Lecture 22: An Introduction to ROS

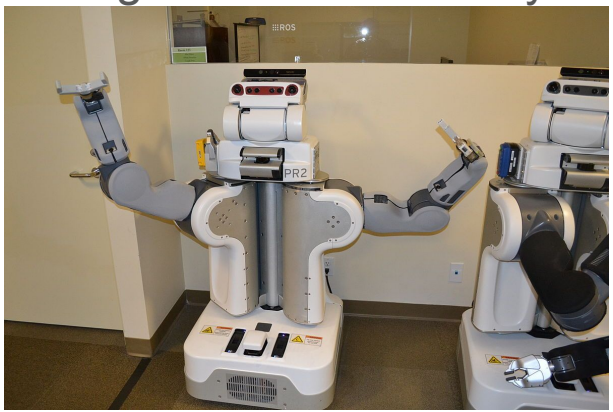
Shane Rozen-Levy

What is ROS?

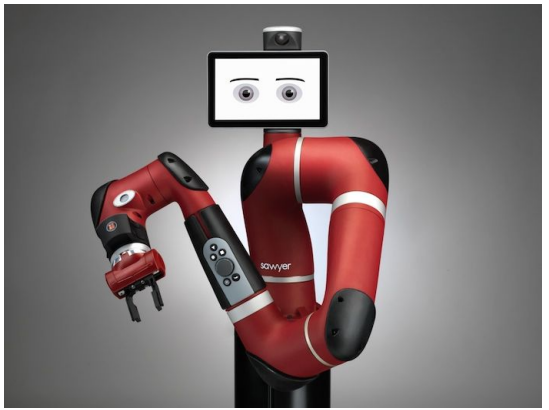
- Stands for Robot Operating System
  - Not an operating system in conventional sense
- Standardized communication protocol between subsystems
  - Each subsystem is a **node**
  - Talk over **topics** with **messages**
- Many open source libraries
- ROS runs on Linux*
- Is not realtime or deterministic
  - Why we don't use it when grading
- Code is written in Python or C++
  - Some minor matlab support

Uses

- Heavily used in academia
- Not used in industry
  - Exception when prototyping
  - Exception when market is academia
- Designed to be a thin layer on top of algorithms



PR2 - Willow Garage



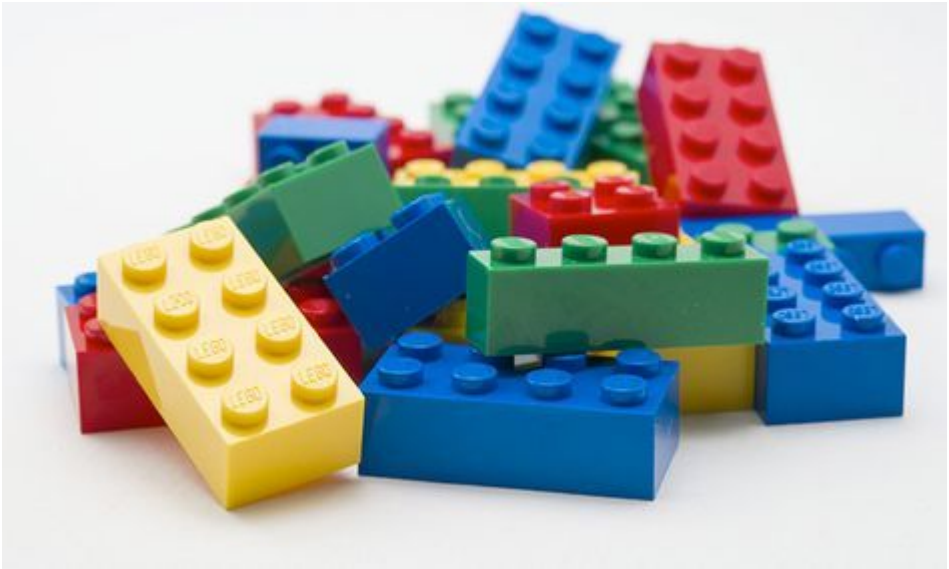
Sawyer - Rethink Robotics

ROS and ROS 2

- ROS is nearing end of official support (March 2025)
  - ROS 2 is still supported
- ROS 2
  - Still in development
  - Realtime and deterministic
  - Better over WiFi
  - Better security
  - About time academics switched from ROS to ROS 2

