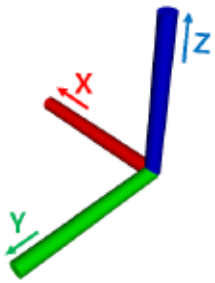
ECE 4703
Mobile Autonomous Robots

Jenny Zhen Yu
zhenyu@cpp.edu
Department of Electrical and Computer Engineering
California State Polytechnic University, Pomona

Lecture 3: ROS Programming

Things to Know Before Programming ROS

- **Standard unit**
  - SI unit



Quantity	Unit
angle	radian
frequency	hertz
force	newton
power	watt
voltage	volt

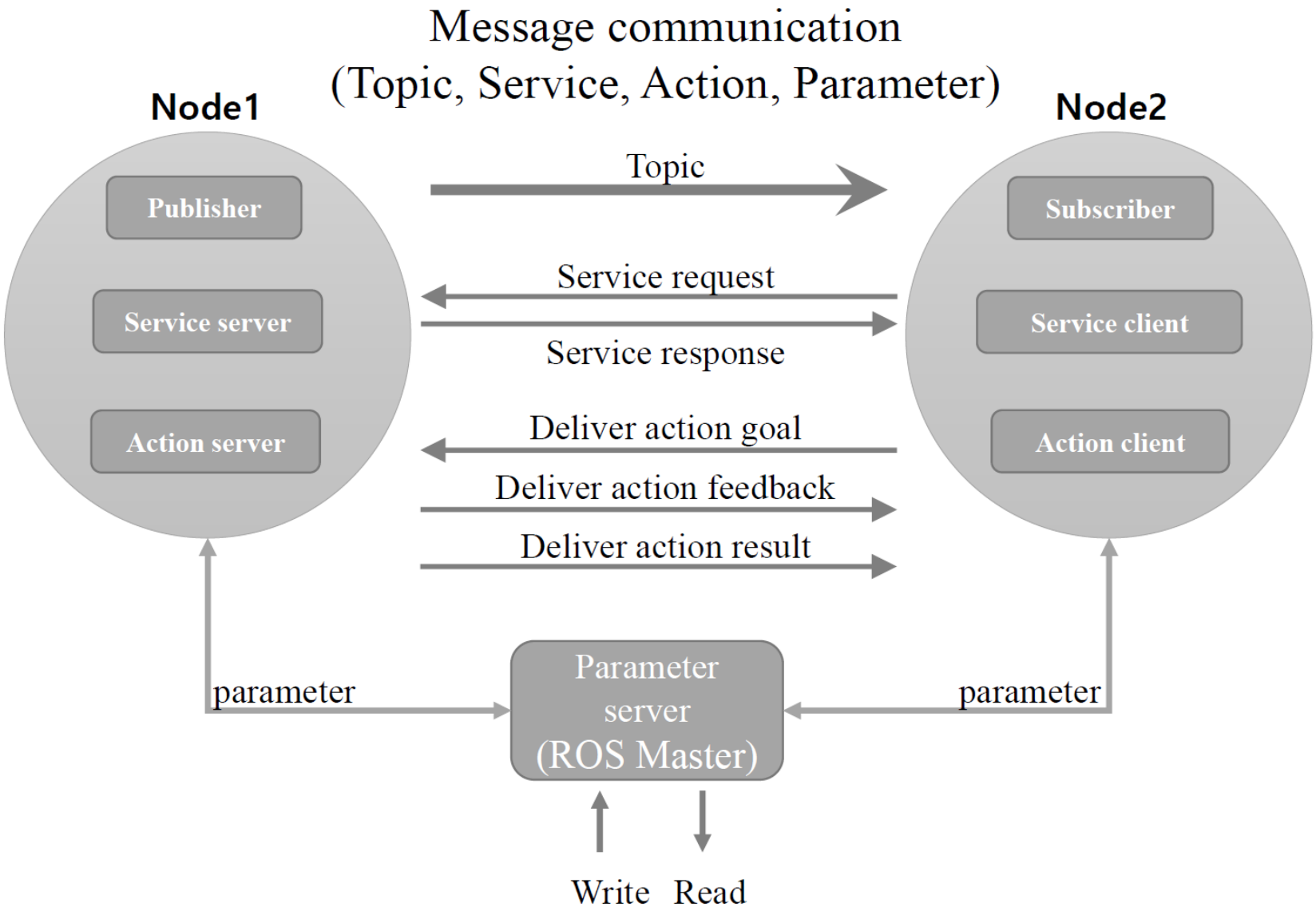
Quantity	Unit
length	meter
mass	kilogram
time	second
current	ampere
temperature	celsius

