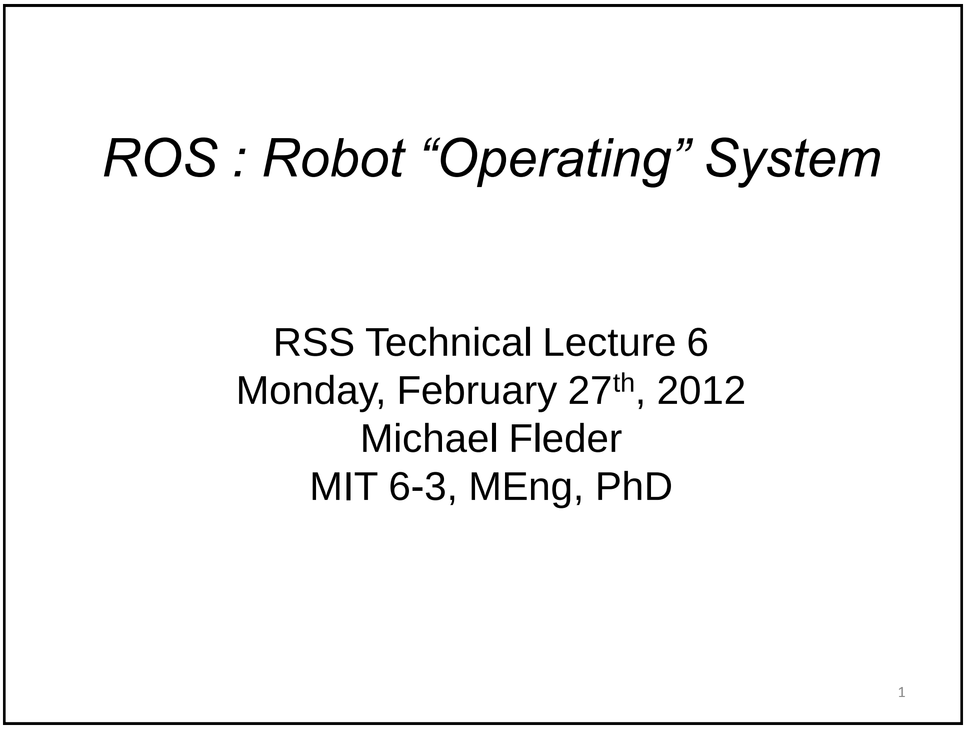
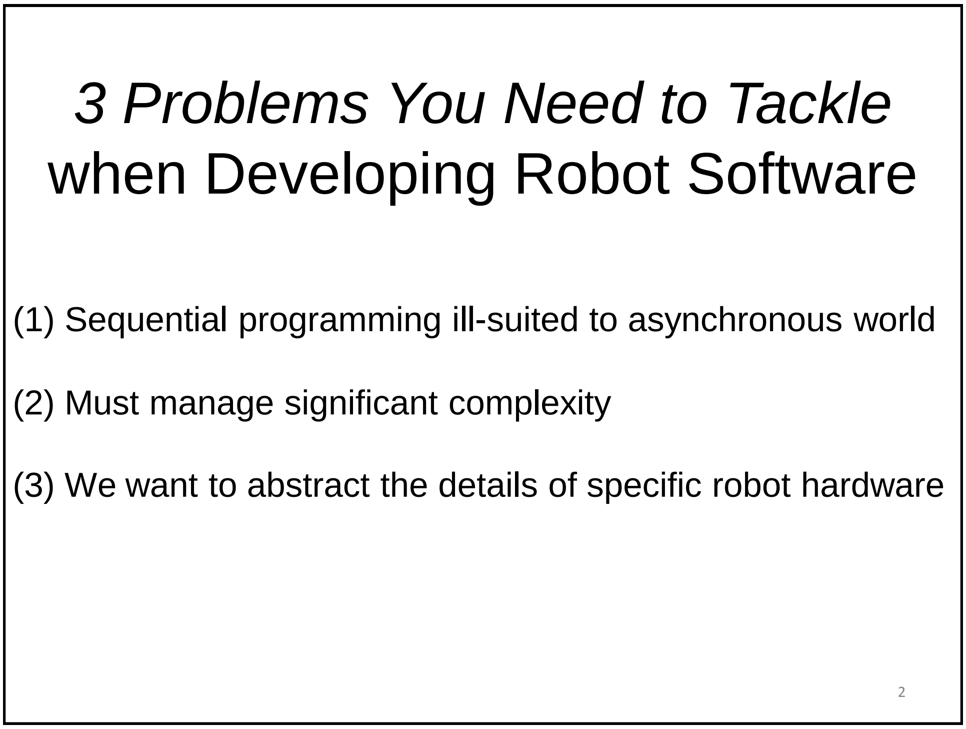


# *ROS : Robot “Operating” System*

RSS Technical Lecture 6
Monday, February 27th, 2012
Michael Fleder
MIT 6-3, MEng, PhD

