Smart Mobility Engineering Lab (IGS3231)

Jump Together, Fly Farther!

School of Global Convergence Studies

인하대학교 국제학부

Week 4

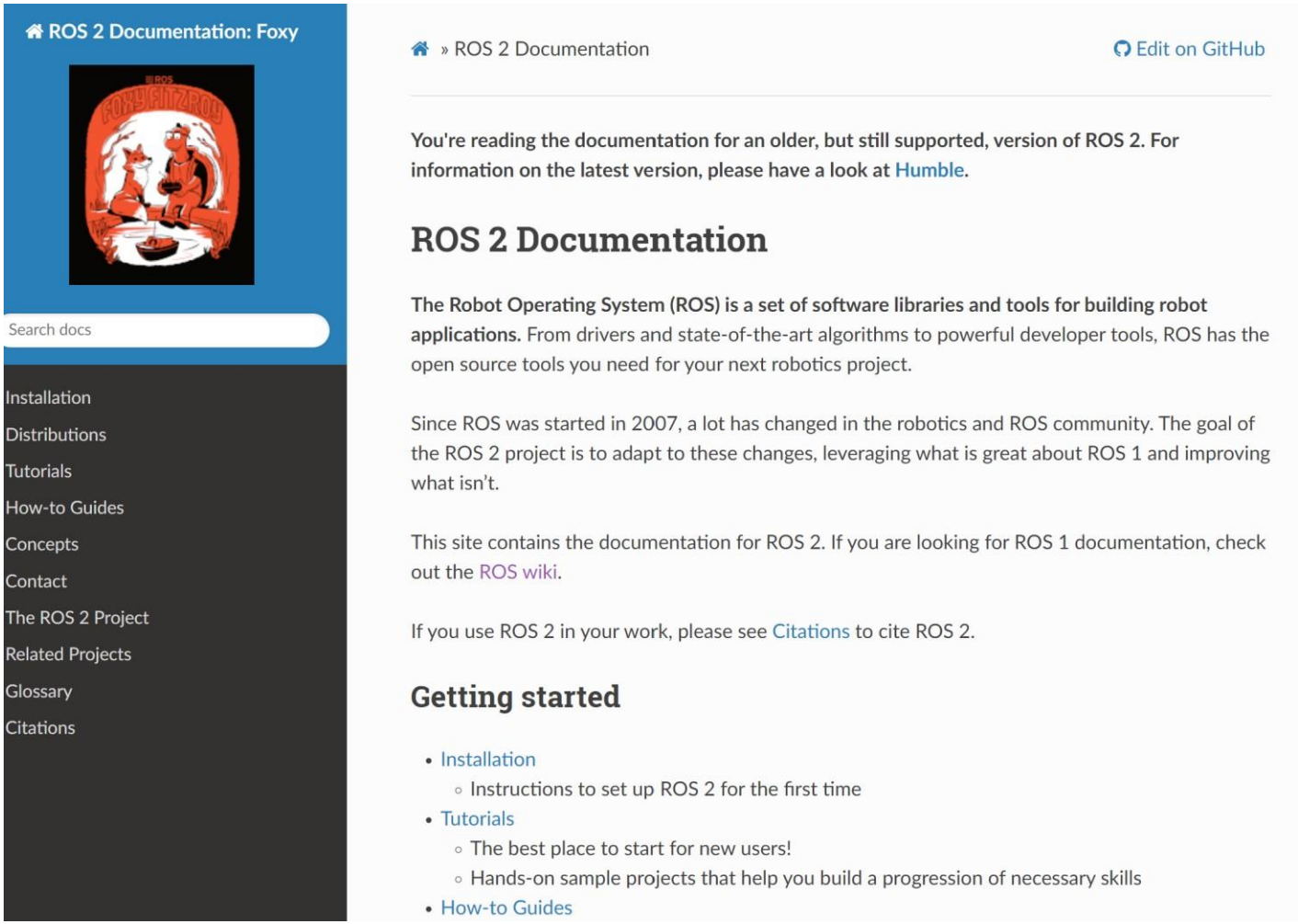
ISE Department

Prof. Mehdi Pirahandeh

Content

- Turtlesim
- ROS2 & RQT
- Understanding the basic concepts
- Colcon
- Creating a workspace
- Creating a package
- Activity session (OS & ROS 2 Test)

Introduction to Course



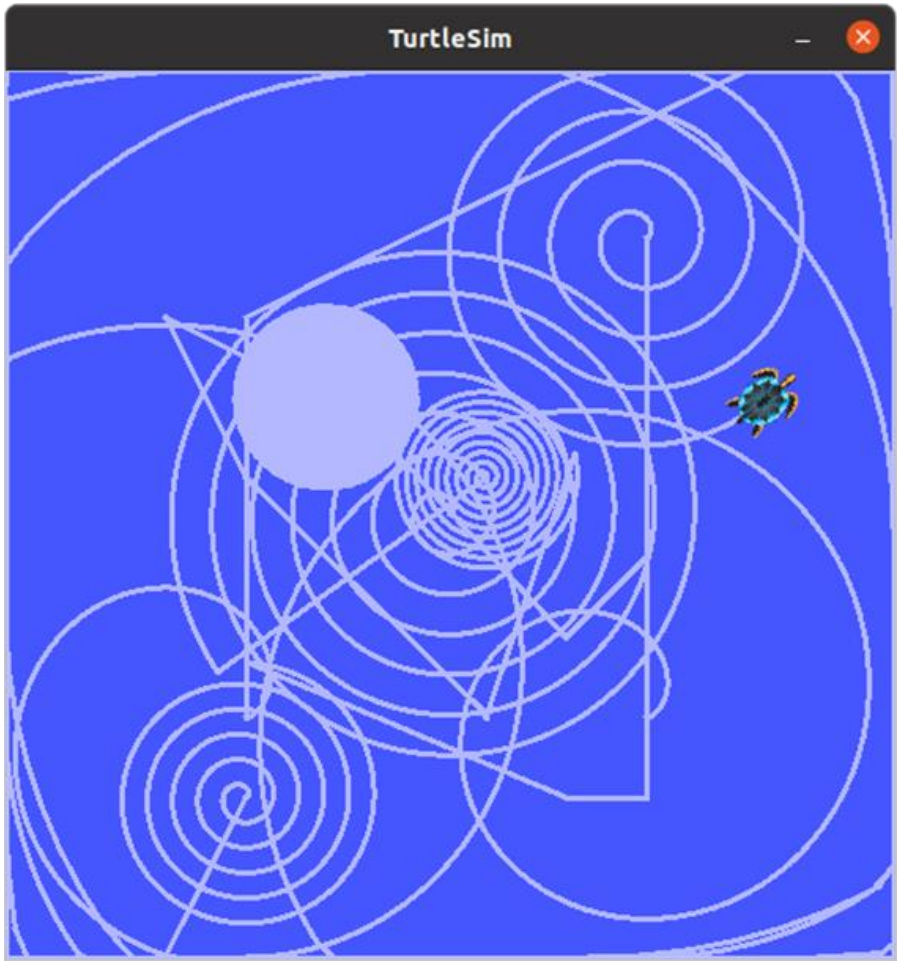
http://docs.ros.org/en/foxy/index.html

turtlesim

Turtlesim

- Turtlesim is a lightweight simulator for learning ROS 2.

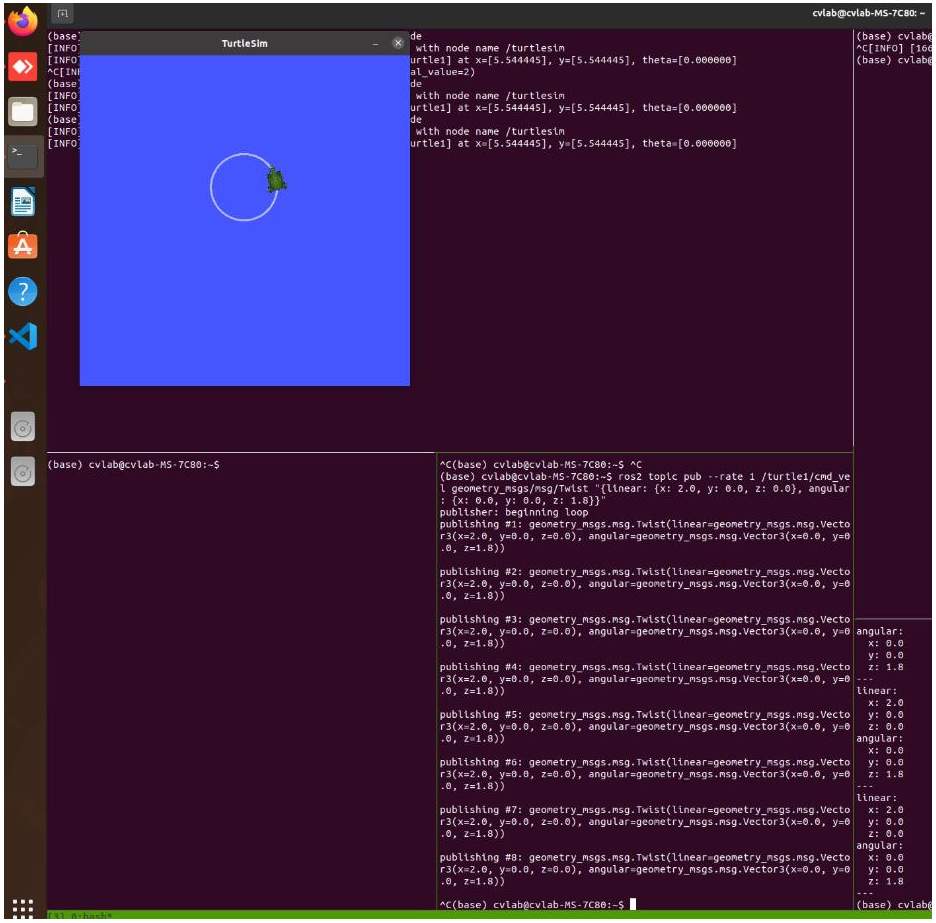
- Turtlesim illustrates what ROS 2 does at the most basic level, to give you an idea of what you will do with a real robot or robot simulation later on.



Turtlesim

- Turtlesim is a lightweight simulator for learning ROS 2.

- Turtlesim illustrates what ROS 2 does at the most basic level, to give you an idea of what you will do with a real robot or robot simulation later on.



Turtlesim: Installation

- Make sure your system is up-to-date
  - sudo apt update
- Install the turtlesim library
  - sudo apt install ros-rolling-turtlesim
- Check that the package is installed:
  - ros2 pkg executables turtlesim

Turtlesim: Installation

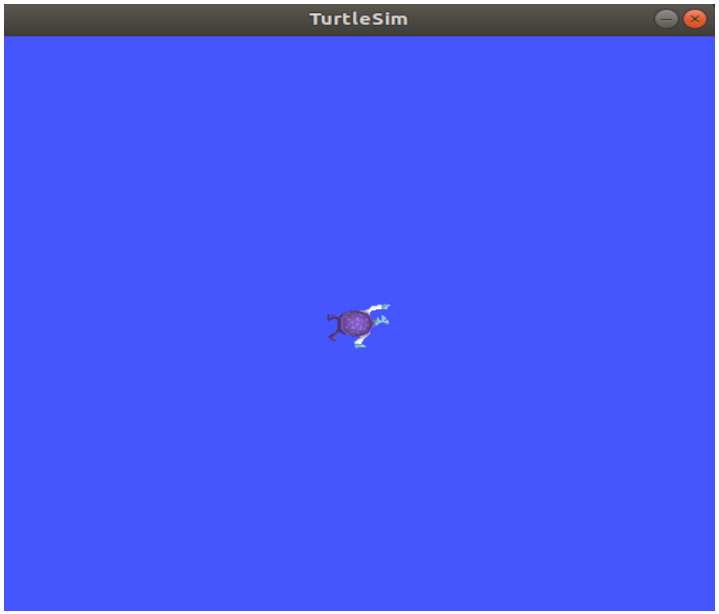
- The above command should return a list of turtlesim's executables:

```
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
```

Turtlesim: Start Turtlesim

- To start turtlesim, enter the following command in your terminal:
  - ros2 run turtlesim turtlesim_node

- The simulator window appears, with a random turtle in the center.



Turtlesim: Using Turtlesim

- To control the turtle type the following command:
  - ros2 run turtlesim turtle_teleop_key

- At this point you should have three windows open:
  - a terminal running turtlesim_node
  - a terminal running turtle_teleop_key
  - the turtlesim window

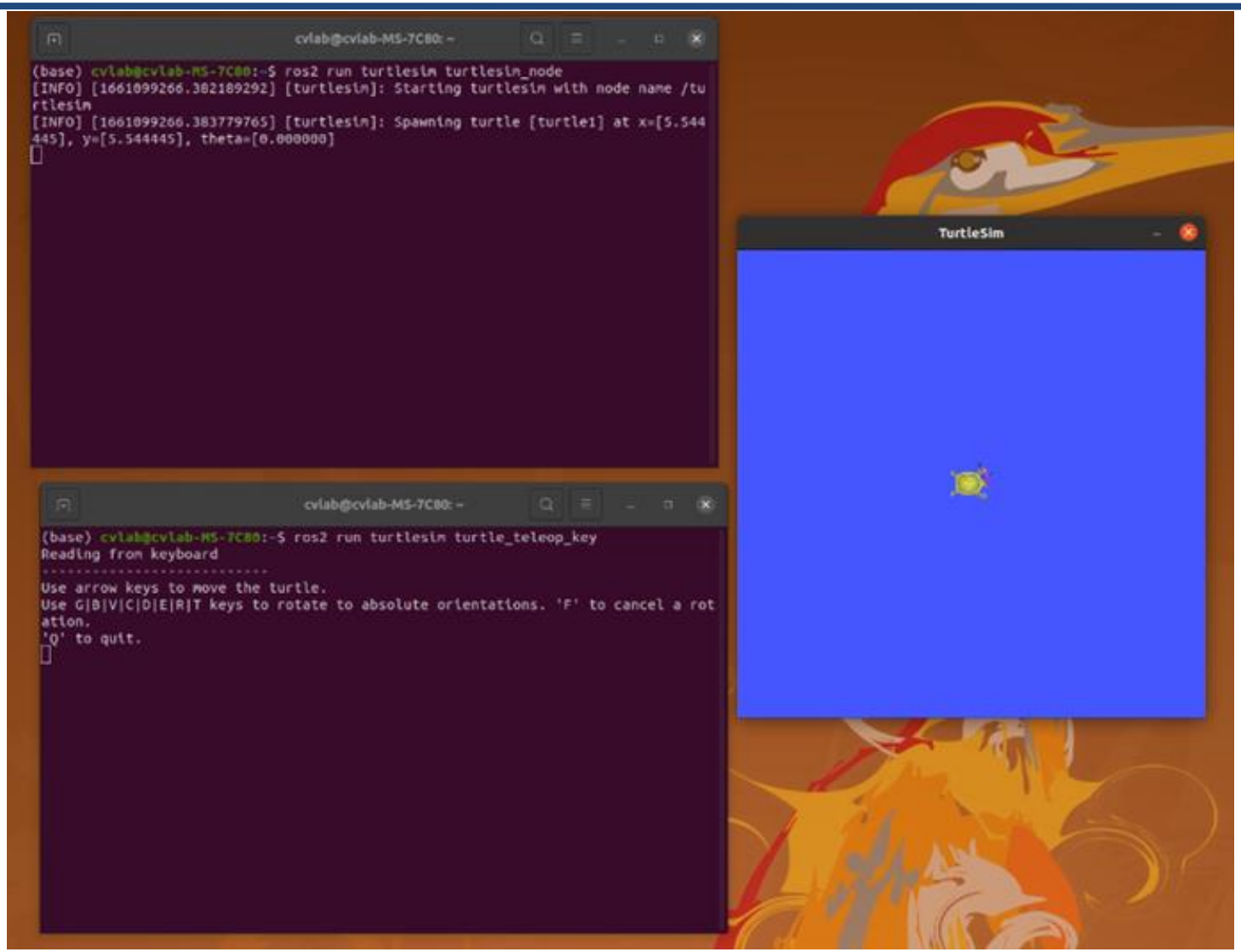
Turtlesim: Using Turtlesim

- To control the turtle type the following command:
  - ros2 run turtlesim turtle_teleop_key

- At this point you should have three windows open:
  - a terminal running turtlesim_node
  - a terminal running turtle_teleop_key
  - the turtlesim window

- Use the arrow keys on your keyboard to control the turtle.
- It will move around the screen, using its attached “pen” to draw the path it followed so far.

Turtlesim: Using Turtlesim

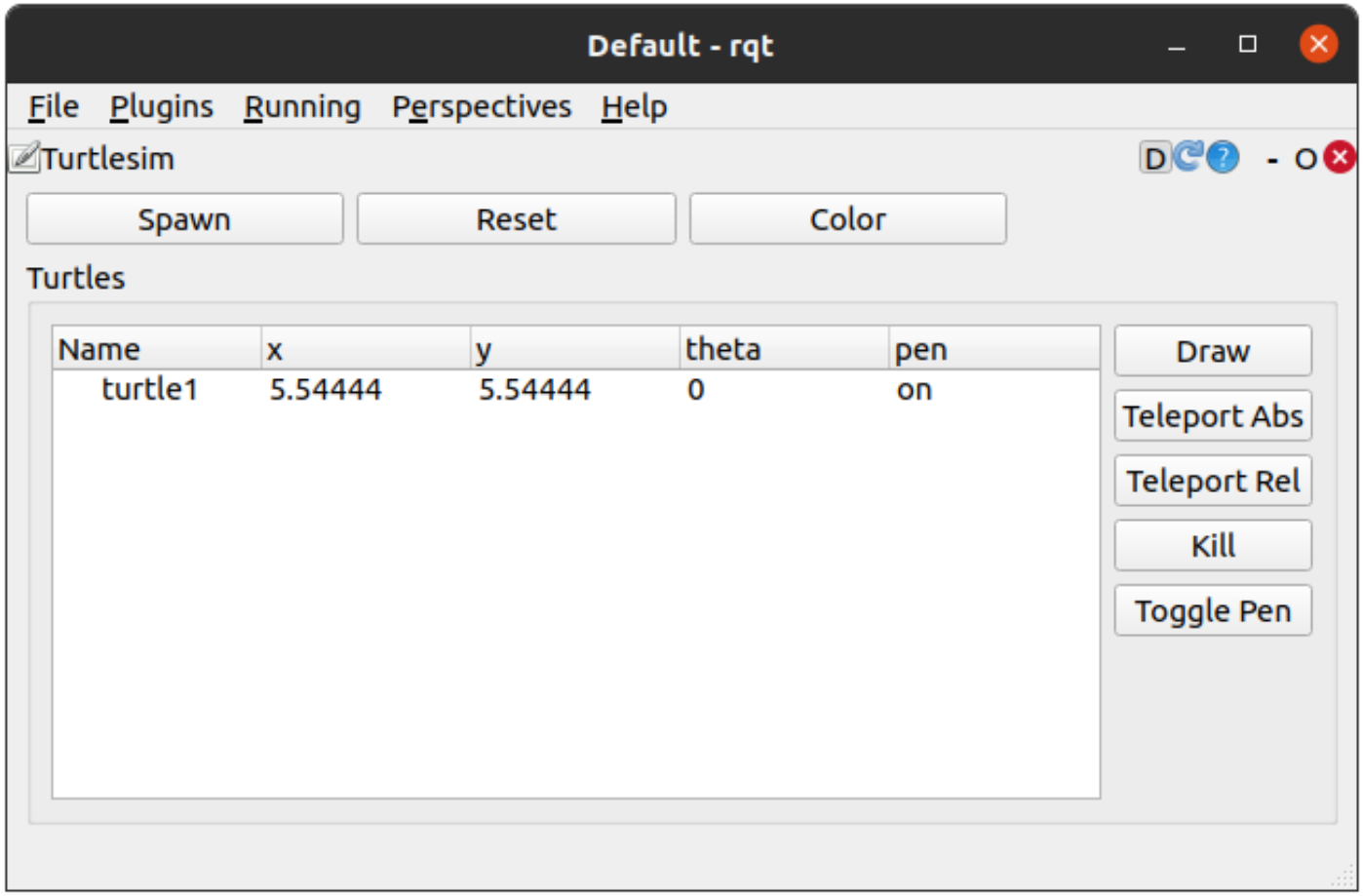


ROS 2 & RQT

ROS2 & RQT

- RQT is a GUI tool for ROS 2.
- Everything done in RQT can be done on the command line.
- RQT provides a more user-friendly way to manipulate ROS 2 elements.

ROS2 & RQT


Default - rqt
File
Plugins
Running
Perspectives
Help
Turtlesim
Spawn
Reset
Color
Turtles
Name
x
y
theta
pen
turtle1
5.54444
5.54444
0
on
Draw
Teleport Abs
Teleport Rel
Kill
Toggle Pen

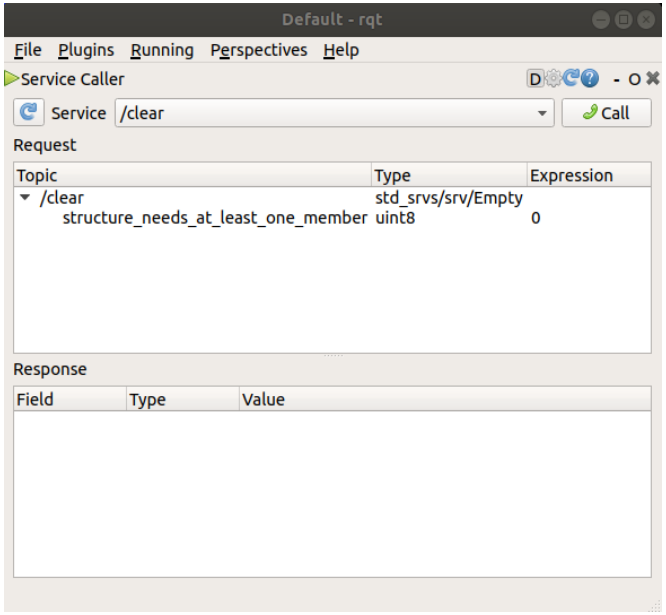
RQT : Installation

- Make sure your system is up-to-date
  - sudo apt update
- Install the rqt library and its plugins
  - sudo apt install ~nros-rolling-rqt*
- Using rqt
  - To run rqt by just typing **rqt** in the command line

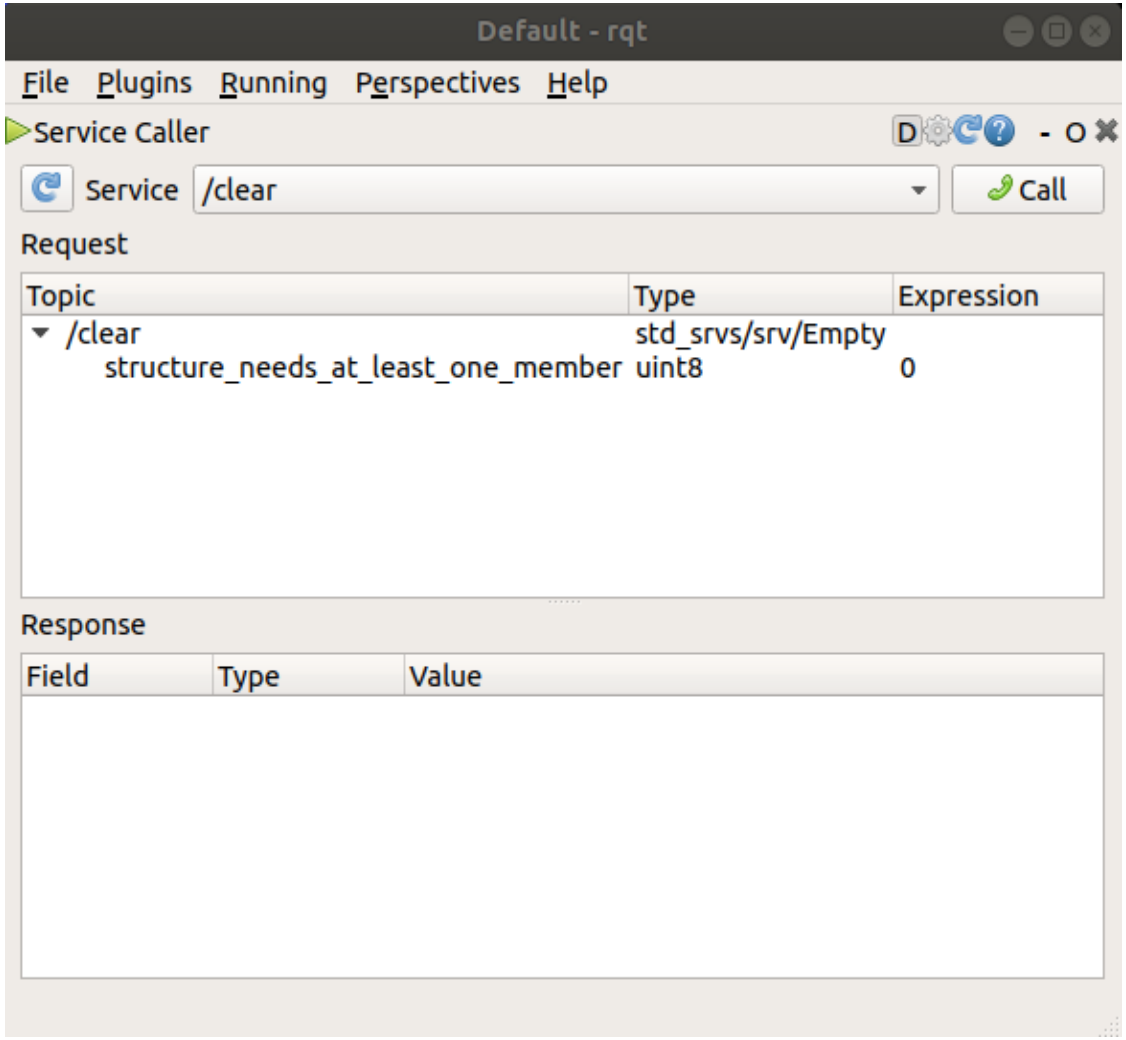
RQT : Running

- After running rqt the first time, the window will be blank.

- Then, select
  Plugins > **Services** > **Service Caller**
  option from the menu bar at the top.



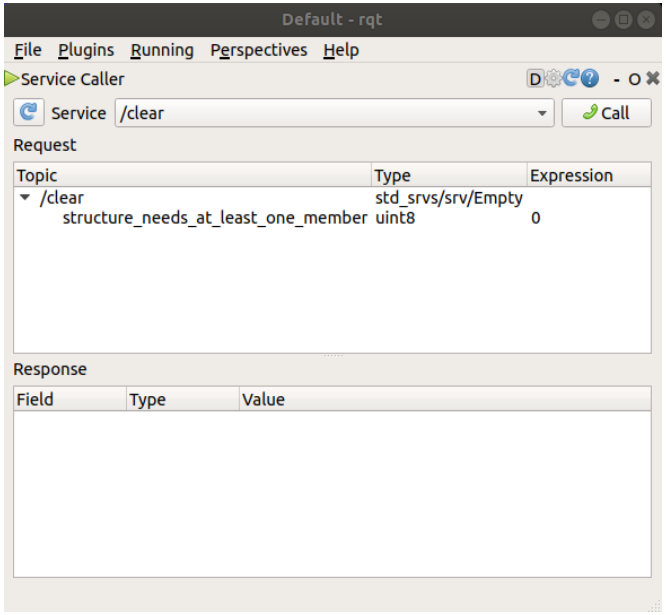
ROS2 & RQT


Default - rqt
File Plugins Running Perspectives Help
Service Caller
Service /clear
Call
Request
Topic Type Expression
/clear std_srvs/srv/Empty
structure_needs_at_least_one_member uint8 0
Response
Field Type Value

RQT : Running

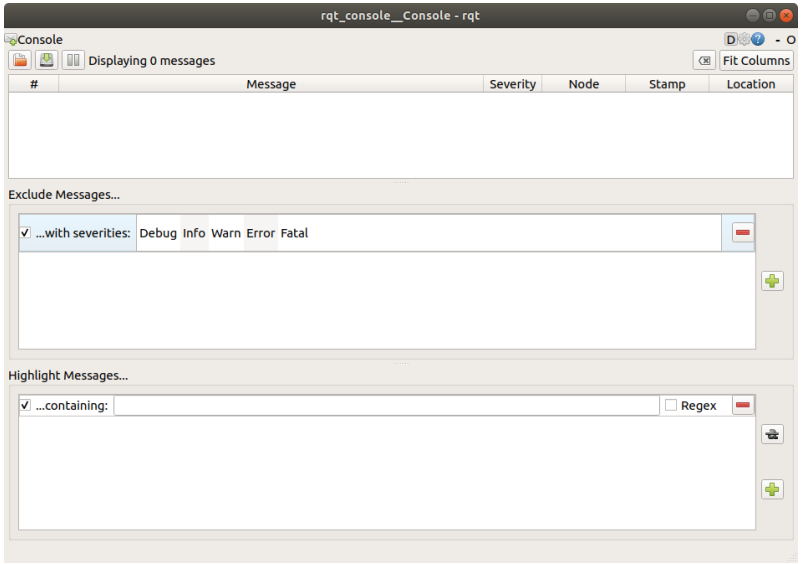
- After running rqt the first time, the window will be blank.

- Then, select
  Plugins > **Services** > **Service Caller**
  option from the menu bar at the top.



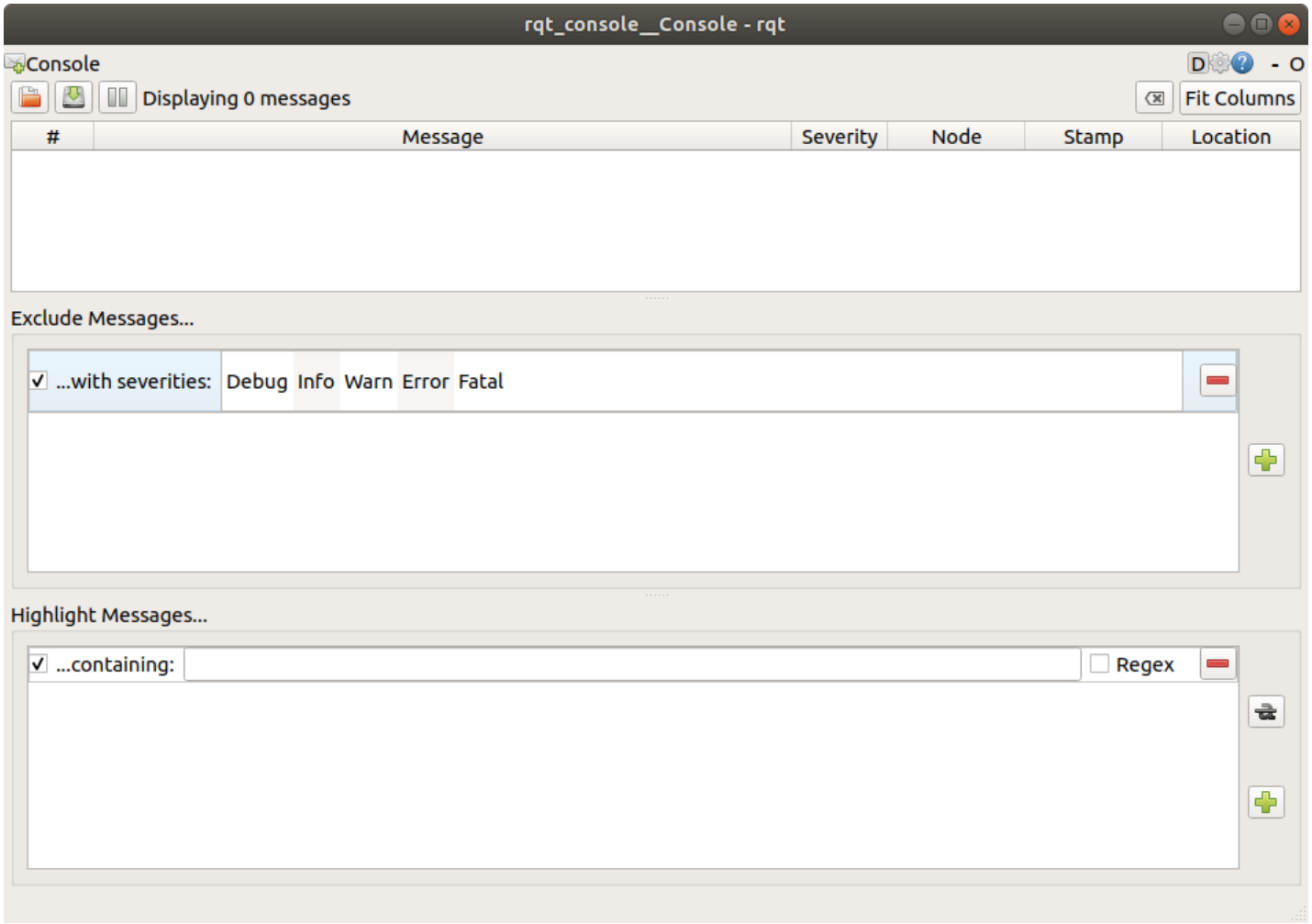
RQT : rqt_console

- **rqt_console** is a GUI tool used to introspect log messages in ROS 2.



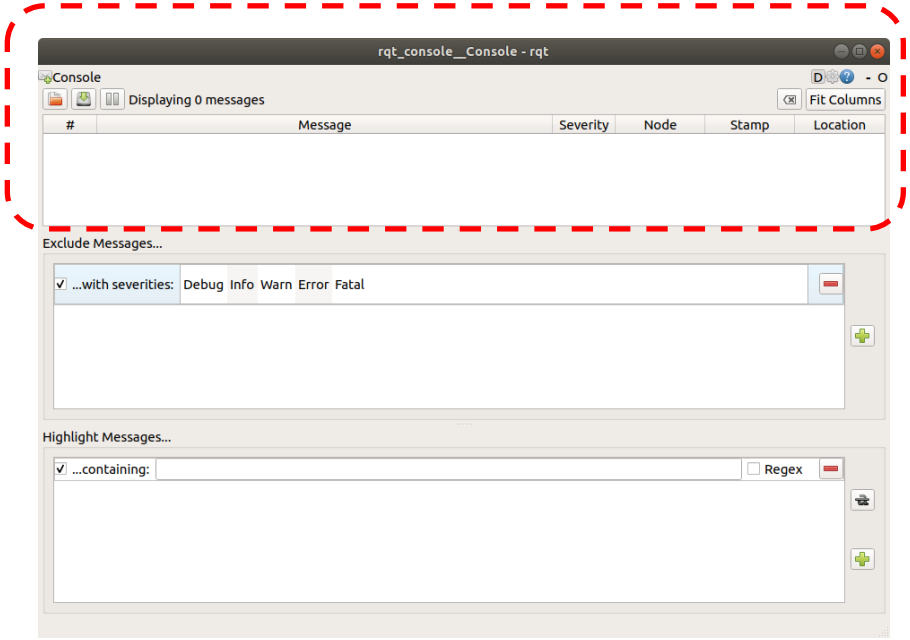
- To start **rqt_console** in a new terminal with the following command:
  - **ros2 run rqt_console rqt_console**

RQT : rqt_console



RQT : rqt_console

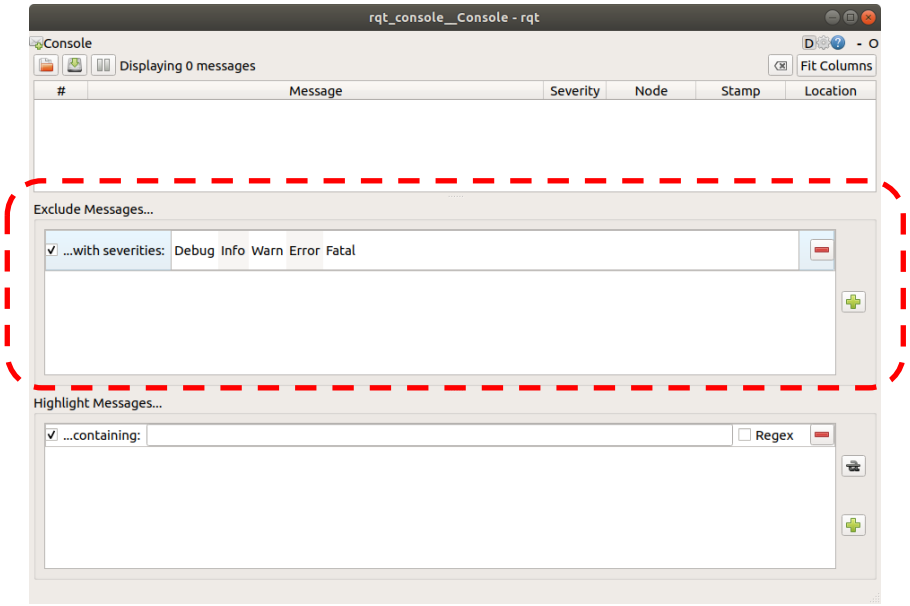
- The first section of the console i s where log messages from your system will display.



RQT : rqt_console

- The first section of the console i s where log messages from your system will display.

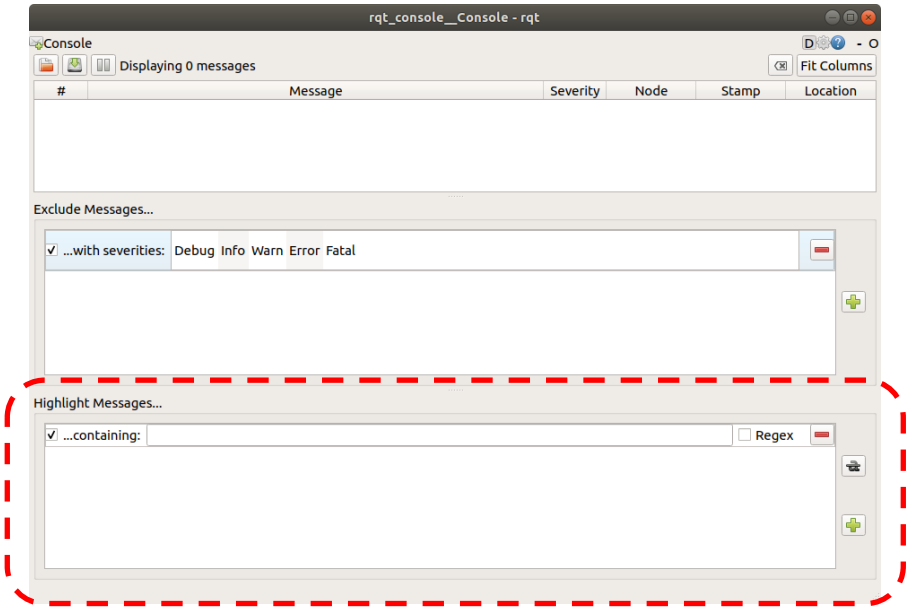
- In the middle, you can filter mes sages by excluding severity level s.



RQT : rqt_console

- The first section of the console i s where log messages from your system will display.

- In the middle, you can filter mes sages by excluding severity level s.
- The bottom section is for highlig hting messages that include a str ing you input.



ROS2 Nodes

ROS2 Graph

- The ROS graph is a network of ROS 2 elements processing data together at one time.
- It encompasses all executables and the connections between them if you were to map them all out and visualize them.

ROS 2 CONCEPTS

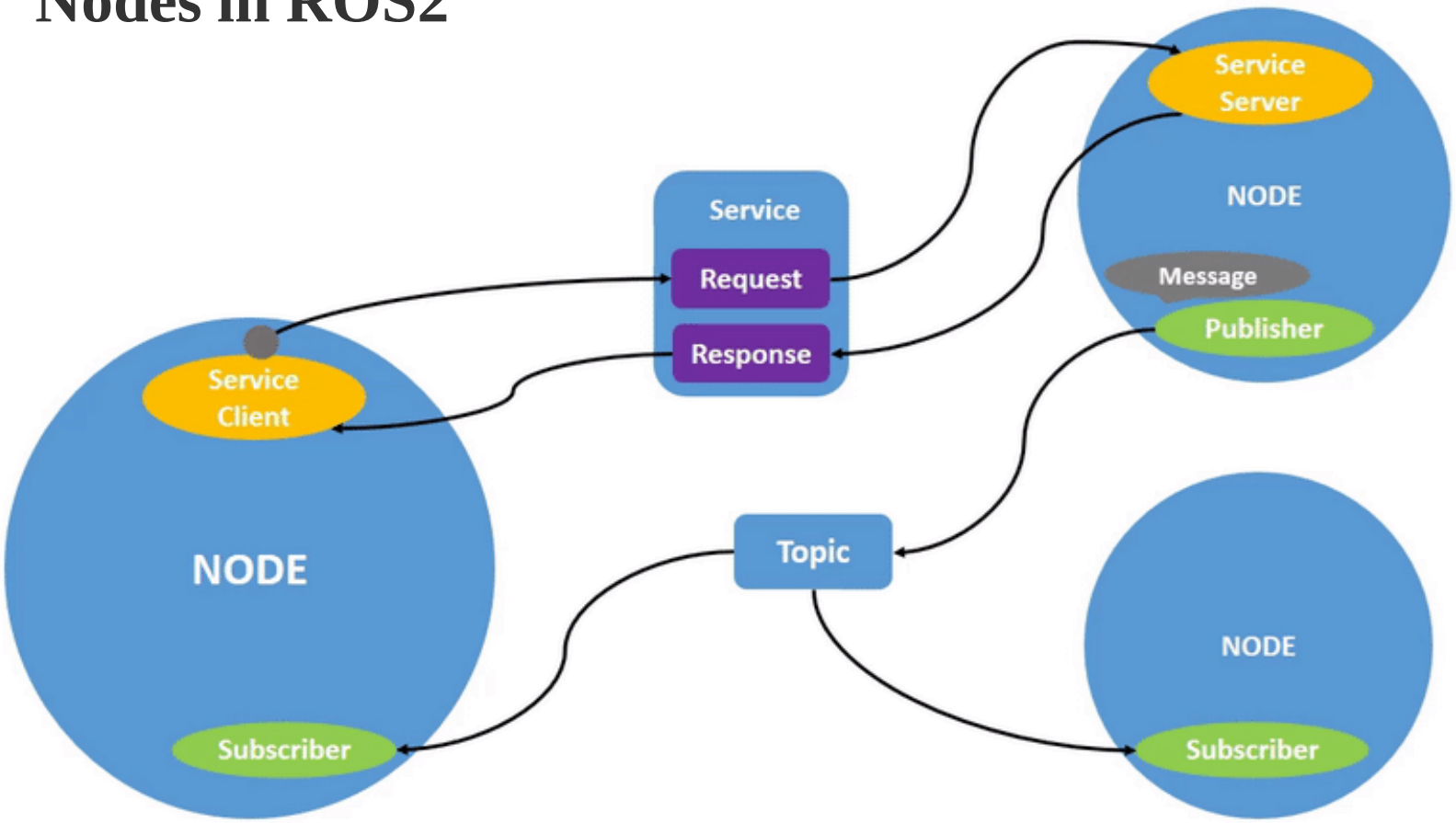
ROS2 Nodes

Nodes in ROS2

- Each node in ROS should be responsible for a single, module purpose.
- For example, one node can be used for controlling wheel motors, one node for controlling a laser range-finder, etc.
- Each node can send and receive data to other nodes via topics, services, actions, or parameters.

