

### 3. ROS2 topic communication

Topic communication is the most frequently used communication mode of ROS2. Topic communication is based on the publish-subscription mode, in which the publisher publishes data of a specified topic, and the subscriber can receive the data as long as he subscribes to the data of the topic. Next, I will explain how to use the Python language to implement topic communication between nodes.

#### 1. Create a new workspace

Terminal input,

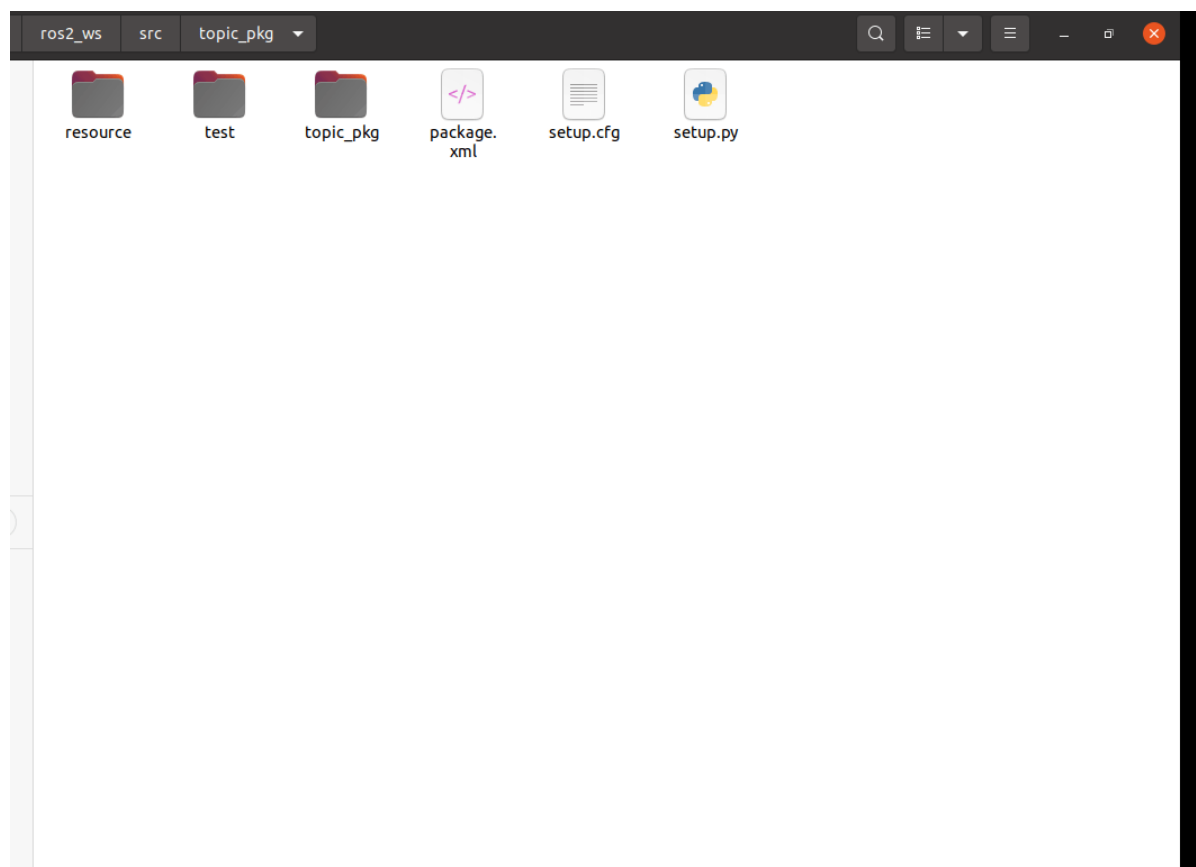
```
mkdir -p ros2_ws/src
cd ros2_ws/src
```

ros2\_ws is the name of the workspace and src is the directory where the feature packs are stored.

