

ROB - Basics of ROS

Adam Fabo

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2. 602 00 Brno - Královo Pole
xfaboa00@fit.vutbr.cz



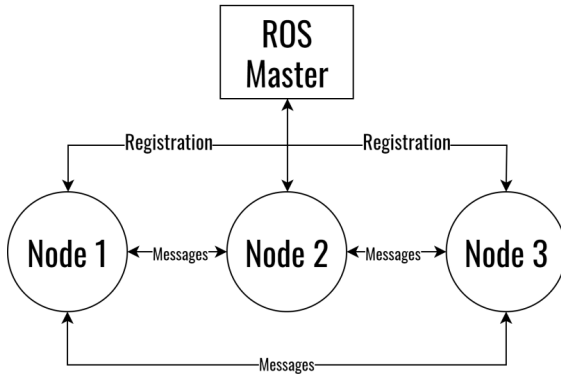
March 23, 2023

Today's lab will be focused on basics of ROS.

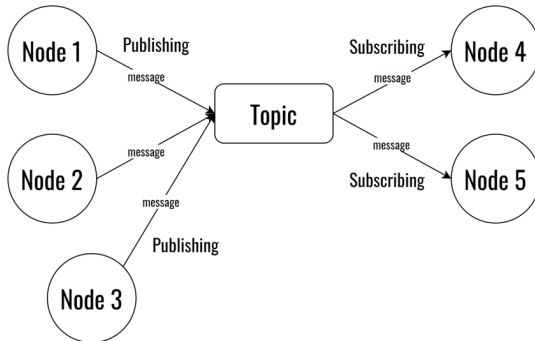
- The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications.
- It is used for prototyping and building robot applications
- Companies like Boston Dynamics or Kuka use ROS or ROS-derived tools



- Main parts of ROS architecture are: ROS master, Node, Topic, Message.
- Single instance of ROS master needs to run, created nodes are registered to ROS master



- Nodes exchange messages through topics
- Node can either publish or subscribe messages to/from topic



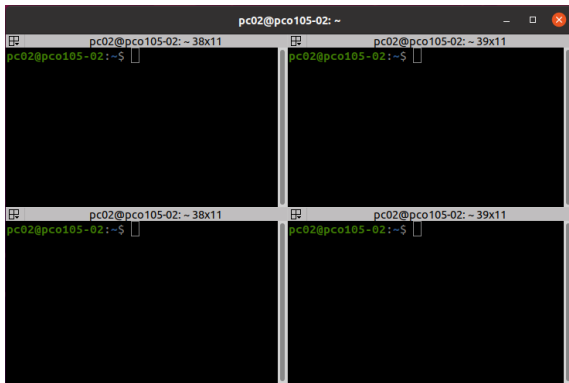
ROS offers command line tools for working/debugging in ROS environment:

- **roscore** - Starts ROS master. Run this command in one terminal window and keep it running until you finish your work.
- **roslaunch** (package) (node) - Starts node. When starting node, package of the node must be specified.

Following commands can be used for displaying info about node/topic/message:

- **rqt_graph** - Opens new window where all Nodes and Topics are visualized.
- **rostopic** - command-line tool that displays information about ROS Topics.
- **rostopic** - command-line tool that displays information about ROS topics.
- **rosmmsg** - command-line tool that displays information about ROS messages.

- It is good practice to have multiple terminal windows opened when working with ROS.
- This is the reason why application like Terminator where window can be split is good to use.



- Open new terminal using **Terminator** app
- Split the terminal into 4 parts.
- Click onto top left terminal and run `roscore` command
- You should see last line of the message say: "started core service [/rosout]"
- Do not exit this terminal, let it run and use other terminal windows

- Create publisher by running command: `roslaunch rospy_tutorials talker`
- In another window, create subscriber by running command: `roslaunch rospy_tutorials listener`
- Now, the publisher is sending messages and subscriber is listening to them
- Visualize this by running command: `rqt_graph`
- A new window should open where you can see visual representation of your running nodes



- Keep the talker and listener from last exercise running
- Run `rostopic list` to see running nodes
- Run `rostopic info /[node name]` where node name is name of one of the node listed by previous command (hint: you can use TAB to auto-complete node names)
- Run `rostopic list` to see active topics
- Run `rostopic info /chatter` to see type of message that this topic accepts and also publishers and subscribers that interact with this topic
- Run `rostopic list` to see list of all message types
- Run `rostopic show std_msgs/String` to see info about one message type

- ROS offers a way to create custom nodes in c++ and Python
- In this lab ROS Python API will be used, called rospy

Simplest ROS publisher can be written in Python as following:

```
5  # Import everything important
6  import rospy
7  from std_msgs.msg import String
8
9  # Start of the program
10 if __name__ == '__main__':
11     # tell ROS name of this node
12     rospy.init_node('Basic_Publisher')
13
14     # Create publisher that sends messages to topic "/chatter" and message type is String
15     pub = rospy.Publisher('/chatter', String, queue_size=10)
16
17     # Set message speed to 2Hz
18     r = rospy.Rate(2)
19
20     # Infinite loop while ROS is running
21     while not rospy.is_shutdown():
22         # Send your own custom message
23         pub.publish("Your message here")
24         # Sleep for a given time
25         r.sleep()
```

Sometimes it is advantageous to write a publisher that sends only single message.