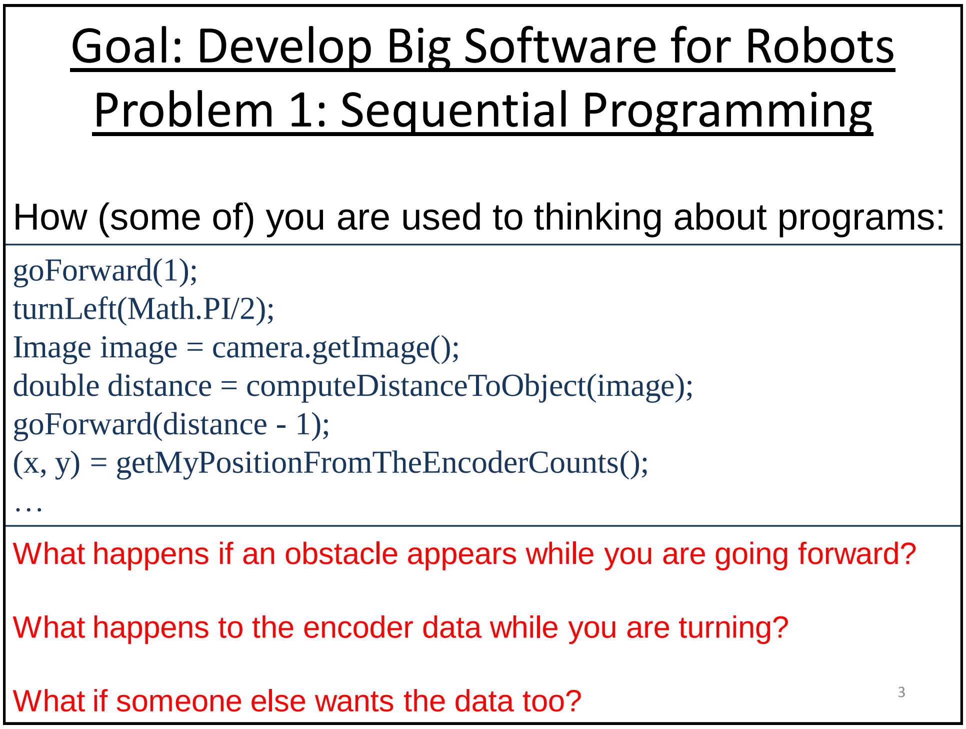


# *3 Problems You Need to Tackle* when Developing Robot Software

(1) Sequential programming ill-suited to asynchronous world

(2) Must manage significant complexity

(3) We want to abstract the details of specific robot hardware



# Goal: Develop Big Software for Robots
# Problem 1: Sequential Programming

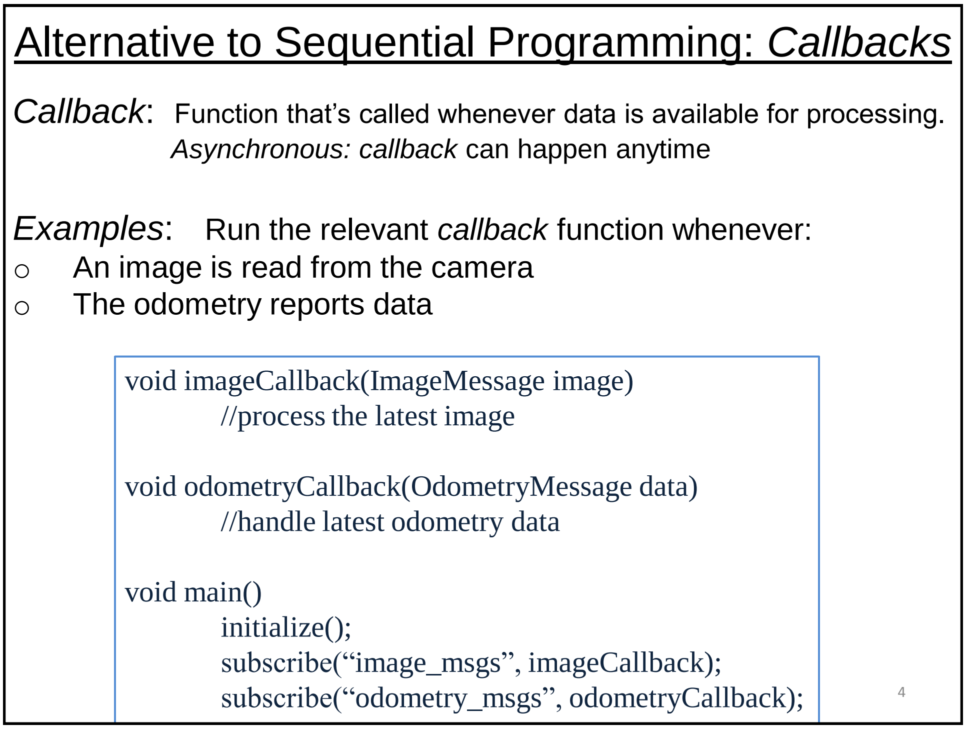
How (some of) you are used to thinking about programs:

```
goForward(1);
turnLeft(Math.PI/2);
Image image = camera.getImage();
double distance = computeDistanceToObject(image);
goForward(distance - 1);
(x, y) = getMyPositionFromTheEncoderCounts();
…
```

What happens if an obstacle appears while you are going forward?

What happens to the encoder data while you are turning?

What if someone else wants the data too?