- **Coordinate representation**
  - x: forward, y: left, z: up
  - Right-hand rule

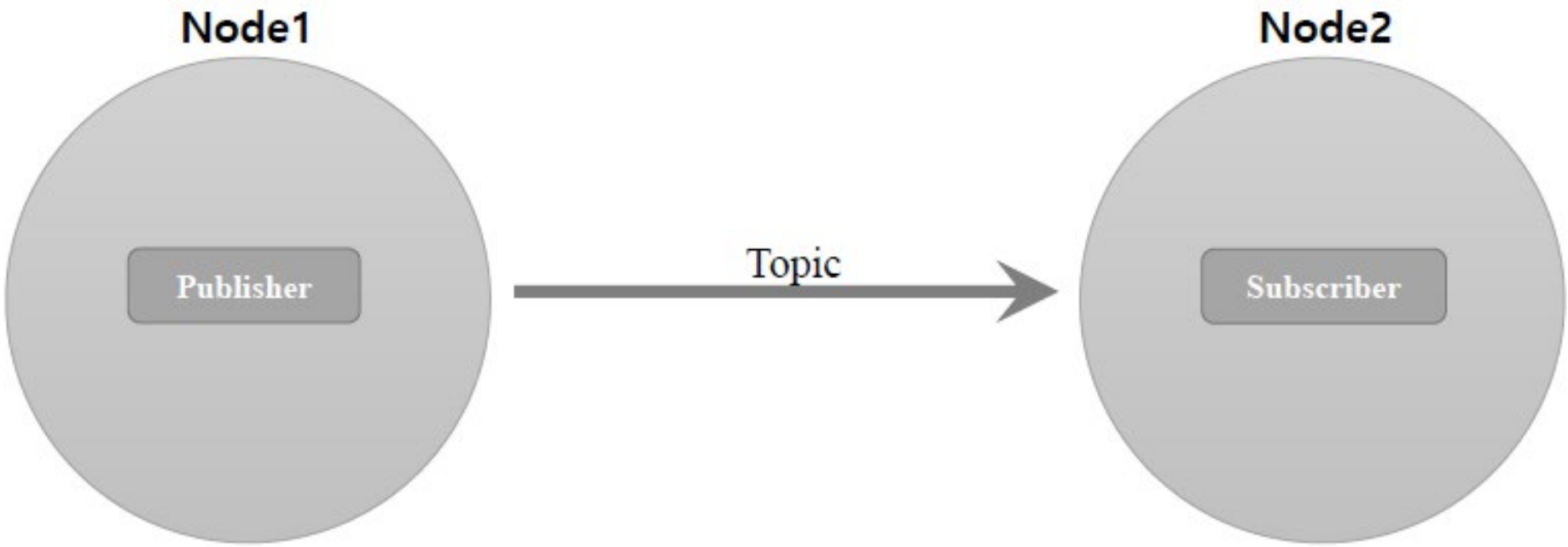
- **Programing rules**

Object	Naming rule	example
Package	under_scored	Ex) first_ros_package
Topic, service	under_scored	Ex) raw_image
File	under_scored	Ex) turtlebot3_fake.cpp
Namespace	under_scored	Ex) ros_awesome_package
Variable	under_scored	Ex) string table_name;
type	CamelCased	Ex) typedef int32_t PropertiesNumber;
Class	CamelCased	Ex) class UrlTable
Structure	CamelCased	Ex) struct UrlTableProperties
Enumeration type	CamelCased	Ex) enum ChoiceNumber
Function	camelCased	Ex) addTableEntry();
Method	camelCased	Ex) void setNumEntries(int32_t num_entries)
Constant	ALL_CAPITALS	Ex) const uint8_t DAYS_IN_A_WEEK = 7;

ROS Message Communication



Topic



Topic / Publisher / Subscriber

- ROS uses 'Topic' message communication for unidirectional communication. In this tutorial, the transmitter is called "Publisher" and the receiver is called "Subscriber".

