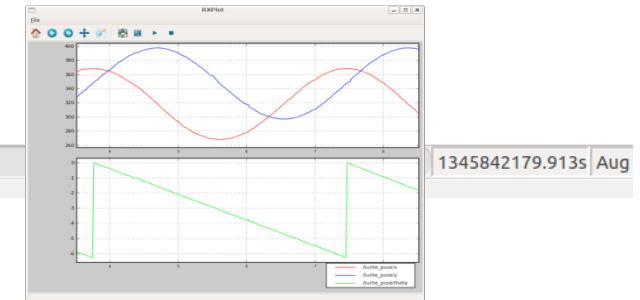
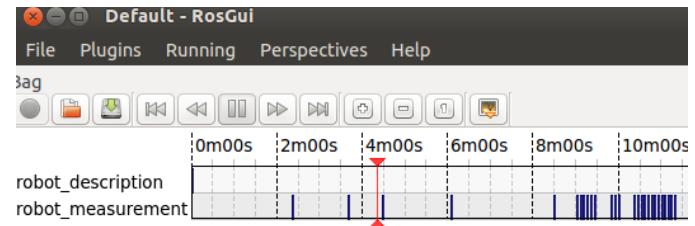
(Adapted from Willow Garage's "What is ROS?" Presentation)



ROS is ... Tools



- logging/plotting
- graph visualization
- diagnostics
- visualization

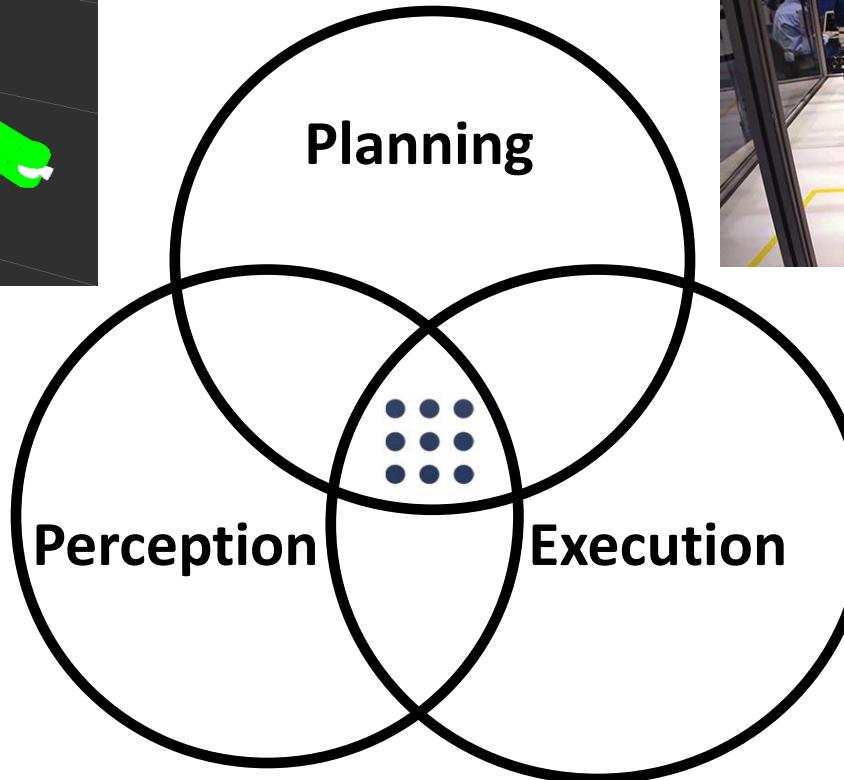
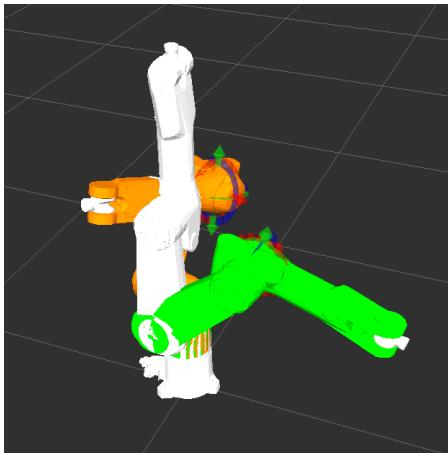


(Adapted from Willow Garage's "What is ROS?" Presentation)





ROS is...Capabilities

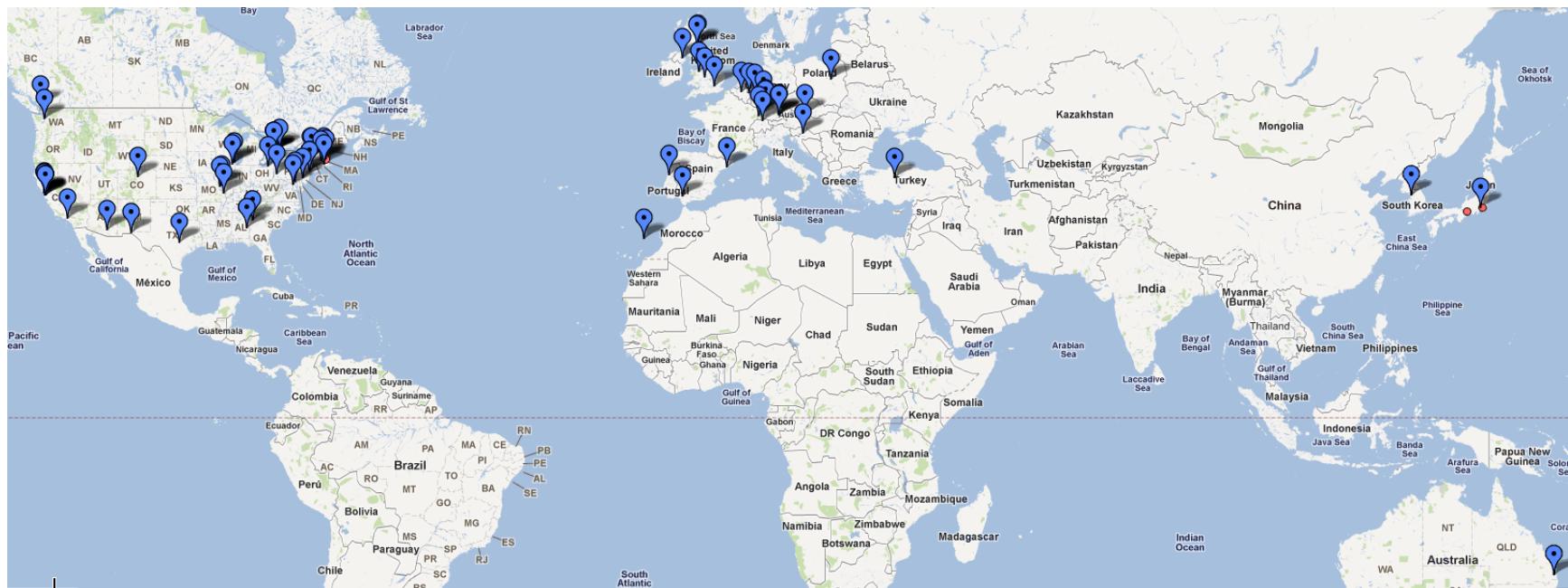


(Adapted from Willow Garage's "What is ROS?" Presentation)





ROS is... an Ecosystem



(Adapted from Willow Garage's "What is ROS?" Presentation)

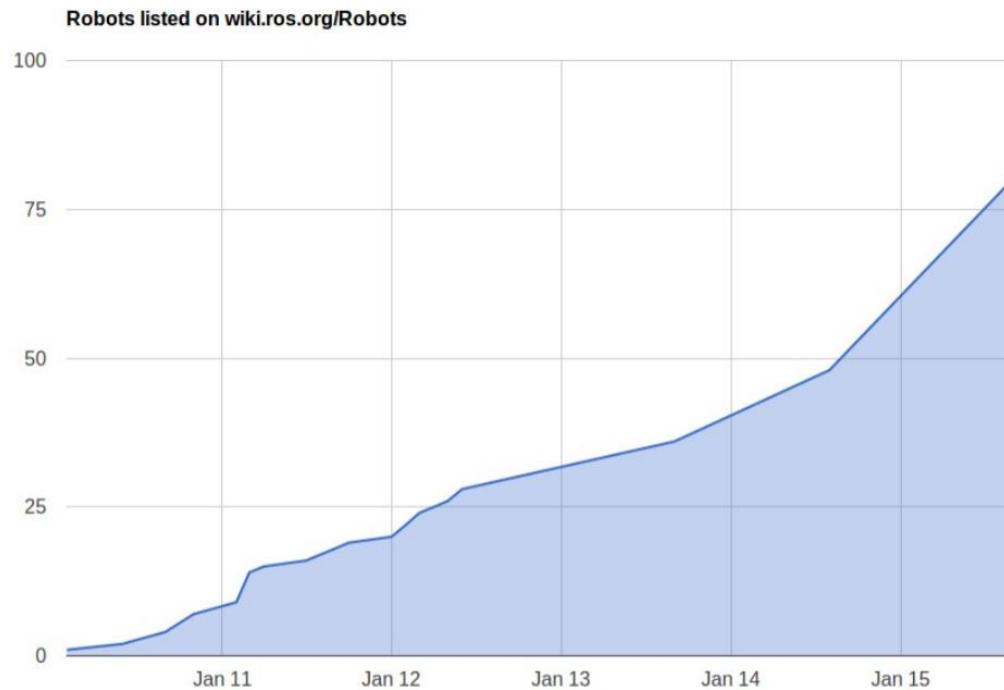


February 2017





ROS is a growing Ecosystem



The number of different types of robots available to the community with ROS drivers.

Source: Ken Conley, Tully Foote, wiki.ros.org/Robots



February 2017





ROS is ...