#### 2. Create a feature pack

Terminal input,

```
cd ~/ros2_ws/src
ros2 pkg create --build-type ament_python topic_pkg
```



Switch to the topic\_pkg folder, there are the following files and folders, we write python programs in this directory under the topic\_pkg folder, that is,

```
~/ros2_ws/src/topic_pkg/topic_pkg
```

## 3. Write the publisher python file

### 3.1 Program source code

Create a new file called publisher\_demo.py,

```
cd ~/ros2_ws/src/topic_pkg/topic_pkg
gedit publisher_demo.py
```

Copy the following sections into the file,

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String
class Topic_Pub(Node):
    def __init__(self,name):
        super().__init__(name)
        self.pub = self.create_publisher(String,"/topic_demo",1)
        self.timer = self.create_timer(1,self.pub_msg)

    def pub_msg(self):
        msg = String()
        msg.data = "Hi,I send a message."
        self.pub.publish(msg)

def main():
    rclpy.init()
    pub_demo = Topic_Pub("publisher_node")
    rclpy.spin(pub_demo)
```

Here is the idea of modularity to achieve, this method is conducive to our migration, modify the code. First, we define a class named Topic\_Pub, which has two parts, one is init and the other is pub\_msg. init initializes the class itself, and pub\_msg is the method of the class, that is, once we create the object of the class, we can use the variables and methods inside.

```
#导入rclpy库
# Import the rclpy library
import rclpy
from rclpy.node import Node
#导入String字符串消息
# Import String message
from std_msgs.msg import String
#创建一个继承于Node基类的Topic_Pub节点子类 传入一个参数name
# Create a Topic_Pub Node subclass inherited from the node base class with a
parameter name
def __init__(self,name):
    super().__init__(name)
    #创建一个发布者,使用create_publisher的函数,传入的参数分别是:
    # Create a publisher, using the create_publisher function, passing in
the following parameters:
    #话题数据类型、话题名称、保存消息的队列长度
    # Topic data type, topic name, queue length to save messages
    self.pub = self.create_publisher(String,"/topic_demo",1)
```

```

        #创建一个定时器，间隔1s进入中断处理函数，传入的参数分别是：
        # Create a timer, interval 1s to enter the interrupt handler, the
parameters are:
        #中断函数执行的间隔时间，中断处理函数
        # Interrupt function execution interval, interrupt handler function
        self.timer = self.create_timer(1,self.pub_msg)
#定义中断处理函数
# Define interrupt handlers
def pub_msg(self):
    msg = String() #创建一个String类型的变量msg
    # Creates a variable msg of type String
    msg.data = "Hi,I send a message." #给msg里边的data赋值
    # Assign a value to the data inside msg
    self.pub.publish(msg) #发布话题数据
    # Publish the topic data

#主函数
# Main function
def main():
    rclpy.init() #初始化
    # initialization
    pub_demo = Topic_Pub("publisher_node") #创建Topic_Pub类对象，传入的参数就是节点的名字
    # Creates an object of the Topic_Pub class, passing in the name of the node
    rclpy.spin(pub_demo) #执行rclpy.spin函数，里边传入一个参数，参数是刚才创建好的
    Topic_Pub类对象
    # Executes the rclpy.spin function, passing in an argument to the Topic_Pub
    class object you just created

```

## 3.2. Modify the setup.py file

Terminal input,

```

cd ~/ros2_ws/src/topic_pkg
gedit setup.py

```

Find the location shown below,

```
1 from setuptools import setup
2
3 package_name = 'topic_pkg'
4
5 setup(
6     name=package_name,
7     version='0.0.0',
8     packages=[package_name],
9     data_files=[
10         ('share/ament_index/resource_index/packages',
11          ['resource/' + package_name]),
12         ('share/' + package_name, ['package.xml']),
13     ],
14     install_requires=['setuptools'],
15     zip_safe=True,
16     maintainer='yahboom',
17     maintainer_email='yahboom@todo.todo',
18     description='TODO: Package description',
19     license='TODO: License declaration',
20     tests_require=['pytest'],
21     entry_points={
22         'console_scripts': [
23             'publisher_demo = topic_pkg.publisher_demo:main'
24         ],
25     },
26 )
```

In 'console\_scripts': [] add the following in the format,

```
console_scripts = [
    'Mcnamu_driver = yahboomcar_bringup.Mcnamu_driver:main',
]
```

Executable file name                      Feature pack name                      The name of the Python file you wrote

The name of the executable file is our custom, that is, which program we need to execute when ros2 run, the name of the function package is the function package where your python file is located, followed by the name of the python file, and the final main is the program execution entrance. As long as the program is written in python format, compiled and generated executable files, they need to be added here in accordance with the above format.

### 3.3. Compile workspace

