SC-635 Advanced Topics in Mobile Robotics

Experiment Module : Introduction to ROS

January 17, 2020

Systems and Control Engineering

Indian Institute of Technology Bombay

Overview

1. ROS : About
2. Workspace settings
3. ROS Concepts and Commands

Supported robots

Exhaustive list of supported robots is available at:

https://robots.ros.org/

What is ROS

Open source framework that facilitates

- Running multiple small programs (nodes) concurrently
- Establish p2p communication between these programs (nodes)

What is ROS

Open source framework that facilitates

- Running multiple small programs (nodes) concurrently
- Establish p2p communication between these programs (nodes)

Interfaced to many useful softwares such as

- Gazebo : Physics simulation
- RViz : 3D robot visualization
- rqt : Framework for graphical UIX with ROS

What is ROS

Open source framework that facilitates

- Running multiple small programs (nodes) concurrently
- Establish p2p communication between these programs (nodes)

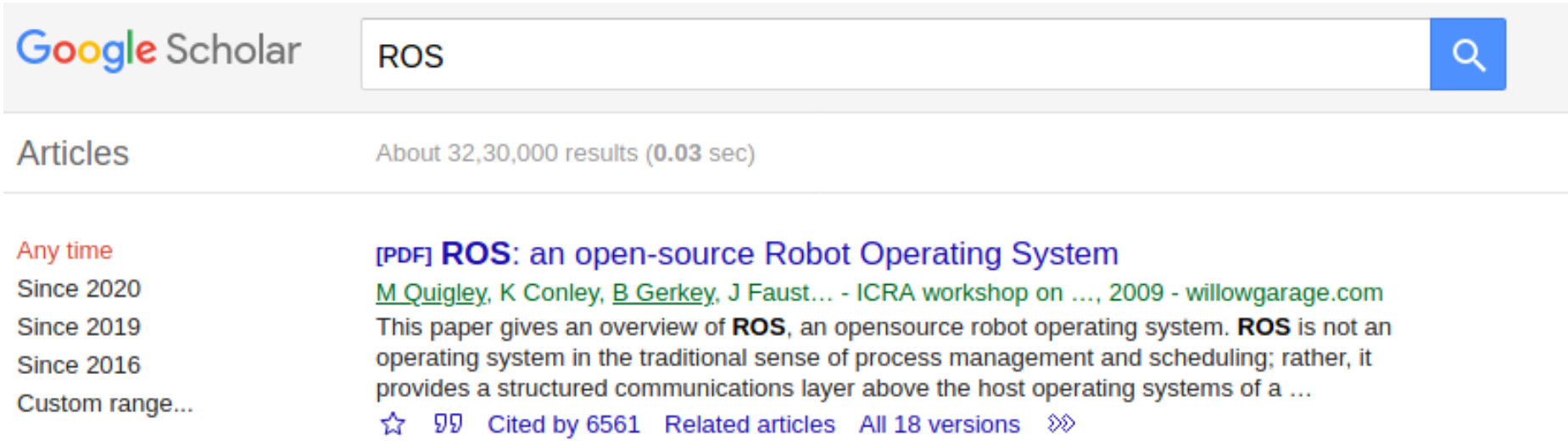
Interfaced to many useful softwares such as

- Gazebo : Physics simulation
- RViz : 3D robot visualization
- rqt : Framework for graphical UIX with ROS

Why it is wildly popular

- Robotics specific algorithms available as ROS package
- Allows write once and run everywhere (TurtleBot, PR2, Husky, AR Drone)
- Out of the box heterogeneous computing

ROS Popularity



6561 Citations of the original ROS paper[1]

[1] source scholar.google.com

Binary Downloads by rosdistro

The fraction of packages downloaded per rosdistro

- Hydro and earlier: 0.00 %
- Indigo: 5.02 % (Long Term Support Release)
- Jade: 0.00 %
- Kinetic: 53.06 % (Long Term Support Release)
- Lunar: 0.85 %
- Melodic: 26.71 % (Long Term Support Release)
- Bouncy and earlier ROS 2: 0.12 %
- Crystal 0.62 %
- Dashing 1.51 % (Long Term Support Release)
- Rosdistro independent: 12.10% (packages like python-rosdistro and python-rosinstall, as well as backported 3rdparty libraries like pcl and colladadom)

Source: Apache2, counting only downloads from the main repository (not the testing repository, shadow-fixed, or mirrors)

[2]

[2] source download.ros.org/downloads/metrics/metrics-report-2019-07.pdf

ROS workspace

- Workspace
  Directory where ROS project/code is located. Can work with only one at a time.