- <https://www.youtube.com/watch?v=PGaXiLZD2KQ>

(Adapted from Willow Garage's "What is ROS?" Presentation)

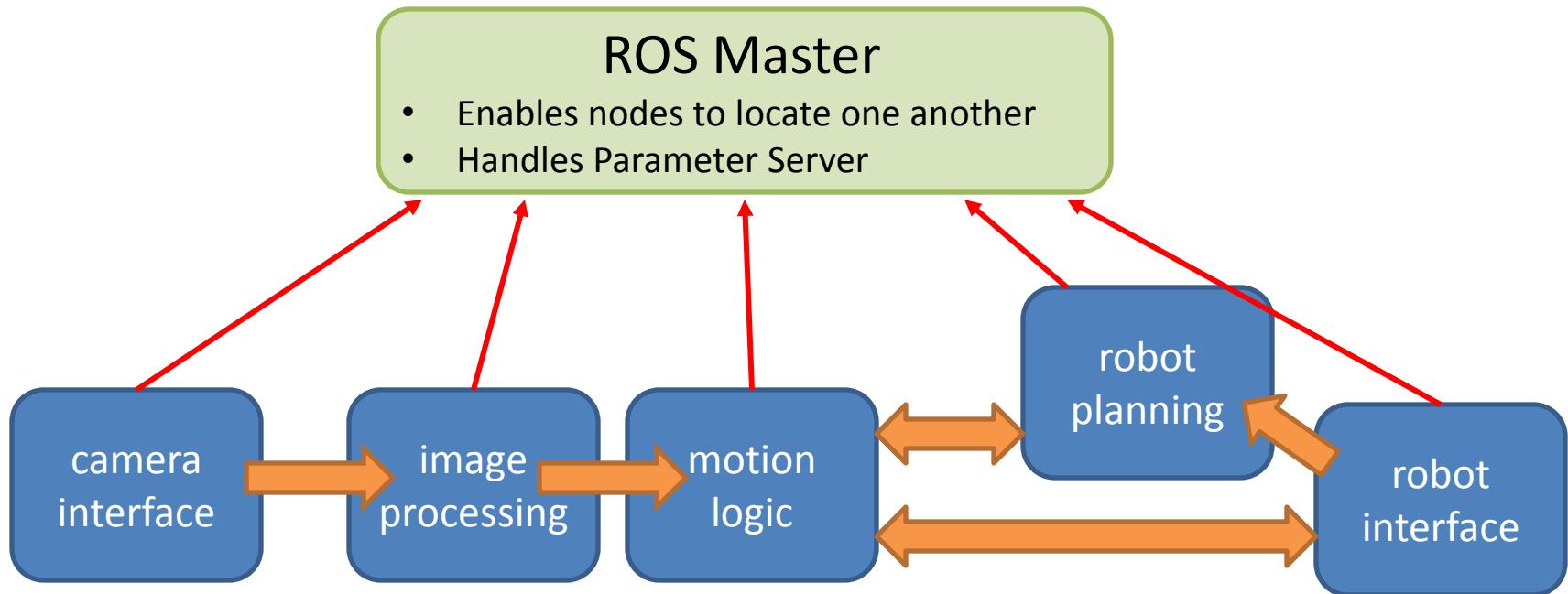


February 2017





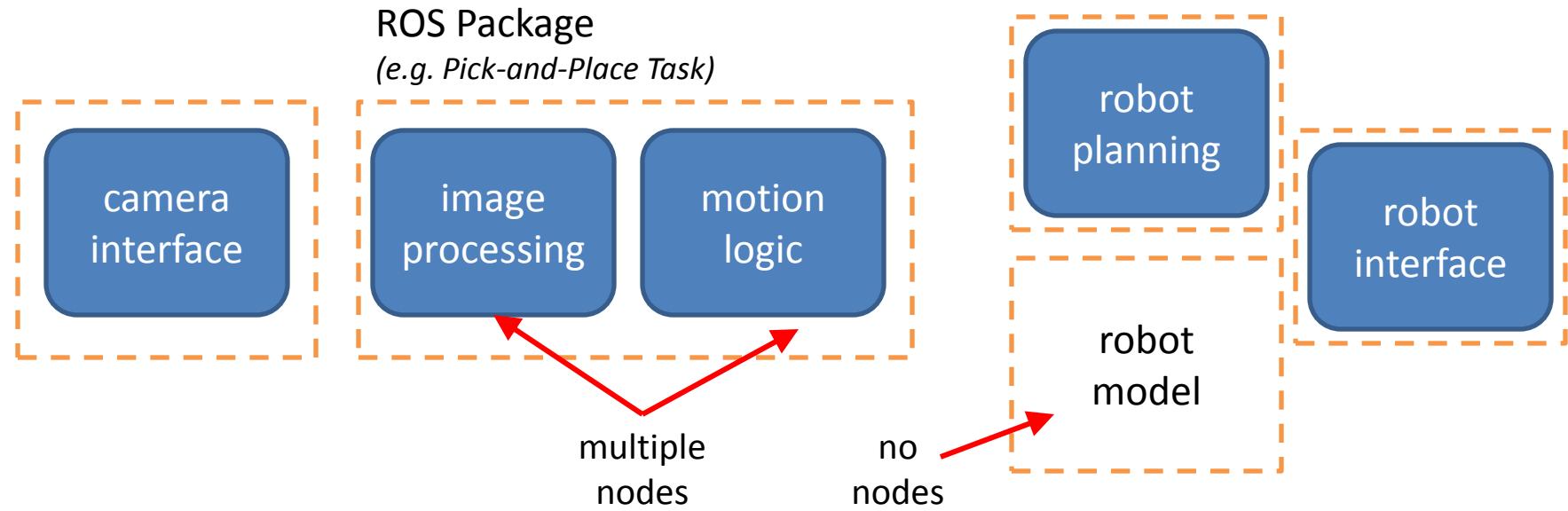
ROS Architecture: Nodes



- A **Node** is a single ROS-enabled program
 - Most communication happens **between** nodes
 - Nodes can run on many different **devices**
- One **Master** per system



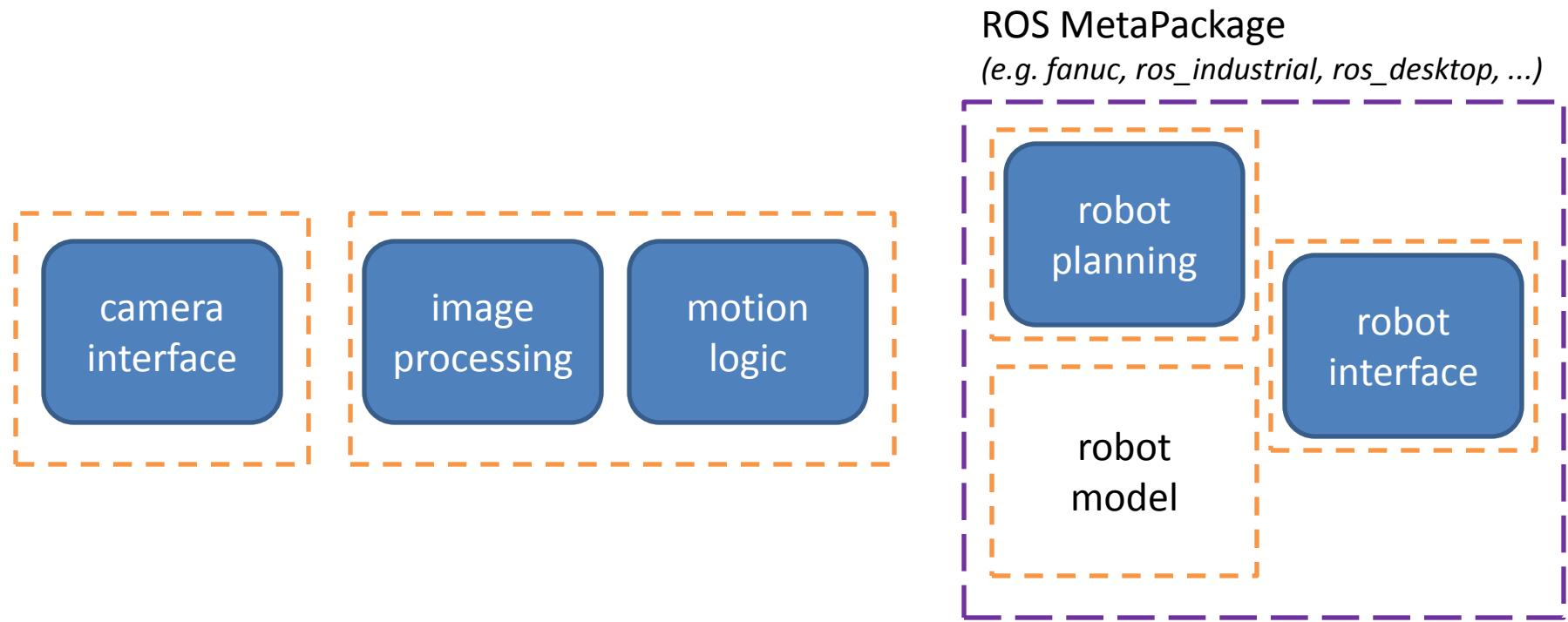
ROS Architecture: Packages



- **ROS Packages** are groups of related nodes/data
 - Many ROS commands are **package-oriented**



ROS Architecture: MetaPkg



- **MetaPackages** are groups of related packages
 - Mostly for convenient install/deployment



February 2017





ROS Programming



- ROS uses **platform-agnostic** methods for most communication
 - TCP/IP Sockets, XML, etc.
- Can intermix programming languages
 - currently: C++, Python, Lisp
 - We will be using C++ for our exercises



ROS Resources



February 2017





ROS.org Website



<http://ros.org>

The screenshot shows the ROS.org website homepage. At the top, there's a blue header bar with the URL "http://ros.org". Below it is the main navigation bar with links: "About", "Why ROS?", "Getting Started" (which has a red arrow pointing to it), "Get Involved", and "Blog". The main content area features a large image of a white PR2 robot. To the left of the robot is a box titled "What is ROS?" containing text about the Robot Operating System and a "Read More" button. Below the robot are several sections: "ROS Indigo Igloo" with a download button, "ROS Spotlight: Jade logo and t-shirt campaign" with a green t-shirt image, and three links on the right: "Wiki", "ROS Answers", and "Blog". Each of these three links has a red arrow pointing to it.

- Install Instructions
- Tutorials
- Links
 - Packages, ROS Answers, etc.





Package Wiki



<http://wiki.ros.org/<packageName>>

tf

Documentation Status

electric fuerte groovy hydro indigo jade

geometry: angles | eigen_conversions | kdl_conversions | tf | tf_conversions

Package Summary

✓ Released ✓ Continuous integration ✓ Documented

tf is a package that lets the user keep track of multiple coordinate frames over time. It maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

- Maintainer status: maintained
- Maintainer: Tully Foote <tfoote AT osrfoundation DOT org>
- Author: Tully Foote, Eitan Marder-Eppstein, Wim Meeussen
- License: BSD
- Source: git <https://github.com/ros/geometry.git> (branch: indigo-devel)

Contents

- What does tf do? Why should I use tf?
- Paper
- Tutorials
- Code API Overview
- Frequently asked questions
- CommandLine Tools

Package Links

[Code API](#)
[Msg/Srv API](#)
[Tutorials](#)
[Troubleshooting](#)
[FAQ](#)
[Changelog](#)
[Change List](#)
[Roadmap](#)
[Reviews](#)

Dependencies (15)
[Used by \(275\)](#)
[Jenkins jobs \(7\)](#)

7.2 change_notifier

change_notifier listens to /tf and periodically republishes any transforms that have changed by a given /tf_changes topic.

7.2.1 Subscribed Topics

/tf ([tf/TfMessage](#))
Transform tree.