```
cd ~/ros2_ws
colcon build
```

Compile successfully, enter the following command to add the workspace to the system environment variable,

```
echo "source ~/ros2_ws/install/setup.bash" >> ~/.bashrc
```

Then, reopen the terminal or refresh the environment variable to take effect,

```
source ~/.bashrc
```

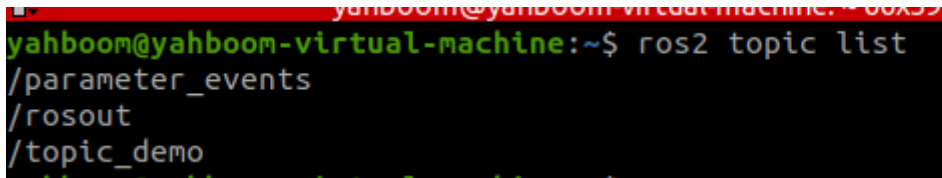
### 3.4. Run the program

Terminal input,

```
ros2 run topic_pkg publisher_demo
```

After the program is successfully run, nothing is printed. We can check the data through the ros2 topic tool. First, check whether there is a topic published in this program.

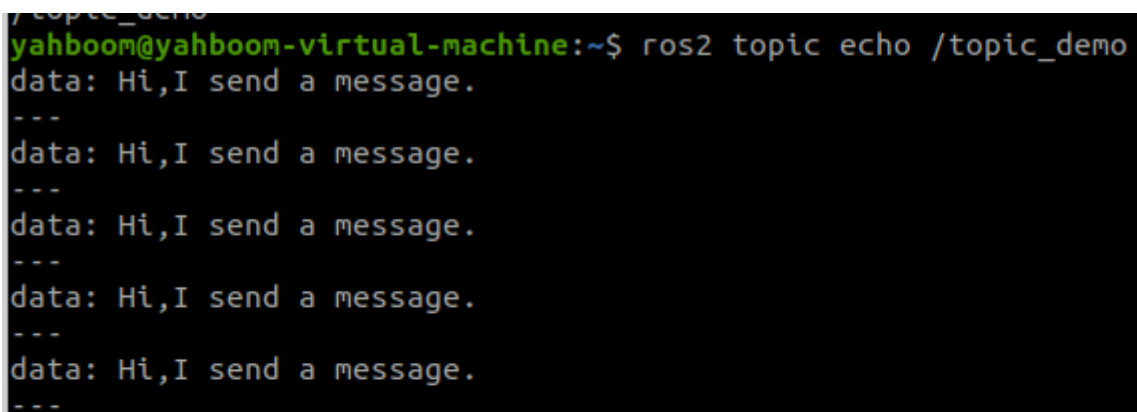
```
ros2 topic list
```



```
yahboom@yahboom-virtual-machine:~$ ros2 topic list
/parameter_events
/rosout
/topic_demo
```

This topic\_demo is the topic data defined in the program, next, we use ros2 topic echo to print this data, terminal input,

```
ros2 topic echo /topic_demo
```



```
/topic_demo
yahboom@yahboom-virtual-machine:~$ ros2 topic echo /topic_demo
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
```

As can be seen, the "Hi,I send a message." printed by the terminal is the same as the msg.data = "Hi,I send a message." in our code.

## 4. Write the subscriber python file

### 4.1. Program source code

Create a new file called subscriber\_demo.py,

```
cd ~/ros2_ws/src/topic_pkg/topic_pkg
gedit subscriber_demo.py
```

Copy the following sections into the file,

```
#导入相关的库
# import the relevant library
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class Topic_Sub(Node):
    def __init__(self,name):
        super().__init__(name)
        #创建订阅者使用的是create_subscription，传入的参数分别是：话题数据类型，话题名称，
        回调函数名称，队列长度
        # create_subscription is used to create a subscriber, passing in the
        following parameters: Topic data type, topic name, callback function name, Queue
        length
        self.sub =
self.create_subscription(String,"/topic_demo",self.sub_callback,1)
        #回调函数执行程序：打印接收到的信息
        # Callback function execution program: print the received information
    def sub_callback(self,msg):
        print(msg.data)
def main():
    rclpy.init() #ROS2 Python接口初始化
    #ROS2 Python interface initialization
    sub_demo = Topic_Sub("subscriber_node") # 创建对象并进行初始化
    # Create the object and initialize it
    rclpy.spin(sub_demo)
```

### 4.2. Modify the setup.py file

Terminal input,