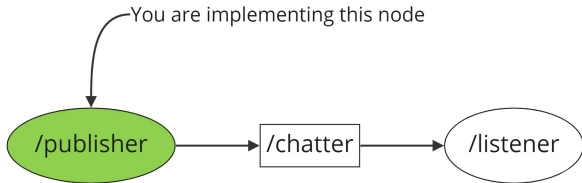
```
13 pub = rospy.Publisher('/chatter', String, queue_size=10)
14
15 while pub.get_num_connections() < 1:
16     pass
17
18 pub.publish("Your message here")
```

The simplest ROS subscriber can be written in Python as the following:

```
5  # Import everything important
6  import rospy
7  from std_msgs.msg import String
8
9
10 # Callback function that is called when subscriber gets data
11 def callback(data):
12     rospy.loginfo("Received data: %s", data.data)
13
14
15 # Start of the program
16 if __name__ == '__main__':
17     # tell ROS name of this node
18     rospy.init_node('Basic_Subscriber', anonymous=True)
19
20     # Create subscriber that subscribes messages from topic "/chatter" of type String
21     # callback function is called when message is received
22     rospy.Subscriber("/chatter", String, callback)
23     rospy.spin()
```

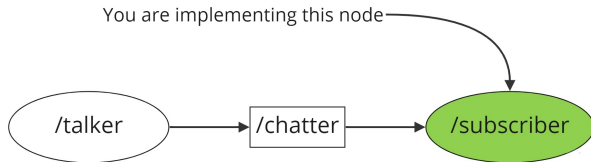
Create your own publisher that sends messages to topic /chatter

- If you have a talker running from the last exercise, kill it. Run only listener
- Open PyCharm by running command `pycharm` in terminal window
- Navigate to file **basic_publisher/src/publisher.py**
- Implement publisher that publishes String message to /chatter topic every 2 seconds.
- Run the node by running `roslaunch basic_publisher publisher.py`
- Take a look at the listener - it should be receiving your messages



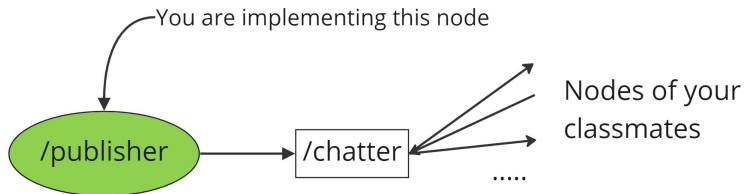
Create your own subscriber that subscribes messages from topic /chatter

- If you have a listener running from the last exercise, kill it.
Run only talker
- Open PyCharm by running command `pycharm` in terminal window (or keep it open from last exercise)
- Navigate to file **basic_subscriber/src/subscriber.py**
- Implement subscriber that subscribes String message from /chatter
- Run the node by running `roslaunch basic_subscriber subscriber.py`
- Take a look at the output from your code - it should be receiving messages



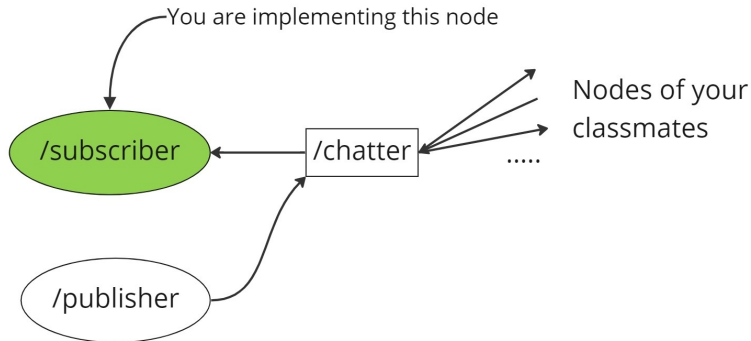
Modify publisher in a way, that user can write custom messages. Make it so that messages are shared between all students.

- Continue working in your previous publisher code, or open file **basic_publisher/src/publisher_input.py**
- Modify publisher by adding function `input()` that reads input from keyboard
- Change topic to `/shared_chatter`
- Call the teacher in this step, or stop running the ROS master and change `ROS_MASTER_URI` in **.bashrc** file
- Run your node with `roslaunch basic_publisher publisher_input.py` and write some messages



Modify subscriber in a way, so that you can receive messages from your classmates.

- Continue working in your previous publisher code, or open file:
basic_subscriber/src/subscriber_auto_reply.py
- Change topic to /shared_chatter
- Run your node with `roslaunch basic_subscriber subscriber_auto_reply.py` and write some messages using your previous publisher. You should see your messages and messages from your classmates

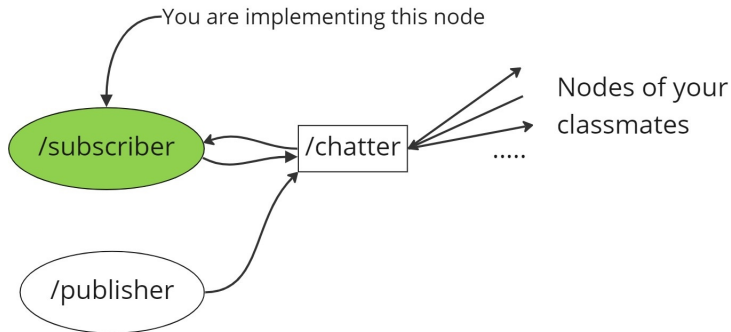


Modify subscribers callback function in a way, that it sends automatic response if it detects word "Hello". Respond by word "Hello" from your own language,

- Continue working in your previous publisher code, or open file:

basic_subscriber/src/subscriber_auto_reply.py

- Modify your subscriber's callback function. Check if received string contains "Hello"
- If string contains "Hello" write single response that contains "Hello" in your own language
- You need to create a new publisher in callback function, that sends only one message.
- Test your implementation by writing "Hello" to the topic.



Thank You For Your Attention !