Practical Work 2: ROS 2 Beginner Level

IA712: Mobile Robotics

Zhi Yan

1 Objective

The goal of this session is to gain hands-on experience with the fundamental concepts of ROS 2: nodes, topics, and services. We will use the `turtlesim` simulator to visualize these concepts and practice using core command-line tools.

2 Prerequisites

- A working ROS 2 Humble installation (from Practical Work 1).
- A Linux terminal. We will need at least two terminals open simultaneously.

3 Exploring the ROS 2 Graph with Turtlesim

3.1 Launch Turtlesim

As we saw last time, we need two nodes to get started. **In Terminal 1**, start the simulator itself:

```
ros2 run turtlesim turtlesim_node
```

A simulator window with a turtle should appear. This is a ROS 2 node that we just launched.

In Terminal 2, start the keyboard teleoperation node:

```
ros2 run turtlesim turtle_teleop_key
```

Now, make sure Terminal 2 is the active window and use the arrow keys to drive the turtle around.

3.2 Introspection with `ros2 node`

Let's see what nodes are currently running. **Open a new terminal (Terminal 3)**. In this new terminal, run:

```
ros2 node list
```

You should see the following output, which lists our two active nodes:

```
/teleop_turtle
/turtlesim
```

3.3 Understanding Topics

The teleop node needs to send velocity commands to the simulator node. It does this using a topic. Let's find it. In Terminal 3, list all active topics:

```
ros2 topic list
```

You will see a list of topics. The most interesting one for us is `/turtle1/cmd_vel`. This is the topic that carries the velocity commands.

We can even "listen in" on the data being published. In Terminal 3, run:

```
ros2 topic echo /turtle1/cmd_vel
```

Now, go back to Terminal 2 and press the arrow keys. You will see messages appearing in Terminal 3 every time you press a key. This shows the data being published by the teleop node. The message type is `geometry_msgs/msg/Twist`. Press `Ctrl+C` in Terminal 3 to stop echoing.

3.4 Publishing your own commands

We can also publish commands manually from the command line. In Terminal 3, run the following command. This will make the turtle drive forward and turn simultaneously. The command structure is `ros2 topic pub <topic_name> <message_type> '<data in YAML format>'`. The `--once` flag means publish one message and exit.

```
ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear:
  {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```

You should see the turtle in the simulator move according to your command.

4 Using Services

Turtlesim also provides services for things that happen once, like resetting the simulation or spawning a new turtle.

4.1 Listing Services

Let's see what services are available. In Terminal 3, run:

```
ros2 service list
```

You will see services like `/clear`, `/reset`, `/spawn`, and others related to parameters.

4.2 Calling a Service

Let's call the `/spawn` service to create a new turtle. The service requires arguments: x, y, theta (orientation), and name. In Terminal 3, run:

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2,
  name: 'turtle2'}"
```

A new turtle named 'turtle2' will appear in the simulator!

Now, if you run `ros2 topic list` again, you will see a new set of topics, like `/turtle2/cmd_vel`, for controlling the new turtle.

Finally, let's call the `/reset` service to clear the screen and return the original turtle to its starting position. This service has no arguments.

```
ros2 service call /reset std_srvs/srv/Empty
```

5 RQT Graph: Visualizing the System

ROS 2 has a wonderful graphical tool called `rqt_graph` for visualizing the computation graph.

In Terminal 3, run:

```
rqt_graph
```

A window will open showing you the nodes and the topics that connect them. It should clearly show that the `/teleop_turtle` node is publishing to the `/turtle1/cmd_vel` topic, and the `/turtlesim` node is subscribed to it.

This concludes the practical session. Close all terminals (`Ctrl+C`) and the simulator window.