```
cd ~/ros2_ws/src/topic_pkg
gedit setup.py
```

Find the location shown below,

```

1 from setuptools import setup
2
3 package_name = 'topic_pkg'
4
5 setup(
6     name=package_name,
7     version='0.0.0',
8     packages=[package_name],
9     data_files=[
10         ('share/ament_index/resource_index/packages',
11          ['resource/' + package_name]),
12         ('share/' + package_name, ['package.xml']),
13     ],
14     install_requires=['setuptools'],
15     zip_safe=True,
16     maintainer='yahboom',
17     maintainer_email='yahboom@todo.todo',
18     description='TODO: Package description',
19     license='TODO: License declaration',
20     tests_require=['pytest'],
21     entry_points={
22         'console_scripts': [
23             'publisher_demo = topic_pkg.publisher_demo:main',
24             'subscriber_demo = topic_pkg.subscriber_demo:main'
25         ],
26     },
27 )

```

In 'console\_scripts': [] add the following in the format,

```
'subscriber_demo = topic_pkg.subscriber_demo:main'
```

```
'subscriber_demo = topic_pkg.subscriber_demo:main'
```

### 4.3. Compile workspace

```
cd ~/ros2_ws
colcon build
```

After compiling, refresh the environment variables in the workspace,

```
source ~/ros2_ws/install/setup.bash
```

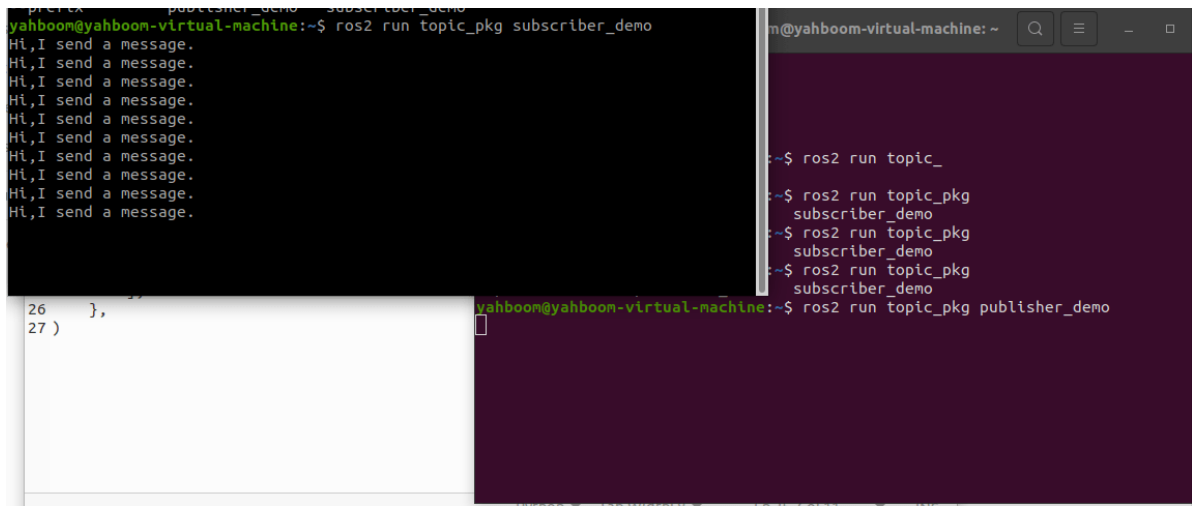
### 4.4. Run the program

Terminal input,

```

#启动发布者节点
# Start the publisher node
ros2 run topic_pkg publisher_demo
#启动订阅者节点
# Start the subscriber node
ros2 run topic_pkg subscriber_demo