ROS workspace

- Workspace
  Directory where ROS project/code is located. Can work with only one at a time.
- Creating a workspace
  $ catkin_init_workspace
  $ catkin_make consequences ..
  ../devel/
  ../build/

ROS workspace

- Workspace
  Directory where ROS project/code is located. Can work with only one at a time.
- Creating a workspace
  $ catkin_init_workspace
  $ catkin_make consequences ..
  ../devel/
  ../build/
- Activating a workspace
  $ source devel/setup.bash

Roscore, Nodes, Topics, and Messages

Roscore

- The program that needs to run first
- Node discovery, topic discovery and other book-keeping task

Node

- A program with independent existence
- Can subscribe to a topic and receive a specific type of messages
- Can publish message to a topic

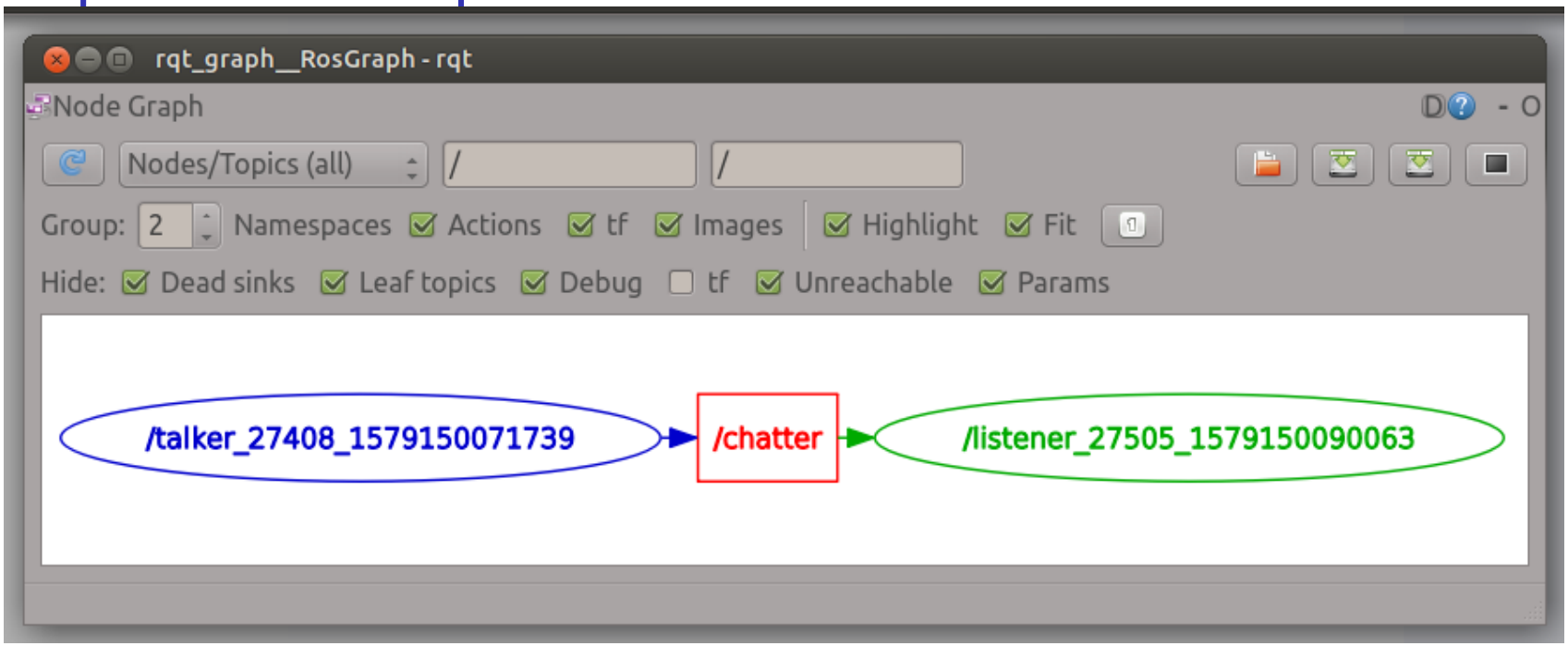
Topic

- Name for stream of messages. Nodes do not talk directly to each other, they communicate with Topics

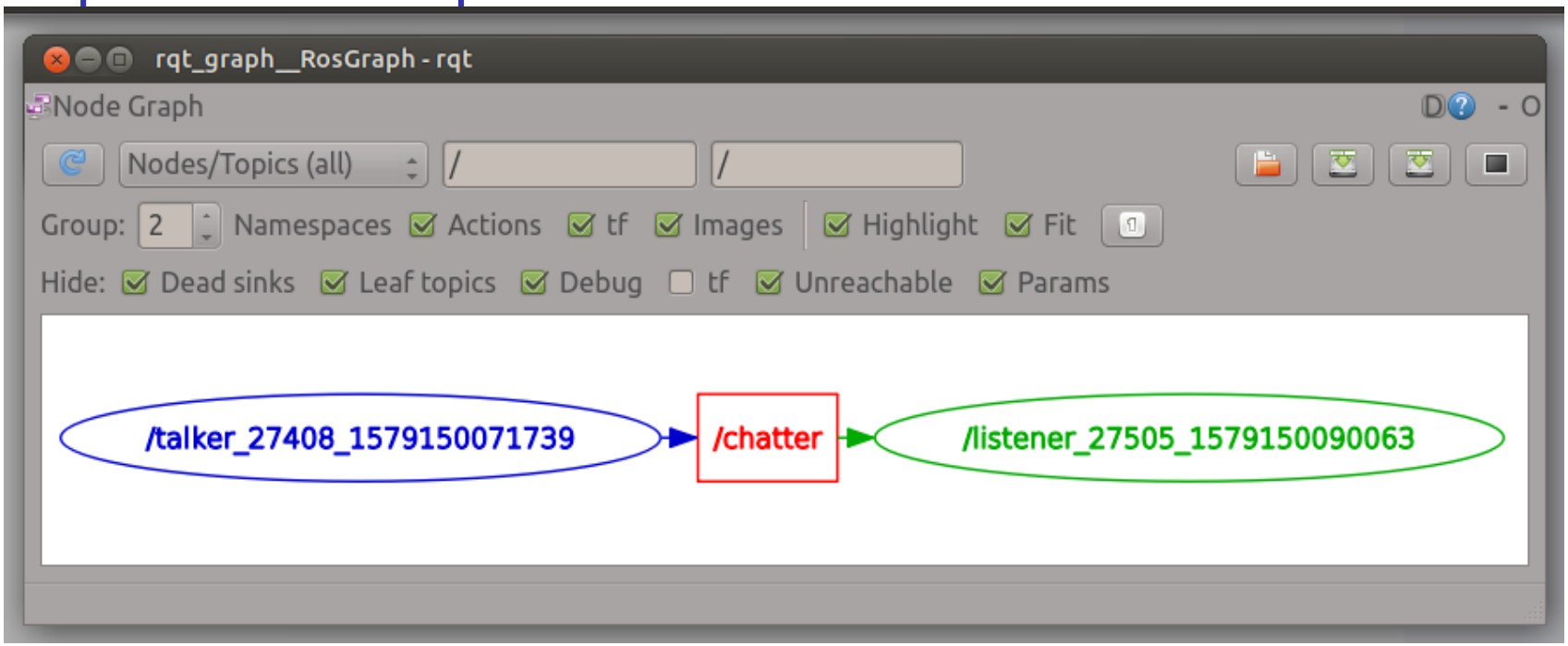
Message

- ROS messages are formatted data (akin to datatypes in programming languages)