1) Creating the package

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg ros_tutorials_topic message_generation std_msgs roscpp
```

```
$ cd ros_tutorials_topic
$ ls
include          → header file folder
src              → source code folder
CMakeLists.txt   → build configuration file
package.xml      → package configuration file
```

Topic / Publisher / Subscriber

2) Modify the package configuration file (package.xml)

- One of the required ROS configuration files, package.xml, is an XML file containing package information that describes the package name, author, license, and dependency package.

```
$ gedit package.xml
```

```
<?xml version="1.0"?>
<package>
  <name>ros_tutorials_topic</name>
  <version>0.1.0</version>
  <description>ROS turtorial package to learn the topic</description>
  <license>Apache License 2.0</license>
  <author email="pyo@robotis.com">Yoonseok Pyo</author>
  <maintainer email="pyo@robotis.com">Yoonseok Pyo</maintainer>
  <url type="bugtracker">https://github.com/ROBOTIS-GIT/ros_tutorials/issues</url>
  <url type="repository">https://github.com/ROBOTIS-GIT/ros_tutorials.git</url>
  <url type="website">http://www.robotis.com</url>
```

Topic / Publisher / Subscriber

```
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>message_generation</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>message_runtime</run_depend>
  <export></export>
</package>
```

Topic / Publisher / Subscriber

3) Modify build configuration file (CMakeLists.txt)

```
$ gedit CMakeLists.txt
```

```
cmake_minimum_required(VERSION 2.8.3)
project(ros_tutorials_topic)

## This is the component package required for catkin build.
##  dependency packages are message_generation, std_msgs, and roscpp. If these packages do not exist, an error occurs when you build.
find_package(catkin REQUIRED COMPONENTS message_generation std_msgs roscpp)

## Message declaration: MsgTutorial.msg
add_message_files(FILES MsgTutorial.msg)

## This is an option to configure dependent messages.
## If std_msgs is not installed, an error occurs when you build.
generate_messages(DEPENDENCIES std_msgs)

## The catkin_package option describes the library, catkin build dependencies, and system dependent packages.
catkin_package(
  LIBRARIES ros_tutorials_topic
  CATKIN_DEPENDS std_msgs roscpp
)
```

Topic / Publisher / Subscriber

```
## Set the include directory.
include_directories(${catkin_INCLUDE_DIRS})

## Build option for the topic_publisher node.
## Configure the executable file, target link library, and additional dependencies.
add_executable(topic_publisher src/topic_publisher.cpp)
add_dependencies(topic_publisher ${${PROJECT_NAME}_EXPORTED_TARGETS}
${catkin_EXPORTED_TARGETS})
target_link_libraries(topic_publisher ${catkin_LIBRARIES})

## Build option for the topic_subscriber  node.
add_executable(topic_subscriber src/topic_subscriber.cpp)
add_dependencies(topic_subscriber ${${PROJECT_NAME}_EXPORTED_TARGETS}
${catkin_EXPORTED_TARGETS})
target_link_libraries(topic_subscriber ${catkin_LIBRARIES})
```

Topic / Publisher / Subscriber

4) Create message file

- Add the following option in the CMakeLists.txt file.

```
add_message_files(FILES MsgTutorial.msg)
```

- This commands means a message should be built based on MsgTutorial.msg when it is built.

