ROS2 Tutorials

Advisor: Li-Chen Fu

Date: 2022/11/14

Install ROS2 [1] (1/3)

- Add the ROS 2 apt repositories to your system

```
sudo apt update && sudo apt install curl gnupg2 lsb-release
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg
```

- Add the repository to your sources list

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

[1] https://docs.ros.org/en/dashing/Installation/Ubuntu-Install-Debians.html

Install ROS2 (2/3)

- Install ROS 2 packages

```
sudo apt update                                  # Update your apt repository caches
sudo apt install ros-dashing-desktop
source /opt/ros/dashing/setup.bash               # Sourcing the setup script
```

- Environment setup

```
sudo apt install -y python3-pip
pip3 install -U argcomplete
```

Install ROS2 (3/3)

- Configuring your ROS 2 environment

```
source /opt/ros/dashing/setup.bash
echo "source /opt/ros/dashing/setup.bash" >> ~/.bashrc
```

- Check environment variables

```
printenv | grep -i ROS
```

Create a Workspace [2]

- Source ROS2 environment

```
source /opt/ros/dashing/setup.bash
```

- Create a new directory

```
mkdir -p ~/colcon_ws/src
cd ~/colcon_ws/src
```

- Clone a sample repo

```
git clone https://github.com/ros/ros_tutorials.git -b dashing-devel
```

- 5 Build the workspace with colcon

```
cd .. & colcon build
```

[2] https://docs.ros.org/en/dashing/Tutorials/Workspace/Creating-A-Workspace.html#id8

Create a Package with python

- Make sure you are in the src folder

```
cd ~/colcon_ws/src
```

- Creating a new package

```
ros2 pkg create --build-type ament_python my_package
```

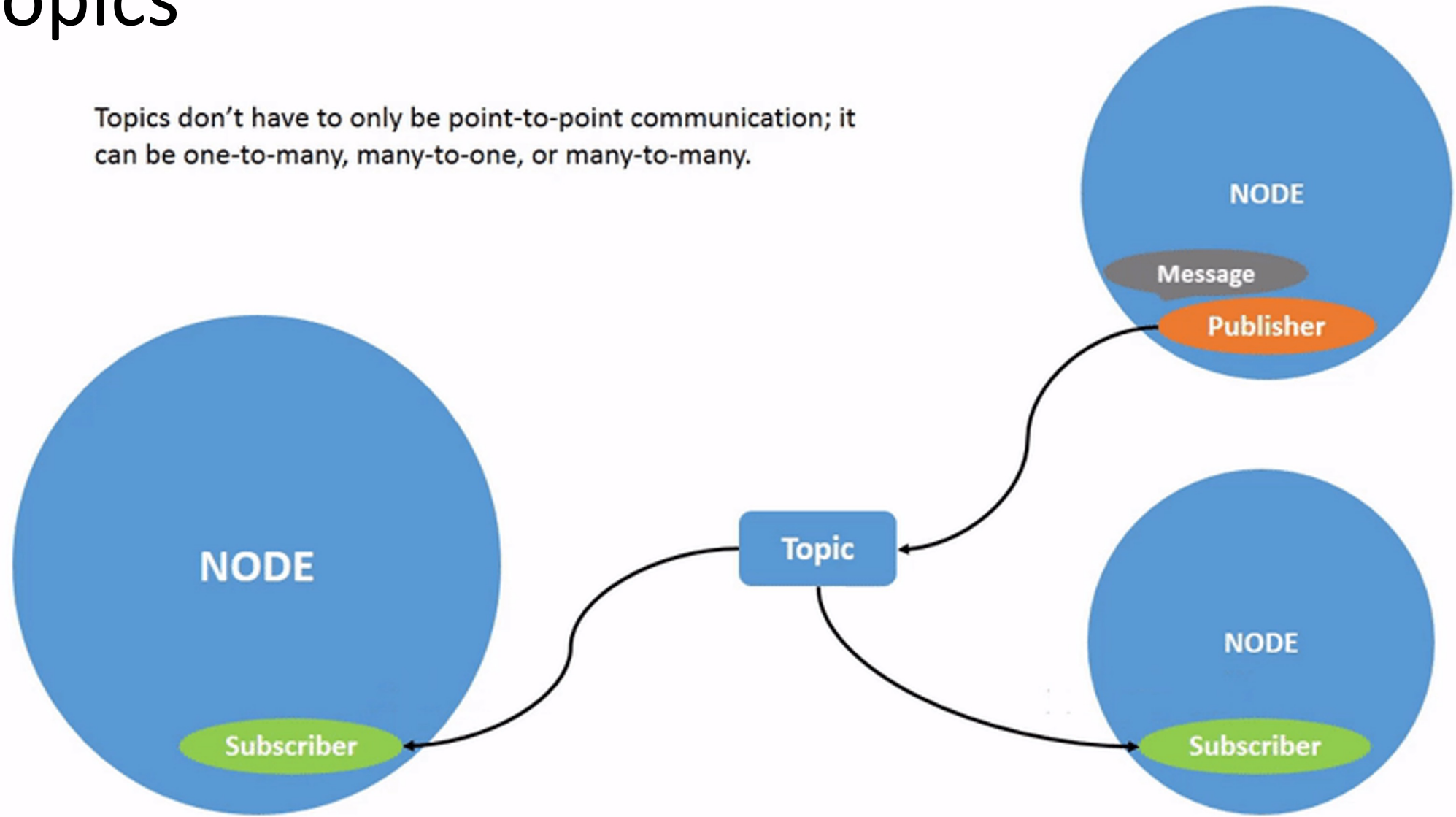
- Build your packages

```
cd  ~/colcon_ws
colcon build
```

```
my_package/
      setup.py
      package.xml
      resource/my_package
```

Topics

Topics don't have to only be point-to-point communication; it can be one-to-many, many-to-one, or many-to-many.



A Simple Publisher and Subscriber (1/4)

- Make sure you are in the package folder

```
cd ~/colcon_ws/src/my_package/my_package
```

- Download the Publisher example

```
wget https://raw.githubusercontent.com/ros2/examples/dashing/rclpy/topics/minimal_publisher/examples_rclpy_minimal_publisher/publisher_member_function.py
```

- Download the Subscriber example

```
wget https://raw.githubusercontent.com/ros2/examples/dashing/rclpy/topics/minimal_subscriber/examples_rclpy_minimal_subscriber/subscriber_member_function.py
```

A Simple Publisher and Subscriber (2/4)

- Add dependencies in colcon_ws/src/my_package/package.xml

```
<exec_depend>rclpy</exec_depend>
<exec_depend>std_msgs</exec_depend>
```

- Add the following line within the console_scripts brackets of the entry_points field in colcon_ws/src/my_package/setup.py

```
entry_points={
        'console_scripts': [
                'talker = py_package.publisher_member_function:main',
                'listener = py_package.subscriber_member_function:main',
        ],
},
```

A Simple Publisher and Subscriber (3/4)

- Check for missing dependencies before building

```
cd  ~/colcon_ws/src
rosdep install -i --from-path src --rosdistro <distro> -y
```

- Build the package

```
colcon build
```

- Source the setup files:

```
. install/setup.bash
```

- Run the talker node

```
ros2 run my_package talker
```

A Simple Publisher and Subscriber (3/4)

- Open another terminal and Make sure you are in the package folder

```
cd  ~/colcon_ws/src
```

- Source the setup files:

```
. install/setup.bash
```

- Run the listener node

```
ros2 run my_package listener
```

```
[INFO] [minimal_publisher]: Publishing: "Hello World: 0"
[INFO] [minimal_publisher]: Publishing: "Hello World: 1"
[INFO] [minimal_publisher]: Publishing: "Hello World: 2"
[INFO] [minimal_publisher]: Publishing: "Hello World: 3"
[INFO] [minimal_publisher]: Publishing: "Hello World: 4"
...
```

```
[INFO] [minimal_subscriber]: I heard: "Hello World: 10"
[INFO] [minimal_subscriber]: I heard: "Hello World: 11"
[INFO] [minimal_subscriber]: I heard: "Hello World: 12"
[INFO] [minimal_subscriber]: I heard: "Hello World: 13"
[INFO] [minimal_subscriber]: I heard: "Hello World: 14"
```

Publisher (talker)