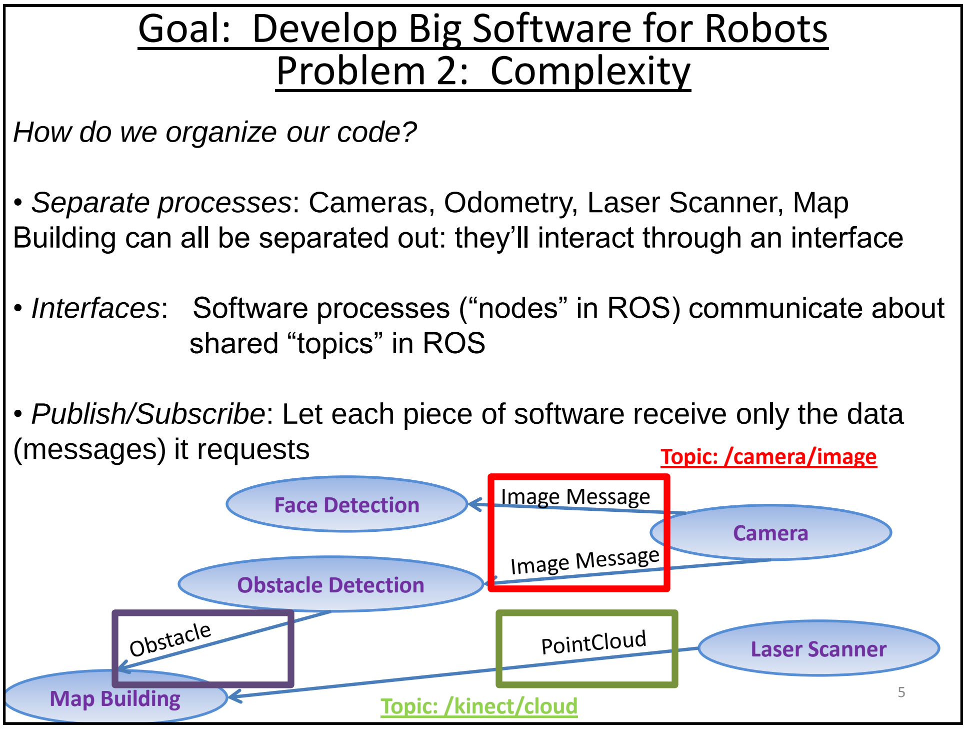
# Alternative to Sequential Programming: *Callbacks*

*Callback*: Function that's called whenever data is available for processing.
*Asynchronous: callback* can happen anytime

*Examples*: Run the relevant *callback* function whenever:

○ An image is read from the camera
○ The odometry reports data

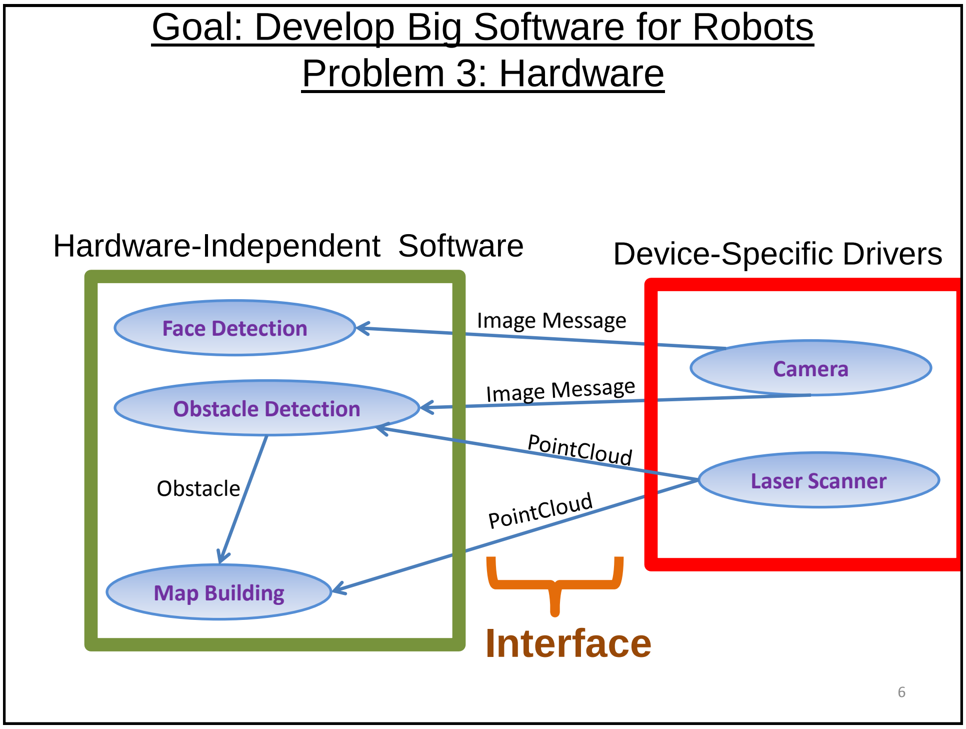
```
void imageCallback(ImageMessage image)
        //process the latest image

void odometryCallback(OdometryMessage data)
        //handle latest odometry data

void main()
        initialize();
        subscribe("image_msgs", imageCallback);
        subscribe("odometry_msgs", odometryCallback);
```



# Goal: Develop Big Software for Robots
## Problem 2: Complexity

*How do we organize our code?*

- *Separate processes*: Cameras, Odometry, Laser Scanner, Map Building can all be separated out: they'll interact through an interface
- *Interfaces*: Software processes ("nodes" in ROS) communicate about shared "topics" in ROS
- *Publish/Subscribe*: Let each piece of software receive only the data (messages) it requests



# Goal: Develop Big Software for Robots
## Problem 3: Hardware

## Goal: Develop Big Software for Robots
## Problem 3: Hardware

PR2

Roomba

Care-O-bot 3

## Summary so Far

(1) ~~Sequential Programming~~

→ Callbacks

(2) ~~Complex, multifunction software~~

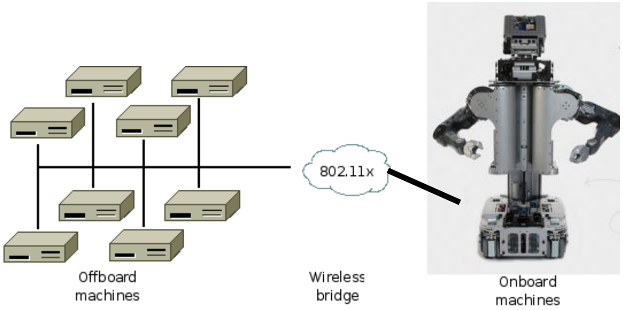
→ Separate processes that communicate through a messaging interface

(3)~~Hardware dependent software~~

→ Messaging interface helps avoid hardware dependencies

→ ROS : Sets up this software structure for you.