```
$ roscd ros_tutorials_topic      → Move to package folder
$ mkdir msg                      → Create a message folder named msg in the package
$ cd msg                         → Move  to the msg folder you created
$ gedit MsgTutorial.msg          → Create new MsgTutorial.msg file and modify contents
```

- Time (message format), stamp (message name)
- int32 (message type), data (message name)
- In addition to time and int32, message types include message basic types such as bool, int8, int16, float32, string, time, duration, and common_msgs which collect messages used in ROS. Here we use time and int32 to create a simple example. (See Appendix C and http://wiki.ros.org/msg)

```
time stamp
int32 data
```

Topic / Publisher / Subscriber

5) Creating the Publisher Node

- Add the option to generate the following executable file in the CMakeLists.txt file.

```
add_executable(topic_publisher src/topic_publisher.cpp)
```

- Build a file called topic_publisher.cpp in the src folder to create an executable called topic_publisher

```
$ roscd ros_tutorials_topic/src          → Move to the src folder, which is the source folder of the package
$ gedit topic_publisher.cpp              → New source file and modify contents
```

```
#include "ros/ros.h"                     // ROS basic header file
#include "ros_tutorials_topic/MsgTutorial.h"     // MsgTutorial message file header(Auto-generated after build)

int main(int argc, char **argv)                  // Node main function
{
  ros::init(argc, argv, "topic_publisher");      // Node name initialization
  ros::NodeHandle nh;                            // Node handle declaration for communication with ROS system
```

Topic / Publisher / Subscriber

```
 // Publisher declaration, using MsgTutorial message file from ros_tutorials_topic package
 // Create the publisher ros_tutorial_pub. The topic name is "ros_tutorial_msg"
 // set the publisher queue size to 100
ros::Publisher ros_tutorial_pub = nh.advertise<ros_tutorials_topic::MsgTutorial>("ros_tutorial_msg", 100);

// Set the loop period. & Quot; 10 & quot; refers to 10 Hz and repeats at 0.1 second intervals
ros::Rate loop_rate(10);

 // Declare msg message in MsgTutorial message file format
ros_tutorials_topic::MsgTutorial msg;

// Declare variable to be used in message
int count = 0;
```

Topic / Publisher / Subscriber

```
while (ros::ok())
{
  msg.stamp = ros::Time::now();        // Put the current time in the msg's substamp message
  msg.data  = count;                   // Put the value of the variable count in the lower data message of msg

  ROS_INFO("send msg = %d", msg.stamp.sec);          // Display the stamp.sec message
  ROS_INFO("send msg = %d", msg.stamp.nsec);         // Display the stamp.nsec message
  ROS_INFO("send msg = %d", msg.data);               // Display  data message

  ros_tutorial_pub.publish(msg);                     // Publish message

  loop_rate.sleep();                                 //  Go to sleep according to the loop cycle defined above

  ++count;                                           //  Increment count variable by 1
 }

 return 0;
}
```

Topic / Publisher / Subscriber

6) Create subscriber node

- In the CMakeLists.txt file, add the option to generate the following executable file.

```
add_executable(topic_subscriber src/topic_subscriber.cpp)
```

- That is, build a file named topic_subscriber.cpp to create an executable called topic_subscriber

```
$ roscd ros_tutorials_topic/src        → Move to the src folder, which is the source folder of the package
$ gedit topic_subscriber.cpp                → New source file and modify contents
```

```
#include "ros/ros.h"                                    // ROS basic header file
#include "ros_tutorials_topic/MsgTutorial.h"      // MsgTutorial  message file header (Auto-generated after build)
// A message callback function, named ros_tutorial_msg below
// This is a function that works when a message is received
// The input message is supposed to receive the MsgTutorial message from the ros_tutorials_topic package
void msgCallback(const ros_tutorials_topic::MsgTutorial::ConstPtr& msg)
{
  ROS_INFO("recieve msg = %d", msg->stamp.sec);          // Disaplay stamp.sec message
  ROS_INFO("recieve msg = %d", msg->stamp.nsec);        // Disaplay stamp.nsec message
  ROS_INFO("recieve msg = %d", msg->data);                // Disaplay data message
}
```

Topic / Publisher / Subscriber

