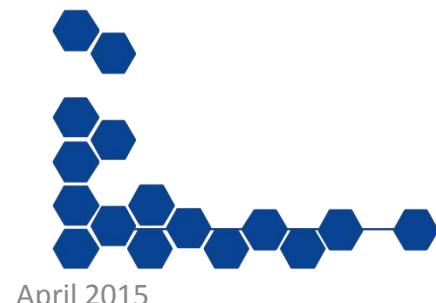




ROS-Industrial Basic Developer's Training Class

April 2015

Southwest Research Institute





Outline



- Intro to ROS
- Installing
- Packages / Nodes
- Parameters
- Messages / Topics



April 2015





An Introduction to ROS



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





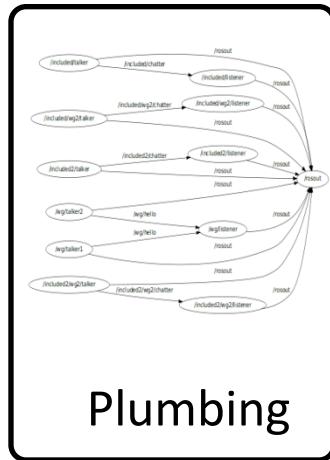
What is ROS?



ROS is...

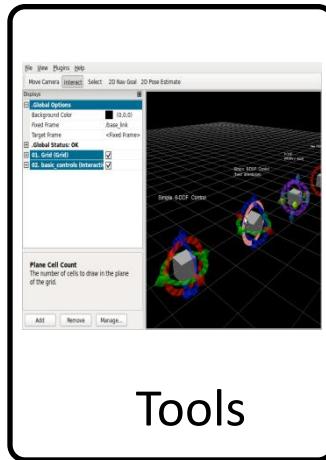


=



Plumbing

+



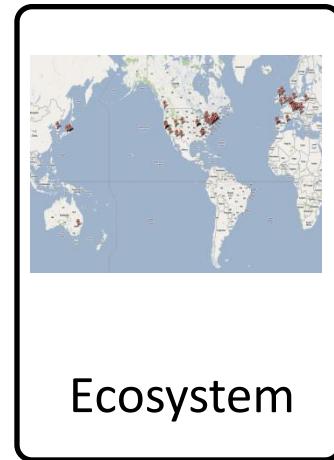
Tools

+



Capabilities

+



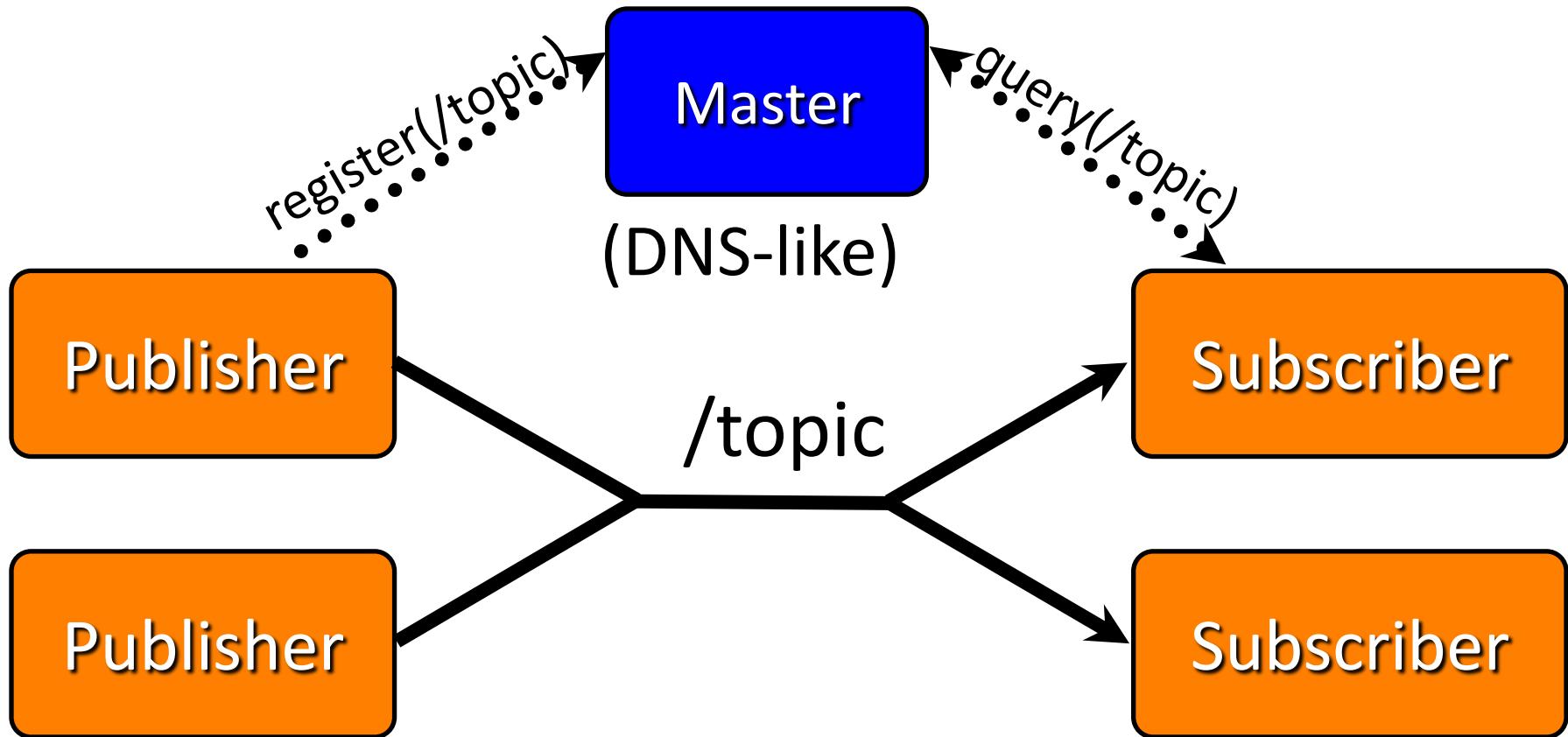
Ecosystem

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





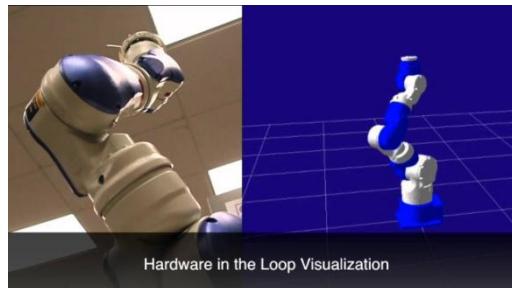
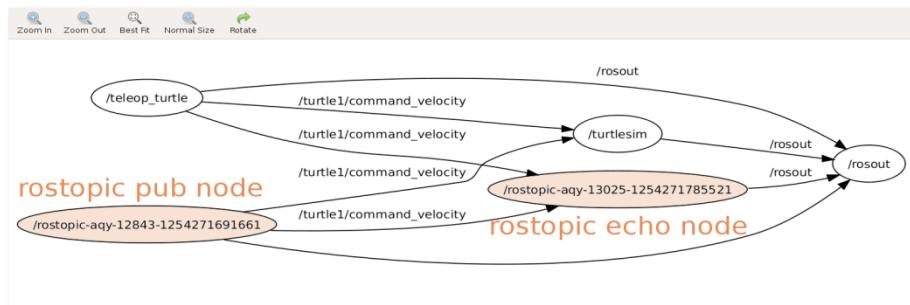
ROS is... plumbing



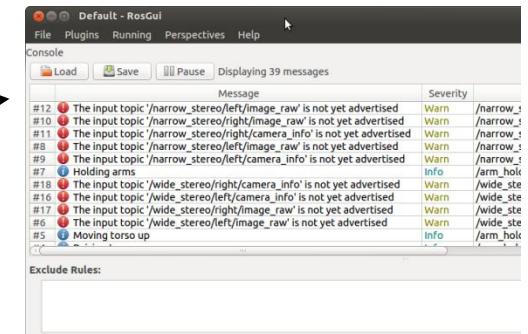
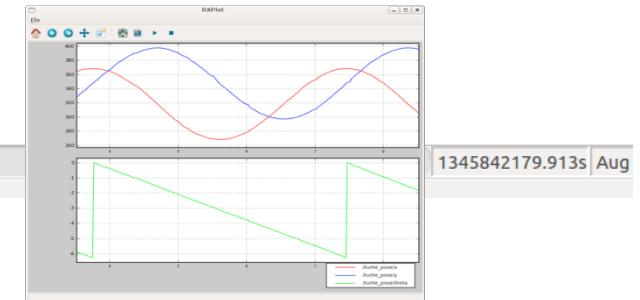
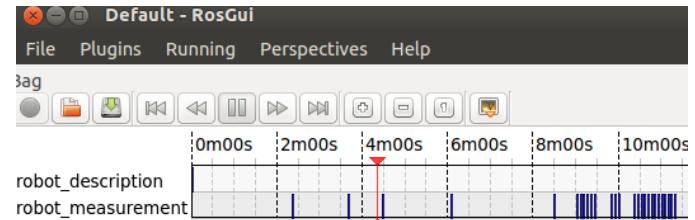
(Adapted from Willow Garage's "What is ROS?" Presentation)



