RabbitMQ RPC клиент

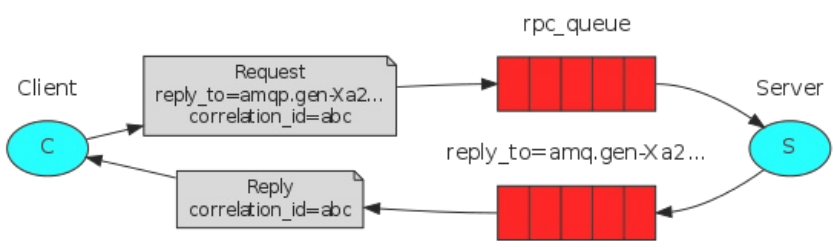
В предыдущем задании мы рассмотрели, что такое rpc, и написали реализацию вызова удаленной функции на сокетах в своем loop. Теперь давайте реализуем RPC с помощью RabbitMQ.

tasks/rpc_rabbit/rpc.py

Нужно дополнить класс RPC и реализовать методы:

- **create** должна создать объект RPC и подготовить все необходимые компоненты
- **register** должна зарегистрировать функцию-обработчик, которая будет вызвана, когда придет сообщение с соответствующим именем
- **call** — функция, которая должна вызвать удаленную процедуру с определенным именем

Для реализации задачи нужно сделать 2 очереди: очередь входящих сообщений и очередь ответов. **Важно:** очередь ответов должна быть уникальна для клиента. При отправке сообщения клиент указывает reply_to (название ответной очереди) для определения, куда серверу слать ответ. Поле reply_to является системные полем для объекта Message RabbitMQ.



Зачем?

RPC через RabbitMQ используется, когда невозможно/очень сложно реализовать http-запрос. Например, вы реализуете взаимодействие с банком, и вам нужно вызвать метод сервера, который находится в контуре банка (сервер расположен на серверах банка во внутренней сети). Согласовать «дырку» в банк ооочень сложно и для этой операции нужно много обоснований. Поэтому более быстрым решение может быть внешний RabbitMQ, через который будет организовано взаимодействие, а запросы из контура банка во внешнюю сеть согласовываются намного проще, чем наоборот.

Интерфейс пользователя-программиста

Для того, чтобы сделать запрос, нужно вызвать await rpc.call. Переменные для вызова удаленной функции нужно передать в параметре kwargs:

```
connection = await aio_pika.connect(os.getenv("RABBITMQ_URL"))
async with connection:
    channel = await connection.channel()
    rpc = await RPC.create(channel)
    print(await rpc.call("sleep", kwargs={"t": 3}))
```

Чтобы сделать RPC-сервер, нужно:

```
async def foo(t):
    pass

connection = await aio_pika.connect(os.getenv("RABBITMQ_URL"))
channel = await connection.channel()
rpc = await RPC.create(channel)
await rpc.register("foo", foo)
```

После регистрации функции-обработчика RPC должен начать принимать и выполнять запросы из очереди RabbitMQ.

Помимо интерфейса выше, RPC-client должен отвечать следующим требованиям:

- Функция call должна возвращать результат, который вернула функция на удаленной стороне
- Функция call должна выбрасывать исключение, если время ожидания ответа истекло
- Если функция-обработчик упала с ошибкой, то нужно выбросить исключение на стороне клиента

Как проверить

Для запуска RPC-сервера нужно:

```
python run_rpc_rabbit.py server
```

Для запуска RPC-клиента нужно:

```
python run_rpc_rabbit.py client
```

Сервер должен работать бесконечно и приниматься новые сообщения. Клиент запускает выполнение удаленной процедуры, ожидает ее выполнения и завершается.

Как проверить тестами

```
pytest tests/rpc_rabbit -vv -s
```