Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

RPC and Rendezvous

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

RPC introduction & Rendezvous (1/3)

- (A)synch msg passing are powerful enough for all four kinds of processes (clients, servers, filters, peers)

- But …
  - Two-way information flow between client ←→ server requires explicitly a **two-message exchange** using two different message channels
  - Each client requires a different message channel

RPC introduction & Rendezvous (2/3)

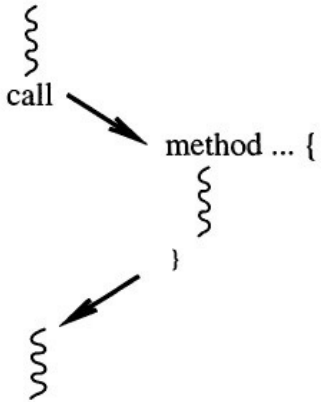
- Client/server interactions via **remote-procedure call (RPC)** or **rendezvous**
  - They both combine aspects of sync msg passing and monitors
    - Server exports operations that are to be invoked
    - Invocation is synchronous → Caller (client) delays until operation completes
  - An operation is a two-way communication channel
    - Caller (client) → Server …. Server → Caller (client)

RPC introduction & Rendezvous (3/3)

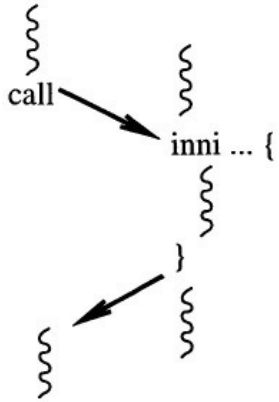
- Involves 2 processes
  - Invoking process (client)
  - Handling process (server)

- The handling process is a process that exists already
  - **No new** process is created!
- It's a synchronous protocol (from the invoking process' perspective)
- Both process can be on different machines

Difference between RPC and Rendezvous



remote procedure call

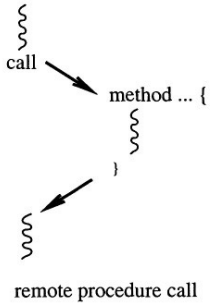


rendezvous

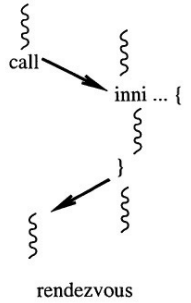
Serviced by:	**method (`op`)**	**input statement (`inni`)**

Key take away messages

- Rendezvous and RPC are sync. mechanisms
- Using a two-way channel for communication
- No new process created for server
- Service by operation (RPC)…
- … or by input statement (Rendezvous)



remote procedure call



rendezvous

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

RPC and Rendezvous

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

RPC and Rendezvous

RPC in Python

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

RPC recap

- RPC = Remote procedure call
- Synchronous message passing
- 2-way communication
  - RPC in Java is called RMI (Remote Method Invocation)
- Invoker and Server can be on different machines → distributed computing

RPC recap

- RPC = Remote procedure call
- Synchronous message passing
- 2-way communication
- ~~Like rendezvous, but a new process will be created on the server side~~
  - RPC in Java is called RMI (Remote Method Invocation)
- Invoker and Server can be on different machines → distributed computing

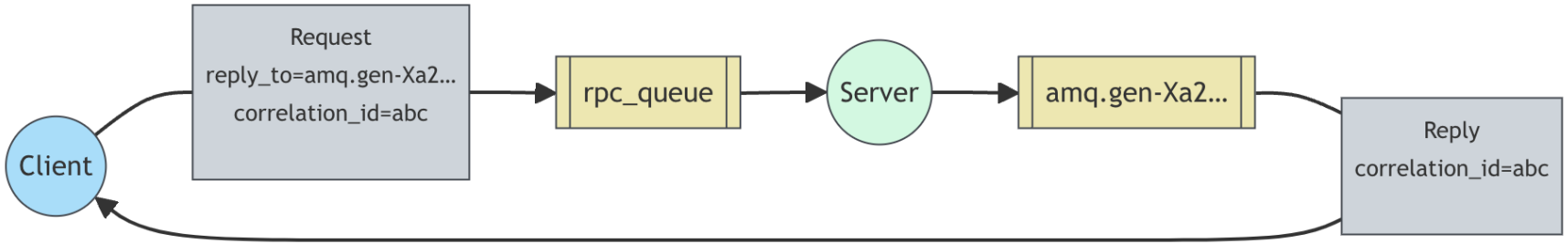
Remote process call - Python

- RPC in Python are not available *out of the box*.
- Additional libraries (e.g. xmlrpc, tinyrpc, …) provides the functionality
- RabbitMQ gives also the needed tools for doing RPC

- Read the RabbitMQ's tutorial:

  https://www.rabbitmq.com/tutorials/tutorial-six-python.html

RPC in Python with RabbitMQ



RPC Python - Server

```
ch.basic_publish(exchange='',
                 routing_key=props.reply_to,
                 properties=pika.BasicProperties(
                     correlation_id = props.correlation_id),
                 body=str(response))

ch.basic_ack(delivery_tag=method.delivery_tag)



channel.basic_qos(prefetch_count=1)
channel.basic_consume(queue=EXCHANGE_NAME,
                      on_message_callback=on_request,
                      auto_ack=False)
```

RPC Python - Client

Creation of a class for the RPC stub

```
def __init__(self):
  [self.connection, self.channel] =
     connect_rabbit_noqueue()

  result = self.channel.queue_declare(queue='',
                       exclusive=True)
  self.callback_queue = result.method.queue

  self.channel.basic_consume(
    queue=self.callback_queue,
    on_message_callback=self.on_response,
    auto_ack=True)

def on_response(self, ch, method, props, body):
  if self.corr_id == props.correlation_id:
    self.response = body
```

```
def call(self, n):  # method exposed for RPC
   self.response = None
   self.corr_id = str(uuid.uuid4())
   self.channel.basic_publish(
      exchange='',
      routing_key=EXCHANGE_NAME,
      properties=pika.BasicProperties(
         reply_to=self.callback_queue,
         correlation_id=self.corr_id,
      ),
      body=str(n))
   while self.response is None:
      self.connection.process_data_events()
   return int(self.response)
```