Introduction to ROS Publisher/Subscriber Nodes

David Swords

University College Dublin

david.swords@ucdconnect.ie

January 31, 2014

roslaunch

- roslaunch is a tool for easily launching multiple ROS nodes
- Can automatically respawn processes that have already died

Usage

```
$ roslaunch [package] [filename.launch]
```

Try

```
$ source ~/catkin_ws/devel/setup.bash
$ roscd beginner_tutorials
```

Try

```
$ mkdir launch
$ cd launch
```

- Download turtlemimic.launch file from the Moodle

turtlemimic.launch Overview

- Here we start two groups with a namespace tag of turtlesim1 and turtlesim2 with a turtlesim node with a name of sim
- Allows us to start two simulators without having name conflicts

Example

```
<group ns="turtlesim1">
    <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
  </group>

  <group ns="turtlesim2">
    <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
  </group>
```

turtlemimic.launch Overview

- Here we start the mimic node with the topics input and output renamed to turtlesim1 and turtlesim2
- Renaming will cause turtlesim2 to mimic turtlesim1

Example

```
<node pkg="turtlesim" name="mimic" type="mimic">
    <remap from="input" to="turtlesim1/turtle1"/>
    <remap from="output" to="turtlesim2/turtle1"/>
  </node>
```

Running turtlemimic.launch

Try

```
$ roslaunch beginner_tutorials turtlemimic.launch
```

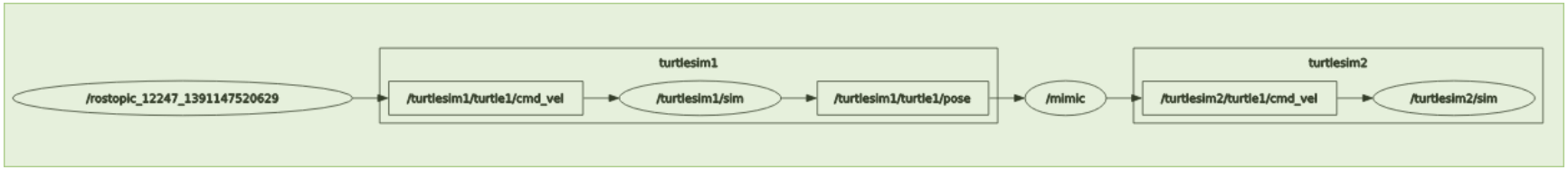
Try

```
$ rostopic pub /turtlesim1/turtle1/cmd_vel
 geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, -1.8]'
```

turtlemimic.launch in rqt_graph

Try

```
$ rqt_graph
```



Creating a Publisher Node

- An executable node that is connected to the ROS network and broadcasts a message

Try

```
$ roscd beginner_tutorials/src
```

- - Download the sample talker code from `ros.org`

Try

```
$ wget https://raw.github.com/ros/ros_tutorials/groovy-
 devel/roscpp_tutorials/talker/talker.cpp
```

talker.cpp Breakdown

Example

```
#include "ros/ros.h"
```

- includes all the headers necessary to use the most common parts of the ROS system

Example

```
#include "std_msgs/String.h"
```

- An automatically generated header file from the String.msg file in the std_msgs package

Example

```
ros::init(argc, argv, "talker");
```

- Initiallization of ROS and how we name our node

talker.cpp Breakdown

Example

```
ros::NodeHandle n;
```

- Takes care of the initialization and destruction of the publisher node

Example

```
ros::Publisher chatter_pub =
 n.advertise<std_msgs::String>("chatter", 1000);
```

- Tell the master that we are going to be publishing a message of type `std_msgs/String` on the topic chatter
- The second argument is the size of our publishing queue
- Maximum of 1000 messages before beginning to throw away old ones
- NodeHandle::advertise() returns a ros::Publisher object, contains publish() method

talker.cpp Breakdown

Example

```
ros::Rate loop_rate(10);
```

- A ros::Rate object allows you to specify a frequency that you would like to loop at

Example

```
while (ros::ok())
 {
```

- a SIGINT is received (Ctrl-C)

talker.cpp Breakdown

Example

```
std_msgs::String msg;

    std::stringstream ss;
    ss << "hello world " << count;
    msg.data = ss.str();
```

- standard String message, which has one member: "data"

Example

```
chatter_pub.publish(msg);
```

- Actually start broadcasting the message

talker.cpp Breakdown

Example

```
ROS_INFO("%s", msg.data.c_str());
```

- ROS_INFO and friends are our replacement for printf/cout

Example

```
ros::spinOnce();
```

- For subscription callback, not necessarily needed here

Example

```
loop_rate.sleep();
```

- Maintains the 10hz publishing rate

Creating a Publisher Node

- An executable node that is connected to the ROS network and listens for a message
- Download the sample listener code from `ros.org`

Try

```
$ wget https://raw.github.com/ros/ros_tutorials/groovy-
 devel/roscpp_tutorials/listener/listener.cpp
```

talker.cpp Breakdown

Example

```
void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
  ROS_INFO("I heard: [%s]", msg->data.c_str());
}
```

- This is the callback function that will get called when a new message has arrived on the chatter topic

Example

```
ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback)
```

- Subscribe to the chatter topic with the master
- chatterCallback() function called whenever a new message arrives
- Queue size 1000 messages
- NodeHandle::subscribe() returns a ros::Subscriber object

talker.cpp Breakdown

Example

```
ros::spin();
```

- ros::spin() enters a loop, calling message callbacks as fast as possible

Building talker.cpp and listener.cpp

- Add the following to the bottom of your package's CMakeLists.txt

Example

```
## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(talker src/talker.cpp)
target_link_libraries(talker ${catkin_LIBRARIES})
add_dependencies(talker beginner_tutorials_generate_messages_cpp)

add_executable(listener src/listener.cpp)
target_link_libraries(listener ${catkin_LIBRARIES})
add_dependencies(listener beginner_tutorials_generate_messages_cpp)
```

Building talker.cpp and listener.cpp

- Return to your catkin_ws and perform catkin_make
- Make sure to source your catkin_ws after running catkin_make

Try

```
$ cd ~/catkin_ws
$ catkin_make
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

Running our Publisher

Try

```
$ rosrun beginner_tutorials talker
```

Example (Would return)

```
[ INFO] [1391150875.937043400]: hello world 0
[ INFO] [1391150876.037035000]: hello world 1
[ INFO] [1391150876.137120400]: hello world 2
[ INFO] [1391150876.237045280]: hello world 3
[ INFO] [1391150876.337045840]: hello world 4
[ INFO] [1391150876.437196640]: hello world 5
[ INFO] [1391150876.537125880]: hello world 6
[ INFO] [1391150876.637042200]: hello world 7
[ INFO] [1391150876.737260440]: hello world 8
```

Running our Subscriber

Try

```
$ rosrun beginner_tutorials listener
```

Example (Would return)

```
[ INFO] [1391150961.242276080]: I heard: [hello world 0]
[ INFO] [1391150961.343101800]: I heard: [hello world 1]
[ INFO] [1391150961.442861160]: I heard: [hello world 2]
[ INFO] [1391150961.542382440]: I heard: [hello world 3]
[ INFO] [1391150961.642622800]: I heard: [hello world 4]
[ INFO] [1391150961.742224720]: I heard: [hello world 5]
[ INFO] [1391150961.842046560]: I heard: [hello world 6]
[ INFO] [1391150961.942048280]: I heard: [hello world 7]
[ INFO] [1391150962.042433120]: I heard: [hello world 8]
```