```
int main(int argc, char **argv)                    // Node main function
{
 ros::init(argc, argv, "topic_subscriber");        // Initialize the node name

 ros::NodeHandle nh;                               // Declare a node handle to communicate with the ROS system

  // Subscriber declaration, using MsgTutorial message file from ros_tutorials_topic package
  // Create subscriber ros_tutorial_sub. The topic name is "ros_tutorial_msg"
  // Set the subscriber queue size to 100
 ros::Subscriber ros_tutorial_sub = nh.subscribe("ros_tutorial_msg", 100, msgCallback);

 // callback function will be waiting for message to be received,
  // Execute callback function when message is received
 ros::spin();

 return 0;
}
```

Topic / Publisher / Subscriber

7) Build the ROS node

- Build the message file, the publisher node, and the subscriber node in the ros_tutorials_topic package with the following command:

```
$ cd ~/catkin_ws          → Move to catkin folder
$ catkin_make             → Execute catkin build
```

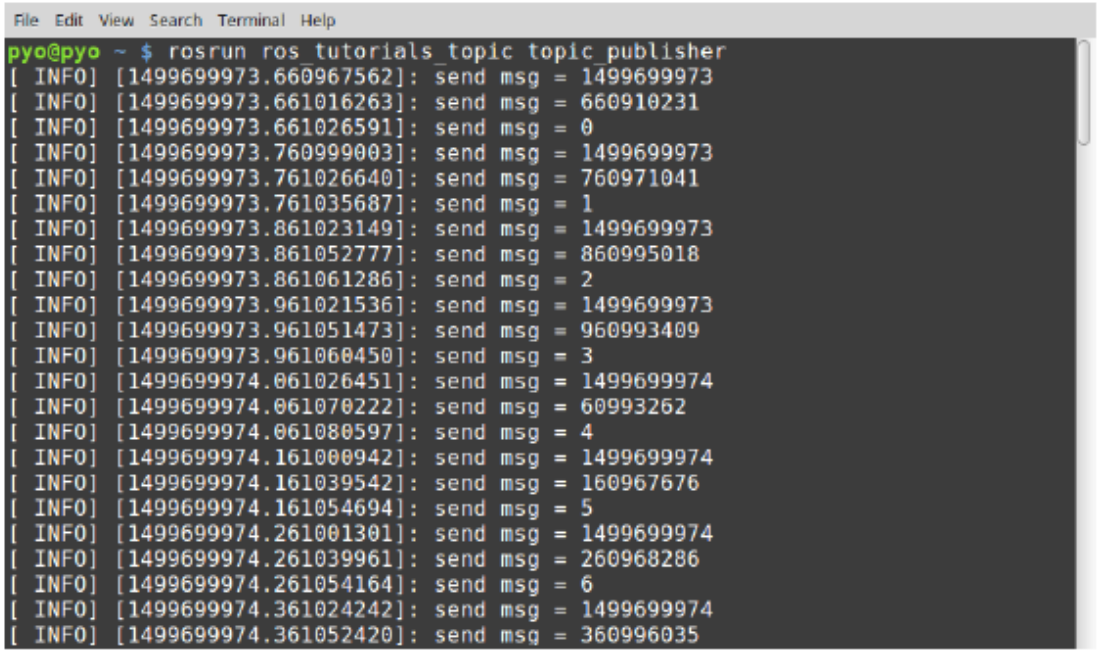
- [Reference] File system
  - Source code file for ros_tutorials_topic package: ~/catkin_ws/src/ros_tutorials_topic/src
  - **Message file** for ros_tutorials_topic package: ~/catkin_ws/src/ros_tutorials_topic/msg
  - Built files are located in the /build and /devel folders in /catkin_ws
    - /build folder saves the settings used in the catkin build
    - /devel/lib/ros_tutorials_topic folder saves **executable file**
    - /devel/include/ros_tutorials_topic folder saves **message header file** that is automatically generated from message file

Topic / Publisher / Subscriber

8) Execute Publisher [Note: Do not forget to run roscore before running the node.]

- Below command runs the topic_publisher node in the ros_tutorials_topic package

```
$ rosrun ros_tutorials_topic topic_publisher
```



Topic / Publisher / Subscriber

- [Reference] rostopic
  - You can use rostopic command to check the topic list, cycle, data bandwidth, and content of the current ROS network.

