ROS - Robot Operating System

Radu Bogdan Rusu
Willow Garage, Inc

http://www.willowgarage.com

November 1, 2010
ROS Tutorial

Topics

- nodes connect via topics
- the discovery of who publishes on what topic is done via a ROS master
ROS Tutorial

Services

- synchronous services

```
ROS Master

0) advertiseService("bar",foo:1234)
1) lookupService("bar")
2) {foo:3456}
4) request data (args)
ROSRPC: foo: 3456

XML/RPC

Talker
XML/RPC: foo: 1234
ROSRPC: foo: 3456

Listener

TCP
5) reply data
```
ROS Tutorial

Actions (1/2)

Using function calls and callbacks

- request goals (client side)
- execute goals (server side)
action protocol relies on ROS topics to transport messages
ROS Tutorial

Messages

- defined in `package-name/msg/*.msg` files, sent over topics
- basic data types:
  - `int{8,16,32,64}`
  - `float{32,64}`
  - `string`
  - `time`
  - `duration`
  - `array[]`

- Example: `Point.msg`
  - `float64 x`
  - `float64 y`
  - `float64 z`

- used in ROS services, defined in `package-name/srv/*.srv`
  Service = Request msg + Response msg
Parameters (1/2)

- nodes can set **parameters** on the server
- any other nodes can read them
ROS Tutorial

Parameters (2/2)

- have unique names
- can represent primitive data types:
  - integers
  - floats
  - boolean
  - dictionaries
  - maps, etc
- can be set and remapped at runtime
- stored on the parameter server
Packages and Stacks

- **Packages**: directories with a certain structure, can contain anything: nodes, messages, tools
- in their most basic form:
  - `package_name`
  - `package_name/Makefile`
  - `package_name/CMakeLists.txt`
  - `package_name/manifest.xml`
- **ROS core** = small, but **ROS universe** = many packages !!!
- **Stacks**: collection of packages
- in their most basic form:
  - `stack_name/`
  - `stack_name/package_name_1`
  - `stack_name/package_name_N`
  - `stack_name/stack.xml`
ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.

ROS:

**ROS Installation Instructions**
Provides instructions on how to install ROS on your machine.

**Getting Started with ROS**
Provides tutorials and technical overview of ROS. Also check out the ROS cheatsheet.pdf

**Getting Help**
Provides a short list of links for how to ask questions and get help with ROS.

**Getting Involved**
How to contribute to the ROS community.

Software Built on ROS:

**Core Libraries by Functionality**
Lists available APIs within ROS by language and topic.

**Code Repositories**
There's a broad community contributing open-source software for ROS. This lists many of the known code repositories that build on top of ROS.

**Search Software**
Search for libraries across the entire ROS community.

Robots/Hardware Using ROS:

Radu Bogdan Rusu
Example

- simulator_stage demo

4. Run stageros with an existing world file

The *stage* package ships with some example world files, including one that puts an Erratic-like robot in a Willow Garage-like environment. To run it:

```
roscd stage
./bin/stageros world/willow-erratic.world
```

You should see a Stage window pop up that looks something like this: