Lecture 07.04 Other considerations

ROS topics have hardly been exhausted, and this will remain true even after we consider a few more aspects of special note.

07.04.1 The `rosmsg` command

The `rosmsg` command comes with the `rosbash` package already installed. It allows us to explore which messages are described and their descriptions.

As always, we need to navigate to our workspace.

```
cd ros_ws_01
```

Then source it!

```
source devel/setup.bash
```

The `show` option lists message descriptions. Even our `Complex` custom definition can be listed in this way.

```
rosmsg show Complex
```

```
[rico_topics/Complex]:
float32 real
float32 imaginary

[my_topics/Complex]:
float32 real
float32 imaginary
```

This is how we could see the message description of a `geometry_msgs` message `Point`.

```
rosmsg show geometry_msgs/Point
```

```
float64 x
float64 y
float64 z
```

The `package` option lets us list those messages defined in a given package.

```
rosmsg package my_topics
```

```
my_topics/Complex
```

For the `tf2_msgs` package, which groups the Error and Transform messages for `tf2_ros`, several message definitions are provided.

```
rosmsg package tf2_msgs
```

```
tf2_msgs/LookupTransformAction
tf2_msgs/LookupTransformActionFeedback
tf2_msgs/LookupTransformActionGoal
tf2_msgs/LookupTransformActionResult
tf2_msgs/LookupTransformFeedback
tf2_msgs/LookupTransformGoal
tf2_msgs/LookupTransformResult
tf2_msgs/TF2Error
tf2_msgs/TFMessage
```

The `list` option lists all messages available to ROS.

```
rosmsg list
```

We have suppressed the output, which is long.

07.04.2 Publishing and subscribing in the same node

Why not? This is actually rather common. Consider the example node `robotics-book-code/rico_topics/doubler.py`, listed in Figure 07.5. This node subscribes to topic `number`, multiplies the received `msg.data` (an `Int32`) by two, and publishes the result (an `Int32`) to topic `doubled`.

Perhaps the most interesting aspect of this is that, instead of publishing at some set rate, the publishing happens *inside the callback*. This means a new message will be published to `doubled` right after a new message is published to topic `number`. This is frequently the most desirable behavior.

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import Int32

rospy.init_node('doubler') # initialize node

def callback(msg):
    doubled = Int32()            # declare
    doubled.data = msg.data * 2 # double
    pub.publish(doubled)         # publish in callback!

sub = rospy.Subscriber('number', Int32, callback)
pub = rospy.Publisher('doubled', Int32, queue_size=3)

rospy.spin() # keep node running until shut down
```

Figure 07.5: `rico_topics/src/doubler.py` listing.