A simple ROS-Graph



A simple ROS-Graph



Execute each command in a new terminal window/tab:

```
$ roscore
$ rosrun rospy_tutorials talker.py
$ rosrun rospy_tutorials listener.py
$ rqt_graph
```

Tip: Use Ctrl+C to stop the process (nodes)

Node 1: talker.py

```
1  #!/usr/bin/env python
2  import rospy
3  from std_msgs.msg import String
4
5  def talker():
6      pub = rospy.Publisher('chatter', String, queue_size
       =10)
7      rospy.init_node('talker', anonymous=True)
8      rate = rospy.Rate(10) # 10hz
9      while not rospy.is_shutdown():
10         hello_str = "hello world %s" % rospy.get_time()
11         rospy.loginfo(hello_str)
12         pub.publish(hello_str)
13         rate.sleep()
14
15 if __name__ == '__main__':
16     try:
17         talker()
18     except rospy.ROSInterruptException:
19         pass
```

Node 2: listener.py

```
1  #!/usr/bin/env python
2  import rospy
3  from std_msgs.msg import String
4
5  def callback(data):
6      rospy.loginfo(rospy.get_caller_id() + 'I heard %s',
       data.data)
7
8  def listener():
9
10     rospy.init_node('listener', anonymous=True)
11
12     rospy.Subscriber('chatter', String, callback)
13
14     # spin() simply keeps python from exiting until
       this node is stopped
15     rospy.spin()
16
17 if __name__ == '__main__':
18     listener()
```

Creating a package

- Inside the catkin_ws/src/ directory

  $ catkin_create_project test_pub_sub rospy
- Create a **script** directory inside the newly created test_pub_sub directory

  $ mkdir scripts
- Inside the **scripts** directory create two files talker.py and listener.py

  $ touch talker.py listener.py
- Give execute permissions to both the files

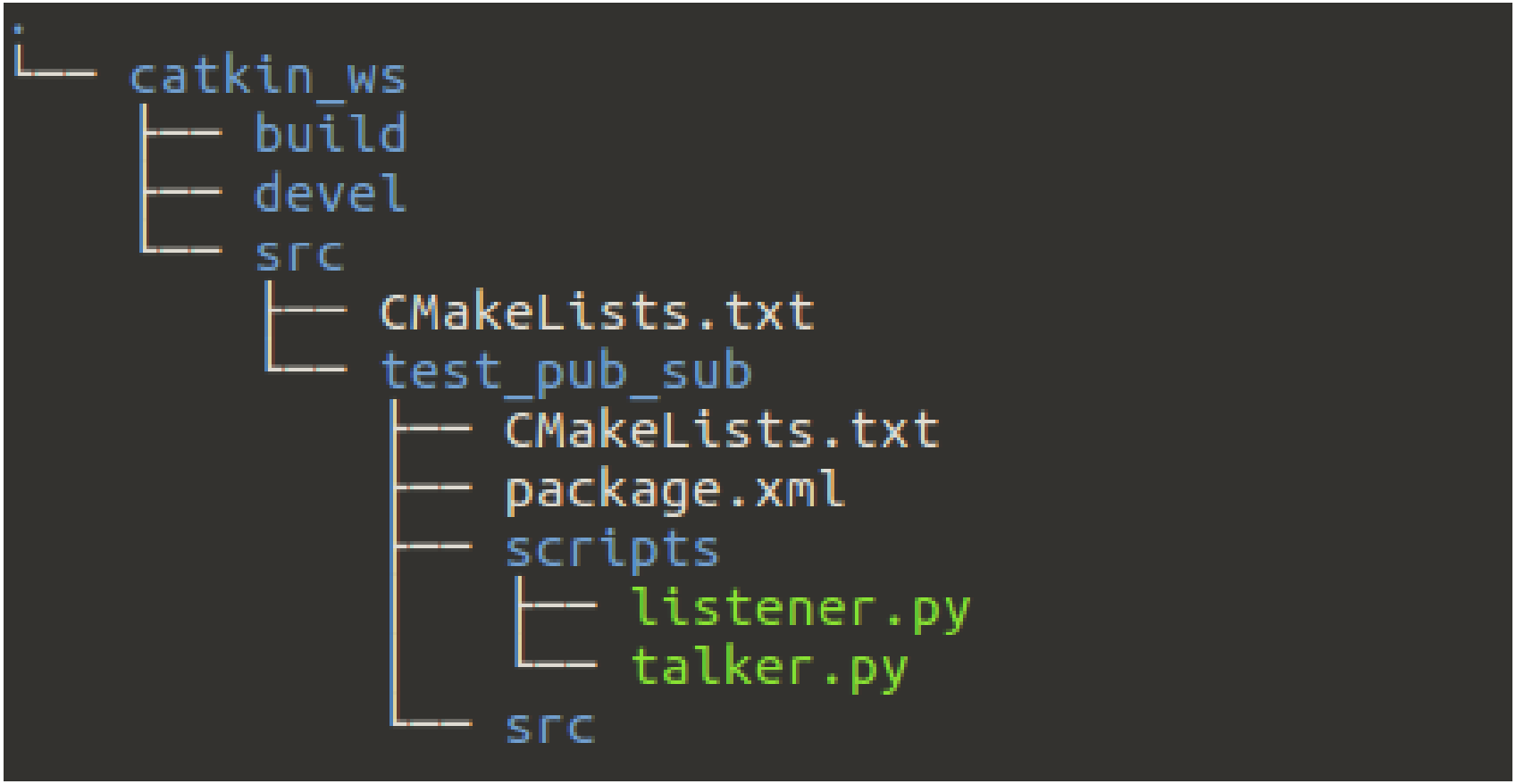
  $ chmod +x talker.py listener.py
- Add the code to the files