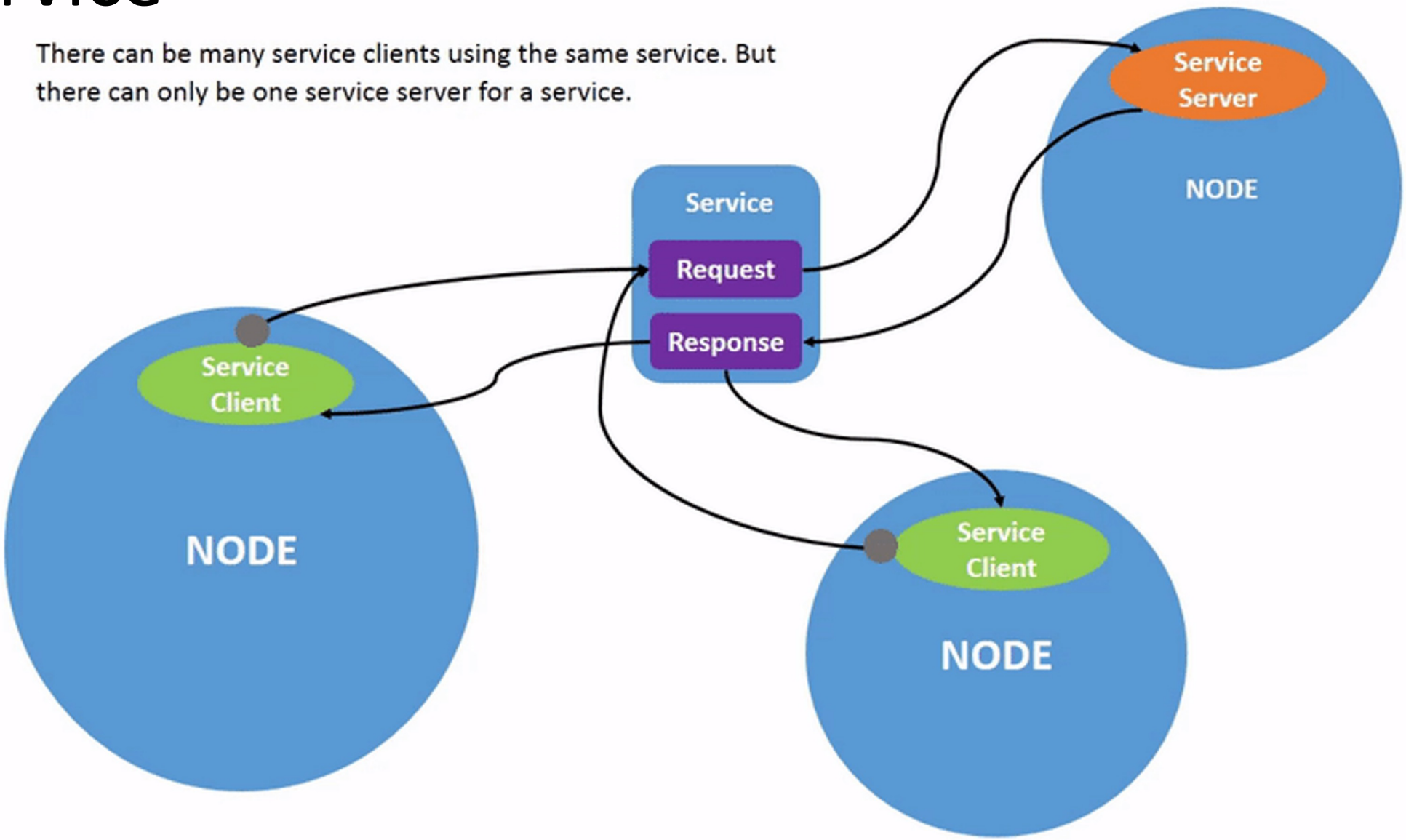
```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class MinimalPublisher(Node):
    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 0.5  # seconds
        self.timer = self.create_timer(timer_period,
self.timer_callback)
        self.i = 0
    def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1

def main(args=None):
    rclpy.init(args=args)
    minimal_publisher = MinimalPublisher()
    rclpy.spin(minimal_publisher)
    minimal_publisher.destroy_node() # Destroy the node explicitly
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Subscriber (listener)

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription(
        String, 'topic', self.listener_callback, 10)
        self.subscription  # prevent unused variable warning

    def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)

def main(args=None):

    rclpy.init(args=args)
    minimal_subscriber = MinimalSubscriber()

    rclpy.spin(minimal_subscriber)

    minimal_subscriber.destroy_node() # Destroy the node explicitly
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Service

There can be many service clients using the same service. But there can only be one service server for a service.



A Simple Server and Client (1/4)

- Make sure you are in the package folder

```
cd ~/colcon_ws/src/my_package/my_package
```

- Create a Service example[3]

```
code service_member_function.py
```

- Make it executable

```
sudo chmod +x service_member_function.py
```

- Create a client example[3]

```
code client_member_function.py
```

- Make it executable

```
sudo chmod +x client_member_function.py
```

[3] https://docs.ros.org/en/dashing/Tutorials/Writing-A-Simple-Py-Service-And-Client.html

A Simple Server and Client (2/4)

- Add dependencies in colcon_ws/src/my_package/package.xml

```
<exec_depend>rclpy</exec_depend>
<exec_depend>example_interfaces</exec_depend>
```

- Add the following line within the console_scripts brackets of the entry_points field in colcon_ws/src/my_package/setup.py

```
entry_points={
        'console_scripts': [
                'service = my_package.service_member_function:main',
                'client = my_package.client_member_function:main',
        ],
},
```

A Simple Server and Client (3/4)

- Build the package

```
cd  ~/colcon_ws & colcon build
```

- Source the setup files

```
. install/setup.bash
```

- Run the service node

```
ros2 run my_package service
```

A Simple Server and Client (4/4)

- Open another terminal and Make sure you are in the package folder

```
cd  ~/colcon_ws/src
```

- Source the setup files

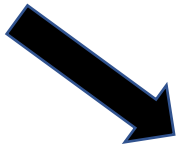
```
. install/setup.bash
```

- Run the client node

```
ros2 run my_package client 2 3
```

Client show

```
[INFO] [minimal_client_async]: Result of add_two_ints: for 2 + 3 = 5
```



Server show

```
[INFO] [minimal_service]: Incoming request
a: 2 b: 3
```

Reference

- ROS2 Documentation
  https://docs.ros.org/en/dashing/
- ROS2 example (dashing)
  https://github.com/ros2/examples/tree/dashing
- rclpy Navigation
  https://docs.ros2.org/foxy/api/rclpy/api/node.html