Distributed Architectures and Programming
ISEP - October 2024
*by Quentin LAURENT and Theophile WEMAERE*

**Note**

The source code of this lab can also be found on Github :
https://github.com/Theophile-Wemaere/ArchProgDistr

## Part 1. Remote Procedure Call (RPC)

For this part, we have created additional functions that allow to subtract, multiply and divide two numbers. As with the `add` function, we register all these new functions on the XMLRPC server so that they can be called by a remote client.

**Code for the server :**

```python
from xmlrpc.server import SimpleXMLRPCServer

# Define the functions that can be called remotely
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    return x / y

# Create the server
server = SimpleXMLRPCServer(("localhost", 8000))
print("Listening on port 8000...")

# Register the functions
server.register_function(add, "add")
server.register_function(subtract, "subtract")
server.register_function(multiply, "multiply")
server.register_function(divide, "divide")

# Run the server
try:
    server.serve_forever()
except KeyboardInterrupt:
    print("\nServer stopped.")
```

On the client side, we simply call the `add`, `subtract` and `multiply` functions. Regarding the `divide` function, we surrounded it in a `try/catch` block because it might raise exceptions (when dividing numbers by zero). In such a case, we simply display the error message sent by the server (which is a serialized version of the exception) instead of the result of the division.

**Code for the client :**

```python
import xmlrpc.client
from xmlrpc.client import Fault

# Create a proxy for the server
server = xmlrpc.client.ServerProxy("http://localhost:8000/")

# Call the remote functions
print("Addition of 5 and 3:", server.add(5, 3))
print("Subtraction of 8 and 2:", server.subtract(8, 2))
print("Multiplication of 3 and 8:", server.multiply(3, 8))

# For the division function, we surround the function call in a try/catch
# so that we can catch any eventual error raised during a division
# (i.e. a division by zero)
try:
    print("Division of 5 and 0:", server.divide(5, 0))
except Fault as fault:
    print(f"Division of 5 and 0: {fault.faultString}")
```

## Part 3. Matrix multiplication app

For this part, we just need to create a new function that multiplies two matrices together:

```python
def multiply_matrix(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:
    print(f"[DEBUG] Resulting matrix will be of size {len(m2[0])}x{len(m1)}")
    # We use a double list comprehension to create an empty matrix of size M2_C x M1_L where:
    # M2_C is the number of columns of the M2 matrix
    # M1_L is the number of lines of the M1 matrix
    res = [[0 for _ in range(len(m2[0]))] for _ in range(len(m1))]

    for i in range(len(m1)):
        for j in range(len(m2[0])):
            for k in range(len(m2)):
                res[i][j] += m1[i][k] * m2[k][j]

    return res
```

We then just need to register the function so that it can be called remotely:

```python
server.register_function(multiply_matrix, "multiply_matrix")
```

The final code, with all the functions implemented, looks like this:

**Code for the server :**

```python
from xmlrpc.server import SimpleXMLRPCServer

# Define the functions that can be called remotely
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    return x / y

def multiply_matrix(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:
    print(f"[DEBUG] Resulting matrix will be of size {len(m2[0])}x{len(m1)}")
    # We use a double list comprehension to create an empty matrix of size M2_C x M1_L where:
    # M2_C is the number of columns of the M2 matrix
    # M!_L is the number of lines of the M1 matrix
    res = [[0 for _ in range(len(m2[0]))] for _ in range(len(m1))]

    for i in range(len(m1)):
        for j in range(len(m2[0])):
            for k in range(len(m2)):
                res[i][j] += m1[i][k] * m2[k][j]

    return res

# Create the server
server = SimpleXMLRPCServer(("localhost", 8000))
print("Listening on port 8000...")

# Register the functions
server.register_function(add, "add")
server.register_function(subtract, "subtract")
server.register_function(multiply, "multiply")
server.register_function(divide, "divide")
server.register_function(multiply_matrix, "multiply_matrix")

# Run the server
try:
    server.serve_forever()
except KeyboardInterrupt:
    print("\nServer stopped.")
```

**Code for the client :**

```python
import xmlrpc.client
from xmlrpc.client import Fault

# Create a proxy for the server
server = xmlrpc.client.ServerProxy("http://localhost:8000/")

# Call the remote function
print("Addition of 5 and 3:", server.add(5, 3))
print("Subtraction of 8 and 2:", server.subtract(8, 2))
print("Multiplication of 3 and 8:", server.multiply(3, 8))

# For the division function, we surround the function call in a try/catch
# so that we can catch any eventual error raised during a division
# (i.e. a division by zero)
try:
    print("Division of 5 and 0:", server.divide(5, 0))
except Fault as fault:
    print(f"Division of 5 and 0: {fault.faultString}")

# Matrix multiplication
m1 = [[12, 7, 3],
      [4, 5, 6],
      [7, 8, 9]]

m2 = [[5, 8, 1, 2],
      [6, 7, 3, 0],
      [4, 5, 9 ,1]]

res = server.multiply_matrix(m1, m2)
print("Matrix multiplication of m1 and m2:")
for row in res:
    print(row)
```