Practical Work 2: RPC File Transfer

Nguyen Thi Minh Nguyet
ID: 22BA13241
Class: CS

December 3, 2025

1 Introduction

Following the previous work on TCP sockets, this practical work aims to upgrade the file transfer system using **Remote Procedure Call (RPC)**. Unlike the socket-based approach where we manually handle data streams, RPC allows the client to execute a procedure (function) located on the server as if it were a local function call.

We utilize Python's `xmlrpc` library to implement this system.

2 RPC Service Design

In this design, the Server exposes a specific method called `upload_file`. The Client does not send raw bytes directly into a stream but instead invokes this method, passing the filename and the file content as arguments.

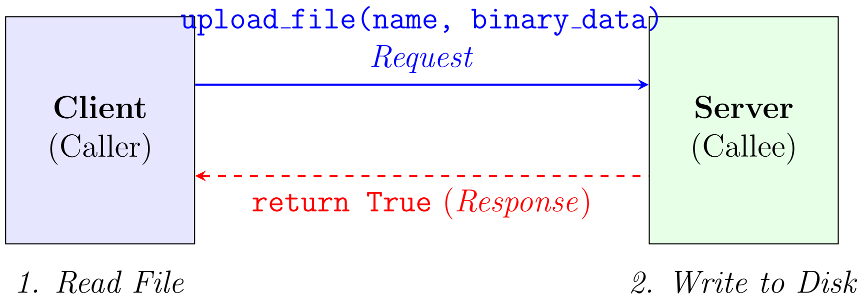

Figure 1: RPC Service Design (Function Call Model)

3 System Organization

The system relies on the RPC Middleware (Marshalling/Unmarshalling). The client application communicates with a "Stub" (Proxy), which serializes the arguments and sends them over the network. The server uses a "Skeleton" (Dispatcher) to interpret the message and execute the actual Python code.

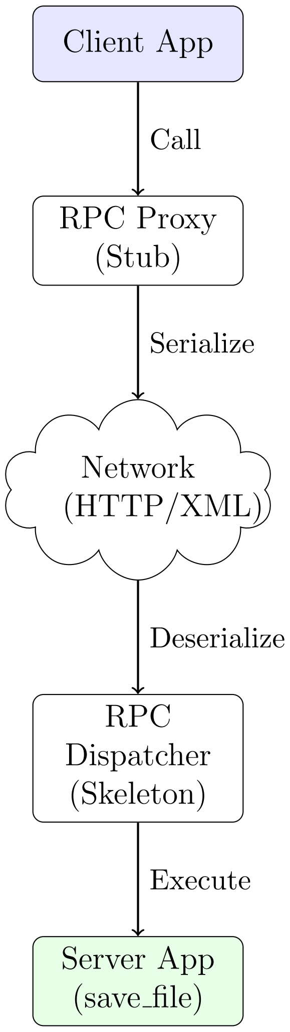

Figure 2: System Organization with RPC Middleware

4 Implementation Details

4.1 Server Implementation

The server registers a function and enters an infinite loop to handle requests.

```python
from xmlrpc.server import SimpleXMLRPCServer

def save_file(filename, binary_data):
    # binary_data is an xmlrpc.client.Binary object
    with open("uploaded_" + filename, "wb") as f:
        f.write(binary_data.data)
    return True

server = SimpleXMLRPCServer(("localhost", 8000))
server.register_function(save_file, "upload_file")
server.serve_forever()
```

Listing 1: Server Code using xmlrpc

4.2 Client Implementation

The client connects to the server proxy and calls the exposed function.

```python
import xmlrpc.client

proxy = xmlrpc.client.ServerProxy("http://localhost:8000/")

with open("image.jpg", "rb") as f:
    # Wrap file content in Binary wrapper
    data = xmlrpc.client.Binary(f.read())

    # Call the remote function
    proxy.upload_file("image.jpg", data)
```

Listing 2: Client Code using xmlrpc

5 Roles and Responsibilities

- **Client (The Caller):**
  - Reads the file from the local file system.
  - Wraps the binary data into an RPC-compatible format.
  - Initiates the remote procedure call `upload_file()`.
- **Server (The Callee):**
  - Exposes the service on a specific port (8000).
  - Listens for incoming XML-RPC requests.
  - Unpacks the received arguments and executes the file writing logic.
  - Returns a success status to the client.