7.2.2 Published Topics

/tf_changes ([tf/TfMessage](#))
Reduced transform tree.

7.2.3 Parameters

`-polling_frequency (float, default: 10.0)`
Frequency (hz) at which to check for any changes to the transform tree.

`-translational_update_distance (float, default: 0.1)`
Minimum distance between the origin of two frames for the transform to be considered changed.

`-angular_update_distance (float, default: 0.1)`
Minimum angle between the rotation of two frames for the transform to be considered changed.

- Description / Usage
- Tutorials
- Code / Msg API
- Source-Code link
- Bug Reporting



February 2017





ROS Answers

<http://answers.ros.org>



ROS ANSWERS

Hi there! Please sign in | help

tags users badges

ALL UNANSWERED search or ask your question ASK YOUR QUESTION

21,783 questions

Sort by: by date by activity by answers by votes RSS

How to save static transforms in bag files? (1 answer, 12 views)

pcl 1.7.2 installation question (no votes, 9 views)

freenect_launch with Kinect (no votes, 3 views)

Problem using serial write (no votes, 14 views)

schunk_svh_driver : Can't locate node svh_controller in package schunk_svh_driver (1 answer, 8 views)

Broken url in tutorial (no votes, 9 views)

Contributors

Tag search

Tags

ROS ANSWERS

Hi there! Please sign in | help

tags users badges

ALL UNANSWERED search or ask your question ASK YOUR QUESTION

How can I use Motoman stack with ROS Indigo?

I have Ubuntu 14.04

Follow 1 follower subscribe to ros feed

Stats

Asked: Aug 25 '14 Sent: 81 times Last updated: 12 hours ago

Related questions

Installing ROS Industrial on INDIGO Problem using ur5/ur10 with moveit in indigo Building Indigo on 14.10 Universal robots calibration offsets I can't install openni on Indigo, what Linux version should I install instead Tablettop or object detector for indigo urg node on ros indigo Macros topics not publishing? ROS Indigo install on ubilinux - Edison failed

Code Block and catkin indigo

```
cd /path/to/your/catkin_ws/src
# download the official version of the motoman repository.
# If you'd rather use the development versions, use '--hydro-devel' OR '--indigo-devel'.
git clone -b hydro https://github.com/ros-industrial/motoman_gits
# we need to make sure you have all dependencies installed.
# this step should install 'industrial_robot_client' for you
cd ..
rospack install --from-path src --ignore-src --rospackdir indigo
# now build
catkin_make
```

This should successfully build the Motoman ROS nodes. You'll still have to set up your controller, but those steps are identical to the Hydro version (see [motoman_driver/Tutorials](#) on the ROS wiki).

- Quick responses to Good Questions
- Search by text or tag
- Don't re-invent the wheel!



February 2017





ROS is a Community



- No Central “Authority” for Help/Support
 - Many users can provide better (?) support
 - ROS-I Consortium can help fill that need
- Most ROS-code is open-source
 - can be reviewed / improved by everyone
 - we count on **YOU** to help ROS grow!



What is ROS to you?

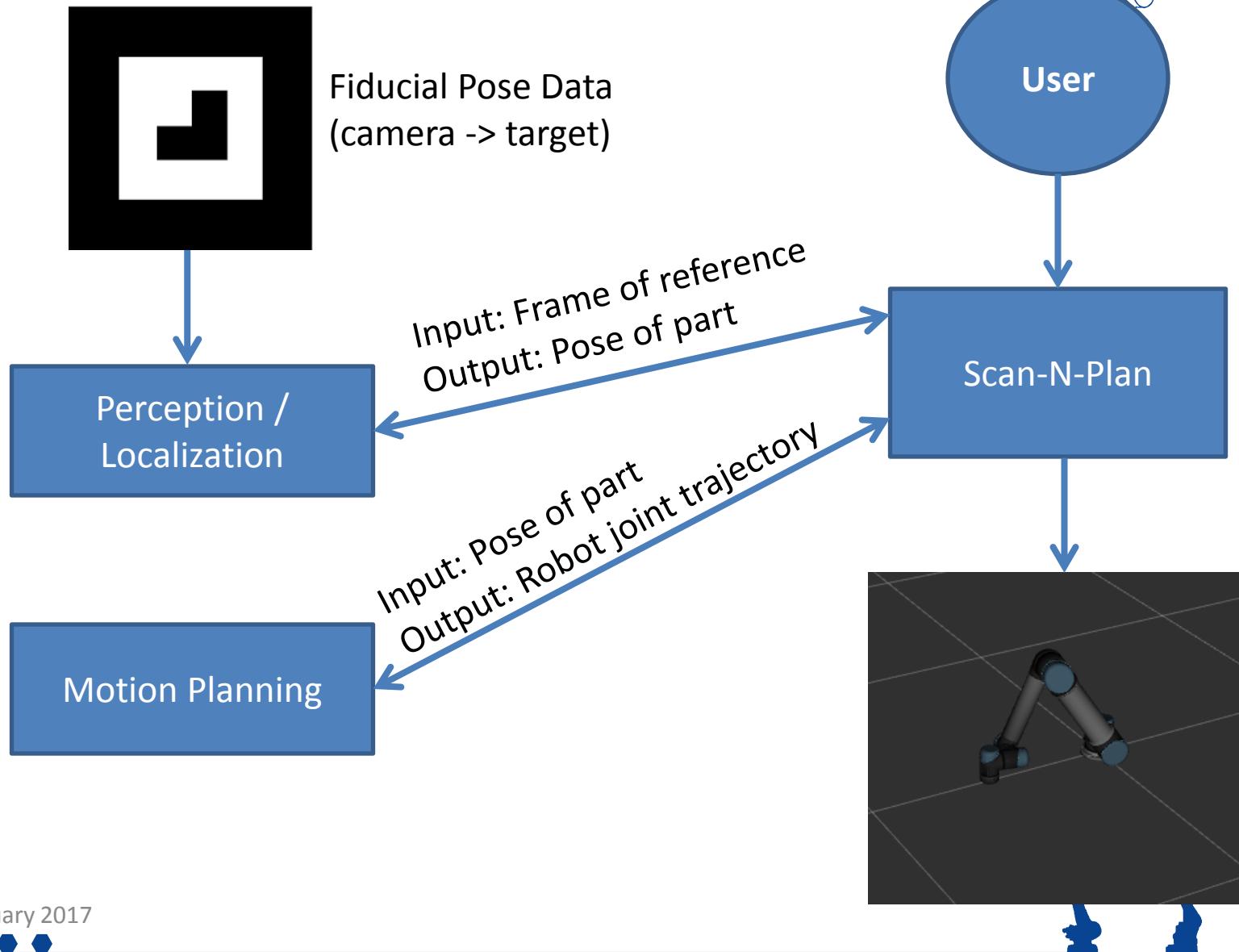


Training Goals:

- Show you ROS as a software framework
- Show you ROS as a tool for problem solving
- Build your own perception driven & dynamically executed *Scan-N-Plan* application
- Ask lots of questions and break things.



The Scan-N-Plan App

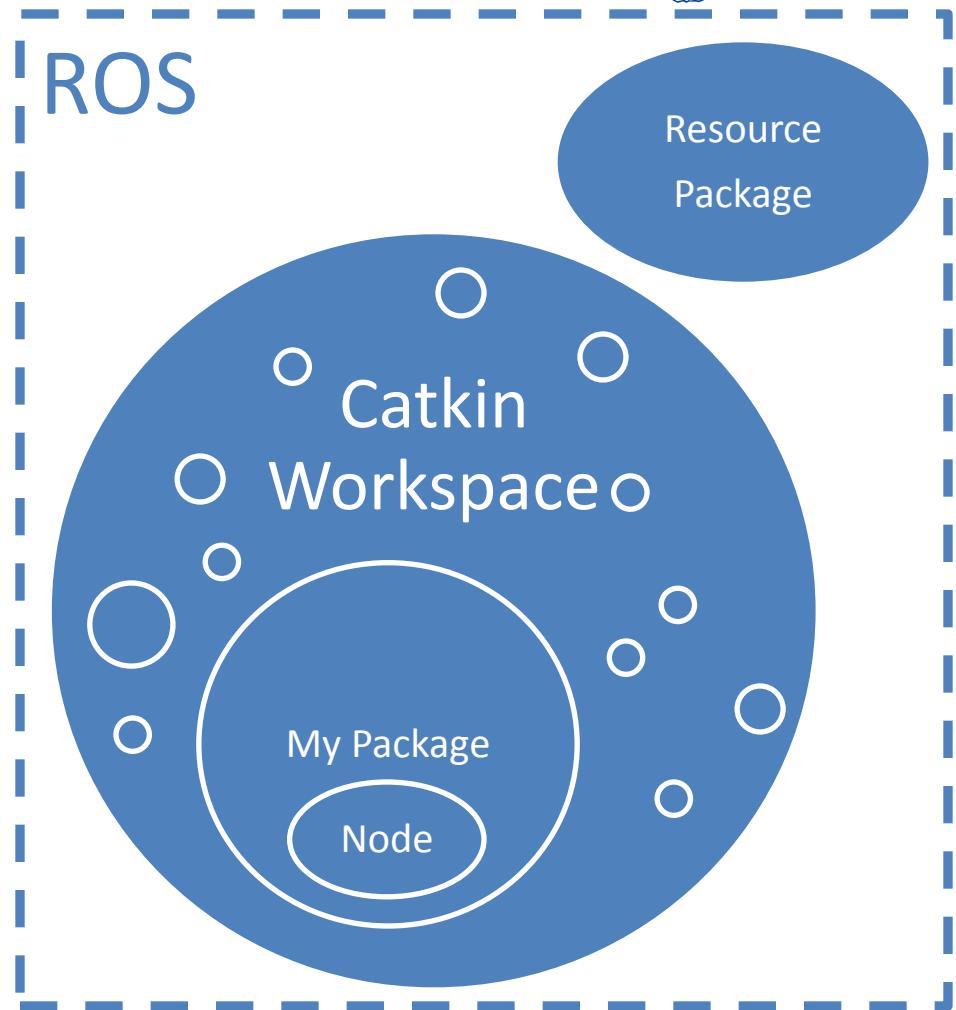




Day 1 Progression



- Install ROS
- Create Workspace
- Add “resources”
- Create Package
- Create Node
 - Basic ROS Node
 - Interact with other nodes
 - Messages
 - Services
- Run Node
 - rosrun
 - roslaunch
 - rosparam

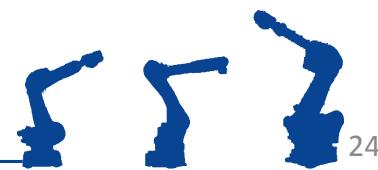




Installing ROS



February 2017





Getting ROS



<http://wiki.ros.org/kinetic/Installation>



Roscore



roscore is a collection of nodes and programs that are pre-requisites of a ROS-based system