ROS is ... Tools



- logging/plotting
- graph visualization
- diagnostics
- visualization

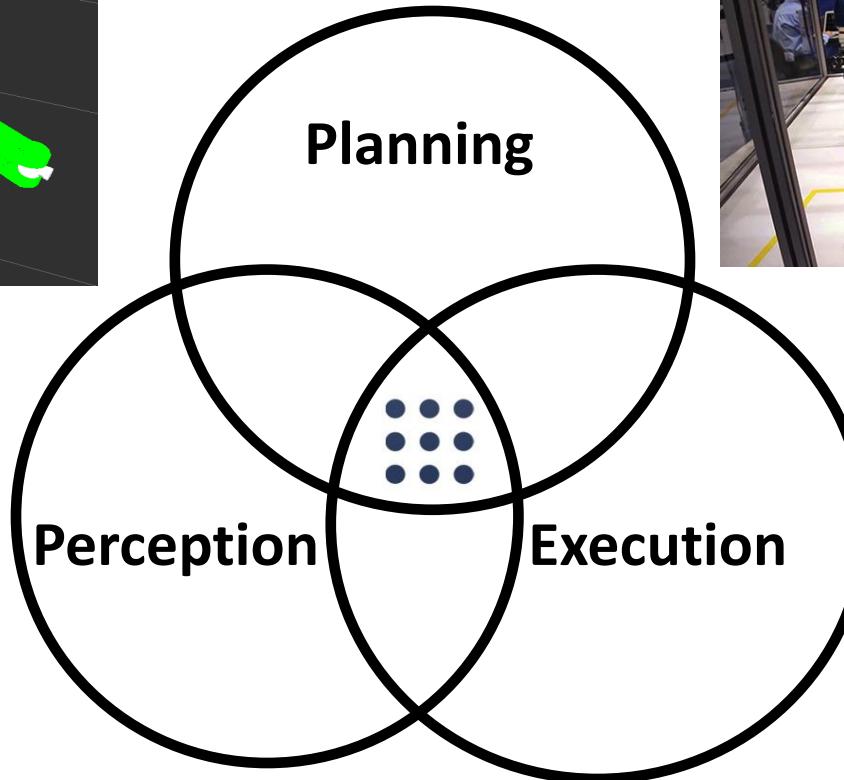
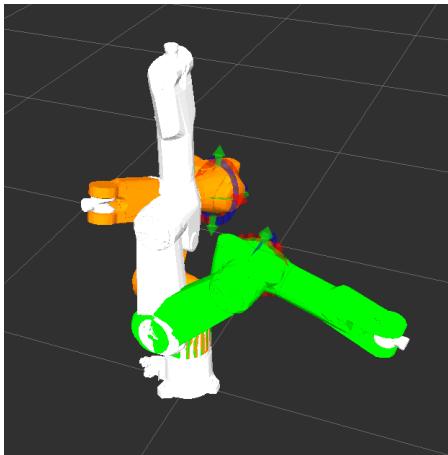


(Adapted from Willow Garage's "What is ROS?" Presentation)





ROS is...Capabilities

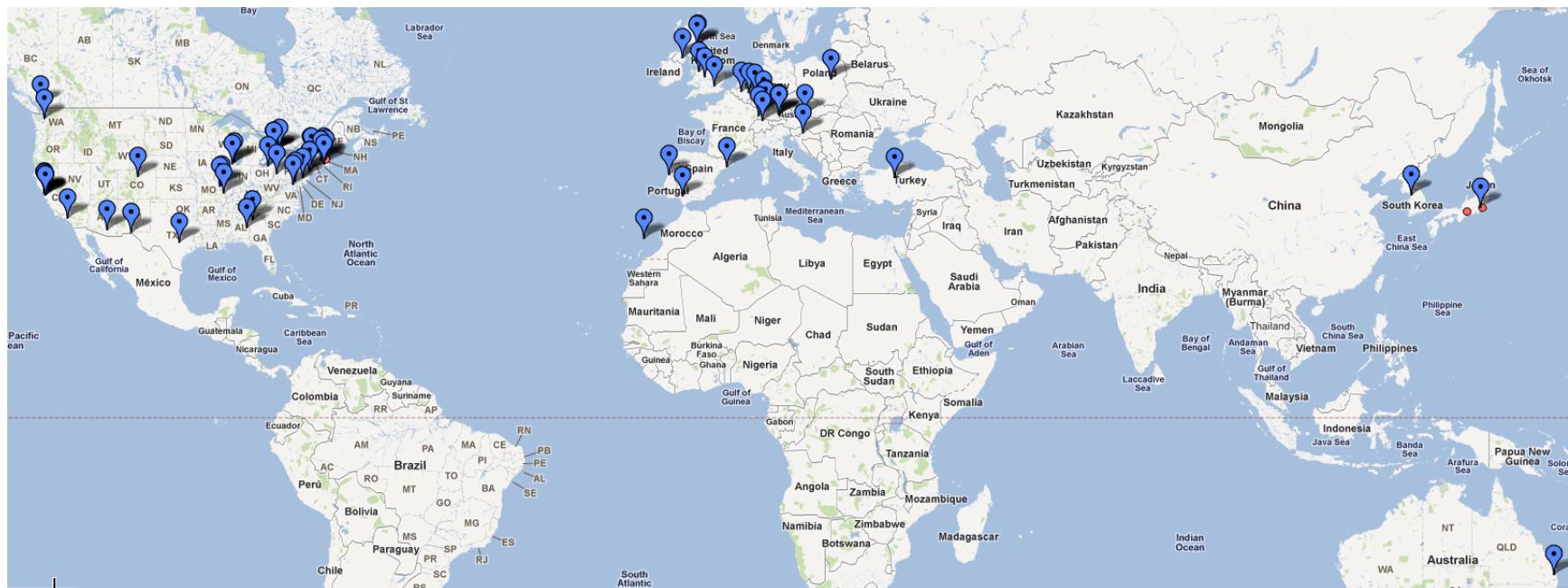


(Adapted from Willow Garage's "What is ROS?" Presentation)