# ROS : Robot "Operating" System

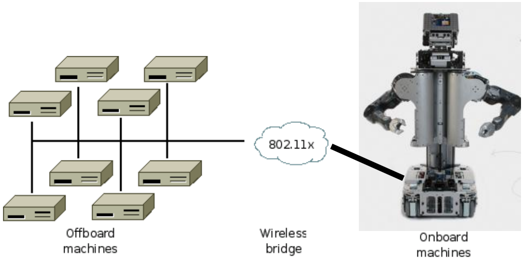


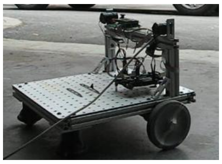
***What is ROS?***

- Message Passing
- Debugging tools
- Visualization tools
- Software Management (compiling, packaging)
- Libraries
- Hardware Abstraction for all of these items

# ROS : Goals for a Meta-Operating System

Hardware Agnostic:





- Peer-to-Peer
- Tools-based
- Multiple Languages
- Lightweight: Only at the edges of your program
- Free + Open Source
- Good for large-scale research

# Outline

☑ **Introduction**
- ☑ 3 Software problems
- ☑ ROS Goals

☐ ***ROS Design***
- ☐ Tools-Based
- ☐ Multiple Languages
- ☐ Lightweight
- ☐ Peer-to-peer
- ☐ Free + Open Source

☐ **Developing Software with ROS**
- ☐ Debugging
- ☐ Visualizing
- ☐ Transforming Coordinate Frames

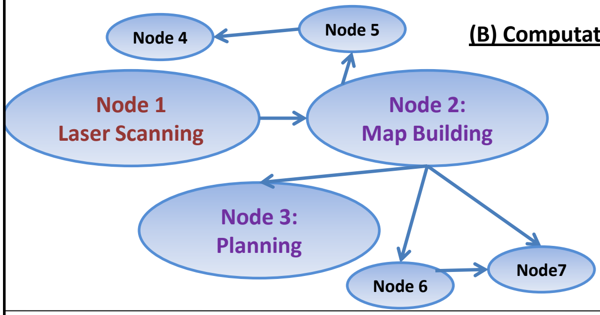
☐ **Packages : ROS and External**
- ☐ Perception
- ☐ Manipulation
- ☐ Navigation

11

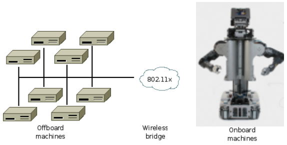
# ROS Design : Conceptual Levels

Carnegie Mellon

**(A) ROS Community:** ROS Distributions, Repositories

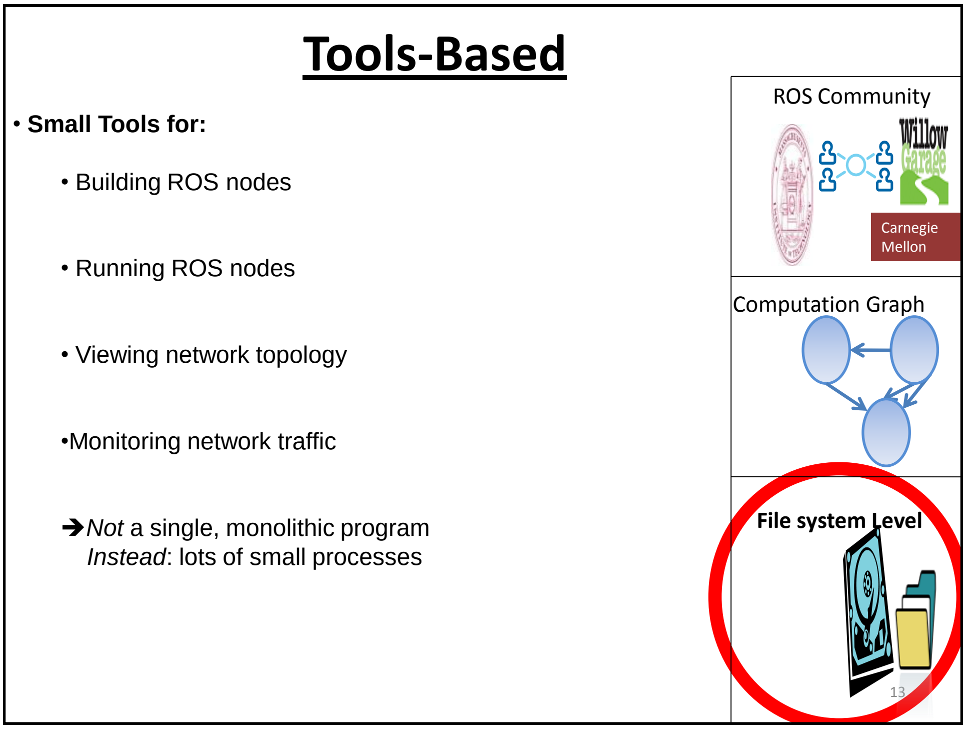


**(B) Computation Graph:** Peer-to-Peer Network of ROS nodes (processes).



**(C) File system Level:** ROS Tools for managing source code, build instructions, and message definitions.

12



# Tools-Based

- **Small Tools for:**
  - Building ROS nodes
  - Running ROS nodes
  - Viewing network topology
  - Monitoring network traffic

