Faculty 3ME
DELFT UNIVERSITY OF TECHNOLOGY

ROS Exercise 3
2017-2018

Priadi Teguh Wibowo (4625862)

December 15, 2017

1 Intro

This report explains exercise in Chapter 7, 8, and 9: Writing ROS programs from a set of exercises for the book "A Gentle Introduction to ROS".

- The first exercise extends the `vel_filter` node to filter out all messages with an angular velocity above a certain value using parameter. And it also extends the launch file to set the default value for that parameter and test the behaviour of `vel_filter` node by changing the parameter value during runtime.
- The second exercise extends the previous `vel_filter` node with a service server that can enable or disable the filter.
- The final exercise extends the previous launch file by recording the messages that the filter node publish on.

2 Implementation

Exercise 7

In order to make the variable `max_ang_vel` as a private parameter, tilde sign is added to the parameter name. The launch file is also changed to include the parameter inside in `vel_filter` node. And `ros::param::get` method is called in while loop so it can get the parameter value during initialisation and run time. The implementation of getting parameter value is shown in Listing 1.

Listing 1: Accessing private parameter

```
const std::string PARAM_NAME = "~max_ang_vel";
 while(ros::ok())
{
  // Get the maximum velocity parameter
```

```
    if(!ros::param::get(PARAM_NAME,max_ang_vel))
    {
      ROS_FATAL_STREAM("Could not get parameter " <<
          PARAM_NAME);
      exit(1);
    }
    ros::spinOnce();
    // Wait until it's time for another iteration.
    rate.sleep();
  }
```

The callback function is also edited a little bit. The filtering condition is simply edited into `(msg.angular.z < max_ang_vel)` so it will only publishes messages with angular velocity below the `max_ang_vel`. And when the node publishes messages, log messages is added to see the change of node behaviour if the parameter value is changed. The edited callback function is shown in Listing 2.

Listing 2: Callback function in exercise 7

```
void twistMessageFilter(const geometry_msgs::Twist& msg)
{
  static int total_msg = 0;
  if(msg.angular.z < max_ang_vel) {
    // Publish the message.
    pub->publish(msg);
    ROS_INFO_STREAM("filtered messaged, angular = " <<
        msg.angular.z);
  } else {
    total_msg++;
  }
  ROS_INFO_STREAM_THROTTLE(5.0,"Total dropped messages
      vel_filter: " << total_msg);
}
```

Exercise 8

A service server called `toggle_filter` is created to disable or enable the filter. By using `rosservice call toggle_filter` from the command line, then the server service will call the service callback function as shown in Listing 3 to toggle the `enable_filter` flag.

Listing 3: Service callback function in exercise 8

```
bool toggle_filter(std_srvs::Empty::Request &req,
    std_srvs::Empty::Response &resp)
{
  enable_filter = !enable_filter;
  ROS_INFO_STREAM( (enable_filter ? "Enable" : "Disable") << "
      filter");
```

```
  return true;
}
```

If the flag is set to true, then the node will only publish messages with angular velocity below `max_ang_vel`. And if the flag is false, the filter node will publish all messages it receives. The new subscriber callback function is shown in Listing 4.

Listing 4: Subscriber callback function in exercise 8

```
....
 if (enable_filter) {
   if(msg.angular.z < max_ang_vel)
   {
     ....
   }
} else {
   // Publish all received message.
   pub->publish(msg);
}
 ....
```

Exercise 9

The previous launch file in exercise 7 is added with a `rosbag` node to record the messages that the filter node publishes on. This addition is shown in Listing 5.

Listing 5: Launch file in exercise 9 to record messages

```
....
<node
   pkg="rosbag"
   type="record"
   name="bag_record"
   args="-O exercise_9.bag /turtle1/cmd_vel_filtered"
/>
```

3 Result

Exercise 7

The initial value of `max_ang_vel` is set to 0, so the messages published by the filter node will only have negative angular velocity, as shown in figure 1. Then that parameter value is changed to 1 by typing `rosparam set /filter_velocity/max_ang_vel 1` in command line. Afterwards, the filter node publishes both negative and positive angular velocity, as shown in figure 2.
The difference between setting `max_ang_vel` as a private parameter and a global parameter is that as a private parameter the original parameter name is added with its

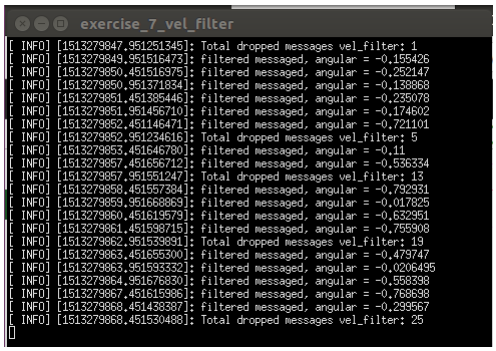

Figure 1: `vel_filter` terminal with enabled filter

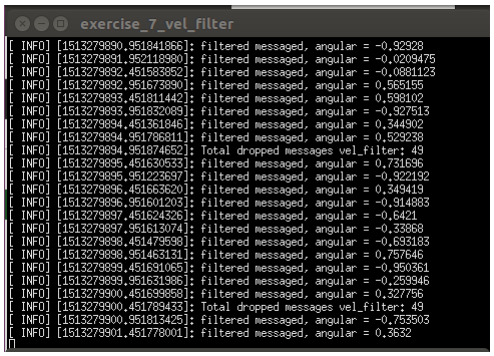

Figure 2: `vel_filter` terminal with disabled filter

node name as its namespace, so it becomes `/filter_velocity/max_ang_vel`, whereas as a global parameter it keeps its original name.

