Introduction to Mobile Robotics

Practical Work 11: ROS 2 Namespaces and DDS

Zhi Yan

ENSTA, Institut Polytechnique de Paris

1 Objective

The goal of this session is to make the leap from single-robot systems to multi-robot systems (MRS). You will launch **two** TurtleBot3 robots in the same Gazebo simulation environment. Specifically, you will:

1. Configure **ROS 2 Namespaces** effectively to prevent topic conflicts.
2. Experiment with **DDS Domain IDs** to understand network isolation.

2 The Challenge of MRS in ROS 2

When you launch a node, it publishes to global topic names like `/scan` or `/cmd_vel`. If you launch two robots without doing anything special, both will try to write to `/cmd_vel`. As a result, if you send a command to move Robot 1, Robot 2 will also move!

To solve this, we must push each robot into its own **Namespace**:

- Robot 1: `/tb3_1/cmd_vel`, `/tb3_1/scan`, `/tb3_1/base_link`, etc.
- Robot 2: `/tb3_2/cmd_vel`, `/tb3_2/scan`, `/tb3_2/base_link`, etc.

3 Experiment A: Creating a MRS Launch File

Launching multiple robots in Gazebo is slightly complex because we need to spawn them at different coordinates and assign namespaces.

1. Create a new package `multi_robot_pkg` in your ROS 2 workspace.
2. Create a `launch` folder in the package and copy the `turtlebot3_world.launch.py` file that we've launched countless times into this folder (yes, we're finally getting to touch it!).
3. Rename `turtlebot3_world.launch.py` to `multi_robot.launch.py` and edit it.
4. Add the following two lines of code to the header of the file:

```
from launch.actions import GroupAction
from launch_ros.actions import PushRosNamespace
```

5. Locate the two sections defining `robot_state_publisher_cmd` and `spawn_turtlebot_cmd` in the file and replace them with the code block below. This code does three things:
   - Defines two robots.

- Assigns a separate namespace to each robot (`tb3_1` and `tb3_2`).
- Sets different coordinates (`y_pose` offset by 1 m).

```
    # Define the position of the second robot (the first robot uses
    the default x_pose, y_pose)
    x_pose_2 = LaunchConfiguration('x_pose_2', default='-2.0')
    y_pose_2 = LaunchConfiguration('y_pose_2', default='0.5')

    # Robot 1 (Namespace: tb3_1)
    spawn_robot1 = GroupAction([
        PushRosNamespace('tb3_1'),
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                os.path.join(launch_file_dir, 'robot_state_publisher.
    launch.py')
            ),
            launch_arguments={'use_sim_time': use_sim_time}.items()
        ),
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                os.path.join(launch_file_dir, 'spawn_turtlebot3.launch
    .py')
            ),
            launch_arguments={
                'x_pose': x_pose,
                'y_pose': y_pose,
                'entity_name': 'tb3_1' # Pass unique entity name
            }.items()
        ),
    ])

    # Robot 2 (Namespace: tb3_2)
    spawn_robot2 = GroupAction([
        PushRosNamespace('tb3_2'),
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                os.path.join(launch_file_dir, 'robot_state_publisher.
    launch.py')
            ),
            launch_arguments={'use_sim_time': use_sim_time}.items()
        ),
        IncludeLaunchDescription(
            PythonLaunchDescriptionSource(
                os.path.join(launch_file_dir, 'spawn_turtlebot3.launch
    .py')
            ),
            launch_arguments={
                'x_pose': x_pose_2,
                'y_pose': y_pose_2,
                'entity_name': 'tb3_2' # Pass unique entity name
            }.items()
        ),
    ])
```

6. Replace:

```
    ld.add_action(spawn_turtlebot_cmd)
    ld.add_action(robot_state_publisher_cmd)
```

by:

```
    ld.add_action(spawn_robot1)
    ld.add_action(spawn_robot2)
```

7. Finally, we need to hack `spawn_turtlebot3.launch.py`. Let's make a backup before editing it:

```
sudo cp /opt/ros/$ROS_DISTRO/share/turtlebot3_gazebo/launch/
   spawn_turtlebot3.launch.py /opt/ros/$ROS_DISTRO/share/
   turtlebot3_gazebo/launch/spawn_turtlebot3.launch.py.org
```

8. Open `spawn_turtlebot3.launch.py` and change `TURTLEBOT3_MODEL` to `LaunchConfiguration('entity_name', default=TURTLEBOT3_MODEL)` in `start_gazebo_ros_spawner_cmd`. **Explain the reason for doing so in your report.**

9. Launch your `multi_robot.launch.py`, you should now see two TurtleBot3 side by side in Gazebo.

Now, if you check the topics by `ros2 topic list`, you should see something like this:

```
/tb3_1/clock
/tb3_1/cmd_vel
/tb3_1/imu
/tb3_1/joint_states
/tb3_1/odom
/tb3_1/robot_description
/tb3_1/scan
/tb3_1/tf
/tb3_2/clock
/tb3_2/cmd_vel
/tb3_2/imu
/tb3_2/joint_states
/tb3_2/odom
/tb3_2/robot_description
/tb3_2/scan
/tb3_2/tf
```

Problem: When you try to drive one robot:

```
ros2 run turtlebot3_teleop teleop_keyboard --ros-args -r /cmd_vel:=/tb3_1/
   cmd_vel
```

another robot also moves along with it. This is one of the problems we need to solve during the project phase.

4 Experiment B: The Domain ID (DDS)

If you have two different groups of robots deployed on the same network and you don't want them to interfere with each other, or you want to effectively distinguish between "intra-group communication" and "inter-group communication", or even safely launch two separate Gazebo simulations at the same time, you might want to use `ROS_DOMAIN_ID` to create some kind of "virtual LAN". Let's do a simple exercise to quickly understand how it works.

1. Open Terminal 1. Set the Domain ID to 6. Start a talker.

```
export ROS_DOMAIN_ID=6
ros2 run demo_nodes_cpp talker
```

2. Open Terminal 2. Leave Domain ID as default (0). Start a listener.

```
# Default ID is 0
ros2 run demo_nodes_cpp listener
```

Observation: The listener should not hear anything. They are on different logical networks.

3. Open Terminal 3. Set Domain ID to 6.

```
export ROS_DOMAIN_ID=6
ros2 run demo_nodes_cpp listener
```

Observation: Now you receive data.
Lesson: Ensure all team members use the same Domain ID!

Congratulations! You have gained some hands-on experience with multi-robot systems. You can now complete the previous exercises that you haven't finished if you have some time left.