To check your install, open a terminal and type:

roscore

To kill the process, press ***Ctrl+C*** while in the window running ***roscore***

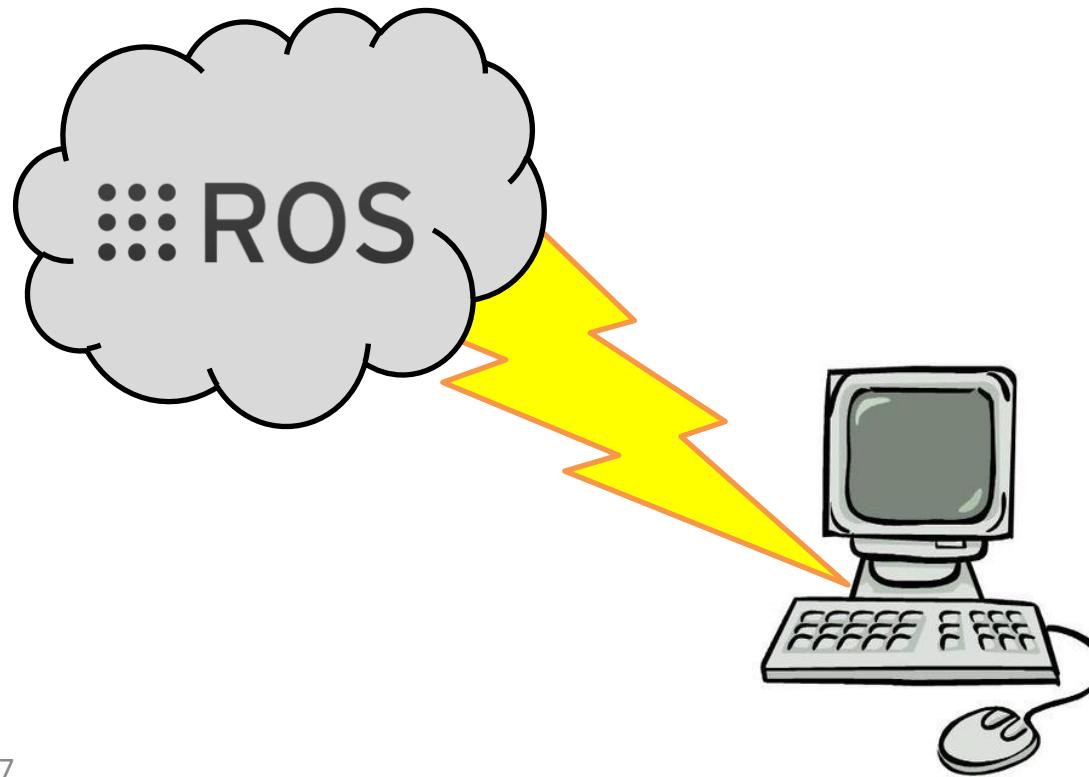


Exercise 1.0



Exercise 1.0

Basic ROS Install/Setup

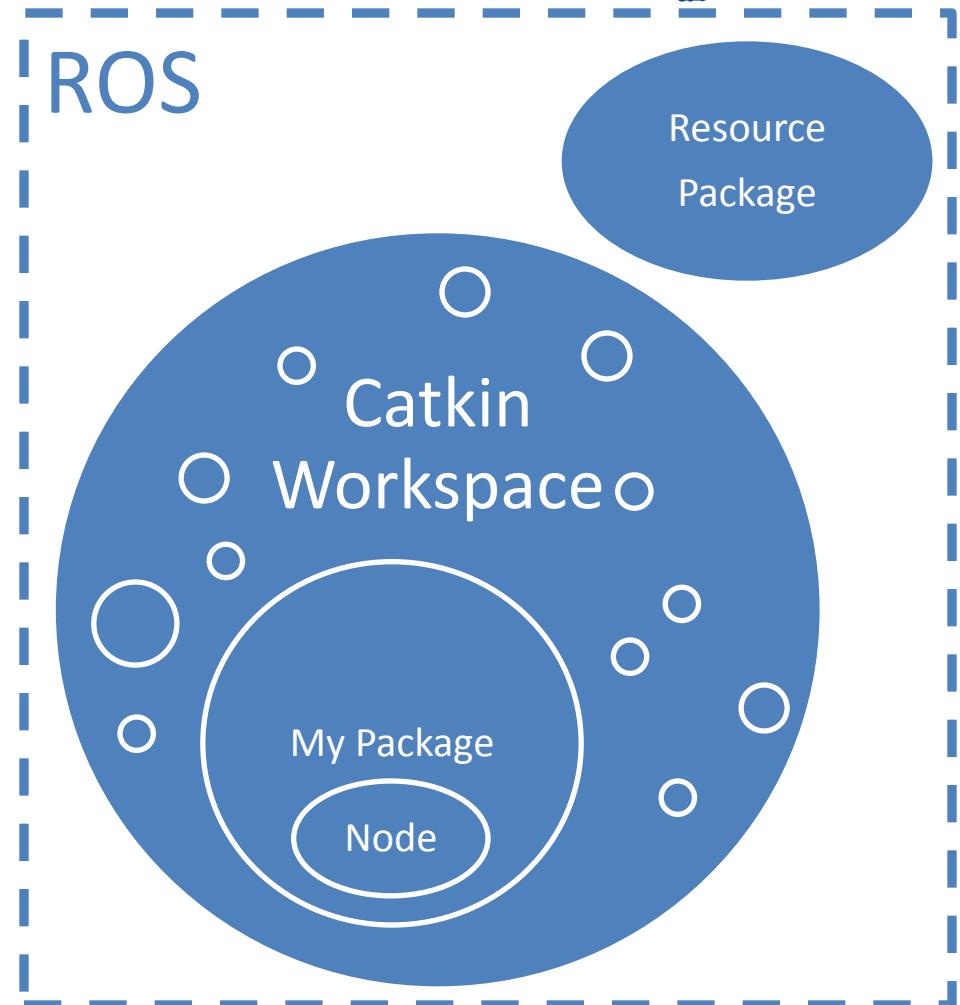




Day 1 Progression



- ✓ Install ROS (check install)
- ❑ Create Workspace
- ❑ Add “resources”
- ❑ Create Package
- ❑ Create Node
 - ❑ Basic ROS Node
 - ❑ Interact with other nodes
 - ❑ Messages
 - ❑ Services
- ❑ Run Node
 - ❑ rosrun
 - ❑ roslaunch



February 2017





Creating a ROS Workspace



- ROS uses the **catkin** build system
 - based on CMAKE
 - cross-platform (Ubuntu, Windows, embedded...)
 - replaces older **rosbuild** system
 - different build commands, directory structure, etc.
 - most packages have already been upgraded to catkin
 - **rosbuild**: `manifest.xml`, **catkin**: `package.xml`

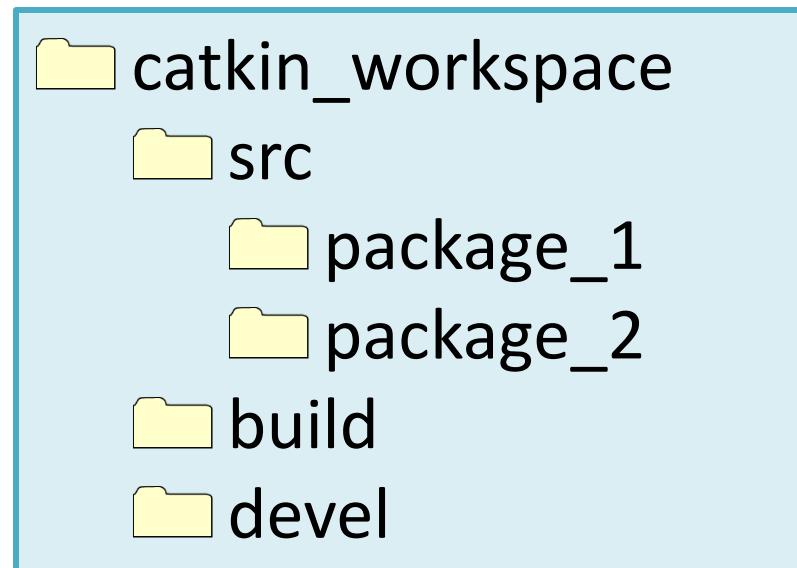




Catkin Workspace



- Catkin uses a specific directory structure:
 - each “project” typically gets its own **catkin workspace**
 - all packages/source files go in the **src** directory
 - temporary build-files are created in **build**
 - results are placed in **devel**





Setup (one-time)

1. Create a catkin workspace somewhere
 - my_catkin_ws/src
 - build, devel directories created automatically
2. Run `catkin init` in **my_catkin_ws**
3. Download/create packages in **src** subdir

Compile-Time

1. Run `catkin build` in **workspace root**
2. Run `source devel/setup.bash` to make workspace visible to ROS
 - Must re-execute in **each** new terminal window