Exercise 8

In order to check if disabling filter is working, two terminals are opened to compare the messages by publish on topic `/turtle1/cmd_vel` and `/turtle1/cmd_vel_filtered`. In each terminal, command `rostopic echo topic-name` is used to see the published messages.
Initially the filter is enabled and `max_ang_vel` is set to 0 so the messages published by `vel_filter` and `pubvel` are different because filter node only publish messages with negative angular velocity, as shown by figure 3. Then after using `rosservice call /toggle_filter` in command line, `vel_filter` publishes messages same as the one published by `pubvel`, as shown by figure 4.

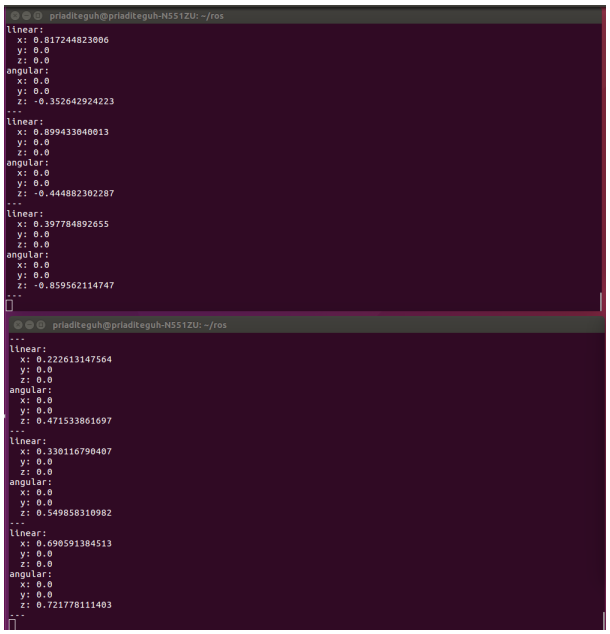

Figure 3: Comparison between `vel_filter` (upper terminal) and `pubvel` (lower terminal) before toggling

Exercise 9

Figure 5 shows the total messages published by `vel_filter` node. After 1 minute, the log message reports that the total messages published by filter node is 60. In order to know the total messages reported by `rosbag`, file `exercise_9.bag` resides in `/.ros`. And by using `rosbag info exercise_9.bag`, the information regarding the total messages can be obtain, which is 61, as shown by figure 6. So the number of messages reported by `rosbag` and the filter node are same.

4 Conclusion

This exercise introduces several fundamental concepts in ROS, which are :

- Parameters in ROS can be accessed to set or get the information from the node. Thus, it can add more flexibility in ROS nodes.
- Services in ROS is alternative method of communication, other than messages. The difference are that services is bi-directional communication and the involved nodes are called client and server.
- The bag files are used to record the messages published on a topic and to replay those recorded messages.

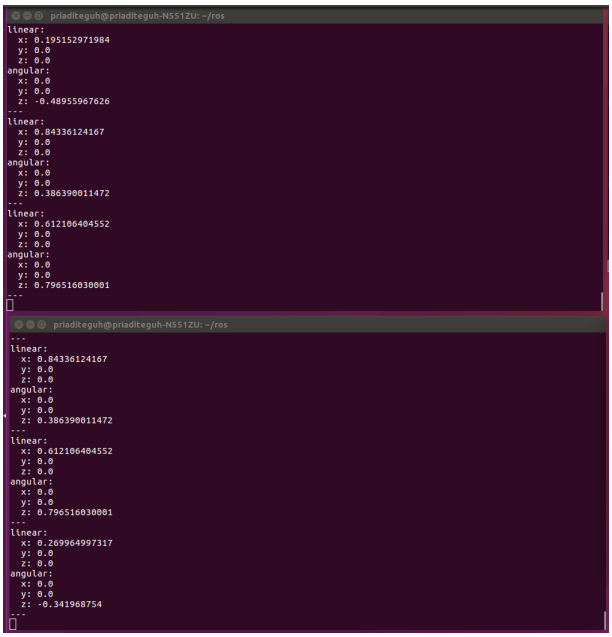

Figure 4: Comparison between `vel_filter` (upper terminal) and `pubvel` (lower terminal) after toggling

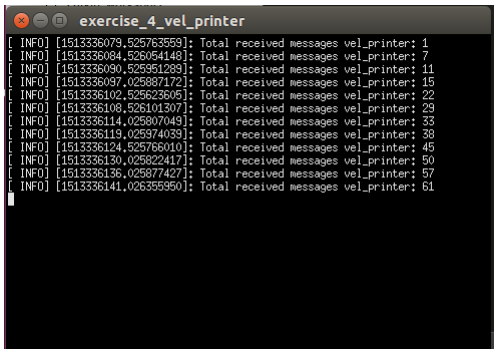

Figure 5: Total messages reported by log messages

```
priaditeguh@priaditeguh-N551ZU:~/.ros$ rosbag info exercise_9.bag
path:        exercise_9.bag
version:     2.0
duration:    1:01s (61s)
start:       Dec 15 2017 12:07:59.53 (1513336079.53)
end:         Dec 15 2017 12:09:01.03 (1513336141.03)
size:        12.2 KB
messages:    61
compression: none [1/1 chunks]
types:       geometry_msgs/Twist [9f195f881246fdfa2798d1d3eebca84a]
topics:      /turtle1/cmd_vel_filtered   61 msgs    : geometry_msgs/Twist
priaditeguh@priaditeguh-N551ZU:~/.ros$
```

Figure 6: Total messages reported by log messages by `rosbag`