➔*Not* a single, monolithic program
*Instead*: lots of small processes

ROS Community

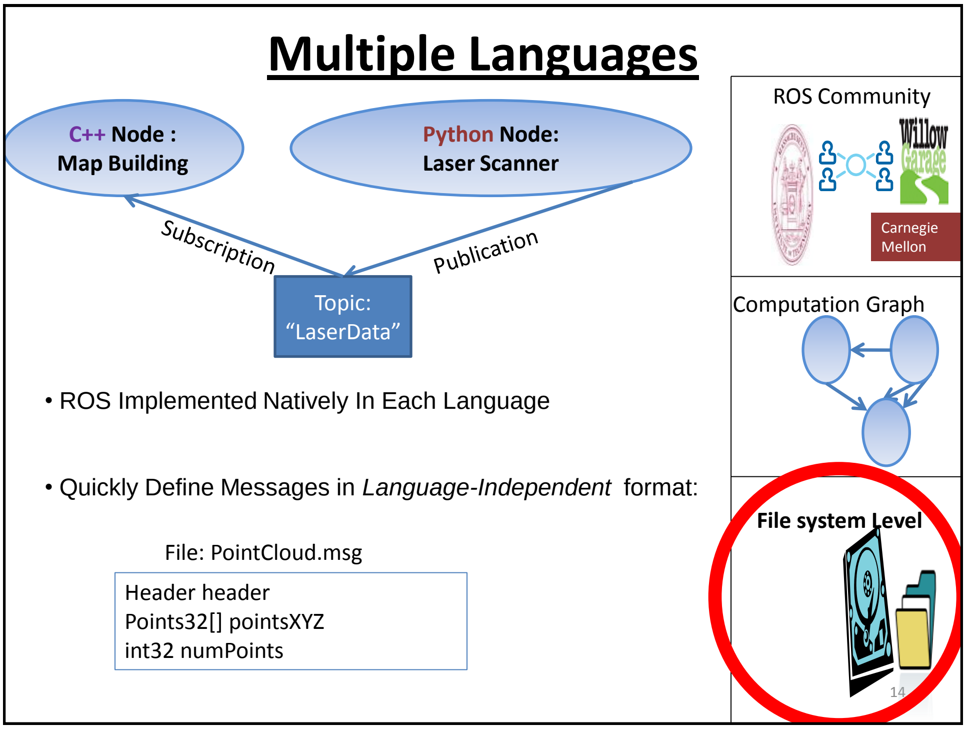
Willow Garage

Carnegie Mellon

Computation Graph

**File system Level**

13



# Multiple Languages

**C++ Node :**
**Map Building**

**Python Node:**
**Laser Scanner**

Subscription

Publication

Topic:
“LaserData”

- ROS Implemented Natively In Each Language
- Quickly Define Messages in *Language-Independent* format:

File: PointCloud.msg

```
Header header
Points32[] pointsXYZ
int32 numPoints
```

ROS Community

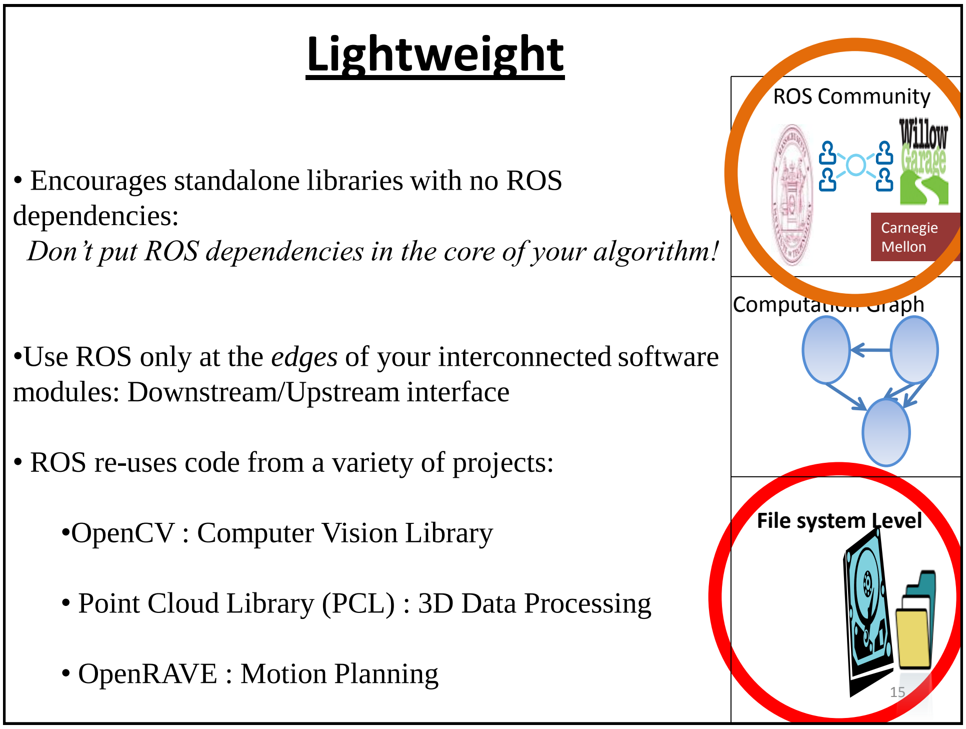
Willow Garage

Carnegie Mellon

Computation Graph

**File system Level**

14



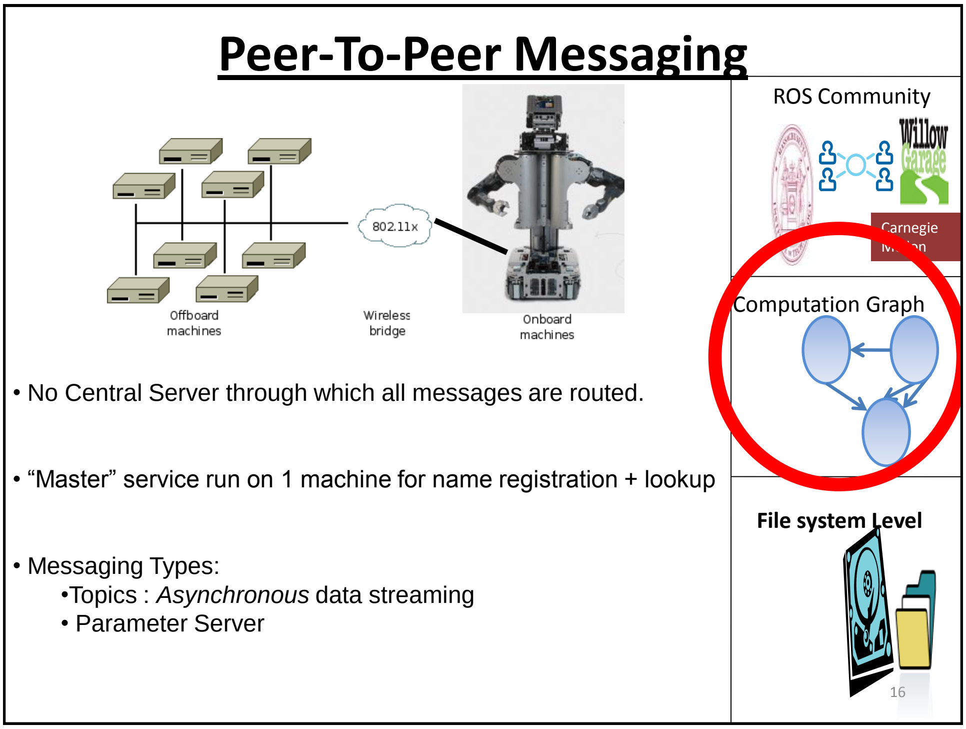
# Lightweight

- Encourages standalone libraries with no ROS dependencies:
  *Don't put ROS dependencies in the core of your algorithm!*

- Use ROS only at the *edges* of your interconnected software modules: Downstream/Upstream interface

- ROS re-uses code from a variety of projects:
  - OpenCV : Computer Vision Library
  - Point Cloud Library (PCL) : 3D Data Processing
  - OpenRAVE : Motion Planning

ROS Community
Willow Garage
Carnegie Mellon
Computation Graph
File system Level
15



# Peer-To-Peer Messaging

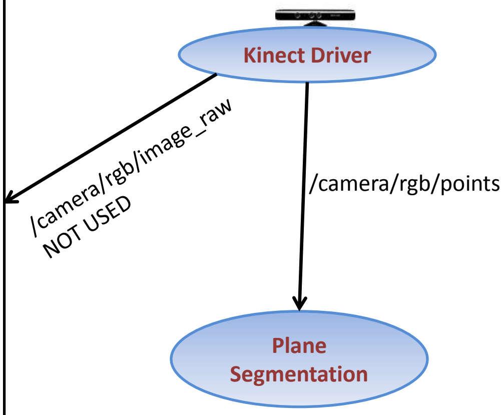
802.11x
Offboard machines
Wireless bridge
Onboard machines
ROS Community
Willow Garage
Carnegie Mellon
Computation Graph
File system Level
16

- No Central Server through which all messages are routed.

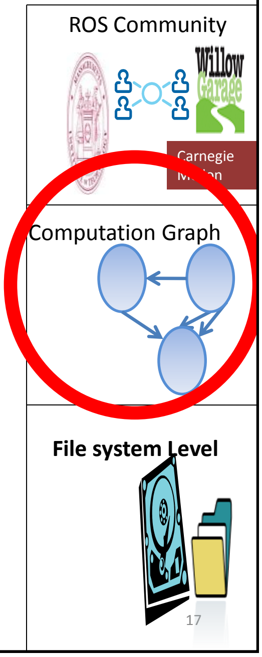
- "Master" service run on 1 machine for name registration + lookup

- Messaging Types:
  - Topics : *Asynchronous* data streaming
  - Parameter Server

# Messaging Demo



DEMO: rostopic, rviz, image_view, roslaunch



# Peer-To-Peer Messaging

*Master:* Lookup information, think DNS

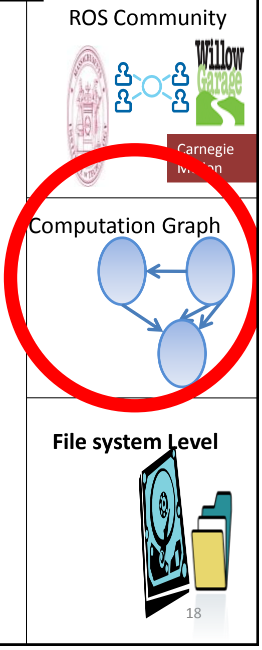
*roscore* command → starts master, parameter server, logging

Publish: Will not block until receipt, messages get queued.

