Distributed Systems

Practical Work 2: RPC File Transfer

Name: Dang Trung Nguyen
Student ID: 22BA13239

University of Science and Technology of Hanoi (USTH)

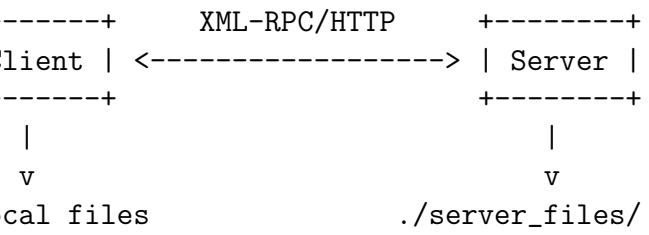
December 3, 2025

1 Introduction

This report presents a simple file transfer system using XML-RPC in Python.

2 RPC Service Design

2.1 Architecture



2.2 RPC Methods

- `upload_file(filename, binary_data)` - Upload file to server
- `download_file(filename)` - Download file from server
- `list_files()` - List all files on server

3 System Organization

```
RPC/
|-- rpc_server.py    # Server code
|-- rpc_client.py    # Client code
|-- server_files/    # Storage folder
```

4 Implementation

4.1 Server Code

```
from xmlrpc.server import SimpleXMLRPCServer
import os

STORAGE = "./server_files"

def upload_file(filename, binary_data):
    filepath = os.path.join(STORAGE, filename)
    with open(filepath, "wb") as handle:
        handle.write(binary_data.data)
    return True

def download_file(filename):
    filepath = os.path.join(STORAGE, filename)
    with open(filepath, "rb") as handle:
        from xmlrpc.client import Binary
        return Binary(handle.read())

def list_files():
    return os.listdir(STORAGE)

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

4.2 Client Code

```
import xmlrpc.client
import os

SERVER_URL = "http://localhost:8000/"

def send_file(filename):
    proxy = xmlrpc.client.ServerProxy(SERVER_URL)

    with open(filename, "rb") as handle:
        binary_data = xmlrpc.client.Binary(handle.read())
        result = proxy.upload_file(filename, binary_data)

    if result:
        print("File␣sent␣successfully!")

def receive_file(filename):
    proxy = xmlrpc.client.ServerProxy(SERVER_URL)
```

```
    result = proxy.download_file(filename)

    with open(filename, "wb") as handle:
        handle.write(result.data)
    print("File␣received!")
```

5 How It Works

1. Client reads file as binary data
2. Client wraps data with `xmlrpc.client.Binary`
3. Client calls RPC method (looks like local function call)
4. XML-RPC converts to XML and sends via HTTP
5. Server receives and saves file
6. Server returns result to client

6 Testing

```
# Terminal 1: Start server
python rpc_server.py

# Terminal 2: Run client
python rpc_client.py
# Choose: 1 to send, 2 to receive, 3 to list
```

7 Conclusion

RPC makes distributed programming simple. Using `xmlrpc.client.Binary` allows easy binary file transfer without manual encoding.