Topics

Prof. Tiago Vieira, PhD

Universidade Federal de Alagoas

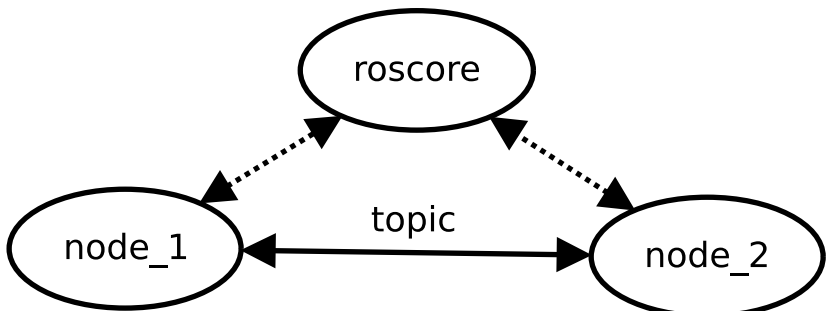
tvieira@ic.ufal.br

September 29, 2017

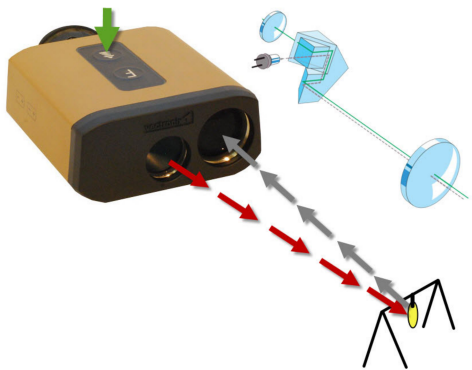
Contents

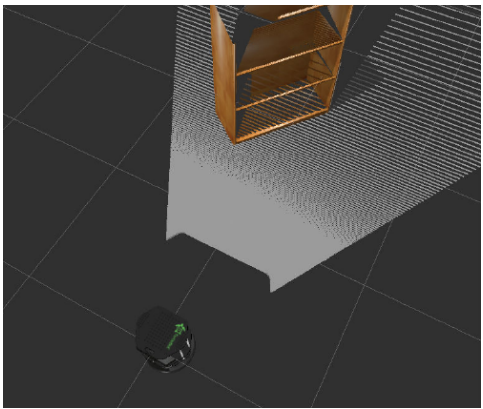
Introduction

- ROS systems comprise nodes forming a GRAPH.
- Data exchange happens through *topics*.
- A topic is a name for a stream of messages with a defined structure.

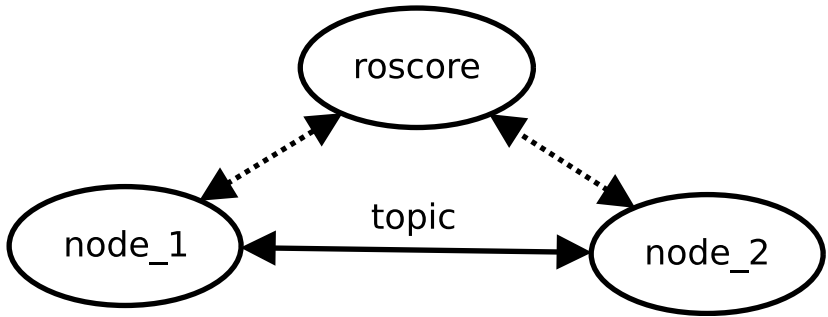


- Examples:
  - The data from a laser range-finders might be send on a topic called scan, with a message type of `LaserScan`.
  - The data from a camera might be sent over a topic called image, with a message type of `Image`.

