AA 274 Principles of Robotic Autonomy

The Robot Operating System (ROS)





Writing Software for Robotics

- Robotics requires very complex software
- The software you will deal with in AA274A has *way* more moving parts than what you've dealt with in most other classes...



Writing Software for Robotics

- We deal with the complexity through **modularity**
- We enable modularity by following the right **design pattern:** "a general, reusable solution to a commonly occurring problem within a given context in software design" Wikipedia

The Pub/Sub Design Pattern

- We divide our software into individual components
- We define "topics" (think chat rooms) where components can broadcast information to anyone listening
- Each component can:
 - Publish: send messages to a topic regardless of whether someone is listening or not
 - Subscribe: receive messages on a topic if anyone is sending them regardless of who

The Pub/Sub Design Pattern



Note: there are countless ways to **IMPLEMENT** pub/sub!

The Pub/Sub Design Pattern



Note: there are countless ways to **IMPLEMENT** pub/sub!

You already use Pub/Sub every day! Where???

6

Alternatives to Pub/Sub

- Request/Reply (RPC)
- Push/Pull
- Data binding (e.g. shared data members)
- Observers

What is ROS?

Depending on who you are talking to...

- An implementation of pub/sub geared towards robotic applications and that is network-aware
- Lots of open-source software shared by the community:
 - SLAM (gmapping, amcl)
 - Vision (OpenCV, PCL, OpenNI)
 - Arm Navigation (Movelt)
 - Simulation (Gazebo)

Are there "Alternatives" to ROS?

- LCM
- Drake
- Player
- YARP
- Orocos
- MRPT
- And many others!









Why is ROS popular in industry?

- Not reinventing the wheel is generally good
- Robotics is hard! It's great to offload some of the work to smart people
- ROS is now 12 years old and still going strong



Why are we using ROS in AA274?

- The closest thing we have to an "industry standard"
- It's an insurance policy for you (stability, online teaching resources)

:::ROS.org	About Support Status answers.ros.org	Se	earch: Submit
Documentation	Browse Software	News	Download
Documentation ROS (Robot Operating System) provides li	braries and tools to help software developers create ro	bot applications. It	Millei
provides hardware abstraction, device drive ROS is licensed under an open source, BS	ers, libraries, visualizers, message-passing, package n D license.	nanagement, and more.	Distributions ROS/Installation
Available Translations: German Spanish Русский (Russian) Thai Turkish 简体中	French Italian Japanese Korean Brazilian Portugi 文	uese Portuguese	ROS/Tutorials RecentChanges
ROS:			Page
Install ROS on your machine. Getting Started			Immutable Page Info Attachments
Learn about various concepts, Tutorials	client libraries, and technical overview of ROS.		More Actions:
Step-by-step instructions for le Contribute	arning ROS hands-on		Login
How to get involved with the R Support	OS community, such as submitting your own repository	y.	
What to do if something doesn Software:	/t work as expected.		
Distributions View the different release Dist	ributions for BOS		
Packages Search the 2000+ software lib	raries available for ROS.		
Core Libraries APIs by language and topic.			
Common Tools Common tools for developing	and debugging ROS software.		





ROS – Robot Operating System

- 2007-Today
 - Stanford AI Robot (STAIR)
 - Willow Garage founded by Scott Hassan (eGroups, Google, Stanford Digital Libraries)
 - Willow awards 11 \$400k PR2 robots to Universities
 - OSRF (Open Source Robotics Foundation) created to maintain ROS and Gazebo
 - ROS is everywhere!

ROS Integrates Existing Projects

- OpenCV (computer vision)
- Stage, Gazebo (simulation)
- OpenSLAM (navigation)
- Orocos KDL (arm navigation)
- Many ROS "wrappers" to existing software

The Main Software Components

1) Master

2) Nodes

- Nodes talk to each other over topics (think chat rooms). Master coordinates the whole thing
- Message types: abstraction away from specific hardware
 - Camera image
 - Laser scan data
 - Motion control

ROS Node

- A process (typically Python or C++) that runs some computation
- The "fundamental" building block
- Can act as a subscriber, publisher or both
- Nodes talk to each other over "topics"
- Run them using rosrun <package> <node>
- Initialize using rospy.init_node()

Note: nodelets are different. They are not individual processes, they share memory

Node Examples

Sensors and actuators are wrapped in self-contained, reusable software containers called "nodes"



Node Examples

Higher level operations also become nodes in the ROS computational architecture



More Concrete Node Examples

- LiDAR node publishes laser scan arrays
- Camera node publishes RGB images (+depth if RGBD) and camera info (resolution, distortion coefficients)
- Mobile robot controller publishes odometry values (e.g. x-y coordinates and velocities, +z for UAVs or underwater vehicles)
- Navigation node subscribes to LiDAR and odometry messages, publishes motion control messages