Delivery Guarantees: Specify a queue size for publishers: If publishing too quickly, will buffer a maximum of X messages before throwing away old ones

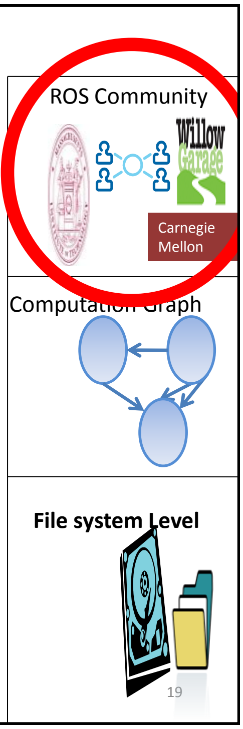
Transport Mechanism: TCPROS, uses TCP/IP

Bandwidth: Consider where your data's going and how



# Free & Open-Source

- BSD License : Can develop commercial applications
- Drivers (Kinect and others)
- Perception, Planning, Control libraries
- MIT ROS Packages : Kinect Demos, etc
- Interfaces to other libraries: OpenCV, etc



# Outline

☑ **Introduction**
- ☑ 3 Software problems
- ☑ ROS Goals

☑ **ROS Design**
- ☑ Tools-Based
- ☑ Multiple Languages
- ☑ Lightweight
- ☑ Peer-to-peer
- ☑ Free + Open Source

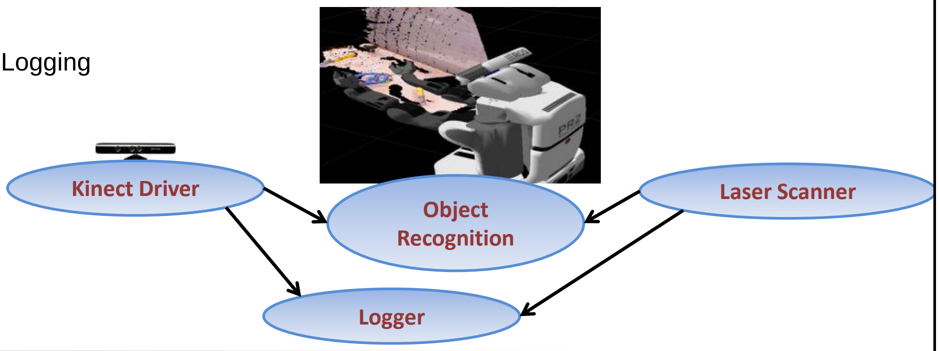
☐ **Developing Software with ROS**
- ☐ Debugging
- ☐ Visualizing
- ☐ Transforming Coordinate Frames

☐ **Packages : ROS and External**
- ☐ Perception
- ☐ Manipulation
- ☐ Navigation

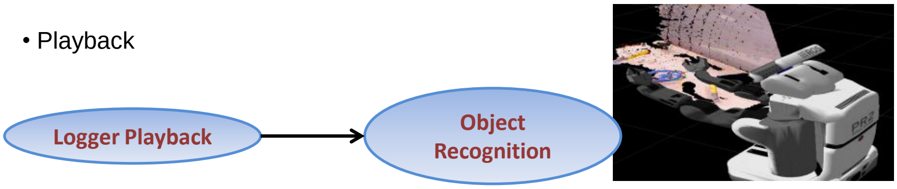
# Development with ROS: Debugging

- Shutdown “Object” node → re-compile → restart : won’t disturb system

- Logging



- Playback



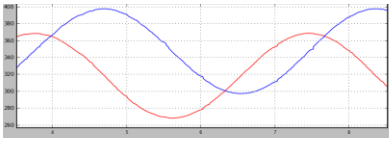
# Useful Debugging Tools

*rostopic:* Display debug information about ROS topics: publishers, subscribers, publishing rate, and message content.

*rostopic echo [topic name] → prints messages to console*
*rostopic list → prints active topics*
*… (several more commands)*

*rxplot :* Plot data from one or more ROS topic fields using matplotlib.

*rxplot* /turtle1/pose/x,/turtle1/pose/y → graph data from 2 topics in 1 plot



***Useful Cheat sheet**:*
http://mirror.umd.edu/roswiki/attachments/Documentation/ROScheatsheet.pdf

# More Useful Development Tools: roslaunch

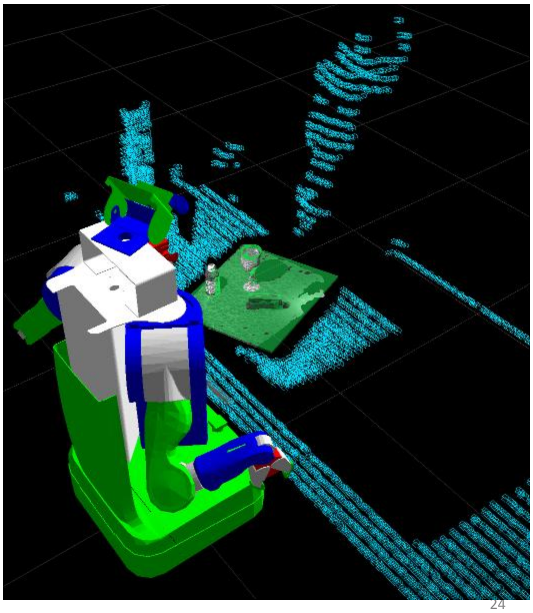
*roslaunch :* Used as a startup script. Starts ROS nodes locally and remotely via SSH, as well as setting parameters on the parameter server