ROS is... an Ecosystem



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

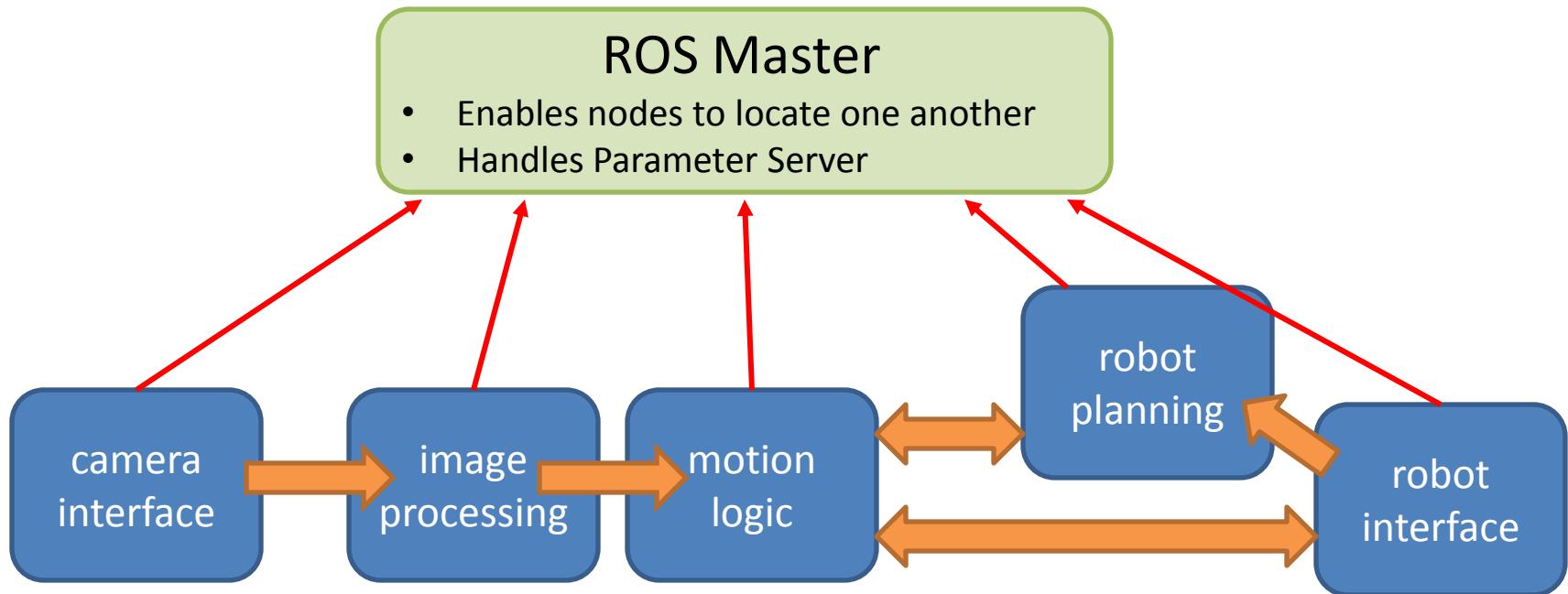


April 2015





ROS Architecture: Nodes

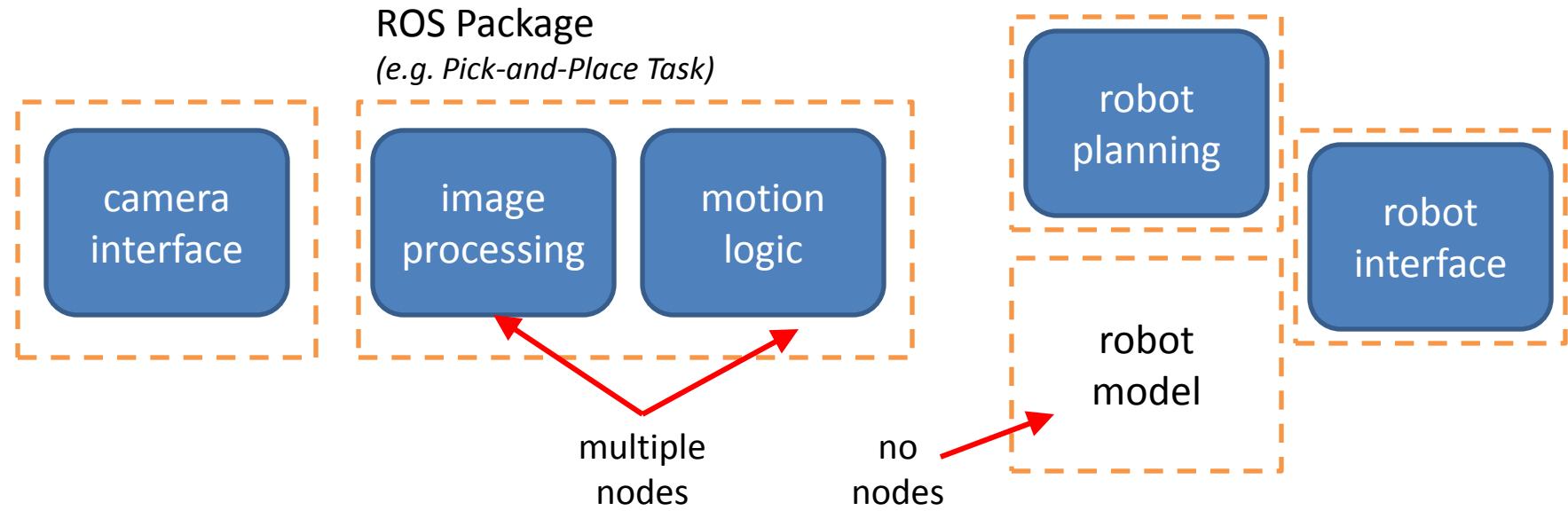


- A **Node** is a single ROS-enabled program
 - Most communication happens **between** nodes
 - Nodes can run on many different **devices**
- One **Master** per system





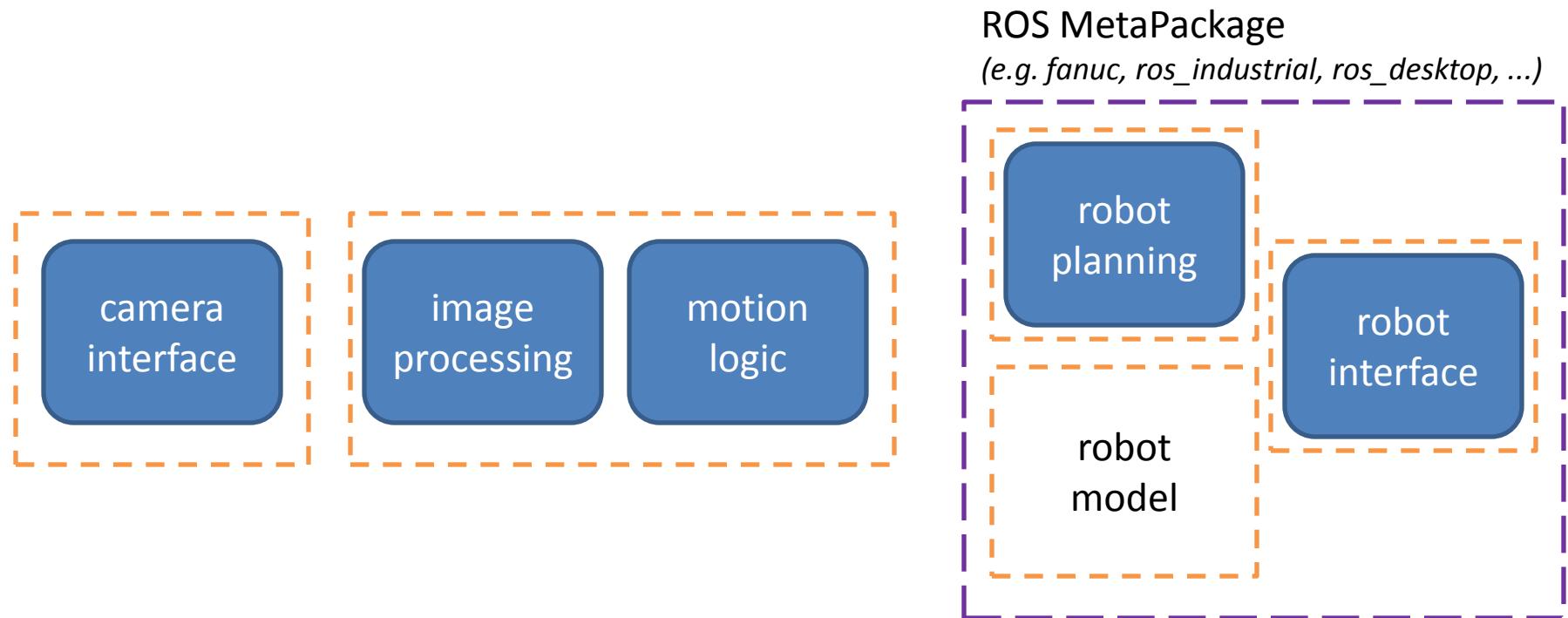
ROS Architecture: Packages



- **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





ROS Programming

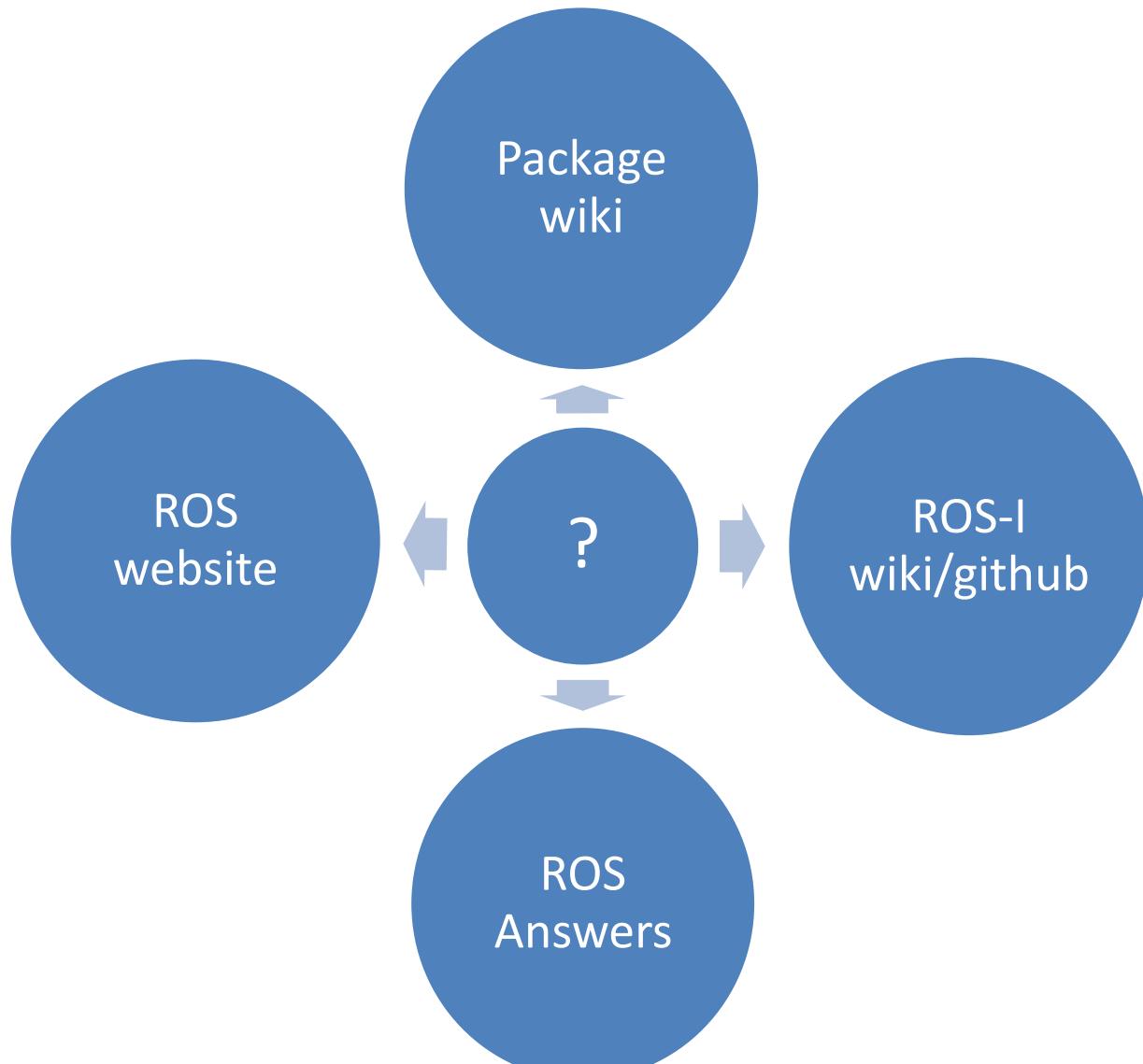


- ROS uses **platform-agnostic** methods for most communication
 - TCP/IP Sockets, XML, etc.
- Can intermix programming languages
 - currently: C++, Python, Lisp
 - We will be using C++ for our exercises





ROS Resources



April 2015





ROS.org Website



<http://ros.org>

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.

