Aula 5 - ROS 2 (Python) — Services & Parameters

Professor: André L. Marcato

Universidade Federal de Juiz de Fora
Engenharia Elétrica — Robótica e Automação Industrial

November 14, 2025

Roteiro

1. Conceitos Fundamentais
2. Services em Python
3. Parameters
4. Prática

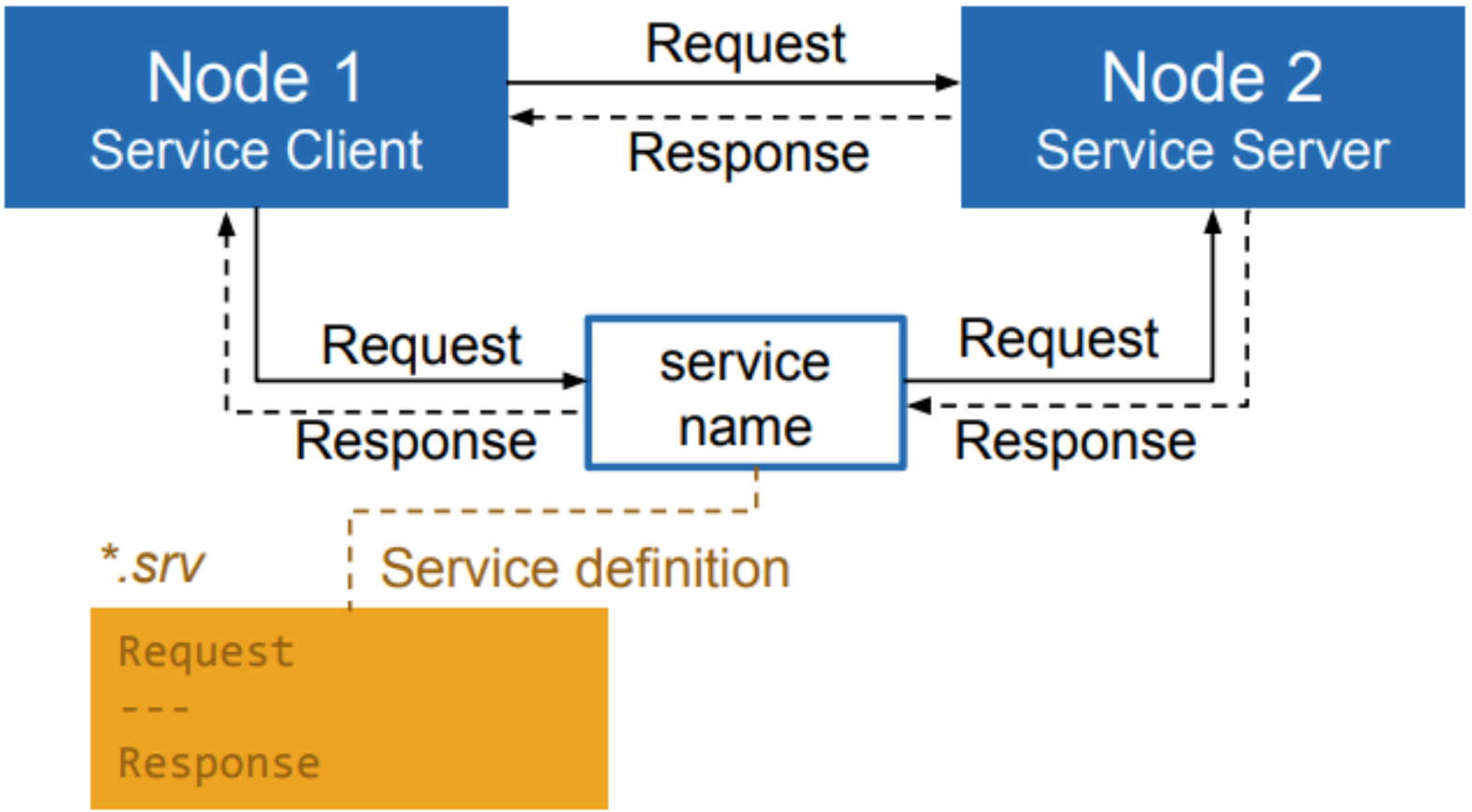
Roteiro

1. Conceitos Fundamentais
2. Services em Python
3. Parameters
4. Prática

ROS 2 (Python) — Services & Parameters

- **Service**: comunicação *request/response* entre dois nós (cliente → servidor).
- **Client**: envia uma **requisição**; **Server**: processa e retorna **resposta**.
- Usamos **Services** para operações pontuais/síncronas (reset, salvar, consultar estado).
- **Parameters**: pares nome/valor no escopo do nó; permitem configurar comportamento (CLI, YAML, API, json, etc.).

