Introduction to ROS
(Robotic Operating System)

Lecture 1

by Ali Shafiekhani (Modified by G. N. DeSouza)
Vision Guided and Intelligent Robotics Lab (ViGIR)
Dep. Of Electrical and Computer Engineering, University of Missouri–Columbia
History

2000-2007: ROS was originally developed by Stanford Artificial Intelligence Laboratory in support of the Stanford AI Robot (STAIR) project.

2008-2013: researcher at more than 20 institutions collaborated with Willow Garage (was started in 2006 by Scott Hassan) engineers in a federated development model.

Feb. 2013: Maintenance moved to the Open Source Robotics Foundation.

Steve Cousins (2011)
Why ROS?

- Open-source under BSD licenses.
- A language-independent architecture (C++, python, lisp, java)
- Allows researcher not to “reinvent the wheel” by providing standard tools and interfaces.

Algorithm, frameworks, hardware drivers are developed and maintained by the international ROS community

- Over 1500 participants on the ros-users mailing list.
- More than 3300 users on the collaborative documentation wiki.

- The wiki has more than 22000 wiki pages and over 30 wiki pages edits per day.

General tools are maintained by Willow Garage, inc and some external developers
ROS Code hierarchy

**Versions:** ROS Box Turtle ('10), ROS C Turtle ('10), ROS Diamonback ('11), ROS Electric Emys ('11), ROS Fuerte ('12), ROS Groovy Galapagos ('12), ROS Hydro Medusa ('13), ROS Indigo Igloo ('14), ROS Jade Turtle ('15)

Repositories;
Stack;
Workspaces & Packages;
Nodes.
**Repository**: Contains all the code from all or a particular development group.

**Stack**: Group of packages for a particular function (e.g. Navigation).

**Packages**: Separate modules that provide different functions/services for particular devices or robots (e.g. Jaco, Husky).

**Nodes**: Executable Tasks/Threads that exist in each model (can be in C++, python, lisp, java, ...).
**Workspace:** private or system (Catkin).

**Packages:** Dependencies
direct/indirect
Manifest
Requirements to be in Catkin.
Nodes

- A node is a process/thread that performs a computation.
- Multi-processing is achieved by the management of multiple nodes. Usually each node takes care of one specific task: e.g. communication with other nodes; control of different devices/components of the robot.
- The communication between nodes can be done by: streaming topics, Remote Procedure Call (RPC) services, and parameter service.
Topics

- Topics are named buses over which nodes exchange messages (similar to a pipe, or a socket, mailbox, etc).
- Each node can send a message by publishing data under a specific topic name and another node can subscribe to the same topic and receive data. The sender and receiver node are not aware of each other’s existence. There can be multiple, simultaneous publishers and subscribers to any single topic.
- Each topic has to have a specific ROS message type, which nodes must “agree” upon.
- There are many useful message types already provided by ROS packages, but anyone can build their own message type.
Messages

- A message is a simple data structure, to which nodes can assign data and publish to a topic.
- msg files are simple text files for specifying the data structure of a message to be published. They are stored in the msg subdirectory of the package.

```
fieldtype1 fieldname1
fieldtype2 fieldname2
fieldtype3 fieldname3
int32 x
int32 y
```
How ROS Works

camera -> advertise ("images") -> ros "master" -> subscribe ("images")

camera -> Images (tcp) -> viewer -> Images (tcp) -> viewer 2

camera -> publish (img) -> viewer

Introduction to ROS
Vinobot & ROS

Hardware:

- JACO Driver (USB)
- IEEE 1394-B
- Husky Driver (RS232)
- Serial (RS232)
- rosserial (USB)
- RFID

Introduction to ROS
Vinobot & ROS

Software:

- JACO node
- JACO Driver
- ROS master
- Husky node
- Husky Driver
- rosserial (python)
- RFID node
- Streaming topic
- RPC Service
- Streaming topic
- sensors node
- camera node
- ?

Introduction to ROS