ROS 1.0, “Box Turtle” Release

Ken Conley (kwc@willowgarage.com)

Helpful Hints

Learn the UNIX command-line, ROS is heavily based on it

Learn YAML, which is a format that is heavily used within ROS

ROS' 3 Layers

computation graph: how programs run

file system: how program files are organized and built

repositories: how files are distributed online

ROS for Sharing

ROS is designed for sharing code, which affects its design at each of these levels.

ROS

Graph	Name service Parameter Server Nodes
Filesystem	Packages Message formats Build system
Community	*-ros-pkg code repositories Wiki Distributions

Community Level

Repositories

Federated model

Easy to contribute and control your code

Open Source, mostly BSD

Mostly SVN access, some GIT

Wiki/ros.org

Indexes all known ROS repositories

Wiki: http://ros.org/wiki

ros.org/browse: Search for ROS software

roslocate

Command-line tool to search for and download source code for ROS packages

```
$ roslocate describe package-name
$ roslocate svn package-name
$ roslocate repo package-name
$ roslocate search keyword
```

```
$ svn co `roslocate svn package-name`
```

roslocate

$ roslocate search irobot
irobot_create_2_1

$ roslocate svn irobot_create
http://brown-ros-pkg.googlecode.com/svn/trunk/rlab/irobot_create

$ svn co `roslocate svn irobot_create`

Filesystem Level

Packages and Stacks

A package is a directory with a manifest.xml file.

A stack is a directory with a stack.xml file.

A package inside of a stack's directory is part of that stack.

Packages

Can contain anything: libraries, Nodes, Messages, tools

Goldilocks principal: enough functionality to be useful, but not so much as to be heavyweight

Many packages = smaller, easier-to-use code, but...

... many packages = MANY (1000+)

Package tools

$ **rospack list**

$ rospack **find** turtlesim

$ rospack **depends-on1** turtlesim

$ **roscd** turtlesim

$ cat manifest.xml

$ **rosmake** turtlesim

Stacks

Release with version numbers

- Stored in CMakeLists.txt

Collect similar packages that work together

- ROS, navigation, vision_opencv

Stack tools

$ rosstack list

$ rostack find navigation

$ roscd navigation

$ ls

$ cat stack.xml

$ cat CMakeLists.txt

roscreate-pkg

roscreate-pkg package-name dependency1...N

```
$ mkdir ~/workspace
$ export ROS_PACKAGE_PATH=~/workspace:$ROS_PACKAGE_PATH
$ cd ~/workspace
$ roscreate-pkg workshop rospy turtlesim
```

rosdep

$ rosdep install package-name

Install system dependencies

rosdep.yaml: OS+version -> OS package

```
wxwidgets:
    ubuntu: libwxgtk2.8-dev
```

msg/Message Description

Simple text files (IDL) compiled to C++, Python, LISP...

package-name/msg/*.msg

Types

- int8, int16, int32, int64 (plus uint*)
- float32, float64
- string
- time, duration
- variable-length array[]
- fixed-length array[C]
- Message

srv/Service Description

package-name/srv/*.msg

Request msg + Response msg

Uses '---' separator between the two, otherwise identical to msg files.

rosmsg/rossrv demo

$ **rosmsg show** Transform

$ **rossrv package** nav_msgs

$ rossrv **show** nav_msgs/GetPlan

Create your own msg

$ roscd workshop
$ mkdir msg
$ echo "int64 num" > msg/
Num.msg

$ rosmsg show workshop/Num
int64 num

$ vi CMakeLists.txt
[uncomment rosbuild_genmsg() line]

$ rosmake workshop

Graph Level

Names

- Syntax:
  - /global-name
  - relative-name
  - ~local-name
- "Pushing down"
  - relative-name -> foo/relative-name
- Wiki page

