Practical Work 2: RPC File Transfer

Nguyen Thanh Dat
Student ID: 23BI14091

1 Introduction

The goal of this practical work is to upgrade the previous TCP file transfer system to use **Remote Procedure Call (RPC)**. Unlike the previous implementation, which handled raw socket streams and data chunking manually, this system abstracts the networking layer, allowing the client to invoke a function on the server as if it were local.

2 RPC Service Design

To implement file transfer, I designed a specific RPC function that is exposed by the server. Instead of defining a packet protocol (headers, delimiters, etc.), I defined a service interface.

2.1 Service Definition

The server registers a single function:

- **Function Name:** `save_file(filename, binary_data)`
- **Input:** A string representing the name of the file and a binary object containing the file's raw bytes.
- **Output:** Returns `True` if the write operation was successful.

2.2 Interaction Diagram

The interaction is synchronous. The client reads the file, wraps it, and calls the remote method.

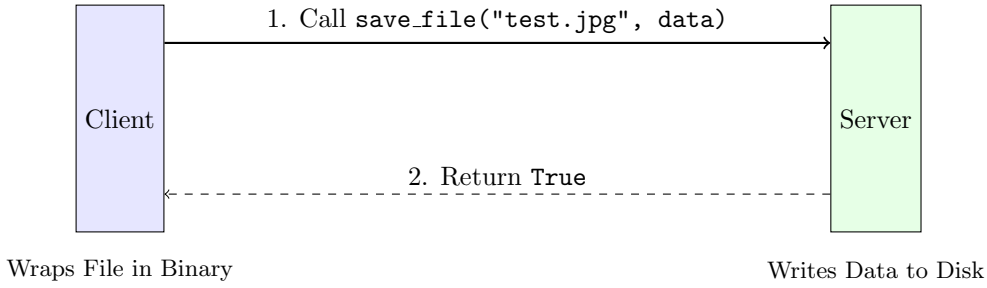

Figure 1: Sequence Diagram of the RPC Call

3 System Organization

The system leverages Python's built-in `xmlrpc` library.

- **Server Side:** Uses `SimpleXMLRPCServer` to listen on a specific port and register the `save_file` function.
- **Client Side:** Uses `ServerProxy` to create a local stub that forwards calls to the server over HTTP/XML.

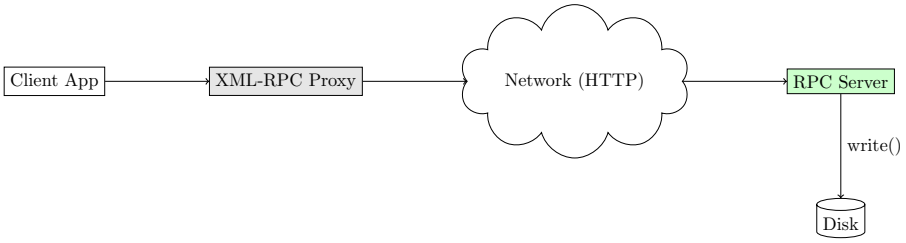

Figure 2: System Architecture

4 Implementation

4.1 Server Implementation

The server defines the logic to unwrap the binary data and write them to a file.

```
import os
from xmlrpc.server import SimpleXMLRPCServer

def save_file(filename, file_data):
    print(f"Receiving file: {filename}...")
    try:
```

```
        # Access .data to get raw bytes from XMLRPC Binary
            object
        with open("received_" + filename, "wb") as handle:
            handle.write(file_data.data)
        return True
    except Exception as e:
        return False

# Registration
server = SimpleXMLRPCServer(('127.0.0.1', 65432))
server.register_function(save_file, "save_file")
server.serve_forever()
```

4.2 Client Implementation

The client manages the reading of files and the calls to remote procedures.

```
import xmlrpc.client

proxy = xmlrpc.client.ServerProxy('http://127.0.0.1:65432')

with open("test.jpg", "rb") as handle:
    # Wrap raw bytes for XML-RPC transfer
    binary_data = xmlrpc.client.Binary(handle.read())

proxy.save_file("test.jpg", binary_data)
```

5 Verification

The following screenshot illustrates the successful execution of the client script and the server's acknowledgement.

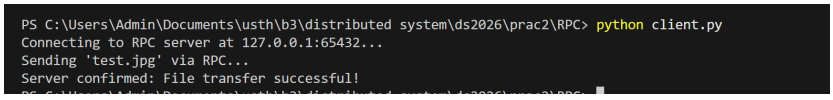

Figure 3: Successful File Transfer Execution