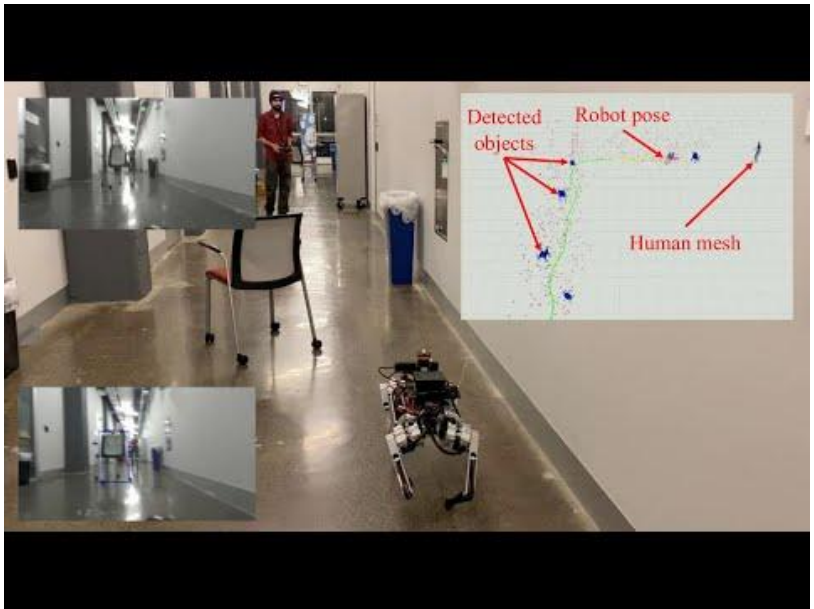
Philosophy

- Modularity
- Reusability
- Open source & sharing



Example Use Case

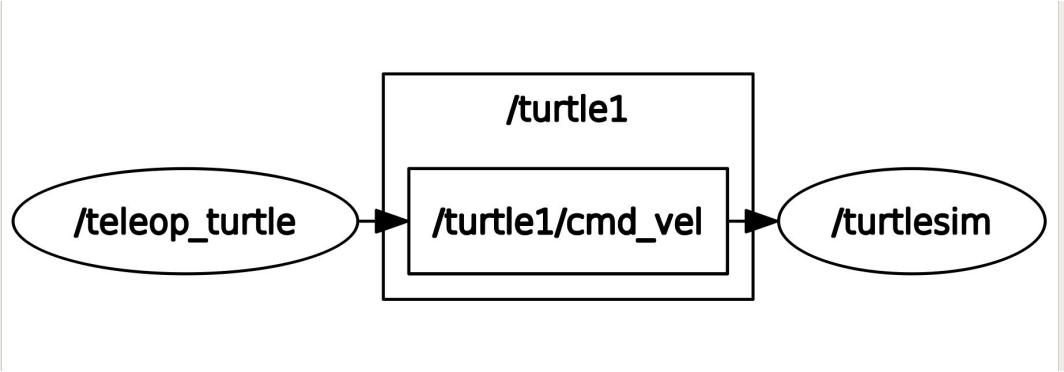
- Vassilis designs new planning algorithm mobile robot
- Quickly change from fixed target to tracking person
- Easily deploy on Turtlebot and Spirit



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} V\cos\phi \\ V\sin\phi \\ \omega \end{bmatrix}$$

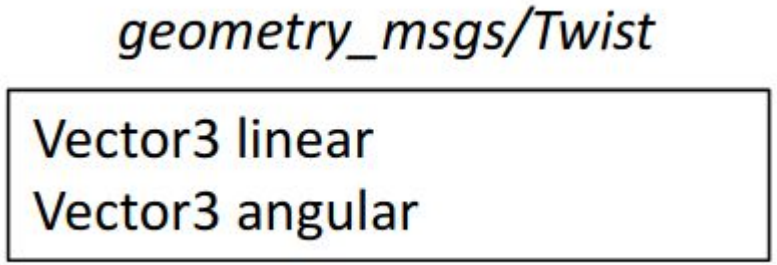
ROS Nodes

- Single purpose subsystem
  - Planning, Joystick, Visualization, Control, SLAM, etc.
- Runs & compiled separately from other nodes
  - Talks to other nodes w/ ROS
- Talks to other nodes using **topics** and **messages**
  - **Publish**, **subscribe**, or both