Arquitetura de Services



Interfaces .srv e fluxo de chamada

- Uma interface `.srv` define **Request** e **Response** (ex.: `AddTwoInts.srv`).
- Fluxo: **client** chama → **server** executa callback → retorna resposta.
- Ferramentas úteis: `ros2 service list`, `ros2 service type`, `ros2 interface show`, `ros2 service call`.

```
# AddTwoInts.srv (exemplo)
int64 a
int64 b
---
int64 sum
```

Services — comandos úteis

- `ros2 service list` — lista os serviços disponíveis.
- `ros2 service type <service_name>` — mostra o tipo `.srv` de um serviço.
- `ros2 interface show <pkg/srv/Tipo>` — exibe a definição (Request/Response).
- `ros2 service call <service_name> <srv_type> "{...}"` — envia uma requisição e imprime a resposta.

```
$ ros2 service list
$ ros2 service type /add_two_ints
$ ros2 interface show example_interfaces/srv/
    AddTwoInts
```

```
$ ros2 service call /add_two_ints \
  example_interfaces/srv/AddTwoInts "{a: 2, b: 3}"
```

Roteiro

1 Conceitos Fundamentais

2 Services em Python

3 Parameters

4 Prática

Servidor (Python) — AddTwoInts

```python
import rclpy
from rclpy.node import Node
from example_interfaces.srv import AddTwoInts

class AddTwoIntsServer(Node):
    def __init__(self):
        super().__init__('add_two_ints_server')
        self.srv = self.create_service(AddTwoInts, 'add_two_ints', self.
    cb)

    def cb(self, request, response):
        response.sum = request.a + request.b
        self.get_logger().info(f"Req: a={request.a}, b={request.b} -> sum
    ={response.sum}")
        return response

def main():
    rclpy.init()
    node = AddTwoIntsServer()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()
```

Cliente (Python) — AddTwoInts

```python
import rclpy
from rclpy.node import Node
from example_interfaces.srv import AddTwoInts

class AddTwoIntsClient(Node):
    def __init__(self):
        super().__init__('add_two_ints_client')
        self.cli = self.create_client(AddTwoInts, 'add_two_ints')
        self.req = AddTwoInts.Request()

    def send_request(self, a: int, b: int):
        while not self.cli.wait_for_service(timeout_sec=1.0):
            self.get_logger().info('Aguardando servio...')
        self.req.a, self.req.b = a, b
        future = self.cli.call_async(self.req)
        rclpy.spin_until_future_complete(self, future)
        return future.result()

def main(a=2, b=3):
    rclpy.init()
    node = AddTwoIntsClient()
    resp = node.send_request(a, b)
    node.get_logger().info(f"Resposta: sum={resp.sum}")
    node.destroy_node()
    rclpy.shutdown()
```

Roteiro

1 Conceitos Fundamentais

2 Services em Python

3 Parameters

4 Prática

Parâmetros — noções e API

- Tipos: `bool`, `int`, `double`, `string`, listas.
- Operações: **declarar**, **obter**, **definir**, **descrever**, **listar**.
- CLI: `ros2 param list/get/set/describe/dump/load`.

```
class UsesParams(Node):
    def __init__(self):
        super().__init__('uses_params')
        self.declare_parameter('rate_hz', 2.0)
        rate = self.get_parameter('rate_hz').
    get_parameter_value().double_value
        self.timer = self.create_timer(1.0 / rate, self.cb)
```

YAML de parâmetros e remapeamentos

Arquivo YAML (exemplo)

```
uses_params:
  ros__parameters:
    rate_hz: 5.0
    greeting: "Ola, ROS 2!"
```

Remapeamentos (--ros-args)

```
# Remapear topico e nome do no
$ ros2 run my_pkg my_node \
  --ros-args \
  -r __node:=renomeado \
  -r old_topic:=new_topic \
  --params-file params.yaml
```

Roteiro

1 Conceitos Fundamentais

2 Services em Python

3 Parameters

4 Prática

Prática — preparar pacote (server & client)

- Crie pacote Python com dependência `example_interfaces`.

```
$ cd ~/ros2_ws/src
$ ros2 pkg create --build-type ament_python
    my_services_py \
  --dependencies rclpy example_interfaces
# registre no setup.py (entry_points):
# 'console_scripts': [
#     'add_two_ints_server = my_services_py.server:
    main',
#     'add_two_ints_client = my_services_py.client:
    main',
#     'uses_params       = my_services_py.uses_params:
    main',
# ]
```

Prática — build e execução (1/2)

- Compile o workspace e execute o **server** em um terminal.

Build (uma vez):

```
$ cd ~/ros2_ws
$ colcon build --packages-select my_services_py \
  --symlink-install
$ source install/setup.bash
```

Terminal 1 — Server:

```
$ source ~/ros2_ws/install/setup.bash
$ ros2 run my_services_py add_two_ints_server
```

Prática — build e execução (2/2)

- Execute o **client** em outro terminal e use ferramentas de inspeção.

Terminal 2 — Client:

```
$ source ~/ros2_ws/install/setup.bash
$ ros2 run my_services_py add_two_ints_client 41 1
```

Inspeção rápida:

```
$ ros2 service list
$ ros2 service type /add_two_ints
$ ros2 interface show example_interfaces/srv/
    AddTwoInts
```

Prática — `ros2 service call`

- Teste o servidor diretamente pela CLI.

```
$ ros2 service call /add_two_ints \
  example_interfaces/srv/AddTwoInts "{a: 2, b: 3}"
```

- deixe o `server` rodando e repita chamadas com valores diferentes.

Prática — parâmetros (CLI & YAML)

CLI de parâmetros

```
$ ros2 param list uses_params
$ ros2 param get  uses_params rate_hz
$ ros2 param set  uses_params rate_hz 10.0
$ ros2 param describe uses_params rate_hz
```

Rodar com YAML & remap

```
$ ros2 run my_services_py uses_params \
  --ros-args \
  --params-file params.yaml \
  -r __node:=cfg_node \
  -r /old:=/new
```

Exercício final

- Adapte o servidor para **usar um parâmetro** `scale` (default = 1.0) e retornar `scale * (a + b)`.
- Exponha `scale` via **YAML** e mostre efeito com `ros2 service call`.
- Remapeie o nome do serviço de `/add_two_ints` para `/sum` usando `-r`.