Exercise 1.1

Create a Catkin Workspace

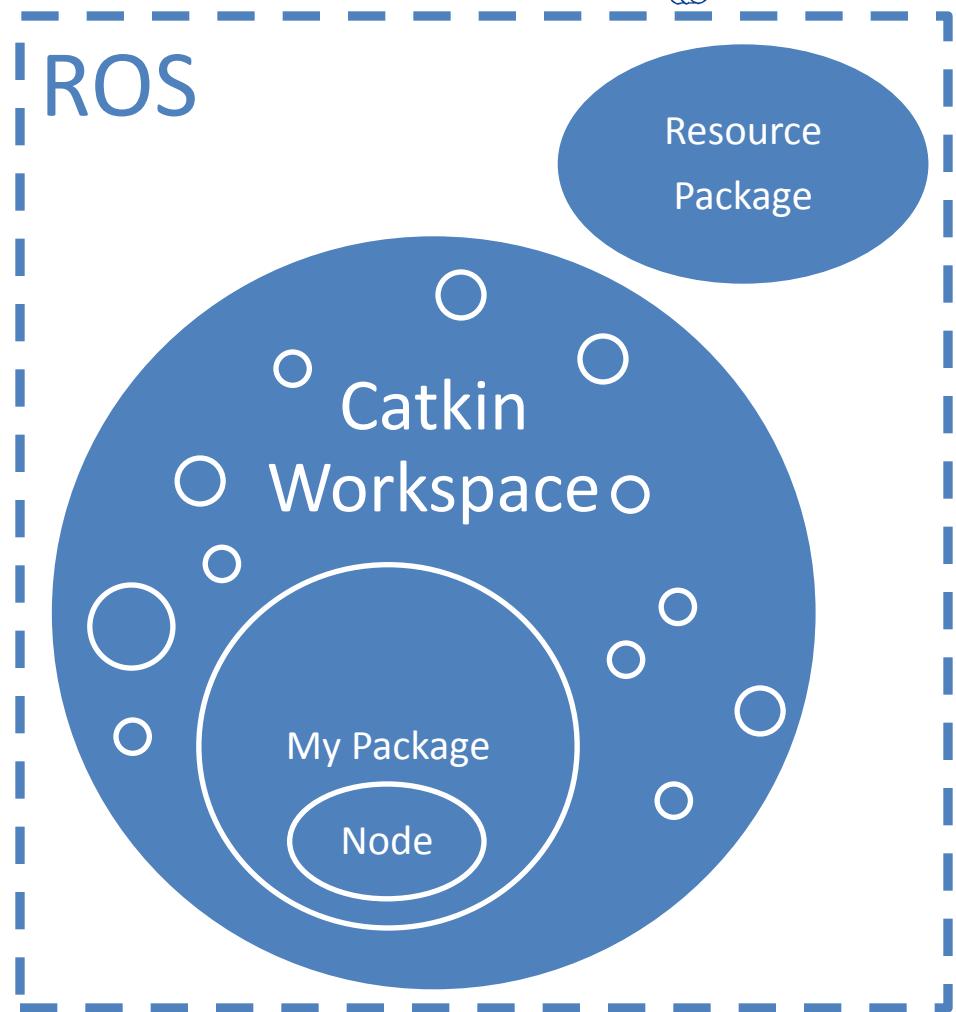
```
50%] Generating Python msg __init__.py for robotiq_c_model_control
50%] Built target robotiq_c_model_control_generate_messages_py
54%] Scanning dependencies of target robotiq_c_model_control_generate_messages_lisp
generating C++ code from robotiq_c_model_control/CModel_robot_input.msg
59%] Generating Lisp code from robotiq_c_model_control/CModel_robot_output.msg
63%] Generating Lisp code from robotiq_c_model_control/CModel_robot_input.msg
63%] Built target robotiq_c_model_control_generate_messages_lisp
63%] Built target robotiq_c_model_control_generate_messages_cpp
Scanning dependencies of target robotiq_s_model_control_generate_messages_cpp
68%] Scanning dependencies of target robotiq_s_model_control_generate_messages_lisp
generating C++ code from robotiq_s_model_control/SModel_robot_output.msg
72%] Generating Lisp code from robotiq_s_model_control/SModel_robot_output.msg
77%] Generating Lisp code from robotiq_s_model_control/SModel_robot_input.msg
77%] Built target robotiq_s_model_control_generate_messages_lisp
81%] Scanning dependencies of target robotiq_s_model_control_generate_messages_py
generating C++ code from robotiq_s_model_control/SModel_robot_input.msg
86%] Generating Python from MSG robotiq_s_model_control/SModel_robot_output
90%] Generating Python from MSG robotiq_s_model_control/SModel_robot_input
95%] Generating Python msg __init__.py for robotiq_s_model_control
95%] Built target robotiq_s_model_control_generate_messages_cpp
```



Day 1 Progression



- ✓ Install ROS
- ✓ Create Workspace
- ❑ Add “resources”
- ❑ Create Package
- ❑ Create Node
 - ❑ Basic ROS Node
 - ❑ Interact with other nodes
 - ❑ Messages
 - ❑ Services
- ❑ Run Node
 - ❑ rosrun
 - ❑ roslaunch



February 2017





Instead of text editor and building from terminal...

Use an IDE! [Wiki instructions here](#)





Debian Packages

- Nearly “automatic”
- Recommended for end-users
- Stable
- Easy

Source Repositories

- Access “latest” code
- Most at Github.com
- More effort to setup
- Unstable*

Can mix both options, as needed



Finding the Right Package



- ROS Website (<http://ros.org/browse/>)
 - Browse/Search for known packages
- ROS Answers (<http://answers.ros.org>)
 - When in doubt... ask someone!
- apt-cache search <search term>
 - Searches titles and descriptions for search term





Install using Debian Packages



```
sudo apt-get install ros-distro-package
```

↑
admin permissions ↑
manage ".deb" ↑
install new ".deb"
all ROS pkgs start with `ros-` ↑
ROS distribution ↑
ROS package name

Use “-” not “_”

- Fully automatic install:
 - Download .deb package from central ROS repository
 - Copies files to standard locations (/opt/ros/indigo/...)
 - Also installs any other required dependencies
- `sudo apt-get remove ros-distro-package`
 - Removes software (but not dependencies!)



Installing from Source



- Find Github repo
- Clone repo into your workspace src directory

```
cd catkin_ws/src  
git clone "repo"
```

- Build your catkin workspace

```
cd catkin_ws  
catkin build
```

- Now the package and its resources are available to you



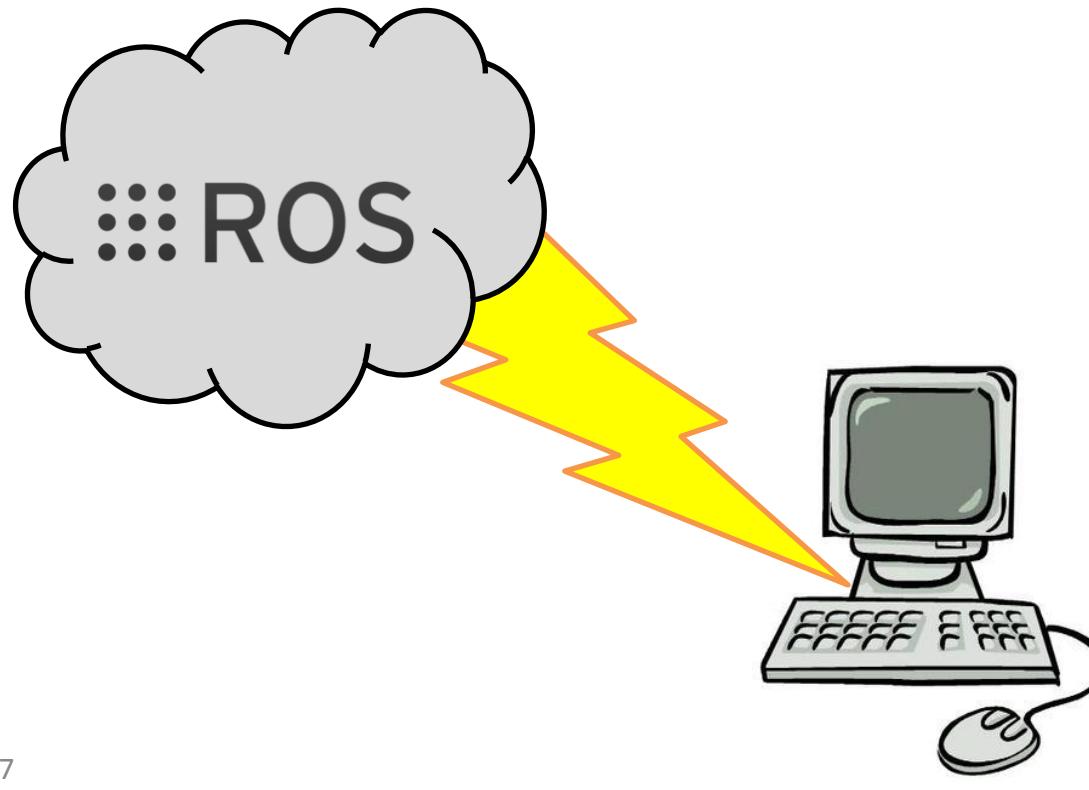
February 2017

<http://www.clearpathrobotics.com/blog/introducing-catkin/>



Exercise 1.2

Install “resource” packages

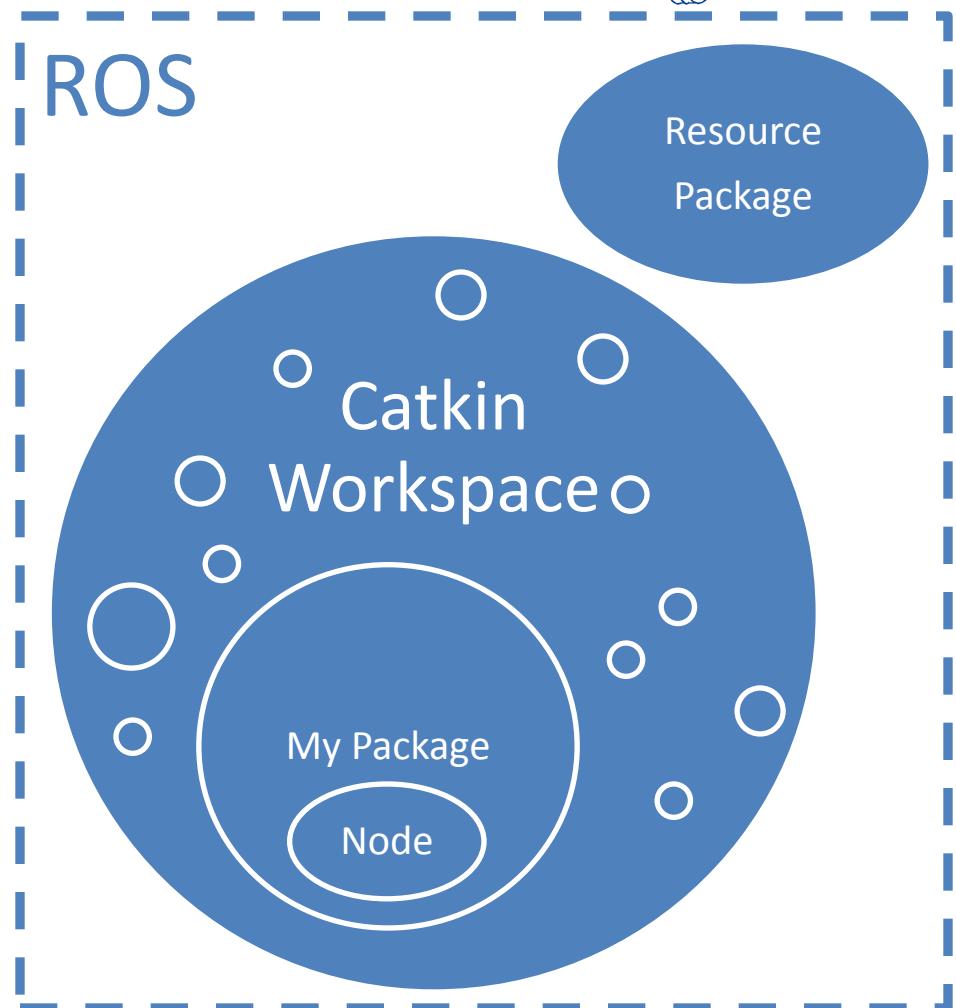




Day 1 Progression



- ✓ Install ROS
- ✓ Create Workspace
- ✓ Add “resources”
- Create Package
- Create Node
 - Basic ROS Node
 - Interact with other nodes
 - Messages
 - Services
- Run Node
 - rosrun
 - roslaunch