  bit.ly/2tdpTalk

  bit.ly/2tdpListen
- Go to catkin_ws directory and run

  $ catkin_make

Directory structure so far



Run the nodes

Execute each command in a new terminal window/tab:

$ roscore

$ rosrun test_pub_sub talker.py

$ rosrun test_pub_sub listener.py

$ rqt_graph

Run the nodes

Execute each command in a new terminal window/tab:

```
$ roscore
$ rosrun test_pub_sub talker.py
$ rosrun test_pub_sub listener.py
$ rqt_graph
```

Before running the nodes we need to **activate** the current workspace.

Go to catkin_make directory and run the following command

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

Frequently used ROS Commands

- rosnode
- rostopic
- roscd
- rosrun
- roslaunch

Launch file

- Create a **launch** directory inside test_pub_sub directory

  $ mkdir launch
- Inside **launch** directory create a **pubsub.launch** file

  $ touch pubsub.launch
- Add following code to the file

  bit.ly/2tdpLaunchBasic
- Now we can start both the nodes by invoking the launch file. Before that we will build the project
- Go to catkin_ws directory and run

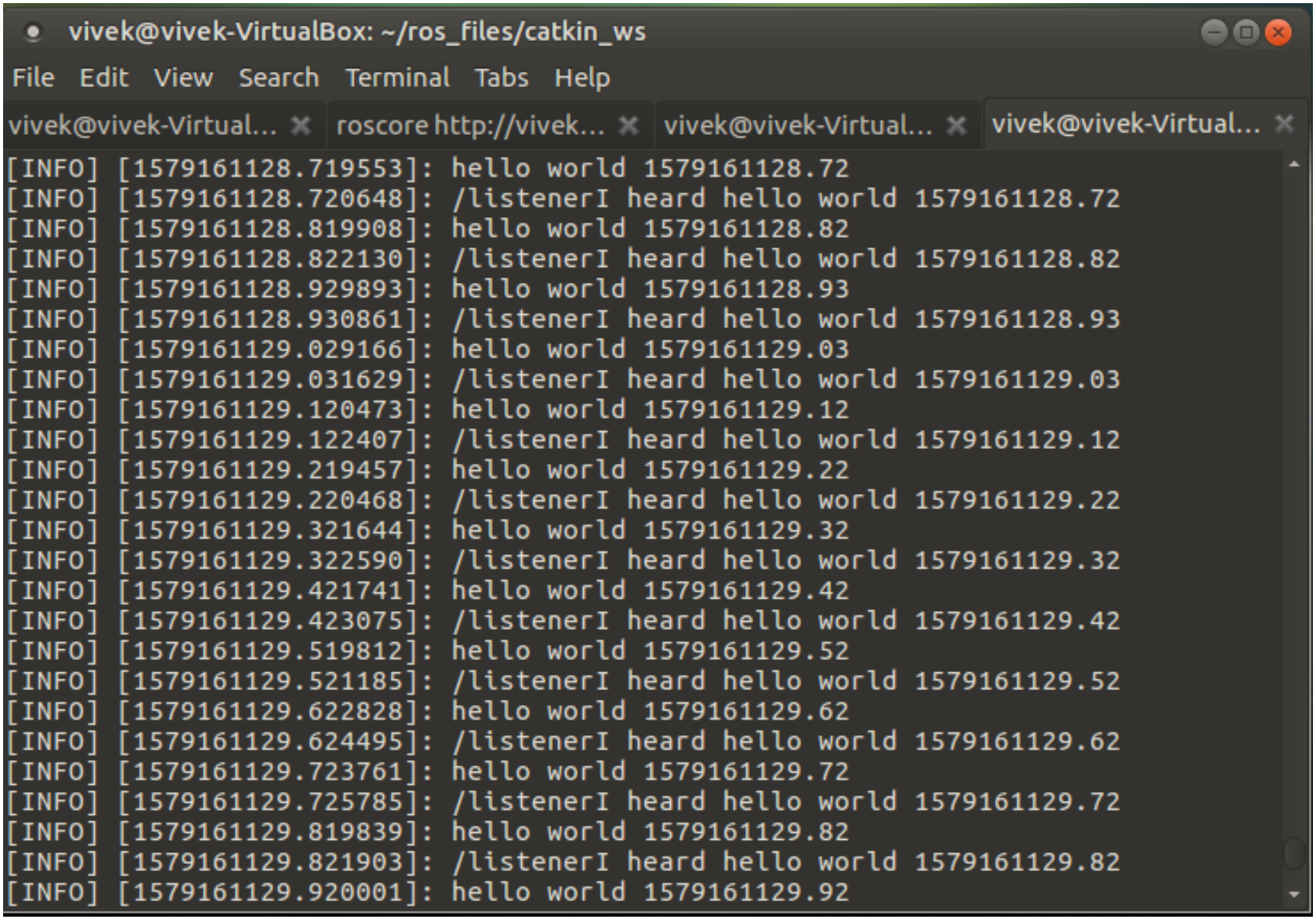
  $ catkin_make
- Start the launch file with