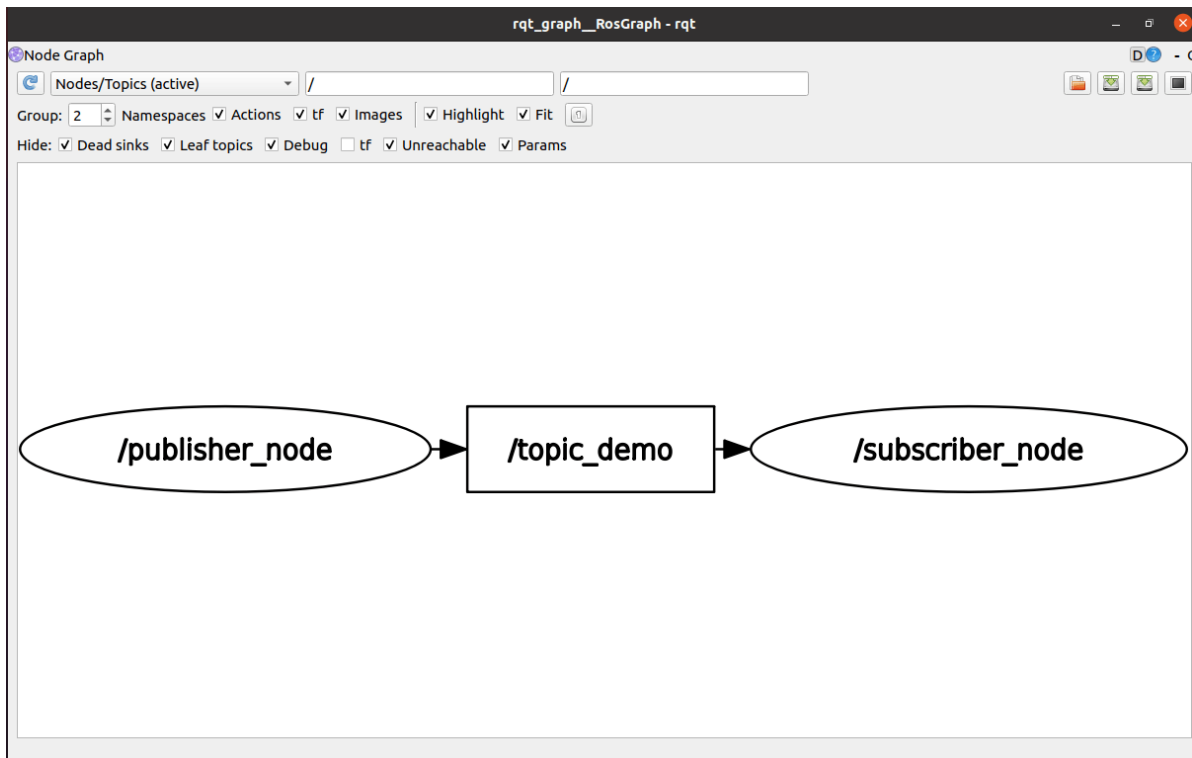
```



As shown in the figure above, the terminal running the subscriber point prints the information for /topic\_demo published by the publisher.

You can view the topic communication between two nodes with the following command,

```
ros2 run rqt_graph rqt_graph
```



## 5. Conclusion

### 5.1 python programming topic communication program framework



1、导入库文件 **Import library file**

```
import rclpy
from rclpy.node import Node
```

2、创建类 **Create class**

```
class ClassName(Node):
    def __init__(self, name):
        super().__init__(name)
        #create subscriber
        self.sub =
self.create_subscription(Msg_Type, Topic_Name, self.CallbackFunction, Msg_Line_Size
)
```

```
        #create publisher
        self.pub = self.create_publisher(Msg_Type, Topic_Name, Msg_Line_Size)
```

```
        #create timer
        self.timer = self.create_timer(Timer, self.TimerExcuteFunction)
```

```
    def CallbackFunction(self):
```

```
        ...
```

```
    def TimerExcuteFunction(self):
```

```
        ...
```

3、主函数main: 主要是初始化节点，创建对象 **The main function: This initializes nodes and creates objects**

```
def main():
    rclpy.init()
    class_object = ClassName("node_name")
    rclpy.spin(class_object)
```

## 5.2. Modify the setup.py file

Use python to write node program, need to modify the setup.py file, refer to the above content, add to generate their own node program.