ROS Master

- A process that is in charge of coordinating nodes, publishers and subscribers
- Also provides a global parameter server
- Exactly one of them running at any time
- Messages do NOT go through Master (i.e. peer-to-peer)
- Nodes will not be able to find each other without Master

Sending Messages

- pub = rospy.Publisher()
- msg = \dots
- pub.publish(msg)

ROS Node - Publisher

```
import rospy
from std_msgs.msg import String
def talker():
    rospy.init_node('talker', anonymous=True)
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rate = rospy.get_param('~rate', 1)
    ros_rate = rospy.Rate(rate)
    rospy.loginfo("Starting ROS node talker...")
    while not rospy.is_shutdown():
        msg = "Greetings humans!"
        pub.publish(msg)
        ros_rate.sleep()
if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Monitoring Messages

- You can check if you are sending messages using the *rostopic* command line tool:
- rostopic list lists all the active topics
- rostopic echo <topic> prints messages received on <topic>
- rostopic hz <topic> measures topic publishing rate

Receiving Messages

- rospy.Subscriber("chatter", String, callback)
- def callback(msg): ...

(in C++ need to call spinOnce(), not in Python)

ROS Node - Subscriber

#!/usr/bin/env python

import rospy
from std_msgs.msg import String

```
def callback(msg):
    rospy.loginfo("Received: %s", msg.data)
```

```
def listener():
    rospy.init_node('listener', anonymous=True)
```

rospy.Subscriber("chatter", String, callback)

rospy.loginfo("Listening on the chatter topic...")

```
rospy.spin()
```

ROS Launch Files

- Simple XML files that allow you to
 - Launch multiple nodes at once
 - Set parameters for those nodes
 - Start Master
- roslaunch <package> <file>.launch

ROS Launch File Example

<launch>

```
<!-- Start the talker node -->
```

<node name="talker" pkg="aa274" type="talker.py" output="screen">
 <param name="rate" value="5"/>

</node>

</launch>

A Case Study

• Edge detection in camera images

Node 1 – Camera Driver Subscribes to: Nothing Publishes: Camera images

Node 2 – Edge Detection Subscribes to: Camera images Publishes: Image with edges Node 3 – image_view Subscribes to: Camera images Publishes: Nothing

Node 4 – image_view Subscribes to: Image with edges Publishes: Nothing

A Case Study

- Edge detection in camera image
- rqt_graph



ROS Launch File for Edge Detection

<launch>