  $ roslaunch test_pub_sub pubsub.launch

Launch file

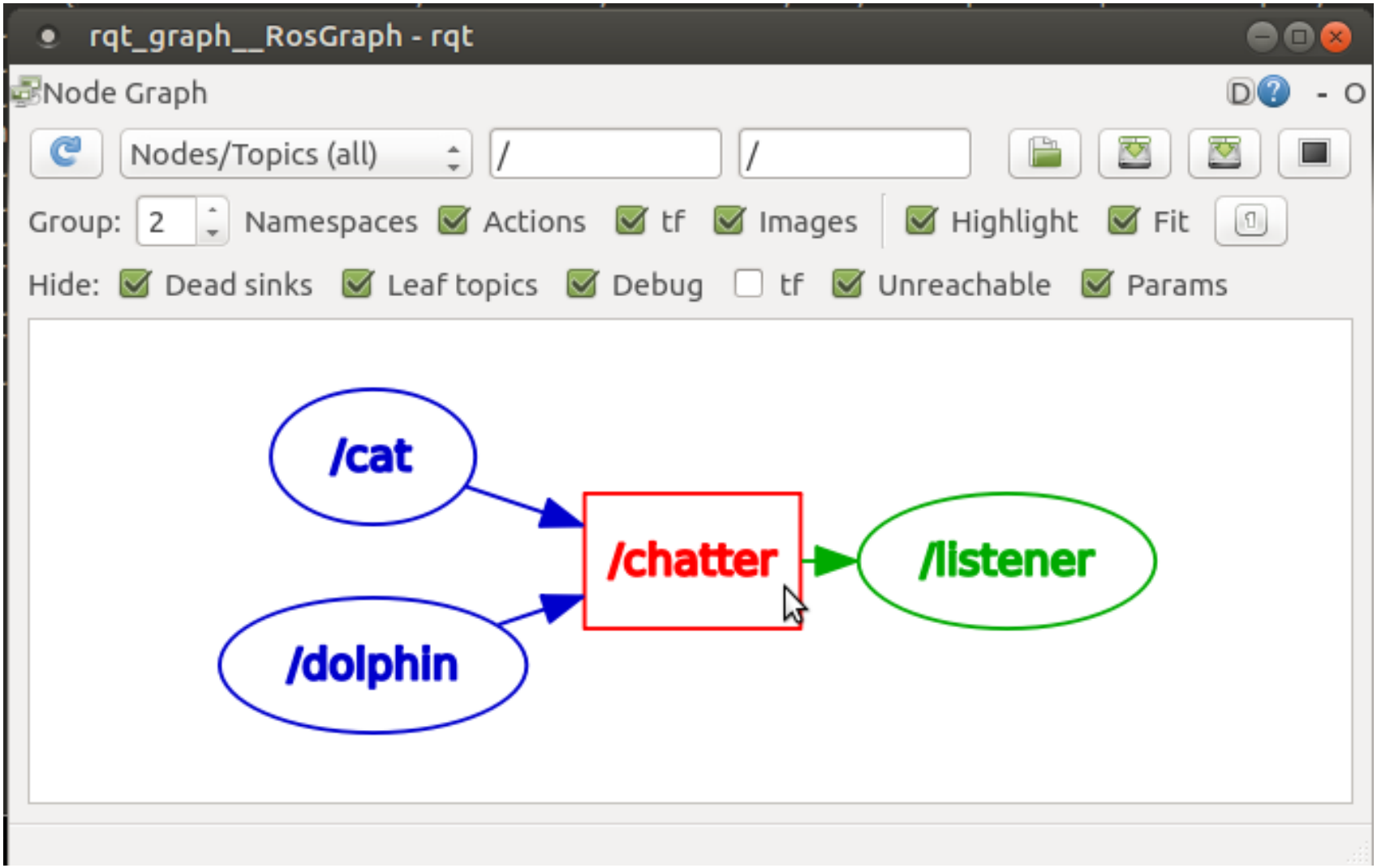
```
1 <launch>
2     <node name="talker" pkg="test_pub_sub" type="talker
      .py" output="screen" />
3     <node name="listener" pkg="test_pub_sub" type="
      listener.py" output="screen" />
4 </launch>
```

Merged outputs



Passing arguments

- Multiple instances of same node with a variation
  Example : Two versions of talker node sending different msg (as shown in the following ROS graph)



Launch file with arguments

```
1 <launch>
2     <arg name="dolphin_msg" default="Click_and_Squeak" />
3     <arg name="cat_msg" default="Meow_and_Purr" />
4
5     <node name="dolphin" pkg="test_pub_sub" type="
      talker_mod.py" args="$(arg dolphin_msg)" />
6     <node name="cat" pkg="test_pub_sub" type="
      talker_mod.py" args="$(arg cat_msg)" />
7     <node name="listener" pkg="test_pub_sub" type="
      listener.py" output="screen" />
8 </launch>
```

3

[3] location bit.ly/2tdpLaunchMod

Node modification

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

def talker(greeting):
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10)
    while not rospy.is_shutdown():
        hello_str = ": {}".format(greeting)
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker(sys.argv[1])
    except rospy.ROSInterruptException:
        pass
```

4

[4] location bit.ly/2tdpTalkerMod

Assignment

- Create a new project with name in following format

  NAME_ROLL_NUMBER
- Three nodes and one launch file
  - First will publish your NAME (String) to topic **name_listner**
  - Second will publish your ROLL NUMBER (Int) **roll_listener**
  - Third node will subscribe to both the above topics and print a assimilated string in following format

    Student **NAME** has roll number : **ROLL_NUMBER**