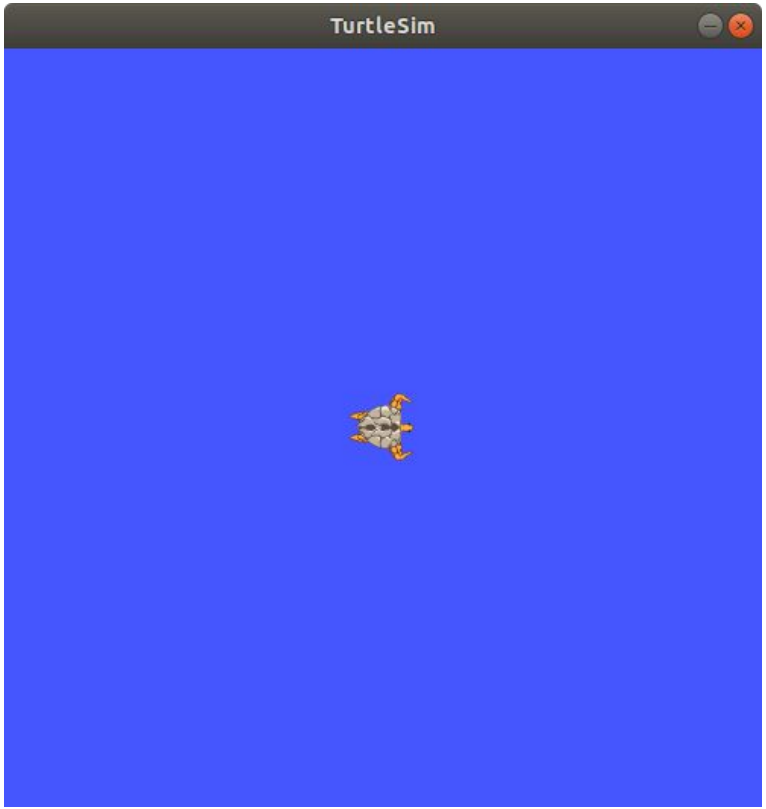


ROS Topics

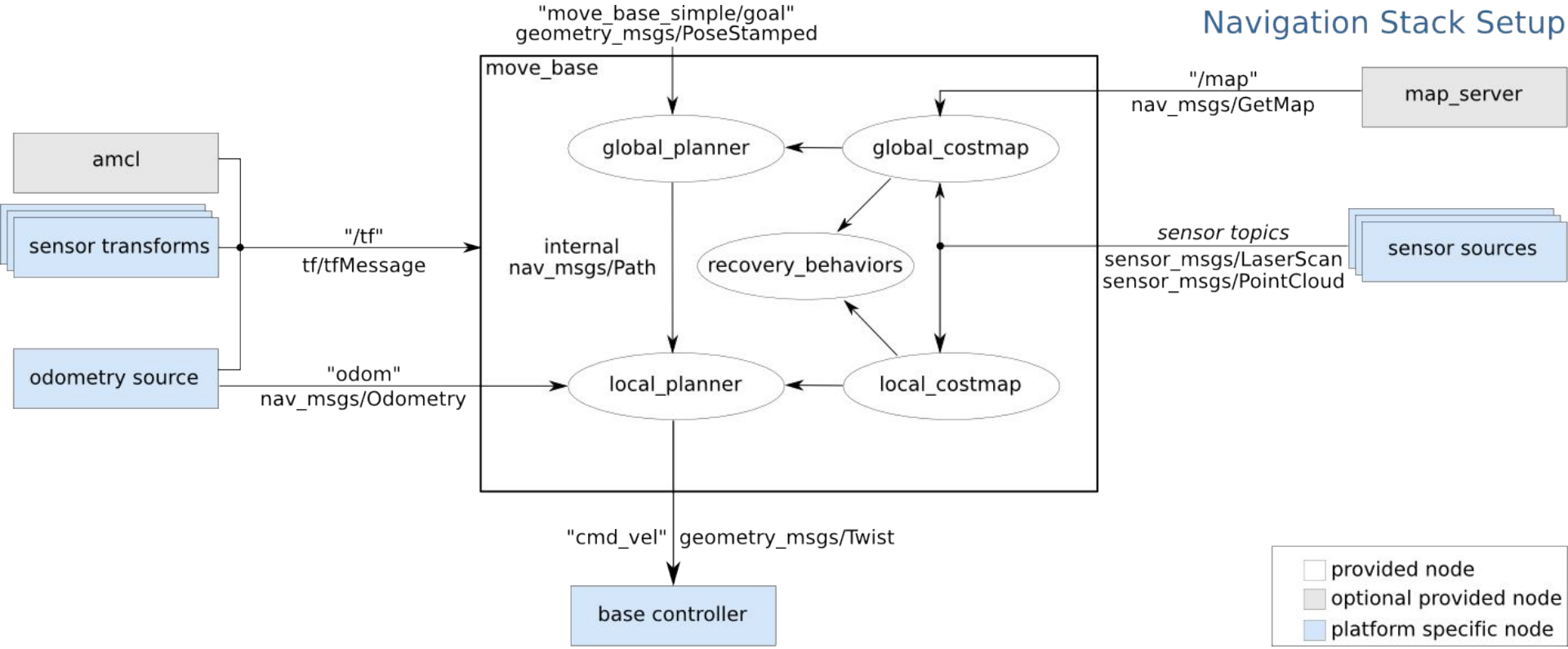
- Named stream of **messages** of set type
  - Robot joint states are jointState message
  - Velocity is Twist
- Nodes talk by **publishing**
  - Yelling to the world
- Nodes listen by **subscribing**
  - Listening to people shouting
- Multiple nodes can subscribe to the same topic