February 2017





ROS Packages

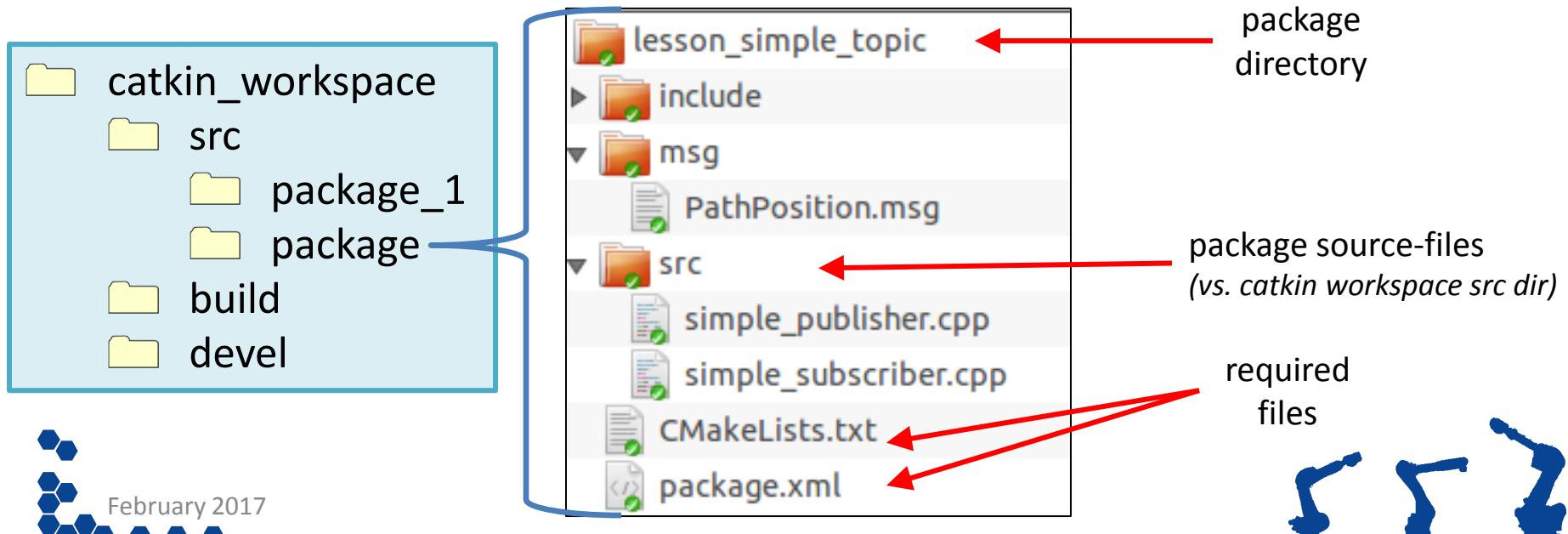




ROS Package Contents



- ROS components are organized into **packages**
- Packages contain several **required files**:
 - package.xml
 - **metadata** for ROS: package name, description, dependencies, ...
 - CMakeLists.txt
 - **build rules** for catkin





package.xml



- Metadata: name, description, author, license ...

```
<package>
  <name>lesson_simple_parameters</name>
  <version>0.0.0</version>
  <description>The lesson_simple_parameters package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example: -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="jane.doe@example.com">Jane Doe</maintainer>

  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>BSD</license>
```



package.xml



- Metadata: name, description, author, license ...
- Dependencies:
 - <depend>: Needed to **build**, **export**, and **execution** dependency.
 - <buildtool_depend>: Needed to **build** itself. (Typically **catkin**)
 - <build_depend>: Needed to **build** this package.
 - <build_export_depend>: Needed to **build against** this package.
 - <exec_depend>: Needed to **run** code in this package.
 - <test_depend>: Only **additional** dependencies for unit tests.
 - <doc_depend>: Needed to generate documentation.



CMakeLists.txt



- Provides **rules for building software**
 - template file contains many examples

include_directories(include \${catkin_INCLUDE_DIRS})

Adds directories to CMAKE include rules

add_executable(myNode src/myNode.cpp src/widget.cpp)

Builds program myNode, from myNode.cpp and widget.cpp

target_link_libraries(myNode \${catkin_LIBRARIES})

Links node myNode to dependency libraries



ROS Package Commands



- `roscd package_name`

Change to package directory

- **rospack**

- `rospack find package_name`

Find directory of package_name

- `rospack list`

List all ros packages installed

- `rospack depends package_name`

List all dependencies of package_name



Create New Package



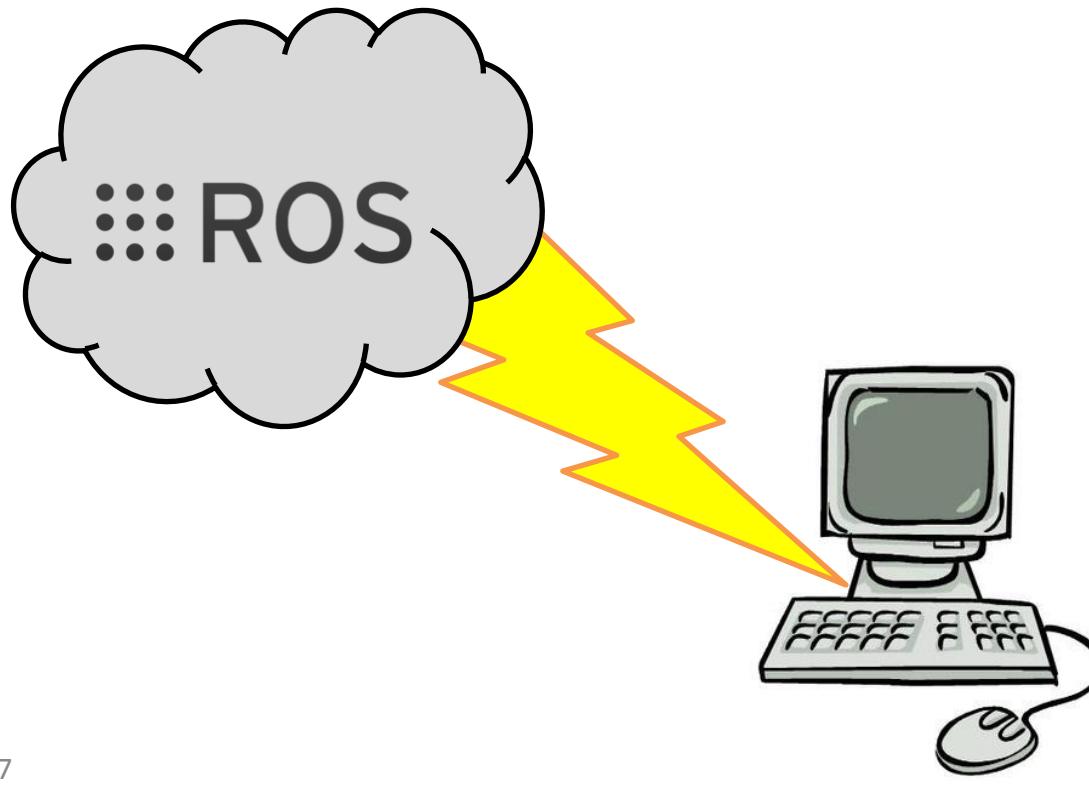
```
catkin create pkg_name dep1 dep2
```

Easiest way to start a new package

- create directory, required files
- `pkg_name` : name of package to be created
- `dep1/2` : dependency package names
 - automatically added to `CMakeLists` and `package.xml`
 - can manually add additional dependencies later

Exercise 1.3.1

Create Package

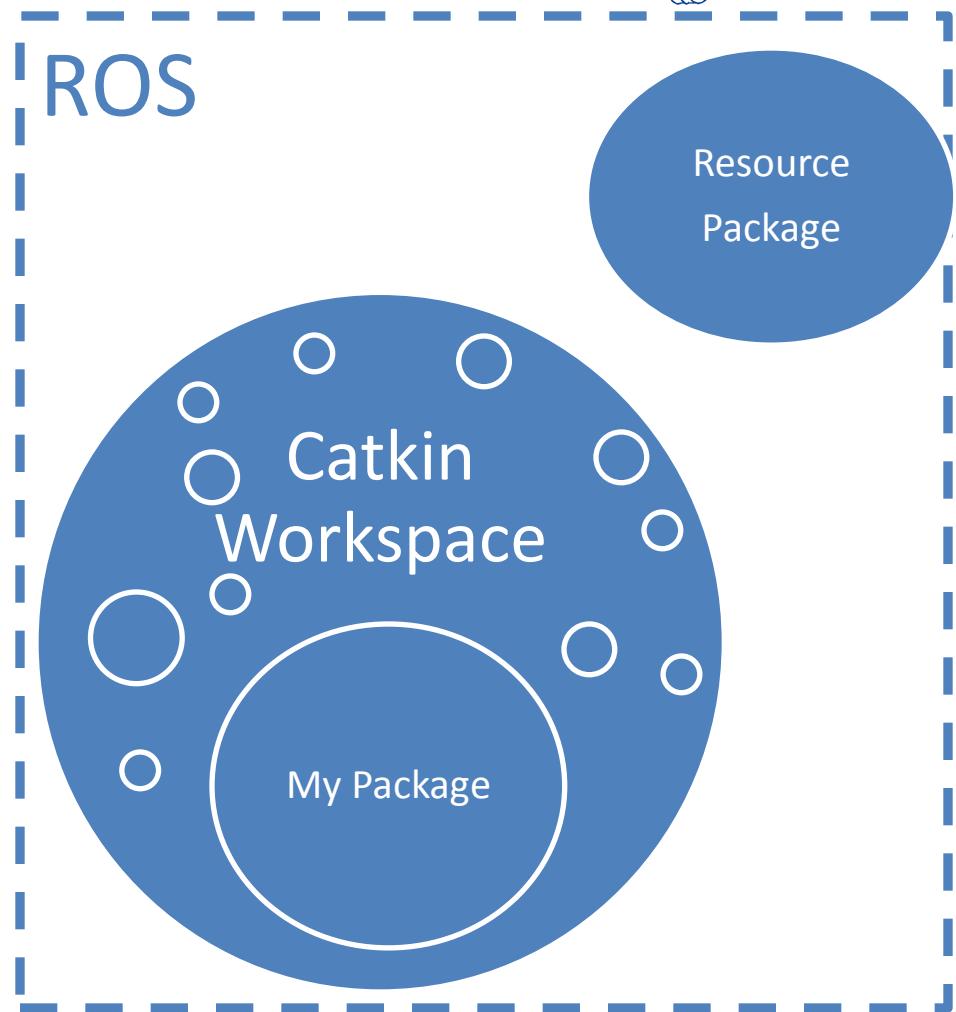




Day 1 Progression



- ✓ Install ROS
- ✓ Create Workspace
- ✓ Add “resources”
- ✓ Create Package
- Create Node
 - Basic ROS Node
 - Interact with other nodes
 - Messages
 - Services
- Run Node
 - rosrun
 - roslaunch