Read More

INDIGO
IGLOO

Indigo Igloo is the 8th official ROS release. It is supported on Ubuntu Saucy and Ubuntu Trusty. Get Indigo Igloo Now!

Download

ROS Spotlight: Jade logo and t-shirt campaign

Wiki Find tutorials and learn more

ROS Answers Ask questions. Get answers

Blog Stay up-to-date

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





Package Wiki



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

tf
electric fuerte groovy hydro indigo jade Documentation Status

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

1. What does tf do? Why should I use tf?
2. Paper
3. Tutorials
4. Code API Overview
5. Frequently asked questions
6. Command-Line 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



April 2015





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) posted 54 mins ago by [fleete](#)

pcl 1.7.2 installation question (9 views) posted 1 hour ago by [mrunzista](#)

freenect_launch with Kinect (3 views) posted 1 hour ago by [seny](#)

Problem using serial write (14 views) posted 2 hours ago by [NightGenie](#)

schunk_svh_driver : Can't locate node svh_controller in package schunk_svh_driver (8 views) posted 4 hours ago by [gvdhoorn](#)

Broken url in tutorial (9 views) posted 4 hours ago by [kerker](#)

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? (0 answers, 0 views)

I have Ubuntu 14.04

asked Aug 26 '14 updated Aug 26 '14 viewed 19 - 40 http://ros-industrial.tut... edited

update Aug 26 '14 posted 19 - 40 http://ros-industrial.tut... updated 12 hours ago

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 urdf/xacro 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 Blocks 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 ..
rosdep install --from-paths src --ignore-src --rospkgdir 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!





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!

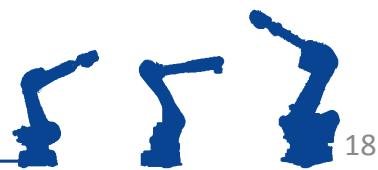




Installing ROS



April 2015





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



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!)



Finding the Right Package

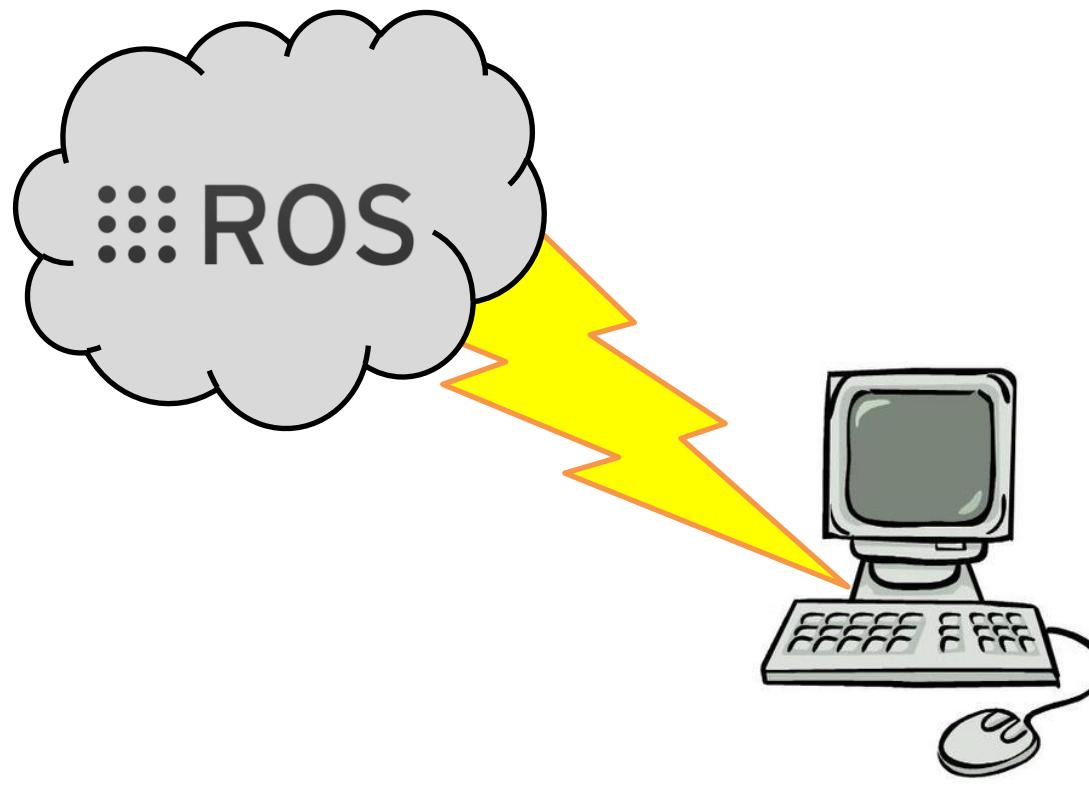


- ROS Website (<http://ros.org/browse/>)
 - Browse/Search for known packages
- ROS Answers (<http://answers.ros.org>)
 - When in doubt... ask someone!
- apt-cache search <search term>
 - Searches titles and descriptions for search term



Exercise 1.1

Basic ROS Install/Setup





Installing from Source



- 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

```
[ 80%] Building C object enu/src/libswiftnav/src/CMakeFiles/swiftnav-static.dir/almanac.c.o
[ 84%] Building C object enu/src/libswiftnav/src/CMakeFiles/swiftnav.dir/gpstimer.c.o
[ 88%] Building C object enu/src/libswiftnav/src/CMakeFiles/swiftnav.dir/gpstimer.c.o
Linking C shared lib
Linking C static lib
[ 88%] Built target
[ 88%] Built target
Scanning dependencies
[ 92%] Building CXX
Linking CXX shared lib
[ 92%] Built target
Scanning dependencies
Scanning dependencies
[ 96%] Building CXX
[100%] Building CXX
Linking CXX executable /home/rgartepy/ros/enu-devel/lib/enu/from_fix
src/to_fix.cpp.o
src/from_fix.cpp.o
```

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



April 2015

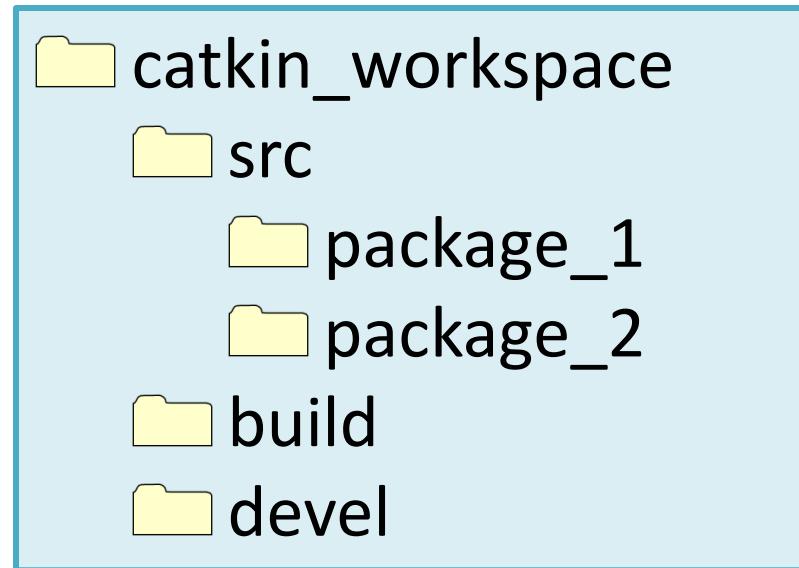




Catkin Workspace



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





Setup (one-time)

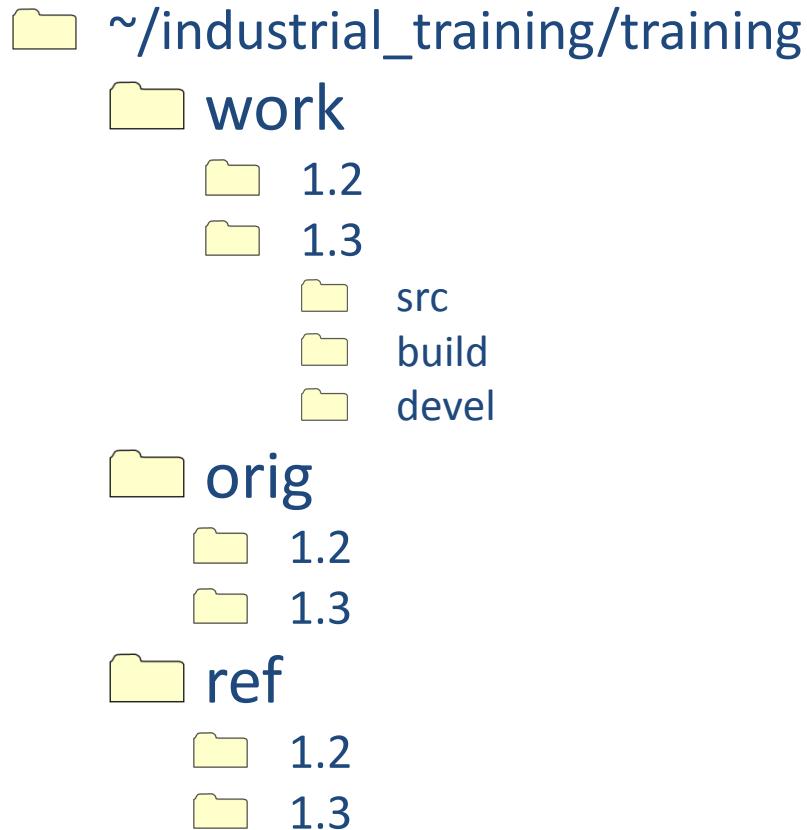
1. Create a catkin workspace somewhere
 - my_catkin_ws/src
 - build, devel directories created automatically
2. Run `catkin_init_workspace` in `src` subdir
3. Download/create packages in `src` subdir