*[Example: Launch file of the demo]*

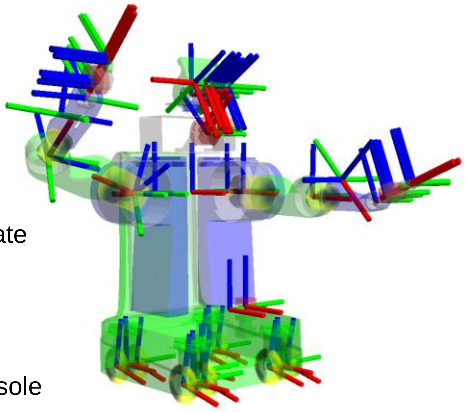
# Development with ROS: Visualization

- Visualize:
  - Sensor Data
  - Robot Joint States
  - Coordinate Frames
  - Maps being built
  - Debugging 3D Markers

**DEMO**



# Development with ROS: Transformations



- “TF” = Name of Transform package
  “Tully Foote” == Person/Developer

- TF Handles transforms between coordinate frames : space + time

- tf_echo : print updated transforms in console

*Example:*

```
rosrun tf tf_echo [reference_frame] [target_frame]
```

*(demo)*

# Outline

☑ **Introduction**
  ☑ 3 Software problems
  ☑ ROS Goals

☑ **ROS Design**
  ☑ Tools-Based
  ☑ Multiple Languages
  ☑ Lightweight
  ☑ Peer-to-peer
  ☑ Free + Open Source

☑ **Developing Software with ROS**
  ☑ Debugging
  ☑ Visualizing
  ☑ Transforming Coordinate Frames

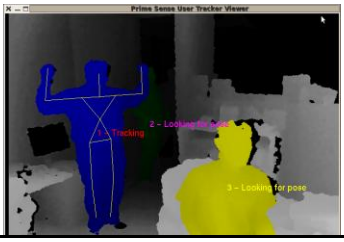
☐ **Packages : ROS and External** ←
  ☐ Perception
  ☐ Manipulation
  ☐ Navigation

# Packages: Perception

- Point Cloud Library (PCL)

- OpenCV

- Kinect / OpenNI :

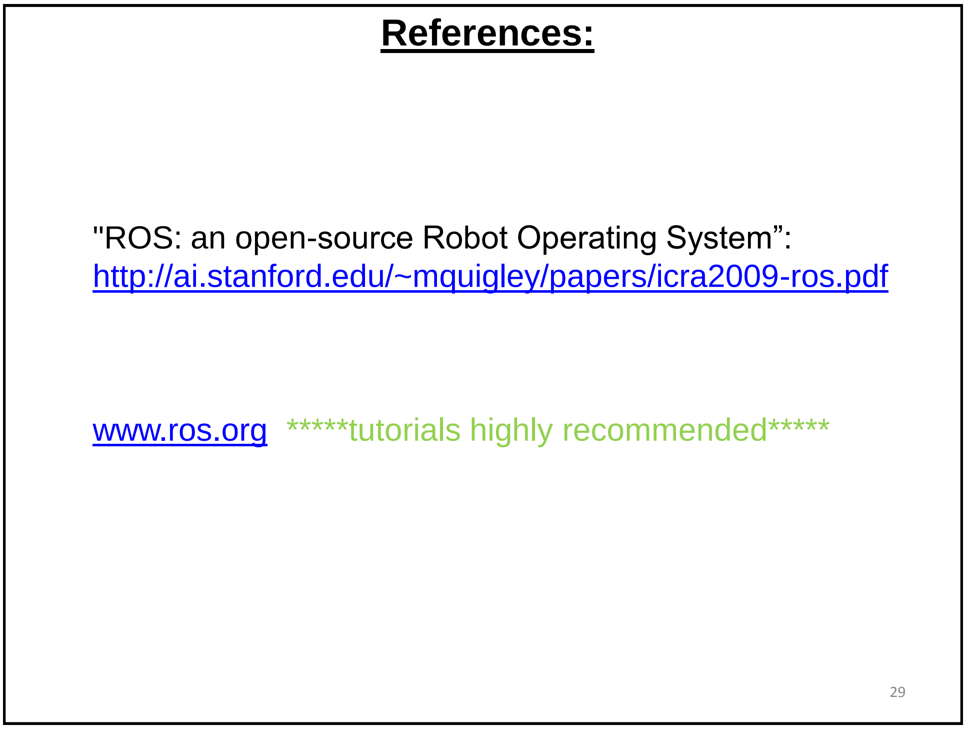


27

# Conclusion: You can now begin to develop complex software for robots

- *Reasons to use ROS:* Asynchronous callbacks
  Complexity management
  Hardware agnostic
- *ROS's Design:* Peer-to-Peer, Multiple Languages, Lightweight
- *Developing Software with ROS:* Debugging, Visualizing
- *Packages*

28



## References:

"ROS: an open-source Robot Operating System”: http://ai.stanford.edu/~mquigley/papers/icra2009-ros.pdf

www.ros.org *****tutorials highly recommended*****