UNIVERSITAT ROVIRA i VIRGILI

Lab 1: Presentation & XML-RPC

Distributed Systems

Assignments

- Assignments will combine guided and autonomous work.

Guided part

(in situ, lab sessions)

- Working on concepts and utilities.
- Practical examples.

tl;dr

[too long; didn't read]

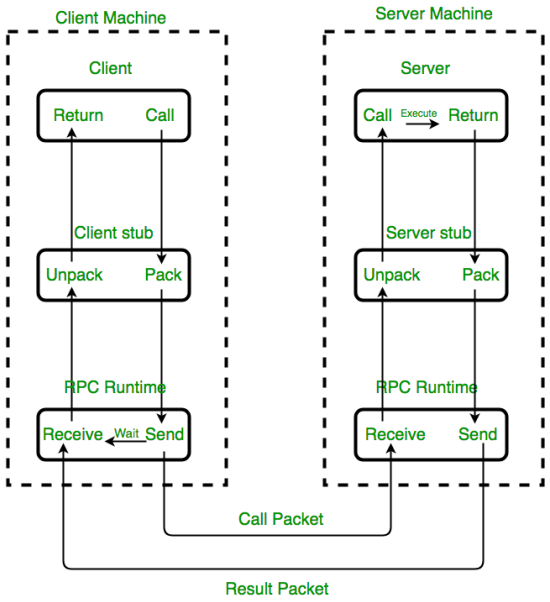
Unguided part

READ THE FUCKING MANUAL

Remote Procedure Calls & XML-RPC

Remote Procedure Calls (RPC)

- Based on the **client-server architecture**.
- **Direct** communication protocol.
- Allows calling processes in the server machine **remotely** (from the client machine).
- The client is agnostic of the function implementation in the server side (**transparent** calls).



Implementation of RPC mechanism

Source: https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system/

XML-RPC

- XML-RPC is an implementation of the RPC protocol.
- Based on **XML** and **HTTP(S)**:
  - **XML** for **data serialisation** and **HTTP** for **data transfer**.
- An XML-RPC **request** contains:
  - **Method name**.
  - **Method parameters**: primitive data types, structs and arrays.
- An XML-RPC server response contains a **response** or a **fault**.

XML-RPC

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
   <methodName>sample.sum</methodName>
   <params>
      <param>
         <value><int>17</int></value>
      </param>

      <param>
         <value><int>13</int></value>
      </param>
   </params>
</methodCall>
```

XML-RPC **request**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
   <params>
      <param>
         <value><int>30</int></value>
      </param>
   </params>
</methodResponse>
```

XML-RPC **successful response**

```
<?xml version="1.0"?>
<methodResponse>
   <fault>
      <value><string>No such method!</string></value>
   </fault>
</methodResponse>
```

XML-RPC **fault response**

XML-RPC sample

```python
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler

# Restrict to a particular path.
class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)

# Create server
with SimpleXMLRPCServer(('localhost', 8000),
                        requestHandler=RequestHandler) as server:
    server.register_introspection_functions()

    # Register pow() function; this will use the value of
    # pow.__name__ as the name, which is just 'pow'.
    server.register_function(pow)

    # Register a function under a different name
    def adder_function(x, y):
        return x + y
    server.register_function(adder_function, 'add')

    # Register an instance; all the methods of the instance are
    # published as XML-RPC methods (in this case, just 'mul').
    class MyFuncs:
        def mul(self, x, y):
            return x * y

    server.register_instance(MyFuncs())

    # Run the server's main loop
    server.serve_forever()
```

```python
import xmlrpc.client

s = xmlrpc.client.ServerProxy('http://localhost:8000')
print(s.pow(2,3))  # Returns 2**3 = 8
print(s.add(2,3))  # Returns 5
print(s.mul(5,2))  # Returns 5*2 = 10

# Print list of available methods
print(s.system.listMethods())
```

Lab 1 assignment

Lab 1 assignment: InsultServer

- **Client / Server architecture**: insulting client, insulting server.
- Communication is implemented via **RPCs**:
  - **add_insult(string insult)**: adds an insult to the server's insults list.
  - **get_insults()**: the server returns the insults list.
  - **insult_me()**: the server returns a random insult from the insults list.

Lab 1 assignment

- **Deploy** your implementation:
  - Deploy the client(s) and the server on the lab's computers
  - Communicate them using their IP addresses.
  - **Warning:** use the lab's computers; your computers may not be able to communicate!

Lab 2 assignment: Observer

- Implement the observer pattern in Python with XMLRPC.
  - The goal is to create a server that can register subscribers (observers) to the Insult server. Every time an insult is added to the server, it will notify all subscribers of this new insult.
  - You must provide the URL of a subscriber server with the notify method to the InsulterService addSubscriber call.
  - Check with three subscribers and one client sending new insults every 3 seconds to the InsultServer.

Resources

- **Tools:** Python 3.8+.
- **Resources:**
  - The Python Tutorial
  - Python XML-RPC documentation
  - ChatGPT/DeepSeek

UNIVERSITAT
ROVIRA i VIRGILI