Compile-Time

1. Run `catkin_make` in **workspace root**
2. Run `source devel/setup.bash` to make workspace visible to ROS
 - Must re-execute in **each** new terminal window
 - Most ROS-I training units automate this process



Training Exercise Layout



- Main sub-directories:
 - **work** : working area
 - **orig** : copy of “work”
 - **ref** : completed exercises
- **training_unit 1.2**
 - change directory
 - **catkin_make**
 - source catkin workspace
- **Also:**
 - **training_ref 1.2**
 - **clear_training_unit**



April 2015





Exercise 1.2



Exercise 1.2

Install from Source

ros-industrial / **industrial_core**

Unwatch 14 stars

202 commits 5 branches 7 releases 10 contributors

branch: **indigo-devel** / +

Merge pull request #106 from shaun-edwards/issue89_simulator_depends

shaun-edwards authored on Mar 21 latest commit ac7588e187

industrial_core	"0.3.4"	a year ago
industrial_deprecated	"0.3.4"	a year ago
industrial_msgs	"0.3.4"	a year ago
industrial_robot_client	Fill stamp of the RobotStatus message Fix: #97	4 months ago
industrial_robot_simulator	Fixed roslaunch test dependency and build depends for robot simulator...	a month ago
industrial_trajectory_filters	Merge pull request #79 from ros-industrial/hydro	10 months ago
industrial_utils	Silent warnings	6 months ago

```
Generating Python msg __init__.py for robotiq_c_model_control
[ target robotiq_c_model_control_generate_messages_py
  dependencies of target robotiq_c_model_control_generate_messages_py
  C++ code from robotiq_c_model_control/CModel_robot_input.msg
  Generating Lisp code from robotiq_c_model_control/CModel_robot_output.msg
  Generating Lisp code from robotiq_c_model_control/CModel_robot_input.msg
[ target robotiq_c_model_control_generate_messages_lisp
  dependencies of target robotiq_c_model_control_generate_messages_lisp
  C++ code from robotiq_s_model_control/SModel_robot_output.msg
  Generating Lisp code from robotiq_s_model_control/SModel_robot_output.msg
  Generating Lisp code from robotiq_s_model_control/SModel_robot_input.msg
[ target robotiq_s_model_control_generate_messages_lisp
  dependencies of target robotiq_s_model_control_generate_messages_lisp
  C++ code from robotiq_s_model_control/SModel_robot_input.msg
  Generating Python from MSG robotiq_s_model_control/SModel_robot_output
  [ 86%] Generating Python msg __init__.py for robotiq_s_model_control
  [ 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
```



April 2015





ROS Packages

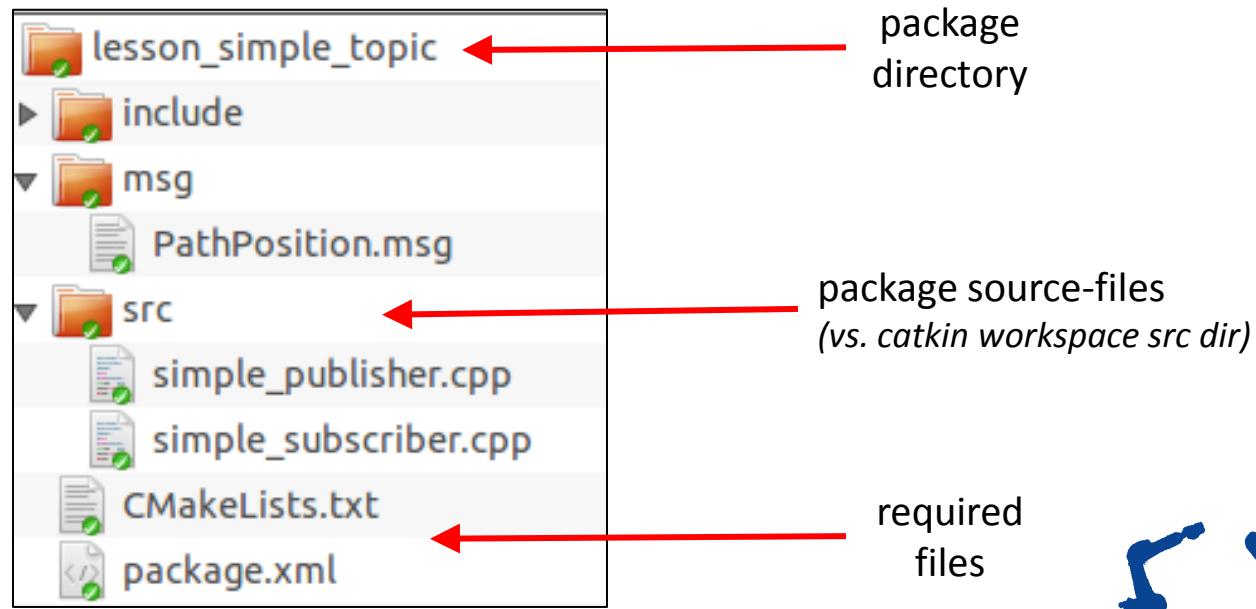




ROS Package Contents



- 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:
 - <build_depend> – needed to **compile** package
 - <run_depend> – needed to **run** package

```
<build_depend>roscpp</build_depend>
<build_depend>industrial_robot_client</build_depend>
<build_depend>simple_message</build_depend>

<run_depend>roscpp</run_depend>
<run_depend>industrial_robot_client</run_depend>
```



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



Create New Package



```
catkin-create-pkg 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



ROS Package Commands



- `roscd package_name`

Change to package directory

- **rospack**

- `rospack find package_name`

Find directory of package_name

- `rospack list`

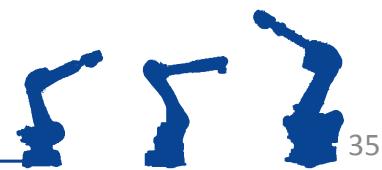
List all ros packages installed

- `rospack depends package_name`

List all dependencies of package_name



Nodes





A Simple C++ ROS Node



Simple C++ Program

```
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
➤ *Ctrl+C is usually easier*



Exercise 1.4



Exercise 1.4

Running a Node

The screenshot shows a terminal window titled "ROS-I Training Unit 1.4 (work)" and a code editor window for "simple_node.cpp".

Terminal Output:

```
ROS-I Training Unit 1.4 (work)  x  roscorehttp://ROS:11311/  x  ROS-I Training Unit 1.4 (work)
ros-industrial@ROS:~/industrial_training/training/work/1.4$ rosrun
_node simple_node
[ INFO] [1429803566.005091727]: We've gone through 0 times.
[ INFO] [1429803567.006018146]: We've gone through 1 times.
[ INFO] [1429803568.005276732]: We've gone through 2 times.
[ INFO] [1429803569.005580592]: We've gone through 3 times.
[ INFO] [1429803570.006208205]: We've gone through 4 times.
[ INFO] [1429803571.006119762]: We've gone through 5 times.
[ INFO] [1429803572.006209800]: We've gone through 6 times.
[ INFO] [1429803573.005258377]: We've gone through 7 times.
[ INFO] [1429803574.007330789]: We've gone through 8 times.
[ INFO] [1429803575.005209314]: We've gone through 9 times.
[ INFO] [1429803576.005255745]: We've gone through 10 times.
[ INFO] [1429803577.005235965]: We've gone through 11 times.
[ INFO] [1429803578.0060072995]: We've gone through 12 times.
[ INFO] [1429803579.005188723]: We've gone through 13 times.
[ INFO] [1429803580.005236333]: We've gone through 14 times.
```

Code Editor Content (simple_node.cpp):

```
1 // Simple ROS node
2
3 /**
4  * \include <ros/ros.h>
5
6
7 int main(int argc, char* argv[])
8 {
9     // This must be called before anything else ROS-related
10    ros::init(argc, argv, "simple_node");
11
12    // Create a ROS node handle
13    ros::NodeHandle node;
14
15    // Set the rate at which we print out our message (1Hz)
16    ros::Rate loop_rate(1.0);
17
18    // A simple counter for the number of times we iterate through the loop
19    int count = 0;
20
21    // Loop through until the ROS system tells the user to shut down
22    while(ros::ok()) {
23        // Print out a message
24        ROS_INFO_STREAM("We've gone through " << count << " times.");
25        ++count;
26        // Wait the stated duration
27        loop_rate.sleep();
28    }
29
30    // Exit the program.
31    return 0;
32 }
```