ROS2 Nodes

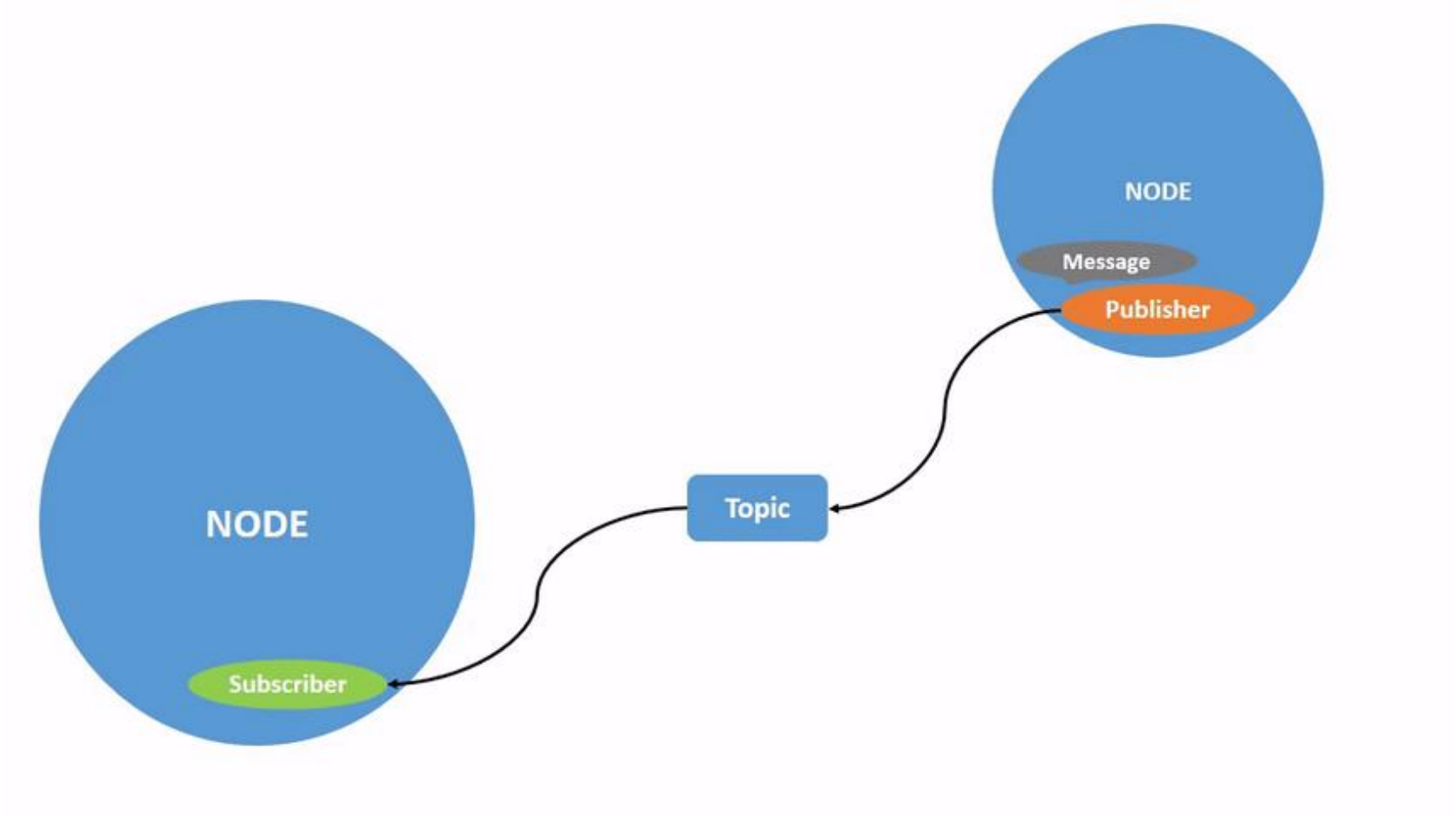
Nodes in ROS2



ROS2 Topics

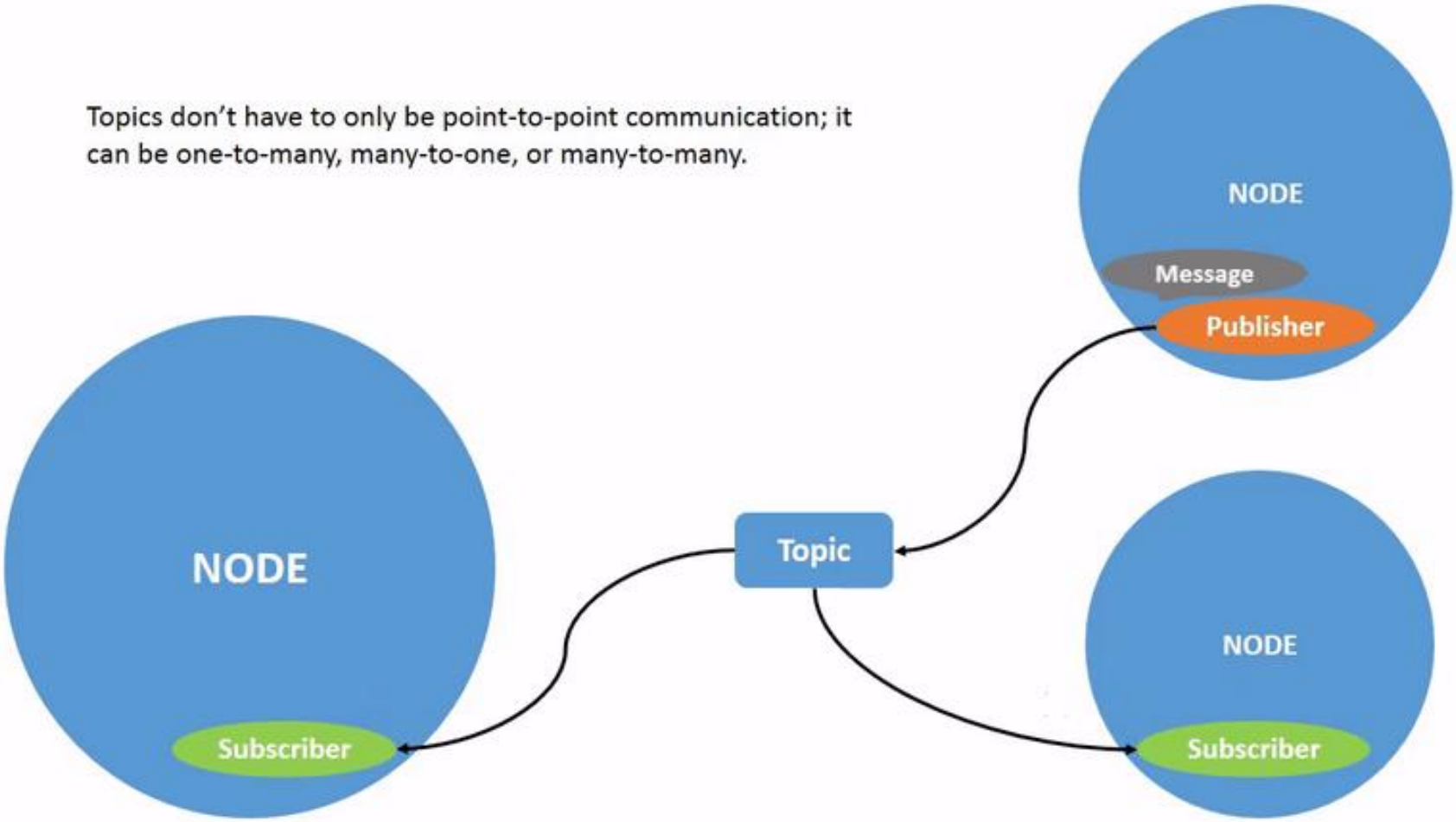
- ROS 2 breaks complex systems down into many modular nodes.
- Topics are a vital element of the ROS graph that acts as a bus for nodes to exchange messages.
- Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.

ROS2 Topics: Example 1



ROS2 Topics: Example 2

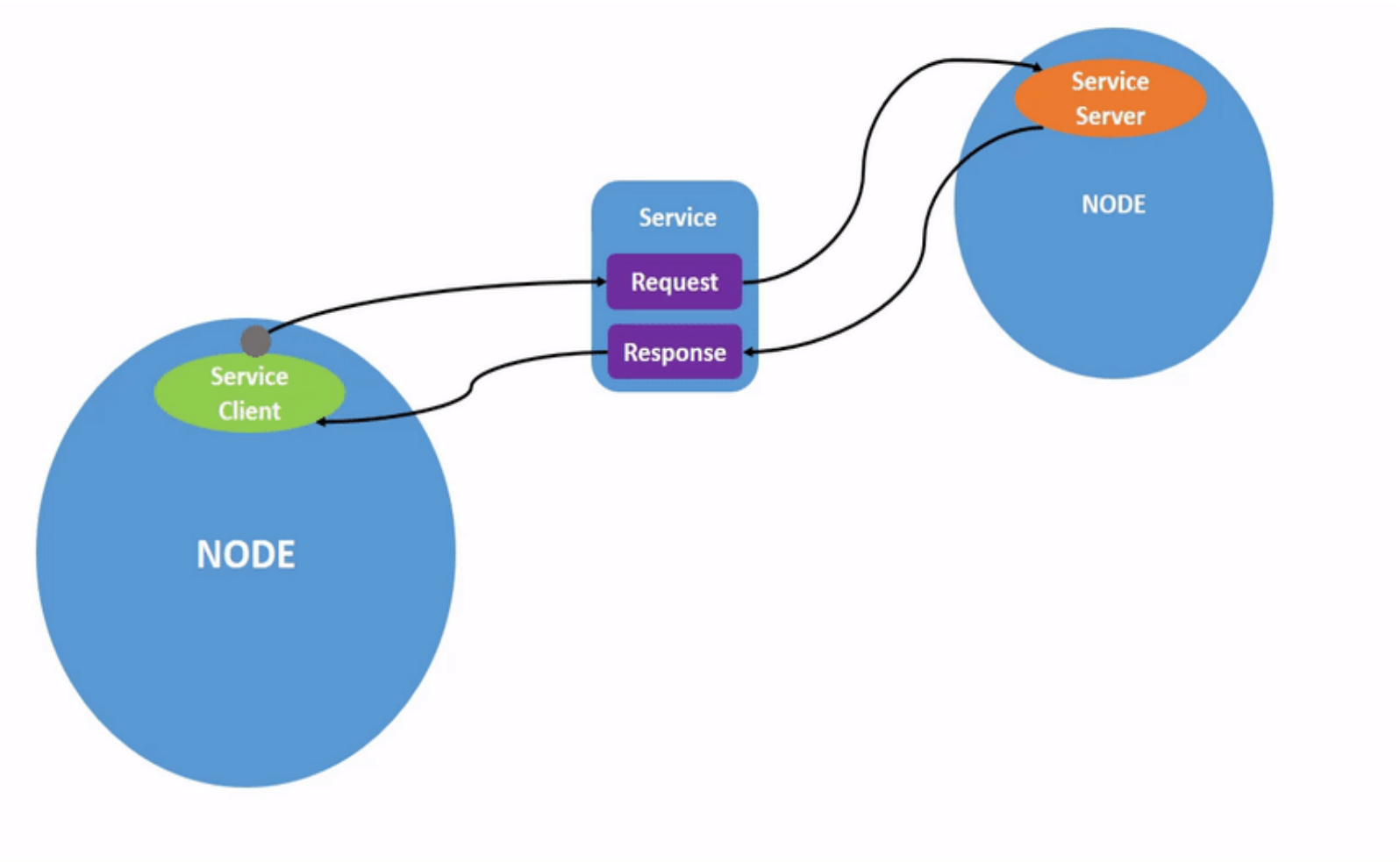
Topics don't have to only be point-to-point communication; it can be one-to-many, many-to-one, or many-to-many.