Turtlesim Demo

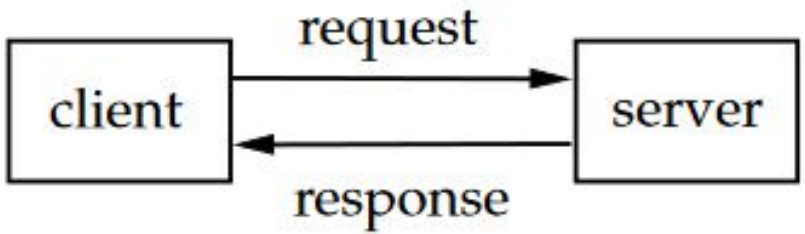


ROS Navigation Stack



Service

- Client server relationship
- Call and response
  - Send message to node and wait for specific response
- 1:1 communication between nodes
- Examples:
  - Spawn a robot
  - Scoring final project



ROS Master

- Remember having to set ROS Master URI
- Handles naming and registration of all the nodes
- Allows nodes to find each other
- Only 1 ROS master per setup
- Houses parameter server

Parameters

- Stored on the parameter server
- Shared dictionary of values
- Used for quasi-static values
- Examples:
  - Controller gains
  - Robot parameters
    - Length, mass, etc.

Launch files

- Many commands needed to start all nodes for robot
- Launches nodes, reads in parameters
- XML in ROS 1, Python in ROS 2
- Create launch files that call other launch files