April 2015

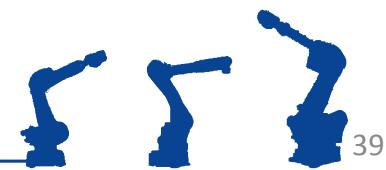




Parameters



April 2015



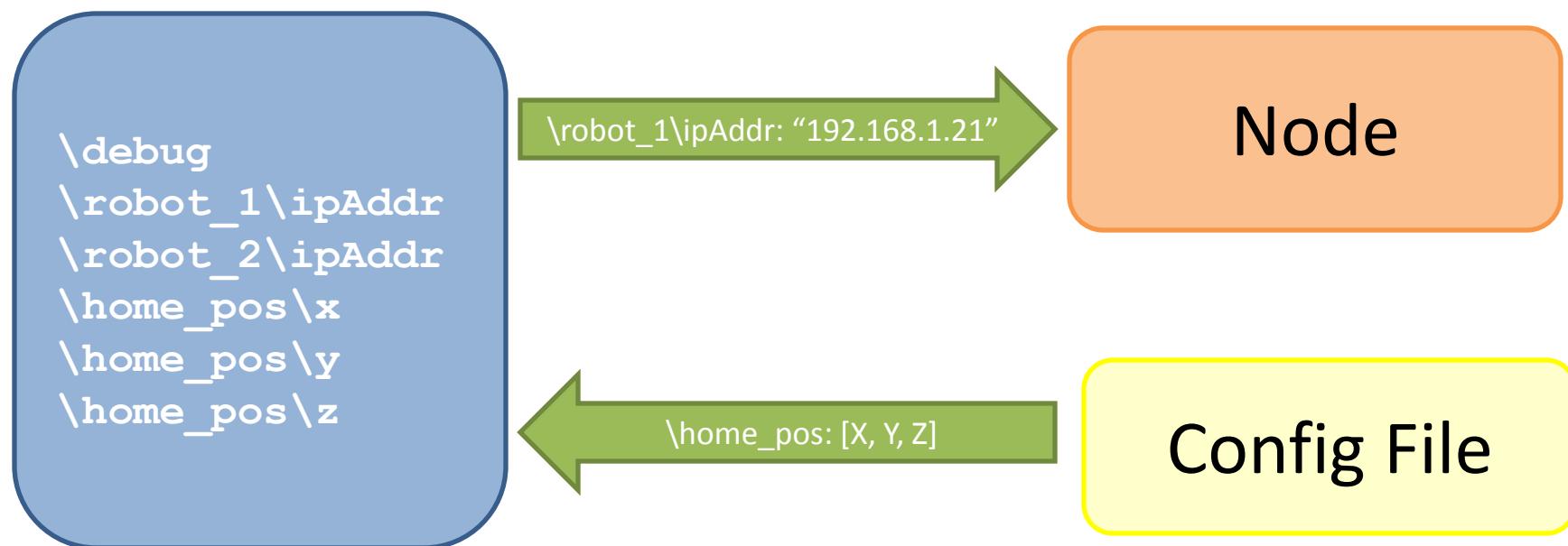


Parameters: Overview



Parameters are like **Global Data**

Parameter Server

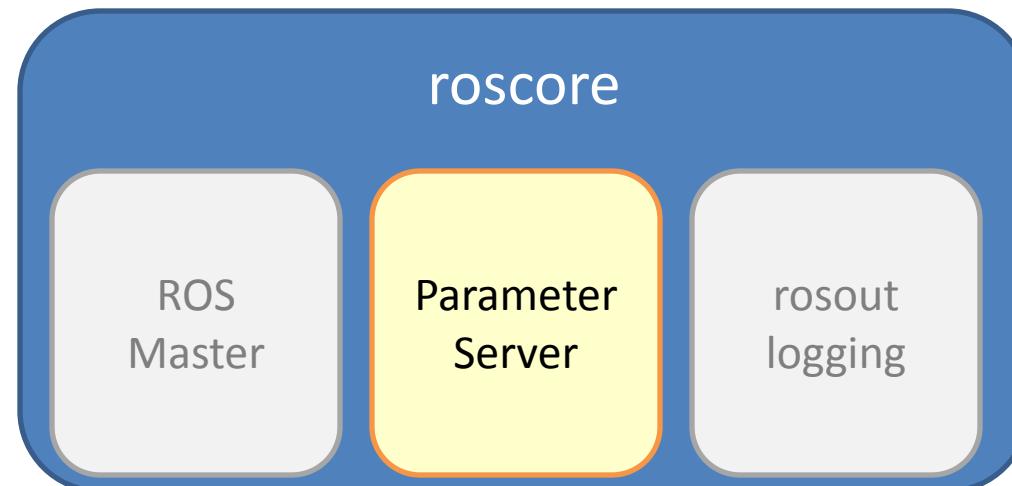




ROS Parameters



- Typically configuration-type values
 - robot kinematics
 - workcell description
 - algorithm limits / tuning
- Accessed through the **Parameter Server**.
 - *Typically handled by `roscore`*





Setting Parameters



- Can set from:

- YAML Files

```
manipulator_kinematics:  
  solver: kdl_plugin/KDLKinematics  
  search_resolution: 0.005  
  timeout: 0.005  
  attempts: 3
```

- Command Line

```
rosrun my_pkg load_robot _ip:="192.168.1.21"  
rosparam set "/debug" true
```