February 2017

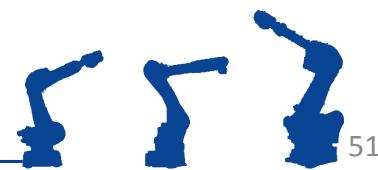




ROS Nodes



February 2017





A Simple C++ ROS Node



Simple C++ Program

```
#include <iostream>

int main(int argc, char* argv[])
{
    std::cout << "Hello World!";

    return 0;
}
```

Simple C++ ROS Node

```
#include <ros/ros.h>

int main(int argc, char* argv[])
{
    ros::init(argc, argv, "hello");
    ros::NodeHandle node;

    ROS_INFO_STREAM("Hello World!");

    return 0;
}
```



ROS Node Commands



- `rosrun package_name node_name`
execute ROS node
- **rosnode**
 - `rosnode list`
View running nodes
 - `rosnode info node_name`
View node details (publishers, subscribers, services, etc.)
 - `rosnode kill node_name`
Kill running node; good for remote machines
 - *Ctrl+C is usually easier*



Exercise 1.3.2



Exercise 1.3.2

Create a Node:

*In myworkcell_core package
called vision_node*

`include_directories(include ${catkin_INCLUDE_DIRS})`

Adds directories to CMAKE include rules

`add_executable(vision_node src/vision_node.cpp)`

Builds program `vision_node`, from `vision_node.cpp`

`target_link_libraries(vision_node ${catkin_LIBRARIES})`

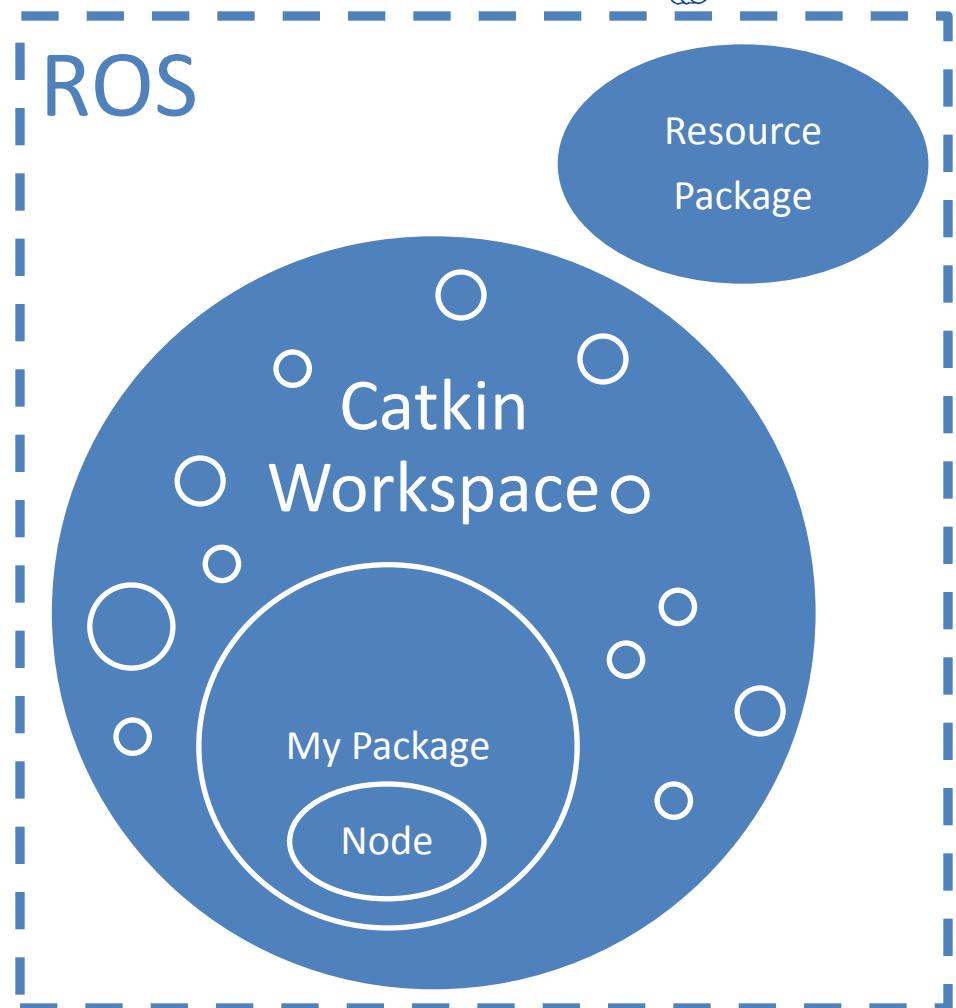
Links node `vision_node` to dependency libraries



Day 1 Progression



- ✓ Install ROS
- ✓ Create Workspace
- ✓ Add “resources”
- ✓ Create Package
- ✓ Create Node
 - ✓ Basic ROS Node
 - ❑ Interact with other nodes
 - ❑ Messages
 - ❑ Services
- ✓ Run Node
 - ✓ rosrun
 - ❑ roslaunch