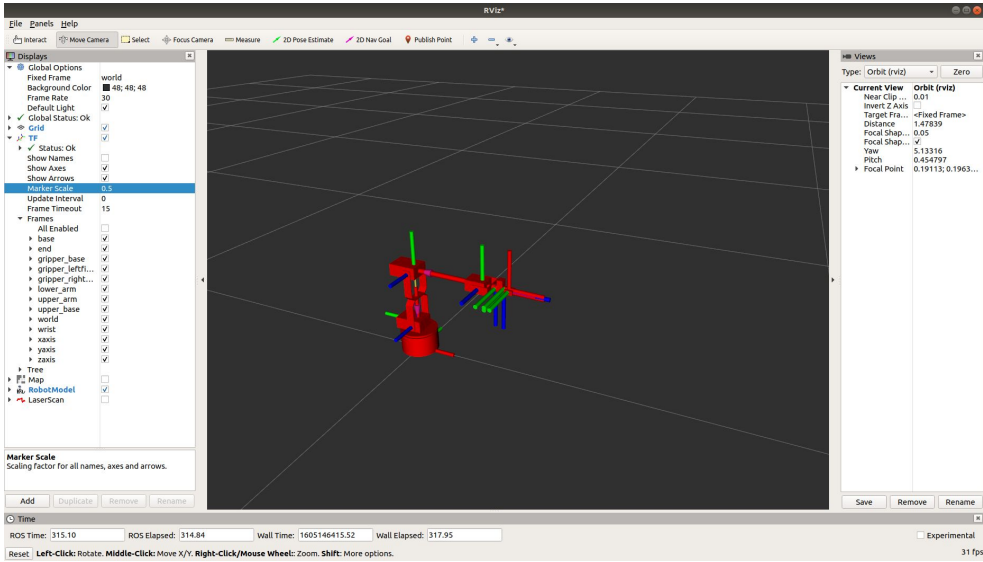
```
<?xml version="1.0"?>
<launch>
  <!-- We resume the logic in empty_world.launch, changing only the name of the world to be launched -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find al5d_gazebo)/worlds/lab0.world"/>
    <!-- more default parameters can be changed here -->
  </include>



  <!-- Convert an xacro and put on parameter server -->
  <!-- <param name="robot_description" command="$(find xacro)/xacro $(find al5d_description)/urdf/al5d_simple.xacro" /> -->
  <param name="robot_description" command="$(find xacro)/xacro $(find al5d_description)/urdf/al5d_cad.xacro" />

  <rosparam command="load" file="$(find al5d_gazebo)/config/al5d_position_controller.yaml" />
  <rosparam command="load" file="$(find al5d_gazebo)/config/joint_state_controller.yaml" />
```

ROS Bags

- Data logging
- Subscribe to various topics and save them
- Can also be used to play them back
- Example:
  - Drive robot around and save sensor data to Bag
  - Experiment playing data back to mapping algorithm
- Demo with turtle sim

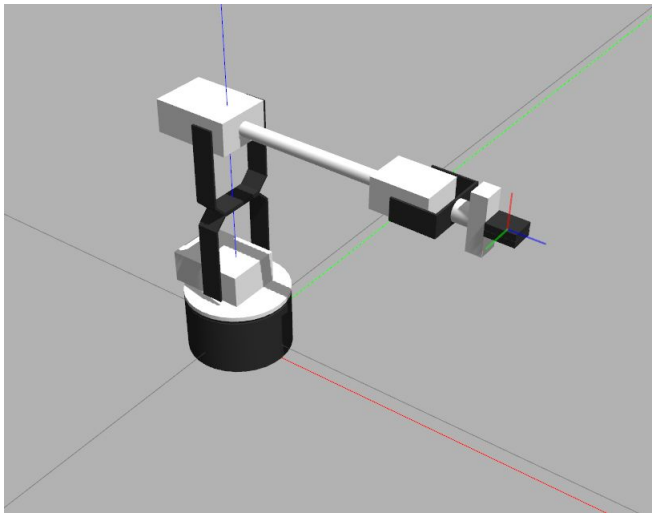
RVIZ

- ROS visualizer
  - Not a simulator
- Subscribes and displays topics
- Useful for:
  - Robot model
  - Laser scans
  - Maps
  - TF
- Brief demo RVIZ with simulator



Overview of Simulator Setup

- Position control of joints from ros_control
  - PID loops
- arm_controller interfaces between student code and ros_control
  - Handles joint limits
  - Interpolates between target states
  - Handles velocity inputs
- Matlab arm_controller publishes to python arm controller (less code)
- Simulator demo



When/how should you use ROS?

- Want people to use your code
- Want easy access to other people code
- Don’t need determinism or realtime
- Wrap around algorithm
  - Don’t put ROS dependencies in the core of algorithm
- Communicating between algorithms
- Need computer
  - Raspberry Pi, NVidia Jetson
  - Not an arduino
- Wifi communication
- Access Vicon data

Turtlebot - Willow Garage

Next steps

- ROS wiki has great tutorials
  - http://wiki.ros.org/ROS/Tutorials
- Gazebo and other simulators mean you don’t need real robot