- Programs

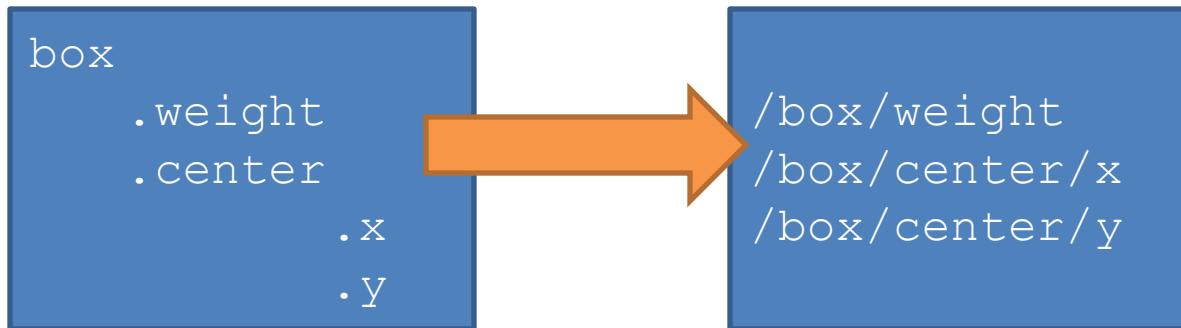
```
nh.setParam("name", "left");
```



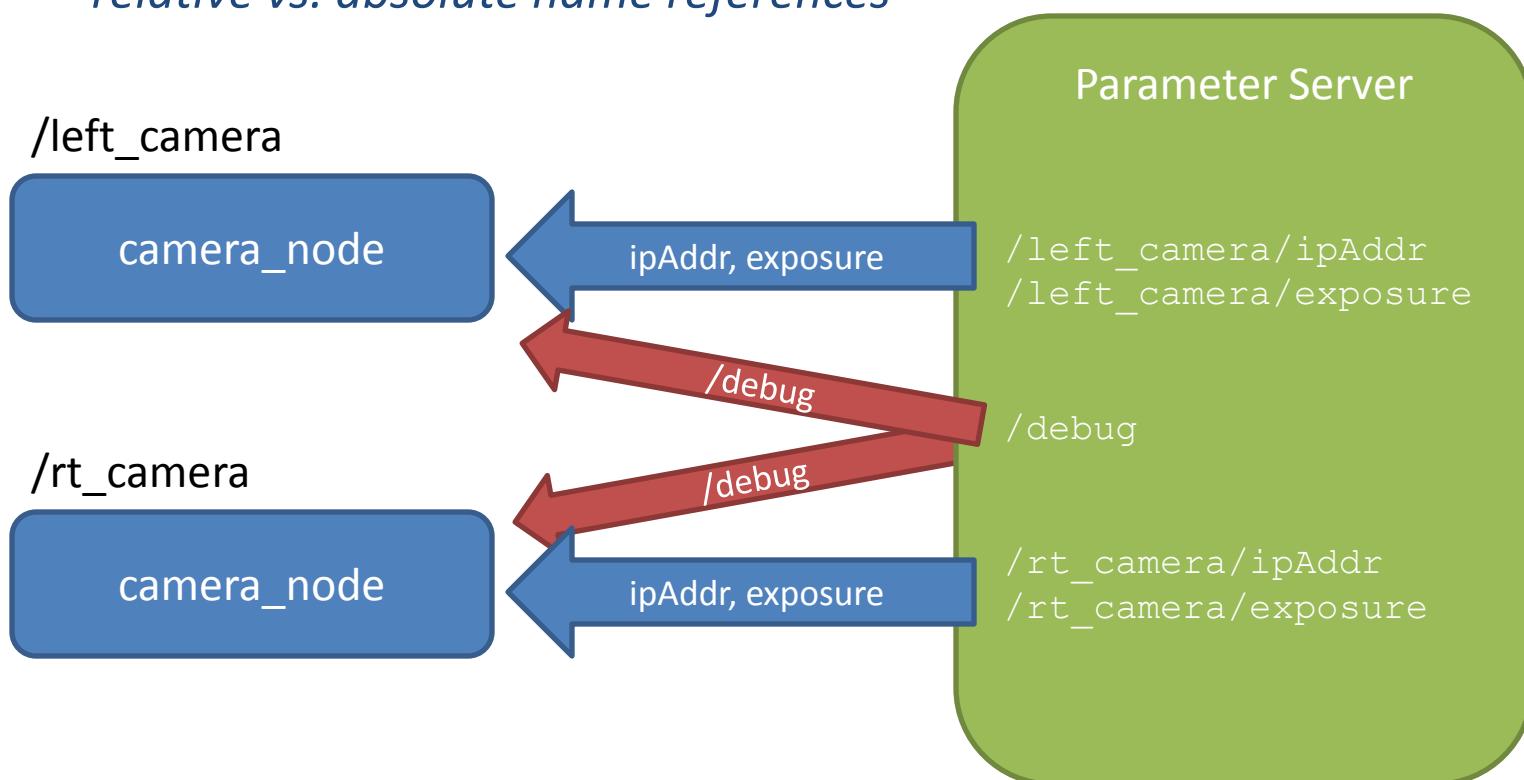
Parameter Datatypes



- Native Types
 - *int, real, boolean, string*
- Lists (vectors)
 - *can be mixed type: [1, str, 3.14159]*
 - *but typically of single type: [1.1, 1.2, 1.3]*
- Dictionaries (structures)
 - *translated to “folder” hierarchy on server*



- **Folder Hierarchy allows Separation:**
 - Separate nodes can co-exist, in different “namespaces”
 - relative vs. absolute name references





Parameter Commands



- **rosparam**

- rosparam set <key> <value>
 - Set parameters
- rosparam get <key>
 - Get parameters
- rosparam delete <key>
 - Delete parameters
- rosparam list
 - List all parameters currently set
- rosparam load <filename> [<namespace>]
 - Load parameters from file





Parameters: C++ API



- Accessed through `ros::NodeHandle` object
 - also sets default **Namespace** for access
 - Global (root) namespace:

```
ros::NodeHandle global();  
global.getParam("test");
```



"/test"

- Fixed namespace:

```
ros::NodeHandle fixed("/myApp");  
global.getParam("test");
```



"/myApp/test"

- Private namespace:

```
ros::NodeHandle priv("~");  
global.getParam("test");
```



"/myNode/test"



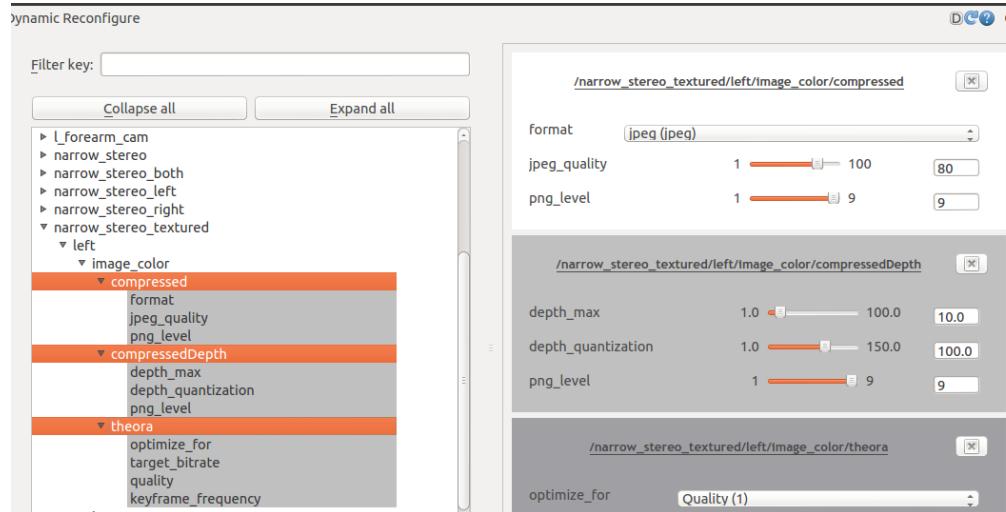
Parameters: C++ API (cont'd)



- NodeHandle object methods
 - `nh.hasParam(key)`
Returns true if parameter exists
 - `nh.getParam(key, &value)`
Gets value, returns T/F if exists.
 - `nh.param(key, &value, default)`
Get value (or default, if doesn't exist)
 - `nh.setParam(key, value)`
Sets value
 - `nh.deleteParam(key)`
Deletes parameter



- Parameters must be read explicitly by nodes
 - no on-the-fly updating
 - typically read only when node first started
- ROS package `dynamic_reconfigure` can help
 - nodes can register callbacks to trigger on change
 - outside the scope of this class, but useful





Exercise 1.5



Exercise 1.5

ROS Parameters

watch as I walk through this example...

```
/global_integer_value  
/rosdistro  
/roslaunch/uris/host_ros_industrial_  
/rosversion  
/run_id  
/simple_parameters/integer_value  
/simple_parameters/point/x  
/simple_parameters/point/y  
/simple_parameters/simple_string
```





