Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Introduction to service and client ROS

04.04.2018

Erick Kramer

What is a Service?

- It is a request/reply communication paradigm.
- It is composed of two pairs of messages, one for the request and one for the reply.
- Two nodes participate in the communication process.
- One node offers the service and other node requests the service and waits for the response.
- A service is defined using **srv** files

Service type

- The service type is the package resource name of the .srv file, i.e. package name + name of the `.srv` file.
- E.g. `mysrvs/srv/PolledImage.srv` has the service type `mysrvs/PolledImage`

Service Tools

- rossrv - Displays information about .srv data structures.
  - `rossrv show` - Show service description
  - `rossrv info` - Alias for rossrv show
  - `rossrv list` - List all services
  - `rossrv md5` - Display service md5sum
  - `rossrv package` - List services in a package
  - `rossrv packages` - List packages that contain services
- rosservice - Lists and queries ROS services
  - `rosservice list` - print information about active services
  - `rosservice call` - call the service with the provided args
  - `rosservice type` - print service type
  - `rosservice find` - find services by service type

Service file

- The service file present the following structure:

```
#Request
message_type message
- - -
#Response
message_type message
```

- E.g.

```
string str
int8 a
uint32 b
int64 c
- - -
float64 result
```

Instructions

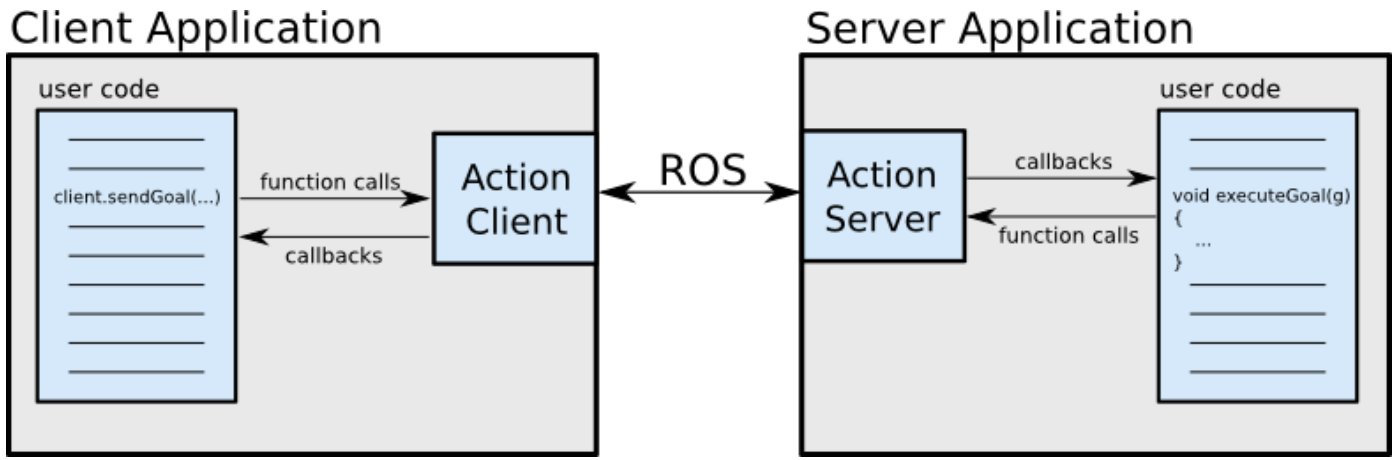
1. Go to the following file `content/ros/patterns.md` and follow the instructions.

Actionlib

- Actionlib stack gives a standardized interface containing preemptable tasks, i.e. tasks capable of being interrupted with the option of resuming the task at a later time.
- Actionlib package allows us to create servers that could perform "long-running" goals.
- Actions give the ability to cancel a service request during execution.
- Actions are also useful to get periodic feedback about the request.

Client-Server Interaction

- The communication is composed of two elements: ActionClient and ActionServer.
- The client allows the users to request "goals", while the server executes those goals.



Message specification

- **Goal:** Provides the sense of accomplishment for a certain task. It is sent to the ActionServer by the ActionClient.
  - *E.g. for a moving base, a goal could be a "PoseStamped" message that contains the information about where the robot should move.*
- **Feedback:** Allows the ActionServer to provides information about the current status of a certain goal to the ActionClient.
  - *E.g for a moving base, the current pose of the robot.*
- **Result:** Message sent from the ActionServer to the ActionClient when the goal is completed.
  - *E.g. the final pose of our moving base.*

Messages specification

- The elements of the actions are specified in the **.action** file.
- The action files are placed in a **./action** directory inside the package.
- An example of the structure would be

```
#Define the goal
uint32 dishwasher_id
---
#Define the result
uint32 total_dishes_cleaned
---
#Define a feedback message
float32 percent_complete
```

- http://wiki.ros.org/actionlib_tutorials/Tutorials/Writing%20a%20Simple%20Action%20Client%20%28Python%29
- http://wiki.ros.org/actionlib_tutorials/Tutorials/SimpleActionServer%28ExecuteCallbackMethod%29
- http://wiki.ros.org/actionlib_tutorials/Tutorials/Writing%20a%20Simple%20Action%20Server%20using%20the%20Execute%20Callback%20%28Python%29

1. Services (Theory)

2. Services (Practice)

3. Actions (Theory)

4. Actions (Example)

5. Reference

References

- http://wiki.ros.org/srv
- http://wiki.ros.org/actionlib