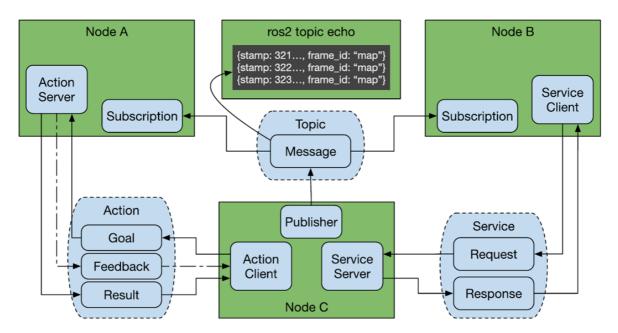
### 13.ROS2 custom interface message

#### 1. Introduction to communication interface

In the ROS system, no matter the topic, service or action, an important concept is used - communication interface.

Communication is not a person talking to himself, but a communication between two or more people. What is the content of the communication? In order for everyone to understand it easily, we can define a standard structure for the transferred data, which is the communication interface.

Interfaces can reduce dependencies between programs, making it easier for us to use other people's code, and for others to use our code. This is the core goal of ROS, to reduce reinvention of the wheel.



ROS has three commonly used communication mechanisms, namely topics, services, and actions. Through the interface defined by each communication type, various nodes can be organically linked together.

### 2. Create a custom interface process

The process of customizing interface messages is similar to the process of writing executable programs in function packages. The main steps are as follows:

- 1. Create an interface function package;
- 2. Create and edit .msg files, .srv files, and .action files
- 3. Edit the configuration file;
- 4. Compile;
- 5. Test.

# 3. Create a custom interface for action communication

In the case of [11.ROS2 action communication server], we have already demonstrated the complete process of creating an action communication interface. You can go back and review it first, so we will not go into details here.

# 4. Create a custom interface for topic communication

1. In the [ROS2 action communication server] course, we have created a custom interface function package. Create a new msg folder under the function package pkg\_interfaces, and create a new Person.msg file under the msg folder. Enter the following content in the file:

```
string name
int32 age
float64 height
```

2. Add the following configuration to package.xml and CMakeLists.txt:

```
M CMdeclistant U ×
yalboomcar.wa > arc > pkg.imerfaces > M CMdeclistant

if (Not CMAKE_CXX_STANDARD)

if (Not CMAKE_CXX_STANDARD)

set(CMAKE_CXX_STANDARD 14)

endif()

if (CMAKE_CXX_STANDARD 14)

endif()

if (CMAKE_CXX_STANDARD 14)

if (CMAKE_CXX_STANDAR
```

3. Enter the current workspace in the terminal and compile the function package:

```
cd ~/yahboomcar_ros2_ws/yahboomcar_ws
colcon build --packages-select pkg_interfaces
source install/setup.bash
```

4. Test

```
ros2 interface show pkg_interfaces/msg/Person
```

Under normal circumstances, the terminal will output content consistent with the Person.msg file.

```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 interface show pkg_interfaces/msg/Person
string name
int32 age
float64 height
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws#
```

#### 5. Create a custom interface for service communication

1. In the [ROS2 action communication server] course, we have created a custom interface function package. Create a new srv folder under the function package pkg\_interfaces, and a new Add.srv file under the srv folder. Enter the following content in the file:

```
int32 num1
int32 num2
int32 sum
```

2. Add the following configuration to package.xml and CMakeLists.txt:

```
find_package(rosidl_default_generators REQUIRED)
# uncomment the following section in order to fill in
                                                                                                                                                                                                                                              <buildtool depend>ament cmake/buildtool depend>
   rosidl_generate_interfaces(${PROJECT_NAME})
                                                                                                                                                                                                                                             cbuildtool_depend>rosidl_default_generators/buildtool_depend>
<exec_depend>rosidl_default_runtime</exec_depend>
<depend>action_msgs</depend>
if(BUILD_TESTING)
find_package(ament_lint_auto REQUIRED)
# the following line skips the linter which checks for copyrights
# uncomment the line when a copyright and license is not present
#set(ament_cmake_copyright_fOUND TRUE)
# the following line skips ceptint (only works in a git repo)
# uncomment the line when this package is not in a git repo
#set(ament_cmake_cpplint_FOUND TRUE)
```

3. Enter the current workspace in the terminal and compile the function package:

```
cd ~/yahboomcar_ros2_ws/yahboomcar_ws
colcon build --packages-select pkg_interfaces
source install/setup.bash
```

4. Test

```
ros2 interface show pkg_interfaces/srv/Add
```

Under normal circumstances, the terminal will output content consistent with the Person.msg file.

```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 interface show pkg_interfaces/srv/Add
int32 num1
int32 num2
int32 sum
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws#
```