Master, roscore

Master

- Name service for ROS (Topics + Services)

roscore

- Master
- rosout: logs debugging messages

Nodes

Executable file within a ROS Package

Publish or Subscribe to Topics

Can also provide Services

Have a unique name, e.g. /hokuyo

Name can be remapped at runtime

$ rosrun hokuyo_node hokuyo_node **__name:=base_hokuyo**

Node tools

rosrun

rosnode

rxgraph: interactive graph

roslaunch: launch many nodes

```
<roslaunch>
  <node pkg="foo" type="bar" name="talker" />
  <node pkg="foo" type="baz" name="listener" />
</roslaunch>
```

rosrun

```
$ roscore
```

```
$ rosrun turtlesim turtlesim_node
```

rosnode

$ rosnode **list**

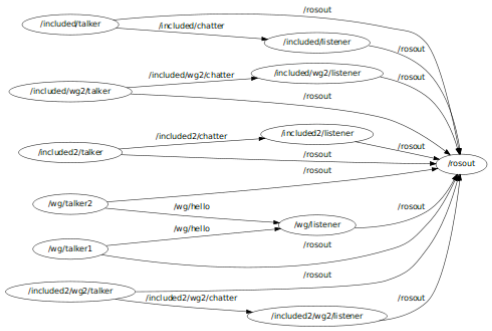
$ rosnode **ping** turtle1

$ rosnode **info** turtle1

rxgraph

$ rxgraph

$ rxgraph -t



Topics

Publisher sends Messages to Subscribers

- Usually TCP/IP transport (UDP in roscpp)
- We don't send messages over XML-RPC

Uniquely named

- Can be remapped at runtime
  `$ rosrun hokuyo_node hokuyo_node laser:=base_laser`

rostopic

Get information about a ROS topic, including printing messages currently being published

$ rostopic **list**
$ rostopic **echo** topic-name
$ rostopic **hz** topic-name
$ rostopic **type** topic-name

rostopic

$ rostopic **list**
$ rostopic **echo** turtle1/pose
$ rostopic **hz** turtle1/pose
$ rostopic **type** turtle1/pose

$ rostopic **pub** turtle1/command_velocity turtlesim/Velocity 10.0 10.0

(If time): mimic remapping demo

Services

RPC

Can be 'persistent', i.e. keep alive connection

Can send custom headers

rosservice

$ rosservice **list**

$ rosservice **type** turtle1/set_pen

$ rosservice type turtle1/set_pen | **rossrv show**

$ rosservice **call** set_pen 255 0 0 4 0

Parameters

Stored on central Parameter Server (the Master)

Unique Names

- Can be remapped at runtime
- Local parameters ~param_name
  - ~param_name = /node_name/param_name
  - Set at runtime
    - `$ rosrun mypkg mynode _param_name:=1`

Types: integers, floats, boolean, dictionaries, maps

Namespaces = Dictionary of dictionaries

rosparam

Get and set parameters from command-line arguments or YAML files

$ rosparam **get** parameter-name
$ rosparam **set** parameter-name
$ rosparam **load** yaml-file [namespace]
$ rosparam **dump** yaml-file

Can use within roslaunch

rosparam

```
$ rosparam set background_g 0
$ rosparam get background_g
0
$ rosservice call clear
$ rosparam set gains "
p: 1.0
i: 2.0
d: 3.0"
$ rosparam get gains/p
1.0
$ rosparam get gains
{d: 3.0, i: 2.0, p: 1.0}
$ rosparam dump params.yaml
$ rosparam load params.yaml copy
$ rosparam get /
```

roslaunch

```
<roslaunch>
    <node pkg="foo" type="bar" name="talker" />
    <node pkg="foo" type="baz" name="listener" />
</roslaunch>
```

Nodes:

- /talker
- /listener

Topics:

- /topic_name

“Pushing down”

```
<roslaunch>
  <group ns="wg">
    <node pkg="foo" type="bar" name="talker" />
    <node pkg="foo" type="baz" name="listener" />
  </group>
  <group ns="stanford">
    <node pkg="foo" type="bar" name="talker" />
    <node pkg="foo" type="baz" name="listener" />
  </group>
</roslaunch>
```

Nodes:

- /wg/talker
- /wg/listener
- /stanford/talker
- /stanford/listener

Topics:

- /wg/topic_name
- /stanford/topic_name

More

rxplot: plot fields from topics

rosbag: logging and playback of topics

rxbag: visualize bag files

roswtf: general problem diagnosis tool

rosbash: roscp, rosls, rosed, rospd

rostest: roslaunch for running tests

More (ros-pkg)

rviz: Robot Visualizer

tf: coordinate system

navigation: 2D+ navigation stack

opencv

actionlib: actions for ROS

More on the way (point cloud library, arm_navigation)

ROS core

Morgan Quigley (Stanford)
Brian Gerkey, Josh Faust, Ken Conley,
Tully Foote, Jeremy Leibs, Tim Field,
Bhaskara Marthi (Willow Garage)