```
$ rostopic list
/ros_tutorial_msg
/rosout
/rosout_agg
```

```
$ rostopic echo /ros_tutorial_msg
```

```
File  Edit  View  Search  Terminal  Help
pyo@pyo ~ $ rostopic echo /ros_tutorial_msg
stamp:
  secs: 1499700351
  nsecs: 684514825
data: 1713
---
stamp:
  secs: 1499700351
  nsecs: 784542724
data: 1714
---
stamp:
  secs: 1499700351
  nsecs: 884544453
data: 1715
---
stamp:
  secs: 1499700351
  nsecs: 984543934
data: 1716
---
stamp:
  secs: 1499700352
  nsecs:  84543178
```

Topic / Publisher / Subscriber

9) Execute Subscriber

- Below command runs the topic_subscriber node of the ros_tutorials_topic package

```
$ rosrun ros_tutorials_topic topic_subscriber
```

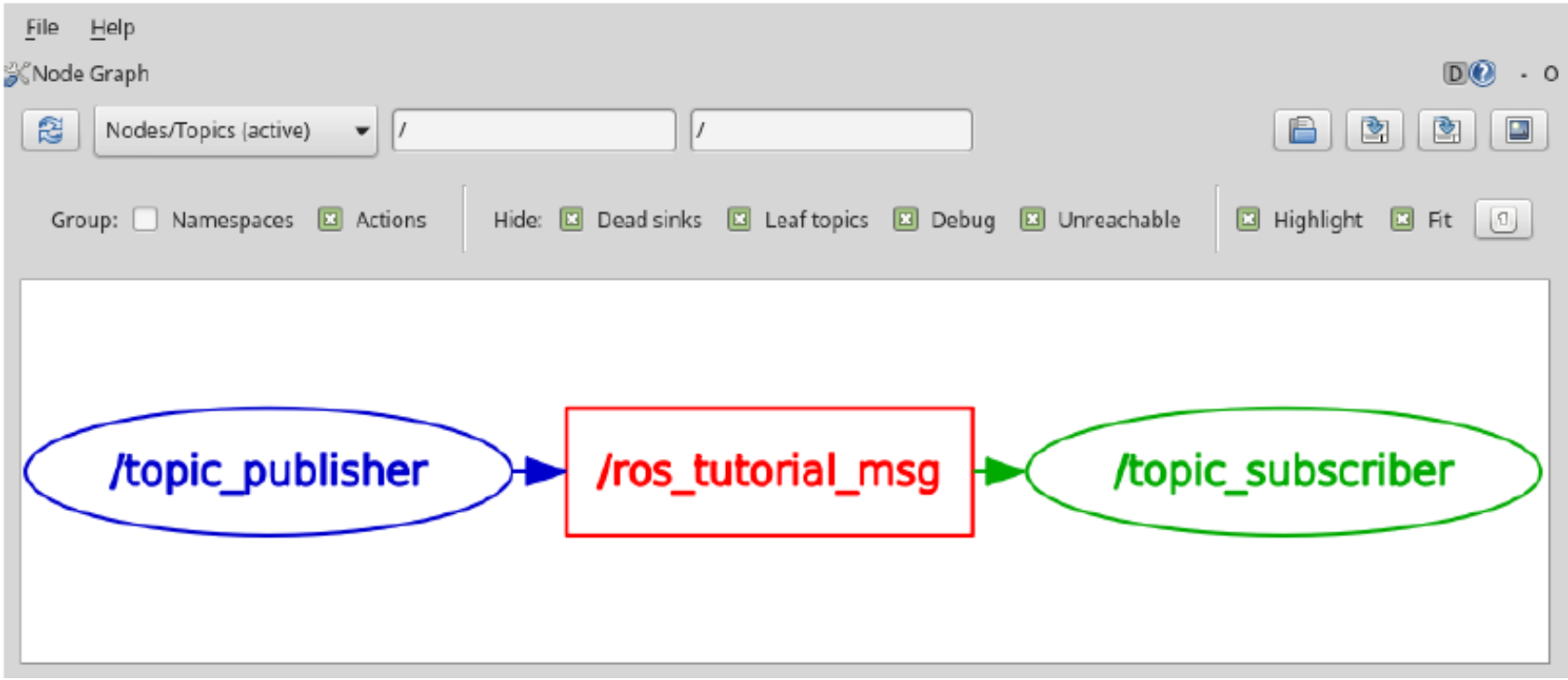
```
File  Edit  View  Search  Terminal  Help
pyo@pyo ~ $ rosrun ros_tutorials_topic topic_subscriber
[ INFO] [1499700485.184875537]: recieve msg = 1499700485
[ INFO] [1499700485.184946471]: recieve msg = 184567102
[ INFO] [1499700485.184957742]: recieve msg = 3048
[ INFO] [1499700485.284812298]: recieve msg = 1499700485
[ INFO] [1499700485.284836776]: recieve msg = 284574255
[ INFO] [1499700485.284844492]: recieve msg = 3049
[ INFO] [1499700485.384811804]: recieve msg = 1499700485
[ INFO] [1499700485.384839629]: recieve msg = 384569171
[ INFO] [1499700485.384849957]: recieve msg = 3050
[ INFO] [1499700485.484795619]: recieve msg = 1499700485
[ INFO] [1499700485.484824179]: recieve msg = 484569717
[ INFO] [1499700485.484838747]: recieve msg = 3051
[ INFO] [1499700485.584792760]: recieve msg = 1499700485
[ INFO] [1499700485.584820628]: recieve msg = 584569677
[ INFO] [1499700485.584830560]: recieve msg = 3052
[ INFO] [1499700485.684824324]: recieve msg = 1499700485
[ INFO] [1499700485.684852121]: recieve msg = 684581217
[ INFO] [1499700485.684861556]: recieve msg = 3053
[ INFO] [1499700485.785495346]: recieve msg = 1499700485
[ INFO] [1499700485.785527583]: recieve msg = 785156898
[ INFO] [1499700485.785552403]: recieve msg = 3054
[ INFO] [1499700485.884855517]: recieve msg = 1499700485
[ INFO] [1499700485.884885781]: recieve msg = 884544763
```

Topic / Publisher / Subscriber

10) Checking communication status of executed nodes

```
$ rqt_graph
```

```
$ rqt  [Plugins] → [Introspection] → [Node Graph]
```



Source Code

- We have created a publisher and a subscriber node that use topic, and executed them to learn how to communicate between nodes. The relevant sources can be found at the github address below.

- https://github.com/ROBOTIS-GIT/ros_tutorials/tree/master/ros_tutorials_topic
- If you want to apply it right away, you can clone the source code with the following command in the catkin_ws/src folder and build it. Then run the topic_publisher and topic_subscriber nodes.

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ROBOTIS-GIT/ros_tutorials.git
$ cd ~/catkin_ws
$ catkin_make
```

```
$ rosrun ros_tutorials_topic topic_publisher
```

```
$ rosrun ros_tutorials_topic topic_subscriber
```

Reference

✓ Download link

✓ Language:

English, Chinese, Japanese, Korean

"ROS Robot Programming"

A Handbook is written by TurtleBot3 Developers

Reference

- R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza. Introduction to Autonomous Mobile Robots. MIT Press, 2nd Edition, 2011, ISBN-10: 0262015358.
- Y. Pyo, H. Cho, R. Jung, and T. Lim, ROS Robot Programming, ROBOTIS Co., Ltd., 2017, ISBN 979-11-962307-1-5
- J. O'Kane, A Gentle Introduction to ROS, CreateSpace Independent Publishing Platform, 2013, ISBN-13: 978-1492143239