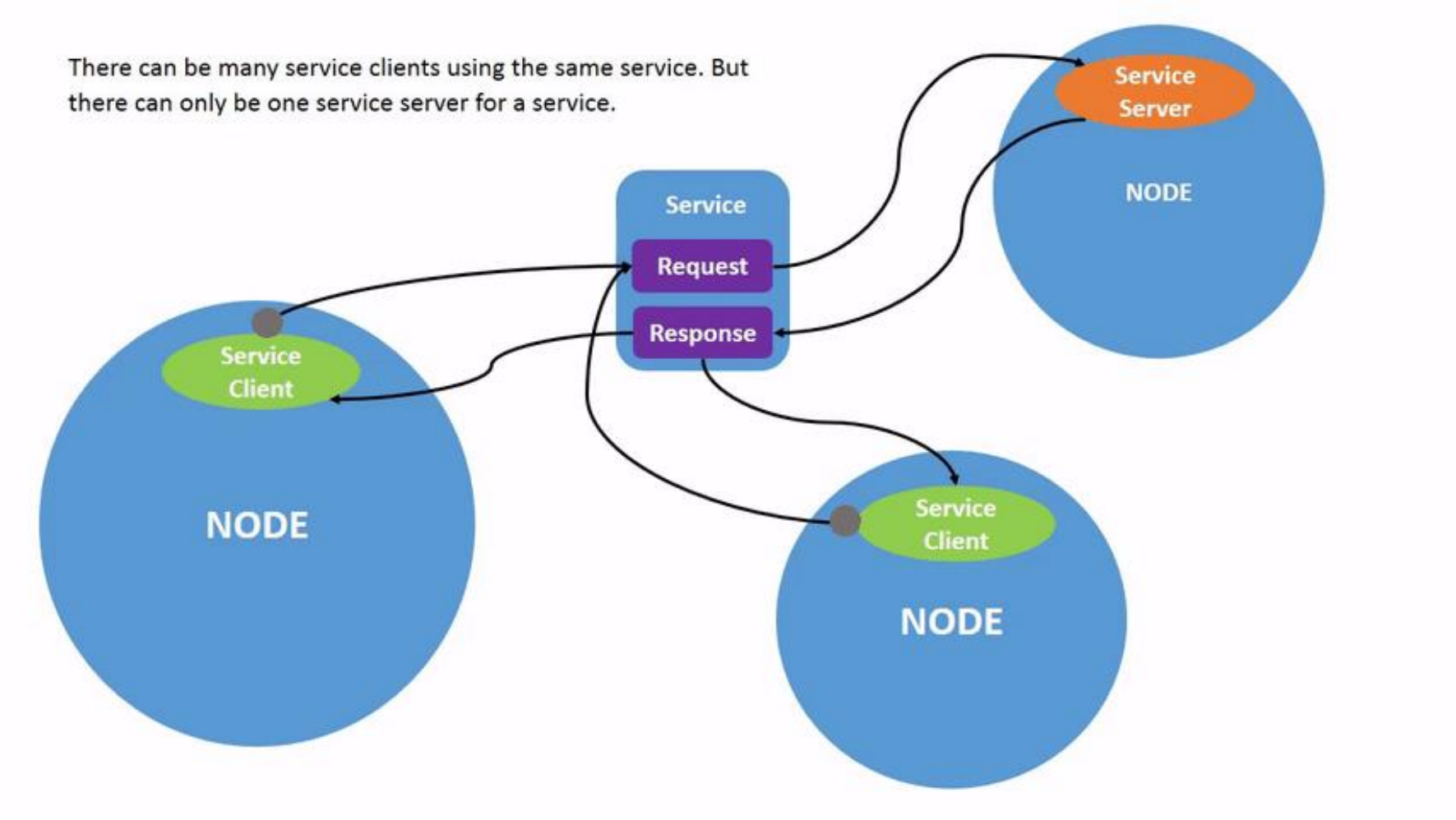
ROS2 Services

- Services are another method of communication for nodes in the ROS graph.
- Services are based on a call-and-response model, versus topics' publisher-subscriber model.
- While topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client.

ROS2 Services: Example 1



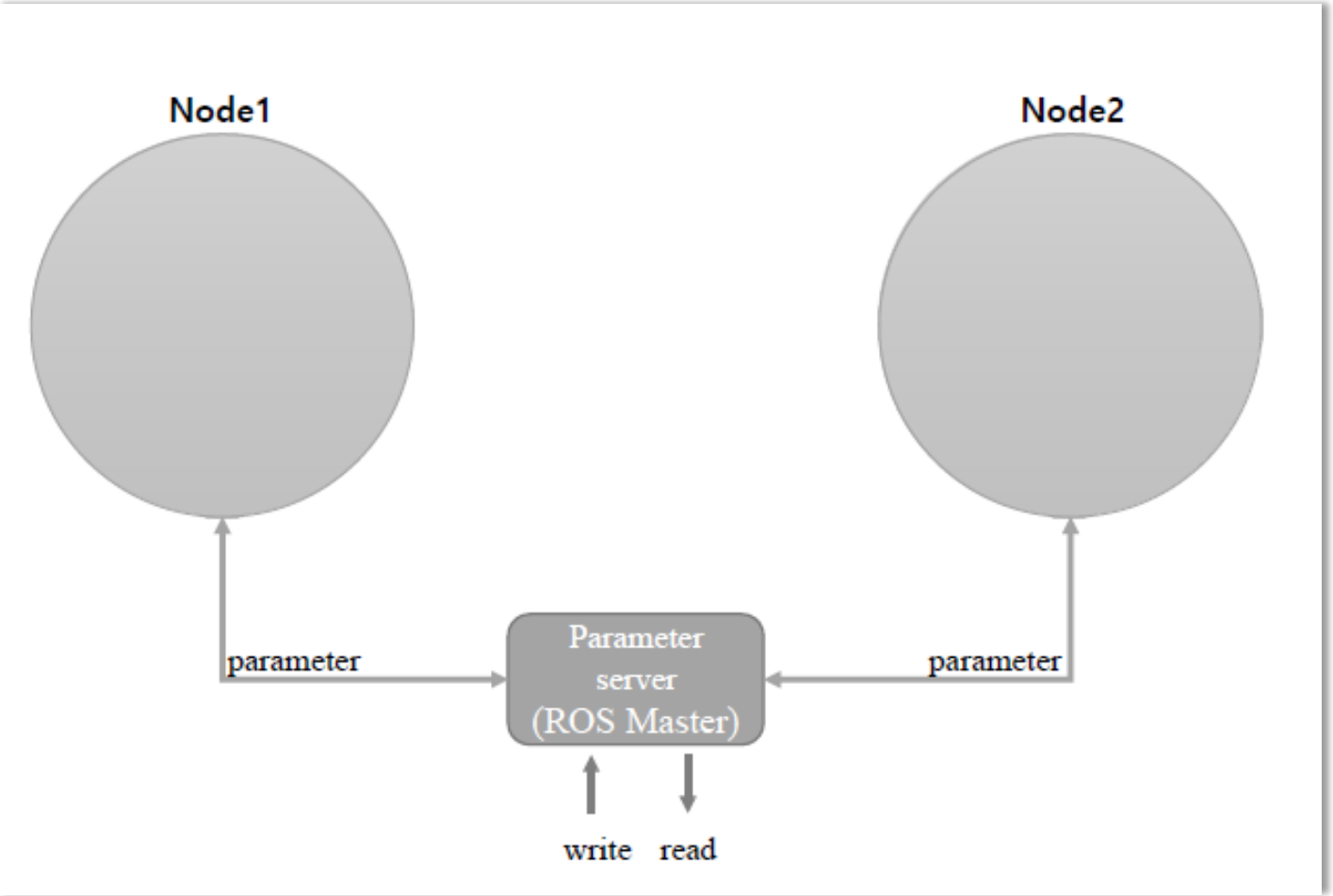
ROS2 Services: Example 2



ROS2 Parameters

- A parameter is a configuration value of a node. You can think of parameters as node settings.
- A node can store parameters as integers, floats, booleans, strings, and lists.
- In ROS 2, each node maintains its own parameters.