Topics and Messages

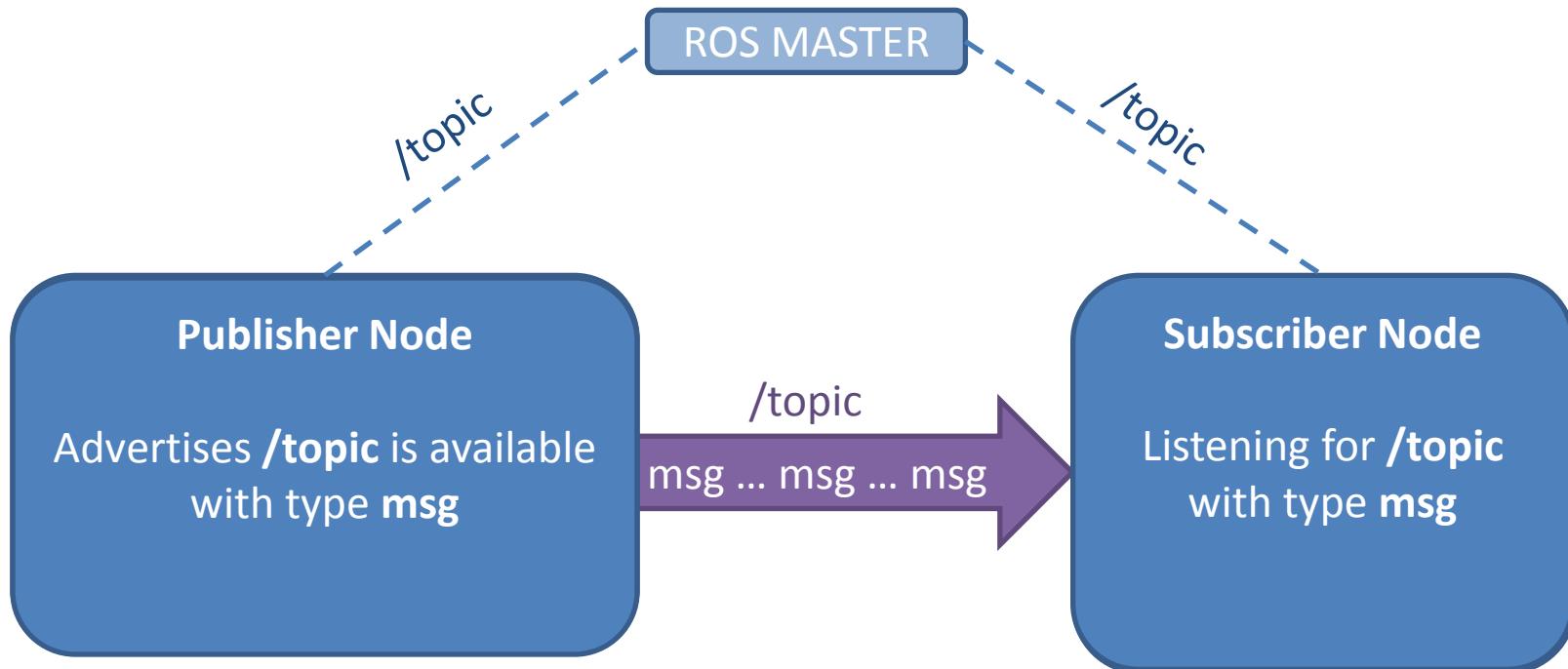




ROS Topics/Messages



Topics are for **Streaming Data**





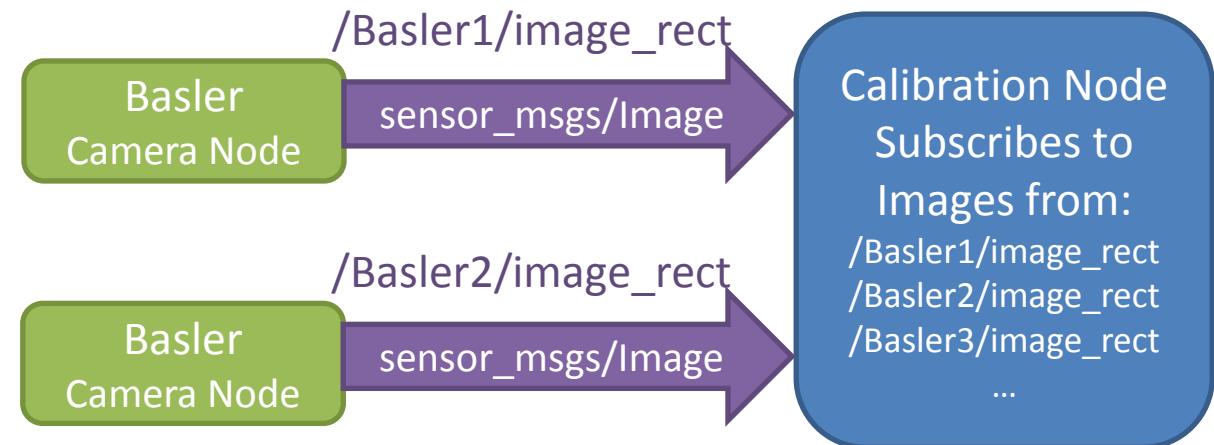
Topics vs. Messages



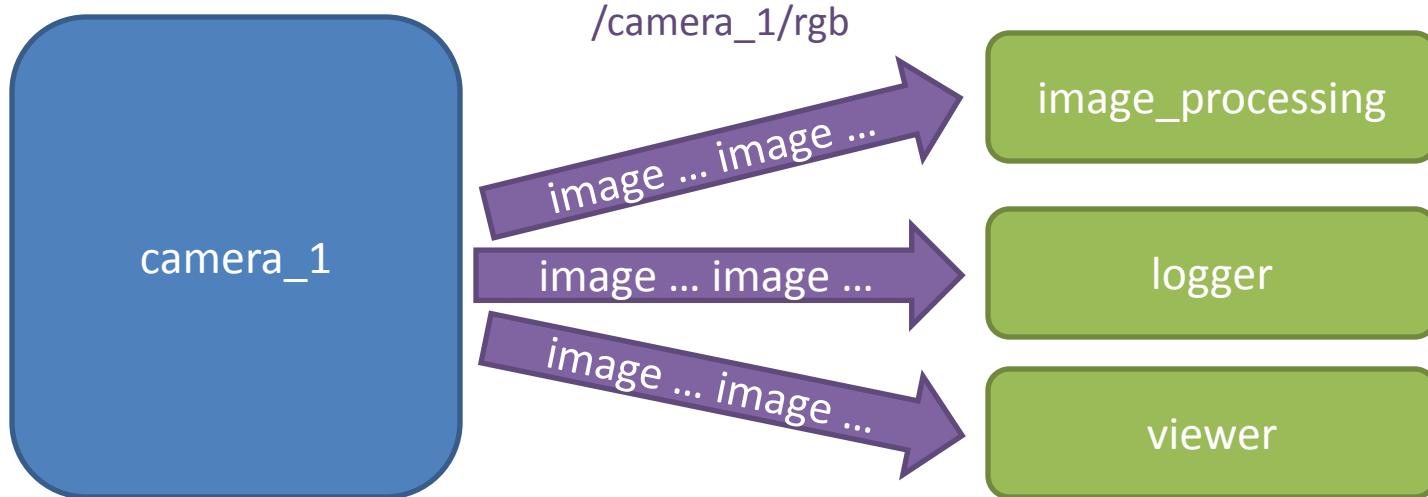
- Topics are **channels**, Messages are **data types**
 - Different topics can use the same Message type



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





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





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



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



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



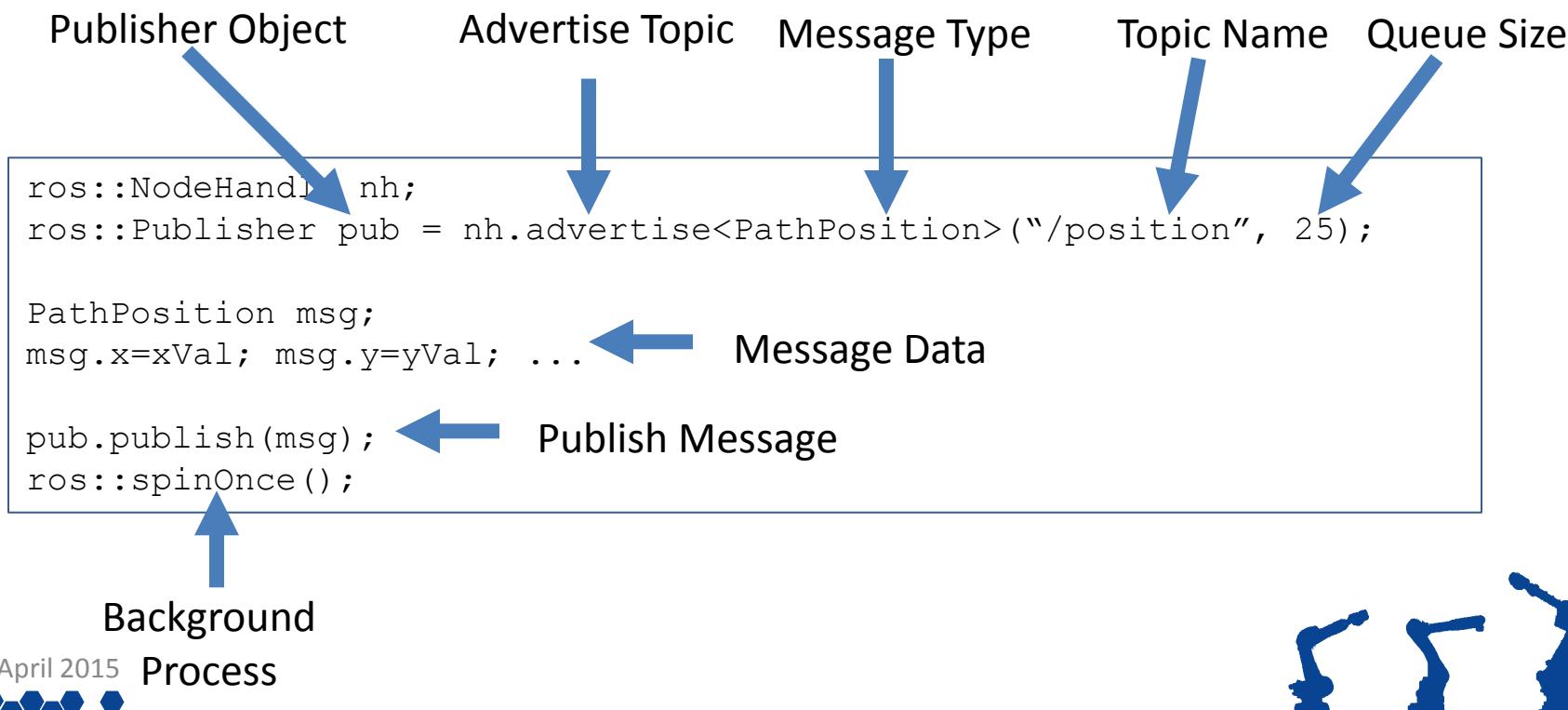


Topics: Syntax



- **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

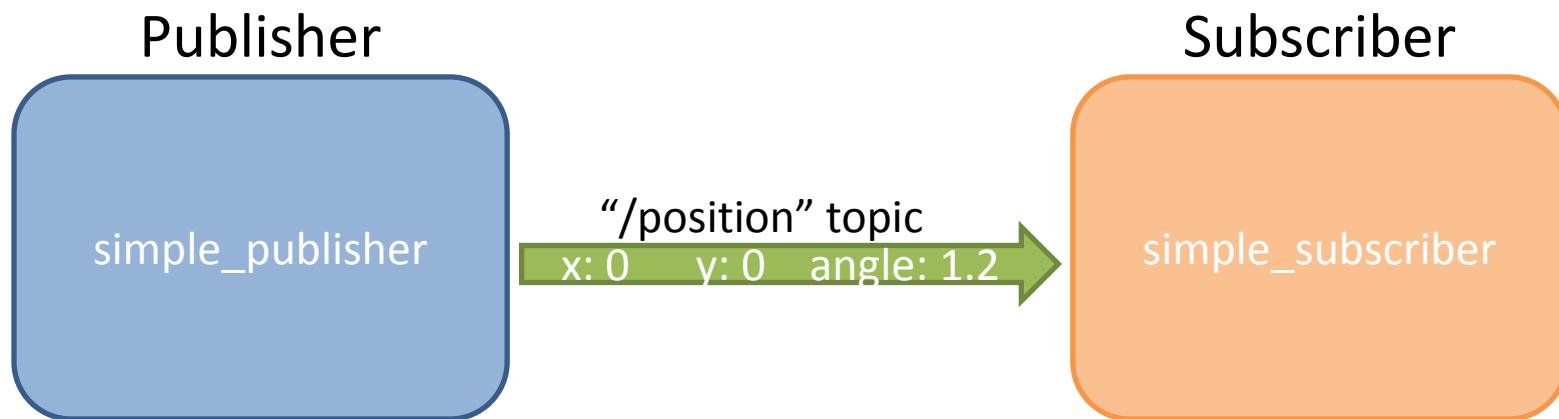


April 2015



Exercise 1.6

ROS Topics and Messages





Contact Info.



Jeremy Zoss
Principal Engineer

SwRI
9503 W. Commerce
San Antonio, TX 78227
USA

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