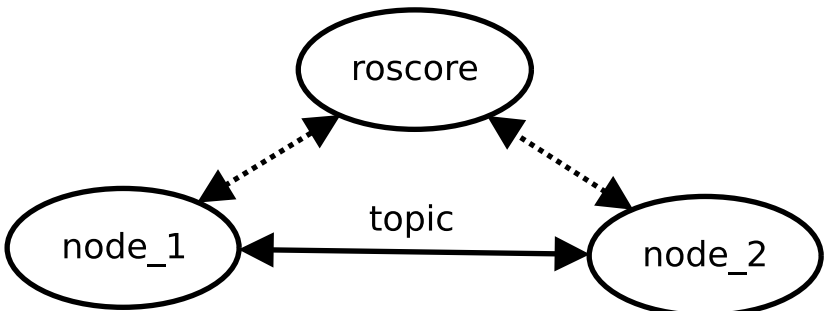




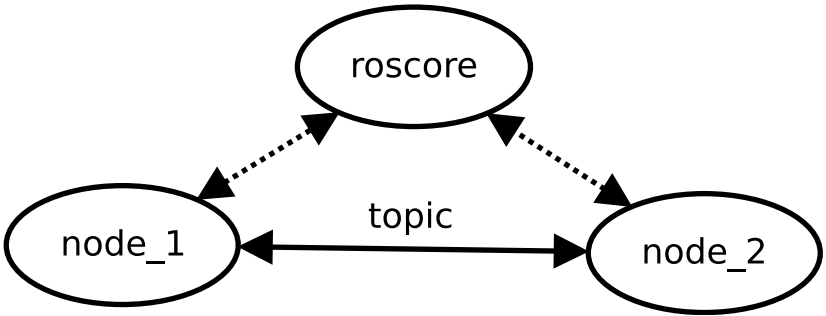
- Nodes must first announce, or advertize:
  - The topic name, and;
  - The type of messages.



- Then they can start to send, or publish, the actual data on the topic.
- Nodes that want to receive messages on a topic can subscribe to that topic by making a request to roscore.



- After subscribing, all messages on the topic are delivered to the node that make the request.
- Topics implement a *publish/subcribe* communications mechanism.



- Messages on the same topic **must** be of the same data type.
- Topic names describe messages sent over them:
  - `image_rgb`.
  - `image_depth`.
  - `position_xyz`.

Publishing to a Topic

How does a node advertises a topic and publishes data over it?

- Initiate a ROS core: `roscore`
- Start Python: $ python
- Type >>> `import sys; sys.path`
- Check if there is `/opt/ros/kinetic/lib/python2.7/dist-packages`
- Now let's write a script file

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import Int32
rospy.init_node('topic_publisher')
rate = rospy.Rate(2)
pub = rospy.Publisher('counter', data_class = Int32, queue_size = 1)
count = 0
while not rospy.is_shutdown():
    pub.publish(count)
    count += 1
    rate.sleep()
```

Steps:

1. Set the rate in Hz.
2. `is_shutdown()` function will return True if the node is ready to be shutdown, and False otherwise.
3. Inside the while loop, we publish the current value of the counter, increment its value by 1, and then sleep for a while.

Checking the functionality.

```
user@hostname$ rostopic list
/rosout
/rosout_agg
user@hostname$ rosrun topic_publisher.py
user@hostname$ rostopic list
/counter
/rosout
/rosout_agg
user@hostname$ rostopic echo counter -n 5
```

You can also find out about an advertised topic with `rostopic info`.

```
Type: std_msgs/Int32
Publishers:
* /topic_publisher (http://hostname:39964/)
```

Find all topics that publish a certain message type using `rostopic find`.

```
user@hostname$ rostopic find std_msgs/Int32
/counter
```

Subscribing to a topic

Subscribe to a topic and print its values as they arrive.

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import Int32
def callback(msg):
    print(msg.data)

rospy.init_node('topic_subscriber')
sub = rospy.Subscriber('counter', Int32, callback)
rospy.spin()
```

Steps:

1. Function `callback` handles the messages as they come in.
2. Subscribe to the topic `counter`. The subscribe passes this information on to `roscore`, and tries to make a direct connection with the publishers of this topic.
3. Give control over to ROS by running `rospy.spin()`. But we can use a `while` as previously.

Checking the functionality.

```
user@hostname$ rosrun basics topic_subscriber
```

```
user@hostname$ rostopic pub counter std_msgs/Int32 1000000
```

```
user@hostname$ rostopic info counter

Type: std_msgs/Int32

Publishers:
 * /topic_publisher (http://tvieira-HP-ProBook-640-G1:43013/)

Subscribers:
 * /topic_subscriber (http://tvieira-HP-ProBook-640-G1:34537/)
```

Latched Topics

Defining New Message Types

ROS's built-in message types:

- `std_msgs`
  - Booleans.
  - Integers.
  - Floating point numbers.
  - Strings.
  - Arrays.
- `sensor_msgs`
  - Laser-range finders.
  - Cameras.
- `geometry_msgs`
  - Positions.
  - Rotations.
  - Derivatives.