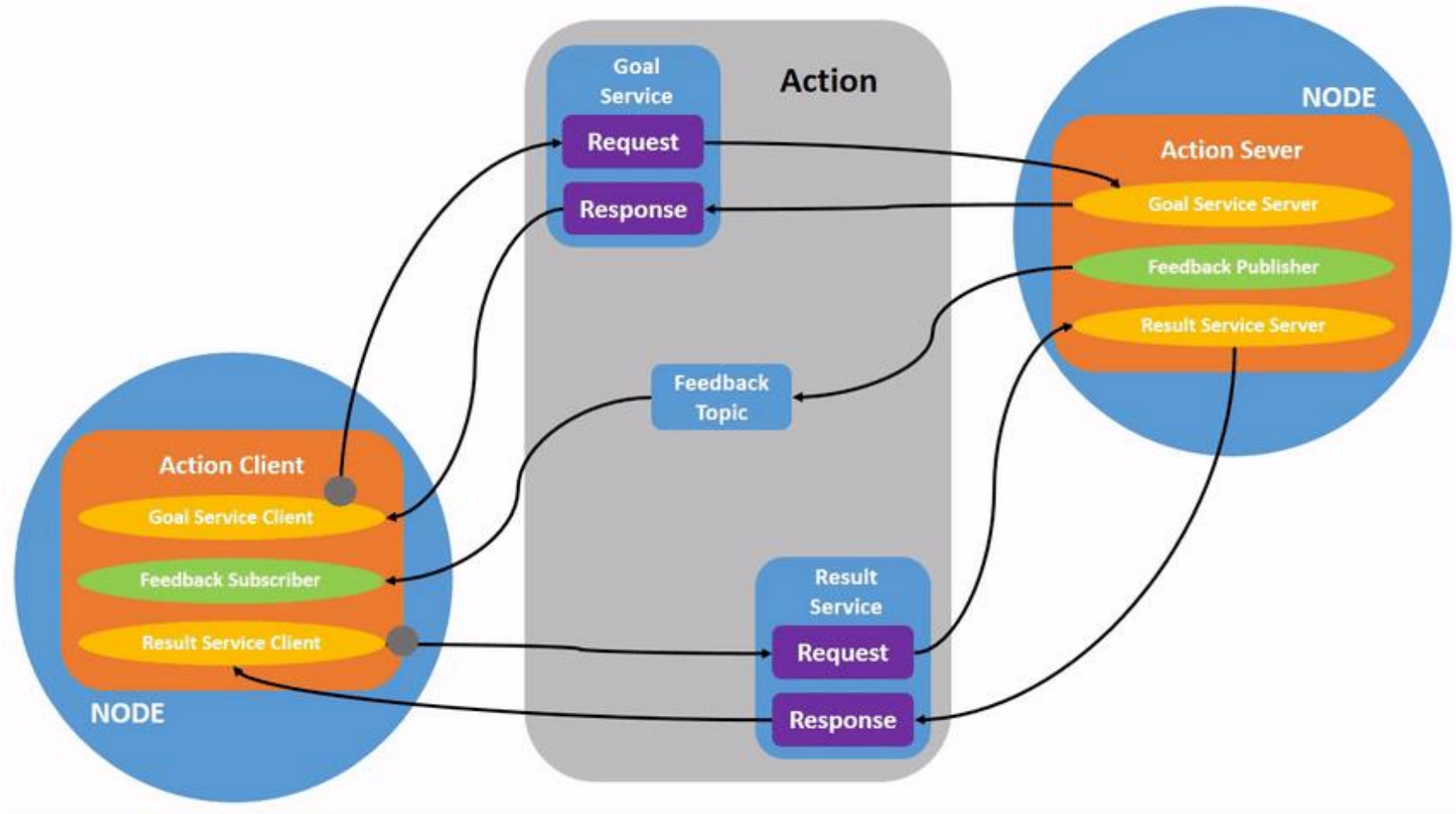
ROS2 Parameters



ROS2 Actions

- Actions are one of the communication types in ROS 2 and are intended for long-running tasks.
- They consist of three parts: a goal, feedback, and a result.
- Actions are built on topics and services.
- Their functionality is similar to services, except actions can be cancelled.
- They also provide steady feedback, as opposed to services which return a single response.

ROS2 Actions



ROS 2 Colcon

ROS2 Colcon

Using Colcon to build packages

- Colcon is an iteration on the ROS build tools *catkin_make*, *catkin_make_isolated*, *catkin_tools*, and *ament_tools*.

- Installing Colcon
  - sudo apt install python3-colcon-common-extensions

ROS2 Colcon: ROS workspace

- A ROS workspace is a directory with a particular structure.
- Colcon does out-of-source builds.
- By default, it will create the following directories as peers of the sr c directory: src, build, install and log

ROS2 Colcon: ROS workspace

- The build directory will be where intermediate files are stored.
- For each package a subfolder will be created in which e.g., CM ake is being invoked.

ROS2 Colcon: ROS workspace

- The install directory is where each package will be installed.
- By default, each package will be installed into a separate subdir ectory.
- The log directory contains various logging information about ea ch colcon invocation.

ROS2 Colcon: Create a workspace

- Create a workspace
  - mkdir -p ~/ros2_ws/src
  - cd ~/ros2_ws
- Add some sources
  - git clone https://github.com/ros2/examples src/examples -b rolling
- Build the workspace
  - colcon build --symlink-install

ROS2 Colcon: Create a workspace

- After the build is finished, we should see the build, install and log directories.

```
.
├── build
├── install
├── log
└── src

4 directories, 0 files
```

ROS2 Colcon: Create a workspace

- Source the environment
  - When *colcon* has completed building successfully, the output will be in the in stall directory.
  - Before you can use any of the installed executables or libraries, you will need to add them to your path and library paths.
  - colcon will have generated bash/bat files in the install directory.

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

ROS2 Colcon: Create a workspace

- Create your own Package
  - For convenience, you can use the tool ros2 pkg create to create a new pack age based on a template.
- Setup
  - The command colcon_cd allows you to quickly change the current working d irectory of your shell to the directory of a package.

```
echo "source /usr/share/colcon_cd/function/colcon_cd.sh" >> ~/.bashrc
echo "export _colcon_cd_root=/opt/ros/rolling/" >> ~/.bashrc
```

ROS2 Colcon: Create a workspace

- A workspace is a directory containing ROS 2 packages.
- Before using ROS 2, it’s necessary to source your ROS 2 installati on workspace in the terminal.
- This makes ROS2’s packages available for you to use in that termin al.

ROS2 Colcon: Create a workspace

- **Step 1: Source ROS 2 environment**
  - source /opt/ros/rolling/setup.bash

- **Step 2: Create a new Directory**
  - mkdir -p ~/ros2_ws/src
  - cd ~/ros2_ws/src

ROS2 Colcon: Create a workspace

- **Step 3: Clone the Github repo**
  - git clone https://github.com/ros/ros_tutorials.git -b rolling-devel
- **Step 4: Resolve dependencies**
  - rosdep install -i --from-path src --rosdistro rolling -y

ROS2 Colcon: Create a workspace

- **Step 5: Build the workspace with colcon**
  - colcon build

- **Step 6: Source the overlay**

  - source /opt/ros/rolling/setup.bash
  - cd ~/ros2_ws
  - . install/local_setup.bash

ROS2 Colcon: Create a workspace

- **Step 7: Modify the overlay**
  - You can modify turtlesim in your overlay by editing the title bar on the turtlesim window.
  - To do this, locate the turtle_frame.cpp file in ~/ros2_ws/src/ros_tutorials/turtlesim/src.
  - Open *turtle_frame.cpp* with your preferred text editor.
  - On line 52 you will see the function *setWindowTitle("TurtleSim")*;.
  - Change the value ”*TurtleSim*” to ”*MyTurtleSim*” and save the file.

- **Return to the second terminal**
  - Run turtlesim again:
    - ros2 run turtlesim turtlesim_node

ROS2 Colcon: Create a package

- **What is ROS2 Package?**

  - A package can be considered a *container* for your ROS 2 code.
  - If you want to be able to install your code or share it with others , then you'll need it organized in a package.
  - With packages, you can release your ROS 2
  - work and allow others to build and use it easily.

ROS2 Colcon: Create a package

- **What makes up a ROS 2 package?**
  - *package.xml* file containing meta-information about the package
  - *setup.py* containing instructions for how to install the package
  - *setup.cfg* is required when a package has executables, so ros2 run can find them
  - */<package_name>* - a directory with the same name as your package, used by ROS 2 tools to find your package, contains *__init__.py* .

ROS2 Colcon: Create a package

- **The simplest possible package may have a file structure th at looks like this:**

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

ROS2 Colcon: Create a package

- **Packages in Workspace**
  - A single workspace can contain as many packages as you want , each in its own folder.
  - You can also have packages of different build types in one
  - workspace (CMake, Python, etc.).
  - You cannot have nested packages

ROS2 Colcon: Create a package

- **A trivial workspace might look like this:**

```
workspace_folder/
    src/
      package_1/
          CMakeLists.txt
          package.xml

      package_2/
          setup.py
          package.xml
          resource/package_2
      ...
      package_n/
          CMakeLists.txt
          package.xml
```

ROS2 Colcon: Create a package

- **Step 1: Create a package**
  - Make sure you are in the src folder before running the package creation com mand.
  - The command syntax for creating a new package in ROS 2 is:

```
cd ~/ros2_ws/src
ros2 pkg create --build-type ament_python <package_name>
```

- **Step 2: Build a package**
  - colcon build

ROS2 Colcon: Create a package

- **Step 3: Source the setup file and use the package**

```
. install/local_setup.bash
ros2 run my_package my_node
```

- **Step 4: Examine the package contents**
  - Inside ros2_ws/src/my_package folder, you will see the files and f olders that *ros2 pkg create* command automatically generated:

```
my_package  package.xml  resource  setup.cfg  setup.py  test
```

ROS2 Colcon: Create a package

- **Step 5: Customize package.xml**
  - You may have noticed in the return message after creating your package that the fields *description* and *license* contain TODO notes
    - Input your name and email on the maintainer line.
    - Edit the description line to summarize the package
    - Update the *license* line

ACTIVITY SESSION

ROS2 Activity Session

- **Individual Task**

  – Review and practice all the commands in your VM machine.

  - Submit your work via the I-class discussion forum.

    - A GitHub link for the task
    - Write an analysis of your commands
    - Deadline for each activity is the next week Monday at 12:00 PM.

Brief break (if on schedule)

Prof. Mehdi Pirahandeh E-mail : mehdi@inha.ac.kr