February 2017



Topics and Messages



February 2017

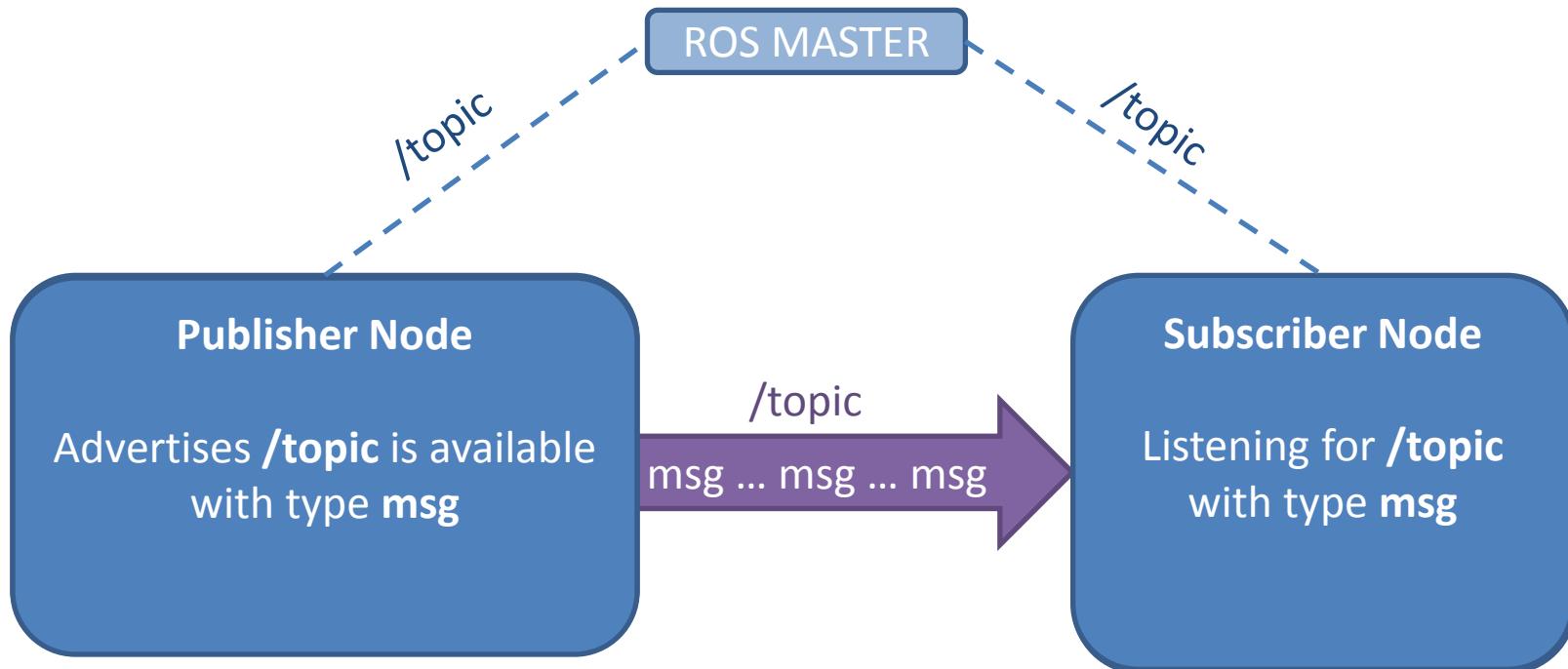




ROS Topics/Messages



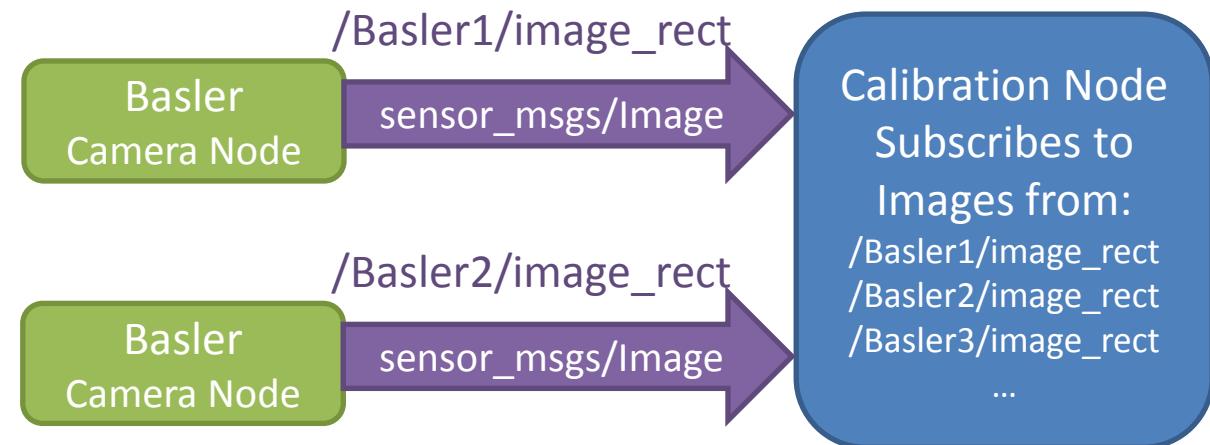
Topics are for **Streaming Data**



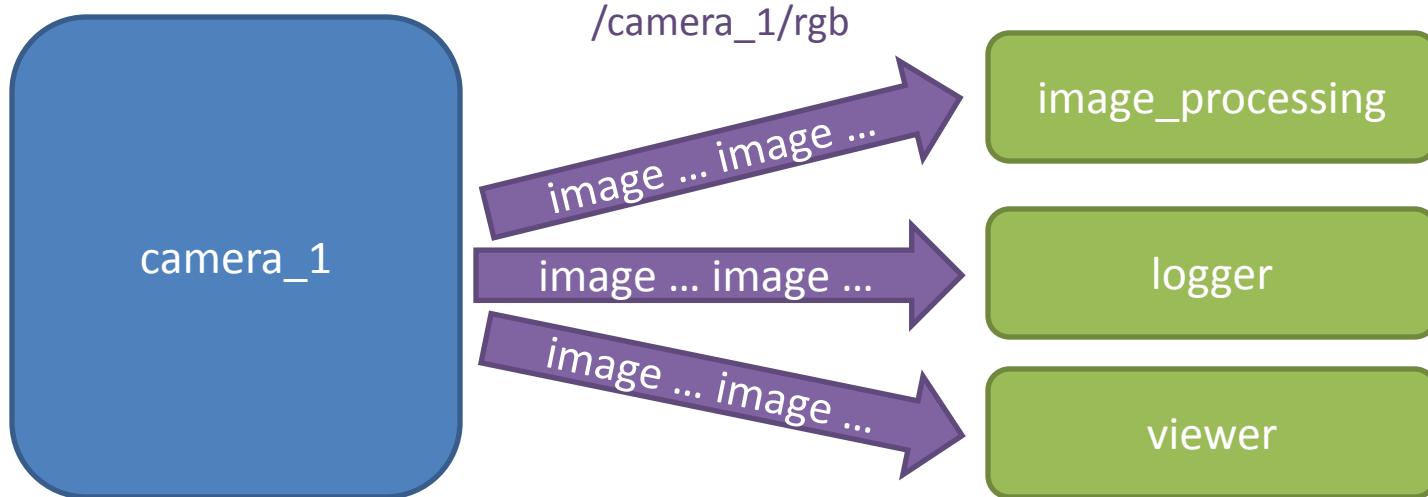
- Topics are **channels**, Messages are **data types**
 - Different topics can use the same Message type



Practical Example



- Many nodes can pub/sub to same topic
 - comms are direct node-to-node





Topics : Details



- Each **Topic** is a stream of **Messages**:
 - sent by **publisher(s)**, received by **subscriber(s)**
- Messages are **asynchronous**
 - publishers don't know if anyone's listening
 - messages may be dropped
 - subscribers are event-triggered (by incoming messages)
- Typical Uses:
 - Sensor Readings: camera images, distance, I/O
 - Feedback: robot status/position
 - Open-Loop Commands: desired position



February 2017





ROS Messages Types



- Similar to C structures
- Standard data primitives
 - Boolean: bool
 - Integer: int8, int16, int32, int64
 - Unsigned Integer: uint8, uint16, uint32, uint64
 - Floating Point: float32, float64
 - String: string
- Fixed length arrays: bool [16]
- Variable length arrays: int32 []
- Other: Nest message types for more complex data structure



February 2017





Message Description File



- All Messages are defined by a .msg file

PathPosition.msg

```
comment → # A 2D position and orientation  
other Msg type → Header header  
float64 x      # X coordinate  
float64 y      # Y coordinate  
float64 angle # Orientation
```

↑
data type

↑
field name



Custom ROS Messages



- Custom message types are defined in msg subfolder of packages
- Modify CMakeLists.txt to enable message generation.

Name	Size	Type
src	2 items	Folder
lesson_simple_topic	5 items	Folder
include	1 item	Folder
msg	1 item	Folder
PathPosition.msg	66 bytes	Text
src	2 items	Folder
package.xml	2.1 kB	Markup
CMakeLists.txt	4.1 kB	Text
CMakeLists.txt	2.1 kB	Link to Text



CMakeLists.txt



- Lines needed to generate custom msg types

```
find_package(catkin REQUIRED COMPONENTS  
message_generation)
```

```
generate_messages(DEPENDENCIES ...)
```

```
catkin_package(CATKIN_DEPENDS roscpp message_runtime)
```

```
add_dependencies(catkin REQUIRED COMPONENTS  
message_generation)
```



package.xml



```
<depend> message_generation </depend>
<exec_depend>message_runtime</exec_depend>
```



ROS Message Commands



- `rosmsg list`
 - Show all ROS topics currently installed on the system
- `rosmsg package <package>`
 - Show all ROS message types in package <package>
- `rosmsg show <package>/<message_type>`
 - Show the structure of the given message type



February 2017





ROS Topic Commands



- `rostopic list`
 - List all topics currently subscribed to and/or publishing
- `rostopic type <topic>`
 - Show the message type of the topic
- `rostopic info <topic>`
 - Show topic message type, subscribers, publishers, etc.
- `rostopic echo <topic>`
 - Echo messages published to the topic to the terminal
- `rostopic find <message_type>`
 - Find topics of the given message type

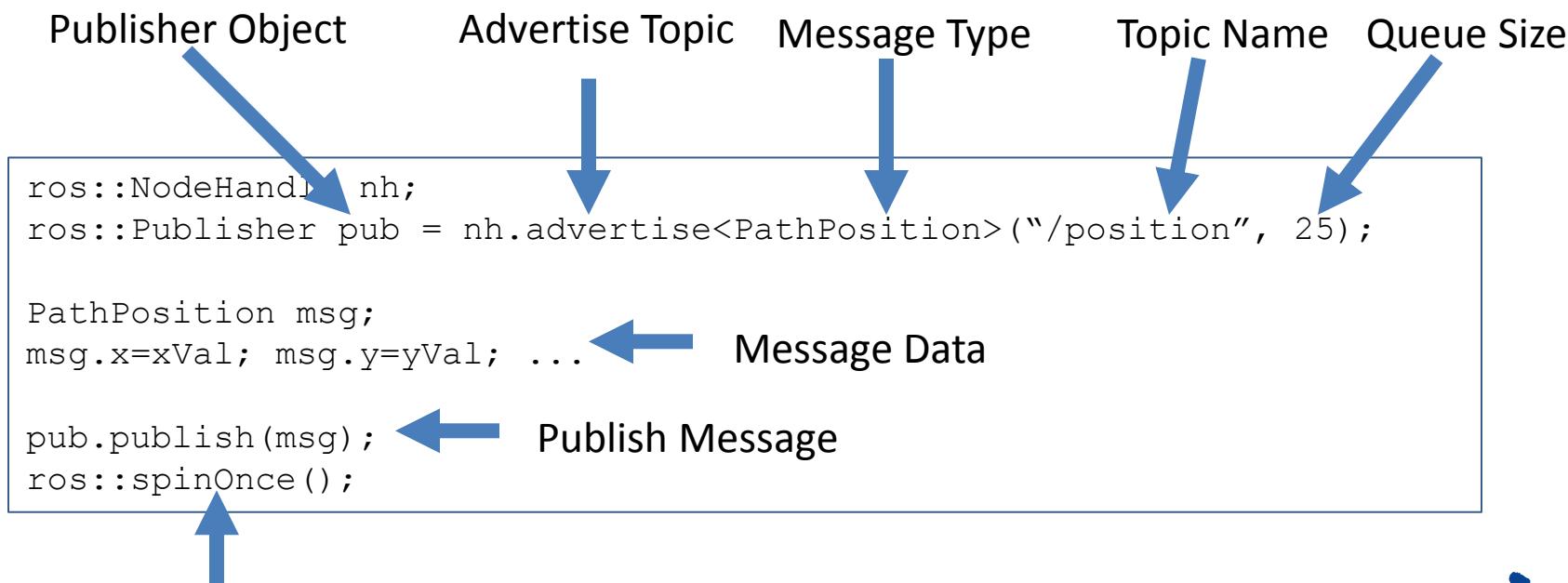


Topics: Syntax



- **Topic Publisher**

- Advertises available topic (*Name, Data Type*)
- Populates message data
- Periodically publishes new data





Topics: Syntax



- **Topic Subscriber**

- Defines callback function
- Listens for available topic (*Name, Data Type*)

```
Callback Function           Message Type           Message Data (IN)
↓                         ↓                         ↗
void msg_callback(const PathPosition& msg) {
    ROS_INFO_STREAM("Received msg: " << msg);
}

ros::Subscriber sub = nh.subscribe("/topic", 25, msg_callback);
```

↑ ↑ ↑
Server Object Service Name Callback Ref

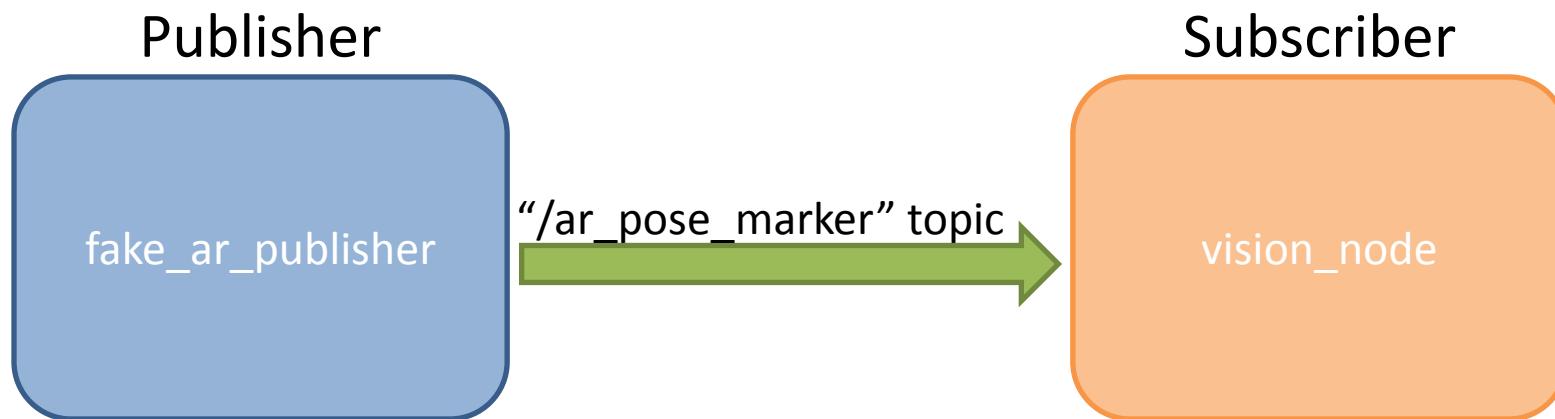


Exercise 1.2



Exercise 1.4

Subscribe to fake_ar_publisher





Contact Info.



Jeremy Zoss

SwRI
9503 W. Commerce
San Antonio, TX 78227
USA

Phone: 210-522-3089
Email: jzoss@swri.org



Levi Armstrong

SwRI
9503 W. Commerce
San Antonio, TX 78227
USA

Phone: 210-522-3801
Email: levi.armstrong@swri.org