```
<arg name="video device" default="/dev/video0" />
```

```
<include file="$(find aa274)/launch/usbcam_driver.launch">
<arg name="video_device" value="$(arg video_device)" />
</include>
```

Developing with ROS

- Catkin workspace: a directory that contains all your ROS development
- It sets the right environment variables
- It knows how to compile your nodes (using *cmake which in turn uses a compiler*)

The commands you need to know:

- mkdir -p ~/catkin_ws/src
- cd ~/catkin_ws
- catkin_make

ROS Packages

- The basic organization structure for your nodes
- Usually corresponds to a "functionality" (e.g. a SLAM package)
- Can contain code for multiple nodes
- Directory structure:



The command you need to know: catkin_create_pkg <name> roscpp rospy std_msgs

Debugging

- rospy.loginfo()
- rqt_console
- rosbag record <topic>
- rosbag play file.bag

- pdb Python Debugger
 - import pdb
 - pdb.set_trace()

En .				Class	Desine	-
	Save Displaying 39 messages			Clear	Resize C	olum
	Message	Severity	Node	Tir	me	
‡12 (The input topic '/narrow_stereo/left/image_raw' is not yet advertised	Warn	/narrow_stereo_textured/	21:39:04.833	(2013-05-06)	1
10	The input topic '/narrow_stereo/right/image_raw' is not yet advertised	Warn	/narrow_stereo/narrow_st	21:39:02.337	(2013-05-06)	1
11	The input topic '/narrow_stereo/right/camera_info' is not yet advertised	Warn	/narrow_stereo/narrow_st	21:39:02.337	(2013-05-06)	1
8	The input topic '/narrow_stereo/left/image_raw' is not yet advertised	Warn	/narrow_stereo/narrow_st	21:39:02.336	(2013-05-06)	1
9	The input topic '/narrow_stereo/left/camera_info' is not yet advertised	Warn	/narrow_stereo/narrow_st	21:39:02.336	(2013-05-06)	1
7 (Holding arms	Info	/arm_holder	21:39:01.402	(2013-05-06)	1
18	The input topic '/wide_stereo/right/camera_info' is not yet advertised	Warn	/wide_stereo/wide_stereo	21:39:01.086	(2013-05-06)	1
16	The input topic '/wide_stereo/left/camera_info' is not yet advertised	Warn	/wide_stereo/wide_stereo	21:39:01.085	(2013-05-06)	1
17	The input topic '/wide_stereo/right/image_raw' is not yet advertised	Warn	/wide_stereo/wide_stereo	21:39:01.085	(2013-05-06)	1
6	The input topic '/wide_stereo/left/image_raw' is not yet advertised	Warn	/wide_stereo/wide_stereo	21:39:01.085	(2013-05-06)	1
5	Moving torso up	Info	/arm_holder	21:38:56.400	(2013-05-06)	/
0	<u>a</u>		/			5
xclud	e Rules:					
	ht Rules:					
ighlig						
ighlig						7

Creating Custom Messages

- Write message definitions (.msg) that are language agnostic
- ROS generates the right files so that roscpp and rospy can use your message
- rosmsg show student

[aa274/Student]:
string name_first
string name_last
uint8 age
uint32 grade

Primitive Type	Serialization	C++	Python
bool (1)	unsigned 8-bit int	uint8_t (2)	bool
int8	signed 8-bit int	int8_t	int
uint8	unsigned 8-bit int	uint8_t	int(3)
int16	signed 16-bit int	int16_t	int
uint16	unsigned 16-bit int	uint16_t	int
int32	signed 32-bit int	int32_t	int
uint32	unsigned 32-bit int	uint32_t	int
int64	signed 64-bit int	int64_t	long
uint64	unsigned 64-bit int	uint64_t	long
float32	32-bit IEEE float	float	float
float64	64-bit IEEE float	double	float
string	ascii string (4)	<pre>std::string</pre>	str
time	secs/nsecs unsigned 32-bit ints	• ros::Time	rospy.Time
duration	secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration

ROS Services

- A different way for nodes to pass messages to each other
- Request/Response scheme (not Pub/Sub!)
- Examples:
 - Turn a light or LED on or off
 - Assign a name to a face and retrain face recognizer
 - Spawn a new model in the Gazebo simulator

The Parameter Server

• Parameters are stored under namespaces; e.g.

- /move_base/local_costmap/height
- /usb_cam/framerate
- /gazebo/time_step
- Setting and getting parameters:
 - rosparam set param_name param_value
 - param_value = rospy.get_param("param_name")
- NOTE: Setting a parameter does not affect a running node!

Dynamic Reconfigure

- Some nodes provide dynamically changeable parameters
 - rosrun rqt_reconfigure rqt_reconfigure

Dynamic Reconfigure	1			DO
Filter key:		/move_base		
Collapse all Expand all	base_global_planner	global_planner/Globa	lPlanner	
	base_local_planner	dwa local planner/DV	VAPlannerRO:	5
amci ▶ camera	planner_frequency	0.0	100.0	0.0
cmd_vel_mux	controller_frequency	0.0	100.0	3.0
depthimage_to_laserscan	planner_patience	0.0	100.0	5.0
r move base	controller_patience	0.0 -0.0	100.0	15.0
DWAPlannerROS	max_planning_retries	-1 0	1000	-1
GlobalPlanner	conservative_reset_dist	0.0	50.0	0.1
▼ local_costmap	recovery_behavior_enabled			
inflation_layer	clearing_rotation_allowed			
obstacle_layer ▶ multisense sl	shutdown_costmaps			
▶ segway	oscillation_timeout	0.0 🥌	60.0	5.0
	oscillation_distance	0.0	10.0	0.1
Refresh	restore_defaults			

URDF

- Universal Robot Description Format
- An XML file that describes the kinematic chain of your robot

```
<link name="base_link">
  <visual>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
   </geometry>
                                                    <joint name="head_swivel" type="continuous">
                                                                                                                        Child frame Child
    <material name="blue">
                                                     <parent link="base_link"/>
      <color rgba="0 0 .8 1"/>
                                                     <child link="head"/>
                                                                                                            Joint axis 🔻
   </material>
                                                     <axis xyz="0 0 1"/>
                                                                                                            in joint frame
  </visual>
                                                      <origin xyz="0 0 0.3"/>
                                                                                                                         OINT
  <collision>
                                                    </joint>
    <geometry>
                                                                                                          Joint origin
      <cylinder length="0.6" radius="0.2"/>
                                                                                                                  Parent
   </geometry>
 </collision>
  <inertial>
   <mass value="10"/>
   <inertia ixx="0.4" ixy="0.0" ixz="0.0" iyy="0.4" iyz="0.0" izz="0.2"/>
 </inertial>
</link>
                                                                                                           Parenthame
```



- Same code that will run in production
- Physics is mostly accurate



Some more libraries you will hear about...

- TF: coordinate frame transform library
- Actionlib: processes with goals and feedback
- dynamic_reconfigure: making nodes configurable on the fly

Getting help

- ROS wiki (http://wiki.ros.org/)
- Github
- Stack Overflow
- The Construct / Robot Ignite Academy
- Google :)

Next time

Motion control

